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

1 Abstract

Emergency departments (EDs) in hospitals are usually under pressure to achieve a target amount of time that describes the arrival of patients and the time it takes to receive treatment. For example in the UK this is often set as 95% of patients to be treated within 4 hours. There is empirical evidence to suggest that imposing targets in the ED results in gaming at the interface of care between the EMS and ED. If the ED is busy and a patient is stable in the ambulance, there is little incentive for the ED to accept the patient whereby the clock will start ticking on the 4 hour target. This in turn impacts on the ability of the EMS to respond to emergency calls.

This study explores the impact that this effect may have on an ambulance's utilisation and their ability to respond to emergency calls. More specifically multiple scenarios are examined where an ambulance service needs to distribute patients between neighbouring hospitals. The interaction between the hospitals and the ambulance service is defined in a game theoretic framework where the ambulance service has to decide how many patients to distribute to each hospital in order to minimise the occurrence of this effect. The methodology involves the use of a queueing model for each hospital that is used to inform the decision process of the ambulance service so as to create a game for which the Nash Equilibria can be calculated.

Contents

2 Introduction

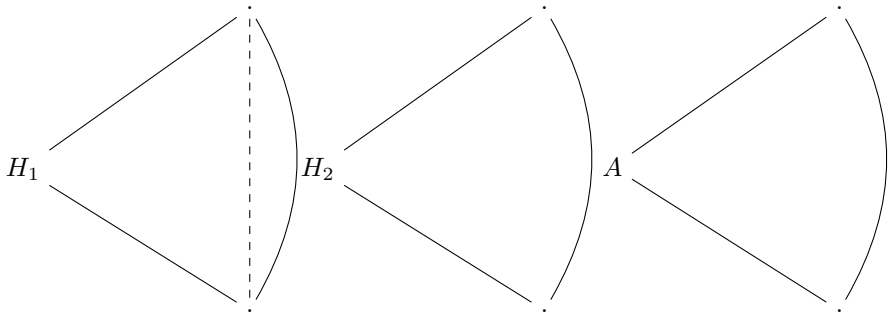


Figure 1: Ambulance Decision Problem

States:

1. A = Ambulance
2. H_i = Hospital i

Notation:

- Λ = total number of patients that need to be hospitalised
- p_i = proportion of patients going to Hospital i ($p_i\Lambda$ = number of patients going to hospital i)
- d_i = distance from Hospital i
- $\hat{c_i}$ = capacity of hospital i
- $W(c, \lambda\, \mu)$ = waiting time in the system function
- μ_i = service rate of hospital i
- λ_i^o = arrival rate of other patients to the hospital (not by ambulance)
- $C_i(p_i) = d_i + W(c = \hat{c_i},\ \lambda = p_i\Lambda + \lambda_i^o,\ \mu = \mu_i)$

3 Game Theory component:

Players:

- Ambulance
- Hospital A
- Hospital B

Strategies of players:

- Hospital i:
    1. Close doors at $\hat{c}_i = 1$
    2. Close doors at $\hat{c}_i = 2$
    3. ...
    4. Close doors at $\hat{c}_i = C_i$
- Ambulance:
    1. Choose $p_1 \in [0, 1]$

Cost Functions: Waiting times + the distance to each hospital.

4 Quick Methodology

- Fix the parameters Λ, λ_i^o, μ_i and C_i.
- $\forall\ \hat{c_i} \in \{1, 2, \ldots, C_A\}$ and $\forall\ \hat{c_j} \in \{1, 2, \ldots, C_B\}$
- Calculate p_A and $p_B = 1 - p_A$ s.t. $(W_q)_A = (W_q)_B$.
- Calculate the probability $P((W_q)_i \leq 4$ hours
- Fill matrix A with $U^A_{\hat{c_i},\hat{c_j}} = 1 - |0.95 - P((W_q)_A \leq 4)|$ and
- fill matrix B with $U^B_{\hat{c_i},\hat{c_j}} = 1 - |0.95 - P((W_q)_B \leq 4)|$

$$\text{A} = \begin{array}{|c|c|c|c|} \hline U^A_{1,1} & U^A_{1,2} & \ldots & U^A_{1,C_B} \\ \hline U^A_{2,1} & U^A_{2,2} & \ldots & U^A_{2,C_B} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline U^A_{C_A,1} & U^A_{C_A,2} & \ldots & U^A_{C_A,C_B} \\ \hline \end{array}$$

$$\text{B} = \begin{array}{|c|c|c|c|} \hline U^B_{1,1} & U^B_{1,2} & \ldots & U^B_{1,C_B} \\ \hline U^B_{2,1} & U^B_{2,2} & \ldots & U^B_{2,C_B} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline U^B_{C_A,1} & U^B_{C_A,2} & \ldots & U^B_{C_A,C_B} \\ \hline \end{array}$$

- Ambulance decides the proportion of people to distribute to each hospital based on optimal patient distribution.

5 Proper Methodology

The problem is formulated as a normal form game where the players are the two hospitals. Each hospital is given C_A and C_B number of strategies where C_A and C_B are the total capacities of the hospitals. In other words, depending on the capacity of each hospital, they may choose to stop receiving patients from arriving ambulances whenever they reach a certain capacity threshold. The goal of this problem is to satisfy the ED regulations which state that 95% of the patients should see a specialist within 4 hours of their arrival to the hospital. The mean of the random variable W_q is the average waiting time in the queue for hospital i.

$$W_q(\lambda_i, \mu_i, \hat{c_i}) = \frac{1}{\hat{c_i}\mu_i} \frac{(\hat{c_i}\rho_i)^{\hat{c_i}}}{\hat{c_i}!(1-\rho_i)^2} P_0, \quad i \in \{A, B\} \tag{1}$$

Thus, the utilities of the two players should be the proportion of people that fall within the 4 hours target. This is also equivalent to the probability of the waiting time of an individual to be less than or equal to 4 hours.

$$P(W_q(\lambda_i, \mu_i, \hat{c_i}) \leq 4), \quad i \in \{A, B\} \tag{2}$$

Therefore, a sensible goal for each player should be to minimise that probability, but the actual target of the hospitals is to satisfy 95% of those patients within the 4-hour time limit. Therefore, the goal should be to get that probability as close to 0.95 as possible. Thus each player should aim to minimise:

$$|0.95 - P(W_q(\lambda_i, \mu_i, \hat{c_i}) \leq 4)|, \quad i \in \{A, B\} \tag{3}$$

The classic formulation of a normal form game looks into the maximisation of each player's payoff. Consequently the utilities can be altered such that the goal of each player is to maximise:

$$U^A_{\hat{c_A}, \hat{c_B}} = 1 - |0.95 - P(W_q(\lambda_A, \mu_A, \hat{c_A}) \leq 4)| \tag{4}$$

$$U^B_{\hat{c_A}, \hat{c_B}} = 1 - |0.95 - P(W_q(\lambda_B, \mu_B, \hat{c_B}) \leq 4)| \tag{5}$$

Finally, the problem can be expressed as a normal form game with two players where each player/hospital has C_A and C_B strategies respectively. The two $C_A \times C_B$ payoff matrices for the utilities of the two hospitals can be defined as:

$$\text{A} = \begin{array}{|c|c|c|c|} \hline U^A_{1,1} & U^A_{1,2} & \dots & U^A_{1,C_2} \\ \hline U^A_{2,1} & U^A_{2,2} & \dots & U^A_{2,C_2} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline U^A_{C_1,1} & U^A_{C_1,2} & \dots & U^A_{C_1,C_2} \\ \hline \end{array} \qquad \text{B} = \begin{array}{|c|c|c|c|} \hline U^B_{1,1} & U^B_{1,2} & \dots & U^B_{1,C_2} \\ \hline U^B_{2,1} & U^B_{2,2} & \dots & U^B_{2,C_2} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline U^B_{C_1,1} & U^B_{C_1,2} & \dots & U^B_{C_1,C_2} \\ \hline \end{array}$$

Once the certain strategies of the game have been selected the ambulance service can decide what would be the optimal way to distribute patients. However, the way the ambulance service distributes patients can affect the utilities of the game. So how would one solve this kind of problem?

5.1 Solution

As mentioned before the problem requires the construction of two queuing models that will be needed for the formulation of the normal form game. Based on those utilities the ambulance service will then decide the percentage of patients that will distribute to each hospital.

First and foremost, the model consists of several parameters that are unknown and are assumed to be fixed. The model will be run multiple times for various values of these parameters.

Λ	Number of patients that need to be distributed
λ_i^o	Arrival rate of other patients that enter hospital i
μ_i	Service rate of hospital i
C_i	Total capacity of hospital i

Table 1: Fixed Parameters

Having established the fixed parameters of the model, the hospitals' utilities need to be calculated. In order to do so a backwards induction approach will be used. The EMS aims to distribute the patients such that the mean waiting time of patients is minimal. This can be further interpreted as when the mean waiting time of hospital A equals the mean waiting time of hospital B. Thus, the minimal mean waiting time can be found for the values of p_A and p_B that solve the following equation:

$$W_q(\lambda_A, \mu_A, \hat{c_A}) = W_q(\lambda_B, \mu_B, \hat{c_B}) \tag{6}$$

Equation (6) needs to be solved for all values of $c_i \in \{1, 2, \dots C_A\}$ and $c_j \in \{1, 2, \dots C_B\}$. Then, for every c_i and c_j the utility equation (4) has to be calculated for both hospitals. In order to solve it though, one must first estimate the probability $P[(W_q)_{\{A,B\}}] \leq 4]$. That is the probability that the waiting time in the queue for one of the hospitals is less than 4 hours. For a multi-server system, the distribution of the waiting time can be given by equation 7. The above expression returns the probability that the waiting time in the queue is less than some time T.

$$P(W_q > T) = \frac{(\frac{\lambda}{\mu})^c P_0}{c!(1 - \frac{\lambda}{c\mu})}(e^{-(c\mu - \lambda)T}) \tag{7}$$

Consequently when incorporating equation (7) into (4) a newer utility equation can be acquired:

$$U^{\{A,B\}}_{\hat{c_i},\hat{c_j}} = 1 - \left| \left[\frac{(\frac{\lambda}{\mu})^c P_0}{c!(1 - \frac{\lambda}{c\mu})} \left(e^{-(c\mu - \lambda)T} \right) \right] - 0.05 \right| \tag{8}$$

$$\text{A} = \begin{array}{|c|c|c|c|} \hline U^A_{1,1} & U^A_{1,2} & \dots & U^A_{1,C_2} \\ \hline U^A_{2,1} & U^A_{2,2} & \dots & U^A_{2,C_2} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline U^A_{C_1,1} & U^A_{C_1,2} & \dots & U^A_{C_1,C_2} \\ \hline \end{array} \qquad \text{B} = \begin{array}{|c|c|c|c|} \hline U^B_{1,1} & U^B_{1,2} & \dots & U^B_{1,C_2} \\ \hline U^B_{2,1} & U^B_{2,2} & \dots & U^B_{2,C_2} \\ \hline \vdots & \vdots & \ddots & \vdots \\ \hline U^B_{C_1,1} & U^B_{C_1,2} & \dots & U^B_{C_1,C_2} \\ \hline \end{array}$$

6 Hospital Markov chain model

The following Markov chain represents the transition between states of a hospital while capturing the EMS interaction with it. The hospital accepts both ambulance and other patients normally until a certain threshold T is reached. When it is reached all ambulances that arrive will be marked as *"parked outside"* until the number of people in the system is reduced below T. Additionally, if the patients in the hospital keep rising, they may exceed the number of servers C available, which will in turn mean that every new patient will have to wait for a server to become free. The states of the Markov chain are denoted by (u, v) where:

- u = number of ambulances parked outside of the hospital
- v = number of patients in the hospital

6.1 Markov-chain state mapping function

The transition matrix of the Markov-chain representation described above can be denoted by a state mapping function. The state space of this function is defined as:

$$\begin{aligned} S(T) =& S_1(T) \cup S_2(T) \text{ where:} \\ S_1(T) =& \left\{(0, v) \in \mathbb{N}_0^2 \mid v < T\right\} \\ S_2(T) =& \{(u, v) \in \mathbb{N}_0^2 \mid v \geq T\} \end{aligned} \tag{9}$$

Therefore, the entries of the transition matrix Q, can be given by $q_{i,j} = q_{(u_i,v_i),(u_j,v_j)}$ which is the transition rate from state $i = (u_i, v_i)$ to state $j = (u_j, v_j)$ for all $(u_i, v_i), (u_j, v_j) \in S$.

$$q_{i,j} = \begin{cases} \Lambda, & \textbf{if } (u_i, v_i) - (u_j, v_j) = (0, -1) \textbf{ and } v_i < \mathrm{t} \\ \lambda^o, & \textbf{if } (u_i, v_i) - (u_j, v_j) = (0, -1) \textbf{ and } v_i \geq \mathrm{t} \\ \lambda^a, & \textbf{if } (u_i, v_i) - (u_j, v_j) = (-1, 0) \\ v_i\mu, & \textbf{if } (u_i, v_i) - (u_j, v_j) = (0, 1) \textbf{ and } v_i \leq C \textbf{ or} \\ & \quad (u_i, v_i) - (u_j, v_j) = (1, 0) \textbf{ and } v_i = T \leq C \\ C\mu, & \textbf{if } (u_i, v_i) - (u_j, v_j) = (0, 1) \textbf{ and } v_i > C \textbf{ or} \\ & \quad (u_i, v_i) - (u_j, v_j) = (1, 0) \textbf{ and } v_i = T > C \\ -\sum_{j=1}^{|Q|} q_{i,j} & \textbf{if } i = j \\ 0, & \textbf{otherwise} \end{cases} \tag{10}$$

In order to acquire an exact solution of the problem a slight adjustment needs to be considered. The problem defined above assumes no upper boundary to the number of people that can wait for service or the number of ambulances that can be parked outside. Therefore, a different state space $\tilde{S}$ needs to be constructed where $\tilde{S} \subseteq S$ and there is a maximum allowed number of people N that can be in the system and a maximum allowed number of ambulances M parked outside:

$$\tilde{S} = \{(u, v) \in S \mid u \leq M, v \leq N\} \tag{11}$$

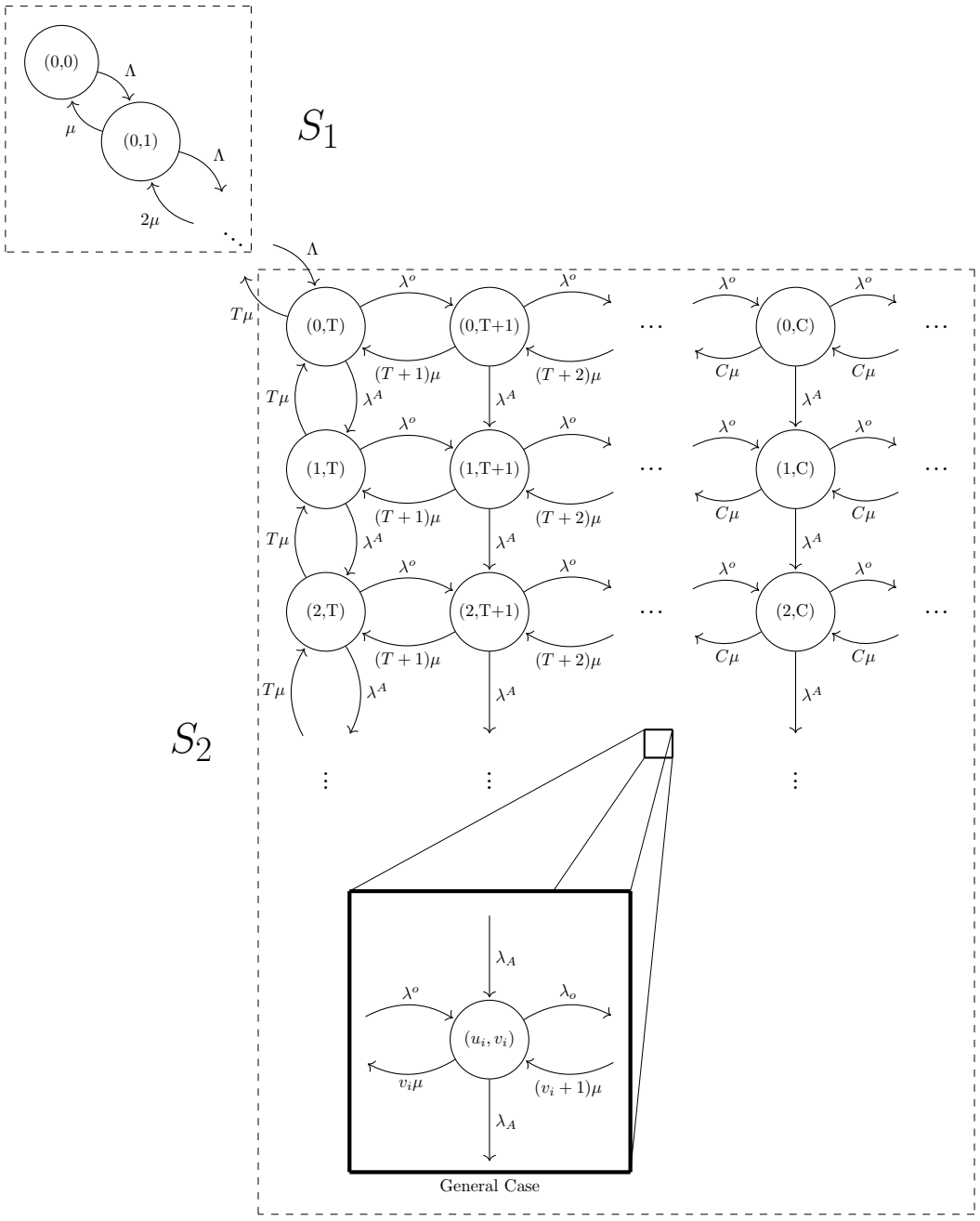


Figure 2: Markov chain

6.2 Steady State

Having calculated the transition matrix Q for a given set of parameters the probability vector π needs to be considered. The vector π is commonly used to study such stochastic systems and it's main purpose is to keep track of the probability of being at any given state of the system. 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:

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

There are numerous methods that can be used to solve problems of such kind. In this paper only numeric and algebraic approaches will be considered.

6.2.1 Numeric integration

The first approach to be considered is to solve the differential equation numerically by observing the behaviour of the model over time. The solution is obtained via python's SciPy library. The functions odeint and solve_ivp have been used in order to find a solution to the problem. Both of these functions can be used to solve any system of first order ODEs.

6.2.2 Linear algebraic approach

Another approach to be considered is the linear algebraic method. The steady state vector can be found algebraically by satisfying the following set of equations:

$$\pi Q = 0$$

$$\sum_i \pi_i = 1$$

These equations can be solved by slightly altering Q such that the final column is replaced by a vector of ones. Thus, the resultant solution occurs from solving the equation $\tilde{Q}^T \pi = b$ where $\tilde{Q}$ and b are defined as:

$$\tilde{q_{i,j}} = \begin{cases} 1, & \textbf{if } j = |Q| \\ q_{i,j}, & \textbf{otherwise} \end{cases}$$

$$b = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_m \end{bmatrix}$$

6.2.3 Least Squares approach

Finally, the last approach to be considered is the least squares method. This approach is considered because while the problem becomes more complex (in terms of input parameters) the computational time required to solve it increases exponentially. Thus, one may obtain the steady state vector π by solving the following equation.

$$\pi = \text{argmin}_{x \in \mathbb{R}^d} \|Mx - b\|_2^2$$

6.3 Graph Theoretical Approach

This section aims to investigate the connection between Markov chains and graph theory by exploring alternative ways of calculating the steady state probabilities and considering the problem from different perspectives.

6.3.1 Parameters

- the number of servers C,
- the threshold T,
- the system capacity N,
- the parking capacity M.

Additional parameters of the model are the ambulance arrival rate, the others' arrival rate and the service rate $(\lambda^A, \lambda^o, \mu)$. More specifically, the way these parameters are translated into the model are:

- **Number of servers (C):** All edges in the Markov chain that correspond to a service rate have a weight of:

$$w_{i,j} = \begin{cases} v_i\mu, & \textbf{if } v_i \leq C \textbf{ or } v_i = T \leq C \\ C\mu, & \textbf{if } v_i > C \textbf{ or } v_i = T > C \end{cases}$$

  Thus, the coefficients of the service rate have a lower bound of 1 and an upper bound of C.

- **Threshold (T):** Determines the length of the left *arm* of the model. In essence the threshold acts as a breakpoint between states where $u = 0$ and states where $0 \leq u \leq M$. Increasing T results in having more set of states where u can only be 0.

- **System capacity (N):** Is the upper bound of v for all states (u, v).

- **Parking capacity (M):** Is the upper bound of u for all states (u, v) such that $u \geq T$.

6.3.2 Example figure of Markov Model

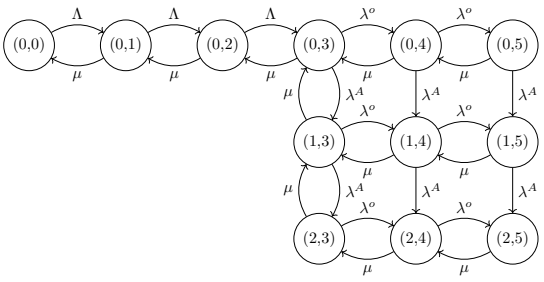


Figure 3: $C = 1, T = 3, N = 5, M = 2$

In figure 3 an example of such Markov model is shown where $C = 1$, $T = 3$ which means that the *left arm* of the model has a length of 3, $N = 5$ that indicates that the right-most states (u, v) are of the form $(u, 5)$ and $M = 2$ that equivalently shows that the bottom states are of the form $(2, v)$.

6.3.3 A graph theoretic model underling the Markov chain

An additional approach that one may consider to get the state probabilities is the graph theoretical approach for state probabilities. Thus, it can be assumed that a Markov chain model M can be translated as a weighted directed graph G_M where every edge has a weight that corresponds to the rate between the states of a given edge.

A *directed spanning tree* of a directed graph is defined as a subset of the graph that visits all the vertices of the graph and does not include any cycles. Unlike undirected spanning trees, directed ones also have a root which means that a directed spanning tree that is rooted at a vertex v has to have a path from any other vertex to vertex v. For example, consider the graph shown in figure 4. The graph points out a spanning tree that is rooted at vertex 3.

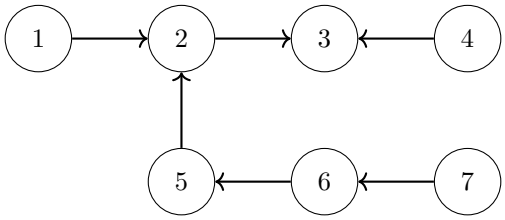


Figure 4: Spanning tree of a graph rooted at vertex 3

Additionally, let us denote the set of all spanning trees of G as $T(G)$ and the subset of $T(G)$ that includes only the spanning trees that are rooted at vertex v as $T_v(G)$. The weight of a spanning tree t can be defined as the product of the weights of the edges it contains:

$$w(t) = \prod_{e \in t} w(e)$$

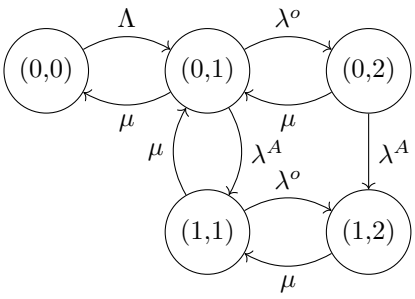
Theorem: Markov chain tree theorem [1]
Let M be an irreducible Markov chain on n states with stationary distribution $\pi_1, \pi_2, \ldots, \pi_n$. Let G_M be the directed graph associated with M. Then the probability of being at state u is given by:

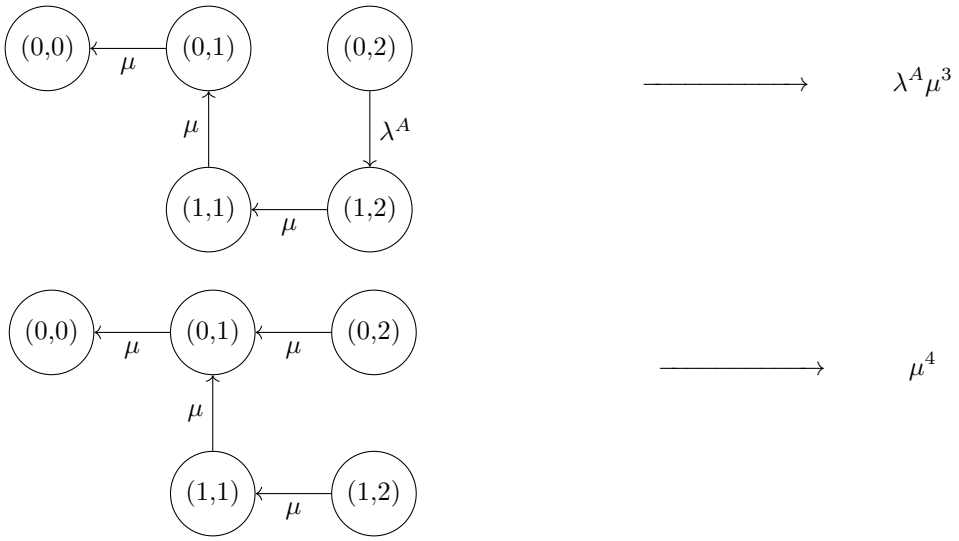
$$\pi_i = \frac{\sum_{t \in T_i(G_M)} w(t)}{\sum_{t \in T(G_M)} w(t)} \tag{12}$$

Equation 12 states that the probability of being at state u can be found by dividing the sum of the weights of all trees in $T_u(G)$ by the sum of the weights of all tress in $T(G)$. Let us ignore the denominator of that fraction for now and focus only on the numerator denoted as $\tilde{\pi}_i = \sum_{t \in T_i(G_M)} w(t)$

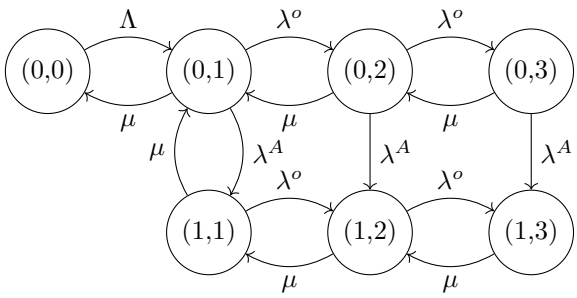
6.3.4 Spanning Trees rooted at $(0,0)$

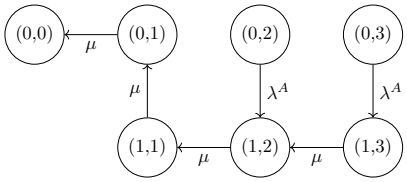
Let us now consider some examples of spanning trees that are rooted at $(0,0)$. For each of the following examples the complete graph G is shown, then all possible trees of $T_{(0,0)}(G)$ along with the weight associated with each spanning tree. As well as this, the sum of all the weights of the spanning trees denoted by $\tilde{\pi}_{(0,0)}$ is also included.



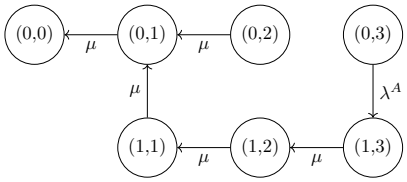


$$\tilde{\pi}_{(0,0)} = \mu^4 + \lambda^A \mu^3$$

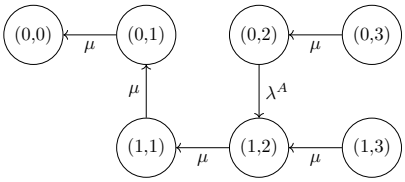




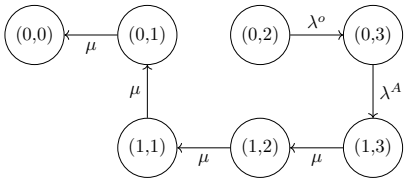
$\longrightarrow \quad (\lambda^A)^2\mu^4$



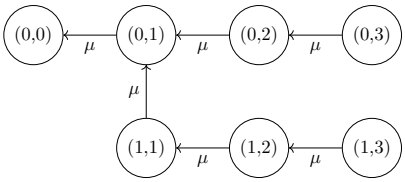
$\longrightarrow \quad \lambda^A\mu^5$



$\longrightarrow \quad \lambda^A\mu^5$

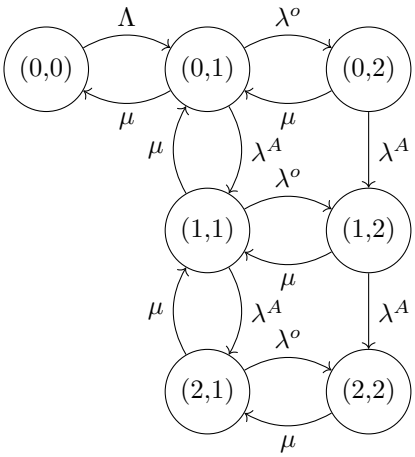


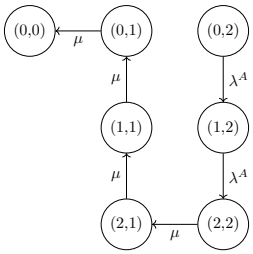
$\longrightarrow \quad \lambda^A\lambda^o\mu^4$



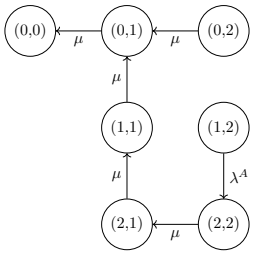
$\longrightarrow \quad \mu^6$

$$\tilde{\pi}_{(0,0)} = (\lambda^A)^2\mu^4 + 2\lambda^A\mu^5 + \lambda^A\lambda^o\mu^4 + \mu^6$$

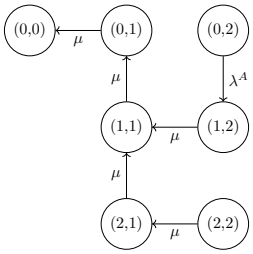




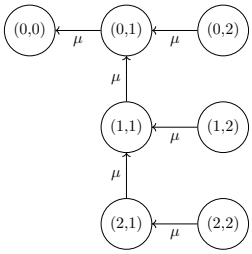
$(\lambda^A)^2\mu^4$



$\lambda^A\mu^5$



$\lambda^A\mu^5$



μ^6

$$\tilde{\pi}_{(0,0)} = (\lambda^A)^2\mu^4 + 2\lambda^A\mu^5 + \mu^6$$

6.3.5 Conjecture of adding rows

Let us consider three Markov models with the same number of servers $C = 1$, the same threshold $T = 1$, the same system capacity $N = 2$ but $M \in \{1, 2, 3\}$.

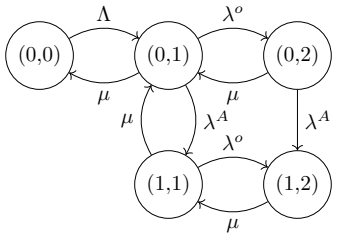


Figure 5: $M = 1$

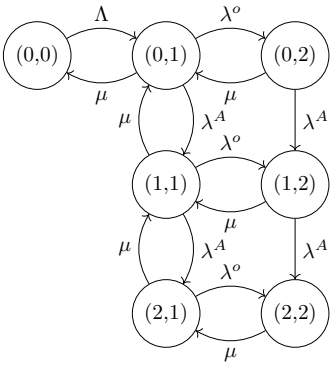


Figure 6: $M = 2$

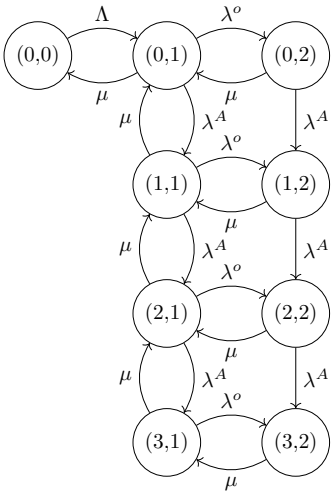


Figure 7: $M = 3$

By increasing the parking capacity of the system it can be observed that $|T_{(0,0)}(G)|$ increases as well since more combinations of paths can be generated using the new edges and vertices. The corresponding values of $\tilde{\pi}_{(0,0)}$ of the three systems are:

$$M = 1 : \tilde{\pi}_{(0,0)} = \mu^4 + \mu^3\lambda_A = \mu^3(\mu + \lambda^A) \tag{13}$$

$$M = 2 : \tilde{\pi}_{(0,0)} = \mu^6 + 2\mu^5\lambda_A + \mu^4(\lambda^A)^2 = \mu^4(\mu^2 + 2\mu\lambda_A + (\lambda^A)^2) = \mu^4(\mu + \lambda^A)^2 \tag{14}$$

$$\begin{aligned} M = 3 : \tilde{\pi}_{(0,0)} &= \mu^8 + 3\mu^7\lambda^A + 3\mu^6(\lambda^A)^2 + \mu^5(\lambda^A)^3 \\ &= \mu^5(\mu^3 + 3\mu^2\lambda^A + 3\mu(\lambda^A)^2 + (\lambda^A)^3) \\ &= \mu^5(\mu + \lambda^A)^3 \end{aligned} \tag{15}$$

Note that in equations (13),(14) and (15), the following equation holds:

$$\tilde{\pi}_{(0,0)} = \mu^{(N+M)}(\mu + \lambda^A)^M \tag{16}$$

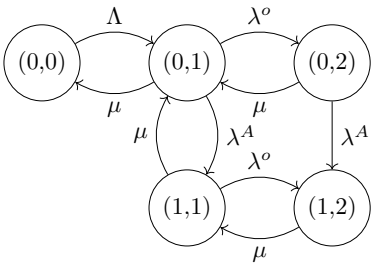
This relationship has been verified experimentally for ... and the data set is archived at ... A generalisation of equation 16, where $N \geq 1$, is given in terms of an unknown function $k(C, T, N)$ as:

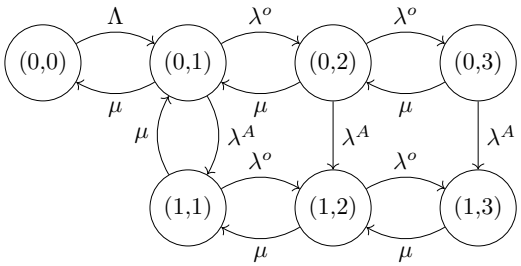
$$\tilde{\pi}_{(0,0)} = \mu^{(N+M)}(k(C, T, N))^M \tag{17}$$

Thus, having investigated the effect of adding rows it remains to investigate the effect of increasing N and finding an expression for $k(C, T, N)$.

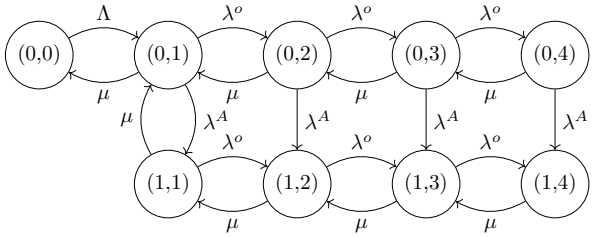
6.3.6 The effect of increasing N

In this section we will consider a parking capacity of $M = 1$ and see the effect of modifying other parameters on $k(C, T, N)$.

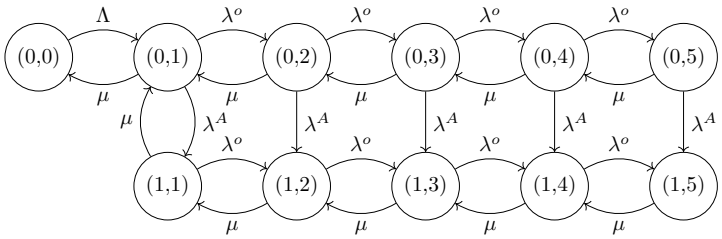




$$\tilde{\pi}_{(0,0)} = \mu^3[\lambda^a + \mu] \qquad \tilde{\pi}_{(0,0)} = \mu^4[(\lambda^A)^2 + \lambda^A\lambda^o + 2\lambda^A\mu + \mu^2] \tag{18}$$



$$\tilde{\pi}_{(0,0)} = \mu^5[(\lambda^A)^3 + 2(\lambda^A)^2\lambda^o + 3(\lambda^A)^2\mu + \lambda^A(\lambda^o)^2 + 2\lambda^A\lambda^o\mu + 3\lambda^A\mu^2 + \mu^3] \tag{19}$$



$$\begin{aligned}\tilde{\pi}_{(0,0)} =& \mu^6[(\lambda^A)^4 + 3(\lambda^A)^3\lambda^o + 4(\lambda^A)^3\mu + 3(\lambda^A)^2(\lambda^o)^2 + 6(\lambda^A)^2\lambda^o\mu \\ &+ 6(\lambda^A)^2\mu^2 + \lambda^A(\lambda^o)^3 + 2\lambda^A(\lambda^o)^2\mu + 3\lambda^A\lambda^o\mu^2 + 4\lambda^A\mu^3 + \mu^4]\end{aligned} \tag{20}$$

As explained in equation 16 the expressions defined above can boil down to a general form equation of the form $\tilde{\pi}_{(0,0)} = \mu^{(N+M)}(k(C,T,N))^M$. The only thing missing is an expression for $k(C,T,N)$. An initial attempt to get such an expression can be seen below:

$$\begin{aligned} k(C,T,N) &= \sum_{p_1=0}^{C-1} \sum_{p_2=0}^{C-p_1-1} \sum_{p_3=C-p_1-p_2-1}^{C-p_1-p_2-1} R(p_1,p_2,p_3)(\lambda^A)^{p_1}(\lambda^o)^{p_2}\mu^{p_3} \\ &= \sum_{p_1=0}^{C-1} \sum_{p_2=0}^{C-p_1-1} R(p_1,p_2,C-p_1-p_2-1)(\lambda^A)^{p_1}(\lambda^o)^{p_2}\mu^{C-p_1-p_2-1} \end{aligned} \tag{21}$$

In equation 21 the coefficient function $R(p_1,p_2,p_3)$ is introduced were takes as arguments the powers of λ^A, λ^o and μ. Note here that p_3, the power of μ, is defined as $p_3 = C-p_1-p_2-1$ since for all base models they need to satisfy $p_1+p_2+p_3 = C-1$. For the starting coefficients of the model the function $R(p_1,p_2,p_3)$ gives the values of the coefficients and is defined as:

$$R(p_1,p_2,p_3) = \begin{cases} 0 & \text{if } p_1 = 0 \text{ and } p_2 > 0 \\ 1 & \text{if } p_1, p_2 = 0 \text{ and } p_3 > 0 \\ \binom{p_1+p_3}{p_3} & \text{if } p_2 = 0 \text{ and } p_1 > 0 \\ \binom{p_1+p_2-1}{p_2} & \text{if } p_3 = 0 \text{ and } p_1, p_2 > 0 \\ p_3+1 & \text{if } p_1 = 1 \\ \binom{p_1+p_3+1}{p_1} + p_3\binom{p_1+p_3}{p_3+1} - \binom{p_1+p_3}{p_3} & \text{if } p_2 = 1 \text{ and } p_1 > 1 \\ \binom{p_1+p_2+1}{p_1} - \binom{p_1+p_2-1}{p_2-1} + \sum_{i=p_2}^{p_1+p_2-2} i\binom{i-1}{p_2-1} & \text{if } p_3 = 1 \text{ and } p_1, p_2 > 1 \\ U_{p_1,p_2,p_3} & \text{otherwise} \end{cases} \tag{22}$$

Note here that the final value U_{p_1,p_2,p_3} corresponds to coefficients that are unknown and are currently investigated. The function $R()$ takes as arguments a possible combination of numbers of λ^A, λ^o and μ for a given system and outputs the coefficient of that term which in turn represents how many spanning trees exist in the graph with that specific combination. For instance consider the coefficients (p_1,p_2,p_3) of some of the terms from the equations above:

- (18) $\Rightarrow (\lambda^A)^2$: $R(2,0,0) = \binom{2+0}{0} = 1$
- (18) $\Rightarrow \lambda^A\lambda^o$: $R(1,1,0) = \binom{1+1-1}{1} = 1$
- (18) $\Rightarrow 2\lambda^A\mu$: $R(1,0,1) = \binom{1+1}{1} = 2$
- (18) $\Rightarrow \mu^2$: $R(0,0,2) = 1$
- (19) $\Rightarrow 2(\lambda^A)^2\lambda^o$: $R(2,1,0) = \binom{2+1-1}{1} = 2$
- (19) $\Rightarrow 3(\lambda^A)^2\mu$: $R(2,0,1) = \binom{2+1}{1} = 3$
- (19) $\Rightarrow 3\lambda^A\mu^2$: $R(1,0,2) = \binom{1+2}{2} = 3$
- (20) $\Rightarrow 3(\lambda^A)^3\lambda^o$: $R(3,1,0) = \binom{3+1-1}{1} = 3$
- (20) $\Rightarrow 3(\lambda^A)^2(\lambda^o)^2$: $R(2,2,0) = \binom{3}{2} = 3$
- (20) $\Rightarrow 6(\lambda^A)^2\mu^2$: $R(2,0,2) = \binom{2+2}{2} = 6$
- (20) $\Rightarrow 6(\lambda^A)^2\lambda^o\mu$: $R(2,1,1) = \binom{2+1+1}{2} + 1\binom{2+1}{1+1} - \binom{2+1}{1} = 6+3-3 = 6$
- (e.g) $\Rightarrow (\lambda^A)^2(\lambda^o)^2\mu$: $R(2,2,1) = \binom{2+2+1}{2} - \binom{2+2-1}{2-1} + \sum_{i=2}^{2+2-2} i\binom{i-1}{2-1} = 10-3+(2\times 1) = 9$

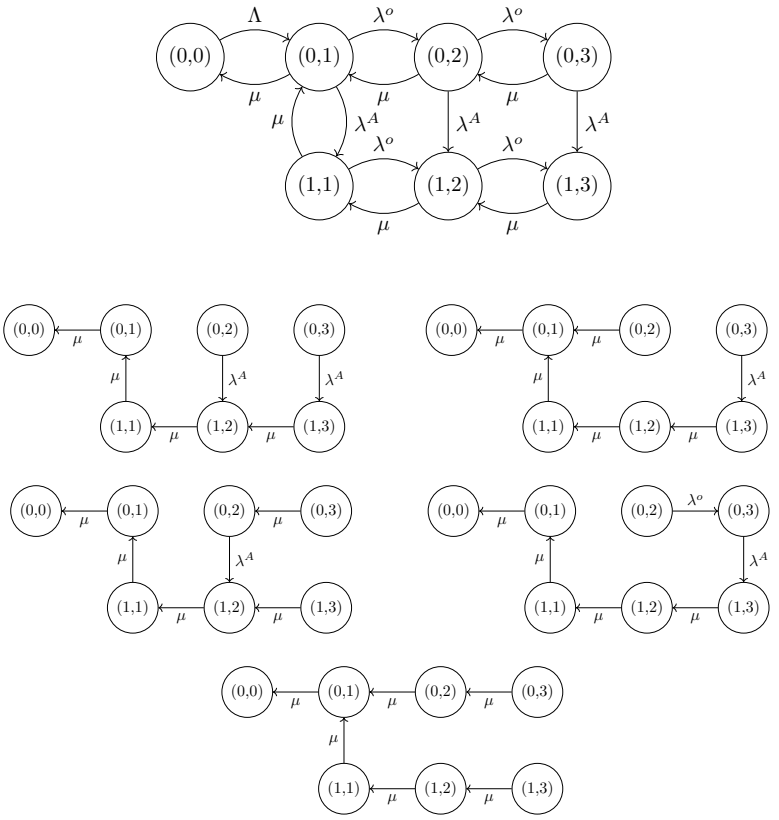
6.3.7 Unknown terms

The terms that remain unknown are the terms where $p_1, p_2, p_3 \geq 2$. Here are some of these values with the corresponding values of the $R(p_1, p_2, p_3)$ function.

- $R(2,2,2) = 18$
- $R(3,2,2) = 60$
- $R(2,3,2) = 24$
- $R(2,2,3) = 30$
- $R(4,2,2) = 150$
- $R(3,3,2) = 100$
- $R(3,2,3) = 120$
- $R(2,4,2) = 30$
- $R(2,3,3) = 40$
- $R(2,2,4) = 45$

6.3.8 DRL arrays

In this section a new combinatorial object is defined: DRL arrays. It will be shown that there is a bijection between DRL arrays and the spanning trees in G_M. DRL arrays will then be enumerated which in turn enumerates the trees of $T_{(0,0)}(G_M)$. Consider the following Markov model and the spanning trees rooted at state $(0,0)$ that are associated with it.



Looking at these spanning trees from a different perspective it can be observed that all spanning trees of the specific model have some edges in common.

- $(0,1) \to (0,0)$
- $(1,1) \to (0,1)$
- $(1,2) \to (1,1)$
- $(1,3) \to (1,2)$

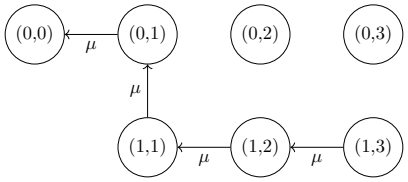
These edges are the ones on the bottom row of the model, on the *threshold column* and on the *arm* of the model. In general the set of edges that are present on all spanning trees can be denoted by:

$$\begin{aligned} S &= S_1 \cup S_2 \cup S_3 \\ S_2 &= \{(M,v) \to (M,v-1) \mid T < v \leq N\} \\ S_1 &= \{(u,T) \to (u-1,T) \mid 0 < u \leq M\} \\ S_3 &= \{(0,v) \to (0,v-1) \mid 0 < v \leq T\} \end{aligned} \tag{23}$$

In addition, these edges that are common to every spanning tree (for a threshold of $T = 1$) have the same weight of μ. In this specified model there are four of these edges, each with a weight of μ. Thus, since these edges exist on all spanning trees, the weight of every spanning tree must have include a term μ^4. Consider the expression of $\tilde{\pi}_{(0,0)}$ associated with this Markov model:

$$\tilde{\pi}_{(0,0)} = \mu^4[(\lambda^A)^2 + \lambda^A\lambda^o + 2\lambda^A\mu + \mu^2] \tag{24}$$

It can be seen that there is a μ^4 term that is a common factor of all the terms. This term can be more generally calculated as μ^{M+N} and by not worrying about all these edges that belong in S the problem can be slightly simplified.



The specific problem has now been reduced to finding all possible combinations of two edges where one starts from $(0,2)$ and the other from $(0,3)$. The possible edges that can be utilised here may have a direction of either left, right, or down. Thus, the objective of the problem can be transformed into finding all possible permutations of an array of size 2 where elements can be L, R or D and obey certain rules so that the permutation corresponds to a valid spanning tree. These rules are:

1. Permutations ending with an R are not valid.
2. Permutations that have an R followed by an L are not valid.

If any of these two rules does not hold then the permutation is immediately invalid. Rule 1 points to the cases where the final state has an edge pointing to the right of it, which cannot occur since that state is the right-most state of the first row. Rule 2 makes sure that there are no

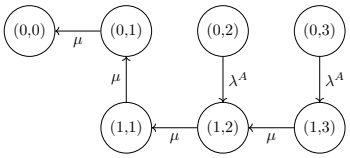
neighbour states that point to each other since that would create a cycle and would not generate a valid spanning tree.

For instance, consider the model above. Shown below, are all possible permutations of the array along with the excluded cases. The valid permutations (on the left) are shown in the same order with their corresponding spanning trees from the above figure and the invalid permutations (on the right) are followed by the rule that determines them invalid.

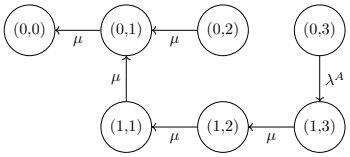
- $[D, D]$
- $[L, D]$
- $[D, L]$
- $[R, D]$
- $[L, L]$

- ~~$[R, R]$~~ → Rule 1
- ~~$[L, R]$~~ → Rule 1
- ~~$[R, L]$~~ → Rule 2
- ~~$[D, R]$~~ → Rule 1

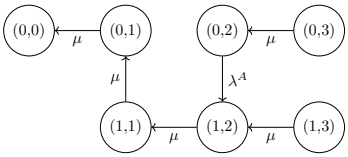
6.3.9 Examples of mappings of directed spanning trees to permutation arrays



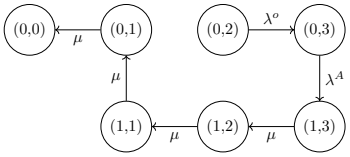
⟶ $[D, D]$



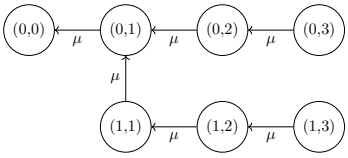
⟶ $[L, D]$



⟶ $[D, L]$



⟶ $[R, D]$



⟶ $[L, L]$

6.3.10 Closed-form formula

A general formula for finding all such permutations can be found, where the inputs are p_1, p_2 and p_3 that correspond to the number of D, R and L respectively and the output would be the coefficient of the term $(\lambda^A)^{p_1}(\lambda^o)^{p_2}\mu^{p_3}$. For instance, by applying such a formula to the example in equation 24, the desired output should be:

- $(\lambda^A)^2 \quad \rightarrow \quad p_1 = 2, p_2 = 0, p_3 = 0 \quad \rightarrow \quad \text{coefficient } = 1$
- $\lambda^A\lambda^o \quad \rightarrow \quad p_1 = 1, p_2 = 1, p_3 = 0 \quad \rightarrow \quad \text{coefficient } = 1$
- $2\lambda^A\mu \quad \rightarrow \quad p_1 = 1, p_2 = 0, p_3 = 1 \quad \rightarrow \quad \text{coefficient } = 2$
- $\mu^2 \quad \rightarrow \quad p_1 = 0, p_2 = 0, p_3 = 2 \quad \rightarrow \quad \text{coefficient } = 1$

Thus, given all possible and valid combinations of powers among λ^A, λ^o and μ (i.e. p_1, p_2, p_3) generated by equation 21, an alternative and improved form of the value of $R(p_1, p_2, p_3)$ described in equation 22 is given by:

$$R(p_1, p_2, p_3) = T(p_1, p_2, p_3) - E_R(p_1, p_2, p_3) - E_D(p_1, p_2, p_3) - E_L(p_1, p_2, p_3) - E_{RL}(p_1, p_2, p_3) \quad (25)$$

Consider the above undefined equations. The term $T(p_1, p_2, p_3)$ denotes the number of all permutations where neither rule is applied, i.e. all possible ways one can arrange the elements of the array. The term $E_R(p_1, p_2, p_3)$ denotes the number of permutations that end in R, which needs to be removed from the total of all permutations so that rule 1 is satisfied. Having excluded all permutations that end in R it only remains to leave out permutations that have an R followed by an L (rule 2). Although, removing all permutations ending in R was relatively straight forward, removing all permutations that follow rule 2 is not that easy. This is because equation $E_R(p_1, p_2, p_3)$ already considers some cases where there is an R followed by an L. Therefore, in order to consider only new cases, permutations of rule 2 are split into three new terms; E_D, E_L and E_{RL}. These terms denote the permutations that have an R followed by an L AND do not end in R. The term E_D considers all permutations that end in D while E_L the ones that end in L. Finally, the last term (E_{RL}) denotes all permutations that end in R, L where there is no other R followed by an L in any other position apart from the last two. This term is used because in the E_L term, such cases (where R and L are in the last two positions) are only considered when there is another R followed by an L somewhere. Thus, the term E_{RL} is a particular set of permutations that the formula of E_L fails to include by itself.

$$T(p_1,p_2,p_3) = \frac{(p_1+p_2+p_3)!}{p_1!\times p_2!\times p_3!} \tag{26}$$

$$E_R(p_1,p_2,p_3) = \frac{(p_1+p_2+p_3-1)!}{p_1!\times (p_2-1)!\times p_3!} \tag{27}$$

$$E_D(p_1,p_2,p_3) = \sum_{i=1}^{\min(R,L)} (-1)^{i+1} \frac{(p_1+p_2+p_3-i-1)!}{(p_1-1)!\times (p_2-i)!\times (p_3-i)!\times (i)!} \tag{28}$$

$$E_L(p_1,p_2,p_3) = \sum_{i=1}^{\min(R,L-1)} (-1)^{i+1} \frac{(p_1+p_2+p_3-i-1)!}{p_1!\times (p_2-i)!\times (p_3-i-1)!\times (i)!} \tag{29}$$

$$E_{RL}(p_1,p_2,p_3) = \sum_{i=1}^{\min(R,L)} (-1)^{i+1} \frac{(p_1+p_2+p_3-i-1)!}{p_1!\times (p_2-i)!\times (p_3-i)!\times (i-1)!} \tag{30}$$

$$R(p_1,p_2,p_3) = T(p_1,p_2,p_3) - E_R(p_1,p_2,p_3) - E_D(p_1,p_2,p_3) - E_L(p_1,p_2,p_3) - E_{RL}(p_1,p_2,p_3)$$

6.3.11 Example of the permutation algorithm

Consider the term $(\lambda^A)(\lambda^o)\mu^2$ and the above expressions. In order to get the coefficient of that term the permutation algorithm needs to be applied with an input of $p_1 = 1, p_2 = 1, p_3 = 2$, i.e. 1 D, 1 R and 2 Ls in the array. The permutations that correspond to each expression can be seen below:

$$T(p_1,p_2,p_3) = \frac{(1+1+2)!}{1!\ 1!\ 2!} = 12$$

$$\begin{array}{llllll} [D,R,L,L] & [R,D,L,L] & [D,L,R,L] & [R,L,D,L] & [D,L,L,R] & [R,L,L,D] \\ [L,D,R,L] & [L,R,D,L] & [L,D,L,R] & [L,R,L,D] & [L,L,D,R] & [L,L,R,D] \end{array}$$

$$E_R(p_1,p_2,p_3) = \frac{(1+1+2-1)!}{1!\ (1-1)!\ 2!} = 3$$

$$[D,L,L,|R] \quad [L,D,L,|R] \quad [L,L,D,|R]$$

$$E_D(p_1,p_2,p_3) = \sum_{i=1}^{1} (-1)^{i+1} \frac{(1+1+2-i-1)!}{0!\ (1-i)!\ (2-i)!\ (i)!} = 1\times \frac{2}{0!\ 0!\ 1!\ 1!} = 2$$

$$[R,L,L,|D] \quad [L,R,L,|D]$$

$$E_L(p_1,p_2,p_3)=\sum_{i=1}^{1}(-1)^{i+1}\frac{(1+1+2-i-1)!}{1!\,(1-i)!\,(2-i-1)!\,(i)!}=1\times\frac{2}{1!\,0!\,0!\,1!}=2$$

$$[D,R,L,|L]\quad[R,L,D,|L]$$

$$E_{RL}(p_1,p_2,p_3)=\sum_{i=1}^{1}(-1)^{i+1}\frac{(1+1+2-i-1)!}{1!\,(1-i)!\,(2-i)!\,(i_1)!}=1\times\frac{2}{1!\,0!\,1!\,0!}=2$$

$$[D,L,|R,L]\quad[L,D,|R,L]$$

6.3.12 Possibly useful theorem: Matrix-tree theorem for directed graphs (Kirchhoff's theorem) [2]:

The number of directed spanning trees rooted at a state i can be found by calculating the determinant of the Laplacian matrix Q of the directed graph and removing row i and column i.

6.4 Expressions derived from π:

One may easily derive the average number of individuals that are at any given state using *pi*. The average number of individuals in state i can be calculated by multiplying the number of individuals that are present in state i with the probability of being at that particular state (i.e $\pi_i(u_i + v_i)$). Using this logic it is possible to calculate any performance measures that are related to the mean number of individuals in the system.

Average number of patients in the system:

$$L = \sum_{i=1}^{|\pi|} \pi_i(u_i + v_i) \tag{31}$$

Average number of patients in the hospital:

$$L_H = \sum_{i=1}^{|\pi|} \pi_i v_i \tag{32}$$

Average number of ambulances being blocked:

$$L_A = \sum_{i=1}^{|\pi|} \pi_i u_i \tag{33}$$

Consequently getting the performance measures that are related to the duration of time is not as straightforward. Such performance measures are the mean waiting time in the system and the mean time blocked in the system. Under the scope of this study two approaches have been considered to calculate these performance measures; a recursive algorithm and consequently a closed-form formula.

The research question that needs to be answered here is: "When an ambulance/other patient enters the system, what is the expected time that they will have to wait?". In order to formulate the answer to that question one needs to consider all possible scenarios of what state the system can be in when an individual arrives. Furthermore, a different recursive formula arises for *ambulance patients* and a different one for *other patients*.

6.5 Mean waiting time

6.5.1 Recursive formula for mean waiting time of *other patients*

To calculate the mean waiting time of *other patients* one must first identify the set of states (u, v) that will imply that 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 hospital 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:

$$S_w = \{(u, v) \in S \mid v > C\} \tag{34}$$

Additionally, there are certain states in the model where arrivals cannot occur. An *other patient* cannot arrive whenever the model is at any state (u, N) for all u where N is the system capacity.

Therefore the set of all such states that an arrival may occur are defined as *accepting states* and are denoted as:

$$S_A^{(o)} = \{(u, v) \in S \mid u < N\} \tag{35}$$

Moreover, another element that needs to be considered is the expected waiting time in each state $c(u, v)$, otherwise known as sojourn 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.

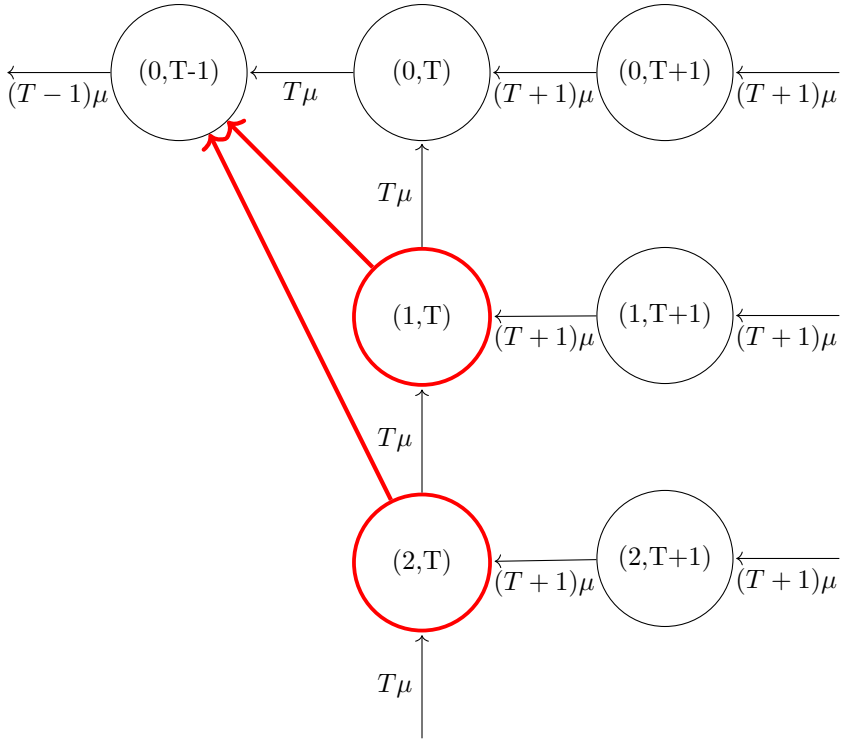


Figure 8: Markov chain - ignoring any arrivals

As illustrated in figure 8 an *other patient*, when in the threshold column, only visits one of the nodes since they are not affected by *ambulance patients*. Thus, one may acquire the desired time by calculating the inverse of the sum of the out-flow rate of that state. Since we are ignoring arrivals though the only way to exit the state will only be via a service. In essence this notion can be expressed as:

$$c^{(o)}(u,v) = \begin{cases} 0, & \textbf{if } u > 0 \textbf{ and } v = T \\ \frac{1}{\min(v,C)\mu}, & \textbf{otherwise} \end{cases} \tag{36}$$

Note that whenever any *other patient* 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 other patients. 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.

Now, using the above equations, and considering all sates that belong in S_w the following recursive formula can be used to get the mean waiting time spent in each state in the Markov model. For *other patients*, whenever the model is at state (u, v), any incoming patient will proceed to arrive at state $(u, v + 1)$. Patients will then proceed to visit all other states until they reach one which has less than C servers occupied (i.e. until a server becomes available). The formula goes through all states from right to left recursively and adds the sojourn times of all these states together until it reaches a state that is not in the set of waiting states. Thus, the expected waiting time of an *other patient* when they arrive at state (u, v) can be given by:

$$w^{(o)}(u,v) = \begin{cases} 0, & \textbf{if } (u,v) \notin S_w \\ c^{(o)}(u,v) + w^{(o)}(u-1,v), & \textbf{if } u > 0 \textbf{ and } v = T \\ c^{(o)}(u,v) + w^{(o)}(u,v-1), & \textbf{otherwise} \end{cases} \tag{37}$$

Finally, the overall 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.

$$W^{(o)} = \frac{\sum_{(u,v)\in S_A^{(o)}} w^{(o)}(u,v)\pi_{(u,v)}}{\sum_{(u,v)\in S_A^{(o)}} \pi_{(u,v)}} \tag{38}$$

6.5.2 Recursive formula for mean waiting time of *ambulance patients*

Equivalently the mean waiting time for *ambulance patients* can be calculated in a similar manner. The set of waiting states is the same as before but there is a slight modification in the set of accepting states.

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

$$S_A^{(a)} = \begin{cases} \{(u,v) \in S \mid v < M\} & \textbf{if } T \leq N \\ \{(u,v) \in S \mid v < N\} & \textbf{otherwise} \end{cases} \tag{39}$$

The set of accepting states is modified in such a way such that an *ambulance patient* cannot arrive in the model when the model is at any state (M, v) for all $v \geq T$ where M is the parking capacity and T is the threshold. An odd situation here is when the threshold is set to a very high number that is more than the actual system capacity. In such cases the set of accepting states is defined in the same way as the *other patients* case. That is because whenever $T > N$ no ambulance will ever be blocked in the model (since that threshold can never be reached) and thus the last accepting state of the model will be state $(0, N - 1)$.

Now just like in the *other patients* case the sojourn time is needed. For *ambulance patients* whenever individuals are at any row apart from the first one they automatically get a wait time of 0 since they are essentially blocked.

$$c^{(a)}(u,v) = \begin{cases} 0, & \textbf{if } u > 0 \\ \frac{1}{\min(v,C)\mu}, & \textbf{otherwise} \end{cases} \tag{40}$$

Finally, the recursive formula and the mean waiting time equation are identical to the ones described above with the exception that they now use $c^{(a)}(u,v)$ instead of $c^{(o)}(u,v)$.

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

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

6.5.3 Mean Waiting Time - Closed-form

Upon closer inspection of the recursive formula a more compact formula can arise. The equivalent closed-form formula eliminates the need for recursion and thus makes the computation of waiting times much more efficient. Just like in the recursive part there are two formulas; one for *ambulance* and one for *other patients*. The formulas are given by:

$$W^{(o)} = \frac{\sum_{\substack{(u,v)\in S_A^{(o)} \\ v\geq C}} \frac{1}{C\mu} \times (v - C + 1) \times \pi(u,v)}{\sum_{(u,v)\in S_A^{(o)}} \pi(u,v)} \tag{43}$$

$$W^{(a)} = \frac{\sum_{\substack{(u,v)\in S_A^{(a)} \\ min(v,T)\geq C}} \frac{1}{C\mu} \times (\min(v+1,T) - C) \times \pi(u,v)}{\sum_{(u,v)\in S_A^{(a)}} \pi(u,v)} \tag{44}$$

Note here that the summation, in both equations 43 and 44, goes through all states in the set of accepting states of either *ambulance or other patients* respectively, where a wait incurs. In equation 43 that includes all states (u,v) in the set of accepting states of other patients such that $v \geq C$; i.e. whenever an arrival occurs and the system is at a state where the number of individuals in the system is more than or equal to C. Consequently, for the states that are included in the summation the expression $v - C + 1$ indicates the amount of people in service one would have to wait for upon arrival at the hospital.

Additionally, the minimisation function in equation 44 (*ambulance patients*) ensures that when an ambulance arrives at any state that is greater than the predetermined threshold, the wait that the individual will have to endure remains the same. In essence, the expression $\min(v+1,T) - C$ returns the number of people in line in front of a particular individual upon arrival.

6.5.4 Overall Waiting Time

Consequently, the overall waiting time should can be estimated by a linear combination of the waiting times of *other and ambulance patients*. The overall waiting time can be then given by the following equation where c_o and c_a are the coefficients of each patient's type waiting time:

$$W = c_o W^{(o)} + c_a W^{(a)} \tag{45}$$

The two coefficients represent the proportion of patients of each patient type that traversed through the model. Theoretically, getting these percentages should be as simple as looking at the arrival rates of each patient type but in practise if the hospital or the parking space is full, some patients may be lost to the system. Thus, one should account for the probability that a patient is lost to the system. This probability can be easily calculated by using the two sets of accepting states $S_A^{(a)}$ and $S_A^{(o)}$ defined earlier in equations 35 and 39. Let us define here the probability, for either patient type, that an individual is not lost in the system by:

$$P(L'_o) = \sum_{(u,v)\,\in S_A^{(o)}} \pi(u,v) \qquad\qquad P(L'_a) = \sum_{(u,v)\,\in S_A^{(a)}} \pi(u,v)$$

Having defined these probabilities one may combine them with the arrival rates of each patient type in such a way to get the expected proportions of ambulance and other patients in the model. Thus, by using these values as the coefficient of equation 45 the resultant equation can be used to get the overall waiting time. Note here that the equation below gets the overall waiting time for both the recursive and the closed-form formula.

$$W = \frac{\lambda_o P(L'_o)}{\lambda_a P(L'_a) + \lambda_o P(L'_o)} W^{(o)} + \frac{\lambda_a P(L'_a)}{\lambda_a P(L'_a) + \lambda_o P(L'_o)} W^{(a)} \tag{46}$$

7 Markov chain VS Simulation

7.1 Example model

Consider the Markov chain paradigm in figure 9. The illustrated model represents the unrealistically small system of a hospital with a system capacity of five and an ambulance parking capacity of three. The hospital in this particular example also has four servers and a threshold of three; meaning that every ambulance that arrives in a time that there are three or more individuals in the hospital, will proceed to the parking space.

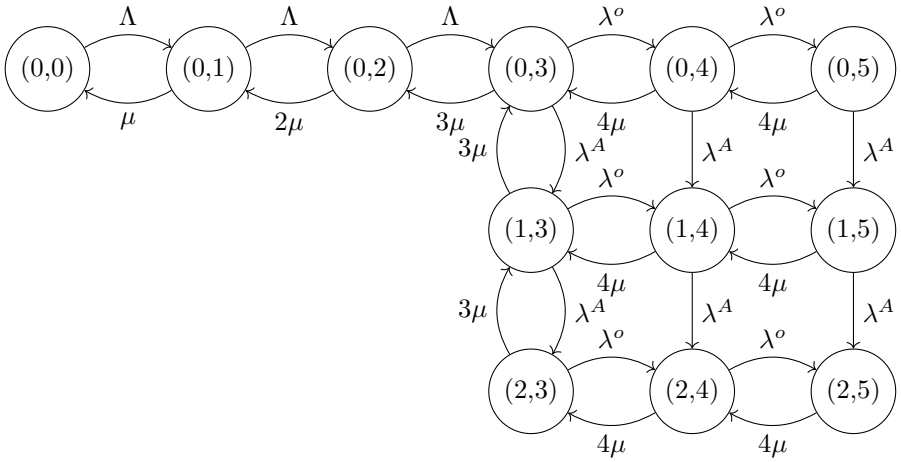


Figure 9: Markov chains: number of servers=4

In addition to the Markov chain model a simulation model has also been built based on the same parameters. Comparing the results of the Markov model and the equivalent simulation model the resultant plots arose.

The heatmaps in figure 10 represent the state probabilities for the Markov chain model, the simulation model and the difference between the two. Each pixel of the heatmap corresponds to the equivalent state of figure 9 and represents the probability of being at that state in any particular moment of time.

It can be observed that both Markov chain and simulation models' state probabilities vary from 5% to 25% and that states $(0, 1)$ and $(0, 2)$ are the most visited ones. Looking at the differences' heatmap, one may identify that the differences between the two are minimal.

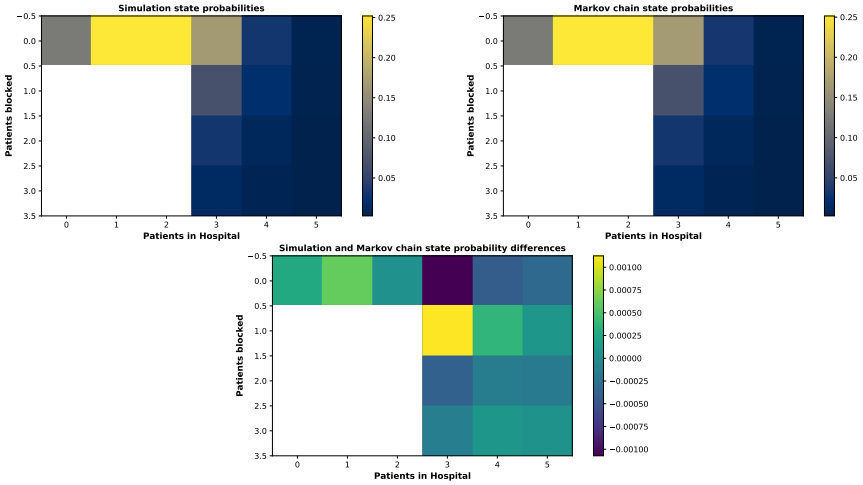
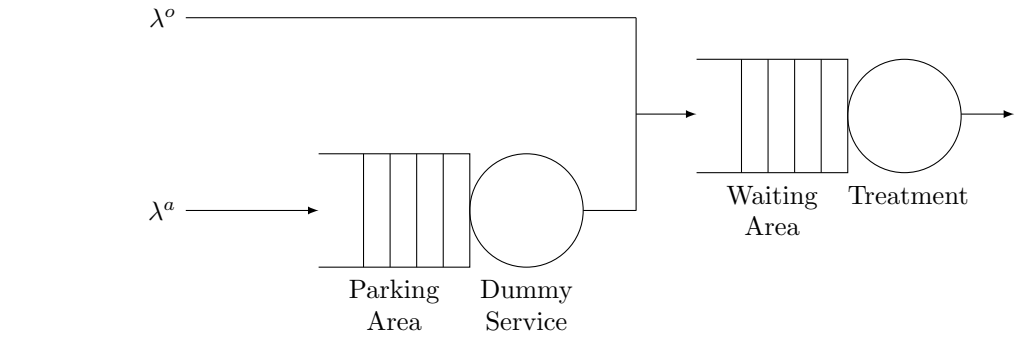
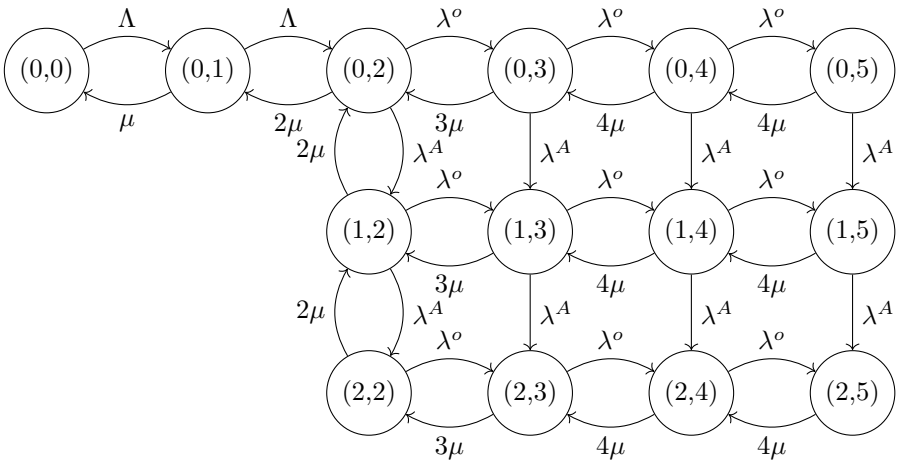


Figure 10: Heatmaps of Simulation, Markov chains and differences of the two

8 Figures that might be useful





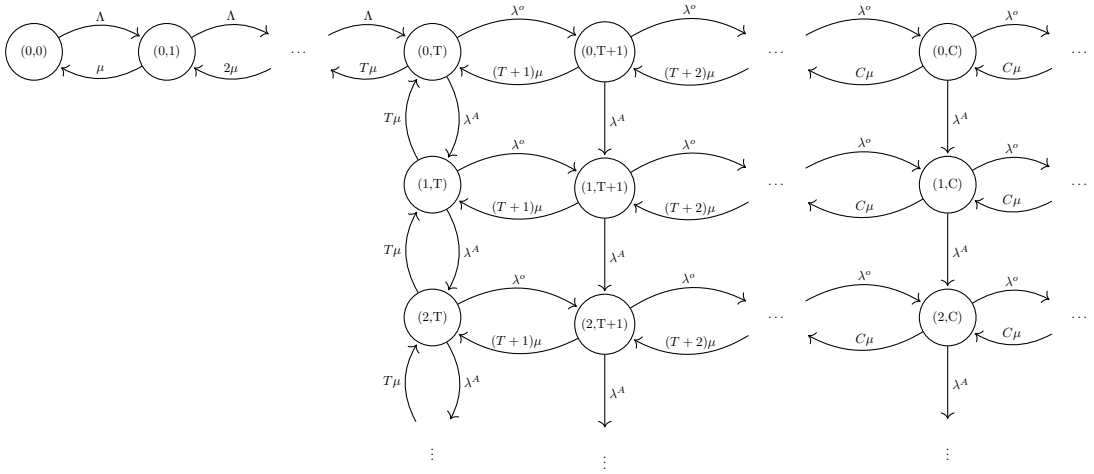


Figure 11: Markov chains

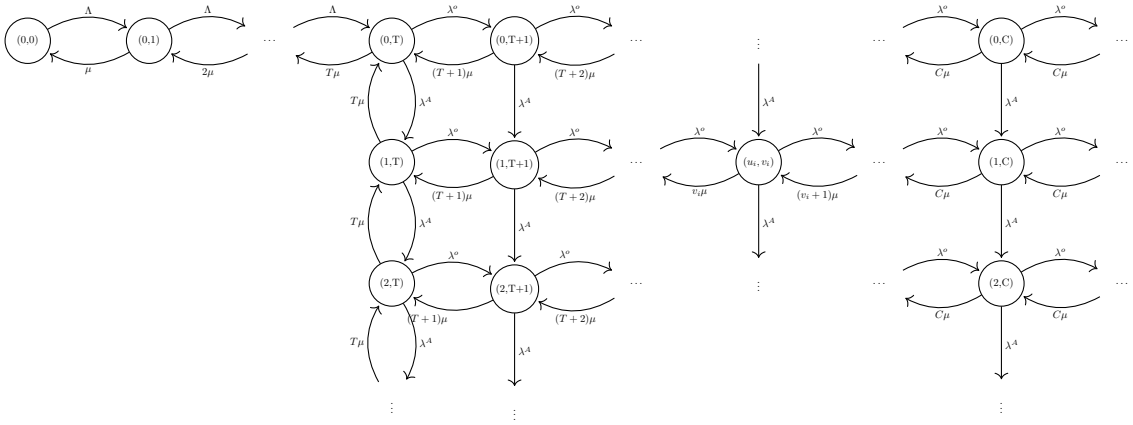


Figure 12: Markov chains

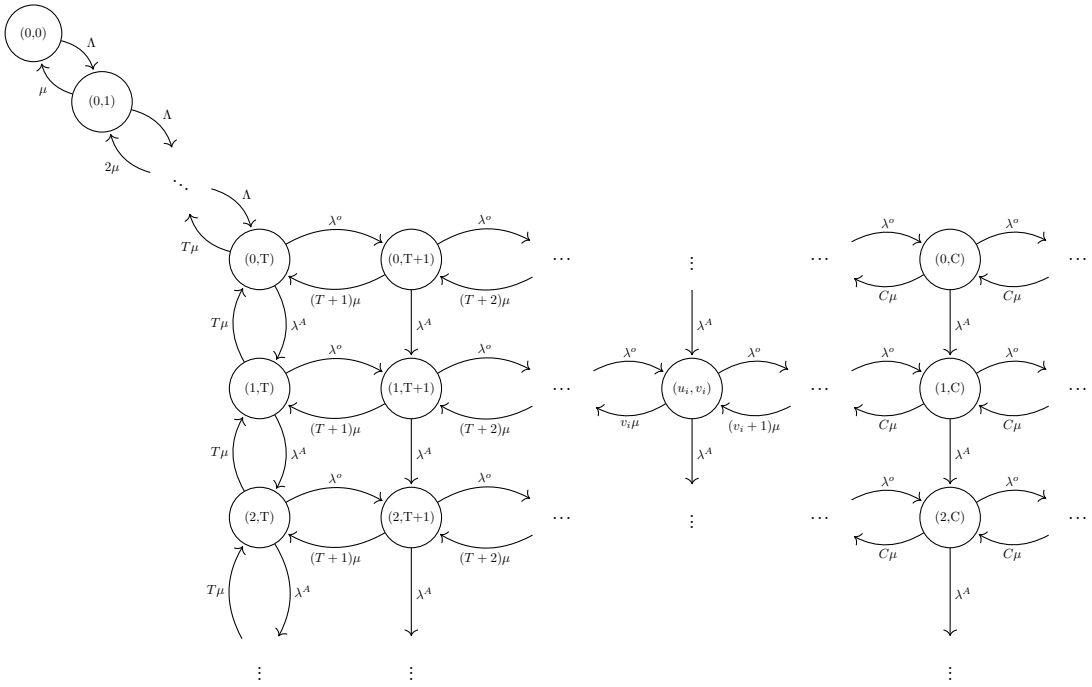


Figure 13: Markov chains

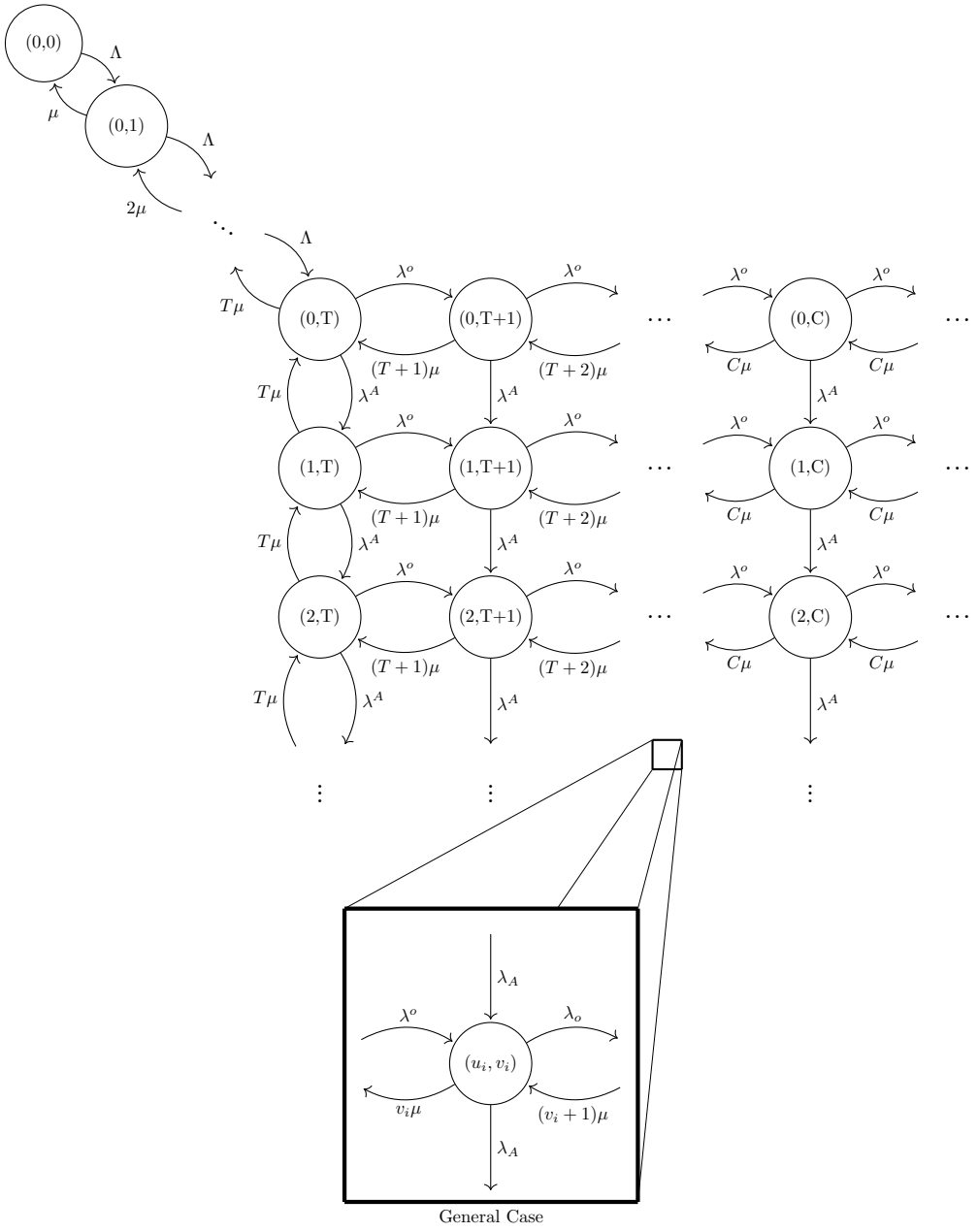


Figure 14: Markov chain

9 Formulas

$$\hat{c_i} \in \{1, 2, \ldots, C_i\}$$

$$\rho_i = \frac{p_i \Lambda + \lambda_i^o}{\hat{c_i} \mu_i}$$

$$(W_q)_i = \frac{1}{\hat{c_i} \mu_i} \frac{(\hat{c_i} \rho_i)^{\hat{c_i}}}{\hat{c_i}!(1-\rho_i)^2} (P_0)_i$$

$$(P_0)_i = \frac{1}{\sum_{n=0}^{\hat{c_i}-1} \left[\frac{(\hat{c_i}\rho_i)^n}{n!} \right] + \frac{(\hat{c_i}\rho_i)^{\hat{c_i}}}{\hat{c_i}!(1-\rho_i)}}$$

$$P(W_q > T) = \frac{(\frac{\lambda}{\mu})^c P_0}{c!(1-\frac{\lambda}{c\mu})} (e^{-(c\mu-\lambda)T})$$

References

[1] Broder, A. (1989). Generating random spanning trees. Annual Symposium on Foundations of Computer Science (Proceedings), 442–447. https://doi.org/10.1109/sfcs.1989.63516

[2] Chaiken, S., & Kleitman, D. J. (1978). Matrix Tree Theorems. Journal of Combinatorial Theory, Series A, 24(3), 377–381. https://doi.org/10.1016/0097-3165(78)90067-5