End of Year Progress Monitoring Report

Geraint Palmer

Supervisors: Professor Paul Harper & Dr. Vincent Knight

1 Project Overview & Plan

This project is supervised by Professor Paul Harper and Dr Vincent Knight. The project aims to give a whole systems view of the healthcare system within the Aneurin Bevan University Health Board, and will investigate the workforce needs of the frail and the elderly across the health board. The whole system will be modelled as a queueing network, with nodes representing large departments in order to capture general patient flows. The model parameters will be obtained by exploring and analysing data aquired from the SAIL data bank. An agent-based simulation model will be created from this model, populated with the obtained data, where further exploritoration can be undertaken.

Frail and elderly patients are currently a particular priority for many healthcare managers. An aging population means and increasing amount of elderly people are accessing to healthcare and are entering the healthcare systems. Frail patients tend to have prolonged hospital stays, and remain in the system longer than younger patients, partly due to insufficient care at home or in the community. Patients who remain in hospitals when they could otherwise have returned home block beds, increasing congestion and pressure on other areas of the healthcare system. This becomes a strain on the workforce, both in hospitals and community care teams. An understanding of the situation could alleviate this pressure.

1.1 Patient Flows

Queueing theory, and in particular queueing networks are great techniques to model the stochastic flow of customers around a system of service nodes. These concepts have been successfully adapted to healthcare systems, both in detailed individual level systems, e.g. [1], [10] and in examples with corser granularity, for example [16].

An example of the sort of queueing network and patient flows that will be modelled is shown in Figure 1.

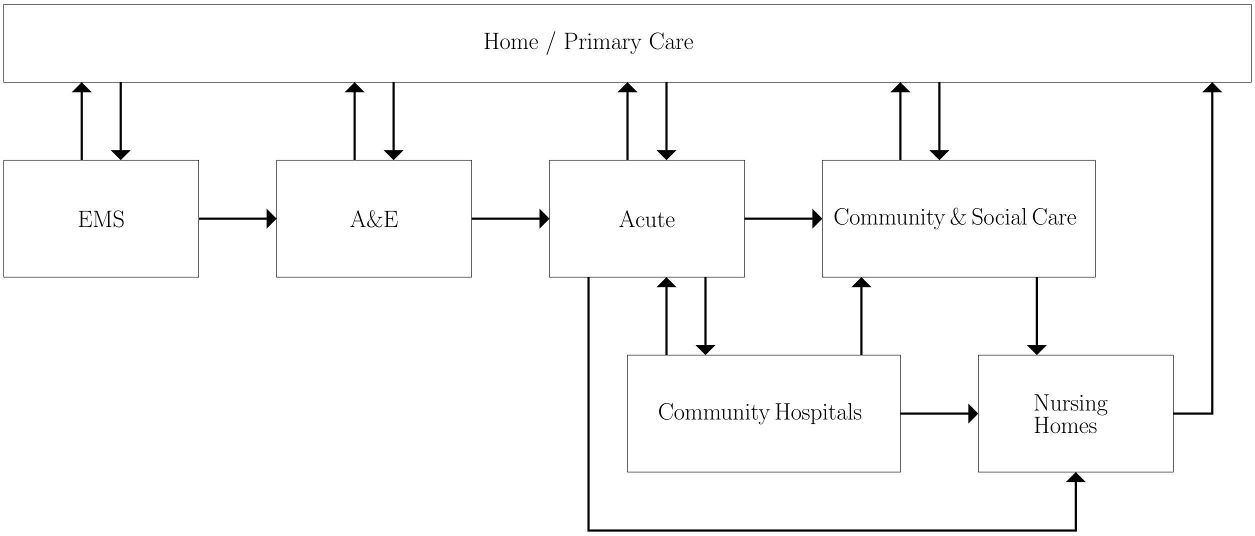


Figure 1: Patient flows around a healthcare system.

From exploring data and discussing with professionals at the ABUHB, the general architecture of the health system can be constructed, and patient flows built up. This will then be modelled as an open queueing network. Using results from the theory of Jackson networks performance measures and steady-state probabilities can be obtained. This model will then be extended to include blocking between nodes so that the model represents reality, in which patients may be unable to progress to their next destination due to lack of capacity. Again performance measures and steady-state probabilities will be obtained.

A simulation framework has been written in Python, therefore these results will be compared to results obtained through simulation. A user friendly Django application of this simulation will be built which will feature simple parameter inputs and graphical results.

1.2 Optimal Routing of Patients

The simulation model will be adapted to become agent-based, and so the agents within the simulation will be able to start leanring about the system itself and finding its own best setup. One interesting avenue to explore is the optimal routing of patients throught the system. That is, when there is a choice of where to send a patient, what is the optimal action to take? This question will be investigated using reinforcement learning algorithms.

Reinforcement learning is a form of machine learning in which an agent learns through simulation, by completing actions and receiving numerical rewards or punishments. Overviews are given in [24] and [25]. The q-learning algorithm will be used to find the optimal action to take given a specific state.

In order to implement this for the queueing network simulation model, the nodes or service centres will become the agents, the state will be the number of patients at that node, and the reward will be some function of expected wait. It will be interesting to see the effect of social vs selfish rewards (that is, does

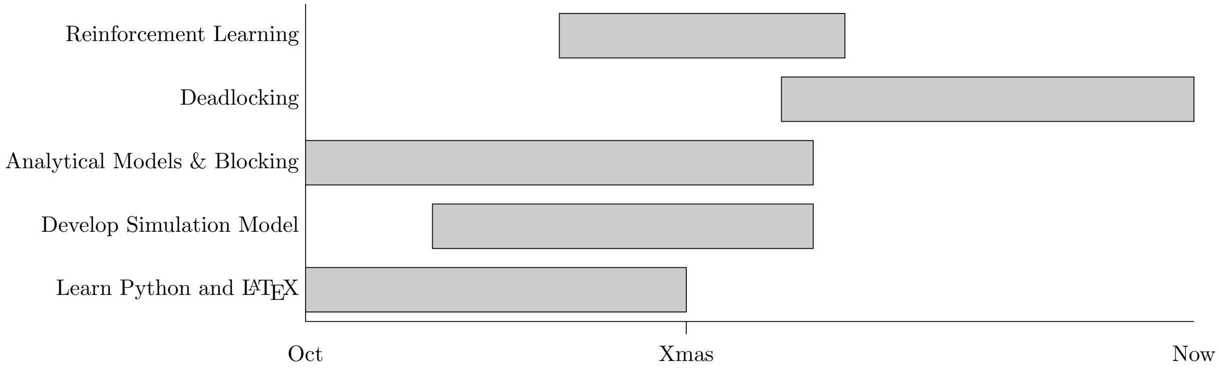


Figure 2: Gantt chart illustrating work so far.

the node recieve rewards according to the system's state or just its own state), and the difference between nodes observing the whole system's state and only observing its own state.

An extension of the above problem could be finding the optimal routing of patients when the agents are blind to their state, that is when we do not know, or do not care what state the system is in. Some results from game theory, and techniques such as genetic algorithms and linear programming can be explored.

1.3 Workforce Planning

The performance measures from each of the scenarios explored will be used to compare the workforce needs and skill mix required to care for frail and elderly patients across the ABUHB. Using methods such as acuity ratios and nurses-per-bed workforce needs can be estimated over time and across seasons. The aim is to find the best courses of actions to take in order to reduce the workforce gap that will extend over time, and gain an understanding of how changes to patient flows will affect the required skills and workforce for frail and elderly patients.

2 Progress & Learning

The Gantt chart in Figure 2 shown what I have been working on over the last nine months.

I have participated, or will be participating in the following activities:

- Oct 2014: Gave a talk on my MSc project at a SWORDS event, Cardiff.
- Feb 2015: Gave a talk at Python Namibia 2015 (http://python-namibia.org) at the University of Namibia in Windhoek.
- Mar 2015: NATCOR Stochastic Modelling course, Lancaster.
- Apr 2015: Data Mining course, Cardiff.

- May 2015: Gave a lightning talk on Linear Programming at the 'Developing Mathematical Models in Healthcare' seminar in Cardiff.
- May 2015: Gave a talk at the Wales Mathematics Colloquium at Gregynog.
- June 2015: On the organising committee of DjangoCon Europe 2015.
- July 2015: Will be giving a talk at EURO 2015, the European Conference on Operational Research, in Glasgow.
- Sept 2015: Am co-stream-organisers on the Stochastic Modelling stream of the Young OR 19 conference in Birmingham.

In addition to the above, I have been involved in with some teaching, assessing and marking for a number of modules on the the MSc programme, the second year module MA0261 and the first year module MA1003. I have participated in two hackathon weekends in order to develop my programming skills, and am a regular tutor for maths and stats support.

3 Literature Review

3.1 Queueing Networks

Queueing theory is an important branch of operational research, and the area of queueing networks have shown great promise in modelling computer and telecommunication systems and manufacturing procedures. An introduction to some of its results is given in [23]. The theory of a single queue has been widely investigated and developed, with Kendall's notation becoming the standard method of describing a queue. Kendall's notation describes a queue as $A/B/C/K/m/Z$: A is the arrival process, B the service process, C the number of servers, K the maximum capacity of the queue, m the size of the population served, and Z the service discipline. Service disciplines may be varied, however the main ones are First In First Out (FIFO), Last Come First Served (LCFS), First In Random Out (FIRO) and Process Sharing (PS) in which each customer in the queue is served for a proportion of the time until their service time in complete. If the last three parameters are left out, then it is assumed that $K = \infty$, $m = \infty$, and $Z = \text{FIFO}$.

Queueing networks are simply a number of the service centres with queues before them arranged into a network, with routing probabilities r_{ij} such that a customer finishing service at node i joins a queue at node j with probability r_{ij}. In [21] a history of their development is narrated, and an overview of the different types of networks studied. The simplest, open networks are those that have at least one node to which customers arrive from the exterior, and and least one node from which customers leave to the exterior. Closed networks are self-contained networks where no customers arrive or depart to or from the exterior, and as a result always contain a constant total of N customers in the system.

A fundamental result for the study of queues in series is shown in both [7] and [22], although derived in different ways. The result, known as Burke's Theorem, states that the departure rate of an $M/M/c$ service station is equal to its arrival rate. This result is fundamental to the study of queues arranged in series or networks. The latter paper notes that this result holds for more general service disciplines too, including LCFS, FIRO and systems where the number of servers varies with the number of customers present.

The study of a system of queues in series, or routed into a network, was first investigated in [14]. This paper investigates the simplest of cases, where all arrivals and service times are Markovian, the service discipline is FIFO, there is room for an infinite length queue between service centres, and the system is open. It is shown that in this case, each service centre, or node, behaves as an independent $M/M/c$ queue, which can be analysed as such, later known as the decomposition method.

This model is extended in [15] to include different classes of customer, each with their own routing probabilities around the network and own service times at each node. The analysis concentrates on obtaining steady-state probabilities for the system state defined as $C = (c_1, c_2, ..., c_J)$. Here for each node i, c_i is a specific ordering of classes of customer in the queue. A few extensions to this model are also presented, including changing the service discipline to PS and LCFS, prohibiting service from beginning until there is a certain amount of customers in the queue, and making service effort rely on the current system state rather than the number of people currently in that queue.

3.2 Other Things in Queueing Theory

More complicated customer behaviours have been studied in queueing theory, such as balking, defined as a customer arriving but deciding not to join the queue, therefore getting lost to the system; and reneging, defined as a customer entering a queue but then leaving before entering service. Queues with reneging are usually denoted with an extension to Kendall's notation, by $A/B/C/K/m/Z + D$, where D denotes the reneging method. In [2] and [3] the authors derive steady-state probabilities and mean values for an $M/M/1$ queue where customers exhibit balking and reneging. These papers give two separate customer behaviours for balking, first where customers balk with probability n/N when there are n customers in the system, $N - 1$ being an upper bound for queue size; and another where customers balk with probability $(1 - \beta/n)$ when there are n customers in the system, with $0 \leq \beta \leq 1$ being a measure of willingness to join the queue. In these papers a customer reneges after waiting time t with probability $\alpha e^{-\alpha t}$.

3.3 Queueing Networks with Blocking

Restricted queueing networks, or queueing networks with no or limited intermediate queues between service stations are more complicted to analyse. In these systems, if a customer finishes service at one node but is unable to join a queue at his destination node due to lack of queueing space, that customer remains in the current node, restricting his servers from starting the next customer's service. This is known as blocking. Since blocking introduces interdependencies between nodes, the product form solution of unrestricted networks is not appropriate. One of the first papers to consider these sorts of systems was [13]. Results are derived by writing out and solving the systems' difference equations. The same method is used in [5]. The thesis investigates two and three node systems, as well as systems with one service sentre with infinite queue routing into a number of these two and three node systems.

A two node system with no intermediate queue and blocking is studied in [4]. In this paper the moment generating functions of waiting time and number of customers in the system are derived, from which further performance measures can be obtained.

Two features of blocking are described in [16]: (*i*) patients completing service at a blocked station remain there until there is sufficient queueing space at the next station, and (*ii*) these patients block other patients from entering that station. If a station only has characteristic (i) then it is referred to as 'classic congestion', and if a station has both characteristics it is referred to as 'blocking'. Three blocking situations are studied in [19], defined by rules on the system reaching 'full blocking' and their 'unblocking rule'. If the node that is subject to blocking has r parallel servers, then once r^* $(1 \leq r \leq r^*)$ servers become blocked all remaining unblocked servers stop service and the node becomes fully blocked. Given that there is a room for M customers to wait between nodes, once there are only k^* $(0 \leq k^* \leq M + r^* - 2)$ customers waiting between the nodes all services may start again and the node becomes unblocked.

And approximation method for solving queueing networks with downstream blocking only in presented in [26]. The algorithm finds the mean values of a queueing network with feedforward flows and single server nodes. Iteratively working from the node furthest downstream and working backwards, if that station does exhibit blocking it finds the *effective service time*, that is the weighted sum of service time and the mean time blocked and waiting to transition to the next node, and computes the effective service time for the next upstream node with a recursive formula. This method is adapted to multi-server queues in [16], and a similar iterative method was used in [17].

Restriced queueing networks can give rise to the phenomenon of deadlock. In the simplest of cases, deadlock occurs when a customer finishes service at node i and is blocked from transitioning to node j; however the individuals in node j are all blocked, directly or indirectly, by the blocked individual in node i. This can cause problems for both analytical and simuation models, and most of the literature on blocking conveniently assumes the networks are deadlock-free. For closed networks of K customers with only one class of customer, [18] proves the following condition to ensures no deadlock: for each minimum cycle C, $K < \sum_{j \in C} B_j$, the total number of customers cannot exceed the total queueing capacity of each minimum subcycle of the network. The paper also presents algorithms for finding the minimum queueing space required to ensure deadlock never occurs, for closed cactus networks, where no two cycles have more than one node in common. This result is extended to multiple classes of customer in [20], with more restrictions such as single servers and each class having the same service time distribution. Here a integer linear program is formulated to find the minimum queueing space assignment that prevents deadlock. The literature does not discuss deadlock properties in open restricted queueing networks.

3.4 Relevant OR Healthcare Stuff

Operational research techniques are well suited to be applied to the kinds of problems that arise in the health care system. This view is supported in [6]. This article describes the problems experienced in health care settings as mostly dealing with uncertainty or variability in the demand of services and resources, which may be analysed using standard operational research methods including queueing theory and simulation. Other health care issues lending themselves to OR listed include scheduling and capacity planning problems. The nature of healthcare systems is described in [12]; there is complexity, for example different flow rules for different patients at different times; there is uncertainty for example in demand, which can vary with month of the year, day of the week and even hour of the day; there is variability, for example the attributes and length of stay distributions patient classes can be vastly different; and the all parts of the system are highly integrated with other parts of the hospital and other hospitals in the region. The paper builds a framework for building OR tools and models, which should be flexible and versatile to be applied to more than one specific area of the system, easy to use and able to answer ”what if?” scenarios, integrated with all other parts of the system, and be validated with data.

Whole hospital and whole systems approaches are sometimes necessary when modelling and optimising healthcare systems in order to capture and investigate patient flows, up- and downstream resources, and identifying bottlenecks in the structures. A review of the operational research literature that involves multi-departmental analysis in presented in [28]. Here the authors conclude that there is an emphasis in the literature on the interaction between the wards, the emergency department and the surgery departments.

A whole system simulation is built in [29], bringing together eye clinics and social care to model elderly patients with age-related macular degeneration. This model is novel as it combines discrete event simulation, systems dynamics and agent-based modelling in one model in order to model patient flow through the eye clinic, patients' sight decay, and their social care respectively.

3.5 Queueing Networks Applied to Healthcare

The use of queueing theory for healthcare purposes is widespread. A geriatric ward is modelled as an $M/PH/c$ queue in [11], a multi server queue with phase-type distribution where no queue is allowed to build up, i.e. patients are sent elsewhere and lost to the system if all c beds are occupied. This model is used to find the optimal number of beds in order to reduce the cost of turning away patients and maintaining empty beds.

Queueing networks have been used to model healthcare systems. In [1] a health centre is modelled as an open queueing network with eight nodes representing five care-givers and three areas prior to seeing a doctor or nurse, in order to assess how to improve waiting times. The model proved successful and disproved a widely held belief that the front desk was the system bottleneck, concluding that including time spent prior to seeing a care-giver in appointment times would reduce waits. In [10] the orthopaedics department of the Middelheim hospital in Antwerpen was modelled as an open queueing network with five service centres and eighteen patient classes. Both preemptive and nonpreemptive interruptions were incorporated into the service times of one node. Three analytical methods of finding flow times were compared: two formulas applied to the decomposition method, Kingman and Whitt, and then by using the Brownian model. These were compared to a simulation model of the system, where it was concluded that the decomposition method far outperformed the Brownian model, and the Kingman formula exerted slightly better results than the Whitt formula.

In [1] the authors list a number of reasons why queueing network models do not match real-life healthcare situations. Queueing network analysis generally focus on steady-state statistics, though in situations where service starts and needs each day the system is reset every day and rarely, if ever reach steady-state. Queueing analysis also assumed that arrival and service times are independent of one another, yet if an appointment system is used this is not the case. However, the model used yielded realistic results and some useful conclusions and recommendations. In [16] a queueing network with upstream blocking is used to model the flow of mental health patients through a care system in Philadelphia, where service stations are housing facilities: extended acute hospitals, residential facilities and supported housing. Its aim was to find an alternative approach to the needs assessment approach of capacity planning, however it was found that the queueing network model with static arrival rates could not effectively model the real-world as customer behaviour exhibited reneging based on waiting times.

3.6 Reinforcement Learning

The concept of machines learning from experience stems from the first papers on computing, notably [27]. In this early paper the idea is discussed that actions chosen at random will be repeated more often if that action resulted in a reward, and less often if that action resulted in punishment. This forms the basis of reinforcement learning.

As subset of unsupervised machine learning, reinforcement learning is a computational way of learning and decision-making through goal seeking means. Comprehensive overviews are given in [24] and [25].

4 Analytical Models of Open Queueing Networks

NEED TO WRITE THIS. GIVE A WORKED EXAMPLE OF JACKSON NETWORK, DIFFERENCE EQUATIONS FOR BLOCKING, MARKOV CHAIN FOR BLOCKING, TAKAHASHI APPORXIMATION THING.

5 Deadlocking Properties of Open Restircted Queueing Networks

This section will define and discuss the properties and detection of deadlock in queueing networks. Throughout the section, when discussing queueing networks, it is assumed that the queueing network is open and connected. Open queueing networks are those networks that have at least one node to which customers arrive from the exterior, and and least one node from which customers leave to the exterior.

5.1 Deadlock

Definition 1. *When a simulation is in a situation where at least one service station, despite having arrivals, ceases to finish any more services due to recursive upstream blocking the system is said to be in deadlock.*

Deadlock can be experienced in any open queueing network that experiences blocking, with at least once cycle containing all service stations with resticted queueing capacity.

Deadlock occurs when a customer finishes service at node i and is blocked from transitioning to node j; however the individuals in node j are all blocked, directly or indirectly, by the blocked individual in node i. That is, deadlock occurs if every individual blocking individual X, directly or indirectly, are also blocked.

In Figure 3 a simple two node queueing network is shown in a deadlocked state. Customer e has finished service at node 1, but remains there as there is not enough queueing space at node 2 to accept them. We say customer e is blocked by customer i, as he is waiting for customer i to be released. Similarly, customer i has finished service at node 2, but remains there as there is not enough queueing space at node 1, customer i is blocked by customer e.

When there are multiple servers, individuals become blocked by all customers in service. at the destination service station. Figure 4a shows two nodes in deadlock, customer i is blocked by both d and e, who are both blocked by customer i. However in 4b, customer i is blocked by both d and e, and customer d isn't blocked, and so there is no deadlock.

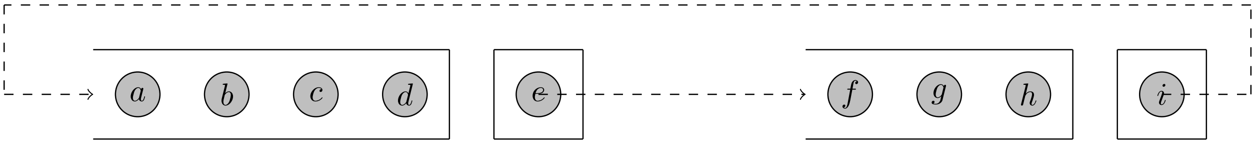
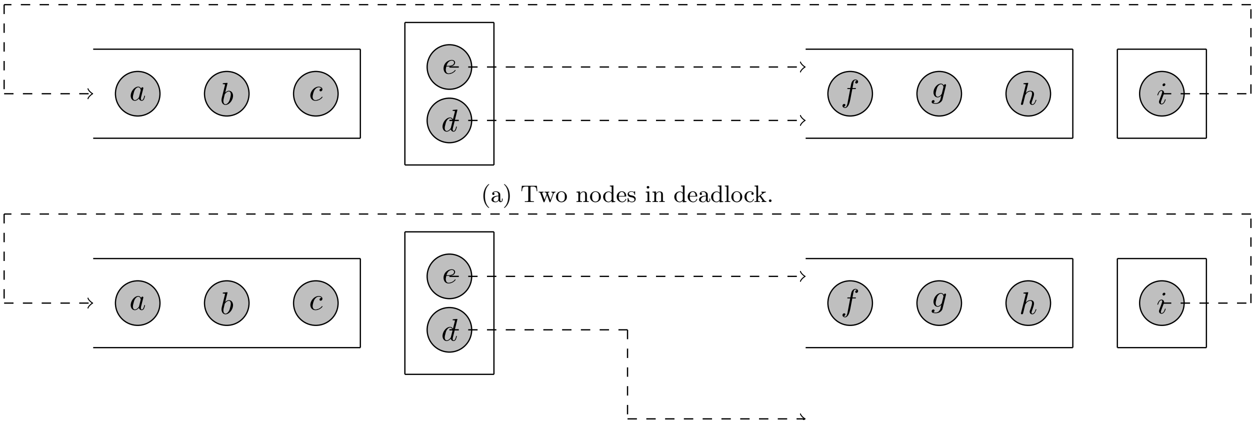


Figure 3: Two nodes in deadlock.



(a) Two nodes in deadlock.

(b) Two nodes not deadlocked

Figure 4: Two nodes: a) in deadlock and b) not in deadlock.

Note that the whole queueing network need not be deadlocked, only a part of it. If one section of the network is in deadlock, then the system is deadlocked, even though customers may still be able to have services and transitions in other areas of the network. An example is shown in Figure 5. Here nodes 1 and 2 are in deadlock, so individuals e and h cannot transition to the next node as they are blocking one another. Individual k on the other hand is free to move to its next destination. This idea is expanded on in the next section.

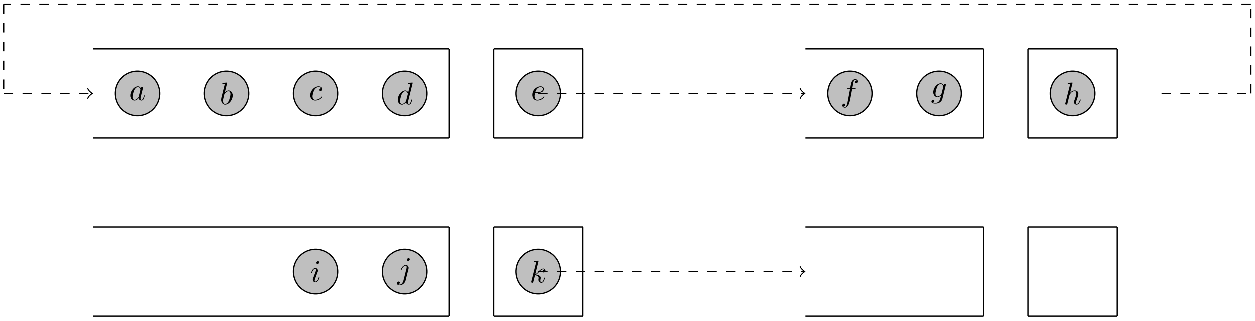


Figure 5: A deadlock situation where not all nodes are deadlocked.

5.2 Types of Deadlock

The previous section introduced the idea that parts of a queueing network can be in a deadlocked state, although other parts will continue to flow. The different configurations of which nodes experience deadlock can be thought of as different types of deadlock. Each different type of deadlocked state can be denoted $(i_1, i_2, i_3...)$ where $i_k = -1$ if node k node is participating in that deadlocked state. The amount of different types of deadlock that a queueing network can experience is equal to the number of directed cycles in in the queueing network's routing matrix.

For connected queueing networks, these deadlocks can be classified into transient deadlocked states and the absorbing deadlocked state.

Definition 2. *A transient deadlock state is when there are still some changes of state whilst a subgraph of the queueing network is itself in deadlock.*

Definition 3. *The absorbing deadlock state is when all subgraphs of the queueing network are in deadlock.*

For a queueing network Q with N service stations, the absorbing deadlocked state corresponds to $(i_1 = -1, ..., i_N = -1)$, the state where all service stations experience deadlock. It should be clear that if the queueing network is connected, then there is a non-zero probability that once one part of the network is in deadlock, the whole system will fall into a deadlocked state, simply by the individuals in the non-deadlocked nodes attempting to transition into a deadlocked node. That is, once Q falls into one of the transient deadlocked states, it will eventually transition, either directly or through other transient deadlocked states, into the absorbing deadlocked state.

If the routing matrix of Q is complete, that is there is a possible route from every service station to every other service station, then there are $\sum_{i=1}^{N} \binom{N}{i}$ possible deadlock types.

5.3 Literature Review

Most of the literature on blocking conveniently assumes the networks are deadlock-free. For closed networks of K customers with only one class of customer, [18] proves the following condition to ensures no deadlock: for each minimum cycle C, $K < \sum_{j \in C} B_j$, the total number of customers cannot exceed the total queueing capacity of each minimum subcycle of the network. The paper also presents algorithms for finding the minimum queueing space required to ensure deadlock never occurs, for closed cactus networks, where no two cycles have more than one node in common. This result is extended to multiple classes of customer in [20], with more restrictions such as single servers and each class having the same service time distribution. Here a integer linear program is formulated to find the minimum queueing space assignment that prevents deadlock. The literature does not discuss deadlock properties in open restricted queueing networks.

General deadlock situations that are not specific to queueing networks are discussed in [9]. Conditions for this type of deadlock, also referred to as deadly embraces, to potentially occur are given:

- Mutual exclusion: Tasks have exclusive control over resources.
- Wait for: Tasks do not release resources while waiting for other resources.
- No preemption: Resources cannot be removed until they have been used to completion.
- Circular wait: A circular chain of tasks exists, where each task requests a resource from another task in the chain.

Dynamic state-graphs are defined, with resources as vertices and requests as edges. For scenarios where there is only one type of each resource, deadlock arises if and only if the state-graph contains a cylce.

In [8] the vertices and edges of the state graph are given labels in relation to a reference node. Using these labels *simple bounded circuits* are defined whose existance within the state graph is sufficient to detect deadlock.

5.4 Definitions

$|V(D)|$ The order of the directed graph D is its number of vertices.

Weakly connected component A weakly connected component of a digraph containing X is the set of all nodes that can be reached from X if we ignore the direction of the edges.

Direct successor If a directed graph contains an edge from X_i to X_j , then we say that X_j is a direct successor of X_i.

Ancestors If a directed graph contains a path from X_i to X_j , then we say that X_i is an ancestor of X_j.

Descendants If a directed graph contains a path from X_i to X_j , then we say that X_j is a descendant of X_i.

$\deg^{\text{out}}(X)$ The out-degree of X is the number of outgoing edges emanating from that vertex.

Subgraph A subgraph H of a graph G is a graph whose vertices are a subset of the vertex set of G, and whose edges are a subset of the edge set of G.

Sink vertex A sink vertex is a vertex in a directed graph that has out-degree of zero.

Knot In a directed graph, a knot is a set of vertices with out-edges such that while traversing the directed edges of that directed graph, once a vertex in the knot is reached, you cannot rech any vertex that is not in the knot.

NEED CONSISTANT NOTATION, REFERENCES FOR DEFINITIONS.

5.5 State Digraph

Presented is a method of detecting when deadlock occurs in an open queueing network Q with N nodes, using a dynamic directed graph, the state graph.

Let the number of servers in node i be denoted by c_i. Define $D(t) = (V(t), E(t))$ as the state graph of Q at time t.

The vertices at time t, $V(t)$ correspond to servers in the queueing system. Thus, $|V(D(t))| = \sum_{i=1}^{N} c_i$ for all $t \geq 0$.

The edges at time t, $E(t)$ correspond to a blocking relationship. There is a directed edge at time t from vertex $X_a \in V(t)$ to vertex $X_b \in V(t)$ if and only if an individual occupying the server corresponding to vertex X_a is being blocked by an individual occupying the server corresponding to vertex X_b.

The state graph $D(t)$ can be partitioned into N service-station subgraphs, $D(t) = \bigcup_{i=1}^{N} d_i(t)$, where the vertices of $d_i(t)$ represent the servers of node i. The vertex set of each subgraph is static over time, however their edge sets may change.

The state graph is dynamically built up as follows. When an individual finishes service at node i, and this individual's next destination is node j, but there is not enough queueing capacity for j to accept that individual, then that individual remains at node i and becomes blocked. At this point c_j directed edges between this individual's server and the vertices of $d_j(t)$ are created in $D(t)$.

When an individual is released and another customer who wasn't blocked occupies their server, that servers out-edges are removed. When an individual is released and another customer who was previously blocked occupies their server, that server's out-edges are removed along with the in-edge from the server who that previously blocked customer occupied. When an individual is released and there isn't another customer to occupy that server, then all edges incident to that server are removed.

This general process of building up the state graph as the queueing network is simulated will now be shown. Customers are labelled (i, j, k) where i denotes the server that customer is occupying, j denotes that individuals i.d. number, and k denotes the service station that customer is waiting to enter. As an example, a customer labelled $(A_2, 10, C)$ would have an i.d. number of 2, is occupying server A_2 and is currently waiting to join node C. If a customer isn't occupying a server the notation $\emptyset$ is used. Similarly for customers occupying a server and still in service, their next destination is yet undecided, so $\emptyset$ is used.

The simulation starts with full queues, and every server occupied by a customer in service. This is shown in Figure 6.

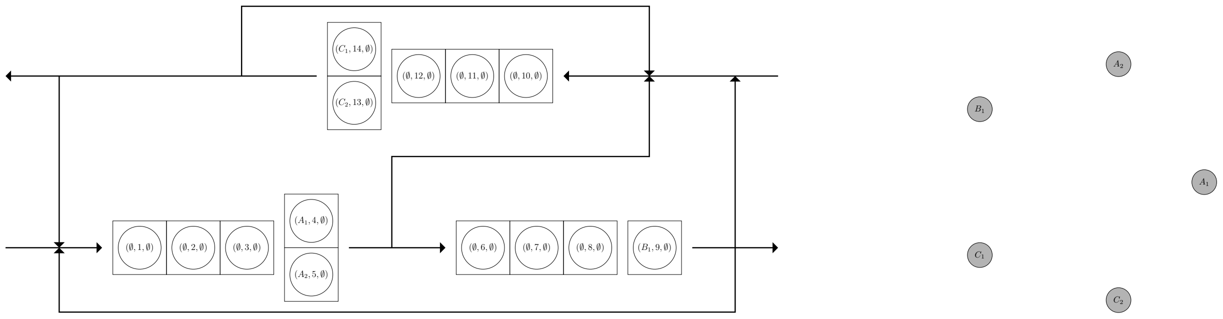

$(C_1, 14, \emptyset)$
$(C_2, 13, \emptyset)$
$(\emptyset, 12, \emptyset)$
$(\emptyset, 11, \emptyset)$
$(\emptyset, 10, \emptyset)$
$(A_1, 4, \emptyset)$
$(A_2, 5, \emptyset)$
$(\emptyset, 1, \emptyset)$
$(\emptyset, 2, \emptyset)$
$(\emptyset, 3, \emptyset)$
$(\emptyset, 6, \emptyset)$
$(\emptyset, 7, \emptyset)$
$(\emptyset, 8, \emptyset)$
$(B_1, 9, \emptyset)$
A_2
B_1
A_1
C_1
C_2

Figure 6

Customer 13 finishes service, and is blocked from entering node A. Figure 7.

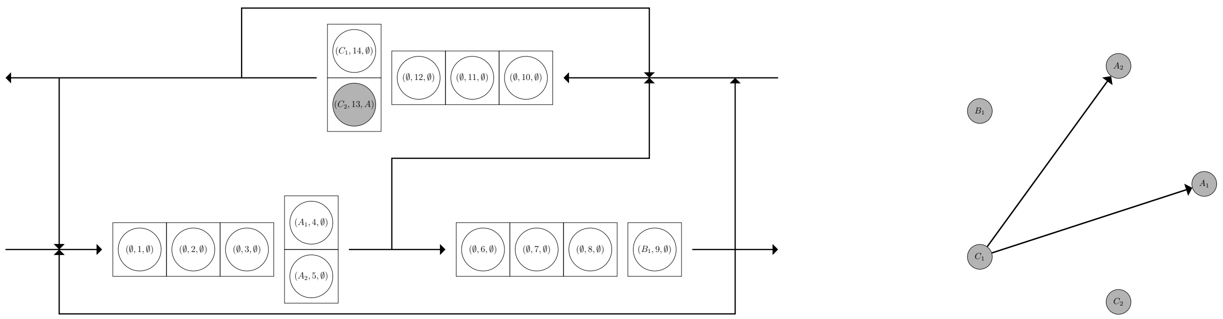

$(C_1, 14, \emptyset)$
$(C_2, 13, A)$
$(\emptyset, 12, \emptyset)$
$(\emptyset, 11, \emptyset)$
$(\emptyset, 10, \emptyset)$
$(A_1, 4, \emptyset)$
$(A_2, 5, \emptyset)$
$(\emptyset, 1, \emptyset)$
$(\emptyset, 2, \emptyset)$
$(\emptyset, 3, \emptyset)$
$(\emptyset, 6, \emptyset)$
$(\emptyset, 7, \emptyset)$
$(\emptyset, 8, \emptyset)$
$(B_1, 9, \emptyset)$
A_2
B_1
A_1
C_1
C_2

Figure 7

Then customer 4 finishes service and is blocked from entering node B. Figure 8.

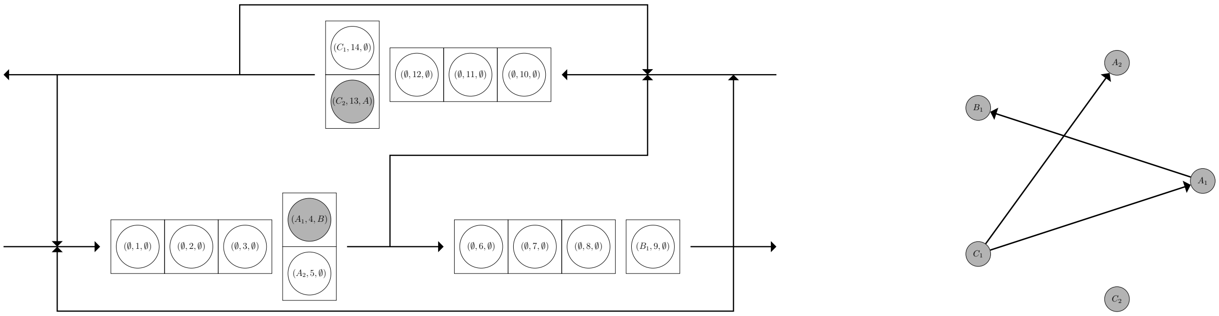


Figure 8

Then customer 9 finishes service and is blocked from entering node A. Figure 9.

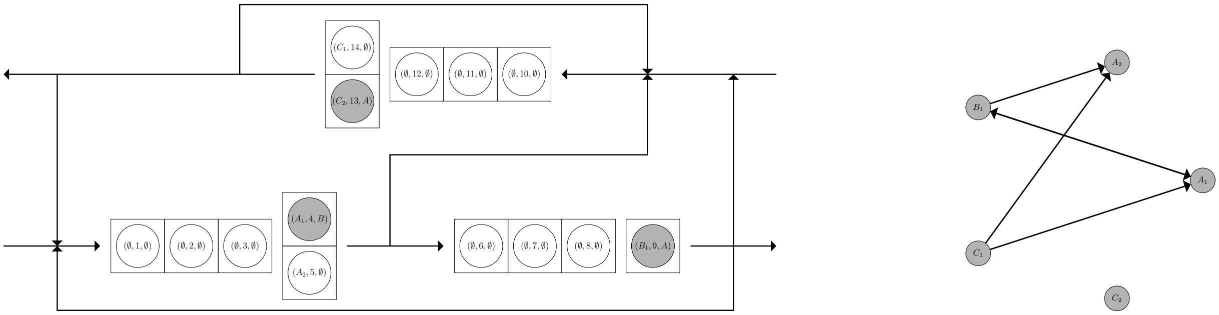


Figure 9

Finally, in Figure 10 customer 5 finishes service and wants to reenter the queue for node A but is blocked. A deadlock situation arises as customer 5 is waiting for customer 4 to move, who is waiting for customer 9 to move, who is waiting for either customer 4 or 5 to move.

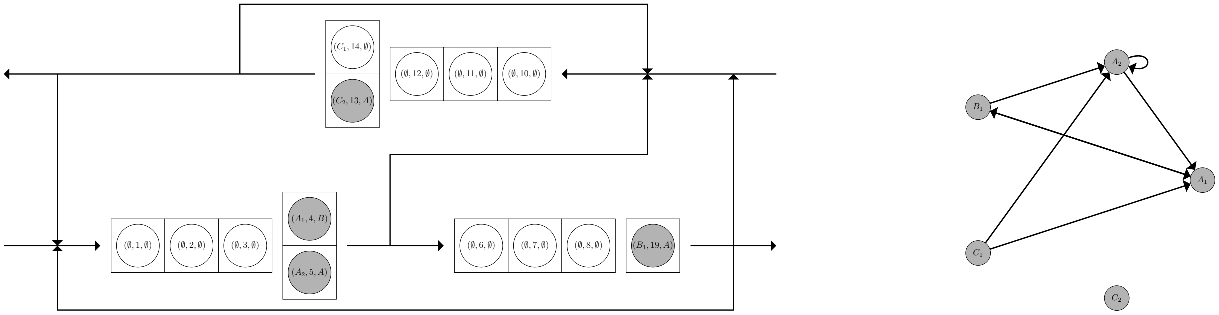
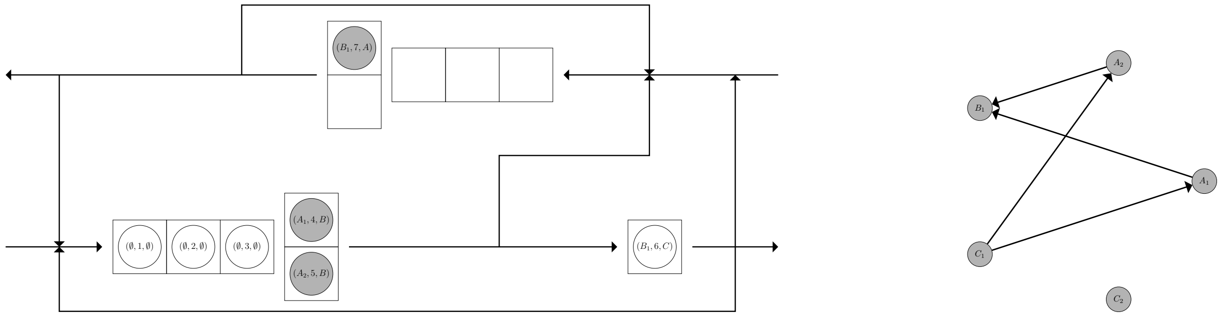


Figure 10

The rules on how edges are removed from the state graph will now be shown. For illustrative purposes the queueing network here is a different queueing network than discussed above.

Here the simulation begins with four customers occupying servers; those at node A blocked to node B, the customer at node C blocked to node A, and the customer at node B still in service. This is shown in Figure 11.



$(B_1, 7, A)$
$(A_1, 4, B)$
$(A_2, 5, B)$
$(\emptyset, 1, \emptyset)$
$(\emptyset, 2, \emptyset)$
$(\emptyset, 3, \emptyset)$
$(B_1, 6, C)$
A_2
B_1
A_1
C_1
C_2

Figure 11

Customer 6 finishes service and immediately joins service at node C. Figure 12.

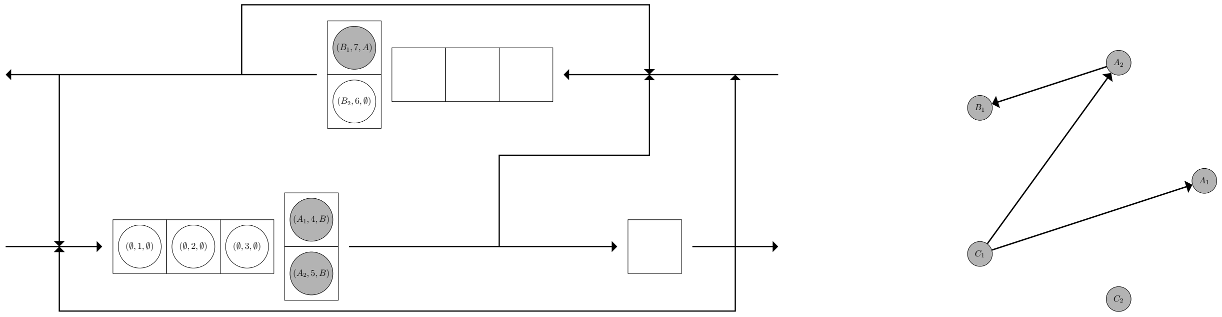


Figure 12

Now there is room for customer 4 to move into service at node B. Figure 13. Notice that the edge $A_2 \longrightarrow B_1$ remains in the state graph, as customer 5 is still blocked by that server.

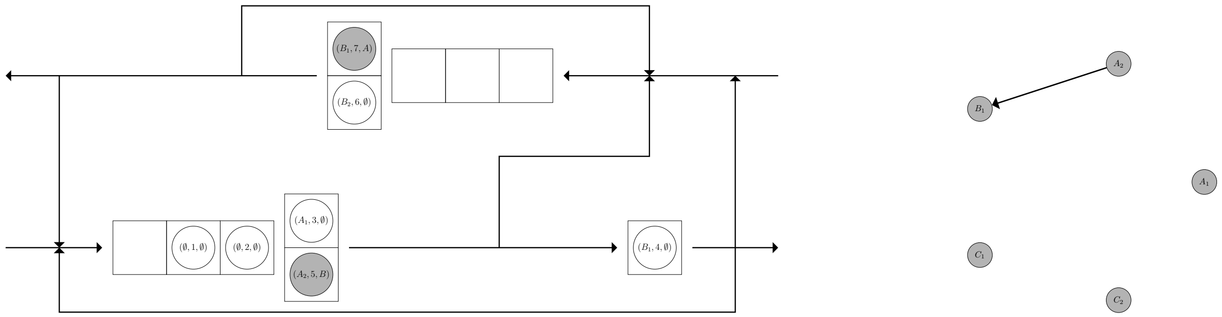


Figure 13

The customers queueing at node A move along the queue, with customer 3 begining service. This leaves enough room for customer 7 to join the back of the queue at A. Figure 14.

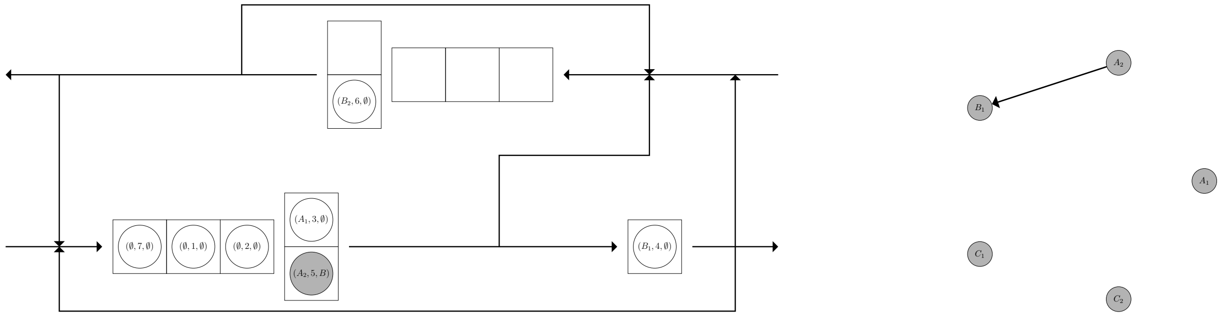


Figure 14

Observations

Consider one weakly connected component $G(t)$ of $D(t)$. Consider the node $X_a \in G(t)$. If X_a is unoccupied, then X_a has no incident edges. Consider the case when X_a is occupied by individual a, whose next destination is node j. Then X_a's direct successors are the servers occupied by individuals who are blocked or in service at node j. We can interpret all X_a's decendents as the servers whose occupants are directly or indirectly blocking a, and we can interpret all X_a's ancestors as those servers whose individuals who are being blocked directly or indirectly by a.

Note that the only possibilities for $\text{deg}^{\text{out}}(X_a)$ are being 0 or c_j. If $\text{deg}^{\text{out}}(X_a) = c_j$ then a is blocked by all its direct successors. The only other situation is that a is not blocked, and $X_a \in G(t)$ because a is in service at X_a and blocking other individuals, in which case $\text{deg}^{\text{out}}(X_a) = 0$.

It is clear that if all of X_a's descendents are occupied by blocked individuals, then the system is deadlocked at time t. We also know that by definition all of X_a's ancestors are occupied by blocked individuals.

Also note that if a service-station subgraph $d_i(t)$ contains edges, then there is an individual in $X_a \in d_i(t)$ that is being blocked by himself. This does not necassarily mean there is deadlock.

5.6 Theorem

Theorem 1. *A deadlocked state arises at time t if and only if $D(t)$ contains a knot.*

Proof. Consider one weakly connected component $G(t)$ of $D(t)$ at time t. All vertices of $G(t)$ are either decendents of another vertex and so are occupied by an individual who is blocking someone; or are ancestors of another vertex, and so are occupied by someone who is blocked.

Assume that $G(t)$ contains a vertex X such that $\text{deg}^{\text{out}}(X) = 0$, and there is a path from every other non-sink vertix to X. This imples that X's occupant is not blocked and is a descendent of another vertrex. Therefore Q is not deadlocked as there does not exist a vertex whose descendents are all blocked.

Now assume that we have deadlock. For a vertex X who is deadlocked, all decendents of X are are occupied by individuals who are blocked, and so must have out-degrees greater than 0. And so there is no path from X to a vertex with out-degree of 0. □

Lemma 1. *For a queueing network with two nodes or less, a deadlocked state arises if and only if there exists a weakly connected component without a sink node.*

Proof. Consider a one node queueing network Q_1.

If there is deadlock, then all servers are occupied by blocked individuals, and so all servers have an out-edge.

Consider a two node queueing network Q_2.

If both nodes are involved in the deadlock, so there is a customer in node 1 blocked from entering node 2, and a customer from node 2 blocked from entering node 1, then all servers in node 1 and node 2 in $D(t)$ will have out edges as they are occupied by a blocked individual. The servers of node 1 and 2 consist of the entirety of $D(t)$, and so there is no sink nodes.

Now consider the case when only one node is involved in the deadlock. Without loss of generality, let's say that node 1 is in deadlock with itself, then the servers of node 1 have out-edges. For the servers of node 2 to be part of that weakly conneced component, there either needs to be an edge from a server in node 1 to a server in node 2, or and edge from a server in node 2 to a server in node 1. An edge from a server in node 1 to a server in node 2 implies that a customer from node 1 is blocked from entering node 2, and so node 1 is not in deadlock with itself. An edge from a server in node 2 to a server in node 1 implies that a customer in node 2 is blocked from entering node 1. In this case the server in node 2 has an out-edge, and so there is still no sink.

For the case of a queueing network with more than two nodes, the following counter-example proves the claim:

Begin with all servers occupied by customers in service. The customer at server B_1 is blocked from entering node A. Then the customer at server C_1 is blocked from entering node B. Then the customer at server A_1 is blocked from entering node A. The resulting state digraph in Figure 15 has a weakly connected component with a sink. □

Lemma 2. *An absorbing deadlocked state arises at time t if $D(t)$ doesn't contain a sink vertex.*

Proof. A vertex with out-degree greater than zero represents an occupied server whose occupant has finished service and is blocked. If all vertices have out-degree greater than zero, then all servers are occupied by blocked individuals. A release at vertex X_a can only be triggered by one of X_a's descendents finishing service. As all servers are occupied by blocked individuals, no server can finish service, and so no server can release their occupant, implying an absorbing deadlocked state. □

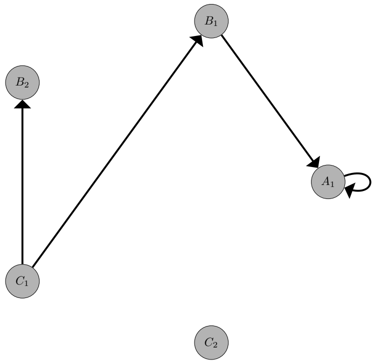


Figure 15: A Counter-Example State Digraph.

5.7 Finding Knots

Here is an initial draft of an algorithm for dynamically finding a knot in the state digraph of the queueing network.

Theorem 2. *The existance of a knot in the state digraph is equivalent to the existance of a vertex with an out-edge but without a path to a sink.*

Proof. Consider a vertex X that has an out-edge, but does not have a path to a sink. All of X's descendants also do not have paths to sinks.

As the number of vertices in $D(t)$ are finite ($|V(D(t))| = \sum_{i=1}^{N} c_i$ for all $t \geq 0$), then all the possible paths leading out from X must either be part of a cycle of lead to a cycle. And so all paths from X must terminate in a knot. □

Keeping track of the list of sinks in D will be useful. Let's call this list sink_list. Initially all vertices of $D(0)$ are sinks, and belong to sink_list. At the point when a vertex gains an out-edge, that vertex is removed from sink_list. At the point when a vertex has an out-edge removed, if $\deg^{\text{out}} = 0$ then that vertex is added to sink_list. Therefore, at any one point we have knowledge of all the sinks in $D(t)$.

Now, the only action that can lead to a vertex transitioning from having a path to a sink and not having a path to a sink is adding an edge. Removing an edge of $D(t)$ will not cause deadlock, as removing an edge implies that a customer has moved, and a customer moving cannot cause deadlock. Therefore, we only need to check the vertex's paths to sinks when an edge is added.

5.8 Markov Chain Model

It is interesting to build an analytical model of the system's behaviour to deadlock. As a Markov chain model the deadlocking state is an absorbing state, and so any queueing network that can experience deadlock is guaranteed to experience deadlock.

We can however find the expected time until deadlock is reached. It is shown in [23] that for a discrete transition matrix of the form $P = \left(\begin{smallmatrix} T & U \\ 0 & V \end{smallmatrix}\right)$ then the expected number of time steps until absorbtion starting from state i is the ith element of the vector

$$(I-T)^{-1}e \tag{1}$$

where e is a vector of 1s.

5.8.1 One Node Network

Consider the one node network with feedback loop shown in Figure 16. There is room for n customers to queue at any one time, customers arrive at a rate of Λ and served at a rate μ. Once a customer has finished service he rejoins the queue with probability r_{11}, and so exits the system with probability $1 - r_{11}$.

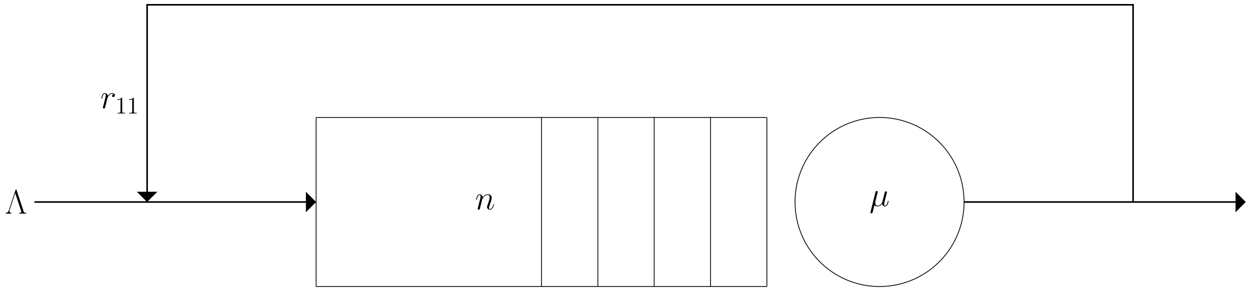


Figure 16: A one node queueing network.

State space:

$$S = \{i \in \mathbb{N} | 0 \leq i \leq n+1\} \cup \{-1\}$$

Where i denotes number of individuals in service or waiting.

If we define $\delta = i_2 - i_1$ for all $i_k \in S | i_k \geq 0$, then the transitions are given by:

$$q_{i_1,i_2} = \begin{cases} \left.\begin{matrix} 0 & \text{if } i = n+1 \\ \Lambda & \text{otherwise} \end{matrix}\right\} & \text{if } \delta = 1 \\ (1-r_{11})\mu & \text{if } \delta = -1 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

$$q_{i,-1} = \begin{cases} r_{11}\mu & \text{if } i = n+1) \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

and

$$q_{-1,i} = 0 \tag{4}$$

The Markov chain is shown in Figure 17.

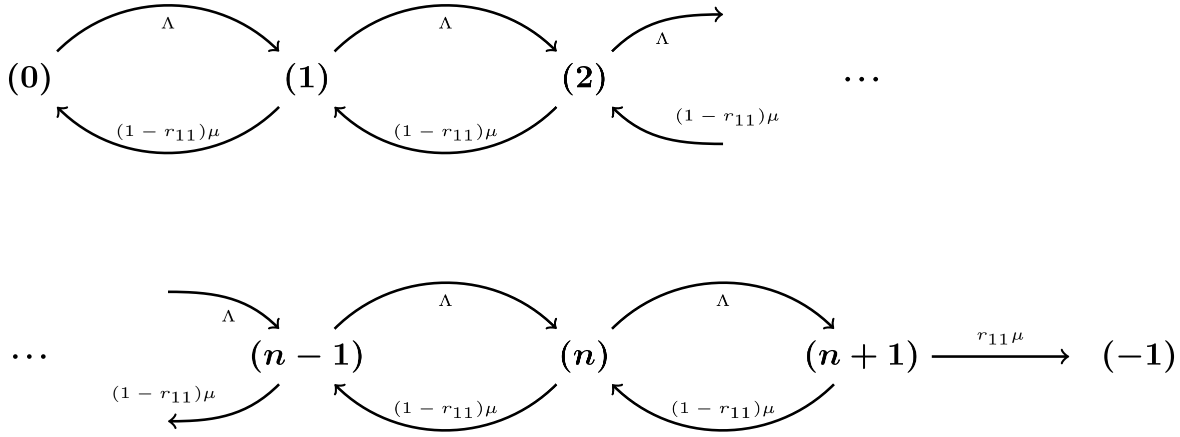


Figure 17: Markov chain of the one node system.

5.8.2 Two Node Network without Self Loops

Consider the queueing network shown in Figure 18. This shows two $M/M/1$ queues, with n_i queueing capacity at at each service station and service rates μ_i. Λ_i is the external arrival rates to each service station. All routing possibilities except self loops are possible, where the routing probability from node i to node j is denoted by r_{ij}.

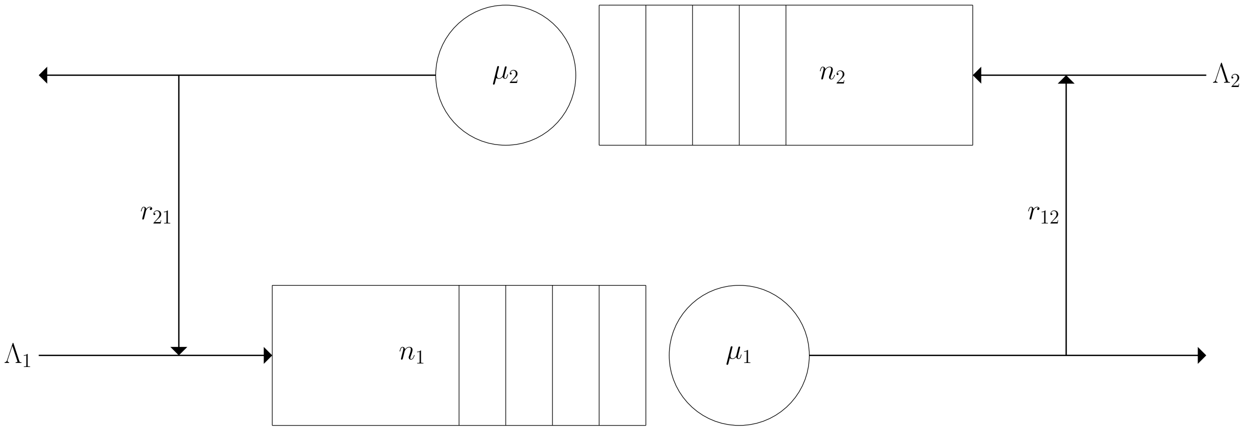


Figure 18: A two node queueing network.

- State space:

$$S = \{(i,j) \in \mathbb{N}^{(n_1+2\times n_2+2)} | 0 \leq i+j \leq n_1+n_2+2\} \cup \{(-1,-1)\}$$

  Where i denotes number of individuals:

  - In service or waiting at the first node.
  - Occupying a server but having finished service at the second node waiting to join the first

Where j denotes number of individuals:

- In service or waiting at the second node.
- Occupying a server but having finished service at the first node waiting to join the first

and the state $(-1,-1)$ denotes the deadlocked state.

If we define $\delta = (i_2, j_2) - (i_1, j_1)$ for all $(i_k, j_k), s \in S | i_k, j_k \geq 0$, then the transitions are given by:

$$q_{(i_1,j_1),(i_2,j_2)} = \begin{cases} \left.\begin{matrix} \Lambda_1 & \text{if } i_1 \leq n_1 \\ 0 & \text{otherwise} \end{matrix}\right\} & \text{if } \delta = (1,0) \\ \left.\begin{matrix} \Lambda_2 & \text{if } j_1 \leq n_2 \\ 0 & \text{otherwise} \end{matrix}\right\} & \text{if } \delta = (0,1) \\ \left.\begin{matrix} 0 & \text{if } j_1 = n_1 + 2 \\ (1 - r_{12})\mu_1 & \text{otherwise} \end{matrix}\right\} & \text{if } \delta = (-1,0) \\ \left.\begin{matrix} 0 & \text{if } i_1 = n_1 + 2 \\ (1 - r_{21})\mu_2 & \text{otherwise} \end{matrix}\right\} & \text{if } \delta = (0,-1) \\ \left.\begin{matrix} 0 & \text{if } j_1 = n_2 + 2 \\ r_{12}\mu_1 & \text{otherwise} \end{matrix}\right\} & \text{if } \delta = (-1,1) \\ \left.\begin{matrix} 0 & \text{if } i_1 = n_1 + 2 \\ r_{21}\mu_2 & \text{otherwise} \end{matrix}\right\} & \text{if } \delta = (1,-1) \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

$$q_{(i_1,j_1),(-1,-1)} = \begin{cases} r_{21}\mu_2 & \text{if } (i,j) = (n_1, n_2 + 2) \\ r_{12}\mu_1 & \text{if } (i,j) = (n_1 + 2, n_2) \\ 0 & \text{otherwise} \end{cases} \tag{6}$$

and

$$q_{-1,s} = 0 \tag{7}$$

For $n_1 = 1$ and $n_2 = 2$, the resulting Markov chain is shown in Figure 19.

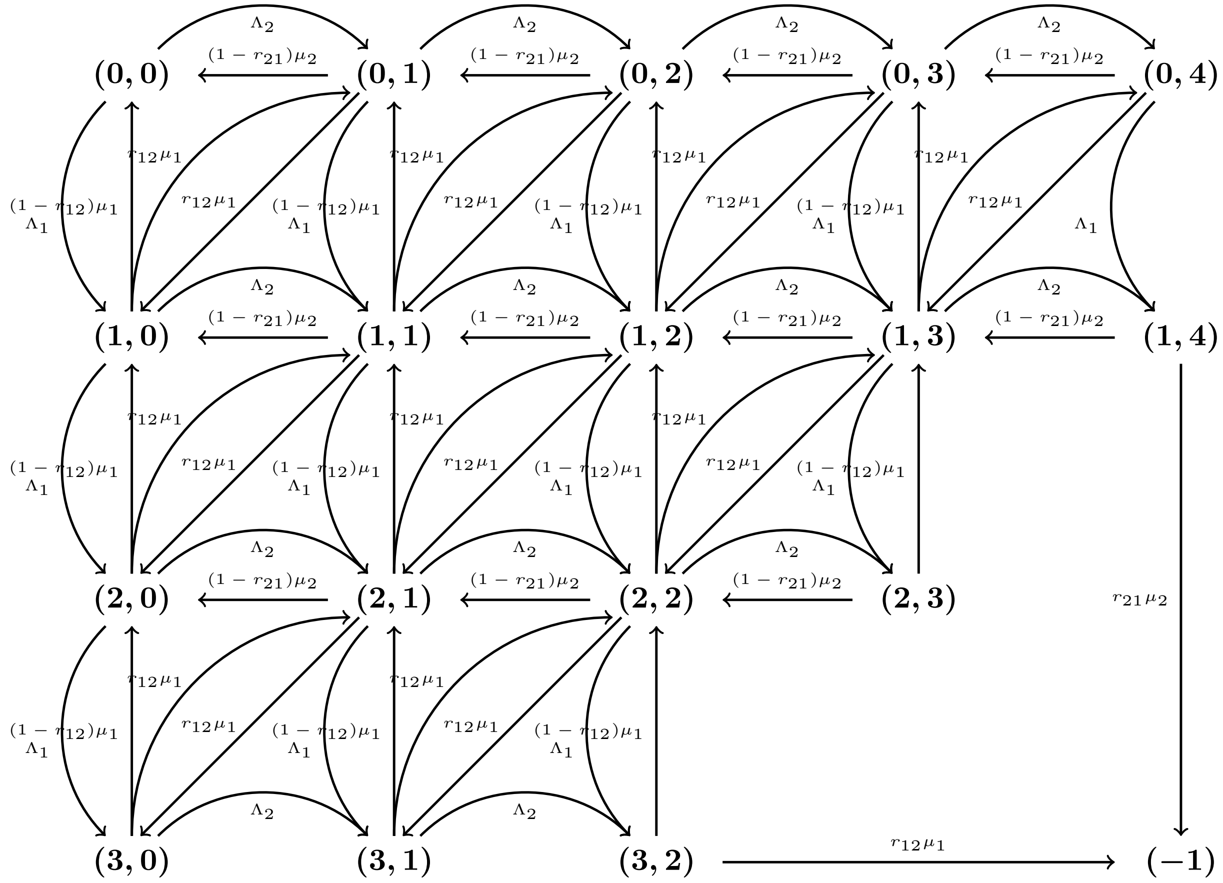
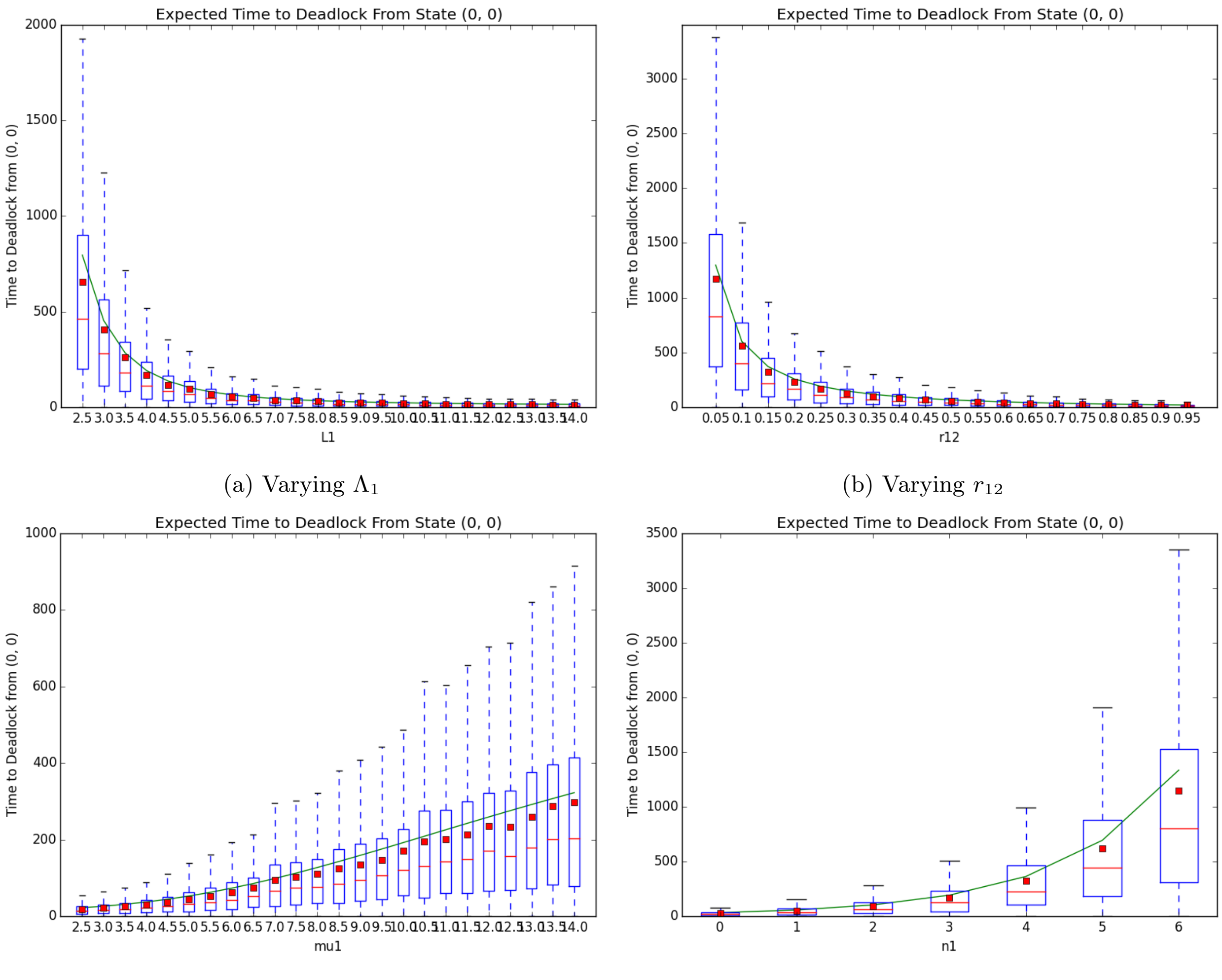


Figure 19: Markov chain of the two node system with $n_1 = 1$ and $n_2 = 2$.

Figure 20 shows the effect of varying the parameters of the above Markov model. Only parameters for one node are shown, as the other node's parameters will have the same affect. We can see that increasing the arrival rate Λ_1 and the transition probability r_{12} results in reaching deadlock faster. This is intuitive as increasing these parameters results in the first node's queue filling up quicker. Increasing the service rate μ_1 and the queueing capacity n_1 results in reaching deadlock slower. Again these are intuitive, as increasing the service rate results in moving customers out of the system quicker, and increasing the queueing capacity allows more customers in the system before becoming deadlock.



(a) Varying Λ_1

(b) Varying r_{12}

(c) Varying μ_1

(d) Varying n_1

Figure 20: Analytical & Simulation Results of Times to Deadlock (1000 iterations)

5.9 Resolving Deadlock

Once the system falls into a deadlocked state for the first time, that is the first transient deadlocked state since the last resolution, the simulated needs to automatically resolve the deadlock and allow services to resume again. This is not necassarily as simple as moving a blocked customer to his next node, as we need to conserve the numbers of customers at each service station. Closer inspection of the state digraph is required in order to find a way to resolve deadlock whilst conserving this property.

At deadlock, the service stations can be classified into the following three mutually exclusive categories:

- Nodes that are not deadlocked: These are nodes that do not contain any blocked individuals.
- Causation nodes, nodes causing deadlock: These are nodes where every server is occupied by a blocked individual, and there is at least one blocked individual waiting to enter that node who is directly or

indirectly being blocked by an individual in this node.

- Affected nodes, nodes affected by deadlock: These are nodes containing at least one blocked individual who is directly or indirectly being blocked by an individual at a node that is causing deadlock, but is not classified as causing deadlock itself.

At the first instance of deadlock, there will only be one knot in $D(t)$. Let us denote the knot as K. The vertices of K correspond to servers. As there is no sink node, all vertices of K have an out-edge, and so all vertices in K contain a blocked individual. Therefore, there are no vertices in K belonging to nodes that are not deadlocked. All vertices of K correspond to servers of causation nodes, and a causation node has all its servers belonging to K. An affected node has servers belonging to the same weakly connected component as K, but does ot have servers in K.

When choosing which customer to move in order to resolve deadlock, we must be careful to conserve the number of customers at each service station. Causation nodes have full queues, and a customer may only be moved into a full queue if this causes another customer to simultaneously move from this node. Another complication arises due to the blocking mechanism used, in which those customers who have been blocked longer must be moved first. This property may have to be broken in order to ensure the conservation property is not. Assume that we have weighted the edges of the digraph with the time that they were created.

The following algorithm is proposed in order to resolve deadlock:

```
Find the knot $K$ in $D(t)$
Find the cycle $C \in K$ whose average edge weight is minimum
Start at $V_0$
for $V_i$ in $C$ do
    Move the individual who is waiting to get to $V_{i+1}$
end
Redraw $D(t)$
```

References

[1] S. Albin, J. Barrett, D. Ito, and J.E. Mueller. A queueing network analysis of a health center. *Queueing systems*, 7(1):51–61, 1990.

[2] C. Ancker Jr and A. Gafarian. Some queueing problems with balking and reneging i. *Operations research*, 11(1):88–100, 1963.

[3] C. Ancker Jr and A. Gafarian. Some queueing problems with balking and reneging ii. *Operations research*, 11(6):928–937, 1963.

[4] B. Avi-Itzhak and M. Yadin. A sequence of two servers with no intermediate queue. *Management science*, 11(5):553–564, 1965.

[5] J. Baber. *Queues in series with blocking.* PhD thesis, Cardiff University, 2008.

[6] H. Buhaug. Long waiting lists in hospitals: operational research needs to be used more ofter and may provide answers. *BMJ: British medical journal*, 324(7332):252, 2002.

[7] P. Burke. The output of a queueing system. *Operations research*, 4(6):699–704, 1956.

[8] H. Cho, T. Kumaran, and R. Wysk. Graph-theoretic deadlock detection and resolution for flexible manufacturing systems. *IEEE transactions on robotics and automation*, 11(3):413–421, 1995.

[9] E. Coffman and M. Elphick. System deadlocks. *Computing surveys*, 3(2):67–78, 1971.

[10] S. Creemers and M. Lambrecht. Modelling a healthcare system as a queueing network: the case of a belgian hospital. *Available at SSRN 1093618*, 2007.

[11] F. Gorunescu, S.I. McClean, and P.H. Millard. A queueing model for bed-occupancy management and planning of hospitals. *The journal of the operational research society*, 53(1):19–24, 2002.

[12] P. Harper. A framework for operational modelling of hospital resources. *Health care management science*, 5(3):165–173, 2002.

[13] G. Hunt. Sequential arrays of waiting lines. *Operations research*, 4(6):674–683, 1956.

[14] J. Jackson. Networks of waiting lines. *Operations research*, 5(4):518–521, 1957.

[15] F. Kelly. Networks of queues with customers of different types. *Journal of applied probability*, 12(3):542–554, 1975.

[16] N. Koizumi, E. Kuno, and T.E. Smith. Modeling patient flows using a queueing network with blocking. *Health care management science*, 8(1):49–60, 2005.

[17] R. Korporaal, A. Ridder, P. Kloprogge, and R. Dekker. An analytic model for capacity planning of prisons in the netherlands. *The journal of the operational research society*, 51(11):1228–1237, 2000.

[18] S. Kundu and I. Akyildiz. Deadlock buffer allocation in closed queueing networks. *Queueing systems*, 4(1):47–56, 1989.

[19] G. Latouche and M. Neuts. Efficient algorithmic solutions to exponential tandem queues with blocking. *SIAM journal on algebraic discrete methods*, 1(1):93–106, 1980.

[20] J. Liebeherr and I. Akyildiz. Deadlock properties of queueing networks with finite capacities and multiple routing chains. *Queueing systems*, 20(3-4):409–431, 1995.

[21] K. Rege. Multi-class queueing models for performance analysis of computer systems. *Sadhana*, 15(4-5):355–363, 1990.

[22] E. Reich. Waiting times when queues are in tandem. *The annuls of mathematical statistics*, 28(3):768–773, 1957.

[23] W. Stewart. *Probability, markov chains, queues, and simulation.* Princeton university press, 2009.

[24] R. Sutton and A. Barto. *Reinforecement learning: an introduction.* MIT press, 1998.

[25] C. Szepesvri. *Algorithms for reinforcement learning.* Morgan & Claypool Publishers, 2010.

[26] Y. Takahashi, H. Miyahara, and T. Hasegawa. An approximation method for open restricted queueing networks. *Operations research*, 28(3):594–602, 1980.

[27] A. Turing. Computing machinery and intelligence. *Mind*, 59(236):243–260, 1950.

[28] P.T. Vanberkel, R.J. Boucherie, E.W. Hans, J.L. Hurink, and N. Litvak. A survey of health care models that encompass multiple departments. 2009.

[29] J. Viana, S. Rossiter, A. Channon, S. Brailsford, and A Lotery. A multi-paradigm, whole system view of health and social care for age-related macular degeneration. In *Proceedings of the Winter Simulation Conference*, 2012.