A game theoretic model of the behavioural gaming that takes place at the EMS - ED interface*

Michalis Panayides*, Vince Knight, Paul Harper

School of Mathematics, Cardiff University, Cardiff CF24 4AX, United Kingdom

Abstract

This research describes the development and application of a 3-player game theoretic model between two queueing systems and a service that distributes individuals to them. The resultant model is used to explore dynamics between all players. The first aspect of this work is the development of a queueing system with two consecutive waiting spaces where the strategic managerial behaviour corresponds to how individuals use these waiting spaces. Two modelling techniques are deployed: discrete event simulation and Markov chains. The state probabilities of the Markov chain system are used to extract the performance measures of the queueing model (e.g. mean time in each waiting room, mean number of individuals in each room, etc.). A 3-player game theoretic model is subsequently proposed between the two queueing systems and the service that distributes individuals to them. In particular this can be viewed as a 2-player normal-form game where the utilities are determined by a third player with its own strategies and objectives. A backwards induction technique is used to get the utilities of the normal-form game between the two queueing systems. This particular system has many applications, including those in healthcare where it captures the emergent behaviour between the Emergency Medical Service (EMS) and the Emergency Department (ED). The impact of time-target measures on patient well-being is explored in this paper.

Keywords: OR in health services, Game theory, Queueing theory

1. Introduction

Emergency departments (EDs) are under increasing pressure to meet patient waiting time targets and satisfy regulations [21]. It is widely reported (e.g. [24, 31, 32]) that ED congestion severely impacts not only patients in the ED but also Emergency Medical Services (EMS). A

*This document is the result of the research project funded by The Healthcare Improvement Studies (THIS) institute.

*Corresponding author

Email addresses: panayidesM@cardiff.ac.uk (Michalis Panayides), knightva@cardiff.ac.uk (Vince Knight), harper@cardiff.ac.uk (Paul Harper)

major concern for ambulances is that they are held waiting parked outside the ED to offload (dispatch) their patient when the ED is particularly busy [12]. Since the patient waiting time in ED is measured from the time they enter the ED itself, there is no incentive, should the patient be stable in the ambulance, to offload them from EMS to ED services. As a result, ambulance blocking not only impacts on patients waiting for ED service, but has a major knock-on effect to delaying the ability of ambulances to respond to new EMS calls, thus placing lives at risk [16].

There are numerous news articles that focus on the complexity that arises when ambulances stay blocked outside of the hospital for a long amount of time [3, 14]. Some news reports comment on the long idle time of ambulances when they are not in use [45] and there are several reports of examples where this became an issue for new patients [34]. A particular article looked into this problem from the point of view of an ambulance paramedic:

> "He knows if a patient arrives more urgently in need of help, his will be pushed back" [13]
>
> "We used to bring poorly patients in, and we were out on the road again in 15 minutes." [13]

This paper aims to describe the EMS-ED interface using a game theoretic model informed by an underlying queueing model. The model describes the situation where an ambulance service would have to distribute its patients between two EDs. The two EDs can be thought of as two queueing systems and the EMS as a distributor that distributes patients to them, aiming to minimise some performance measure. The patients that are distributed by the EMS arrive at the hospital via an ambulance and are then either offloaded at the ED or stay blocked outside in the ambulance. Whether or not the ambulance and its patient stay blocked is determined by the threshold that the given ED chooses to play. High threshold indicates that the ED accepts ambulance patients even if it is relatively full, while low threshold means that the ED blocks ambulances more frequently. In the United Kingdom the National Heath Services (NHS) sets some regulations on ED performance. One of these regulations is that 95% of patients that arrive at the ED should be admitted, transferred or discharged within four hours. This is where gaming behaviour might be observed between the EDs and the EMS. An assumption of this work is that some managerial decision making is involved in choosing when to start blocking ambulances. This is similar to [17].

The major contributions of this paper are:

- A queueing model with 2 consecutive waiting spaces where one would serve as a parking space for the ambulances
- Analytic performance measure formulas for the queueing model
- A 3-player game theoretic model between the EMS and two EDs
- Numerical experiments showing emergent behaviour of gaming between EDs and the EMS.

Specifically, our focus is on the construction of a 3-player game theoretic model between two queueing systems and a service that distributes individuals to them. The resultant model is then used to explore the emergent dynamics between the three players. This study explores two new concepts: getting performance measures for a new queueing theoretic model with a parking space and a service centre and using a learning algorithm to model the emergence of behaviour. The developed theoretical model is illustrated through the application to a healthcare system of two EDs and the EMS, exploring the inefficiencies that emerge and ways to apply some incentive mechanisms to improve them. The EDs are modelled as two queueing systems each with a tandem buffer and a service centre. The performance measures are then used as the utilities of the game. The novelty of the queueing model here is a contribution not only the game theoretic literature but also to the queueing theoretic literature. To the authors knowledge, no such model of a tandem queueing model with a pair of parameters for the buffer has been considered.

This paper consists of two main sections. Section 3 presents a novel queueing model for a hospital with two types of patients and two waiting zones. A detailed description of how to acquire the performance measure formulas of such queueing system is given. Section 4 gives an overview of the game theoretic model and several theoretic results pertaining to the performance measures of this model which are used to build the utilities of the game.

2. Literature review

A number of papers have been published that touch upon the use of queueing models together with game theoretic concepts. In [9] the authors study a simultaneous price competition between two firms that are modelled as two distinct queueing systems with a fixed capacity and a combined arrival rate. They calculate the Nash equilibrium both for identical and heterogeneous firms and show that for the former a pure Nash equilibrium always exist and for the latter a unique equilibrium exists where only one firm operates. The authors have also extended their model in [10] by allowing the players (firms) to choose capacities. A main result from this paper was that when both firms operate independently as a monopoly, the equilibria are socially optimal, but this is not the case when the firms operate together. Another extension of [9] was introduced in [11] where a long-run version of the competition was considered that also had capacity as a decision variable. An additional paper that focuses on competition is [18] where the authors created a competition between two sellers where seller 1 supplies a product instantly and seller 2 is modelled as a make-to-order M/M/1 queue. The game that is played requires the two sellers to make a choice on the price of the product and then seller 2 to set a capacity that guarantees a maximum expected delay. In this paper, while giving some consideration to equilibrium behaviour, similar to the work of [9, 10], emergent behaviour is more precisely addressed by considering learning algorithms like asymmetric replicator dynamics [20].

In the above models, the players are attempting to increase their share of individuals choosing to queue. In public healthcare type settings, this is not necessarily the case. Rational usage

of public services will not necessarily lead to a socially optimal outcome. Rather, the overall service needs to be considered as players aim to minimise their experienced congestion. In [38] a healthcare application was studied where patients could choose between two hospitals, where a utility function is derived that is based on patients' perceived quality of life. In [26] the authors place the individuals' choices between different public services within the formulation of routing games and measure inefficiencies using a concept known as the price of anarchy (PoA) [27]. They show that the price of anarchy increases with worth of service and that is low for systems with insufficient capacities. In [25] a normal form game is built that is informed by a two-dimensional Markov chain in order to model interactions between critical care units. In [17] the authors study the network effect of ambulance diversion by proposing a non-cooperative game between two EDs that are modelled as a queueing network. Each ED's objective is to minimise its own waiting time and chooses a diversion threshold based on the patients it has. In equilibrium both EDs choose to divert ambulances in order to avoid getting arrivals from the other ED. In this paper this concept is extended by allowing the ambulance service to decide how to distribute its patients among the two EDs. The players of the game are both the hospitals and the customers of the hospitals, as opposed to the previous models which are one or the other. Thus, the novelty of this paper in a way is combining both these aspects.

Another specific part of our research, as described later in the paper, is the construction of a queueing system with a tandem buffer and a single service centre. There are several examples from literature that touch upon queueing models with tandem queues. In [15] the authors explore threshold joining strategies in a Markov model that has two tandem queues. Another example is the one described in [8] where they investigated a network of multiple tandem queues where customers decide which queue to attend before joining. Similarly, in [5] the authors examine a network of N tandem M/M/1 queues and with multi-type customers. The customers in this paper react to a price p by picking demand rates that maximise utility. In [46] a profit maximisation problem is studied that has two servers; an M/M/1 queue and a parking service providing complementary service while the customer is in the first service. The providers gain a reward when customers complete both services and no reward when they finish one of them. One of the main conclusions of this study is that by increasing the general demand both providers lower their prices to compensate for the increase in wait. The problem was later extended by [42] where they considered arrivals of batches that can share the parking service. Finally, [2] examines a tandem network of two M/M/1 queues that are ran by two different profit-maximising service providers. The network receives three types of customers; those requiring both services, customers requiring the first service and customers requiring the second service. The authors showed that optimal prices also maximise social utility and that removing two types of customers that don't need both services leads to higher profit and lower demand rate. In this paper the concepts described in [15, 8, 5] are extended by introducing a threshold parameter that determines when individuals can progress from one queue to the other.

3. A queueing model for the ED-EMS interface

In this section, a more in-depth explanation of the queueing model shown in figure 1 will be given. This is a queueing model that consists of two waiting zones; the parking space and the hospital waiting space.

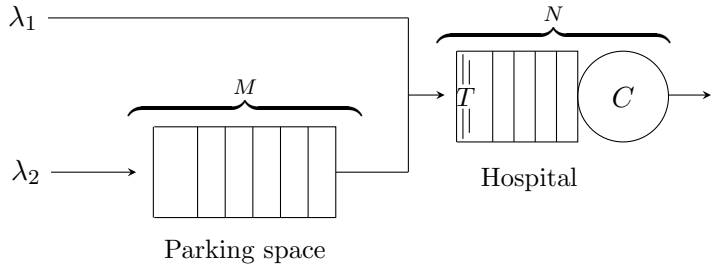

Figure 1: A diagrammatic representation of the queueing model. The threshold T only applies to type 2 individuals. If the number of individuals in the hospital is T, only individuals of type 1 are accepted (at a rate λ_1) and individuals of type 2 (arriving at a rate λ_2) are blocked in the parking space.

The model consists of two types of individuals; type 1 and type 2. Type 2 individuals are patients arriving in ambulances who can be blocked (usually patients that are deemed not to be critical) and type 1 individuals are individuals arriving from other sources (e.g walk-in patients, urgent patients from either walk-ins or ambulances). Type 1 individuals arrive instantly at the hospital's waiting space and wait to receive their service. Type 2 individuals arrive at the parking space and wait there until they are allowed to move to the hospital. They are allowed to proceed only when the number of patients in the hospital is less than the pre-determined threshold T. When the number of individuals is equal to or exceeds this threshold, all type 2 patients that arrive will stay *blocked* in the parking space until the number of patients in the hospital falls below T. This is shown diagrammatically in Figure 1. The parameters of the described queueing model are:

- λ_i: The arrival rate of individuals of type $i \in \{1, 2\}$
- μ: The service rate for individuals receiving service
- C: The number of servers (either healthcare professionals or available beds in the ED)
- T: The threshold at which type 2 individuals are blocked

Under the assumption that all rates (arrival and service) are Markovian the queueing system corresponds to a Markov chain [22]. The states of the Markov chain are denoted by (u, v) where:

- u is the number of individuals blocked in the parking space
- v is the number of individuals waiting or being served in the hospital

We denote the state space of the Markov chain as $S = S(T)$ which can be written as the disjoint union (1).

$$\begin{aligned} S(T) =& S_1(T) \cup S_2(T) \text{ where:} \\ S_1(T) =& \left\{(0, v) \in \mathbb{N}_0^2 \mid v < T\right\} \\ S_2(T) =& \{(u, v) \in \mathbb{N}_0^2 \mid v \geq T\} \end{aligned} \tag{1}$$

The generator matrix Q of the Markov chain consists of the rates between the numerous states of the model. Every entry $Q_{ij} = Q_{(u_i,v_i),(u_j,v_j)}$ represents the rate from state $i = (u_i, v_i)$ to state $j = (u_j, v_j)$ for all $(u_i, v_i), (u_j, v_j) \in S$. The entries of Q can be calculated using the state-mapping function described in (2):

$$Q_{ij} = \begin{cases} \Lambda, & \textbf{if } (u_i, v_i) - (u_j, v_j) = (0, -1) \textbf{ and } v_i < \mathrm{t} \\ \lambda_1, & \textbf{if } (u_i, v_i) - (u_j, v_j) = (0, -1) \textbf{ and } v_i \geq \mathrm{t} \\ \lambda_2, & \textbf{if } (u_i, v_i) - (u_j, v_j) = (-1, 0) \\ v_i\mu, & \textbf{if } (u_i, v_i) - (u_j, v_j) = (0, 1) \textbf{ and } v_i \leq C \textbf{ or} \\ & \quad (u_i, v_i) - (u_j, v_j) = (1, 0) \textbf{ and } v_i = T \leq C \\ C\mu, & \textbf{if } (u_i, v_i) - (u_j, v_j) = (0, 1) \textbf{ and } v_i > C \textbf{ or} \\ & \quad (u_i, v_i) - (u_j, v_j) = (1, 0) \textbf{ and } v_i = T > C \\ -\sum_{j=1}^{|Q|} Q_{ij} & \textbf{if } i = j \\ 0, & \textbf{otherwise} \end{cases} \tag{2}$$

Note that Λ here denotes the overall arrival rate in the model by both types of individuals (i.e. $\Lambda = \lambda_1 + \lambda_2$). A visualisation of how the transition rates relate to the states of the model can be seen in the general Markov chain model shown in Figure 2.

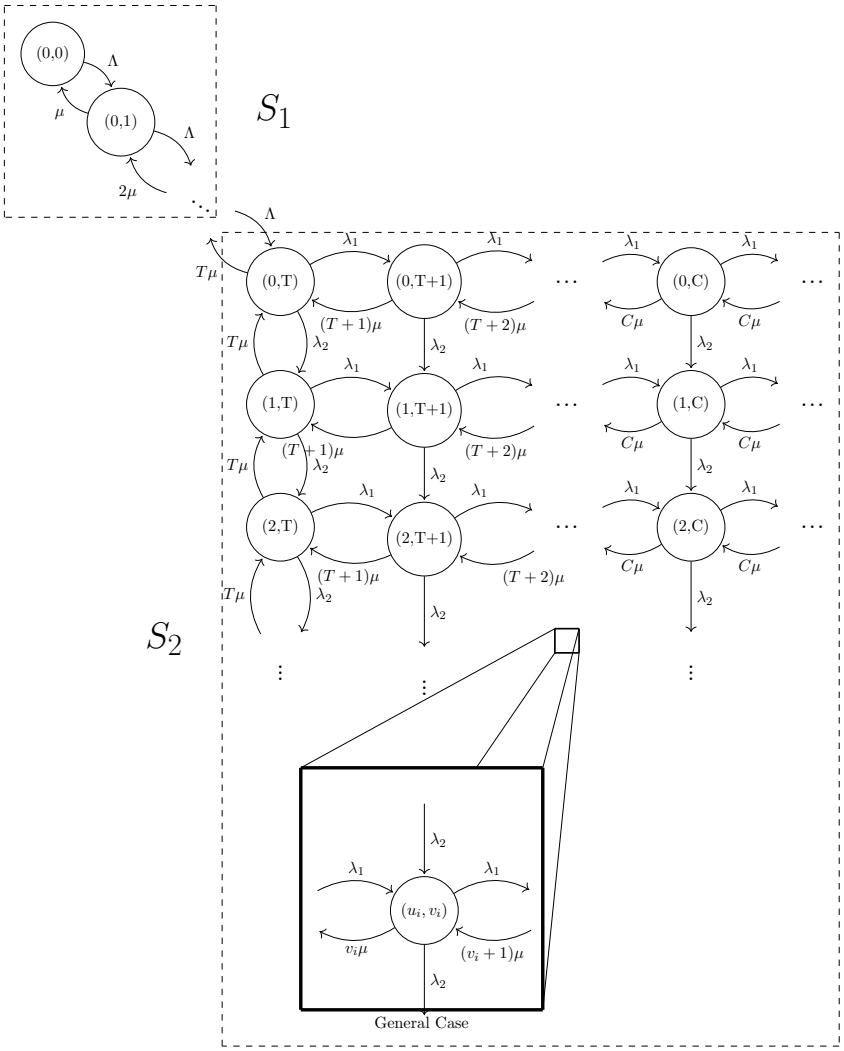

Figure 2: General case of the Markov chain model

In order to consider this model numerically an adjustment needs to be made. The problem defined above assumes no upper boundary to the number of patients that can wait in the hospital or to the ones that are blocked in the parking space. Therefore, a different state space $\tilde{S}$ is constructed where $\tilde{S} \subseteq S$ and there is a maximum allowed number of patients N that can be in the hospital and a maximum allowed number of ambulances M that can be blocked in the parking space:

$$\tilde{S} = \{(u, v) \in S \,|u \leq M, v \leq N\} \tag{3}$$

The generator matrix Q defined in (2) can be used to get the probability vector π. The vector π is commonly used to study stochastic systems and it's main purpose is to keep track of the probability of being at any given state of the system. π_i is the steady state probability of being in

state $(u_i, v_i) \in \tilde{S}$ which is the i^{th} state of $\tilde{S}$ for some ordering of $\tilde{S}$. The term *steady state* refers to the instance of the vector π where the probabilities of being at any state become stable over time. Thus, by considering the steady state vector π the relationship between it and Q is given by:

$$\frac{d\pi}{dt} = \pi Q = 0$$

Using vector π there are numerous performance measures of the model that can be calculated. The following equations utilise π to get performance measures for the average number of patients at the different nodes of the queueing model:

- Average number of patients in the entire system:

$$L = \sum_{i=1}^{|\pi|} \pi_i (u_i + v_i)$$

- Average number of patients in the hospital:

$$L_H = \sum_{i=1}^{|\pi|} \pi_i v_i$$

- Average number of patients/ambulances in the parking space:

$$L_A = \sum_{i=1}^{|\pi|} \pi_i u_i$$

Consequently, there are some additional performance measures of interest that are not as straightforward to calculate. Such performance measures are the mean waiting time in the system (for both type 1 and type 2 individuals), the mean time blocked in the parking space (only valid for type 2 individuals) and the proportion of individuals in the hospital whose waiting time falls within a predefined time target (for both types).

3.1. Waiting time

Waiting time is the amount of time that patients wait in the hospital's waiting space before they can receive their service. For a given set of parameters there are three different performance measures around the mean waiting time that can be calculated. The mean waiting time of type 1 individuals:

$$W^{(1)} = \frac{\sum_{\substack{(u,v) \in S_A^{(1)} \\ v \geq C}} \frac{1}{C\mu} \times (v - C + 1) \times \pi(u, v)}{\sum_{(u,v) \in S_A^{(1)}} \pi(u, v)} \tag{4}$$

The mean waiting time of type 2 individuals:

$$W^{(2)} = \frac{\sum_{\substack{(u,v)\in S_A^{(2)} \\ min(v,T)\geq C}} \frac{1}{C\mu} \times (\min(v+1,T) - C) \times \pi(u,v)}{\sum_{(u,v)\in S_A^{(2)}} \pi(u,v)} \tag{5}$$

The overall mean waiting time:

$$W = \frac{\lambda_1 P_{L_1'}}{\lambda_2 P_{L_2'} + \lambda_1 P_{L_1'}} W^{(1)} + \frac{\lambda_2 P_{L_2'}}{\lambda_2 P_{L_2'} + \lambda_1 P_{L_1'}} W^{(2)} \tag{6}$$

Here $S_A^{(1)}$ and $S_A^{(2)}$ are the set of accepting states for type 1 and type 2 individuals. These are the set of states that the model is able to accept a specific type of individuals.

$$S_A^{(1)} = \{(u,v) \in S \mid v < N\} \tag{7}$$

$$S_A^{(2)} = \begin{cases} \{(u,v) \in S \mid u < M\}, & \textbf{if } T \leq N \\ \{(u,v) \in S \mid v < N\}, & \textbf{otherwise} \end{cases} \tag{8}$$

Equation 6 makes use of the proportion of type 1 and type 2 individuals that are not lost to the system. These probabilities are given by $P_{L_1'}$ and $P_{L_2'}$ where:

$$P_{L_1'} = \sum_{(u,v)\in S_A^{(1)}} \pi(u,v) \qquad\qquad P_{L_2'} = \sum_{(u,v)\in S_A^{(2)}} \pi(u,v) \tag{9}$$

Appendix B gives more details on the recursive formula that equations (4), (5) and (6) originate from.

Figure 3 shows a comparison between the calculated mean waiting time using Markov chains and the simulated waiting time using discrete event simulation over a range of values of λ_2 (details of the discrete event simulation model are given in appendix A). The figure is used to demonstrate the accuracy of the waiting time formula of the constructed queueing model as well as the effect of truncating the model. The simulation was ran 100 times and the recorded mean waiting time at each iteration is used to populate the violin plots. In detail, figure 3 shows the calculated mean waiting time using the Markov chain, using a truncated simulation and using a simulation with infinite capacity (without the artificial parameters N and M). Each plot corresponds to different values of N and M and is run over different values of λ_2. The untruncated simulation values are the same at all three graphs since the effect of truncation does not apply to it. The waiting times generated by the truncated simulation match the ones generated by the Markov chains model. Note that this comparison includes both type 1 and type 2 individuals. A separate comparison of only type 1 and only type 2 individuals can be found in appendix E.

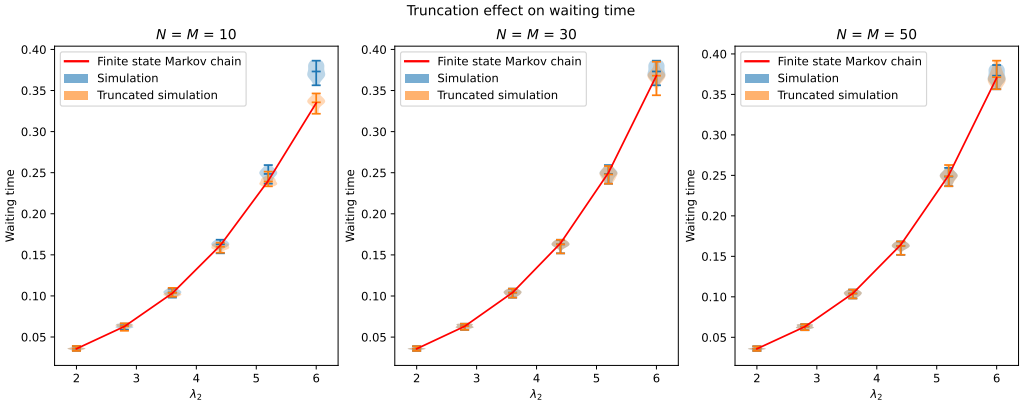

Figure 3: Comparison of mean waiting time between values obtained from the Markov chain formula, values obtained from the truncated simulation and values obtained from the untruncated simulation.

3.2. Blocking time

Blocking time is the amount of time that type 2 patients wait in the parking space before they are allowed to proceed to the hospital. Unlike the waiting time, the blocking time is only calculated for type 2 individuals. That is because type 1 individuals cannot be blocked. Thus, one only needs to consider the pathway of type 2 individuals to get the mean blocking time of the system. The mean blocking time can by calculated using:

$$B = \frac{\sum_{(u,v)\in S_A^{(2)}} \pi_{(u,v)}\, b(\mathcal{A}_2(u,v))}{\sum_{(u,v)\in S_A^{(2)}} \pi_{(u,v)}} \tag{10}$$

Here $S_A^{(2)}$ is the set of accepting states of type 2 individuals (defined in equation (8)) and $\mathcal{A}_i(u,v)$ for $i \in \{1,2\}$ is the state that the system would go to when the system is at state (u,v) and an individual of type i arrives.

$$\mathcal{A}_1(u,v) = (u, v+1) \tag{11}$$

$$\mathcal{A}_2(u,v) = \begin{cases} (u, v+1), & \text{if } v < T \\ (u+1, v), & \text{if } v \geq T \end{cases} \tag{12}$$

The term $b(u,v)$ is the mean time that an individual will be blocked for, when the individual arrives in the system at state (u,v). For all the states of the system $b(u,v)$ is given by:

$$b(u,v) = \begin{cases} 0, & \textbf{if } (u,v) \notin S_b \\ c(u,v) + b(u-1,v), & \textbf{if } v = N = T \\ c(u,v) + b(u,v-1), & \textbf{if } v = N \neq T \\ c(u,v) + p_s(u,v)b(u-1,v) + p_a(u,v)b(u,v+1), & \textbf{if } u > 0 \textbf{ and } \\ & \quad v = T \\ c(u,v) + p_s(u,v)b(u,v-1) + p_a(u,v)b(u,v+1), & \textbf{otherwise} \end{cases} \tag{13}$$

Note that S_b is defined as the set of states where individuals can be blocked and is given by:

$$S_b = \{(u,v) \in S \mid u > 0\} \tag{14}$$

Additionally, $c(u,v)$ is the mean sojourn time for each state and p_s and p_a are the probabilities that the next event to occur will be a service completion or an arrival of a type 1 individual:

$$c(u,v) = \begin{cases} \frac{1}{\min(v,C)\mu}, & \text{if } v = N \\ \frac{1}{\lambda_1 + \min(v,C)\mu}, & \text{otherwise} \end{cases} \tag{15}$$

$$p_s(u,v) = \frac{\min(v,C)\mu}{\lambda_1 + \min(v,C)\mu}, \qquad p_a(u,v) = \frac{\lambda_1}{\lambda_1 + \min(v,C)\mu} \tag{16}$$

The system of equations produced by (13) can be solved by considering the linear system $Zx = y$. Assuming i and j represent states $(u_i, v_i), (u_j, v_j) \in S_b$ then Z_{ij} is given by:

$$Z_{ij} = \begin{cases} p_a, & \textbf{if } j = i+1 \textbf{ and } v_i \neq N \\ p_s, & \textbf{if } j = i-1 \textbf{ and } v_i \neq N, v_i \neq T \\ & \textbf{or } j = i - N + T \textbf{ and } u_i \geq 2,\ v_i = T \\ 1, & \textbf{if } j = i-1 \textbf{ and } v_i = N \\ -1, & \textbf{if } i = j \\ 0, & \textbf{otherwise} \end{cases} \tag{17}$$

Equation (18) shows this.

$$Z = \begin{pmatrix} -1 & p_a & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ p_s & -1 & p_a & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ 0 & p_s & -1 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 1 & -1 & 0 & 0 & 0 & \dots & 0 & 0 \\ p_s & 0 & 0 & \dots & 0 & 0 & -1 & p_a & 0 & \dots & 0 & 0 \\ 0 & 0 & 0 & \dots & 0 & 0 & p_s & -1 & p_a & \dots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 1 & -1 \end{pmatrix}, x = \begin{pmatrix} b(1,T) \\ b(1,T+1) \\ b(1,T+2) \\ \vdots \\ b(1,N) \\ b(2,T) \\ b(2,T+1) \\ \vdots \\ b(M,N) \end{pmatrix}, y = \begin{pmatrix} -c(1,T) \\ -c(1,T+1) \\ -c(1,T+2) \\ \vdots \\ -c(1,N) \\ -c(2,T) \\ -c(2,T+1) \\ \vdots \\ -c(M,N) \end{pmatrix} \tag{18}$$

Additional details on the blocking time formula (10) can be found in appendix C.

Figure 4 illustrates a comparison between the formulas that arise from the Markov chain model and the equivalent values of the blocking time extracted from discrete event simulation (appendix A). The blocking time is calculated using both methods for a range of values of λ_2. The figure is used to demonstrate the accuracy of the blocking time formula of the constructed queueing model as well as the effect of truncating the model. The simulation was ran 100 times and the recorded mean blocking time at each iteration is used to populate the violin plots. Similar to figure 3, these plots shows a comparison between the calculated mean blocking time using Markov chain, using a truncated simulation and using a simulation without the artificial parameters N and M. The blocking times generated by the truncated simulation match the ones generated by the Markov chains model. Note that this comparison includes only type 2 individuals since type 1 individuals cannot be blocked.

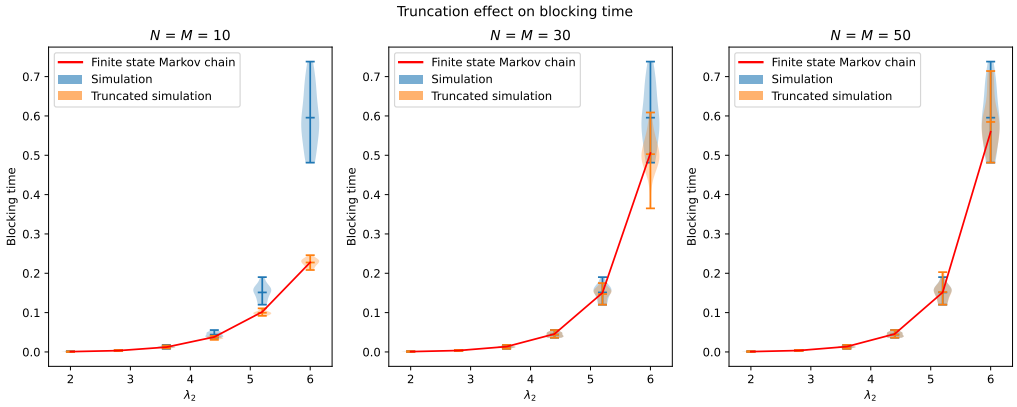

Figure 4: Comparison of mean blocking time between values obtained from the Markov chain formula, values obtained from the truncated simulation and values obtained from the untruncated simulation.

3.3. Proportion of individuals within target

Another performance measure that is taken into consideration is the proportion of individuals whose time in the hospital (waiting and service time) is within a specified time target t. Similar to section 3.1, three formulas are needed for this performance measure.

The proportion of type 1 individuals within a time target:

$$P(X^{(1)} < t) = \frac{\sum_{(u,v)\in S_A^{(1)}} P(X^{(1)}_{\mathcal{A}_1(u,v)} < t)\pi_{u,v}}{\sum_{(u,v)\in S_A^{(1)}} \pi_{u,v}} \tag{19}$$

The proportion of type 2 individuals within a time target:

$$P(X^{(2)} < t) = \frac{\sum_{(u,v)\in S_A^{(2)}} P(X^{(2)}_{\mathcal{A}_2(u,v)} < t)\pi_{u,v}}{\sum_{(u,v)\in S_A^{(2)}} \pi_{u,v}} \tag{20}$$

The terms $\mathcal{A}_1(u,v)$ and $\mathcal{A}_2(u,v)$ are defined by equations 11 and 12 in section 3.2. The overall proportion individuals within a time target (where $P_{L_1'}$ and $P_{L_1'}$ are defined in (9)):

$$P(X<t) = \frac{\lambda_1 P_{L_1'}}{\lambda_2 P_{L_2'} + \lambda_1 P_{L_1'}} P(X^{(1)}<t) + \frac{\lambda_2 P_{L_2'}}{\lambda_2 P_{L_2'} + \lambda_1 P_{L_1'}} P(X^{(2)}<t) \tag{21}$$

Here $P(X^{(1)}_{(u,v)})$ and $P(X^{(2)}_{(u,v)})$ are defined as the proportion of individuals within the time target t when starting from state (u,v). These expression can be calculated by:

$$P(X^{(1)}_{(u,v)}<t) = \begin{cases} 1 - \sum_{i=0}^{v-1} \frac{1}{i!} e^{-\mu t}(\mu t)^i, & \textbf{if } C=1 \\ & \textbf{and } v>1 \\ 1 - (\mu C)^{v-C} \mu \sum_{k=1}^{|\vec{r}|} \sum_{l=1}^{r_k} \frac{\Psi_{k,l}(-\lambda_k) t^{r_k-l} e^{-\lambda_k t}}{(r_k-l)!(l-1)!}, & \textbf{if } C>1 \\ & \textbf{and } v>C \\ 1 - e^{-\mu t}, & \textbf{if } v \leq C \end{cases} \tag{22}$$

where $\vec{r} = (v-C, 1)$, $\vec{\lambda} = (C\mu, \mu)$ and $\lambda_0 = 0, r_0 = 1$.

$$P(X^{(2)}_{(u,v)}<t) = \begin{cases} 1 - \sum_{i=0}^{\min(v,T)-1} \frac{1}{i!} e^{-\mu t}(\mu t)^i, & \textbf{if } C=1 \\ & \textbf{and } v,T>1 \\ 1 - (\mu C)^{\min(v,T)-C} \mu \sum_{k=1}^{|\vec{r}|} & \textbf{if } C>1 \\ \quad \times \sum_{l=1}^{r_k} \frac{\Psi_{k,l}(-\lambda_k) t^{r_k-l} e^{-\lambda_k t}}{(r_k-l)!(l-1)!}, & \textbf{and } v,T>C \\ 1 - e^{-\mu t}, & \textbf{if } v \leq C \\ & \textbf{or } T \leq C \end{cases} \tag{23}$$

where $\vec{r} = (\min(v,T)-C, 1)$, $\vec{\lambda} = (C\mu, \mu)$ and $\lambda_0 = 0, r_0 = 1$.

The function $\Psi_{k,l}$ used in equations (22) and (23) is defined as:

$$\Psi_{k,l}(t) = \begin{cases} \frac{(-1)^l (l-1)!}{\lambda_2} \left[\frac{1}{t^l} - \frac{1}{(t+\lambda_2)^l} \right], & k=1 \\ -\frac{1}{t(t+\lambda_1)^{r_1}}, & k=2 \end{cases}$$

Please refer to appendix D for a more in-depth explanation of the origins of equations (19) - (23).

Figure 5 shows a comparison of the mean proportion of individuals within target when using Markov chains and discrete event simulation (appendix A). The figure is used to demonstrate the accuracy of the formula for the proportion of individuals within time of the constructed queueing model as well as the effect that truncating the model has on the formula. The simulation was ran 100 times and the recorded proportions at each iteration is used to populate the violin plots. Similar to figures 3 and 4, these plots shows a comparison between the calculated mean proportion of individuals within time using Markov chain, using a truncated simulation and using a simulation

without the artificial parameters N and M. The proportions generated by the truncated simulation match the ones generated by the Markov chains model. Note that this comparison includes both type 1 and type 2 individuals. A separate comparison of only type 1 and only type 2 individuals can be found in appendix E.

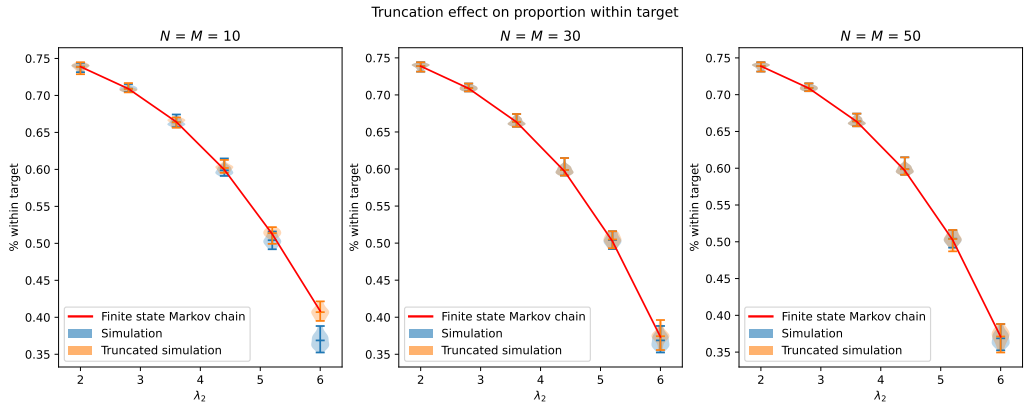

Figure 5: Comparison of mean proportion of individuals within target time between values obtained from the Markov chain formula, values obtained from the truncated simulation and values obtained from the untruncated simulation.

3.4. Truncation effect timings

The choice of the artificial parameters N and M is an important decision of the model. In the untruncated simulation these values are not needed. This is not possible when obtaining the steady state probabilities of the finite state Markov chain. Table 1 shows the relative timings of the different approaches used to get the performance measures.

	Simulation timings		**Markov chain timings**		
Value of N and M	**Single trial**	**A hundred trials**	**Waiting time formula**	**Blocking time formula**	**Proportion within time formula**
10	1	144.3	0.015	0.014	0.014
30	1	143.4	3.731	3.828	3.649
50	1	139.8	31.57	38.39	31.98
∞	1	142.1	N/A	N/A	N/A

Table 1: Relative timings of the simulation and Markov chain model

4. Strategic manipulation of the ED-EMS interface

The problem studied is a 3-player normal form game. The players are:

- the decision makers of two Emergency Departments (EDs)

- the Emergency Medical Services (EMS) that distribute individuals in ambulances to the EDs

This is a standard normal form game [33], in that each player in this game has their own objectives which they aim to optimise. More specifically, the EDs' objective is captured by an upper bound of the time that a fixed proportion of individuals spend in the system, while the EMS aims to minimise the time that its ambulances are blocked. This can be generalised for any such system where instead of EDs there are some queueing systems and instead of the EMS there is some distributor that allocates individuals to the queueing systems.

The parameters of the model correspond to the following parameters of the ED and the EMS:

- λ_2: The rate of patients (who can be blocked) that the EMS receives and distributes to EDs
- λ_{1_i}: The arrival rate of other patients to ED $i \in \{A, B\}$
- μ_i: The service rate of patients at ED $i \in \{A, B\}$
- C_i: The number of available resources (healthcare professionals) in the ED $i \in \{A, B\}$
- T_i: The action that ED $i \in \{A, B\}$ chooses to play which corresponds to the threshold at which they do not accept EMS patients.
- N_i: The total patient capacity of the ED $i \in \{A, B\}$
- M_i: The total parking capacity of the ED $i \in \{A, B\}$
- t: The time target for both EDs
- $\alpha \in [0, 1]$: Weighted average of blocking time and lost individuals (equation 25)

The strategies of the two EDs are the range of thresholds that they can choose from and their utilities is the proportion of individuals whose time in the system is within a predetermined target time. The EMS has to decide how to distribute its patients among the two EDs so that the weighted combination of the ambulance blocking time and the percentage of lost ambulances is minimised. This can be illustrated by figure 6. The interaction between the two EDs is a normal form game that is then used to inform the decision of the EMS. Note that the formulated game here assumes that prior to making a choice the EMS knows the strategies that each ED is playing (figure 7). This corresponds to reacting to experienced delays.

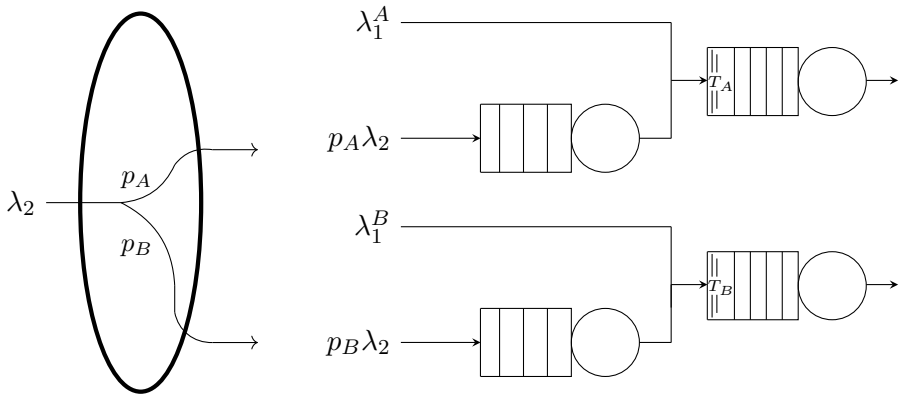

Figure 6: A diagrammatic representation of the game theoretic model. Patients arrive at the EMS at a rate of λ_2 and then a proportion of them are distributed to hospitals A (p_A) and the remaining proportion to hospital B (p_B) so that $p_A + p_B = 1$. The corresponding arrival rates of type 2 patients to hospitals A and B are thus given by: $p_A\lambda_2$ and $p_B\lambda_2$.

The queueing systems of the hospitals are designed in such a way where they can accept two types of individuals (section 3). Each hospital may then choose to block type 2 individuals when the hospital reaches a certain capacity. The strategy sets for each hospital is the set $\{T \in \mathbb{N} \mid 1 \leq T \leq N\}$ where $N \in \{N_A, N_B\}$ are the total capacities of hospitals A and B. We denote the chosen actions from the strategy set as T_A, T_B and call these *threshold*s.

Both hospitals follow a queueing model with two waiting spaces for individuals. The first waiting space (i.e. the waiting space of the hospital) is where the patients queue right before receiving their service and has a queue capacity of $N - C$, where N is the total capacity of the hospital and C is the number of healthcare professionals able to see them. The second waiting space (i.e. the parking space for ambulances) is where ambulances, that are sent from the ambulance service, stay until their patients are allowed to enter the hospital. The parking space has a capacity of M and no servers. This is shown diagrammatically in Figure 1.

Note here that both types of individuals can become lost to the system. An individual allocated from the ambulance service becomes lost to the system whenever an arrival occurs and the parking space is at full capacity (M ambulances already parked). Similarly, type 1 individuals get lost whenever they arrive at the waiting space of the hospital and it is at full capacity ($N-C$ individuals already waiting).

Following this queueing model, the two queueing systems' choice of strategy will then rely solely on satisfying their own objective. The objective function is defined as:

$$\arg\max_{T_i} \quad -\left(\hat{P} - P(W_i < t)\right)^2 \qquad i \in A, B \tag{24}$$

where W is the waiting time of a potential individual, t is the time target and $\hat{P}$ is the percentage of individuals need to be within that target. In other words, their aim is to find the threshold that

minimises the difference between the probability $P(W_i < t)$ and the percentage goal (or maximise its negation).

The third player, the ambulance service, has their own choices to make and their own goals to satisfy. The strategy set of the third player is the proportion $0 \leq p_A \leq 1$ of individuals to send to hospital A. Similarly the proportion of individuals to send to hospital B is given by $p_B = 1 - p_A$. In addition, the ambulance service aims to minimise any potential blockages that may occur, given the pair of thresholds chosen by the two hospitals. Thus, its objective is to minimise the blocked time of the individuals (B_A and B_B) that they send to hospitals A and B. Apart from the time being blocked, an additional aspect that may affect the decision of the distributor is the proportion of lost individuals L_A and L_B. Equation 25 can be used to capture a mixture between the two objectives L_i and B_i where $i \in \{A, B\}$:

$$(p_A, p_B) \quad s.t. \quad \alpha L_A(p_A) + (1-\alpha)B_A(p_A) = \alpha L_B(p_B) + (1-\alpha)B_B(p_B) \tag{25}$$

Here, α represents the "importance" of each objective, where high α indicates a higher weight on the proportion of lost individuals and smaller α a higher weight on the time blocked. The choice of p_A and p_B rely solely on equation 25. Note that the current system is modelled using unobservable queues which is not necessarily an unrealistic approach [39]. However, there are several other factors that can affect the routing of the patients that are outside the scope of this paper. For example the distance from each hospital or even the priority level of each patient may be a significant factor that affects the ambulance service's decision.

Using either equation (25) or (24) gives an imperfect information extensive form game. An imperfect information game is defined as an extensive form game where some of the information about the game state is hidden for at least one of the players [6]. In this study the state of the problem that is hidden is the threshold that each of the hospitals chooses to play. In other words, each hospital chooses to play a strategy without knowing the other hospital's strategy. The ambulance service then, fully aware of the chosen threshold strategies, distributes individuals among the two systems in order to minimise the time that its ambulances will be blocked. Figure 7 illustrates this.

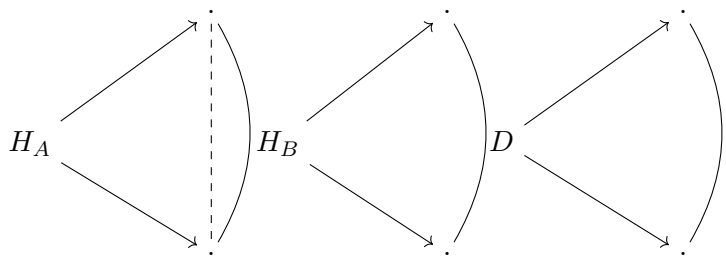

Figure 7: Imperfect information Extensive Form Game between the distributor and the 2 queueing systems

Hospital H_A decides on a threshold, then the hospital H_B chooses its own threshold, without knowing the strategy of H_A, and finally the ambulance service makes its choice. Note here that

the dotted line represents the fact that H_B is unaware of the state of the game when making its own decisions.

From equation 24 the utilities of the hospitals can be formulated. The 2-player normal form game between the two hospitals is defined by:

- **Players**: Hospitals H_A and H_B
- **Strategy spaces**: $T_A = \{1, 2, \ldots, N_A\}, T_B = \{1, 2, \ldots, N_B\}$
- **Utilities**:

$$U^i_{T_A,T_B} = 1 - \left(\hat{P} - P(W_i < t)\right)^2 \quad \textbf{where } i \in \{A, B\} \tag{26}$$

Consequently, the payoff matrices of the game can be populated by these utilities:

$$A = \begin{pmatrix} U^A_{1,1} & U^A_{1,2} & \ldots & U^A_{1,N_B} \\ U^A_{2,1} & U^A_{2,2} & \ldots & U^A_{2,N_B} \\ \vdots & \vdots & \ddots & \vdots \\ U^A_{N_A,1} & U^A_{N_A,2} & \ldots & U^A_{N_A,N_B} \end{pmatrix}, B = \begin{pmatrix} U^B_{1,1} & U^B_{1,2} & \ldots & U^B_{1,N_B} \\ U^B_{2,1} & U^B_{2,2} & \ldots & U^B_{2,N_B} \\ \vdots & \vdots & \ddots & \vdots \\ U^B_{N_A,1} & U^B_{N_A,2} & \ldots & U^B_{N_A,N_B} \end{pmatrix} \tag{27}$$

Based on the choice of strategy of these two hospitals, the ambulance service will then make their own choice of the proportion of individuals to send to each system.

4.1. Building the game

The problem defined in this section describes a normal-form game between the decision makers of two hospitals and a third player, the ambulance service, that decides how to distribute individuals to the two systems. The strategy space of the two hospitals is defined as the possible values that the threshold parameter can take $T_i \in [1, N_i]$. Then, the ambulance service has to decide on the proportion of individuals to send to each hospital p_A and p_B, where $p_A, p_B \in [0, 1]$ and $p_A + p_B = 1$. In practice this would correspond to a learned behaviour through experience of waiting at each hospital. Figure 7 shows a diagrammatic representation of the game to be played and the decisions to be made. As described in section 4, hospital A decides on a strategy and at the same time, hospital B chooses its own threshold but unaware of the first hospital's choice. Finally, the ambulance service makes its choice based on the strategies that the hospitals chose to play.

The utilities to each player can be represented by 3 matrices; the two payoff matrices of the normal form game and the routing matrix. The payoff matrices and their utilities are defined by equations (26) and (27).

The routing matrix holds the values (p_A, p_B) which are the proportion of ambulance patients to send to queueing systems A and B. Each pair (p_A, p_B) can be calculated using equation (25) and is essentially a best response to the actions of the hospitals. Thus, using equation (25) for all

possible sets of thresholds we can get the full routing matrix that consists of the proportions to send to hospital A (p_A) and to hospital B (p_B).

$$R = \begin{pmatrix} (p^A_{1,1}, p^B_{1,1}) & (p^A_{1,2}, p^B_{1,2}) & \cdots & (p^A_{1,N_B}, p^B_{1,N_B}) \\ (p^A_{2,1}, p^B_{2,1}) & (p^A_{2,2}, p^B_{2,2}) & \cdots & (p^A_{2,N_B}, p^B_{2,N_B}) \\ \vdots & \vdots & \ddots & \vdots \\ (p^A_{N_A,1}, p^B_{N_A,1}) & (p^A_{N_A,2}, p^B_{N_A,2}) & \cdots & (p^A_{N_A,N_B}, p^B_{N_A,N_B}) \end{pmatrix} \tag{28}$$

Note that since $p^A_{i,j} + p^B_{i,j} = 1$ the routing matrix needs only to store one of the two values; either $p^A_{i,j}$ or $p^B_{i,j}$. Thus, the routing matrix R can be simplified to:

$$R = \begin{pmatrix} p^A_{1,1} & p^A_{1,2} & \cdots & p^A_{1,N_B} \\ p^A_{2,1} & p^A_{2,2} & \cdots & p^A_{2,N_B} \\ \vdots & \vdots & \ddots & \vdots \\ p^A_{N_A,1} & p^A_{N_A,2} & \cdots & p^A_{N_A,N_B} \end{pmatrix} \tag{29}$$

The game can thus be partitioned into a normal form game between the two hospitals and then finding the ambulance service's best strategy.

Consider Figure 7 and the flow of the game that was described (i.e. $H_A, H_B \rightarrow D$). Due to the fact that the payoff matrices A and B depend on the routing matrix R the entries of the matrices are calculated in a backwards way ($D \rightarrow H_A, H_B$). This is done using backwards induction. For each action choice of the hospitals, first solve the game from the ambulance's point of view. This in effect results in a 2-player normal form game representing the hospital's point of view. Thus, for every pair of strategies T_A, T_B, the values of p_A and p_B that satisfy equation (25) are found numerically using Brent's bisection algorithm [7]. Each pair (p_A, p_B) corresponds to the best response of the ambulance service to the two hospitals' played strategies. Finally, using the routing matrix, equation (26) can also be used to populate the payoff matrices of the hospitals since we now know the arrival rate of each hospital.

Having calculated the payoff matrices A and B, several algorithms can be used to measure some form of the emergent behaviour. One possibility would be to compute the Nash equilibrium which is the point of the game were both players have no incentive to deviate from their played strategies [28]. In other words their chosen strategies are a best response to each other. Computation of Nash equilibria can be done in a relatively efficient way using the Lemke Howson algorithm [30]. Lemke-Howson uses best response polytopes to get one of the Nash equilibrium of the game. Other algorithms exists that will compute all Nash equilibria but for large games the computational complexity becomes problematic. All game theoretic calculations were done in Python using the Nashpy library [44].

Another approach to measuring emergent behaviour is to consider the emergence itself and not only stable end points. Indeed, some Nash equilibria might not arise naturally. Thus in order to analyse the strategies played by the two hospitals, the learning algorithm asymmetric replicator

dynamics is used [1]. The two hospitals are modelled as two separate populations where each individual in the population is assigned a strategy. As the game progresses the proportion of each player playing each strategy changes based on their previous interactions. The fitness of each strategy is defined as:

$$f_x = Ay, \quad f_y = x^T B \tag{30}$$

Here, $x \in \mathbb{R}^{m\times 1}$ and $y \in \mathbb{R}^{n\times 1}$ correspond to the proportion of individuals that play each strategy in each population. Similarly, the average fitness of each strategy is given by:

$$\phi_x = f_x x^T, \quad \phi_y = f_y y \tag{31}$$

The rate of change of strategy i of both types of individuals is captured by:

$$\frac{dx}{dt}_i = x_i((f_x)_i - \phi_x), \quad \text{for all } i \tag{32}$$

$$\frac{dy}{dt}_i = y_i((f_y)_i - \phi_y), \quad \text{for all } i \tag{33}$$

In addition to asymmetric replicator dynamics, the learning algorithms fictitious play and stochastic fictitious play [20] were used.

4.2. Results

This subsection aims to analyse how the gaming framework can affect the performance measures of the two hospitals and how to escape certain inefficient situations.

The most commonly used method for analysing normal form games is the Nash equilibrium described in section 4.1. Consider the following game:

- $\lambda_{1_A} = 1$
- $\lambda_{1_B} = 2$
- $\lambda_2 = 2$
- $\mu_A = 2$
- $\mu_B = 2.5$
- $C_A = 2$
- $C_B = 2$
- $R = 2$
- $N_A = 10$
- $N_B = 10$
- $M_A = 6$
- $M_B = 6$
- $\alpha = 0.5$

The set of possible actions to choose from for player 1 and player 2 is the set of thresholds that the EDs can choose from:

$$T_A \in [1, N_A], \quad T_B \in [1, N_B] \tag{34}$$

For this example, the only Nash equilibrium of the game is achieved when both players choose a threshold of $T_A = 10, T_B = 10$. This means that the two players' best response to each other is to only block ambulances when they are full.

Nash equilibria is a theoretical measure which can be inconsistent with intuitive notions about what should be the outcome of a game [35]. Therefore it might not be the best way to describe human behaviour. Since the work of Maynard Smith [40], evolutionary game theory gives the tools for the emergence of stable behaviour. One such model that allows for asymmetric payoffs, as is the case above, is replicator dynamics described in section (4.1). Stable outcomes of this algorithm will correspond to a subset of Nash equilibria but more importantly, will give a model of emergent behaviour.

The use of a learning algorithm allows to investigate, not only the outcome of the game, but also how that outcome is reached. Consider Figure 8. By running asymmetric replicator dynamics on the system the behaviour that emerges can be observed. It can be seen that for this particular set of parameters the strategies of the two hospitals converge over time. Both hospital 1 (row player) and hospital 2 (column player) seem to be playing the same strategy s_{10} which indicates that thresholds $T_A = 10$ and $T_B = 10$ are played. What is more important in this example is how the two hospitals reached these decisions which also highlights the importance of using a learning algorithm. Hospital 2 is able to reach the decision in a short amount of time while hospital 1 takes longer and goes through numerous strategies to get there.

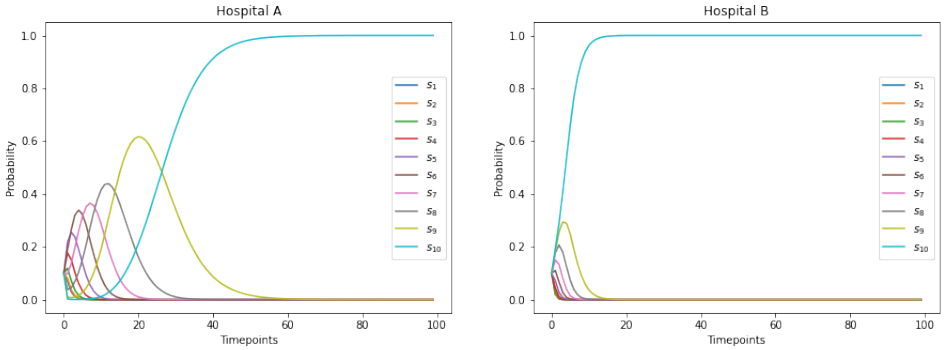

Figure 8: Asymmetric replicator dynamics run

In order to analyse how efficient the strategies played at every iteration are, the concept of the price of anarchy is used. Price of anarchy is a measure that is used to quantify the efficiency of a behaviour [37]. In other words the price of anarchy is the worst-case equilibria measure and it is defined as:

$$\text{PoA} = \frac{\max_{s \in E} F(s)}{\min_{s \in S} F(s)} \tag{35}$$

Here, S is the set of all sets of strategies (s_A, s_B), E is the set of all possible sets of equilibria and F is the cost function to measure the efficiency for.

To study the efficiency of the strategies being played, a new concept is introduced that considers the ratio between each hospital's best achievable blocking time and the one that is being played.

This new concept is defined as the compartmentalised price of anarchy of the players of the game and is defined as $PoA_i(\tilde{s})$, where $i \in \{A, B\}$ to distinguish among the two players/hospitals where $\tilde{s}$ is the strategy that is being played by player i. The compartmentalised price of anarchy aims to measure inefficiencies in the model. The PoA for the blocking time of player i for strategy $\tilde{s}$ is given by:

$$PoA_i(\tilde{s}) = \frac{B_i(\tilde{s})}{\min_{s \in S_i} B_i(s)} \tag{36}$$

For this particular scenario, two busy queueing systems will be used with a high traffic intensity ($\rho > 1$). Consider a game with two smaller (lower N_i, M_i) and busier (higher λ_{1_i} and λ_2) hospitals with the following set of parameters:

- $\lambda_{1_A} = 4.5$
- $\mu_A = 2$
- $C_A = 3$
- $T_A \in [1, N_A]$
- $N_A = 6$
- $M_A = 5$
- $\lambda_{1_B} = 6$
- $\mu_B = 3$
- $C_B = 2$
- $T_B \in [1, N_B]$
- $N_B = 7$
- $M_B = 4$
- $\lambda_2 = 10.7$
- $R = 2$
- $\alpha = 0.9$

Initial scenario: Using equation (36) and asymmetric replicator dynamics, the emergent behaviour can be measured and the compartmentalised price of anarchy at every iteration for both players. Figure 9 shows the strategies that are being played and the values of $PoA_A(s)$ and $PoA_B(s)$ for all iterations of the learning algorithm until it reaches an evolutionary stable pair of strategies.

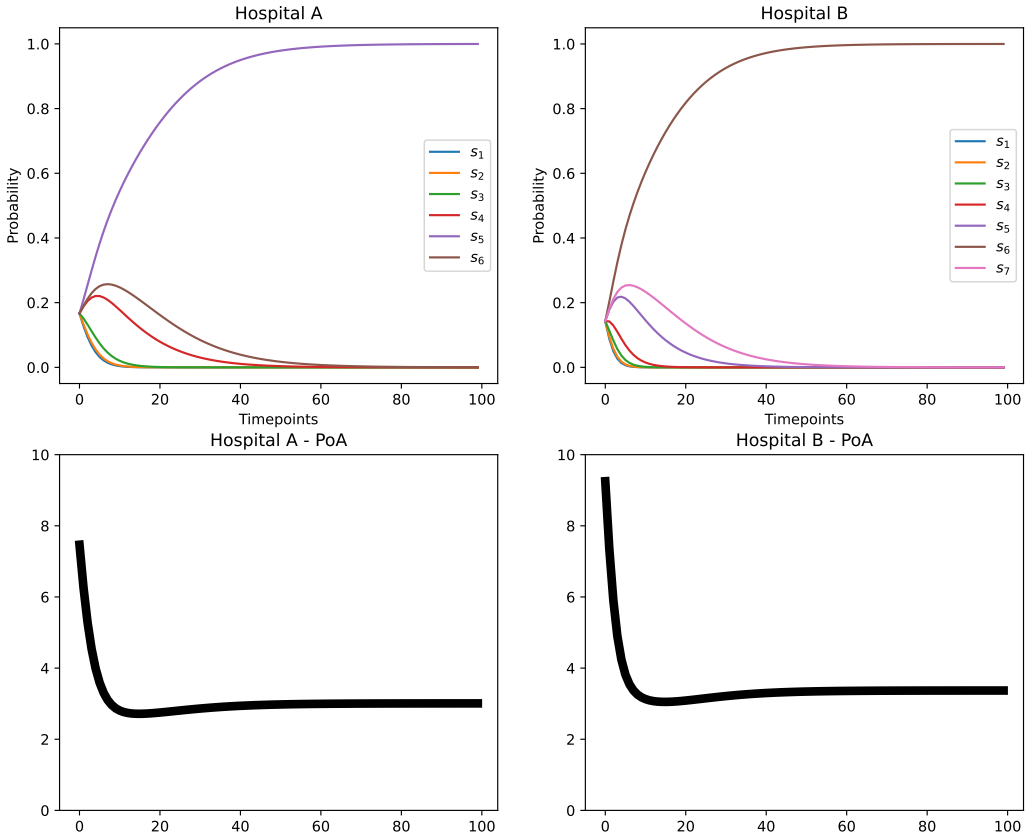

Figure 9: The strategies played when running asymmetric replicator dynamics along with the compartmentalised price of anarchy of the blocking time at each iteration of the learning algorithm

The learning algorithm reaches a stable pair of strategies where $T_A = 5$ and $T_B = 6$. After a number of iterations the price of anarchy for both players stabilises and barely increases until the final iteration.

Increasing λ_2: Figure 10 shows a similar run of the algorithm but when the strategies begin to stabilise an increase in the arrival rate of ambulances occurs (i.e $\lambda_2 = 24$).

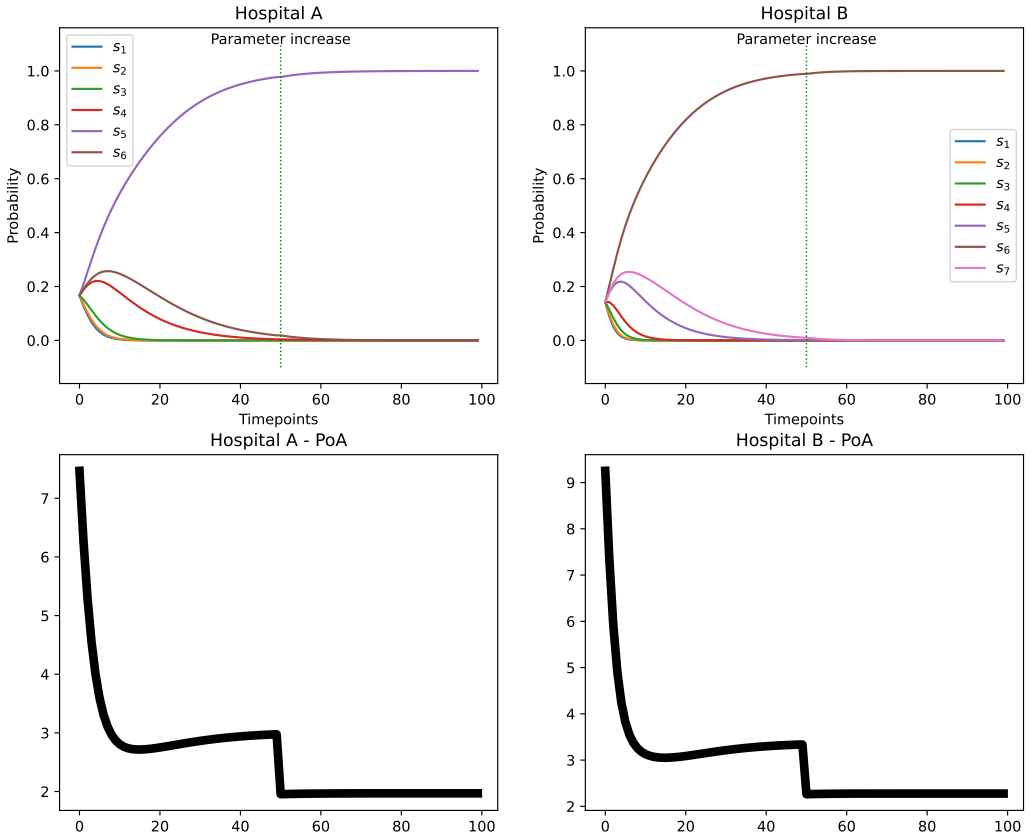

Figure 10: The strategies played when running asymmetric replicator dynamics along with the compartmentalised price of anarchy of the blocking time at each iteration of the learning algorithm. After a number of iterations the arrival rate of ambulance patients is significantly increased to flood the system completely $\lambda_2 = 24$.

By increasing λ_2 there is no change as to how players behave ($T_A = 5, T_B = 6$), but the efficiency of the system does change. There is a decline in the price of anarchy of the blocking time which at first glance indicates that upon flooding the system it becomes the loss in efficiency due to rational individual behaviour decreases. This is non-sensical though. What it really shows is that the steep increase in λ_2 leaves the system unable to cope regardless of the decisions made.

Increasing number of servers C_A and C_B: Figure 11 shows a run of asymmetric replicator dynamics with a change in the number of servers of the hospitals. The number of servers are increased from $C_A = 3, C_B = 2$ to $C_A = 4, C_B = 3$.

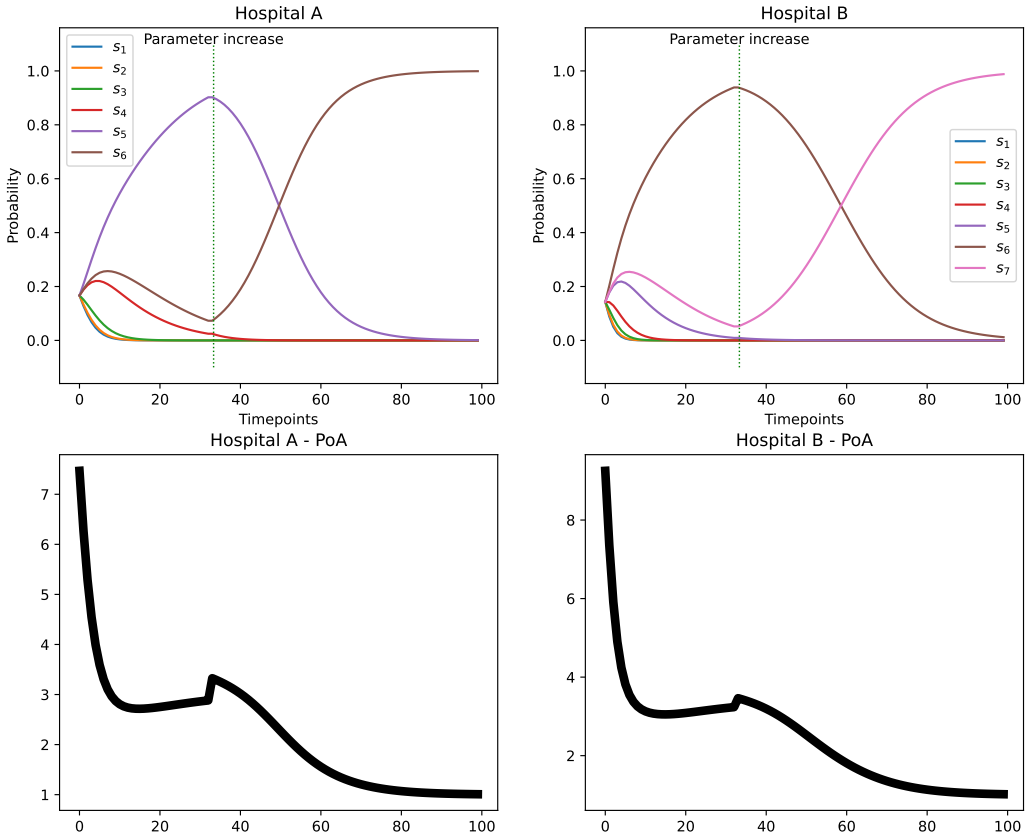

Figure 11: The strategies played when running asymmetric replicator dynamics along with the compartmentalised price of anarchy of the blocking time at each iteration of the learning algorithm. After a number of iterations the number of servers for both systems are increased by one.

In this case, both the behaviour as well as the price of anarchy change. The players change their strategies from $T_A = 5, T_B = 6$ to $T_A = 6, T_B = 7$ and the PoA of the blocking time goes down. By adding more resources to the models they are able to increase their efficiency. Although this is a good way to escape such inefficiencies, it might not always be cost efficient.

Incentivising players: From Figures 10 and 11 it can be seen that we can change some parameters of the model to make it more efficient. The approach used on Figure 12 is slightly different than the previous cases. Once the played strategies in asymmetric replicator dynamics strategies converge the payoff matrices of the two are scaled in such a way so that the utilities of the selected strategy are penalised. This corresponds to a precise policy change where more societally beneficial behaviours are incentivised.

Matrices A and B represent the original payoff matrices while matrices $\tilde{A}$ and $\tilde{B}$ represent the incentivised payoff matrices. It can be observed that matrix $\tilde{A}$ is a scaled version of matrix A only on the row that is most frequently played and similarly matrix $\tilde{B}$ of matrix B only on the column that is most frequently played (matrix A: row 5, matrix B: column 6, see Figure 12). Note that for the presentation of data, an affine transformation has been applied to obtain the values of the payoff matrices ($A_ij = 10000(a_{ij} - 0.999)$ and $B_{ij} = 10000(b_{ij} - 0.999)$ where a_{ij} and b_{ij} are the raw utilities). The results are not affected by this scaling.

$$A = \begin{bmatrix} 5.0518 & 5.0518 & 5.0518 & 5.0518 & 5.0518 & 5.0518 & 5.0518 \\ 5.4989 & 5.4977 & 5.4960 & 5.4924 & 5.4844 & 5.4654 & 5.3875 \\ 6.8232 & 6.8192 & 6.8150 & 6.8065 & 6.7871 & 6.7334 & 6.4906 \\ 9.0298 & 9.0244 & 9.0187 & 9.0078 & 8.9827 & 8.9082 & 8.5145 \\ 9.9996 & 9.9994 & 9.9992 & 9.9987 & 9.9972 & 9.9893 & 9.8571 \\ 8.7740 & 8.8006 & 8.8249 & 8.8660 & 8.9438 & 9.1295 & 9.7157 \end{bmatrix}$$

$$B = \begin{bmatrix} 1.7127 & 2.5822 & 4.6186 & 6.8497 & 8.9418 & 9.9999 & 8.2148 \\ 1.7127 & 2.5477 & 4.5634 & 6.8047 & 8.9150 & 9.9996 & 8.3358 \\ 1.7127 & 2.4528 & 4.3784 & 6.6441 & 8.8278 & 9.9965 & 8.5306 \\ 1.7127 & 2.4141 & 4.2867 & 6.5470 & 8.7656 & 9.9919 & 8.6745 \\ 1.7127 & 2.3415 & 4.0998 & 6.3265 & 8.6058 & 9.9716 & 8.9634 \\ 1.7127 & 2.1269 & 3.4930 & 5.4885 & 7.8353 & 9.7075 & 9.7322 \end{bmatrix}$$

$$\tilde{A} = \begin{bmatrix} 5.0518 & 5.0518 & 5.0518 & 5.0518 & 5.0518 & 5.0518 & 5.0518 \\ 5.4989 & 5.4977 & 5.4960 & 5.4924 & 5.4844 & 5.4654 & 5.3875 \\ 6.8232 & 6.8192 & 6.8150 & 6.8065 & 6.7871 & 6.7334 & 6.4906 \\ 9.0298 & 9.0244 & 9.0187 & 9.0078 & 8.9827 & 8.9082 & 8.5145 \\ 6.9996 & 6.9994 & 6.9992 & 6.9987 & 6.9972 & 6.9893 & 6.8571 \\ 8.7740 & 8.8006 & 8.8249 & 8.8660 & 8.9438 & 9.1295 & 9.7157 \end{bmatrix}$$

$$\tilde{B} = \begin{bmatrix} 1.7127 & 2.5822 & 4.6186 & 6.8497 & 8.9418 & 6.9999 & 8.2148 \\ 1.7127 & 2.5477 & 4.5634 & 6.8047 & 8.9150 & 6.9996 & 8.3358 \\ 1.7127 & 2.4528 & 4.3784 & 6.6441 & 8.8278 & 6.9965 & 8.5306 \\ 1.7127 & 2.4141 & 4.2867 & 6.5470 & 8.7656 & 6.9919 & 8.6745 \\ 1.7127 & 2.3415 & 4.0998 & 6.3265 & 8.6058 & 6.9716 & 8.9634 \\ 1.7127 & 2.1269 & 3.4930 & 5.4885 & 7.8353 & 6.7076 & 9.7322 \end{bmatrix}$$

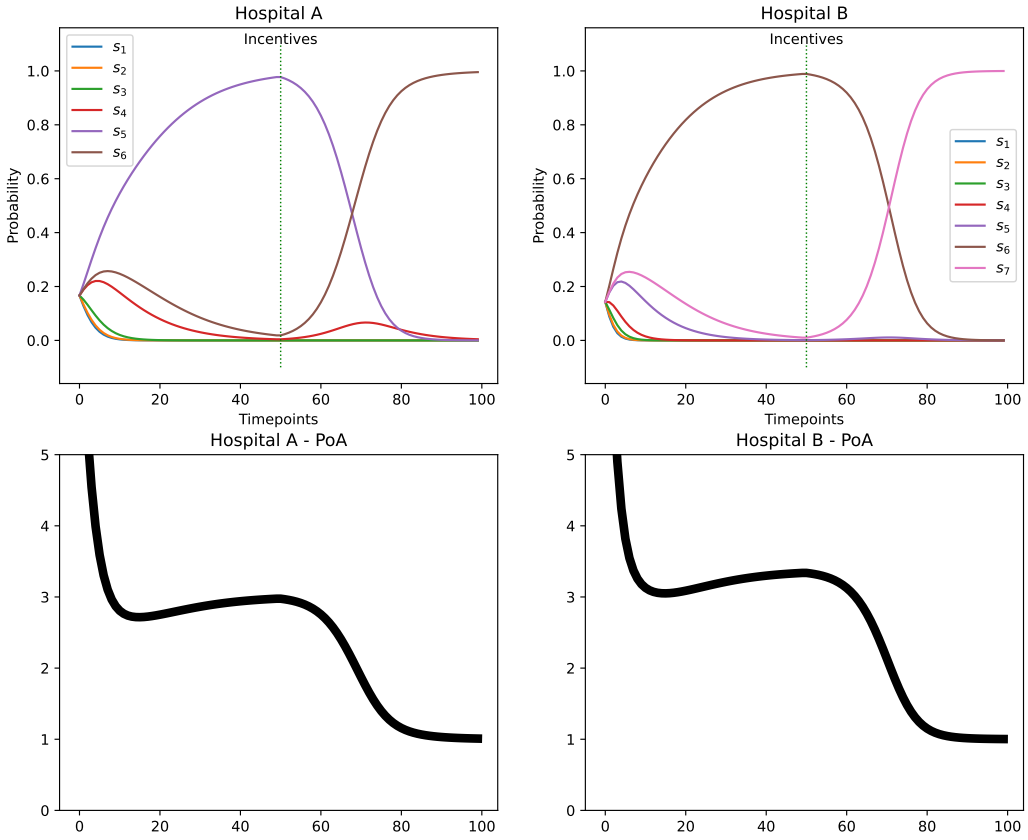

Figure 12: The strategies played when running asymmetric replicator dynamics along with the compartmentalised price of anarchy of the blocking time at each iteration of the learning algorithm. After a number of iterations the most dominant strategy is being penalised.

Figure 12 shows that players start playing strategies $T_A = 5$ and $T_B = 6$ and mid-run of the learning algorithm a penalty is applied to these strategies on the payoff matrix. By incentivising the players in such a way the players change their strategies to $T_A = 6$ and $T_B = 7$, and thus ambulance patients are accepted in the ED more often. Hence, the PoA for both EDs is decreased, meaning that the whole system is more efficient in terms of the blocking time.

5. Conclusion

The motivation behind this study has been that emergency departments are under a lot of pressure to treat patients. This is in practice often centrally controlled through a mechanism of some sort of performance measure target. This paper shows how this can negatively impact the

pathway of both the ambulance patients and the ambulance service itself. Due to some managerial decision making that takes place at the ED, ambulances stay blocked outside of the ED at the hospital's parking zone in an attempt to satisfy these regulations. The main contributions of this paper are:

- A queueing model with 2 consecutive waiting spaces where one would serve as a parking space for the ambulances;
- Analytic performance measure formulas for the queueing model;
- A 3-player game theoretic model between the EMS and two EDs;
- Numerical experiments showing emergent behaviour of gaming between EDs and the EMS.

Although our research is motivated by the particular EMS-ED example, our developed modelling framework and behavioural insights has application to similar systems across a range of sectors and settings. The queueing model can be applied to any setting where individuals may be blocked on a separate queue. An example of such setting can be any type of delivery service where customers can purchase goods either online or walk in the store. At busier times, the person delivering the food may be blocked outside the restaurant in an attempt to improve the waiting times for walk-in customers.

This study explores a generic 3-player game theoretic model between the decision makers of two queueing systems and a service that distributes individuals to these two systems (section 4). It also describes the construction of the underlying queueing theoretic model that has a tandem buffer and a single service centre (section 3). Furthermore, the formulas for the performance measures of the queueing model are also derived (sections 3.1, 3.2, 3.3). This novel queueing model is the first contribution of the paper. The game theoretic model is then applied to a healthcare scenario by looking at the interface between the EDs and the EMS. The inefficiencies that emerge from the perspective of the EMS were explored along with ways to apply some incentive mechanisms to improve them. The key findings from this paper that were observed when playing the game between two EDs and the EMS are:

- Inefficiencies can be learned and emerge naturally
- Targeted incentivisation of behaviours can help escape inefficiencies

The former relates to the results of asymmetric replicator dynamics that showed that inefficient scenarios can arise by playing the game while the latter implies that the learned inefficiencies can be escaped by carefully applying certain incentives to the players. This applied game theoretic model is the second main contribution of this paper.

The model presented here assumes the presence of only two players that can receive individuals. However, in a realistic healthcare scenario an ambulance may have to decide among multiple

EDs. An immediate extension of this work would be to consider a multiplayer system that could represent a group of hospitals in a concentrated area. Additionally, the game theoretic model that was created uses a discrete strategy space for the EDs (something that is also present in various related literature [17, 25]). The single threshold parameter that is used for the ED's decision may not be the best way to describe the model. In reality ED managers could have far more complex parameters for their decision making process. Moreover this work assumes that the EMS and EDs act in a selfish and rational way by only aiming to satisfy their own objectives. In fact, modelling behaviour in a healthcare setting is a much more complex task where some cooperation may often be observed. Another extension would be to explore the behaviour of the ED staff via an agent-based model. This in turn can be used to model emergent behaviour based on assumptions of individual behavioural traits of ED staff. Finally, future work can touch upon the derivation of a closed form formula for the steady state probability vector of the queueing model (section 3) to allow for faster computations of π.

Acknowledgements

The authors are most grateful for the funding and support received by The Healthcare Improvement Studies (THIS) institute for supporting Michalis Panayides with a Fellowship.

References

[1] Accinelli, E. and Carrera, E. J. S. (2011). Evolutionarily stable strategies and replicator dynamics in asymmetric two-population games. In *Dynamics, Games and Science I*, pages 25–35. Springer.

[2] Afeche, P. (2007). Decentralized service supply chains with multiple time-sensitive customer segments: Pricing capacity decisions and coordination. Technical report, Working paper, University of Toronto.

[3] Aitken, V. (2021). The red cross drafted into scotland's biggest hospital as ambulances queue up. *Daily Record*.

[4] Akkouchi, M. (2008). On the convolution of exponential distributions. *J. Chungcheong Math. Soc*, 21(4):501–510.

[5] Başar, T. and Srikant, R. (2002). A stackelberg network game with a large number of followers. *Journal of optimization theory and applications*, 115(3):479–490.

[6] Berwanger, D. and Doyen, L. (2008). On the power of imperfect information. *Leibniz International Proceedings in Informatics, LIPIcs*, 2:73–82.

[7] Brent, R. P. (1973). Algorithms for minimization without derivatives, chap. 4.

[8] Burnetas, A. N. (2013). Customer equilibrium and optimal strategies in markovian queues in series. *Annals of Operations Research*, 208(1):515–529.

[9] Chen, H. and Wan, Y.-W. (2003). Price competition of make-to-order firms. *IIE Transactions*, 35(9):817–832.

[10] Chen, H. and Wan, Y.-w. (2005). Capacity competition of make-to-order firms. *Operations Research Letters*, 33(2):187–194.

[11] Cheng, H. K., Demirkan, H., and Koehler, G. J. (2003). Price and capacity competition of application services duopoly. *Information Systems and e-Business Management*, 1(3):305–329.

[12] Clarey, A., Allen, M., Brace-McDonnell, S., and Cooke, M. (2014). Ambulance handovers: can a dedicated ed nurse solve the delay in ambulance turnaround times? *Emergency Medicine Journal*, 31(5):419–420.

[13] Clarke, O. (2021). A&e queues mean wales ambulances can't take 999 calls. *BBC Wales*.

[14] Crouch, J. (2021). Thousands of hours lost as ambulance idle outside hospitals in derbyshire. *Staffordshire Live*.

[15] D'Auria, B. and Kanta, S. (2015). Pure threshold strategies for a two-node tandem network under partial information. *Operations Research Letters*, 43(5):467–470.

[16] Day, S. (2021). Woman, 85, has 'appalling' three hour wait for ambulance. *East Anglia*.

[17] Deo, S. and Gurvich, I. (2011). Centralized vs. decentralized ambulance diversion: A network perspective. *Management Science*, 57(7):1300–1319.

[18] Fan, M., Kumar, S., and Whinston, A. B. (2009). Short-term and long-term competition between providers of shrink-wrap software and software as a service. *European Journal of Operational Research*, 196(2):661–671.

[19] Favaro, S. and Walker, S. G. (2010). On the distribution of sums of independent exponential random variables via wilks' integral representation. *Acta applicandae mathematicae*, 109(3):1035–1042.

[20] Fudenberg, D., Drew, F., Levine, D. K., and Levine, D. K. (1998). *The theory of learning in games*, volume 2. MIT press.

[21] Halliwell, A. (2021). Cqc covid insight: Winter pressures on emergency care. *Practice Management*, 31(4):28–30.

[22] Kemeny, J. G. and Snell, J. L. (1976). *Markov chains*, volume 6. Springer-Verlag, New York.

[23] Khaled, S., Kadri, T., and Kadry, S. (2013). Hypoexponential distribution with different parameters. *Journal of Applied Mathematics*, 4:624–631.

[24] Knapper, Dave & Dresch, M. (2021). Ambulances queue 10 in a row outside hospital buckling under 'significant' pressure. *Mirror*.

[25] Knight, V., Komenda, I., and Griffiths, J. (2017). Measuring the price of anarchy in critical care unit interactions. *Journal of the Operational Research Society*, 68(6):630–642.

[26] Knight, V. A. and Harper, P. R. (2013). Selfish routing in public services. *European Journal of Operational Research*, 230(1):122–132.

[27] Koutsoupias, E. and Papadimitriou, C. (1999). Worst-case equilibria. In *Annual Symposium on Theoretical Aspects of Computer Science*, pages 404–413. Springer.

[28] Kreps, D. M. (1989). Nash equilibrium. In *Game Theory*, pages 167–177. Springer.

[29] Legros, B. and Jouini, O. (2015). A linear algebraic approach for the computation of sums of erlang random variables. *Applied Mathematical Modelling*, 39(16):4971–4977.

[30] Lemke, C. E. and Howson, Jr., J. T. (1964). Equilibrium points of bimatrix games. *Journal of the Society for Industrial and Applied Mathematics*, 12(2):413–423.

[31] Lemmer, R. (2021). Portsmouth's queen alexandra hospital patients see more than 250 hour-long ambulance handover delays. *The News*.

[32] Mahase, E. (2020). Covid-19: Hospitals in crisis as ambulances queue and staff are asked to cancel leave. *The BMJ*.

[33] Maschler, M., Solan, E., and Zamir, E. (2013). *Game Theory*. Cambridge University Press.

[34] McAdams, B. (2021). Caldicot girl waited nine hours for ambulance after fall. *South Wales Argus*.

[35] Myerson, R. B. (1978). Refinements of the nash equilibrium concept. *International journal of game theory*, 7(2):73–80.

[36] Panayides, M. (2021). 11michalis11/ambulancedecisiongame: v0.0.2.

[37] Roughgarden, T. (2005). *Selfish routing and the price of anarchy*. MIT press.

[38] Sadat, S., Abouee-Mehrizi, H., and Carter, M. W. (2015). Can hospitals compete on quality? *Health care management science*, 18(3):376–388.

[39] Shone, R., Knight, V. A., and Williams, J. E. (2013). Comparisons between observable and unobservable m/m/1 queues with respect to optimal customer behavior. *European Journal of Operational Research*, 227(1):133–141.

[40] Smith, J. M. (1986). Evolutionary game theory. *Physica D: Nonlinear Phenomena*, 22(1-3):43–49.

[41] Stewart, W. J. (2019). *Probability, Markov Chains, Queues, and Simulation*. Princeton University Press.

[42] Sun, W., Li, S.-y., Tian, N.-s., and Zhang, H.-k. (2009). Equilibrium analysis in batch-arrival queues with complementary services. *Applied Mathematical Modelling*, 33(1):224–241.

[43] The Ciw library developers (2020). Ciwpython/ciw: v2.1.3.

[44] The Nashpy project developers (2021). Nashpy: v0.0.25.

[45] Thomas, J. (2021). Ambulances queue at hereford hospital as nhs pressure mounts. *Hereford Times*.

[46] Veltman, A. and Hassin, R. (2005). Equilibrium in queueing systems with complementary products. *Queueing Systems*, 50(2):325–342.

Appendices

A. Discrete event simulation

For the purposes of this study, a discrete event simulation (DES) model was constructed to support the Markov chain version described in section 3. The queueing model was built in python using the Ciw library [43].

The constructed model simulates a queueing system with two waiting spaces and two types of individuals. The expected behaviour of the nodes in Ciw have been modified such that individuals moving from waiting zone 2 into waiting zone 1 get blocked if there are more than T individuals in waiting zone 1.

The same performance measures described in sections 3.1, 3.2 and 3.3 can also be calculated using the DES model. The simulation can be ran a number of times to eliminate stochasticity and the outcomes of the two methods can be directly comparable.

A.1. Tutorial: building the DES model

The DES model is constructed in a generic way that so that it can be used for any queueing system with two waiting spaces and two types of individuals. For instance, consider a queueing system with the following parameters, as described in section 3:

- $\lambda_1 = 2$
- $\mu = 1$
- $T = 10$
- $M = 10$
- $\lambda_2 = 3$
- $C = 6$
- $N = 20$

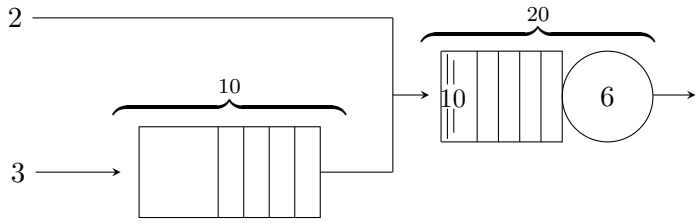

Figure 13: A diagrammatic representation of the queueing model example

This model will be studied by using `ambulance_game` [36]. Install the created library in your python environment, by running the following command in the command line:

```
$ python -m pip install ambulance_game
```

Having installed the package, the following code can be used to simulate the queueing system defined earlier and get all the data records for a single run.

```
>>> import ambulance_game as abg
>>> Simulation = abg.simulation.simulate_model(
```

```
...     lambda_1=3,
...     lambda_2=2,
...     mu=1,
...     num_of_servers=6,
...     threshold=10,
...     system_capacity=20,
...     buffer_capacity=10,
...     seed_num=0,
... )
>>> all_records = Simulation.get_all_records()
>>> all_records[3]
Record(id_number=1, customer_class=0, node=2, arrival_date=0.4728763843239206,
 waiting_time=0.0, service_start_date=0.4728763843239206, service_time=0
.5457131455415929, service_end_date=1.0185895298655134, time_blocked=0.0,
exit_date=1.0185895298655134, destination=-1, queue_size_at_arrival=0,
queue_size_at_departure=4)
```

The above block code outputs the fourth individual record from the simulation object. The simulation object can be used to view every event that occurred in the simulation The data records can then be used to get overall performance measures about the constructed queueing model. The overall waiting time that individuals wait in waiting zone 1 can be acquired by running:

```
>>> import numpy as np
>>> mean_wait = np.mean(
...     [record.waiting_time for record in all_records if record.node == 2]
... )
>>> mean_wait
0.3608431242229529
```

This value is the average waiting time of all the customers in the system for a single run. By nature, discrete event simulation can output different results for different runs of the same set of parameters. This stochasticity can be reduced by running the simulation multiple times and then getting the mean waiting time from all the runs.

```
>>> all_simulations = abg.simulation.get_multiple_runs_results(
...     lambda_1=3,
...     lambda_2=2,
...     mu=1,
...     num_of_servers=6,
...     threshold=10,
...     system_capacity=20,
...     buffer_capacity=10,
...     seed_num=0,
...     num_of_trials=10,
... )
>>> mean_wait = np.mean(
...     [np.mean(w.waiting_times) for w in all_simulations]
... )
```

```
>>> mean_wait
0.3566390561071839
```

A.2. How-to guide

A.2.1. How to install:

The package can be installed by either running:

```
$ python -m pip install ambulance_game
```

in the command line or via the instructions provided in the GitHub repository.

A.2.2. How to simulate the model:

The required arguments that need to be passed to the `simulate_model()` function are the following:

- lambda_1 (λ_1): The arrival rate of class 1 individuals.
- lambda_2 (λ_2): The arrival rate of class 2 individuals.
- mu (μ): The service rate of the servers.
- num_of_servers (C): The number of servers in the system.
- threshold (T): The threshold that indicates when to start blocking class 2 individuals.

To get the simulation object with all the data records, the following code can be used:

```
>>> import ambulance_game as abg
>>> Simulation = abg.simulation.simulate_model(
...     lambda_1=3,
...     lambda_2=2,
...     mu=1,
...     num_of_servers=6,
...     threshold=10,
...     seed_num=0,
... )
>>> Simulation.get_all_records()[4]
Record(id_number=2, customer_class=0, node=2, arrival_date=0.5727571550618586,
 waiting_time=0.0, service_start_date=0.5727571550618586, service_time=0
.7159547497671506, service_end_date=1.2887119048290092, time_blocked=0.0,
exit_date=1.2887119048290092, destination=-1, queue_size_at_arrival=1,
queue_size_at_departure=3)
```

Additional arguments that can be passed to the function are:

- *system_capacity* (N): The maximum number of individuals in waiting zone 1.
- *buffer_capacity* M: The maximum number of individuals in waiting zone 2.

- *seed_num*: The seed number for the random number generator.
- *runtime*: How long to run the simulation for.

A.2.3. How to get the performance measures for a single run:

From a single run of the simulation the data records can be used to get the average for certain performance measures. The following code can be used to get the mean waiting time, blocking time, service time and the proportion of individuals within target.

```
>>> records = Simulation.get_all_records()
>>> mean_wait = np.mean(
...     [w.waiting_time for w in records]
... )
>>> mean_wait
0.23845862661827116

>>> mean_block = np.mean(
...     [b.time_blocked for b in records]
... )
>>> mean_block
0.08501727452006658

>>> mean_service = np.mean(
...     [s.service_time for s in records]
... )
>>> mean_service
0.7102610863960119

>>> target = 1
>>> proportion_within_target = np.mean(
...     [w.waiting_time <= target for w in records]
... )
>>> proportion_within_target
0.9387470071827614
```

A.2.4. How to get the average performance measures:

To reduce the effects of stochasticity in the simulation, the simulation can be run numerous times and get the average performance measures out of all the runs.

```
>>> import numpy as np
>>> import ambulance_game as abg
>>> all_simulations = abg.simulation.get_multiple_runs_results(
...     lambda_1=3,
...     lambda_2=2,
...     mu=1,
...     num_of_servers=6,
```

```
...     threshold=10,
...     system_capacity=20,
...     buffer_capacity=10,
...     seed_num=0,
...     runtime=2000,
...     num_of_trials=10,
...     target=1,
... )
>>> mean_wait = np.mean([
...     np.mean(w.waiting_times) for w in all_simulations
... ])
>>> mean_wait
0.35585979549204577

>>> mean_service = np.mean([
...     np.mean(s.service_times) for s in all_simulations
... ])
>>> mean_service
1.002184850213415

>>> mean_block = np.mean([
...     np.mean(b.blocking_times) for b in all_simulations
... ])
>>> mean_block
0.3976966024549059

>>> np.mean([p.proportion_within_target for p in all_simulations])
0.45785790578122043
```

A.2.5. How to get the steady state probabilities vector π:

To get the steady state probabilities of the model based on the simulation the following code can be used:

```
>>> import numpy as np
>>> import ambulance_game as abg
>>> simulation_object = abg.simulation.simulate_model(
...     lambda_1=1,
...     lambda_2=2,
...     mu=2,
...     num_of_servers=2,
...     threshold=3,
...     system_capacity=4,
...     buffer_capacity=2,
...     seed_num=0,
...     runtime=2000,
... )
>>> probs = abg.simulation.get_simulated_state_probabilities(
```

```
...     simulation_object=simulation_object,
... )
>>> np.round(probs, decimals=3)
array([[0.166, 0.266, 0.192, 0.147, 0.025],
       [  nan,   nan,   nan, 0.094, 0.024],
       [  nan,   nan,   nan, 0.058, 0.027]])

>>> total = np.nansum(probs)
>>> np.round(total, decimals=5)
1.0
```

Similarly to get the average steady state probabilities over multiple runs, one can use:

```
>>> import numpy as np
>>> import ambulance_game as abg
>>> probs = abg.simulation.get_average_simulated_state_probabilities(
...     lambda_1=1,
...     lambda_2=2,
...     mu=2,
...     num_of_servers=2,
...     threshold=3,
...     system_capacity=4,
...     buffer_capacity=2,
...     seed_num=0,
...     runtime=2000,
...     num_of_trials=10,
... )
>>> np.round(probs, decimals=3)
array([[0.18 , 0.267, 0.197, 0.144, 0.024],
       [  nan,   nan,   nan, 0.085, 0.022],
       [  nan,   nan,   nan, 0.054, 0.026]])

>>> total = np.nansum(probs)
>>> np.round(total, decimals=5)
1.0
```

A.2.6. How to get the optimal distribution of class 2 individuals among 2 queueing models:

In the scenario where there are two queueing models and a service that distributes individuals to the models, (i.e. the scenario described in this paper) the simulation can be used to decide what proportion of individuals to send to each the model. Note that the output of the function shows the value of p_1, the proportion of class 2 individuals to be sent to queueing model 1.

```
>>> import ambulance_game as abg
>>> abg.simulation.calculate_class_2_individuals_best_response(
...     lambda_2=4,
...     lambda_1_1=1,
...     lambda_1_2=1,
```

```
...     mu_1=4,
...     mu_2=3,
...     num_of_servers_1=2,
...     num_of_servers_2=2,
...     threshold_1=3,
...     threshold_2=3,
...     system_capacity_1=3,
...     system_capacity_2=3,
...     buffer_capacity_1=2,
...     buffer_capacity_2=2,
...     seed_num_1=0,
...     seed_num_2=0,
...     num_of_trials=3,
...     warm_up_time=100,
...     runtime=1000,
... )
0.6343260586929469
```

A.3. Reference

The primary tool that was used in the construction of the discrete event simulation model was the python library `ciw`. See Ciw's documentation for a more detailed explanation of how it works and what are its capabilities [43].

Find below a detailed list of the functions that were created for the simulation model:

- `build_model`: *Builds a ciw object that represents a model of a queueing network with two waiting spaces.*
- `build_custom_node`: *Build a custom node to replace the default ciw.Node. Inherits from the original ciw.Node class and replaces two methods.*
- `simulate_model`: *Simulate the model by using the custom node and returning the simulation object*
- `extract_times_from_records`: *Get the required times (waiting, service, blocking) out of ciw's records where all individuals are treated the same way*
- `extract_times_from_individuals`: *Extract waiting times and service times for all individuals and proceed to extract blocking times for only class 2 individuals*
- `get_list_of_results`: *Modify the outputs so that they are returned in a list format where it is sometimes easier to be used by other functions.*
- `get_multiple_runs_results`: *Get the waiting times, service times and blocking times for multiple runs of the simulation*

- `extract_total_individuals_and_the_ones_within_target_for_both_classes`: *Extract the total number of individuals that pass through the model and the total number of individuals that exit the model within the given target.*
- `get_mean_proportion_of_individuals_within_target_for_multiple_runs`: *Get the average proportion of individuals within target by running the simulation multiple times*
- `get_simulated_state_probabilities`: *Calculates the vector π in a dictionary format or an array format*
- `get_average_simulated_state_probabilities`: *This function runs get_simulated_state_probabilities for multiple iterations to eliminate any stochasticity from the results*
- `get_mean_blocking_difference_between_two_systems`: *Given a predefined proportion of class's 2 arrival rate calculate the mean difference between blocking times of two systems with a given set of parameters*
- `calculate_class_2_individuals_best_response` *Obtains the optimal distribution of class 2 individuals such that the blocking times in the two systems are identical and thus minimised*

A.4. Explanation

Based on Ciw's functionality the simulation model stores all data records in a Record object. For every event that takes place a record is created with all the relevant information. For this specific library, the records that are stored, along with the range of values that they can take are as follows:

- **id_number** $\in R$.
- **customer_class** $= 0$
- **node** $= \{0, 1, 2, -1\}$
- **arrival_date** $\in R^+$
- **waiting_time** $\in R^+$
- **service_start_date** $\in R^+$
- **service_time** $\in R^+$
- **service_end_time** $\in R^+$
- **time_blocked** $\in R^+$
- **exit_date** $\in R^+$
- **destination** $= \{1, 2, -1\}$
- **queue_size_at_arrival** $\in N$
- **queue_size_at_departure** $\in N$

B. Mean waiting time

The recursive formula described here is the origin of the closed-form formula described in section 3.1.

To calculate the mean waiting time one must first identify the set of states (u, v) where a wait will occur. For this particular Markov chain, this points to all states that satisfy $v > C$ i.e. all states where the number of individuals in the service centre exceed the number of servers. The set of such states is defined as *waiting states* and can be denoted as a subset of all the states, where:

$$S_w = \{(u, v) \in S \mid v > C\} \tag{37}$$

Additionally, there are certain states in the model where arrivals cannot occur. A type 1 individual cannot arrive whenever the model is at any state (u, N) for all u where N is the system capacity. Equivalently, a type 2 individual cannot arrive in the model when the model is at any state (M, v) for all $v \geq T$. Therefore the set of all such states that an arrival may occur are defined as *accepting states* and are denoted as:

$$S_A^{(1)} = \{(u, v) \in S \mid v < N\} \tag{7 revisited}$$

$$S_A^{(2)} = \begin{cases} \{(u, v) \in S \mid u < M\} & \textbf{if } T \leq N \\ \{(u, v) \in S \mid v < N\} & \textbf{otherwise} \end{cases} \tag{8 revisited}$$

Moreover, another element that needs to be considered is the expected waiting time In order to do so a variation of the Markov model has to be considered where when the individual arrives at any of the states of the model no more arrivals can occur after that.

Thus, one may acquire the desired time by calculating the inverse of the sum of the out-flow rate of that state. Since arrivals are ignored though the only way to exit the state will only be via a service. In essence this notion can be expressed as:

$$c^{(1)}(u, v) = \begin{cases} 0, & \textbf{if } u > 0 \textbf{ and } v = T \\ \frac{1}{\min(v, C)\mu}, & \textbf{otherwise} \end{cases} \tag{38}$$

Now, like in the type 1 individuals case, the sojourn time is needed. For type 2 individuals whenever individuals are at any row apart from the first one they automatically get a wait time of 0 since they are essentially blocked.

$$c^{(2)}(u, v) = \begin{cases} 0, & \textbf{if } u > 0 \\ \frac{1}{\min(v, C)\mu}, & \textbf{otherwise} \end{cases} \tag{39}$$

Note that whenever any type 1 individual is at a state (u, v) where $u > 0$ and $v = T$ (i.e. all states $(1, T), (2, T) \ldots, (M, T)$) the sojourn time is set to 0. This is done to capture the trip thorough the Markov chain from the perspective of individuals. Meaning that they will visit all states of the threshold column but only the one in the first row will return a non-zero sojourn time.

Thus, the expected waiting time of type 1 and type 2 individuals when upon arriving at state (u, v) can be given by the following recursive formulas:

$$w^{(1)}(u,v) = \begin{cases} 0, & \textbf{if } (u,v) \notin S_w \\ c^{(1)}(u,v) + w^{(1)}(u-1,v), & \textbf{if } u > 0 \textbf{ and } v = T \\ c^{(1)}(u,v) + w^{(1)}(u,v-1), & \textbf{otherwise} \end{cases} \tag{40}$$

$$w^{(2)}(u,v) = \begin{cases} 0, & \textbf{if } (u,v) \notin S_w \\ c^{(2)}(u,v) + w^{(2)}(u-1,v), & \textbf{if } u > 0 \textbf{ and } v = T \\ c^{(2)}(u,v) + w^{(2)}(u,v-1), & \textbf{otherwise} \end{cases} \tag{41}$$

Finally, the mean waiting time can be calculated by summing over all expected waiting times of accepting states multiplied by the probability of being at that state and dividing by the probability of being in any accepting state. Note here that $\mathcal{A}_i$ is defined in section 3.2 by equations 11 and 12.

$$W^{(1)} = \frac{\sum_{(u,v)\in S_A^{(1)}} w^{(1)}(\mathcal{A}_1(u,v))\pi_{(u,v)}}{\sum_{(u,v)\in S_A^{(1)}} \pi_{(u,v)}} \tag{42}$$

$$W^{(2)} = \frac{\sum_{(u,v)\in S_A^{(2)}} w^{(2)}(\mathcal{A}_2(u,v))\pi_{(u,v)}}{\sum_{(u,v)\in S_A^{(2)}} \pi_{(u,v)}} \tag{43}$$

C. Mean blocking time

The set of states where individuals can be blocked is defined as:

$$S_b = \{(u,v) \in S \mid u > 0\} \tag{14 revisited}$$

The mean sojourn time for each state is given by the inverse of the out-flow of that state [41]. However, whenever a type 2 individual arrives at the system, no subsequent arrival of another type 2 individual can affect its pathway or total time in the system. Therefore, looking at the mean time in the system from the perspective of an individual of the second type, all such type 2 arrivals need to be ignored. Note here that this is not the case for individuals of the first type. Whenever a type 2 individual is blocked and a type 1 individual arrives the type 2 individuals will stay blocked for some additional amount of time. Thus, the mean time that a type 2 individual spends at each state is given by:

$$c(u,v) = \begin{cases} \frac{1}{\min(v,C)\mu}, & \text{if } v = N \\ \frac{1}{\lambda_1 + \min(v,C)\mu}, & \text{otherwise} \end{cases} \tag{15 revisited}$$

In equation (15), both service completions and type 1 arrivals are considered. Thus, from a blocked individual's perspective whenever the system moves from one state (u, v) to another state it can either:

- be because of a service being completed: we will denote the probability of this happening by $p_s(u, v)$.
- be because of an arrival of an individual of type 1: denoting such probability by $p_a(u, v)$.

The probabilities are given by:

$$p_s(u,v) = \frac{\min(v,C)\mu}{\lambda_1 + \min(v,C)\mu}, \qquad p_a(u,v) = \frac{\lambda_1}{\lambda_1 + \min(v,C)\mu} \tag{16 revisited}$$

Having defined $c(u, v)$ and S_b a formula for the blocking time that is expected to occur at each state can be given by:

$$b(u,v) = \begin{cases} 0, & \textbf{if } (u,v) \notin S_b \\ c(u,v) + b(u-1,v), & \textbf{if } v = N = T \\ c(u,v) + b(u,v-1), & \textbf{if } v = N \neq T \\ c(u,v) + p_s(u,v)b(u-1,v) + p_a(u,v)b(u,v+1), & \textbf{if } u > 0 \textbf{ and } \\ & \quad v = T \\ c(u,v) + p_s(u,v)b(u,v-1) + p_a(u,v)b(u,v+1), & \textbf{otherwise} \end{cases} \tag{13 revisited}$$

A direct approach will be used to solve this equation here. By enumerating all equations of (13) for all states (u, v) that belong in S_b a system of linear equations arises where the unknown variables are all the $b(u, v)$ terms. Note here that these equations correspond to all blocking states as defined in (14). Equations that correspond to non-blocking states have a value of 0 as defined in (13) The general form of the equation in terms of C, T, N and M is given by:

$$
\begin{aligned}
b(1,T) &= c(1,T) + p_a b(1,T+1) && (44)\\
b(1,T+1) &= c(1,T+1) + p_s b(1,T) + p_a b(1,T+1) && (45)\\
b(1,T+2) &= c(1,T+2) + p_s b(1,T+1) + p_a b(1,T+3) && (46)\\
&\vdots\\
b(1,N) &= c(1,N) + b(1,N-1) && (47)\\
b(2,T) &= c(2,T) + p_s b(1,T) + p_a b(2,T+1) && (48)\\
b(2,T+1) &= c(2,T+1) + p_s b(2,T) + p_a b(2,T+2) && (49)\\
&\vdots\\
b(M-1,N) &= c(M,N-1) + b(M,N-1) && (50)\\
b(M,T) &= c(T,N) + p_s b(T-1,N) + p_a b(T,N+1) && (51)\\
&\vdots\\
b(M,N) &= c(M,N) + b(M,N-1) && (52)
\end{aligned}
$$

The equivalent matrix notation of the linear system of equations (44) - (52) is given by $Zx = y$, where:

$$
Z = \begin{pmatrix}
-1 & p_a & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0\\
p_s & -1 & p_a & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0\\
0 & p_s & -1 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 0 & 0\\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\
0 & 0 & 0 & \dots & 1 & -1 & 0 & 0 & 0 & \dots & 0 & 0\\
p_s & 0 & 0 & \dots & 0 & 0 & -1 & p_a & 0 & \dots & 0 & 0\\
0 & 0 & 0 & \dots & 0 & 0 & p_s & -1 & p_a & \dots & 0 & 0\\
\vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots\\
0 & 0 & 0 & \dots & 0 & 0 & 0 & 0 & 0 & \dots & 1 & -1
\end{pmatrix}, x = \begin{pmatrix}
b(1,T)\\ b(1,T+1)\\ b(1,T+2)\\ \vdots\\ b(1,N)\\ b(2,T)\\ b(2,T+1)\\ \vdots\\ b(M,N)
\end{pmatrix}, y = \begin{pmatrix}
-c(1,T)\\ -c(1,T+1)\\ -c(1,T+2)\\ \vdots\\ -c(1,N)\\ -c(2,T)\\ -c(2,T+1)\\ \vdots\\ -c(M,N)
\end{pmatrix} \qquad \text{(18 revisited)}
$$

The elements of the matrix Z can be acquired using Z_{ij} defined in equation (17) where i and j are states $(u_i, v_i), (u_j, v_j) \in S_b$ (14).

$$
Z_{ij} = \begin{cases}
p_a, & \textbf{if } j = i+1 \textbf{ and } v_i \neq N\\
p_s, & \textbf{if } j = i-1 \textbf{ and } v_i \neq N, v_i \neq T\\
 & \textbf{or } j = i - N + T \textbf{ and } u_i \geq 2,\ v_i = T\\
1, & \textbf{if } j = i-1 \textbf{ and } v_i = N\\
-1, & \textbf{if } i = j\\
0, & \textbf{otherwise}
\end{cases} \qquad \text{(17 revisited)}
$$

Thus, having calculated the mean blocking time for all blocking states $b(u, v)$, they can be

combined together in a formula. Using the arriving states $\mathcal{A}_2$ defined in section 3.2 by equation 12 the resultant formula for the mean blocking time is given by:

$$B = \frac{\sum_{(u,v)\in S_A} \pi_{(u,v)}\ b(\mathcal{A}_2(u,v))}{\sum_{(u,v)\in S_A} \pi_{(u,v)}} \qquad \text{(10 revisited)}$$

To illustrate how the described formula works consider a Markov model where $C = 2, T = 2, N = 4, M = 2$ (figure 14). The equations that correspond to such a model are shown in (53)-(58) and their equivalent matrix notation form is shown in (59).

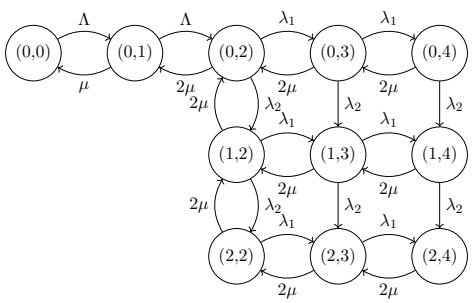

Figure 14: Example of Markov chain with $C = 2, T = 2, N = 4, M = 2$

$$b(1,2) = c(1,2) + p_a b(1,3) \qquad (53)$$

$$b(1,3) = c(1,3) + p_s b(1,2) + p_a b(1,4) \qquad (54)$$

$$b(1,4) = c(1,4) + b(1,3) \qquad (55)$$

$$b(2,2) = c(2,2) + p_s b(1,2) + p_a b(2,3) \qquad (56)$$

$$b(2,3) = c(2,3) + p_s b(2,2) + p_a b(1,4) \qquad (57)$$

$$b(2,4) = c(2,4) + b(2,3) \qquad (58)$$

$$Z = \begin{pmatrix} -1 & p_a & 0 & 0 & 0 & 0 \\ p_s & -1 & p_a & 0 & 0 & 0 \\ 0 & 1 & -1 & 0 & 0 & 0 \\ p_s & 0 & 0 & -1 & p_a & 0 \\ 0 & 0 & 0 & p_s & -1 & p_a \\ 0 & 0 & 0 & 0 & 1 & -1 \end{pmatrix}, x = \begin{pmatrix} b(1,2) \\ b(1,3) \\ b(1,4) \\ b(2,2) \\ b(2,3) \\ b(2,4) \end{pmatrix}, y = \begin{pmatrix} -c(1,2) \\ -c(1,3) \\ -c(1,4) \\ -c(2,2) \\ -c(2,3) \\ -c(2,4) \end{pmatrix} \qquad (59)$$

D. Mean proportion of individuals within target

In order to consider such measure though one would need to obtain the distribution of time in the system for all individuals. The complexity of such task is caused by the fact that different individuals arrive at different states of the Markov model. Consider the case when an arrival occurs when the model is at a specific state.

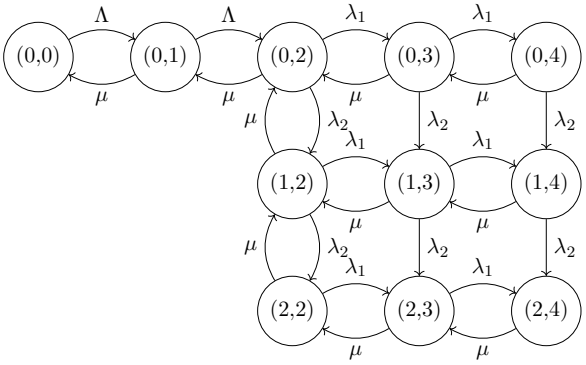

Figure 15: Example Markov model $C = 1, T = 2, N = 4, M = 2$

Time distribution at specific state (1 server):. Consider the Markov model of Figure 15 with one server and a threshold of two individuals. Assume that an individual of the first type arrives when the model is at state $(0, 3)$, thus forcing the model to move to state $(0, 4)$. The distribution of the time needed for the specified individual to exit the system from state $(0, 4)$ is given by the sum of exponentially distributed random variables with the same parameter μ. The sum of such random variables forms an Erlang distribution which is defined by the number of random variables that are added and their exponential parameter. Note here that these random variables represent the individual's pathway from the perspective of the individual. Thus, X_i represents the time that it takes to move from the i^{th} position of the queue to the $(i-1)^{\text{th}}$ position (i.e. for someone in front of them to finish their service) and X_0 is the time it takes to move from having a service to exiting the system.

$$\begin{aligned}
(0,4) \Rightarrow \quad & X_3 \sim Exp(\mu) \\
(0,3) \Rightarrow \quad & X_2 \sim Exp(\mu) \\
(0,2) \Rightarrow \quad & X_1 \sim Exp(\mu) \\
(0,1) \Rightarrow \quad & X_0 \sim Exp(\mu) \\
S = X_3 + X_2 + & X_1 + X_0 = Erlang(4, \mu)
\end{aligned} \tag{60}$$

Thus, the waiting and service time of an individual in the model of Figure 15 can be captured by an erlang distributed random variable. The general CDF of the erlang distribution $Erlang(k, \mu)$ is given by:

$$P(S < t) = 1 - \sum_{i=0}^{k-1} \frac{1}{i!} e^{-\mu t} (\mu t)^i \tag{61}$$

Unfortunately, the erlang distribution can only be used for the sum of identically distributed random variables from the exponential distribution. Therefore, this approach cannot be used when

one of the random variables has a different parameter than the others. In fact the only case where it can be used is only when the number of servers are $C = 1$, or when an individual arrives and goes straight to service (i.e. when there is no other individual waiting and there is an empty server).

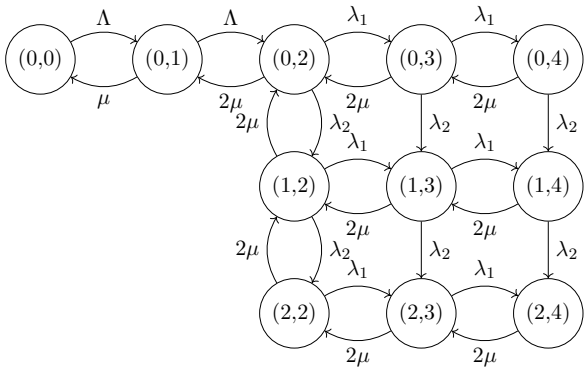

Figure 16: Example Markov model $C = 2, T = 2, N = 4, M = 2$

Time distribution at a state (multiple servers):. Figure 16 represents the same Markov model as Figure 15 with the only exception that there are 2 servers here. By applying the same logic, assuming that an individual arrives at state $(0, 4)$, the sum of the following random variables arises.

$$\begin{aligned} (0,4) \Rightarrow \quad & X_2 \sim Exp(2\mu) \\ (0,3) \Rightarrow \quad & X_1 \sim Exp(2\mu) \\ (0,2) \Rightarrow \quad & X_0 \sim Exp(\mu) \end{aligned} \tag{62}$$

Since these exponentially distributed random variables do not share the same parameter, an erlang distribution cannot be used. In fact, the problem can now be viewed either as the sum of exponentially distributed random variables with different parameters or as the sum of erlang distributed random variables. The sum of erlang distributed random variables is said to follow the hypoexponential distribution. The hypoexponential distribution is defined with two vectors of size equal to the number of Erlang random variables [4], [23]. The vector $\vec{r}$ contains all the k-values of the erlang distributions and $\vec{\lambda}$ is a vector of the distinct parameters as illustrated in equation (63).

$$\left.\begin{array}{l} Erlang(k_1, \lambda_1) \\ Erlang(k_2, \lambda_2) \\ \quad \vdots \\ Erlang(k_n, \lambda_n) \end{array}\right\} Hypo(\underbrace{(k_1, k_2, \ldots k_n)}_{\vec{k}}, \underbrace{(\lambda_1, \lambda_2, \ldots \lambda_n)}_{\vec{\lambda}}) \tag{63}$$

Equivalently, for this particular example:

$$\left.\begin{array}{l} X_2 \sim Exp(2\mu) \\ X_1 \sim Exp(2\mu) \end{array}\right\} X_1+X_2=S_1 \sim Erlang(2,2\mu) \left.\begin{array}{l} \\ \\ X_0 \sim Exp(\mu) \Rightarrow \end{array}\right\} S_1+S_2=H \sim Hypo((2,1),(2\mu,\mu)) \quad X_0=S_2 \sim Erlang(1,\mu) \tag{64}$$

Therefore, the CDF of this distribution can be used to get the probability of the time in spent in the system being less than a given target. The general CDF of the hypoexponential distribution $Hypo(\vec{r},\vec{\lambda})$, is given by the following expression [19]:

$$\begin{aligned} & P(H<t) = 1 - \left(\prod_{j=1}^{|\vec{r}|} \lambda_j^{r_j}\right) \sum_{k=1}^{|\vec{r}|} \sum_{l=1}^{r_k} \frac{\Psi_{k,l}(-\lambda_k) t^{r_k-l} e^{-\lambda_k t}}{(r_k-l)!(l-1)!} \\ & \textbf{where} \qquad \Psi_{k,l}(t) = -\frac{\partial^{l-1}}{\partial t^{l-1}} \left(\prod_{j=0, j\neq k}^{|\vec{r}|} (\lambda_j + t)^{-r_j}\right) \\ & \textbf{and} \qquad \lambda_0 = 0, r_0 = 1 \end{aligned} \tag{65}$$

The computation of the derivative makes equation (65) computationally expensive. In [29] an alternative linear version of that CDF is explored via matrix analysis, and is given by the following formula:

$$\begin{aligned} F(x) =& 1 - \sum_{k=1}^{n} \sum_{l=0}^{k-1} (-1)^{k-1} \binom{n}{k} \binom{k-1}{l} \sum_{j=1}^{n} \sum_{s=1}^{j-1} e^{-x\lambda_s} \prod_{l=1}^{s-1} \left(\frac{\lambda_l}{\lambda_l - \lambda_s}\right)^{k_s} \\ & \times \sum_{s<a_1<\cdots<a_{l-1}<j} \left(\frac{\lambda_s}{\lambda_s - \lambda_{a_1}}\right)^{k_s} \prod_{m=s+1}^{a_1-1} \left(\frac{\lambda_m}{\lambda_m - \lambda_{a_1}}\right)^{k_m} \\ & \times \prod_{n=a_1}^{a_2-1} \left(\frac{\lambda_n}{\lambda_n - \lambda_{a_2}}\right)^{k_n} \cdots \prod_{r=a_l-1}^{j-1} \left(\frac{\lambda_r}{\lambda_r - \lambda_{a_j}}\right)^{k_r} \sum_{q=0}^{k_s-1} \frac{((\lambda_s - \lambda_{a_1})x)^q}{q!}, \\ & \text{for } x \geq 0 \end{aligned} \tag{66}$$

Specific CDF of hypoexponential distribution. Equations (65) and (66) refers to the general CDF of the hypoexponential distribution where the size of the vector parameters can be of any size [19]. In the Markov chain models described in Figures 15 and 16 the parameter vectors of the hypoexponential distribution are of size two, and in fact, for any possible version of the investigated Markov chain model the vectors can only be of size two. This is true since for any dimensions of this Markov chain model there will always be at most two distinct exponential parameters; the parameter for finishing a service (μ) and the parameter for moving forward in the queue ($C\mu$). For the case of $C = 1$ the hypoexponential distribution will not be used as this is equivalent to an

erlang distribution. Therefore, by fixing the sizes of $\vec{r}$ and $\vec{\lambda}$ to 2, the following specific expression for the CDF of the hypoexponential distribution arises, where the derivative is removed:

$$
\begin{aligned}
P(H < t) &= 1 - \left(\prod_{j=1}^{|\vec{r}|} \lambda_j^{r_j} \right) \sum_{k=1}^{|\vec{r}|} \sum_{l=1}^{r_k} \frac{\Psi_{k,l}(-\lambda_k) t^{r_k - l} e^{-\lambda_k t}}{(r_k - l)!(l-1)!} \\
\textbf{where} \qquad \Psi_{k,l}(t) &= \begin{cases} \frac{(-1)^l (l-1)!}{\lambda_2} \left[\frac{1}{t^l} - \frac{1}{(t+\lambda_2)^l} \right], & k = 1 \\ -\frac{1}{t(t+\lambda_1)^{r_1}}, & k = 2 \end{cases} \\
\textbf{and} \qquad & \lambda_0 = 0, r_0 = 1
\end{aligned} \tag{67}
$$

Note here that the only difference between equations (65) and (67) is the Ψ function. The next part proves that the expression for Ψ can be simplified for the cases of $k = 1, 2$. Equation (68) shows the expression to be proved.

$$
\Psi_{(k,l)}(t) = -\frac{\partial^{l-1}}{\partial t^{l-1}} \left(\prod_{j=0, j \neq k}^{|\vec{r}|} (\lambda_j + t)^{-r_j} \right) = \begin{cases} \frac{(-1)^l (l-1)!}{\lambda_2} \left[\frac{1}{t^l} - \frac{1}{(t+\lambda_2)^l} \right], & k = 1 \\ -\frac{1}{t(t+\lambda_1)^{r_1}}, & k = 2 \end{cases} \tag{68}
$$

Proof of equation (68). This section aims to show that there exists a simplified version of equation (65) that is specific to the proposed Markov model. Function Ψ is defined using the parameter t and the variables k and l. Given the Markov model, the range of values that k and l can take can be bounded. First, from the range of the double summation in equation (65), it can be seen that $k = 1, 2, \ldots, |\vec{r}|$. Now, $|\vec{r}|$ represents the size of the parameter vectors that, for the Markov model, will always be 2. That is because, for all the exponentially distributed random variables that are added together to form the new distribution, there only two distinct parameters, thus forming two erlang distributions. Therefore:

$$
k = 1, 2
$$

By observing equation (65) once more, the range of values that l takes are $l = 1, 2, \ldots, r_k$, where r_1 is subject to the individual's position in the queue and $r_2 = 1$. In essence, the hypoexponential distribution will be used with these bounds:

$$
\begin{aligned}
k = 1 \qquad &\Rightarrow \qquad l = 1, 2, \ldots, r_1 \\
k = 2 \qquad &\Rightarrow \qquad l = 1
\end{aligned} \tag{69}
$$

Thus the left hand side of equation (68) needs only to be defined for these bounds. The specific hypoexponential distribution investigated here is of the form $Hypo((r_1, 1)(\lambda_1, \lambda_2))$. Note the initial conditions $\lambda_0 = 0, r_0 = 1$ defined in equation (65) also hold here. Thus the proof is split into two parts, for $k = 1$ and $k = 2$.

- $k = 2, l = 1$

$$\begin{aligned} LHS &= -\frac{\partial^{1-1}}{\partial t^{1-1}} \left(\prod_{j=0, j\neq 2}^{2} (\lambda_j + t)^{-r_j} \right) \\ &= -\left((\lambda_0 + t)^{-r_0} \times (\lambda_1 + t)^{-r_1} \right) \\ &= -\left(t^{-1} \times (\lambda_1 + t)^{-r_1} \right) \\ &= -\frac{1}{t(t + \lambda_1)^{r_1}} \end{aligned}$$

□

- $k = 1, l = 1, \ldots, r_1$

$$\begin{aligned} LHS &= -\frac{\partial^{l-1}}{\partial t^{l-1}} \left(\prod_{j=0, j\neq 1}^{2} (\lambda_j + t)^{-r_j} \right) \\ &= -\frac{\partial^{l-1}}{\partial t^{l-1}} \left((\lambda_o + t)^{-r_0} \times (\lambda_2 + t)^{-r_2} \right) \\ &= -\frac{\partial^{l-1}}{\partial t^{l-1}} \left(\frac{1}{t(t + \lambda_2)} \right) \end{aligned}$$

In essence, it is only needed to show that:

$$-\frac{\partial^{l-1}}{\partial t^{l-1}} \left(\frac{1}{t(t + \lambda_2)} \right) = \frac{(-1)^l (l-1)!}{\lambda_2} \left[\frac{1}{t^l} - \frac{1}{(t + \lambda_2)^l} \right]$$

Proof by Induction:

1. Base case ($l = 1$):

$$\begin{aligned} LHS &= -\frac{\partial^{1-1}}{\partial t^{1-1}} \left(\frac{1}{t(t + \lambda_2)} \right) = -\frac{1}{t(t + \lambda_2)} \\ RHS &= \frac{(-1)^1 (1-1)!}{\lambda_2} \left[\frac{1}{t^1} - \frac{1}{(t + \lambda_2)^1} \right] \\ &= -\frac{t + \lambda_2 - t}{\lambda_2 t(t + \lambda_2)} \\ &= -\frac{1}{t(t + \lambda_2)} \\ LHS &= RHS \end{aligned}$$

2. Assume true for $l = x$:

$$-\frac{\partial^{x-1}}{\partial t^{x-1}} \left(\frac{1}{t(t + \lambda_2)} \right) = \frac{(-1)^x (x-1)!}{\lambda_2} \left[\frac{1}{t^x} - \frac{1}{(t + \lambda_2)^x} \right]$$

3. Prove true for $l = x + 1$. Need to show that:

$$\frac{\partial^x}{\partial t^x}\left(\frac{-1}{t(t+\lambda_2)}\right) = \frac{(-1)^{x+1}(x)!}{\lambda_2}\left[\frac{1}{t^{x+1}} - \frac{1}{(t+\lambda_2)^{x+1}}\right]$$

$$\begin{aligned}
LHS &= \frac{\partial}{\partial t}\left[\frac{\partial^{x-1}}{\partial t^{x-1}}\left(\frac{-1}{t(t+\lambda_2)}\right)\right] \\
&= \frac{\partial}{\partial t}\left[\frac{(-1)^x(x-1)!}{\lambda_2}\left(\frac{1}{t^x} - \frac{1}{(t+\lambda_2)^x}\right)\right] \\
&= \frac{(-1)^x(x-1)!}{\lambda_2}\left(\frac{(-x)}{t^{x+1}} - \frac{(-x)}{(t+\lambda_2)^x}\right) \\
&= \frac{(-1)^x(x-1)!(-x)}{\lambda_2}\left(\frac{1}{t^{x+1}} - \frac{1}{(t+\lambda_2)^x}\right) \\
&= \frac{(-1)^{x+1}(x)!}{\lambda_2}\left(\frac{1}{t^{x+1}} - \frac{1}{(t+\lambda_2)^x}\right) \\
&= RHS
\end{aligned}$$

□

Proportion within target for both types of individuals. Given the two CDFs of the Erlang and Hypoexponential distributions a new function has to be defined to decide which one to use among the two. Based on the state of the model, there can be three scenarios when an individual arrives.

1. There is a free server and the individual does not have to wait

$$X_{(u,v)} \sim Erlang(1, \mu)$$

2. The individual arrives at a queue at the n^{th} position and the model has $C > 1$ servers

$$X_{(u,v)} \sim Hypo((n, 1), (C\mu, \mu))$$

3. The individual arrives at a queue at the n^{th} position and the model has $C = 1$ servers

$$X_{(u,v)} \sim Erlang(n + 1, \mu)$$

Note here that for the first case $Erlang(1, \mu)$ is equivalent to $Exp(\mu)$. Consider $X^{(1)}_{(u,v)}$ to be the distribution of type 1 individuals and $X^{(2)}_{(u,v)}$ the distribution of type 2 individuals, when arriving at state (u, v) of the model.

$$X^{(1)}_{(u,v)} \sim \begin{cases} \textbf{Erlang}(v, \mu), & \textbf{if } C = 1 \textbf{ and } v > 1 \\ \textbf{Hypo}\left([v - C, 1], [C\mu, \mu]\right), & \textbf{if } C > 1 \textbf{ and } v > C \\ \textbf{Erlang}(1, \mu), & \textbf{if } v \leq C \end{cases} \tag{70}$$

$$X^{(2)}_{(u,v)} \sim \begin{cases} \textbf{Erlang}(\min(v,T),\mu), & \textbf{if } C=1 \textbf{ and } v,T>1 \\ \textbf{Hypo}\left([\min(v,T)-C,1],[C\mu,\mu]\right), & \textbf{if } C>1 \textbf{ and } v,T>C \\ \textbf{Erlang}(1,\mu), & \textbf{if } v\leq C \textbf{ or } T\leq C \end{cases} \tag{71}$$

Thus, the CDF of the random variables $X^{(1)}_{(u,v)}$ and $X^{(2)}_{(u,v)}$ can be calculated using equations (61) and (67):

$$P(X^{(1)}_{(u,v)}<t) = \begin{cases} 1-\sum_{i=0}^{v-1}\frac{1}{i!}e^{-\mu t}(\mu t)^i, & \textbf{if } C=1 \\ & \textbf{and } v>1 \\ 1-(\mu C)^{v-C}\mu\sum_{k=1}^{|\vec{r}|}\sum_{l=1}^{r_k}\frac{\Psi_{k,l}(-\lambda_k)t^{r_k-l}e^{-\lambda_k t}}{(r_k-l)!(l-1)!}, & \textbf{if } C>1 \\ \textbf{where } \vec{r}=(v-C,1) \textbf{ and } \vec{\lambda}=(C\mu,\mu) & \textbf{and } v>C \\ 1-e^{-\mu t}, & \textbf{if } v\leq C \end{cases} \tag{19 revisited}$$

$$P(X^{(2)}_{(u,v)}<t) = \begin{cases} 1-\sum_{i=0}^{\min(v,T)-1}\frac{1}{i!}e^{-\mu t}(\mu t)^i, & \textbf{if } C=1 \\ & \textbf{and } v,T>1 \\ 1-(\mu C)^{\min(v,T)-C}\mu & \textbf{if } C>1 \\ \quad\times\sum_{k=1}^{|\vec{r}|}\sum_{l=1}^{r_k}\frac{\Psi_{k,l}(-\lambda_k)t^{r_k-l}e^{-\lambda_k t}}{(r_k-l)!(l-1)!}, & \textbf{and } v,T>C \\ \textbf{where } \vec{r}=(\min(v,T)-C,1) & \\ \quad\vec{\lambda}=(C\mu,\mu) & \\ 1-e^{-\mu t}, & \textbf{if } v\leq C \\ & \textbf{or } T\leq C \end{cases} \tag{20 revisited}$$

In addition, the set of accepting states for type 1 ($S^{(1)}_A$) and type 2 ($S^{(2)}_A$) individuals defined in (7) and (8) are also needed here. Note here that, S denotes the set of all states of the Markov chain model.

$$S^{(1)}_A = \{(u,v)\in S \mid v<N\}$$

$$S^{(2)}_A = \begin{cases} \{(u,v)\in S \mid u<M\}, & \textbf{if } T\leq N \\ \{(u,v)\in S \mid v<N\}, & \textbf{otherwise} \end{cases}$$

The following formula uses the state probability vector π to get the weighted average of the probability below target of all states in the Markov model.

$$P(X^{(1)} < t) = \frac{\sum_{(u,v)\in S_A^{(1)}} P(X^{(1)}_{\mathcal{A}_1(u,v)} < t)\pi_{u,v}}{\sum_{(u,v)\in S_A^{(1)}} \pi_{u,v}} \tag{72}$$

$$P(X^{(2)} < t) = \frac{\sum_{(u,v)\in S_A^{(2)}} P(X^{(2)}_{\mathcal{A}_2(u,v)} < t)\pi_{u,v}}{\sum_{(u,v)\in S_A^{(2)}} \pi_{u,v}} \tag{73}$$

Note that $\mathcal{A}_1(u,v)$ and $\mathcal{A}_2(u,v)$ are defined in section 3.2 by equations 11 and 12.

Overall proportion within target. The overall proportion of individuals for both types of individuals is given by the equivalent formula of equation (6). The following formula uses the probability of lost individuals from both types to get the weighted sum of the two probabilities.

$$P_{L'_1} = \sum_{(u,v)\,\in S_A^{(1)}} \pi(u,v), \qquad P_{L'_2} = \sum_{(u,v)\,\in S_A^{(2)}} \pi(u,v)$$

$$\begin{aligned} P(X < t) = {} & \frac{\lambda_1 P_{L'_1}}{\lambda_2 P_{L'_2} + \lambda_1 P_{L'_1}} P(X^{(1)} < t) \\ & + \frac{\lambda_2 P_{L'_2}}{\lambda_2 P_{L'_2} + \lambda_1 P_{L'_1}} P(X^{(2)} < t) \end{aligned} \tag{21 revisited}$$

E. Type 1 and type 2 performance measure comparisons using simulation and Markov chains

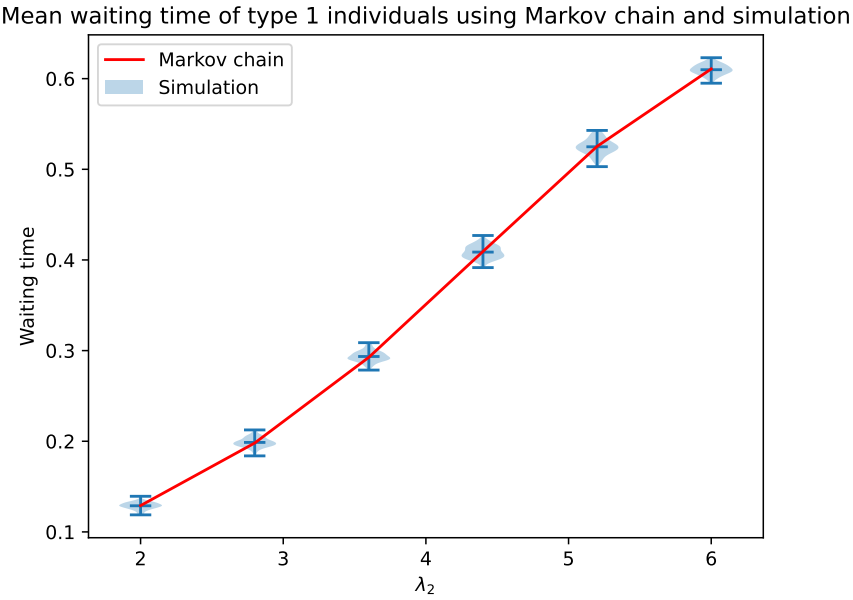

Figure 17: Comparison of mean waiting time for type 1 individuals between values obtained from the Markov chain formulas and values obtained from simulation.

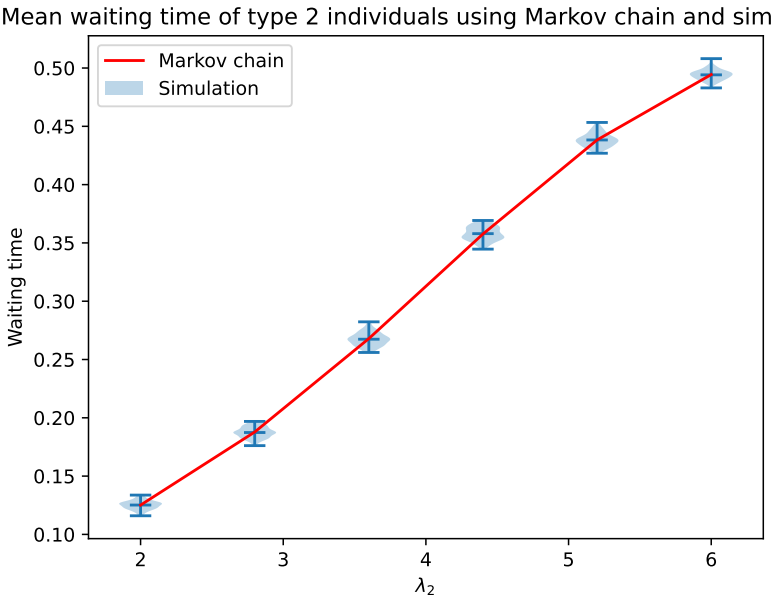

Figure 18: Comparison of mean waiting time for type 2 individuals between values obtained from the Markov chain formulas and values obtained from simulation.

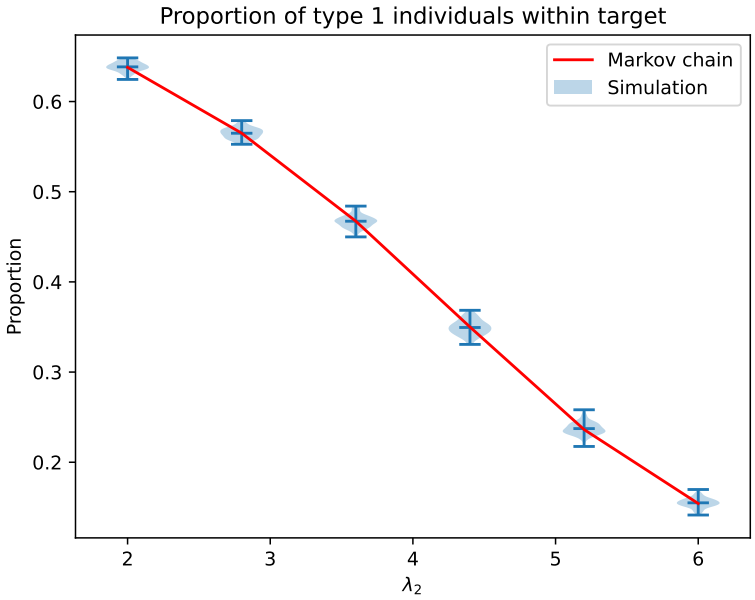

Figure 19: Comparison of proportion within target time for type 1 individuals between values obtained from the Markov chain formulas and values obtained from simulation.

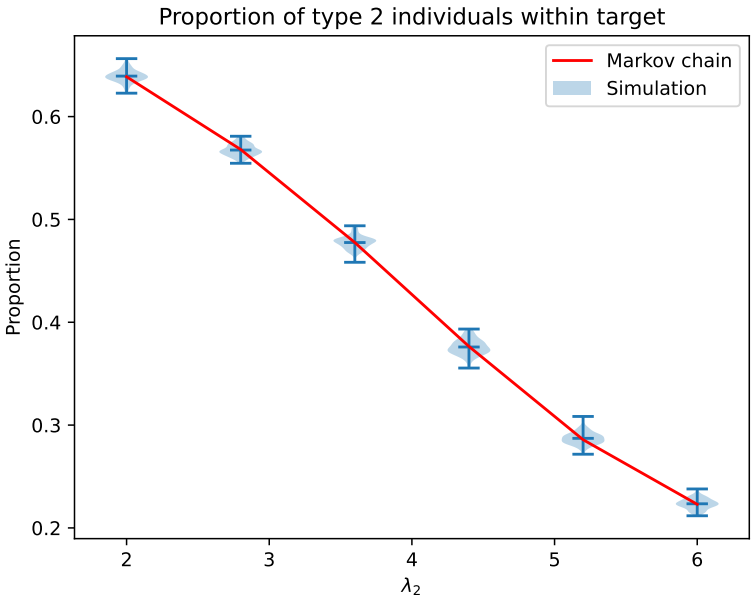

Figure 20: Comparison of proportion within target time for type 2 individuals between values obtained from the Markov chain formulas and values obtained from simulation.