Modelling Deadlock in Open Restricted Queueing Networks

Geraint I. Palmer, Paul R. Harper*, and Vincent A. Knight

School of Mathematics, Cardiff University, Senghennydd Road, Cardiff, CF24 4AG

August 23, 2017

Abstract

Open restricted queueing networks give rise to the phenomenon of deadlock, whereby some customers may be unable to ever leave a server due to mutual blocking. This paper explores deadlock in queueing networks with limited queueing capacity, presents a method of detecting deadlock in discrete event simulations, and builds Markov chain models of these deadlocking networks. The three networks for which Markov models are given include single and multi-server networks for one and two node systems. The expected time to deadlock of these models are compared to results obtained using a simulation of the stochastic process, together with the developed deadlock detection method. This paper aims to be of value to simulation modelling of queues.

Keywords: queueing, queueing networks, deadlock, Markov models

1 Introduction

The study and modelling of queueing networks with blocking is an important tool in many aspects of operational research, both analytically and through simulation. These models have applications in many varied settings such as healthcare, supply chains, manufacturing and communications systems. However, these types of models have their limitations, due to their potential to become permanently blocked in deadlock, or a deadly embrace of resources. These deadlocks can be real and observed in reality, in which case accurate modelling of deadlock is needed; or they can be a symptom of a model unable to capture certain behaviours. This may occur in models where deadlock situations are easily adjusted in reality. In the latter case, such as by swapping two customers, a good understanding of deadlock is needed in order to model the adjusted reality.

Queueing networks are described as open if customers can enter and leave the system from the exterior. Restricted networks are those where at least one service centre has limited queueing space or capacity before it. =Deadlock is caused by blocking. This paper considers Type I blocking: after service a customer will be blocked from joining a queue at another node if that node's queueing capacity is full. While blocked, that customer remains with its server until space becomes available at its destination. During this time that server is unavailable to begin another customer's service.

*Corresponding author Prof. Paul Harper, harper@cardiff.ac.uk

For the purposes of this paper, deadlock is defined as follows.

Definition 1. *When there is a subset of blocked customers who are blocked directly or indirectly by customers in that subset only, then the system is said to be in deadlock.*

This implies that a system is in deadlock when at least one service station permanently ceases to begin of finish any more services, due to circular blocking. Figure 1 shows an open two node restricted queueing network in deadlock. The customer at the top server is blocked from entering the bottom node as there is a full queue, and similarly the customer at the bottom server is blocked from entering the top node as there is a full queue. It is clear that by following the rules of a blocking defined above, no more natural movement can happen. This system is in deadlock as there is a subset of blocked customers (the customer with server A_1 and the customer with server B_1) who are blocking eachother only.

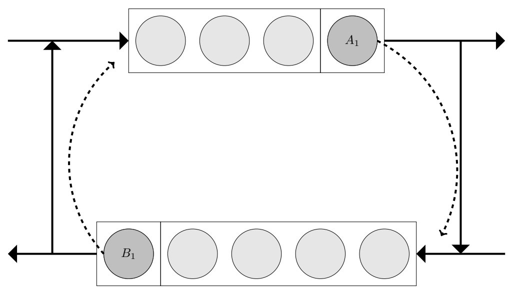


Figure 1: Example of an open two node restricted queueing network in deadlock.

This paper is concerned with open restricted queueing networks that experience Type I blocking. Throughout the paper service centres will be referred to as nodes, and for the ith node of an open restricted queueing network the following notation is used:

- Λ_i denotes the external arrival rate.
- μ_i denotes the service rate.
- c_i denotes the number of parallel servers.
- n_i denotes the queueing capacity.
- r_{ij} denotes the routing probability from node i to node j upon completion of service at node i.

Exponential service times and Poisson arrivals are assumed. First in first out, or FIFO service discipline is also assumed.

In particular this paper looks at detecting when deadlock occurs, and the time until a deadlock occurs from an empty system. First, a method for detecting deadlock in simulations of queueing networks is presented. Then, Markov models of simple deadlocking queueing networks are built.

The remainder of this paper is structured as follows: Section 2 gives a motivating example to put the work in context. Section 3 gives an overview of existing literature on the subject. Section 4 presents a method of detecting deadlock in simulations of queueing networks. Section 5 presents Markov models of three

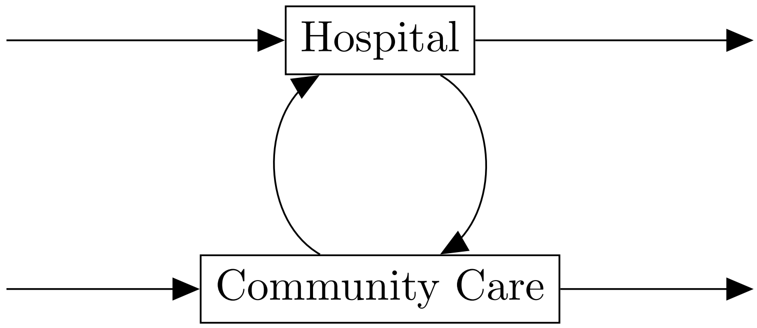


Figure 2: Diagram of patient flows at an interface between secondary care services at a hospital and community care services.

deadlocking queueing networks, derives their expected time to deadlock, and compares these with results obtained through the simulation model.

2 A Motivating Example

Here we present a motivating example of a healthcare system. In this example deadlock may be easily resolved in reality, however analytical stochastic models and simulations may be restricted by deadlock. Therefore and understanding of this phenomenon, and an ability to overcome this effect is discrete event simulations, is essential for modelling this system.

Consider the interface between secondary care services at a hospital and community care services. Patients can be admitted to hospital via a variety of routes (through emergency services, outpatients), and via referral from community care services. Patients can begin receiving community care packages due to referral from GP, or via referral from the hospital. Considering only the hospital and community care services as nodes, this system is shown in Figure 2.

If there are no free hospital beds, then patients being referred from community care services will be sustained by community care workers until beds become available. If there are no community care packages available, then patients requiring packages but unfit to return home after a hospital stay will remain in hospital, blocking beds until a community care package becomes available. Type I blocking occurs here, as patients and staff do not know the future capacity of their next destination prior to service. This type of bed blocking is well known [26]. This causes problems for patients as they are being cared for in an inappropriate setting for their condition. There are also for the health care providers as secondary care may be more expensive than primary care, and resolution of this causes administrative stress.

In this model there is a non-zero probability of everyone at the hospital blocking beds waiting for community care packages, and everyone at community care being sustained waiting for beds at the hospital. Thus the model will exhibit deadlock. In reality, there is communication between these services and patients can swap places. This ensures no deadlock.

Restricted feedback loops that exhibit mutual blocking such as this one have been observed in real healthcare systems, as described in a case study in [30]. However the authors here state that this type of blocking "may be irrelevant in practice given that the swapping of patients can be identified and carried out easily". In [21] a health and community care system is described as having restricted feedback loops. However due to ease of modelling and to avoid the restrictions caused by deadlock, these feedback loops are omitted from the model. This emphasises the discrepancies that occur between common modelling techniques and reality in systems that may reach deadlock.

An understanding of how deadlock behaves in these models will aid in the modelling process. A deadlock detection method for the simulation model will be invaluable in modelling realistic deadlock resolution methods, thus ensuring correct models can be built of systems like this with circular blocking.

3 Literature Review

Restricted queueing networks that exhibit blocking are well discussed in the literature, both exact [3,4,16,19, 21,24,31] and approximate methods [2,9,22,28,30,31,36]. Discussions on restricted queueing networks with feedback loops, that may exhibit deadlock, are sparse however. In fact the problem of deadlock in queueing networks has either been ignored, not studied, or assumed resolved in much of the literature [28,30,31].

Central to the study of deadlock in queueing networks is the concept of blocking. In [29] three types of blocking are described. Type I blocking (blocked at service, BAS, transfer blocking) occurs when a customer is blocked after completing service, and remains with the server until capacity at their destination node becomes available. Type II blocking (blocked before service, BBS, service blocking) occurs when a customer declares their destination before beginning service, and is only granted service if there is available capacity at their destination node. In Type III blocking (repetitive services, RS-FD and RS-RD, rejection blocking) instead of getting blocked a customer is required to repeat their service if there is no capacity at their destination. This type of blocking comes in two forms, fixed destination where the customer's destination does not change at each repetition of service, and random destination, where the customer's destination is re-sampled from a probability distribution after each repetition.

There has been a body of research into detection and prevention of deadlock which doesn't consider the underlying stochastic structure of the system [8,32,33]. These general deadlocks occur in flexible manufacturing systems and distributed communication systems. This type of deadlock, also referred to as deadly embraces [8], can potentially occur under the following conditions:

- Mutual exclusion: Tasks have exclusive control over resources.
- Wait for: Tasks do not release resources while waiting for other resources.
- No pre-emption: 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.

In open restricted queueing networks the mutual exclusion condition is satisfied as customers cannot share servers; the wait for condition is satisfied due to the blocking rules defined previously; the no pre-emption condition is satisfied in networks that have no or non-pre-emptive priority (this paper only considers networks with no priority); and the circular wait condition is satisfied if the queueing network contains a cycle where all nodes have limited queueing capacity, that is, a feedback loop.

Allowing a system to reach deadlock can be problematic in cases where automated systems cannot continue operations, or where simulations cannot accurately model reality. In general there are three strategies for dealing with the problem of deadlock [12, 20, 41, 43]:

- Avoidance, in which decisions are made as time unfolds to avoid reaching deadlock.
- Prevention, in which the system is designed such that in cannot possibly deadlock in the first place.
- Detection and recovery.

Note that [17] lists the three strategies as prevention, detection and crashing, which is equivalent to having no deadlock strategy. Allowing the system to crash now and again may be more economical in some systems where deadlocks do not occur often enough to justify the investment and effort of implementing an avoidance/resolution strategy.

The first two of these strategies, prevention and avoidance, have been used extensively in an area known as Discrete Event Systems (DES) [32,33]. A distinction between *online* and *offline* resource allocation strategies is made [41]. Deadlock avoidance techniques are online strategies, decisions are made as the system runs, whereas deadlock prevention techniques are offline strategies as decisions are made in the planning and designing of the system.

Various techniques and methods have been used to implement deadlock avoidance such as the Banker's algorithm [11, 20], petri net models [13, 27, 44], and resource allocation graphs [5]. A survey of avoidance techniques for automated guided vehicle systems is given in [43]. These techniques generally determine when resources cannot be allocated as that allocation would lead to deadlock. In [14] a priority based deadlock avoidance algorithm is implemented in a traffic simulation model. The purpose of the avoidance scheme here is not to reflect deadlock avoidance in reality, but to avoid deadlocks that will occur in the simulation due to missing information or incomplete models.

The literature has discussed deadlock prevention in closed queueing networks when blocking of Type I occurs. For closed networks of K customers with only one class of customer, [23] proves the following condition to ensures no deadlock: given that each node j has total capacity B_j, 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 sub-cycle 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 [25], 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.

Further conditions on deadlock prevention in closed queueing networks are reviewed in [28], including closed networks under different blocking mechanisms such Type II and Type III blocking. For simulation modelling however, prevention and avoidance techniques may not be appropriate as they can potentially inhibit realism in the simulation by taking actions that do not occur in the system being modelled [42].

In [34] a deadlock prevention/avoidance mechanism for open restricted queueing networks is given. Here, arrivals from outside the system are turned away if certain nodes are full. This ensures that the whole system cannot become saturated, and reach deadlock.

A popular method of detecting general deadlock is the use of wait-for graphs, state-graphs and other variants [6, 7, 8, 10, 12, 17, 40, 41, 42]. These wait-for graphs, keep track of all circular wait relations between tasks. In [8] 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 cycle. In [7] 'simple bounded circuits' are defined by giving the vertices and edges of the state graph labels in relation to a reference node. The existence of these circuits within the state graph indicates if the system is in deadlock. A strategy of this type is developed in this paper to detect deadlock in queueing systems.

Bipartite graphs are used in [17] to detect deadlock in systems with both consumable and reusable resources, where the concept of reducibility and knots are used to detect deadlock in certain situations. These graphs are developed and used in [10] and [40] where these dynamic bipartite entity-resource graphs (E/R graphs) are used to detect two different types of deadlock, transient deadlock and permanent deadlock. A deadlock resolution procedure is proposed that attempts to break cycles in the E/R graph These E/R graphs contain resources and entities as vertices, and request or holding relationships as edges; strongly connected components and closed strongly connected components indicate a deadlock situation. This work is furthered in [42] where deadlock is detected and resolved for situations where entities may request more than one resource.

A number of deadlock types are defined in [42], and their relationships to one another. This is shown in Figure 3. 'Transient deadlock' defined there corresponds to the blocking considered in this manuscript, and isn't considered a deadlock situation here; whereas 'permanent deadlock' and 'absolute deadlock' defined there is equivalent to the type of deadlock that is discussed in this paper. The authors of [42] state that in overlapping deadlock situations (the kind that arise in the queueing networks discussed in this paper) there is no conceptual difference between transient and permanent deadlocks. However using the definitions given in this paper transient deadlock is not considered as deadlock at all. We consider these two deadlock situations as fundamentally different.

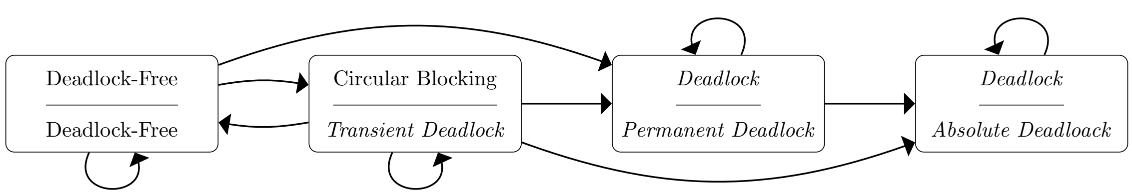


Figure 3: Illustrating the types of deadlock, and the differences between the meaning of deadlock in this paper (top writing) and in some of the literature (bottom writing). Adapted from [42]. States considered as deadlock are italicised.

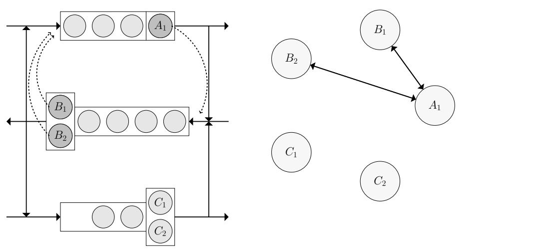
Deadlock detection and recovery in closed queueing networks through swapping customers is assumed in [31], with zero transition time assumed between deadlocked states and the corresponding resolved state. Time to resolve deadlock may not be negligible in reality. Deadlock detection and recovery is listed as one of the two possible solutions for handling deadlock in queueing networks in [1], although there is no further discussion.

4 Deadlock Detection

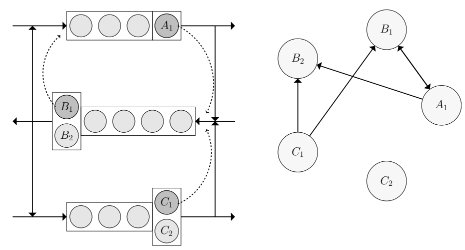
In order to detect when deadlock has occurred in a queueing network simulation, a state digraph is used, a form of wait-for graph. In previous literature on wait-for graphs these are bespoke graphs that represent system states, where edges denote some form of waiting or blockage relationships. Here we present a generic state digraph that is defined for all FIFO queueing networks that exhibit Type I blocking:

Definition 2. *The state digraph $D(t)$ of a queueing network is defined by that network's state at any time t. Vertices of the state digraph correspond to servers of the network. A directed edge denotes a blockage relationship in the following manner: if a customer at the kth server of node i is blocked from entering node j, then there are directed edges from the vertex corresponding to node i's kth server to every vertex corresponding to the servers of node j.*

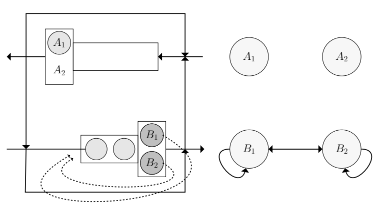
To illustrate this concept Figure 4 shows examples of queueing networks in and out of deadlock, and the corresponding state digraph in each case.



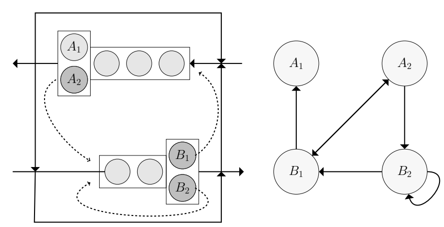
(a) A three node queueing network in deadlock, with state digraph.



(b) A three node queueing network not in deadlock, with state digraph.



(c) A two node queueing network in deadlock, with state digraph.



(d) A two node queueing network not in deadlock, with state digraph.

Figure 4: Examples of state digraphs with their corresponding queueing networks.

These graph theoretic terms will be used throughout this paper [15, 45]:

- Two vertices v_1 and v_2 are said to be *strongly connected* if there is a directed path from v_1 to v_2 and a directed path from v_2 to v_1.
- Two vertices v_1 and v_2 are said to be *weakly connected* if there is either a directed path from v_1 to v_2 or a directed path from v_2 to v_1.
- A *strongly connected component* of a digraph is a subgraph induced by a maximal subset of strongly

connected nodes.

- A *weakly connected component* of a digraph is a subgraph induced by a maximal subset of weakly connected nodes.
- v_1 is an *ancestor* of v_2 if there is a path from v_1 to v_2.
- v_2 is a *descendant* of v_1 if v_1 is an ancestor of v_2.
- The *in-degree* of v_1, denoted $\deg^{\text{in}}(v_1)$ is the number of in-edges incident to v_1. Similarly the *out-degree* of v_1, denoted $\deg^{\text{out}}(v_1)$ is the number of out-edges incident to v_1.
- A *source* is a vertex whose in-degree is zero. A *sink* is a vertex whose out-degree is zero.

Consider one weakly connected component $G(t)$ of $D(t)$ and a node $X \in G(t)$. Some observations:

- If the server corresponding to X is unoccupied, then X has no incident edges.
- Consider the case when the server corresponding to X is occupied by individual a, whose next destination is node j. Then X's direct successors correspond to servers occupied by individuals who are blocked or in service at node j.
- It can interpreted that all X's descendants correspond to servers whose occupants are directly or indirectly blocking a, and interpret all X's ancestors as those servers whose individuals are being blocked directly or indirectly by a.
- All vertices of $G(t)$ are either descendants of another vertex, and so correspond to servers occupied by an individual who is blocking someone; or are ancestors of another vertex, and so are occupied by someone who is blocked.
- Note that the only possibilities for $\deg^{\text{out}}(X)$ are 0 or c_j. If $\deg^{\text{out}}(X) = c_j$ then a is blocked by all its direct successors. The only other situation is that a is not blocked, and $X \in G(t)$ because a is in service at X and blocking other individuals, in which case $\deg^{\text{out}}(X) = 0$.
- It is clear that if all of X's descendants correspond to servers occupied by blocked individuals, then the system is deadlocked at time t.
- By definition all of X's ancestors correspond to servers occupied by blocked individuals.

Recall that a knot is defined as a strongly connected component where no vertex has a path to any vertices not in that strongly connected component. Therefore there can be no path from a vertex in the knot to a vertex without any out-edges. The following results detect deadlock for open restricted queueing networks.

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

Proof. Proof Consider one weakly connected component $G(t)$ of $D(t)$ at time t.

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

Now assume that we have deadlock. For a vertex X which corresponds to a server involved in the deadlock, all descendants 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.

□

The knot condition can be simplified for specific cases.

Proposition 1. *For queueing networks:*

1. *with one node*
2. *with two nodes, each with two or fewer parallel servers*
3. *with a finite amount of nodes, each with a single-server*

a deadlocked state arises if and only if there exists a weakly connected component without a sink node.

Proof. Proof There are three parts to this proof:

1. Consider a one node queueing network.

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

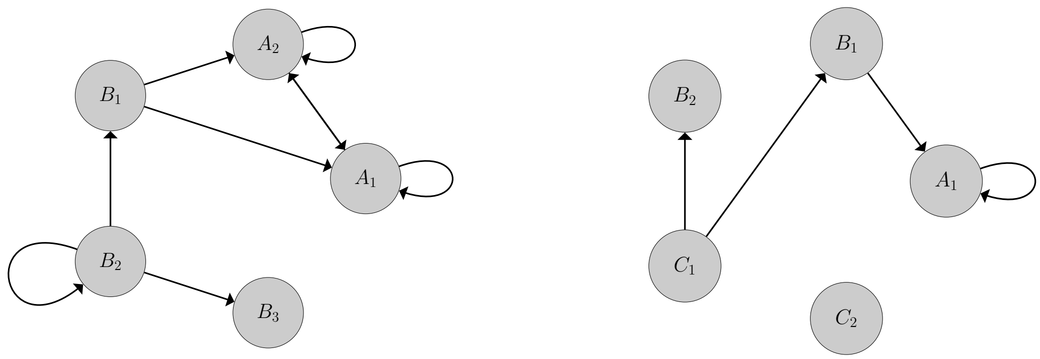
2. Consider a two node queueing network, each node with 2 or fewer parallel servers.

   If both nodes are involved in the deadlock, so there is at least one customer in node 1 blocked from entering node 2, and at least one 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, consider 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 connected component, there either needs to be an edge from a server in node 1 to a server in node 2, or an 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 one server in node 2 has an out-edge. Now either the other server of node two is empty or still in service, and so isn't part of that weakly connected component, or the other server's customer is blocked and so has an out edge.

   For the case of a two node queueing network with at least one node with more than 2 servers, consider the following counter-example:

   Node A has two parallel servers, node B has three parallel servers. Begin with all servers occupied by customers in service and full queues. The customer at server A_1 is blocked to node A. The customer at server B_1 is blocked to node A. The customer at server B_2 is blocked to node B. The customer at



(a) State Digraph of Counter-Example 1.

(b) State Digraph of Counter-Example 2.

Figure 5: Counter examples.

server A_2 is blocked to node A. Node A is a deadlocked. The resulting state digraph in Figure 5a has a weakly connected component with a sink.

3. Consider a queueing network with N nodes, each with a single-server.

   If $1 \leq n \leq N$ nodes are involved in the deadlock, then each server in those n nodes has a blocked customer, and so has an out-edge. Of the other nodes, they can only be in the same weakly connected component if either they contain a blocked individual, in which case they will have an out edge; or they are blocking someone who is blocked directly or indirectly by those in deadlock. However this last case cannot happen, as every node is single-server each person can only be blocked by one other individual at a time.

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

   Node A has one parallel server, node B has two parallel servers, and node C has two parallel servers. Begin with all servers occupied by customers in service and full queues. 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. Node A is deadlocked. The resulting state digraph in Figure 5b has a weakly connected component with a sink.

□

For the purposes of this paper, a simulation model is used to verify that the results of this section and the analytical model in Section 5 are in agreement. Specifically the time taken to reach deadlock from an empty system is investigated, and the simulation model gives information on the distribution of times to deadlock. The model is built using Ciw [38]. This is an object oriented framework in Python [39], with care taken to ensure reproducibility of the results [18].

The digraph $D(t)$ is implemented as an attribute of the queueing network and is updated at the appropriate events. Note that a brute force algorithm is used to check whether any strongly connected component of $D(t)$ is a knot in order to implement Theorem 1. More efficient algorithms could be used for other specific use cases.

The next section builds Markov models of three queueing networks, and discusses their deadlocking properties.

5 Markovian Models of Deadlocking Queueing Networks

The following three networks describe all possible configurations of deadlocking queueing networks with two or fewer nodes:

1. Open one node, multi-server restricted queueing network with feedback loop. (Section 5.1)
2. Open two node, multi-server restricted queueing network with routes between nodes. (Section 5.2)
3. Open two node, multi-server restricted queueing network with routes between nodes and self-loops.

In this section Markov models are built for networks 1 and 2, and expressions for their expected time to deadlock found. The state space for network 3 is too large to model in a similar way to the others, and so isn't considered in this paper. A single server version is modelled however, and the multi-server system will be discussed in Section 6.

In general a continuous Markov chain model of a deadlocking queueing network is defined by a set of states $s \in S$ and the transition rates between these states q_{s_1,s_2}. Each state s uniquely defines a configuration of customers around the queueing network. Deadlocked states are also present, either denoted by that specific configuration of customers, or by negative numbers, for example (-1). Deadlocked states cannot transition to any other state, and so are absorbing states of the Markov chain. Therefore any queueing network that can experience deadlock is guaranteed to experience deadlock, as absorbing Markov chains are guaranteed to enter one of its absorbing states.

The expected time until deadlock is reached is equivalent to the expected time to absorption of the Markov chain, which can easily be found [35]. The canonical form of an absorbing Markov chain is

$$P = \begin{pmatrix} T & U \\ 0 & I \end{pmatrix}$$

where I is the identity matrix.

Now the expected number of time steps until absorption starting from state i is the ith element of the vector

$$(I - T)^{-1} e \tag{5.1}$$

where e is a vector of 1s.

Therefore by discretising the continuous Markov chain and ensuring the correct order of states, the expected number of time steps to absorption, which corresponds to deadlock, can be found.

When there is more than one deadlocked state, there is more than one absorbing state in the Markov chain. Here the expected time to absorption is the expected time to a deadlocked state, whichever one that may be. The probabilities of which absorbing state a Markov chain will reach are given by the $(i, j)^{\text{th}}$ element of the following matrix:

$$(I - T)^{-1}U \tag{5.2}$$

which corresponds to the probability of reaching absorbing state j from transient state i.

5.1 One Node Multi-Server

Consider the open one node multi-server restricted queueing network shown in Figure 6. This shows an $M/M/c/n$ queue where customers arrive at a rate of Λ and served at a rate μ. Once a customer has finished service they rejoin the queue with probability r_{11}, and so exit the system with probability $1 - r_{11}$. Let this system be denoted by Ω_1 with parameter set (Λ, μ, c, n, r_{11}), and the time to deadlock of this system be denoted by ω_1.

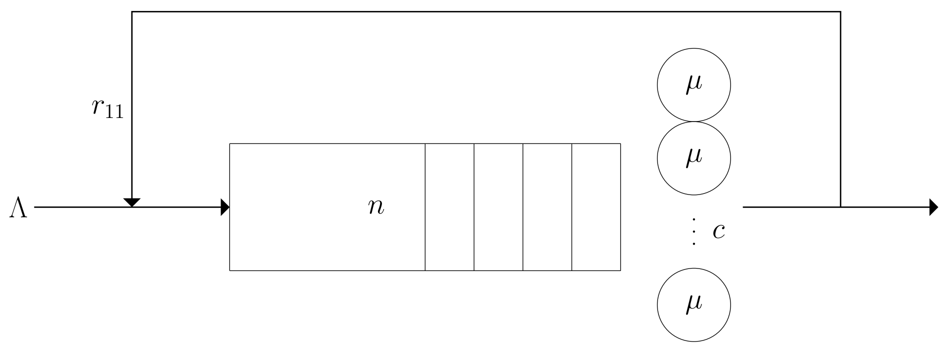


Figure 6: An open one node multi-server restricted queueing network.

The state space is given by:

$$S = \{i \in \mathbb{N} \mid 0 \leq i \leq n + 2c\}$$

where i denotes the number of individuals in the system plus the number of individuals who are blocked. For example, $i = n + c + 2$ denotes a full system, $n + c$ individuals in the node, and 2 of those individuals are also blocked. The state $i = n + 2c$ denotes the deadlocked state, that is every customer with a server is blocked.

Define $\delta = i_2 - i_1$ for all $i_k \in S$. The transitions are given by Equations 5.3 and 5.4.

$$q_{i_1,i_2} = \left\{ \begin{array}{rr} \Lambda & \text{if } \delta = 1 \\ (1 - r_{11})\mu\min(i, c) & \text{if } \delta = -1 \\ 0 & \text{otherwise} \end{array} \right\} \quad \text{if } i_1 < n + c \tag{5.3}$$

$$q_{i_1,i_2} = \left\{ \begin{array}{rr} (c-b)r_{11}\mu & \text{if } \delta = 1 \\ (1-r_{11})(b-k)\mu & \text{if } \delta = -b-1 \\ 0 & \text{otherwise} \end{array} \right\} \quad \text{if } i_1 = n + c + b \quad \forall \quad 0 \leq b \leq c \tag{5.4}$$

where b denotes the number of blocked customers. The Markov chain is shown in Figure 7.

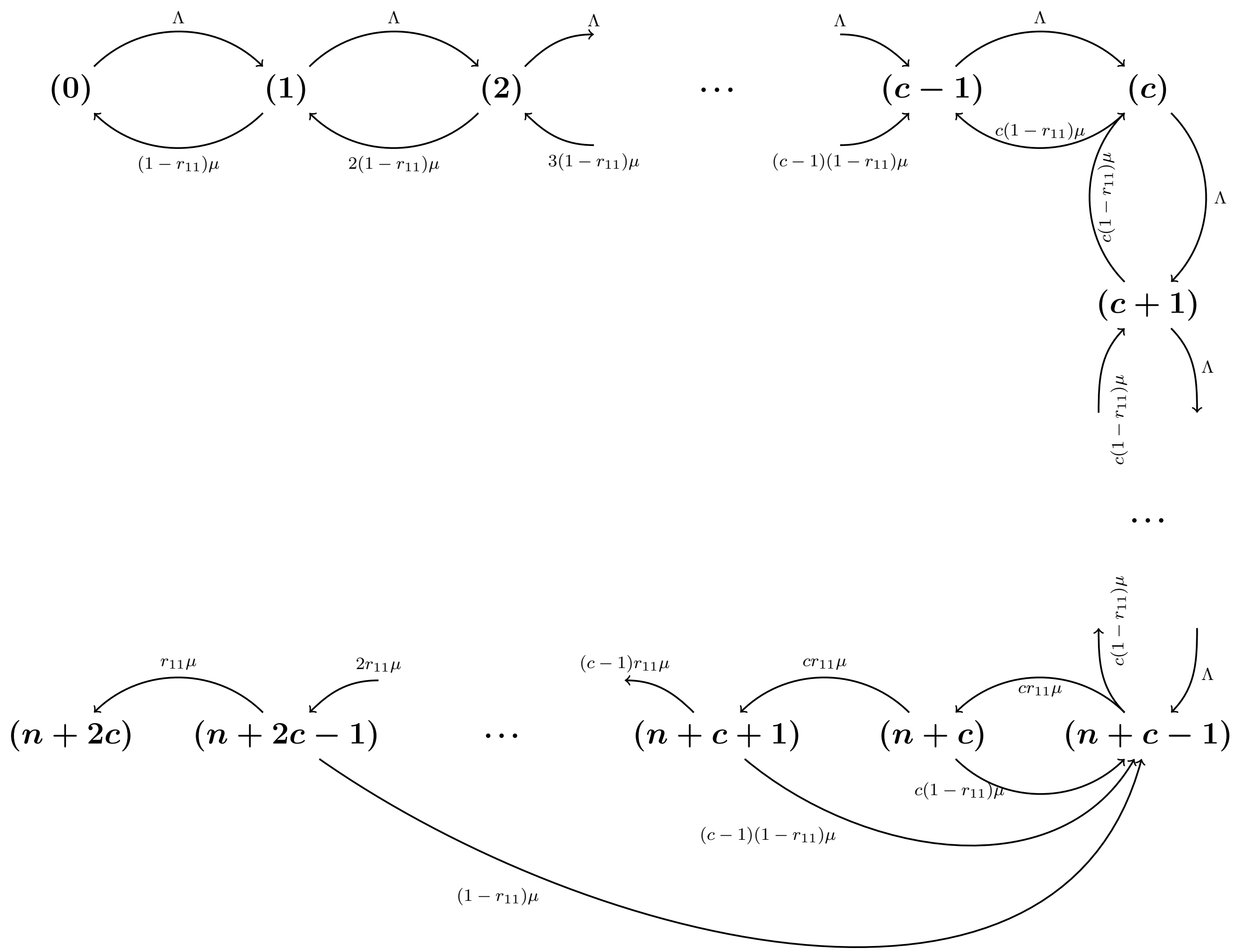
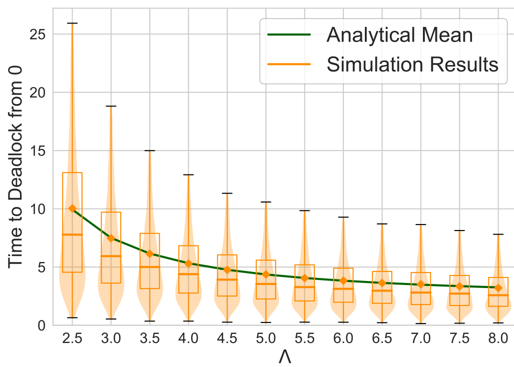


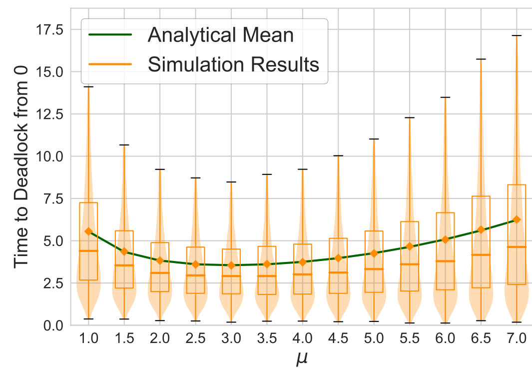
Figure 7: Diagrammatic representation of the Markov chain for a multi-server Ω_1 system.

Figure 8 shows the effect of varying the parameters of the above Markov model. Base parameters of $\Lambda = 6$, $n = 3$, $\mu = 2$, $r_{11} = 0.5$ and $c = 2$ were used.

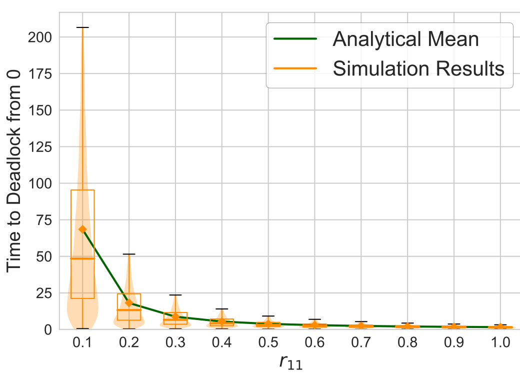
It can be seen that increasing the arrival rate Λ and the routing probability r_{11} results in reaching deadlock faster. This is intuitive as increasing these parameters results in the queue filling up quicker. Increasing the queueing capacity n results in reaching deadlock later. Again this is intuitive, as increasing the queueing capacity allows more customers in the system before becoming deadlocked.



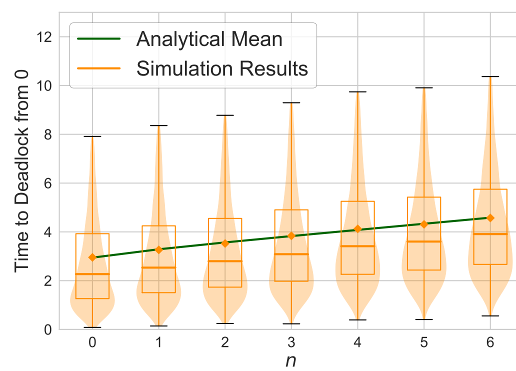
(a) Varying Λ



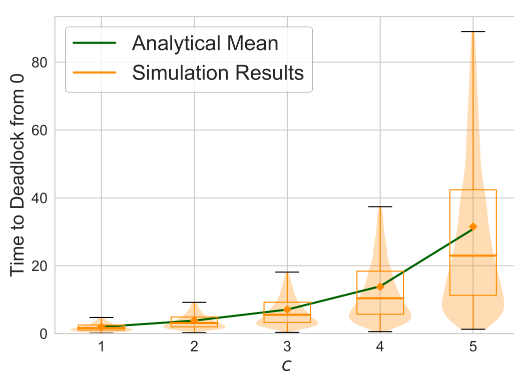
(b) Varying μ



(c) Varying r_{11}



(d) Varying n



(e) Varying c

Figure 8: Time to deadlock in multi-server Ω_1, analytical & simulation results (10,000 repetitions).

Increasing the amount of servers has a similar effect to increasing the queueing capacity, there are now more transient states to go through before reaching the deadlocked state. Varying the amount of servers has a greater effect on the time to deadlock however, as any states in which customers are blocked, $i \in [n+c+1, n+2c]$, can jump back to state $i = n+c-1$ simply with a service where the customer doesn't rejoin the queue. Increasing the amount of servers also increases the rate at which customer leave the system, but not the rate at which customers enter the system. This means that the rate of increase of the number of customers in the system increases, however the rate of decrease of number of customers in the system does not change, thus it would take longer to reach a full system and deadlock.

The behaviour as the service rate μ varies is not monotonic, as the service rate contributes towards both moving customers from the system and allowing customers to rejoin the queue, causing blockages and deadlock. This behaviour is described in the following remark.

Remark 1. *The function $\omega_1(\mu)$ that describes the expected time to deadlock of an Ω_1 system as the service rate μ varies, and all other parameters are fixed, has one critical point and is a local minimum for $\mu \in (0, \infty)$.*

The behaviour of $\omega_1(\mu)$ can be interpreted as follows:

- At $\lim_{\mu \to 0} \omega_1(\mu)$ there is infinite service time, and so infinite time until deadlock.
- At $\lim_{\mu \to \infty} \omega_1(\mu)$ there is zero service time, the queue can never fill up, and so infinite time to deadlock.
- At low service rates below a certain threshold $\hat{\mu}$, the arrival rate is relatively large compared to the service rate, and we can assume a saturated system. At this point services where a customer exits the system does not have much of an effect, as we can assume another arrival immediately. However services where a customer wishes to rejoin the queue results in a blockage as the system is saturated. Therefore, increasing the service rate here increases the chance of a blockage, and so the chance of deadlock.
- Above $\hat{\mu}$ the service rate is large enough that we cannot assume a saturated system, and so services where the customer exits the system does have an affect on the number of customers in the system. Thus increasing the service rate removes people from the system, and as such there is less chance of getting blocked and reaching deadlock.

The following remark identifies the maximum of $\omega_1(\mu)$ on a closed interval.

Remark 2. $\arg\max_{\mu \in [a,b]} \omega_1(\mu) \in \{a, b\}$.

From the closed interval method of finding absolute maximum [37], the absolute maximum of $\omega_1(\mu)$ on the closed interval $[a, b]$ is either the critical points in (a, b), a or b. The only possible critical point in (a, b) is $\hat{\mu}$, and is a local minimum (from Remark 1), and so $\omega_1(\hat{\mu}) \leq \omega_1(a)$ and $\omega_1(\hat{\mu}) \leq \omega_1(b)$. Therefore $\omega_1(\mu)$ obtains its maximum at either a or b.

5.2 Two Node Multi-Server without Self-Loops

Consider the open two node multi-server restricted queueing network shown in Figure 9. This shows two $M/M/c_i/n_i$ queues, with service rates μ_i and external arrival rates Λ_i. All routing probabilities r_{ij} may be positive apart from self-loops r_{ii}, for each node i. Note that this system is equivalent to the one described in Section 2.

Let this system be denoted by Ω_2 with parameter set (Λ_1, Λ_2, μ_1, μ_2, c_1, c_2, n_1, n_2, r_{12}, r_{21}), and the time to deadlock of this system be denoted by ω_2.

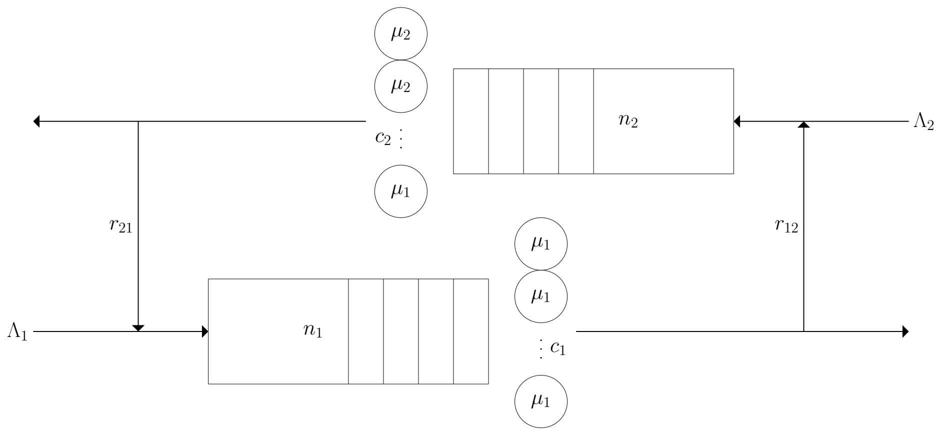


Figure 9: An open two node multi-server restricted queueing network.

The state space is given by:

$$S = \{(i, j) \in \mathbb{N}^{(n_1+c_1+c_2)\times(n_2+c_2+c_1)} \mid i \leq n_1 + c_1 + j,\ j \leq n_2 + c_2 + i\}$$

where i denotes the number of individuals at Node 1 plus the number of individuals blocked waiting to enter Node 1, and j denotes the number of individuals at Node 2 plus the number of individuals blocked waiting to enter Node 2. For example, $(i, j) = (n_1 + c_1 + 2, n_2 + c_2 + 1)$ denotes a full system, $n_1 + c_1$ individuals at Node 1, two of whom are blocked waiting to enter Node 2; $n_2 + c_2$ individuals at Node 2, one of whom is blocked waiting to enter Node 1. The state $(i, j) = (n_1 + c_1 + c_2, n_2 + c_2 + c_1)$ denotes the deadlocked state.

The Markov chain is shown in Figure 10.

Define $\delta = (i_2, j_2) - (i_1, j_1)$, $b_1 = \max(0, i_1 - n_1 - c_1)$, $b_2 = \max(0, i_2 - n_2 - c_2)$, $s_1 = \min(i_1, c_1) - b_2$ and $s_2 = \min(i_2, c_2) - b_1$ for all $(i_k, j_k) \in S$. Then the transitions $q_{(i_1,j_1),(i_2,j_2)}$ are given by Table 1.

The values b_1 and b_2 correspond to the number of people blocked to Node 1 and Node 2 respectively. The values s_1 and s_2 correspond to the amount of people currently in service at Node 1 and Node 2 respectively.

Figure 11 shows the effect of varying the parameters of the above Markov model. Base parameters of $\Lambda_1 = 9$, $\Lambda_2 = 7.5$, $n_1 = 2$, $n_2 = 1$, $\mu_1 = 5.5$, $\mu_2 = 6.5$, $r_{12} = 0.7$, $r_{21} = 0.6$, $c_1 = 2$ and $c_2 = 2$ were used. Only plots for the parameter corresponding to Node 1 are shown, Node 2 shows similar behaviour.

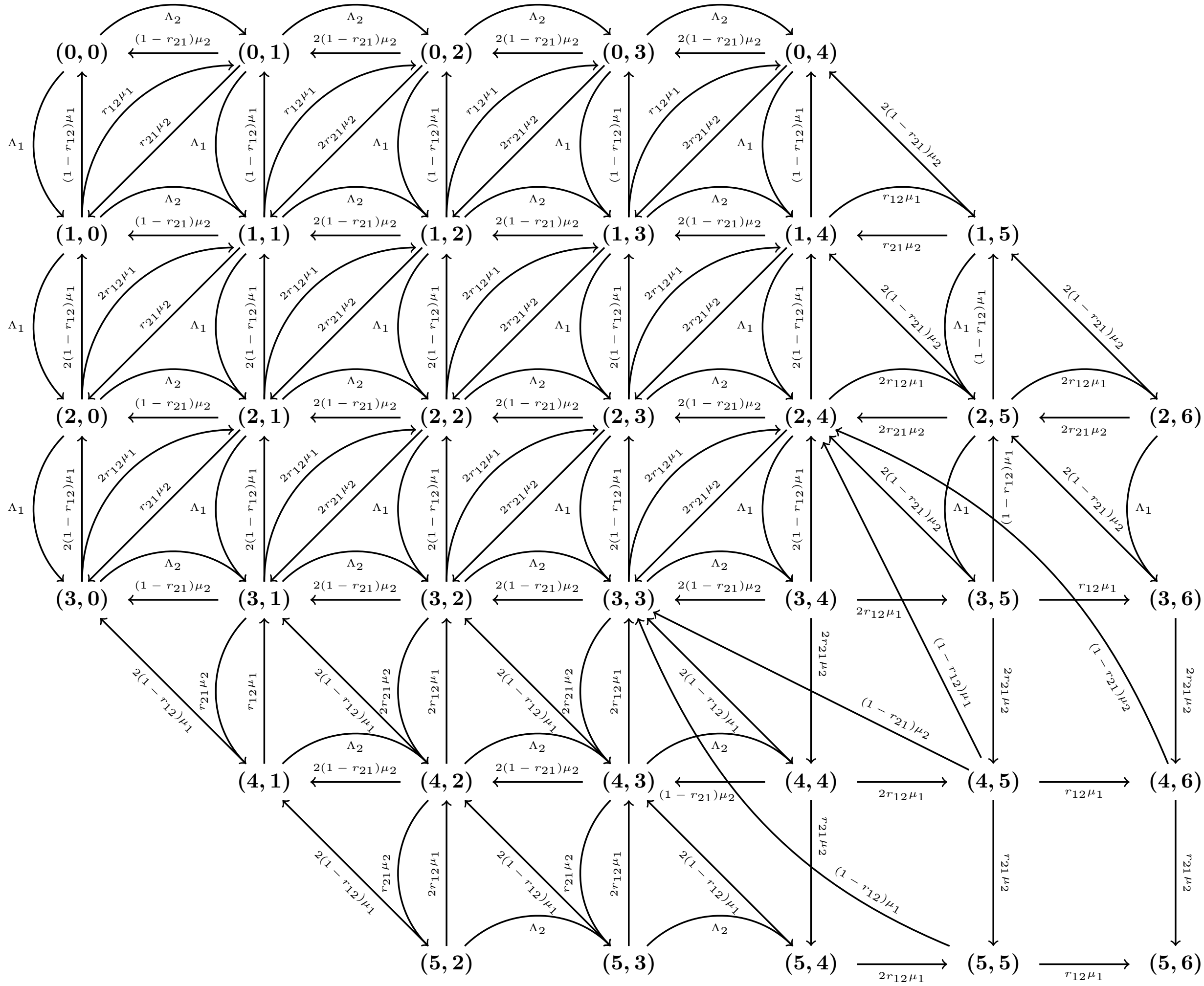
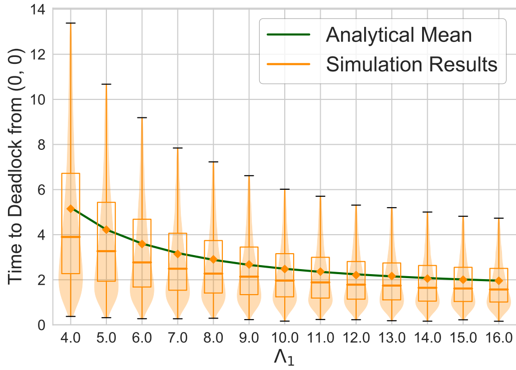


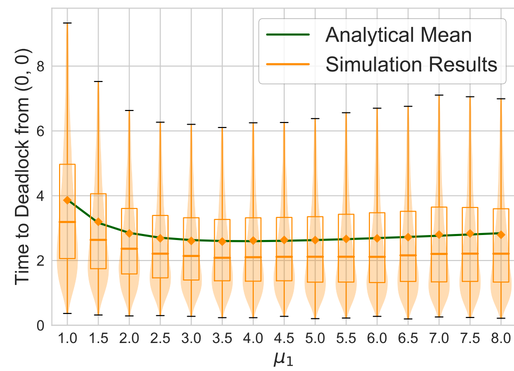
Figure 10: Diagrammatic representation of the Markov chain for a multi-server Ω_2 system with $n_1 = 1$, $n_2 = c_1 = c_2 = 2$. The deadlocked state is $(5, 6)$.

	$j_1 < n_2 + c_2$	$j_1 = n_2 + c_2$	$j_1 > n_2 + c_2$
$i_1 < n_1 + c_1$	Λ_1 if $\delta = (1, 0)$ Λ_2 if $\delta = (0, 1)$ $r_{12}s_1\mu_1$ if $\delta = (-1, 1)$ $r_{21}s_2\mu_2$ if $\delta = (1, -1)$ $(1 - r_{12})s_1\mu_1$ if $\delta = (-1, 0)$ $(1 - r_{21})s_2\mu_2$ if $\delta = (0, -1)$	Λ_1 if $\delta = (1, 0)$ $r_{12}s_1\mu_1$ if $\delta = (0, 1)$ $r_{21}s_2\mu_2$ if $\delta = (1, -1)$ $(1 - r_{12})s_1\mu_1$ if $\delta = (-1, 0)$ $(1 - r_{21})s_2\mu_2$ if $\delta = (0, -1)$	Λ_1 if $\delta = (1, 0)$ $r_{12}s_1\mu_1$ if $\delta = (0, 1)$ $r_{21}s_2\mu_2$ if $\delta = (0, -1)$ $(1 - r_{12})s_1\mu_1$ if $\delta = (-1, 0)$ $(1 - r_{21})s_2\mu_2$ if $\delta = (-1, -1)$
$i_1 = n_1 + c_1$	Λ_2 if $\delta = (0, 1)$ $r_{12}s_1\mu_1$ if $\delta = (-1, 1)$ $r_{21}s_2\mu_2$ if $\delta = (1, 0)$ $(1 - r_{12})s_1\mu_1$ if $\delta = (-1, 0)$ $(1 - r_{21})s_2\mu_2$ if $\delta = (0, -1)$	$r_{12}s_1\mu_1$ if $\delta = (0, 1)$ $r_{21}s_2\mu_2$ if $\delta = (1, 0)$ $(1 - r_{12})s_1\mu_1$ if $\delta = (-1, 0)$ $(1 - r_{21})s_2\mu_2$ if $\delta = (0, -1)$	$r_{12}s_1\mu_1$ if $\delta = (0, 1)$ $r_{21}s_2\mu_2$ if $\delta = (1, 0)$ $(1 - r_{12})s_1\mu_1$ if $\delta = (-1, 0)$ $(1 - r_{21})s_2\mu_2$ if $\delta = (-1, -1)$
$i_1 > n_1 + c_1$	Λ_2 if $\delta = (0, 1)$ $r_{12}s_1\mu_1$ if $\delta = (-1, 0)$ $r_{21}s_2\mu_2$ if $\delta = (1, 0)$ $(1 - r_{12})s_1\mu_1$ if $\delta = (-1, -1)$ $(1 - r_{21})s_2\mu_2$ if $\delta = (0, -1)$	$r_{12}s_1\mu_1$ if $\delta = (0, 1)$ $r_{21}s_2\mu_2$ if $\delta = (1, 0)$ $(1 - r_{12})s_1\mu_1$ if $\delta = (-1, -1)$ $(1 - r_{21})s_2\mu_2$ if $\delta = (0, -1)$	$r_{12}s_1\mu_1$ if $\delta = (0, 1)$ $r_{21}s_2\mu_2$ if $\delta = (1, 0)$ $(1 - r_{12})s_1\mu_1$ if $\delta = (-\min(b_1 + 1, b_2 + 1), -\min(b_1, b_2 + 1))$ $(1 - r_{21})s_2\mu_2$ if $\delta = (-\min(b_1 + 1, b_2), -\min(b_1 + 1, b_2 + 1))$

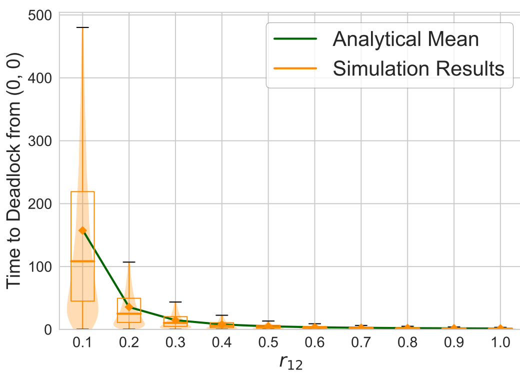
Table 1: Table of transitions $q_{(i_1,j_1),(i_2,j_2)}$ for a multi-server two node network.



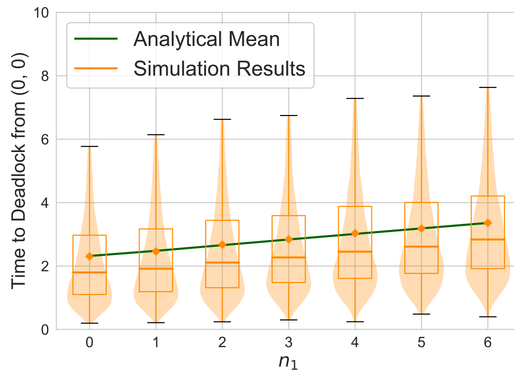
(a) Varying Λ_1



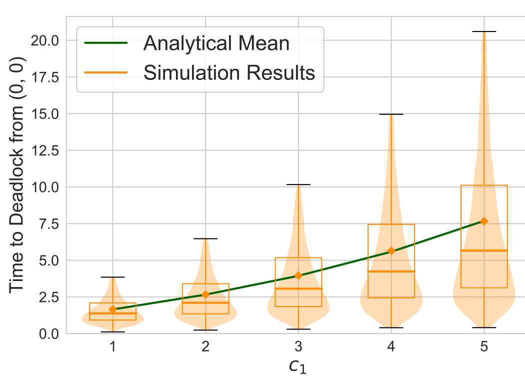
(b) Varying μ_1



(c) Varying r_{12}



(d) Varying n_1



(e) Varying c_1

Figure 11: Time to deadlock in multi-server Ω_2, analytical & simulation results (10,000 repetitions).

5.3 Two Node Single-Server with Self-Loops

Consider the open two node single-server restricted queueing network shown in Figure 12. This shows two $M/M/1/n_i$ queues with service rates μ_i and external arrival rates Λ_i. All routes are possible, where the routing probability from node i to node j is denoted by r_{ij}.

Let this system be denoted by Ω with parameter set (Λ_1, Λ_2, μ_1, μ_2, n_1, n_2, r_{11}, r_{12}, r_{21}, r_{22}).

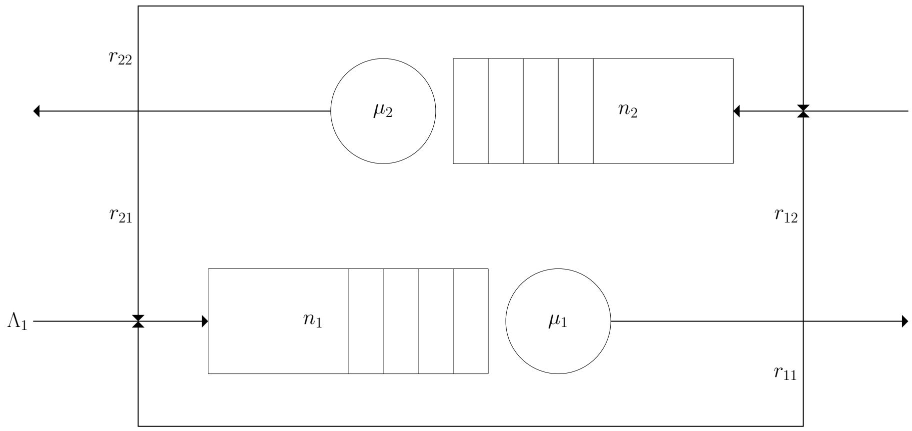


Figure 12: An open two node single-server restricted queueing network.

The state space is given by:

$$S = \{(i,j) \in \mathbb{N}^{(n_1+2)\times(n_2+2)} \mid 0 \leq i + j \leq n_1 + n_2 + 2\} \cup \{(-1), (-2), (-3)\}$$

where i denotes the 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 the 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 second.

and the state (-3) denotes the deadlocked state caused by both nodes; (-1) denotes the deadlocked state caused by the first node only; and (-2) denotes the deadlocked state caused by the second node only.

Define $\delta = (i_2, j_2) - (i_1, j_1)$ for all $(i_k, j_k) \in S$. The transitions are given by Equations 5.5, 5.6, 5.7 and 5.8.

$$
q_{(i_1,j_1),(i_2,j_2)} = \begin{cases} \left.\begin{array}{rr} \Lambda_1 & \text{if } i_1 < n_1 + 1 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (1,0) \\ \left.\begin{array}{rr} \Lambda_2 & \text{if } j_1 < n_2 + 1 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (0,1) \\ \left.\begin{array}{rr} (1 - r_{11} - r_{12})\mu_1 & \text{if } j_1 < n_2 + 2 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (-1,0) \\ \left.\begin{array}{rr} (1 - r_{21} - r_{22})\mu_2 & \text{if } i_1 < n_1 + 2 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (0,-1) \\ \left.\begin{array}{rr} r_{12}\mu_1 & \text{if } j_1 < n_2 + 2 \text{ and } (i_1, j_1) \neq (n_1 + 2, n_2) \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (-1,1) \\ \left.\begin{array}{rr} r_{21}\mu_2 & \text{if } i_1 < n_1 + 2 \text{ and } (i_1, j_1) \neq (n_1, n_2 + 2) \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (1,-1) \\ 0 & \text{otherwise} \end{cases} \tag{5.5}
$$

$$
q_{(i_1,j_1),(-1)} = \begin{cases} r_{11}\mu_1 & \text{if } i > n_1 \text{ and } j < n_2 + 2 \\ 0 & \text{otherwise} \end{cases} \tag{5.6}
$$

$$
q_{(i_1,j_1),(-2)} = \begin{cases} r_{22}\mu_2 & \text{if } j > n_2 \text{ and } i < n_1 + 2 \\ 0 & \text{otherwise} \end{cases} \tag{5.7}
$$

$$
q_{(i_1,j_1),(-3)} = \begin{cases} r_{21}\mu_2 & \text{if } (i,j) = (n_1, n_2 + 2) \\ r_{12}\mu_1 & \text{if } (i,j) = (n_1 + 2, n_2) \\ 0 & \text{otherwise} \end{cases} \tag{5.8}
$$

$$
q_{-1,s} = 0 \tag{5.9}
$$

$$
q_{-2,s} = 0 \tag{5.10}
$$

$$
q_{-3,s} = 0 \tag{5.11}
$$

Note that there are now three different deadlock states, thus two more ways to reach deadlock, Equation 5.6 and Equation 5.7.

For $n_1 = 1$ and $n_2 = 2$, the resulting Markov chain is shown in Figure 13.

Figure 14 shows the effect on the time to deadlock of varying the parameters of the above Markov model. Base parameters of $\Lambda_1 = 4$, $\Lambda_2 = 5$, $n_1 = 3$, $n_2 = 2$, $\mu_1 = 10$, $\mu_2 = 8$, $r_{11} = 0.1$, $r_{12} = 0.25$, $r_{21} = 0.15$ and $r_{22} = 0.1$ are used.

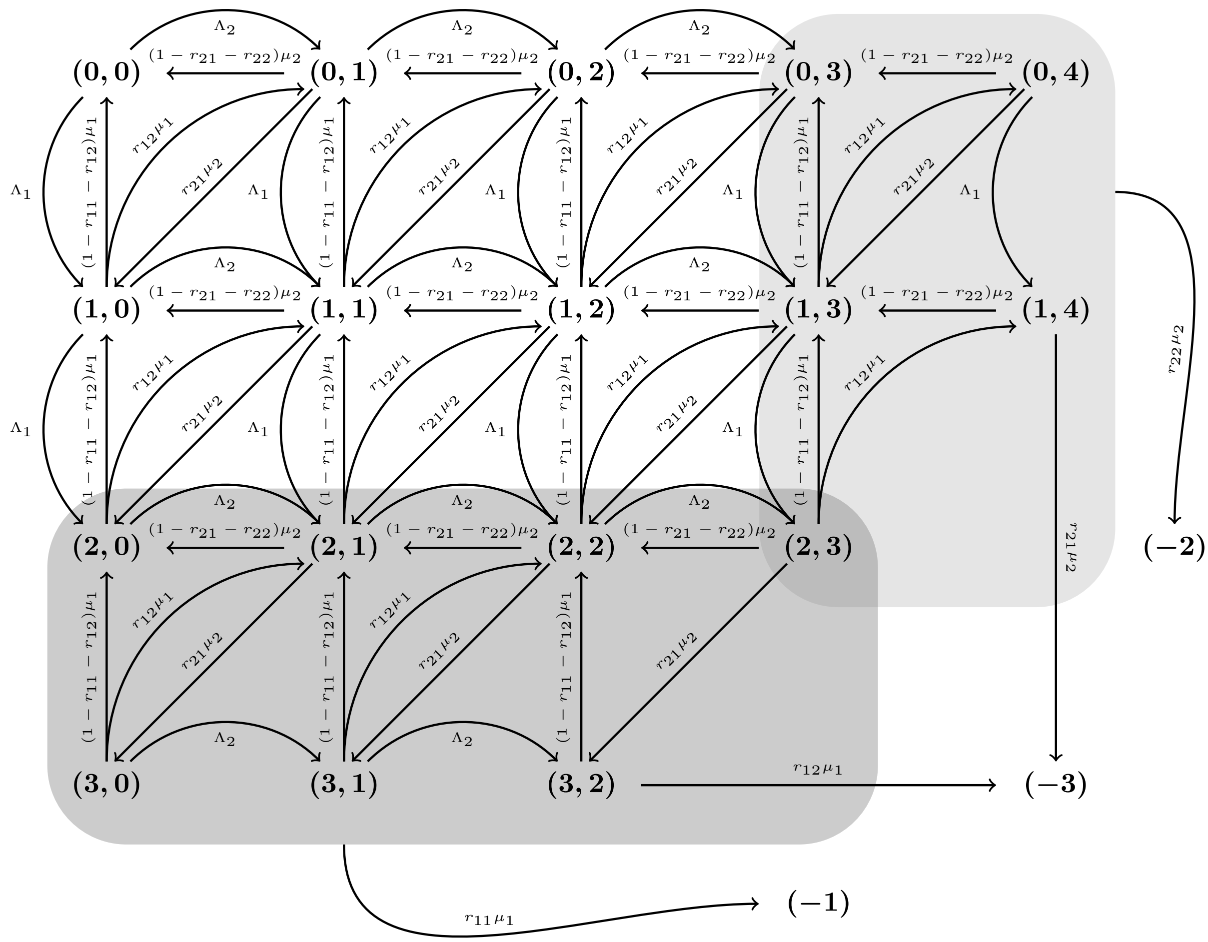
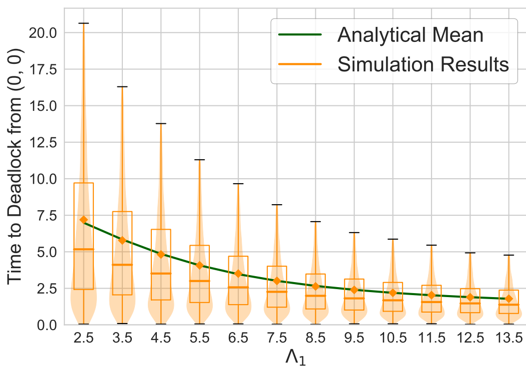
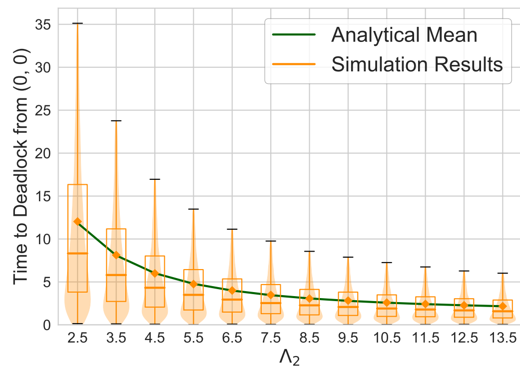


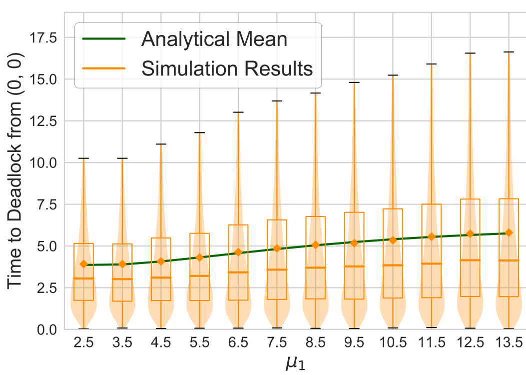
Figure 13: Diagrammatic representation of the Markov chain for Ω with $n_1 = 1$ and $n_2 = 2$.



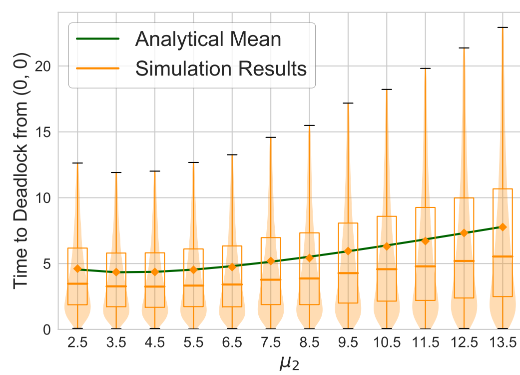
(a) Varying Λ_1



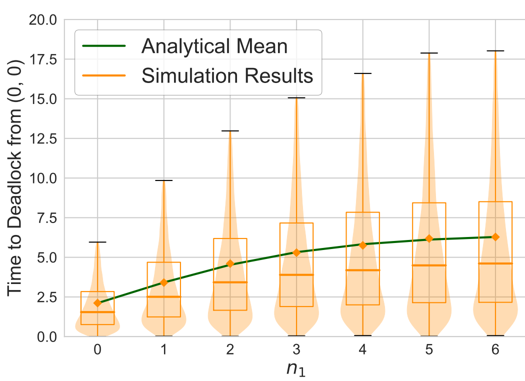
(b) Varying Λ_2



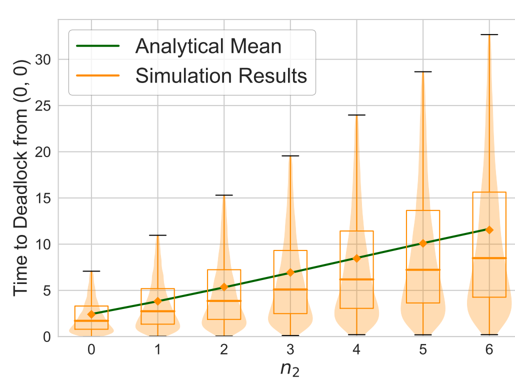
(c) Varying μ_1



(d) Varying μ_2



(e) Varying n_1



(f) Varying n_2

Figure 14: Time to deadlock in Ω, analytical & simulation results (10,000 repetitions).

6 Conclusions

This paper has explored deadlock in open restricted queueing networks. It has been shown that analysing a queueing network's corresponding state digraph is sufficient to detect when deadlock occurs in queueing networks. In general the presence of a knot in the state digraph will highlight that deadlock has occurred in the network, however for special cases only the presence of a weakly connected component with no sink is required. Incorporating this into a simulation model, time to deadlock can be observed.

Markov models of three deadlocking queueing networks have been built. Using linear algebraic techniques the expected time to deadlock from each state was found, and its behaviour as system parameters varied explored. These analytical results were compared with results obtained from the simulation model.

Further research is needed to build a Markov model of the open two node, multi-server restricted queueing network with routes between nodes and feedback loops, that is network 3 from Section 5. In networks 1 and 2 customers only have one potential destination, and so customers may only get blocked from moving to one destination. In network 3 with single servers, although customers have two destinations, a blockage to the same node immediately results in deadlock. In all these cases, the unblocking mechanism is simple, as there is only ever one option of which node a customer joins when unblocked. However in network 3 with multiple servers, there are two destination nodes to which a customer may join when unblocked. Therefore, any representations of any states with blocked customers also need to hold information about these customers' destination nodes.

In addition to this, the order in which customers become blocked is important. In networks 1 and 2 when space become available at a node there is only one other node from which a blocked customer can become unblocked, however in network 3 a node that has space available must accept the customer that has been blocked longest to that node. Therefore all states with blocked customers are also required to record the order in which the customers become blocked. Combining the two requirements above, it is clear that as the number of servers increases, the size of the state space for this queueing network quickly grows combinatorially. Therefore it is not possible to consider this state space in the same way as for networks 1 and 2.

For the Markov models built in this paper Poisson arrivals and exponential service rates were assumed, and only blocking of Type I is considered. A future research direction could be to model other service and arrival distributions using phase-type distributions, and incorporating these into the Markov models of deadlocking queueing networks. Blocking of Type II and III should also be considered, both in the analytical models and whether the deadlock detection method presented here still holds. Systems under Type III blocking with random destination (RS-RD) will not reach deadlock, as there is a non-zero probability of a blocked customer leaving the system. This type of blocking may be considered a deadlock prevention mechanism.

7 Acknowledgements

The authors would like to thank and express gratitude to all anonymous referees whose comments and suggestions have been very helpful.

References

[1] I. Akyildiz. Product form approximations for queueing networks with multiple servers and blocking. *IEEE transactions on computers*, 38(1):99–114, 1989.

[2] G. Allon, S. Deo, and W. Lin. The impact of size and occupancy of hospital on the extent of ambulance diversion: Theory and evidence. *Operations research*, 61(3):544–562, 2013.

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

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

[5] F. Belik. An efficient deadlock avoidance technique. *IEEE transactions on computers*, 39(7):882–888, 1990.

[6] J. Cheng. Task-wait-for graphs and their application to handling tasking deadlocks. In *Proceedings of the conference on TRI-ADA'90*, pages 376–390. ACM, 1990.

[7] 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.

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

[9] Y. Dallery and Y. Frein. On decomposition methods for tandem queueing networks with blocking. *Operations research*, 41(2):386–399, 1993.

[10] B. Deuermeyer, G. Curry, A. Duchowski, and S. Venkatesh. An automatic approach to deadlock detection and resolution in discrete simulation systems. *INFORMS journal on computing*, 9(2):195–205, 1997.

[11] E. Dijkstra. The mathematics behind the bankers algorithm. In *Selected writings on computing: a personal perspective*, pages 308–312. Springer, 1982.

[12] A. Elmagarmid. A survey of distributed deadlock detection algorithms. *ACM sigmod record*, 15(3):37–45, 1986.

[13] J. Ezpeleta, F. Tricas, F. Garca-Valls, and J.M. Colom. A banker's solution for deadlock avoidance in fms with flexible routing and multiresource states. *IEEE transactions on robotics and automation*, 18(4):621–625, 2002.

[14] M. Florian, M. Mahut, and N. Tremblay. Application of a simulation-based dynamic traffic assignment model. *European journal of operational research*, 189(3):1381–1392, 2008.

[15] A. Gibbons. *Algorithmic graph theory.* Cambridge University Press, 1985.

[16] W.J. Gordon and G.F. Newell. Cyclic queuing systems with restricted length queues. *Operations research*, 15(2):266–277, 1967.

[17] R. Holt. Some deadlock properties of computer systems. *ACM computing surveys*, 4(3):179–196, 1972.

[18] N.P.C. Hong, T. Crick, I.P. Gent, and L. Kotthoff. Top tips to make your research irreproducible. *arXiv preprint arXiv:1504.00062*, 2015.

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

[20] P. Kawadkar, S. Prasad, and A.D. Dwivedi. Deadlock avoidance based on bankers algorithm for waiting state processes. *International journal of innovative science and modern engineering*, 2(12), 2014.

[21] 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.

[22] 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.

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

[24] 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.

[25] 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.

[26] A. Manzano-Santaella. From bed-blocking to delayed discharges: precursors and interpretations of a contested concept. *Health services management research*, 23(3):121–127, 2010.

[27] O. Marchetti and A. Munier-Kordon. A sufficient condition for the liveness of weighted event graphs. *European journal of operational research*, 197(2):532–540, 2009.

[28] R. Onvural. Survey of closed queueing networks with blocking. *ACM computing surveys*, 22(2):83–121, 1990.

[29] R. Onvural and H.G. Perros. On equivalencies of blocking mechanisms in queueing networks with blocking. *Operations research letters*, 5(6):293–297, 1986.

[30] C. Osorio and M. Bierlaire. An analytic finite capacity queueing network model capturing the propagation of congestion and blocking. *European journal of operational research*, 196(3):996–1007, 2009.

[31] H.G. Perros, A.A. Nilsson, and Y.C. Liu. Approximate analysis of product-form type queueing networks with blocking and deadlock. *Performance evaluation*, 8(1):19–39, 1988.

[32] S. Reveliotis. Coordinating autonomy. *IEEE robotics & automation magazine*, 22(2):77–94, 2015.

[33] S. Reveliotis. Real-time management of complex resource allocation systems: necessity, achievements and further challenges. *IFAC-PapersOnLine*, 48(7):50–57, 2015.

[34] L. C Schmidt and J. Jackman. Modeling recirculating conveyors with blocking. *European journal of operational research*, 124(2):422–436, 2000.

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

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

[37] S.T. Tan. *Calculus*. Brooks/Cole/Cengage Learning, 2009.

[38] The Ciw library developers. Ciw: 1.0.0, 2017.

[39] The Python Software Foundation. Python 3.5.1. `www.python.org`, 2015.

[40] S. Venkatesh and J. Smith. A graph-theoretic, linear-time scheme to detect and resolve deadlocks in flexible manufacturing cells. *Journal of manufacturing systems*, 22(3):220–238, 2003.

[41] S. Venkatesh and J. Smith. An evaluation of deadlock-handling strategies in semiconductor cluster tools. *IEEE transactions on semiconductor manufacturing*, 18(1):197–201, 2005.

[42] S. Venkatesh, J. Smith, B. Deuermeyer, and G. Curry. Deadlock detection and resolution for discrete-event simulation: multiple-unit seizes. *IIE transactions*, 30(3):201–206, 1998.

[43] I.F.A. Vis. Survey of research in the design and control of automated guided vehicle systems. *European journal of operational research*, 170(3):677–709, 2006.

[44] N. Viswanadham, Y. Narahari, and T.L. Johnson. Deadlock prevention and deadlock avoidance in flexible manufacturing systems using petri net models. *IEEE transactions on robotics and automation*, 6(6):713–723, 1990.

[45] R.J. Wilson. *An introduction to graph theory*. Pearson Education India, 1970.