Deadlocking in Queueing Networks

Geraint Palmer

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.

1 Introduction

The simplest representation of deadlock is shown in Figure 1. Here deadlock, or gridlock, is caused by the mutual blocking of cars. The blue cars are blocked from movement due to the yellow cars; the yellow cars are blocked from movement due to the green cars, the green cars are blocked from movement due to the red cars; and the red cars are blocked from movement due to the blue cars. This can be interperated as the blue cars indirectly blocking themselves.

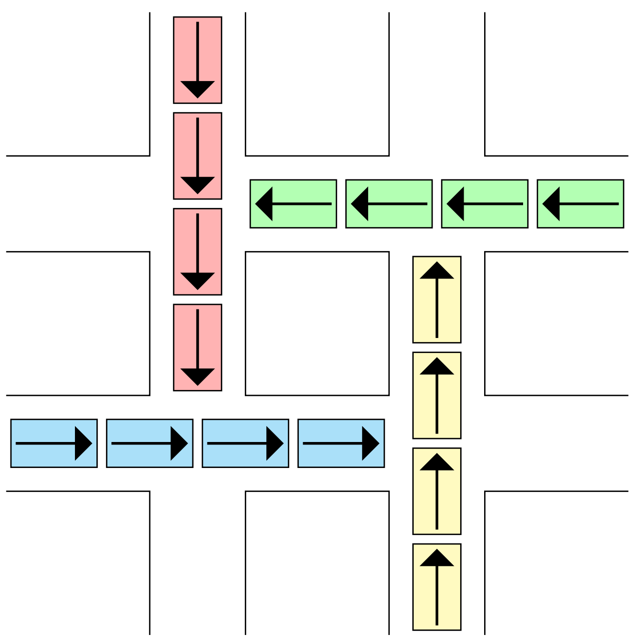


Figure 1: Gridlock, a representation of general deadlock.

This work will investigate deadlocking in queueing networks. The following definition will be used.

Definition 1. *When a simulation is in a situation where at least one service station, despite having arrivals, ceases to begin or 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 restricted 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 2 a simple two node queueing network is shown in a deadlocked state. Customer occupying server A_1 has finished service at node A, but remains there as there is not enough queueing space at node B to accept them. We say the customer at server A_1 is blocked by the customer at server B_1, as he is waiting for that customer to be released. Similarly, the customer occupying server B_1 has finished service at node B, but remains there as there is not enough queueing space at node A, and so the customer at server B_1 is blocked by the customer at server A_1.

When there are multiple servers, individuals become blocked by all customers in service at the destination service station. Figure 3a shows two nodes in deadlock, the customer occupying server A_1 is blocked by customers at both B_1 and B_2, who are both blocked by the customer at A_1. However in 3b, customer at A_1 is blocked by customers at both B_1 and B_2, but the customer at B_2 isn't blocked, and so there is no deadlock.

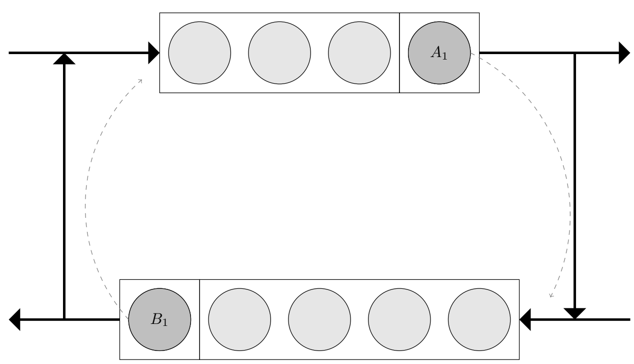
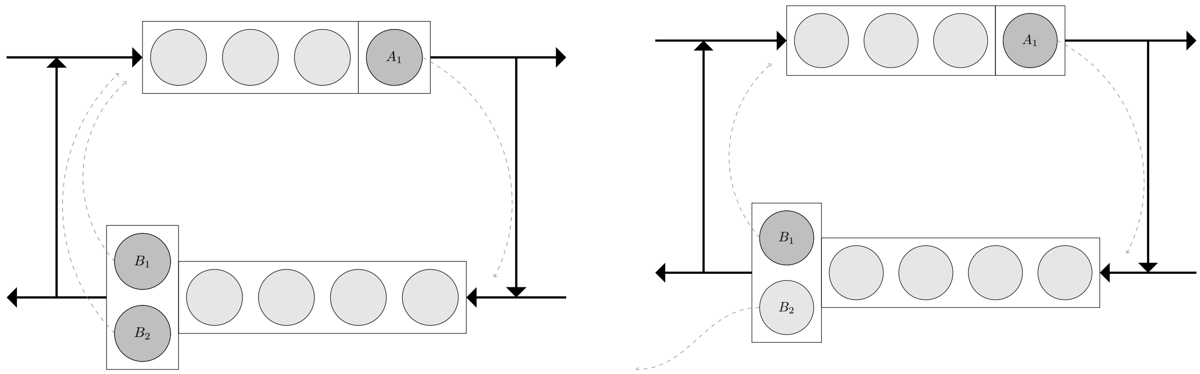


Figure 2: Two nodes in deadlock.

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 4a. This idea is expanded on in Section 3.



(a) Two nodes in deadlock.

(b) Two nodes not deadlocked

Figure 3: Two nodes: a) in deadlock and b) not in deadlock.

2 Literature Review

Restricted queueing networks, or queueing networks with no or limited intermediate queues between service stations are more complicated 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 [6]. Results are derived by writing out and solving the systems' difference equations. The same method is used in [2]. The thesis investigates two and three node systems, as well as systems with one service centre 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 [1]. 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 [8]: (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 [11], 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 [14]. 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 [8], and a similar iterative method was used in [9].

Restricted queueing networks can give rise to the phenomenon of deadlock. This can cause problems for both analytical and simulation models, and most of the literature on blocking either does not consider networks with feedback loops, ignores the possibility of deadlock, or conveniently assumes the networks are deadlock-free.

General deadlock situations that are not specific to queueing networks are discussed in [4]. 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.

In [7] it is noted that in general there are three strategies for dealing with deadlock:

- Prevention, in which the system cannot possibly deadlock in the first place.
- Dynamic avoidance, in which decisions are made as time unfolds to avoid reaching deadlock.
- Detection and recovery.

Deadlock prevention has been discussed in queueing networks. For closed networks of K customers with only one class of customer, [10] 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 [12], 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.

There are algorithms discussed in the literature for the dynamic avoidance of deadlock. In the Banker's Algorithm [5, 7], unsafe states, those that will lead to deadlock, are avoided by ensuring actions leading to these states are not carried out.

General deadlock detection in systems unspecific to queueing networks are discussed in [4]. Here 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 [3] 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 existence within the state graph is sufficient to detect deadlock.

3 Types of Deadlock

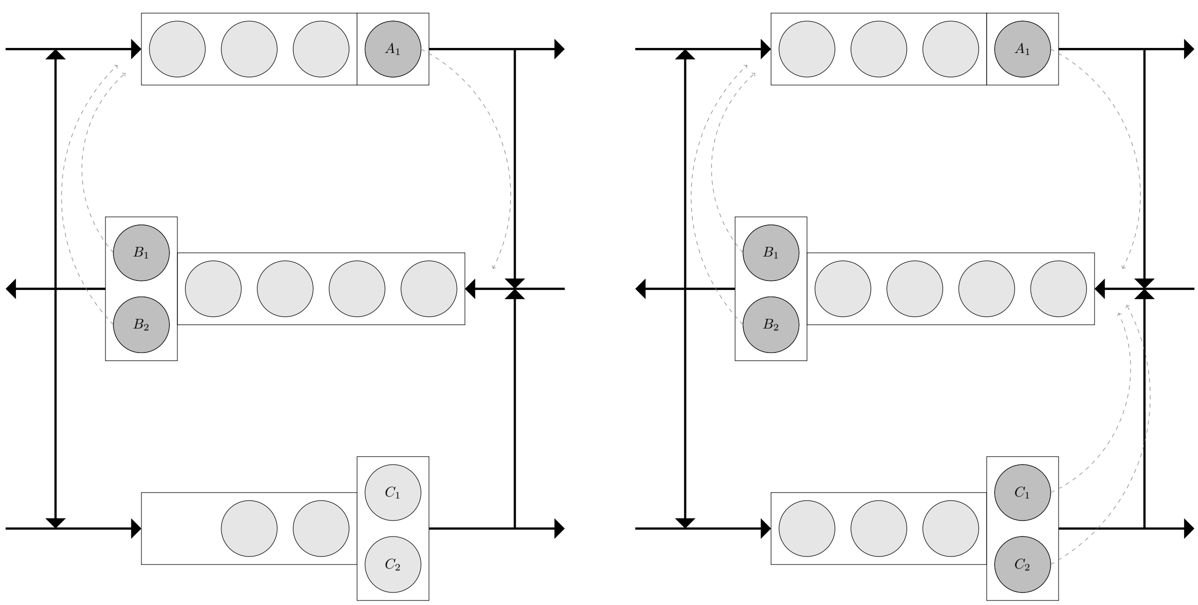
In the introduction an idea was introduced that parts of a queueing network can be in a deadlocked state, although other parts will continue to flow. The different configurations of which nodes experience deadlock can be thought of as different types of deadlock. The amount of different types of deadlock 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.*

Figure 4 shows a three nodes network in a transient deadlocked state, and an absorbing deadlocked state. In Figure 4a the occupants of servers B_1 and B_2 are blocked from entering node A; and the occupant of server A_1 is blocked from entering node B, and so these two nodes are in deadlock. However, node C can continue with regular services, until the occupants of every server of C attempt to join a deadlocked node. At which point, the whole system is deadlocked, and so has reached absorbing deadlock, show in Figure 4b.



(a) Transient deadlock. (b) Absorbing deadlock.

Figure 4: Types of deadlock: a) transient and b) absorbing.

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.

4 Dynamically Detecting Deadlock

In the following subsections, a method will be presented to dynamically detect deadlock in discrete event simulations of queueing networks. The following definitions will be required in this section:

$\lvert V(D)\rvert$	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.

4.1 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, $\lvert V(D(t))\rvert = \sum_{i=1}^{N} c_i$ 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 5.

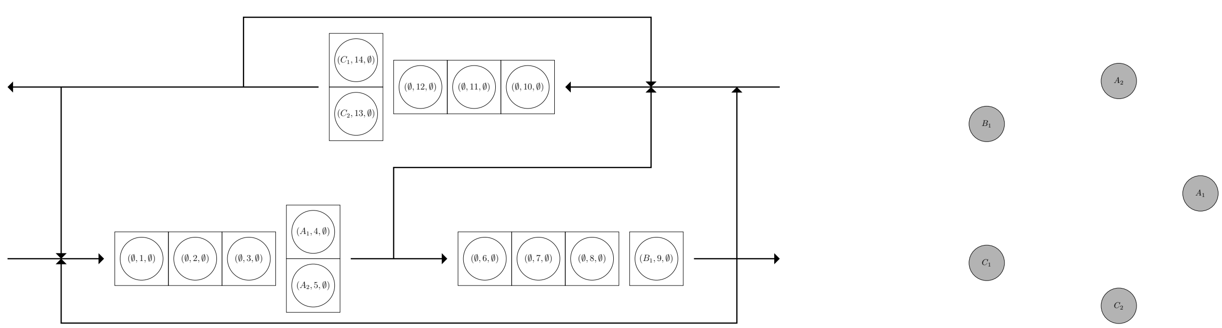


Figure 5

Customer 13 finishes service, and is blocked from entering node A. Figure 6.

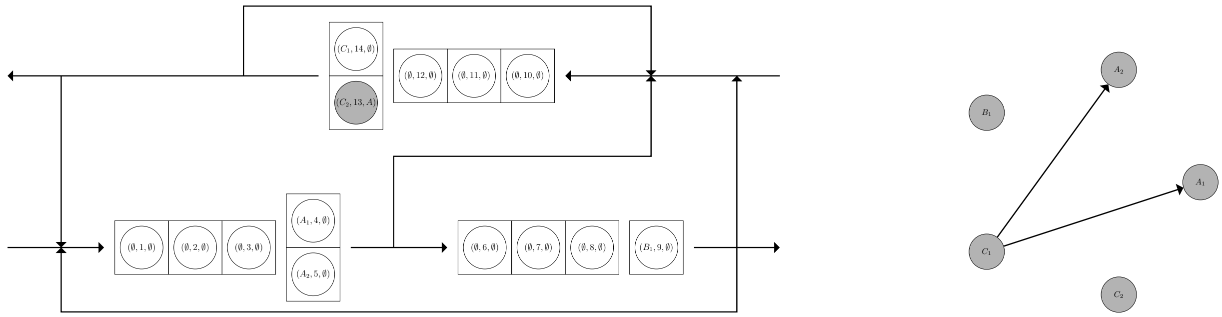


Figure 6

Then customer 4 finishes service and is blocked from entering node B. Figure 7.

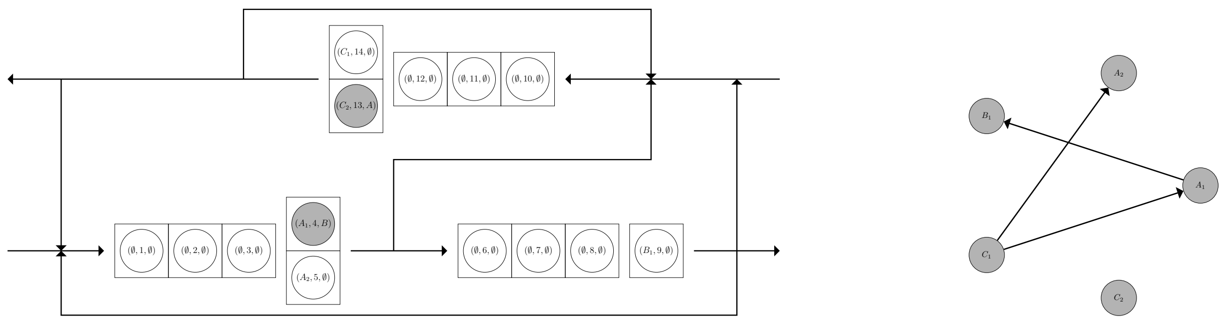


Figure 7

Then customer 9 finishes service and is blocked from entering node A. Figure 8.

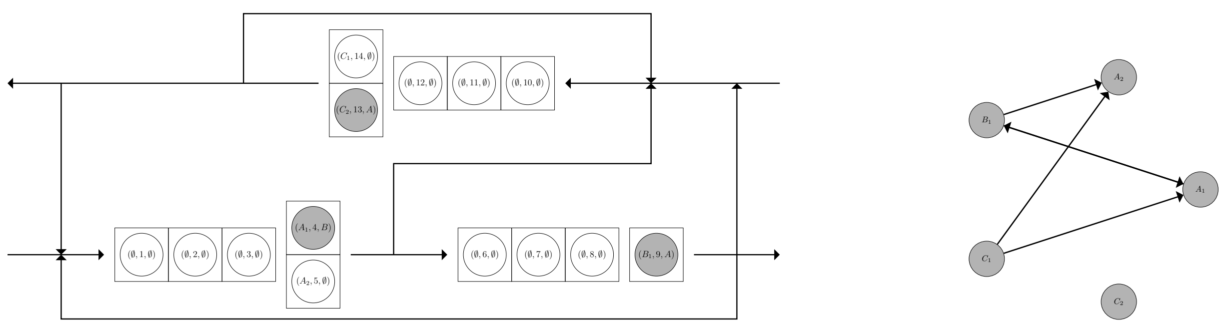


Figure 8

Finally, in Figure 9 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.

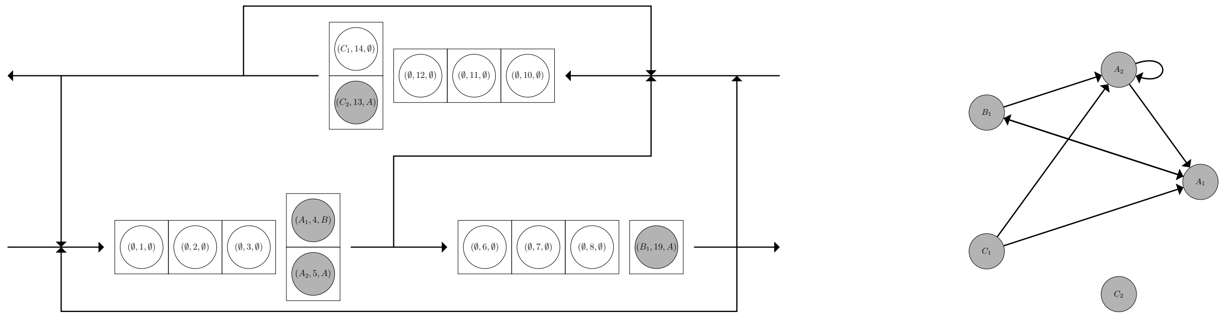


Figure 9

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 10.

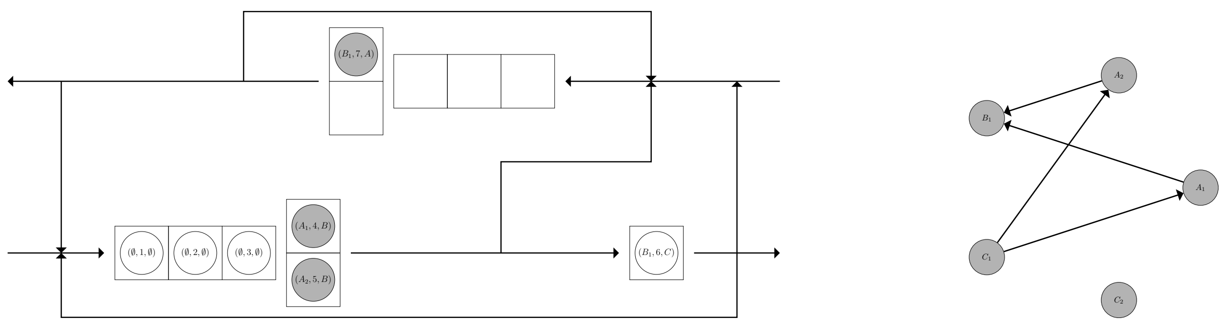


Figure 10

Customer 6 finishes service and immediately joins service at node C. Figure 11.

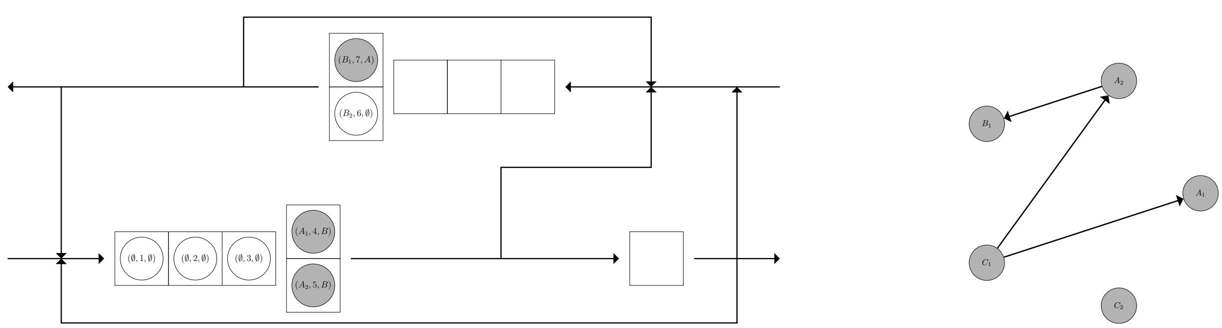


Figure 11

Now there is room for customer 4 to move into service at node B. Figure 12. Notice that the edge $A_2 \longrightarrow B_1$

remains in the state graph, as customer 5 is still blocked by that server.

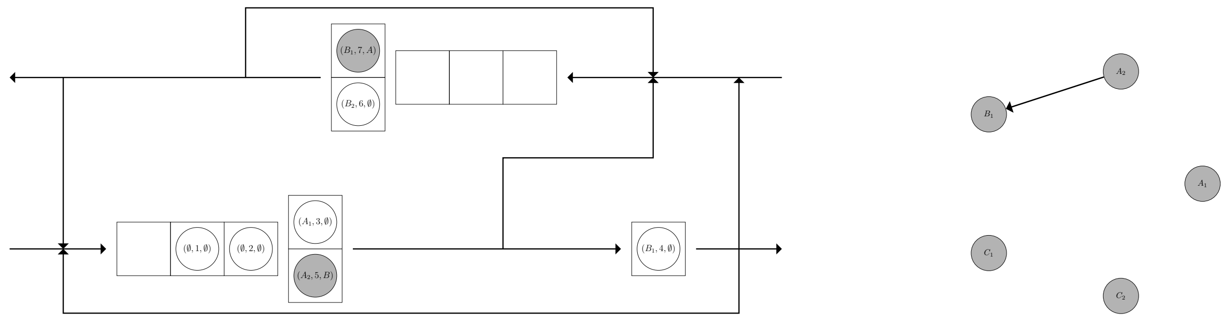


Figure 12

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 13.

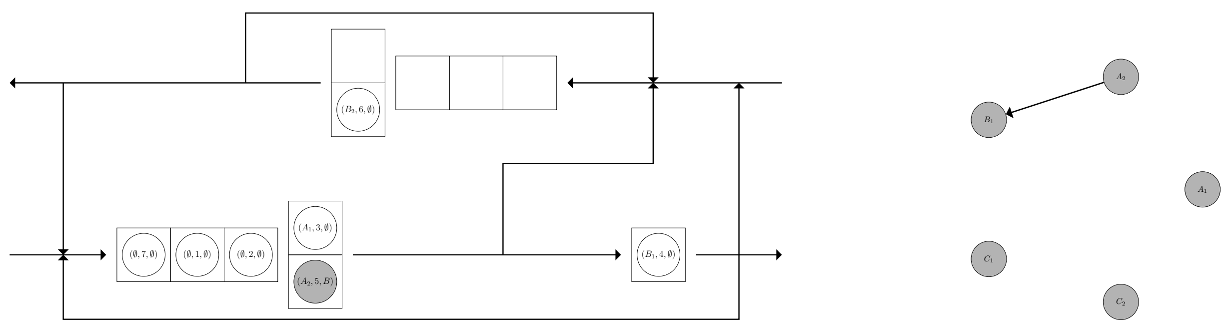


Figure 13

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 descendants 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 $\deg^{\text{out}}(X_a)$ are being 0 or c_j. If $\deg^{\text{out}}(X_a) = c_j$ 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 $\deg^{\text{out}}(X_a) = 0$.

It is clear that if all of X_a's descendants 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 necessarily mean there is deadlock.

4.2 Results on the State Digraph

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 descendants 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 $\deg^{\text{out}}(X) = 0$, 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 Q 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 who is deadlocked, 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. □

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 connected 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

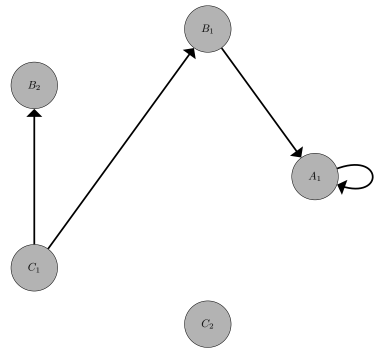


Figure 14: A Counter-Example State Digraph.

blocked from entering node A. The resulting state digraph in Figure 14 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. □

4.3 Finding Knots

By definition knots are strongly connected subgraphs where every member's descendants belong to that subgraph.

Using the Python package NetworkX, finding strongly connected components, and finding a vertex's descendants are built-in methods. The following algorithm, taken from the NetworkX developer zone ticket #663, is sufficient to identify knots in a directed graph.

Find the strongly connected subgraphs of $D(t)$

for *strongly connected subgraph SCS in list of strongly connected subgraphs of* $D(t)$ **do**

  for *vertex v in SCS* **do**

    if *number of v's decendants > number of vertices in SCS* **then**

      Add SCS to list of knots

      break

    end

  end

end

5 Markov Chain Models

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 [13] that for a discrete transition matrix of the form $P = \left(\begin{smallmatrix} T & U \\ 0 & I \end{smallmatrix}\right)$ then the expected number of time steps until absorption starting from state i is the ith element of the vector

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

where e is a vector of 1s.

5.1 One Node Network

Consider the one node network with feedback loop shown in Figure 15. 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 $1 - r_{11}$.

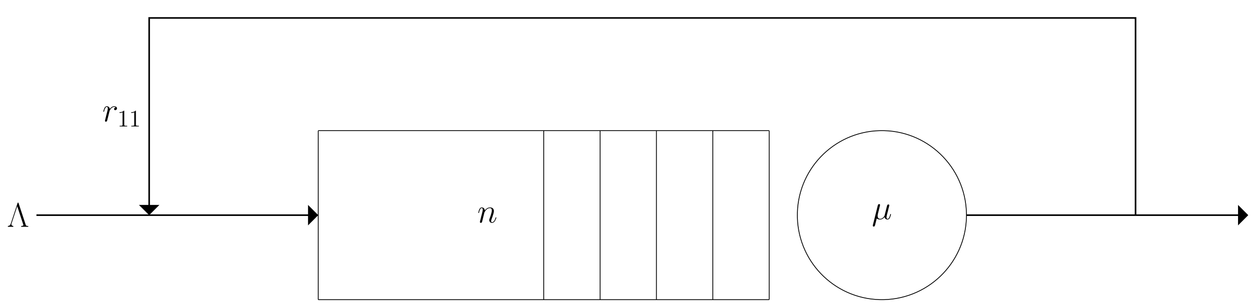


Figure 15: A one node queueing network.

State space:

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

where i denotes the number of individuals in service or waiting.

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

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

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

and

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

The Markov chain is shown in Figure 16.

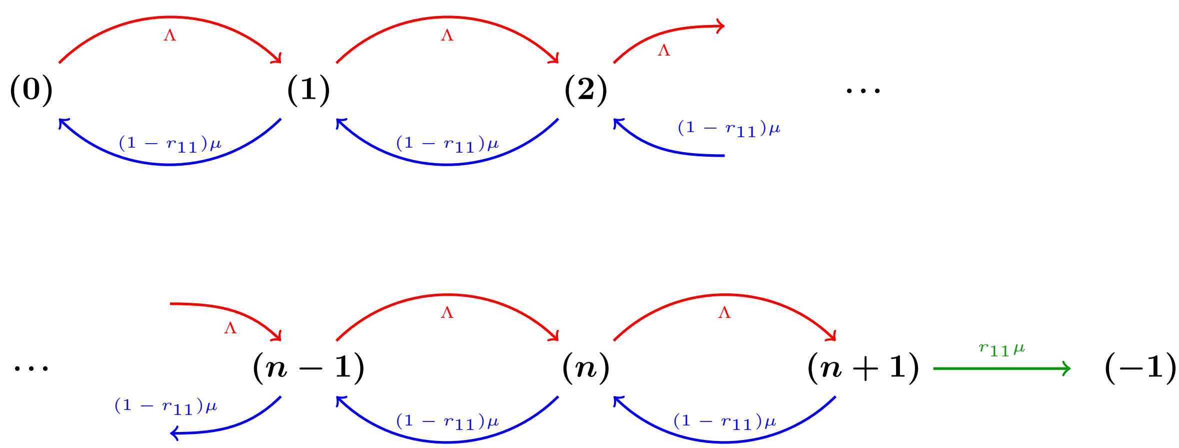
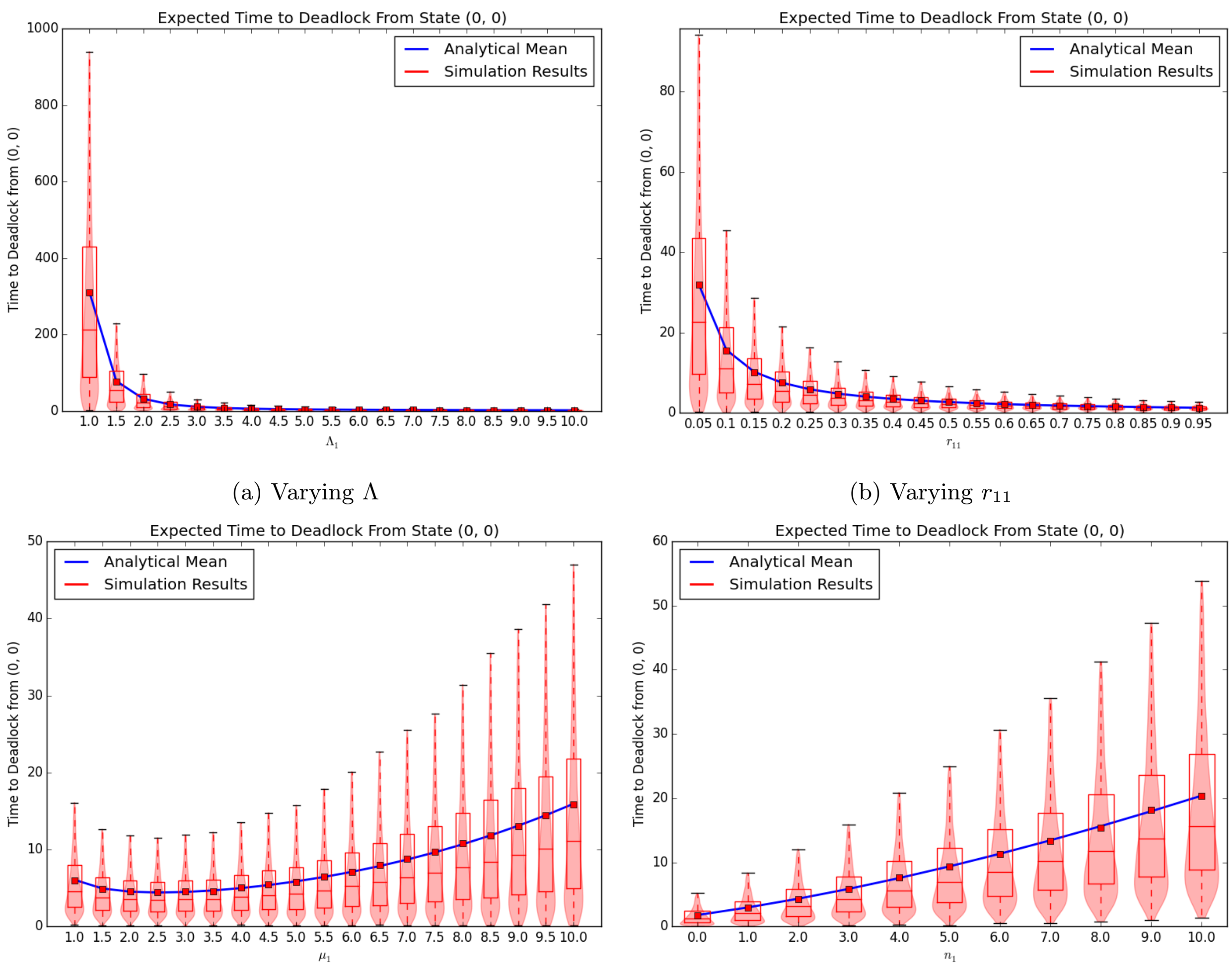


Figure 16: Markov chain of the one node system.

Figure 17 shows the effect of varying the parameters of the queueing network on times to deadlock. Base parameters of $\Lambda = 10$, $n = 3$, $\mu = 5$ and $r_{11} = 0.25$ were used.

We can see that increasing the arrival rate Λ and the transition probability r_{11} results in reaching deadlock faster. This is intuitive as increasing these parameters results in the first node's queue filling up quicker. Increasing the queueing capacity n results in reaching deadlock slower. Again this are intuitive, as increasing the queueing capacity allows more customers in the system before becoming deadlock.

We get interesting behaviour as the service rate μ varies, as the service rate contributed towards both moving



(a) Varying Λ

(b) Varying r_{11}

(c) Varying μ

(d) Varying n

Figure 17: Analytical & Simulation Results of Times to Deadlock (10,000 iterations)

customers from the system and allowing customers to rejoin the queue, causing blockages and deadlock. This behaviour can be interpreted as follows:

- At low service rates below a certain threshold, 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 this threshold 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 deadlocked.

This effect is closely related to the transition rate r_{11}, as the rate at which the system enters deadlock from a full queue is $r_{11}\mu$. Figure 18 shows the effect of the transition rate on the behaviour of varying the service rate.

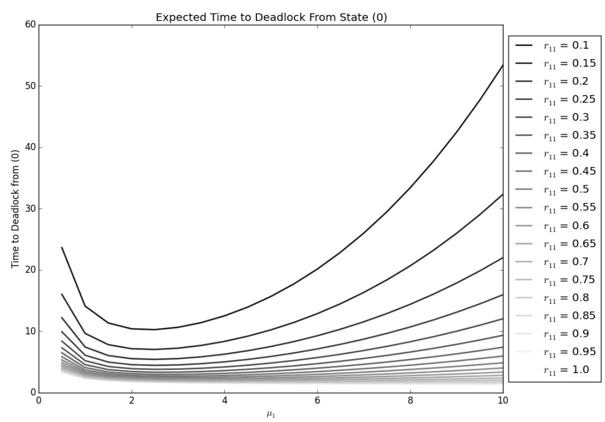


Figure 18: The effect of r_{11} and μ on times to deadlock.

5.2 Two Node Network without Self Loops

Consider the queueing network shown in Figure 19. 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}.

- State space:

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

  where i denotes the number of individuals:

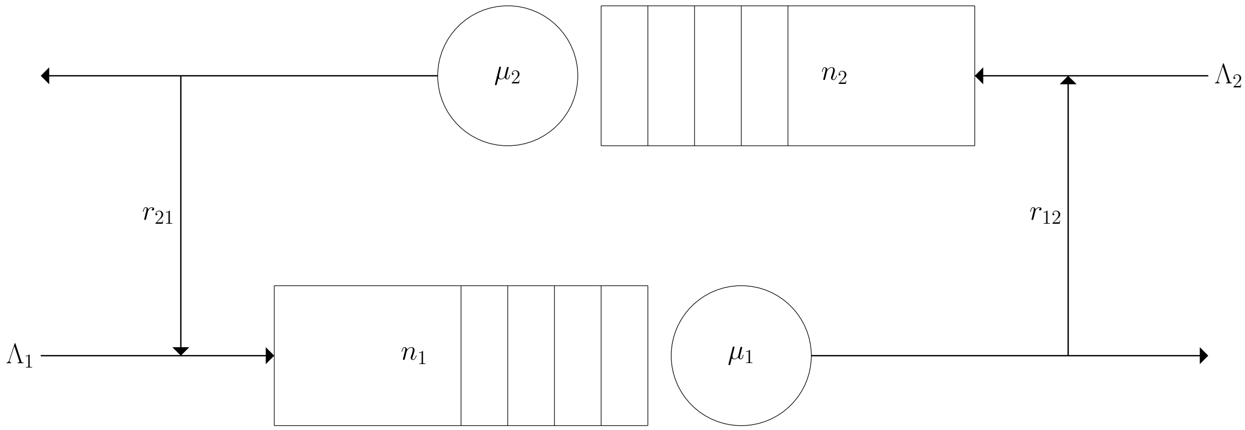


Figure 19: A two node queueing network.

- 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 (-1) denotes the deadlocked state.

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

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

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

and

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

For $n_1 = 1$ and $n_2 = 2$, the resulting Markov chain is shown in Figure 20.

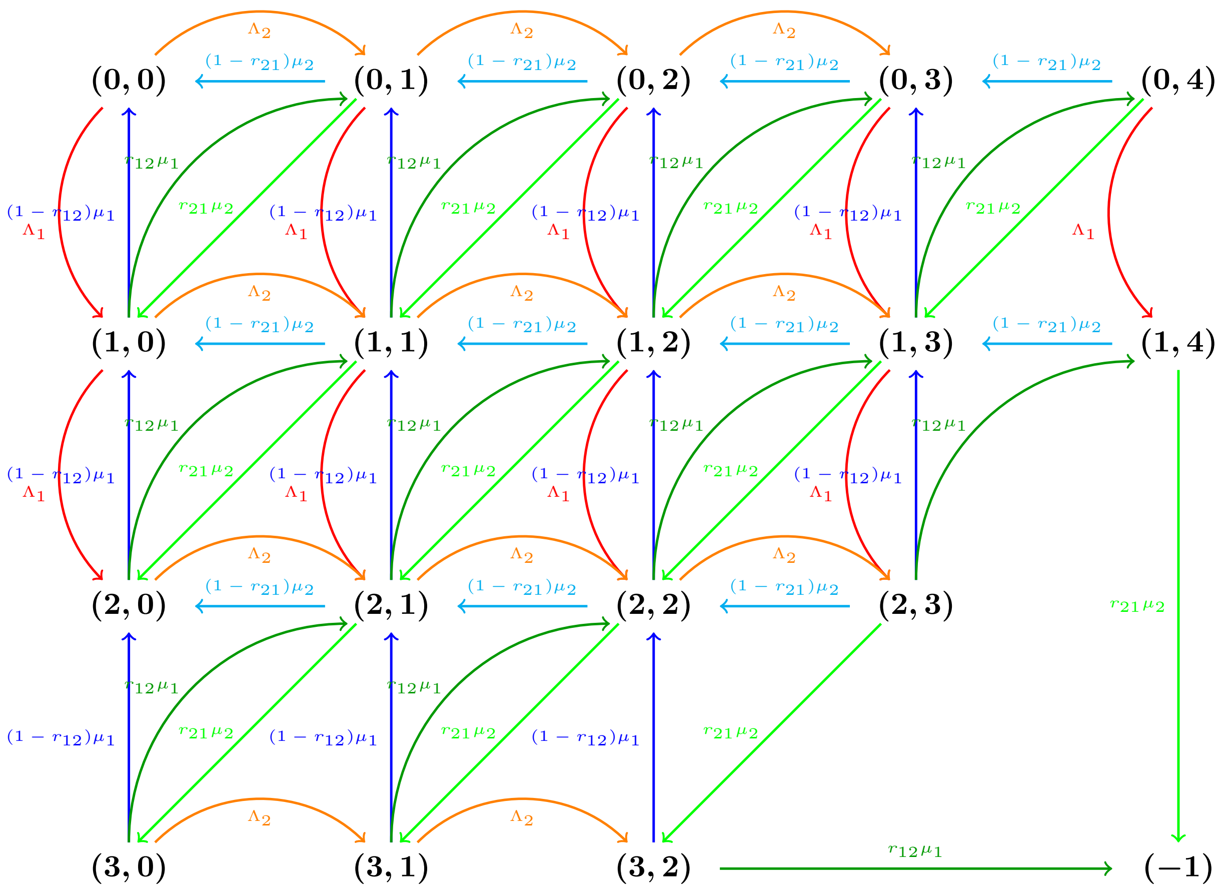
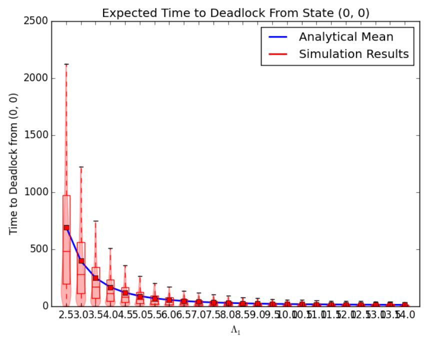


Figure 20: Markov chain of the two node system without self loops, $n_1 = 1$ and $n_2 = 2$.

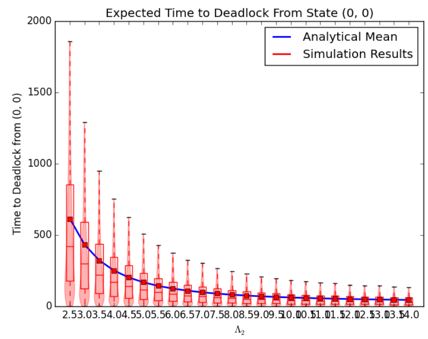
Figure 21 shows the effect of varying the parameters of the above Markov model. Base parameters of $\Lambda_1 = 4$, $\Lambda_2 = 5$, $n_1 = 3$, $n_2 = 2$, $\mu_1 = 10$, $\mu_2 = 8$, $r_{12} = 0.25$ and $r_{21} = 0.15$ were used. We can see that we get similar behaviour as the 1 node network.

5.3 Two Node Network each with Self Loops

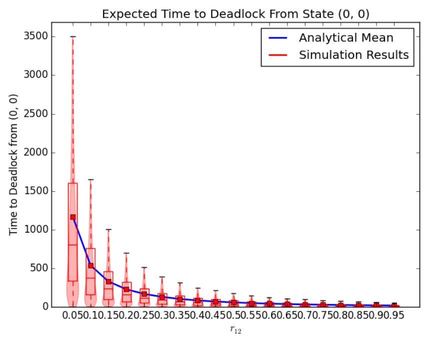
Consider the queueing network shown in Figure 22. This shows two $M/M/1$ queues, with n_i queueing capacity at at each service station and service rates μ_i. Λ_i is the external arrival rates to each service station. All routing possibilities are possible, where the routing probability from node i to node j is denoted by r_{ij}.



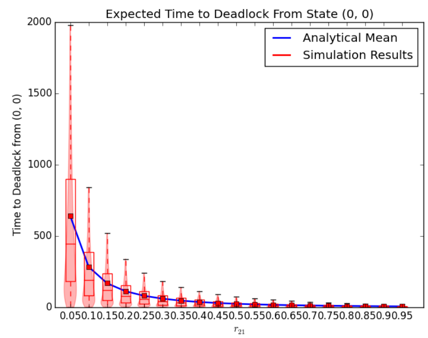
(a) Varying Λ_1



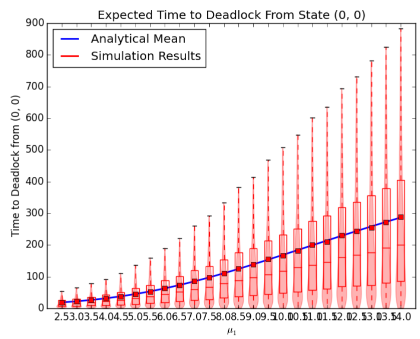
(b) Varying Λ_2



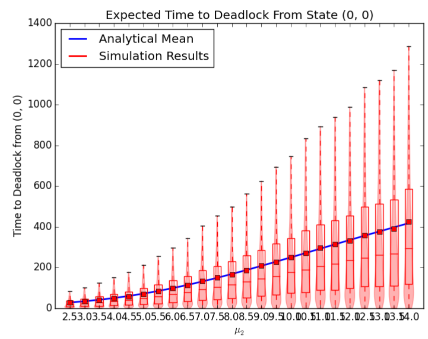
(c) Varying r_{12}



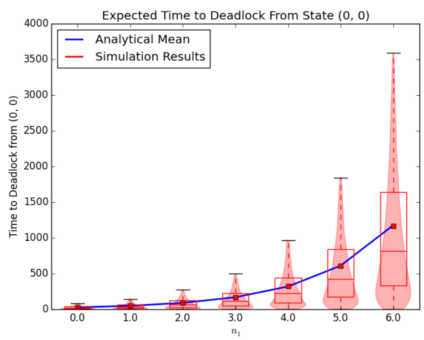
(d) Varying r_{21}



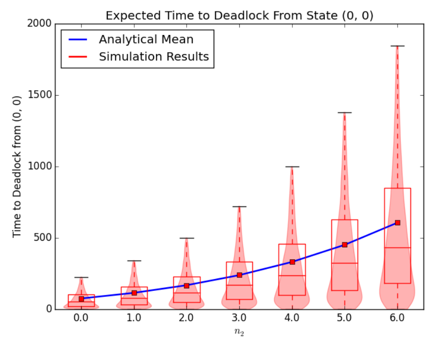
(e) Varying μ_1



(f) Varying μ_2



(g) Varying n_1



(h) Varying n_2

Figure 21: Analytical & Simulation Results of Times to Deadlock (10,000 iterations)

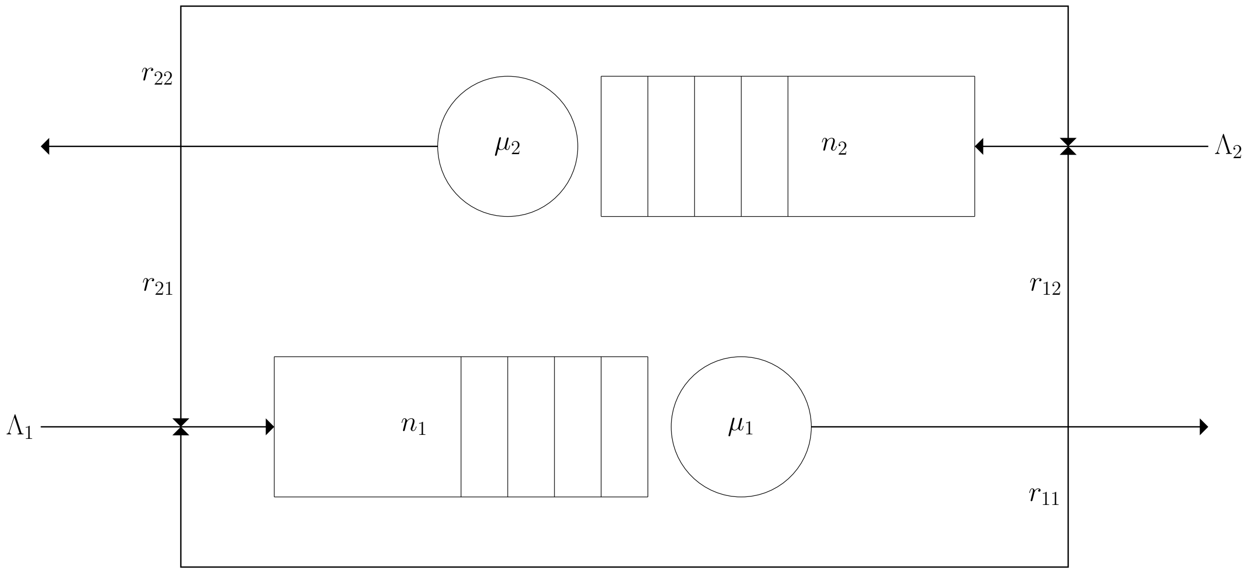


Figure 22: A two node queueing network, each with feedback loops.

- State space:

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

  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 cause 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.

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

$$q_{(i_1,j_1),(i_2,j_2)} = \begin{cases} \left.\begin{array}{rr} \Lambda_1 & \text{if } i_1 < n_1 + 1 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (1,0) \\ \left.\begin{array}{rr} \Lambda_2 & \text{if } j_1 < n_2 + 1 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (0,1) \\ \left.\begin{array}{rr} (1 - r_{11} - r_{12})\mu_1 & \text{if } j_1 < n_1 + 2 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (-1,0) \\ \left.\begin{array}{rr} (1 - r_{21} - r_{22})\mu_2 & \text{if } i_1 < n_1 + 2 \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (0,-1) \\ \left.\begin{array}{rr} r_{12}\mu_1 & \text{if } j_1 < n_2 + 2 \text{ and } (i_1,j_1) \neq (n_1+2, n_2) \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (-1,1) \\ \left.\begin{array}{rr} r_{21}\mu_2 & \text{if } i_1 < n_1 + 2 \text{ and } (i_1,j_1) \neq (n_1, n_2+2) \\ 0 & \text{otherwise} \end{array}\right\} & \text{if } \delta = (1,-1) \\ 0 & \text{otherwise} \end{cases} \tag{8}$$

$$q_{(i_1,j_1),(-1)} = \begin{cases} r_{11}\mu_1 & \text{if } i > n_1 \text{ and } j < n_2 + 2 \\ 0 & \text{otherwise} \end{cases} \tag{9}$$

$$q_{(i_1,j_1),(-2)} = \begin{cases} r_{22}\mu_2 & \text{if } j > n_2 \text{ and } i < n_1 + 2 \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

$$q_{(i_1,j_1),(-3)} = \begin{cases} r_{21}\mu_2 & \text{if } (i,j) = (n_1, n_2+2) \\ r_{12}\mu_1 & \text{if } (i,j) = (n_1+2, n_2) \\ 0 & \text{otherwise} \end{cases} \tag{11}$$

and

$$q_{-1,s} = 0 \tag{12}$$

$$q_{-2,s} = 0 \tag{13}$$

$$q_{-3,s} = 0 \tag{14}$$

Note that the there are only two differences between this formulation and the formulation given in Subsection 5.2: the probabilities of leaving nodes 1 and 2 are now $(1 - r_{11} - r12)\mu_1$ and $(1 - r_{21} - r_{22})\mu_2$; and there are now two more ways to reach deadlock, Equation 9 and Equation 10.

For $n_1 = 1$ and $n_2 = 2$, the resulting Markov chain is shown in Figure 23.

Figure 24 shows the effect 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 $r_{22} = 0.1$ were used.

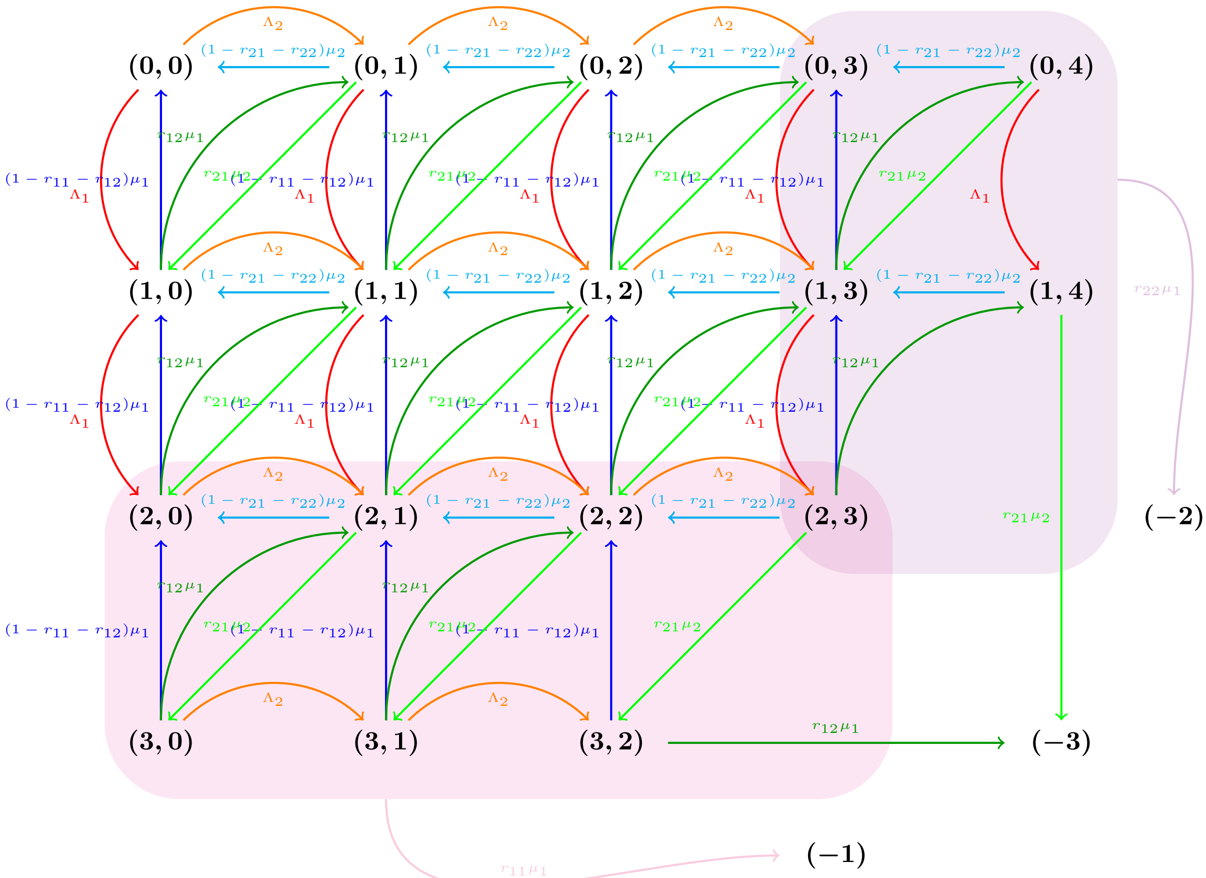
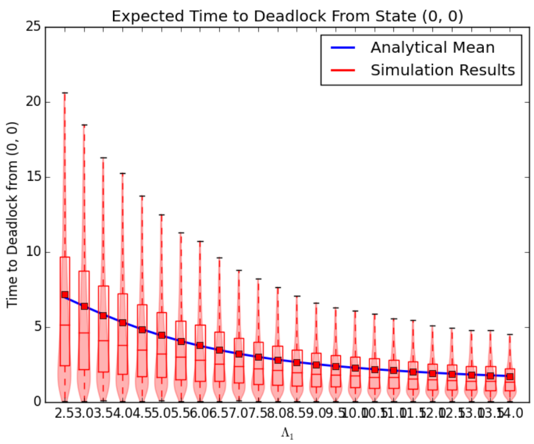
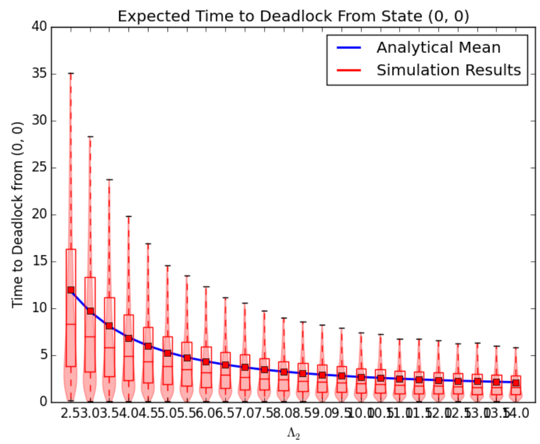


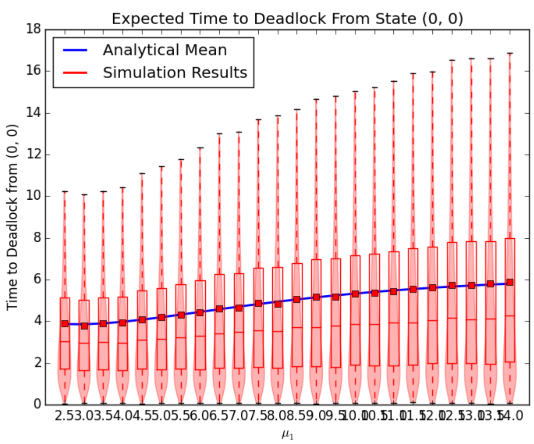
Figure 23: Markov chain of the two node system with self loops, $n_1 = 1$ and $n_2 = 2$.



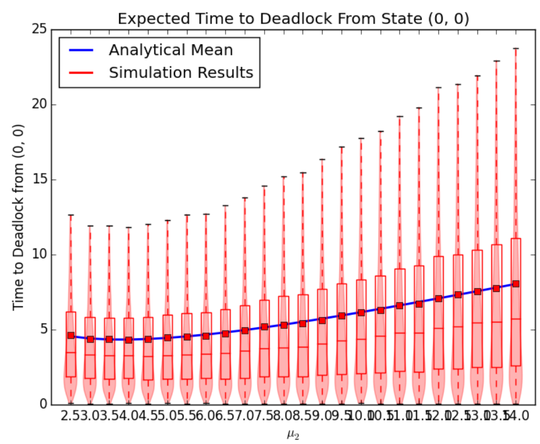
(a) Varying Λ_1



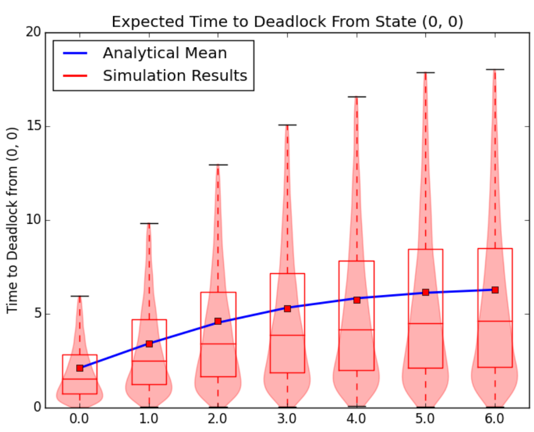
(b) Varying Λ_2



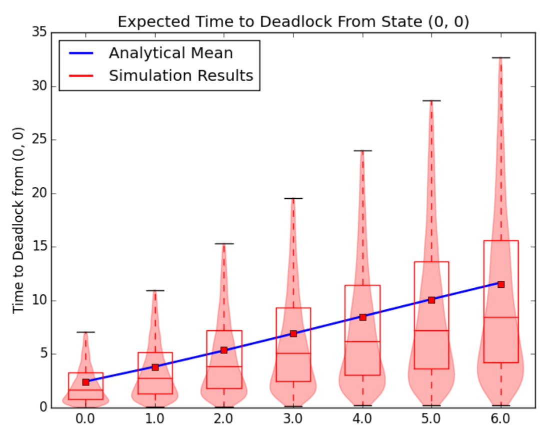
(c) Varying μ_1



(d) Varying μ_2



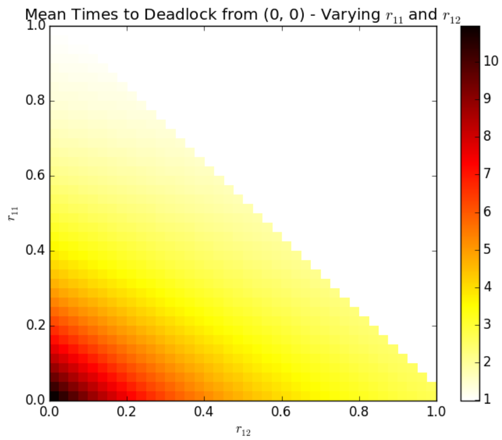
(e) Varying n_1



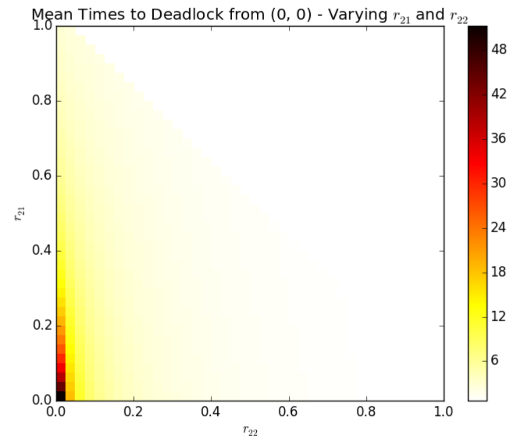
(f) Varying n_2

Figure 24: Analytical & Simulation Results of Times to Deadlock (10,000 iterations)

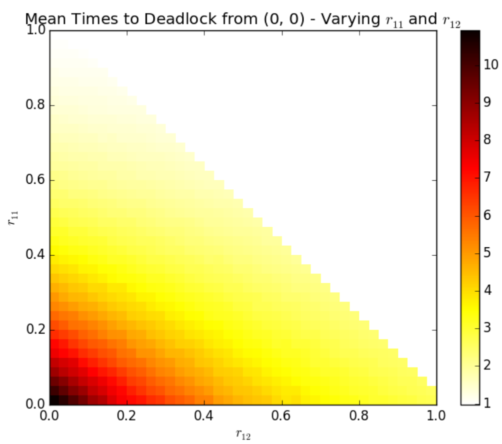
The heatmaps in Figure 25 illustrate how varying the two transition probabilities out of each node affects the time to deadlock. Note the shape of the heatmap, this is due to the restriction than $r_{11} + r12 \leq 1$ and $r_{21} + r_{22} \leq 1$. We can see for both nodes it is the rejoining probability (r_{11} and r_{22}) that has the most drastic effect on time to deadlock. This effect is greater for Node 2, the node that has the smaller queueing capacity.



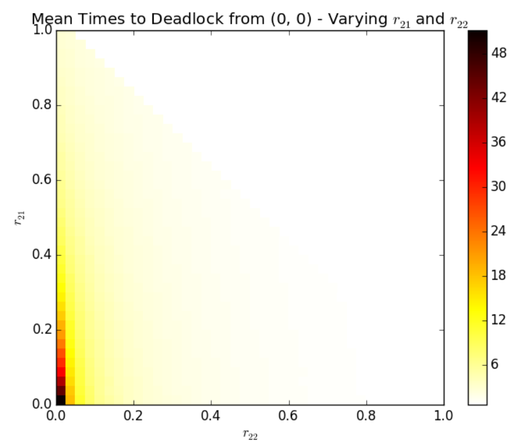
(a) Analytical: Varying transition probabilities at Node 1



(b) Analytical: Varying transition probabilities at Node 2



(c) Simulation: Varying transition probabilities at Node 1



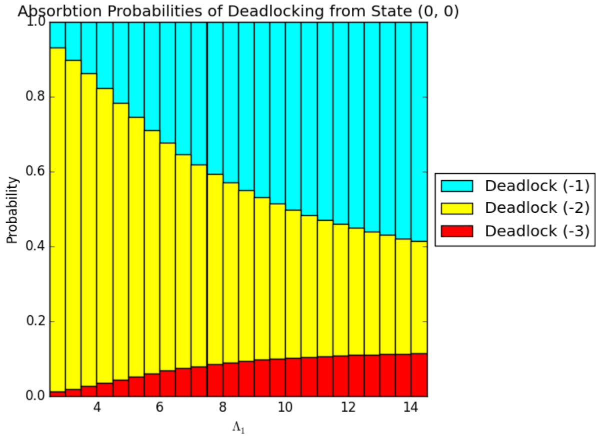
(d) Simulation: Varying transition probabilities at Node 2

Figure 25: Analytical & Simulation Results of Times to Deadlock, varying Transition Probabilities (10,000 iterations)

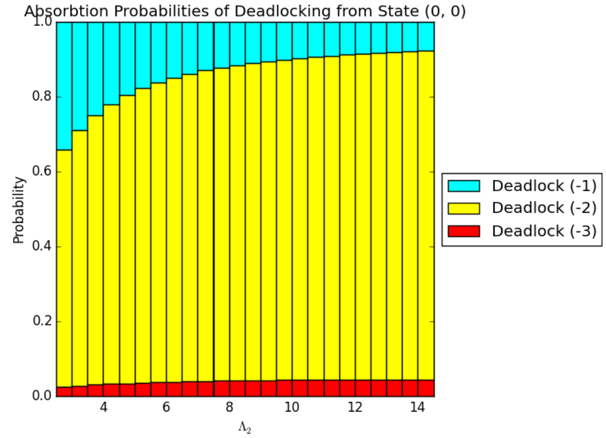
Times to deadlock in this case means the time to the first instance of deadlock. There is however three different deadlocked states which the queueing network can fall into, -1, -2 and -3.

A method is shown in [13] to find the probabilities of which absorbing state a Markov chain will reach. The discrete transition matrix must be in the form $P = \left(\begin{smallmatrix} T & U \\ 0 & I \end{smallmatrix}\right)$. Then $A = (I - T)^{-1}U$, and the $(i, j)^{\text{th}}$ element of A corresponds to the probability of reaching absorbing state j from transient state i.

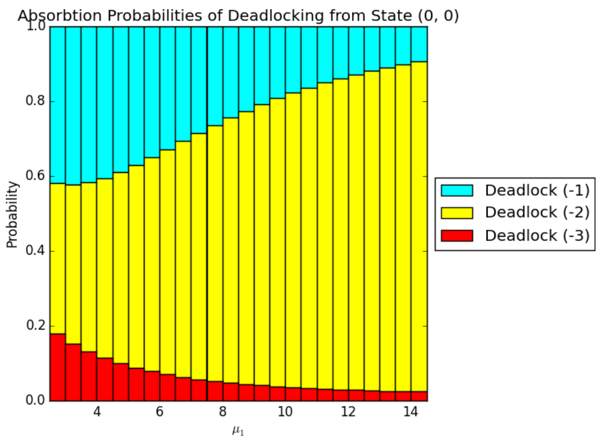
Figure 26 shows how varying the parameters of the queueing network affects the absorption probabilities.



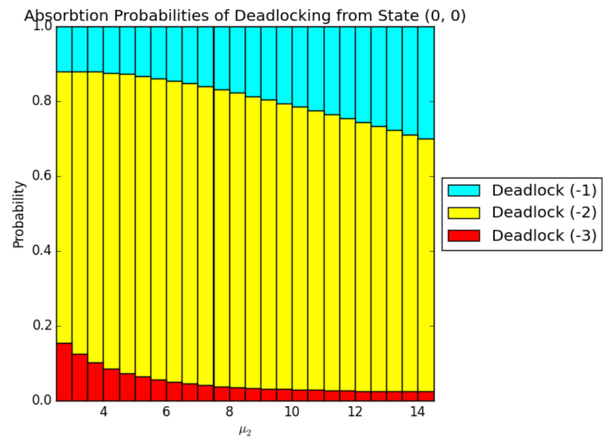
(a) Varying Λ_1



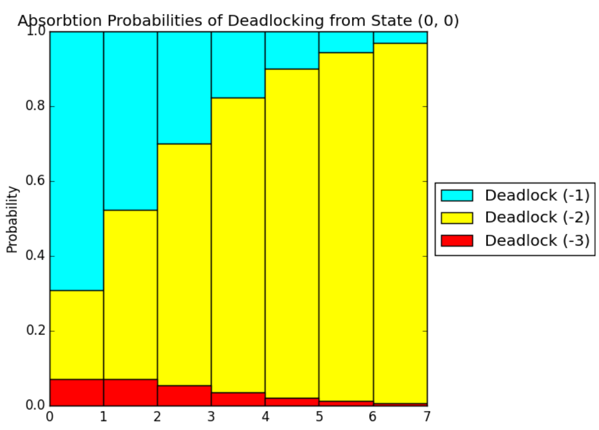
(b) Varying Λ_2



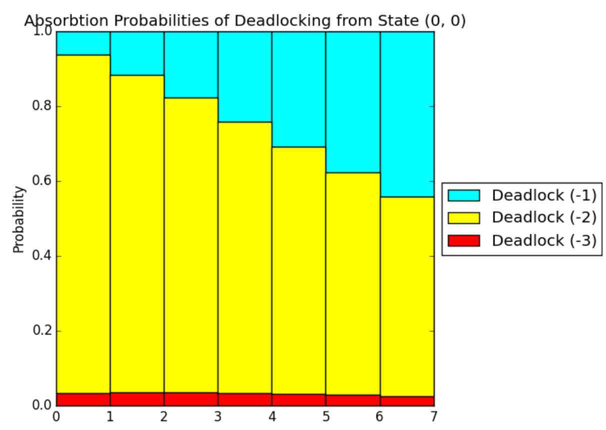
(c) Varying μ_1



(d) Varying μ_2



(e) Varying n_1



(f) Varying n_2

Figure 26: Probabilities of reaching each deadlocked state.

Figure 24e shows interesting behaviour. Increasing n_1 causes a longer time to deadlock up to a point, where increasing n_1 any more does not affect the time to deadlock. This is due to the fact that there are other ways to get to deadlock, and we are only interested in the first instance of any of these deadlocks. Getting to deadlocked state (-2) is not affected by n_1, and so once n_1 goes over a certain threshold the system always reaches (-2) before any other deadlocked state, and so the time to deadlock is unaffected by increasing n_1 any further.

This behaviour will be exhibited while n_2 increases too, however the threshold will take longer to reach as the base parameter for n_1 is larger than the base parameter for n_2.

This behaviour is highlighted in the heatmap shown in Figure 27, where the analytical mean time to deadlock is shown with combinations of n_1 and n_2. For each value of n_2, once a certain threshold of n_1 is reached no more change in time to deadlock occurs. Similarly for each value of n_1, once a certain threshold o n_2 is reached no more change in time to deadlock occurs. The behaviour is unsymmetrical due to the other parameters associated with Node 1 and Node 2 being unsymmetrical.

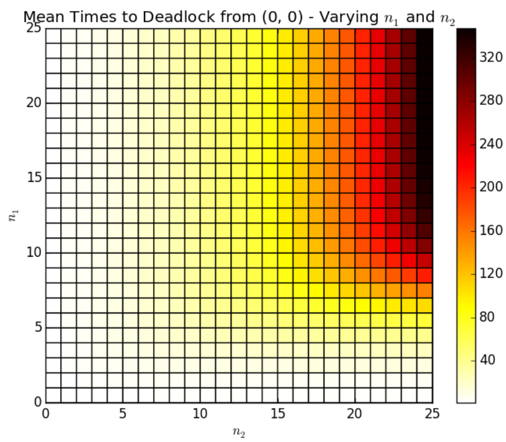
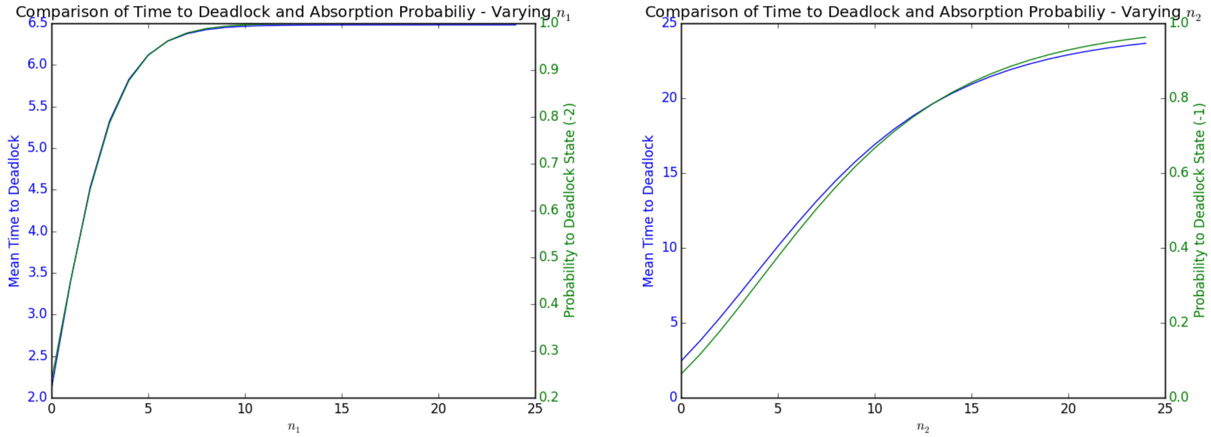


Figure 27: Deadlocking behaviour, varying n_1 and n_2

Figure 28 compares the deadlocking behaviour as n_1 and n_2 increase to absorption probabilities. Time to deadlock while varying n_1 is compared to the probability of absorption to state (-2), as n_1 plays no part in getting to state (-2). Similarly time to deadlock while varying n_2 is compared to the probability of absorption to state (-1). Here it is shown that once it is almost certain that deadlock will be caused by the other node only, then increasing the queueing capacity at this node does not affect time to deadlock.

5.4 Multi-Server Networks

It is possible to model the networks discussed in Section 5.1 and Section 5.2 where each node has multiple parallel servers. These models are discussed in the next subsections.



(a) Varying n_1

(b) Varying n_2

Figure 28: Comparing time to deadlock and absorbtion probabilities as queue capacities increase.

5.4.1 Multi-Server: One Node Network

Consider the one node network with feedback loop discussed in Subsection 5.1, now with c parallel servers.

State space:

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

where i denotes the number of individuals at the node plus the number of individuals blocked at that node. 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.

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

$$q_{i_1,i_2} = \begin{Bmatrix} \Lambda & \text{if } \delta = 1 \\ (1 - r_{11})\mu\min(i, c) & \text{if } \delta = -1 \\ 0 & \text{otherwise} \end{Bmatrix} \quad \text{if } i_1 < n + c \tag{15}$$

$$q_{i_1,i_2} = \begin{Bmatrix} (c - k)r_{11}\mu & \text{if } \delta = 1 \\ (1 - r_{11})(c - k)\mu & \text{if } \delta = -j - 1 \\ 0 & \text{otherwise} \end{Bmatrix} \quad \text{if } i_1 = n + c + j \quad \forall \quad 0 \leq j \leq c \tag{16}$$

Increasing the amount of servers has a similar effect to increasing the queueing capacity, there are now more transient spaces 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 ($i = n + c + 1$ to $i = n + 2c$) can jump back to state $i = n + c - 1$ simply with a service and an exit. Increasing the amount of servers also increases the rate at which i are reduced for most states, but not the rates at which i is increased.

Figure 29 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.

5.4.2 Multi-Server: Two Node Network without Self Loops

Consider the two node network with feedback loops discussed in Subsection 5.2, now with c_1 parallel servers at the first node, and c_2 parallel servers at the second node.

State space:

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

where i denotes the number of individuals at the 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, $(i = n_1 + c_1 + 2, j = n_2 + c_2 + 1)$ 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 $(i = n_1 + c_1 + c_2, j = n_2 + c_2 + c_1)$ denotes the deadlocked state.

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 $s_2 = \min(i_2, c_2) - b_1$,then the transitions $q_{(i_1,j_1),(i_2,j_2)}$ are given by the following table:

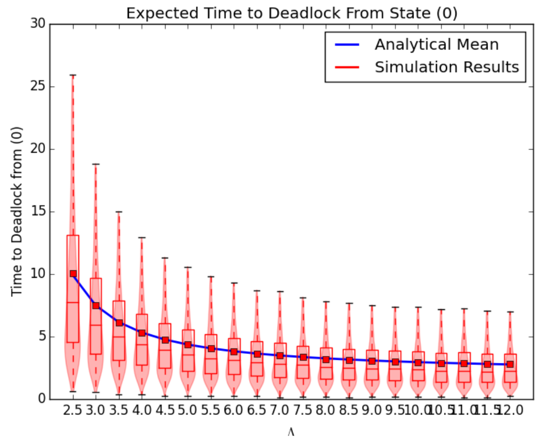
	$j_1 < n_2 + c_2$	$j_1 = n_2 + c_2$	$j_1 > n_2 + c_2$
$i_1 < n_1 + c_1$	Λ_1 if $\delta = (1, 0)$ Λ_2 if $\delta = (0, 1)$ $r_{12}s_1\mu_1$ if $\delta = (-1, 1)$ $r_{21}s_2\mu_2$ if $\delta = (1, -1)$ $(1 - r_{12})s_1\mu_1$ if $\delta = (-1, 0)$ $(1 - r_{21})s_2\mu_2$ if $\delta = (0, -1)$	Λ_1 if $\delta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

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.

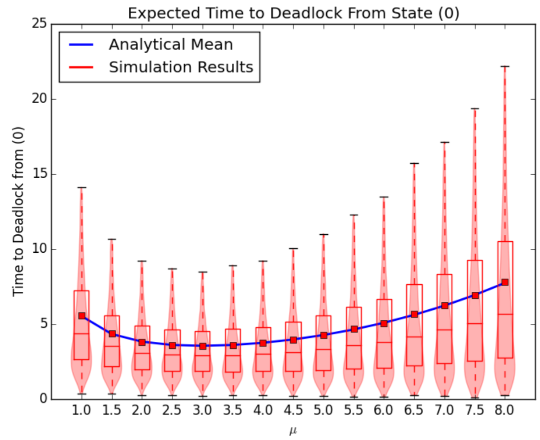
This formulation of the Markov chain makes use of the following proposition:

Proposition 1. *In the two node queueing network described, if there are b_1 customers blocked to Node 1, and b_2 customers blocked to Node 2, where $b_1, b_2 > 0$, then:*

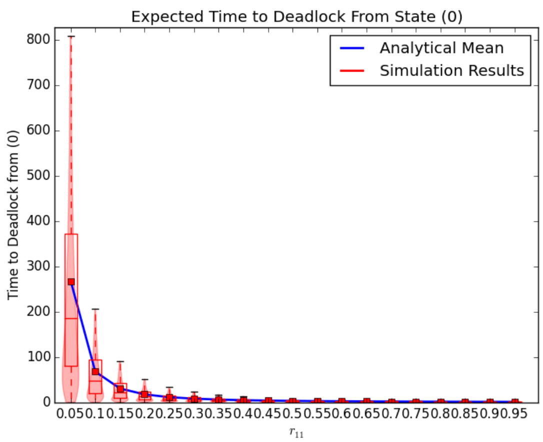
i. *A customer finishing service and leaving the system at Node 1 yields a difference in state of $\delta = (-\min(b_1 + 1, b_2 + 1), -\min(b_1, b_2 + 1))$.*



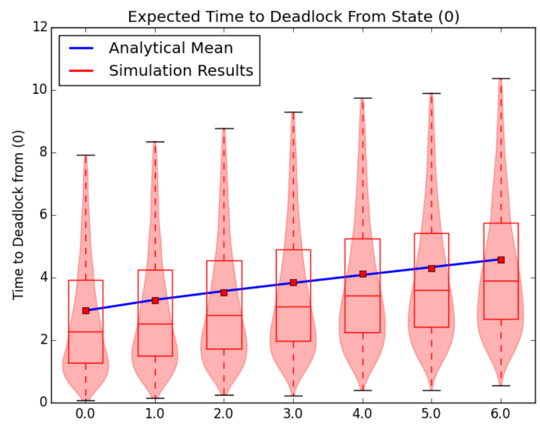
(a) Varying Λ



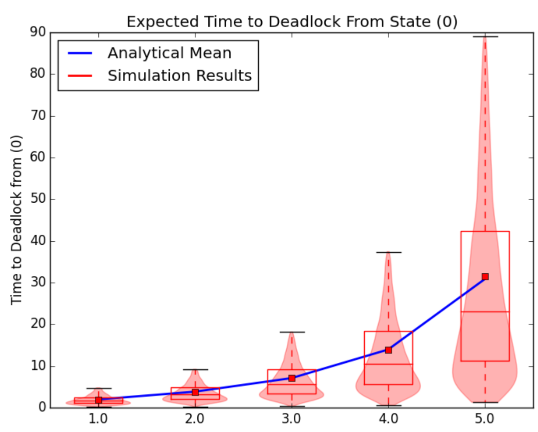
(b) Varying μ



(c) Varying r_{11}



(d) Varying n



(e) Varying c

Figure 29: The effect of varying parameters on times to deadlock.

ii. A customer finishing service and leaving the system at Node 2 yields a difference in state of $\delta = (-\min(b_1+1, b_2), -\min(b_1+1, b_2+1))$.

Proof. Each blocked individual is counted in the state space twice, once as an i and once as a j. A blocked customer is counted as existing in the Node he is occupying, but also has a blocked part counted at the other node.

Now an unblocking where a customer transitions from being blocked at Node 1 to being at Node 2 yields $\delta = (-1, 0)$; his blocked part at j disappears, while his existence part at i transitions to Node 2 to become a j. This type of unblocking leaves space at Node 1 for more potential unblockings.

Similarly an unblocking where a customer transitions from being blocked at Node 2 to being at Node 1 yields $\delta = (0, -1)$; his blocked part at i disappears, while his existence part at j transitions to Node 2 to become a i. This type of unblocking leaves space at Node 2 for more potential unblockings.

Consider a customer finishing service and exiting the system at Node 1. Breaking down the overall unblocking process into subprocesses, we can break down the overall difference δ into sub-differences δ_1, δ_2, δ_3, etc, where $\delta = \sum_i \delta_i$. These subprocesses are:

- Exit at 1: $\delta_1 = (-1, 0)$
- Unblock customer from Node 2: $\delta_2 = (0, -1)$ if there is someone at Node 2 to unblock, else stop.
- Unblock customer from Node 1: $\delta_3 = (-1, 0)$ if there is someone at Node 1 to unblock, else stop.
- Repeat the last two steps until stopping criteria is reached.

Now:

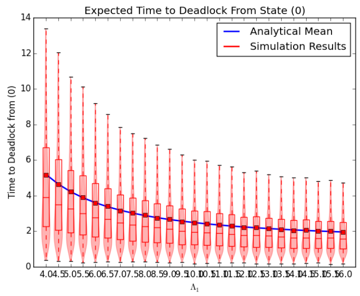
$$\begin{aligned}
\delta &= \sum_i \delta_i \\
&= (-1,0) + (0,-1) + (-1,0) + (0,-1) + \dots \\
&= (-1,0) + \min(b_1, b_2+1)(0,-1) + \min(b_2, b_1)(-1,0) \\
&= (-\min(b_1+1, b_2+1), -\min(b_1, b_2+1))
\end{aligned}$$

A similar argument yields $\delta = (-\min(b_1+1, b_2), -\min(b_1+1, b_2+1))$ for the situation where a customer finishes service at Node 2 and exits the system. □

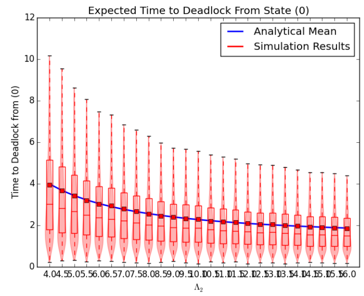
Figure 30 shows the effect of varying the parameters of the above Markov model. Base parameters of $\Lambda_1 = 9$, $\Lambda_2 = 7.5$, $n_1 = 2$, $n_2 = 1$, $\mu_1 = 5.5$, $\mu_2 = 6.5$, $r_{12} = 0.7$, $r_{21} = 0.6$, $c_1 = 2$ and $c_2 = 2$ were used.

5.5 A Bound on the Mean Time to Deadlock

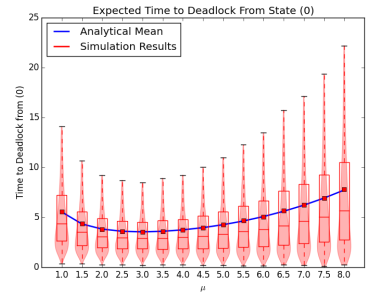
In this section we shall define six deadlocking queueing networks as follows:



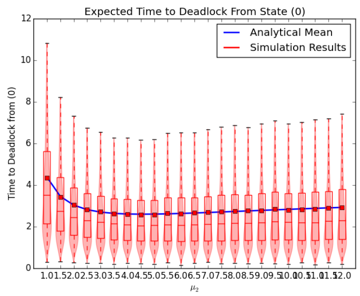
(a) Varying Λ_1



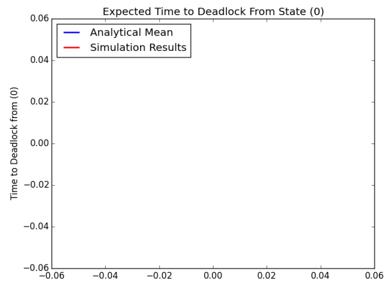
(b) Varying $\Lambda_@$



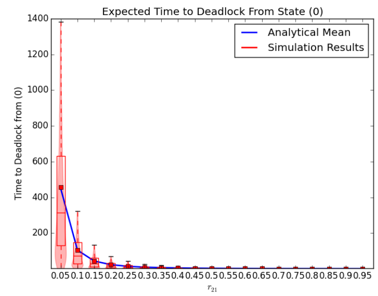
(c) Varying μ_1



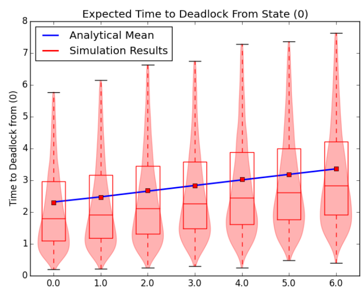
(d) Varying μ_2



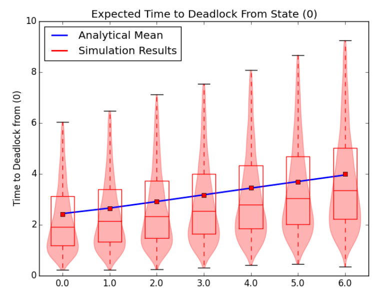
(e) Varying r_{12}



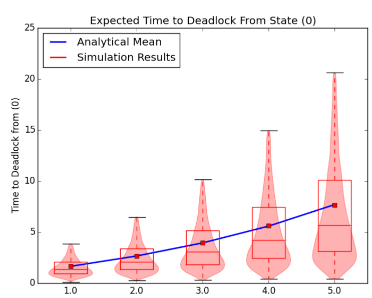
(f) Varying r_{21}



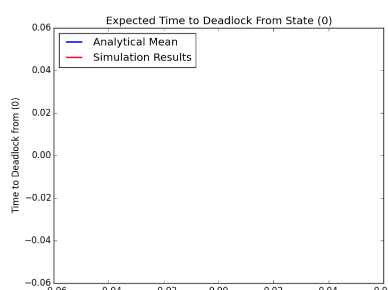
(g) Varying n_1



(h) Varying n_2



(i) Varying c_1



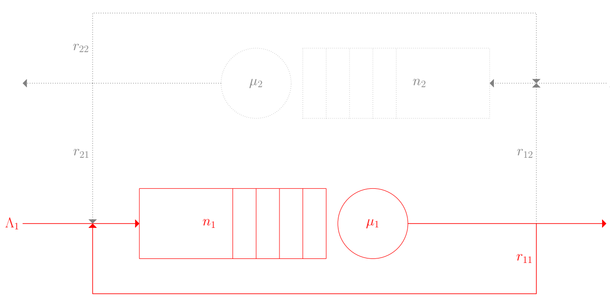
(j) Varying c_2

Figure 30: The effect of varying parameters on times to deadlock.

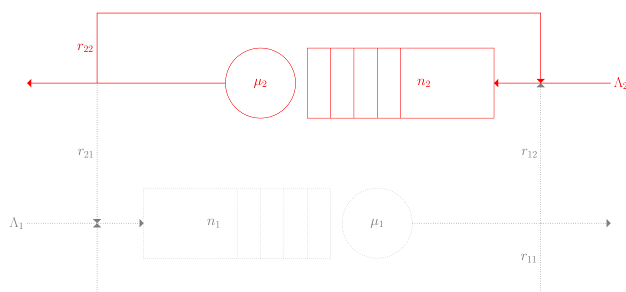
- Define $\Omega_{1_1}^{\star}$ as the 1 node queueing network described in Subsection 5.1 with the parameter set $\{\Lambda_1, \mu_1, n_1, r_{11}\}$. Let its mean time to deadlock be denoted by $\omega_{1_1}^{\star}$.
- Define $\Omega_{1_1}^{\star\star}$ as the 1 node queueing network described in Subsection 5.1 with the parameter set $\{\Lambda_1, m_1, n_1, r_{11}\}$. Let its mean time to deadlock be denoted by $\omega_{1_1}^{\star\star}$.
- Define $\Omega_{1_2}^{\star}$ as the 1 node queueing network described in Subsection 5.1 with the parameter set $\{\Lambda_2, \mu_2, n_2, r_{22}\}$. Let its mean time to deadlock be denoted by $\omega_{1_2}^{\star}$.
- Define $\Omega_{1_2}^{\star\star}$ as the 1 node queueing network described in Subsection 5.1 with the parameter set $\{\Lambda_2, m_2, n_2, r_{22}\}$. Let its mean time to deadlock be denoted by $\omega_{1_2}^{\star\star}$.
- Define Ω_2 as the 2 node queueing network described in Subsection 5.2 with the parameter set $\{\Lambda_1, \Lambda_2, \mu_1, \mu_2, n_1, n_2, r_{12}, r_{21}\}$. Let its mean time to deadlock be denoted by ω_2.
- Define Ω as the 2 node queueing network described in Subsection 5.3 with the parameter set $\{\Lambda_1, \Lambda_2, \mu_1, \mu_2, n_1, n_2, r_{11}, r_{12}, r_{21}, r_{22}\}$. Let its mean time to deadlock be denoted by ω.

where $m_1 = \frac{\mu_2}{3+2\frac{\mu_2}{\mu_1}+\frac{\mu_1}{\mu_2}}$, and $m_2 = \frac{\mu_1}{3+2\frac{\mu_1}{\mu_2}+\frac{\mu_2}{\mu_1}}$.

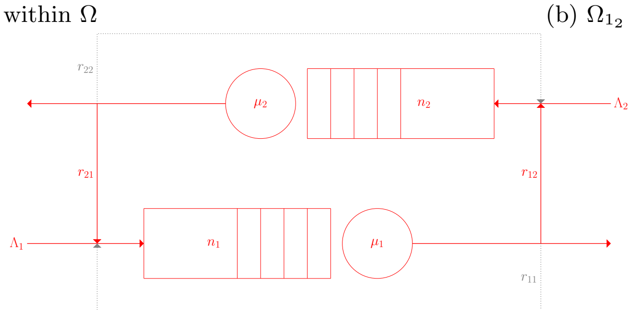
Figure 31 shows how Ω contains, and is made up by, Ω_{1_1}, Ω_{1_2} and Ω_2.



(a) Ω_{1_1} within Ω



(b) Ω_{1_2} within Ω



(c) Ω_2 within Ω

Figure 31: Decomposition of Ω into Ω_{1_1}, Ω_{1_2} and Ω_2

Defining $\omega_{1_1} = \max(\omega_{1_1}^{\star}, \omega_{1_1}^{\star\star})$ and $\omega_{1_2} = \max(\omega_{1_2}^{\star}, \omega_{1_2}^{\star\star})$ we get the following bound:

Theorem 2. *For any parameter sets the following inequality holds:* $\omega \leq \min(\omega_{1_1}, \omega_{1_2}, \omega_2)$

Proof. First, define some systems:

- Let $\widetilde{\Omega}_{1_1}$ denote the Ω_{1_1} system embedded within Ω. Let $\widetilde{\omega}_{1_1}$ denote the mean time to deadlock of $\widetilde{\Omega}_{1_1}$.

- Let $\widetilde{\Omega}_{1_2}$ denote the Ω_{1_2} system embedded within Ω. Let $\widetilde{\omega}_{1_2}$ denote the mean time to deadlock of $\widetilde{\Omega}_{1_2}$.
- Let $\widetilde{\Omega}_2$ denote the Ω_2 system embedded within Ω. Let $\widetilde{\omega}_2$ denote the mean time to deadlock of $\widetilde{\Omega}_2$.

Now $\widetilde{\omega}_{1_1}$ is the mean time to state (-1) in Ω, $\widetilde{\omega}_{1_2}$ is the mean time to state (-2) in Ω and $\widetilde{\omega}_2$ is the mean time to state (-3) in Ω. Therefore $\omega = \min(\widetilde{\omega}_{1_1}, \widetilde{\omega}_{1_2}, \widetilde{\omega}_2)$, as the mean time to deadlock in Ω is the expected time it takes to reach either (-1), (-2) or (-3), whichever comes first.

Comparing $\widetilde{\Omega}_2$ to Ω_2: The effective arrival rate to Node 1 in $\widetilde{\Omega}_2$ is greater than or equal to the effective arrival rate to Node 1 in Ω_2. This is due to the extra customers who are rejoining the queue after service. Similarly the effective arrival rate to Node 2 in $\widetilde{\Omega}_2$ is greater than or equal to the effective arrival rate to Node 2 in Ω_2. As an increase in the arrival rate causes the mean time to deadlock to decrease, we can conclude $\widetilde{\omega}_2 \leq \omega_2$

Consider $\widetilde{\Omega}_{1_1}$. Consider the expected effective service time, the time that $\widetilde{\Omega}_{1_1}$'s state does not change due to services or outside factors.

The shortest expected effective service time is equal to $\frac{1}{\mu_1}$, corresponding to when neither Node 1 nor Node 2 are full. Therefore the largest effective service rate is μ_1.

The longest service time corresponds to when Node 2 is full, Node 1 finishes service and gets blocked. Here the customer in service at Node 1 spends $\frac{1}{\mu_1}$ in service, then gets blocked. That customer remains blocked until there is room at Node 2, that is when Node 2 has a service. As all service rates are Markovian, the time spent blocked is $\frac{1}{\mu_2}$.

If the customer finishing service at Node 2 transitions to Node 1, then $\widetilde{\Omega}_{1_1}$'s state hasn't changed, and Node 2 is full again. The next customer at Node 1 now spends $\frac{1}{\mu_1}$ in service. This process will repeat if Node 1 is blocked before Node 2 releases a customer, and so the process is repeated with probability P_{repeat}.

And so the longest effective service time is:

$$
\begin{aligned}
&= \frac{2}{\mu_1} + \frac{1}{\mu_2} + P_{\text{repeat}} \left(\frac{2}{\mu_1} + \frac{1}{\mu_2} + P_{\text{repeat}} \left(\frac{2}{\mu_1} + \frac{1}{\mu_2} + P_{\text{repeat}} \left(\cdots \right.\right.\right. \\
&= \left(\frac{2}{\mu_1} + \frac{1}{\mu_2} \right) \times \left(1 + P_{\text{repeat}} + P_{\text{repeat}}^2 + P_{\text{repeat}}^3 + \dots \right) \\
&= \left(\frac{2}{\mu_1} + \frac{1}{\mu_2} \right) \times \left(\frac{1}{1 - P_{\text{repeat}}} \right)
\end{aligned}
$$

If S_1 is the time the customer at Node 1 spends in service, and S_2 is the time the customer at node 2 spends in service, then $S_1 \sim \text{Exp}(\mu_1)$ and $S_2 \sim \text{Exp}(\mu_2)$. Now $P_{\text{repeat}} = P(S_1 < S_2) = \frac{\mu_1}{\mu_1 + \mu_2}$.

Therefore the smallest effective service rate is:

$$
\begin{aligned}
&= \frac{1}{\left(\frac{2}{\mu_1}+\frac{1}{\mu_2}\right)\left(\frac{1}{1-P_{\text{repeat}}}\right)} \\
&= \frac{1}{\left(\frac{2}{\mu_1}+\frac{1}{\mu_2}\right)\left(\frac{1}{1-\frac{\mu_1}{\mu_1+\mu_2}}\right)} \\
&= \frac{1}{\left(\frac{2}{\mu_1}+\frac{1}{\mu_2}\right)\left(\frac{\mu_1+\mu_2}{\mu_2}\right)} \\
&= \frac{\mu_2}{\left(\frac{2}{\mu_1}+\frac{1}{\mu_2}\right)(\mu_1+\mu_2)} \\
&= \frac{\mu_2}{3+2\frac{\mu_2}{\mu_1}+\frac{\mu_1}{\mu_2}} \\
&= m_1
\end{aligned}
$$

As time to deadlock does not increase monotonically with service rate (see Figure 17c) we cannot simply compare $\widetilde{\Omega}_{1_1}$ with the system that has the largest of smallest service rate; instead we must compare $\widetilde{\Omega}_{1_1}$ with either the system with the smallest service rate or the largest, whichever yields the longest time to deadlock.

The largest service rate is μ_1, corresponding to system $\Omega^{\star}_{1_1}$ and the smallest is m corresponding to $\Omega^{\star\star}_{1_1}$. And so we compare with whichever system takes longer to reach deadlock, we compare with $\omega_{1_1} = \max(\omega^{\star}_{1_1}, \omega^{\star\star}_{1_1})$.

And so comparing this to $\widetilde{\Omega}_{1_1}$: the effective arrival rate in $\widetilde{\Omega}_{1_1}$ is greater than or equal to the effective arrival rate in that system; this is due to the extra customers who have transitioned from the Node 2 to Node 1. An increase in the arrival rate causes the mean time to deadlock to decrease.

We can conclude that $\widetilde{\omega}_{1_1} \leq \omega_{1_1}$ at $\widetilde{\omega}_{1_1}$'s worst case scenrio, and so $\widetilde{\omega}_{1_1} \leq \omega_{1_1}$ in general.

Similarly we can conclude $\widetilde{\omega}_{1_2} \leq \omega_{1_2}$.

And so

$$
\begin{aligned}
\min(\widetilde{\omega}_{1_1}, \widetilde{\omega}_{1_2}, \widetilde{\omega}_2) &\leq \min(\omega_{1_1}, \omega_{1_2}, \omega_2) \\
\omega &\leq \min(\omega_{1_1}, \omega_{1_2}, \omega_2)
\end{aligned}
$$

□

References

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

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

[3] 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.

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

[5] Edsger W Dijkstra. The mathematics behind the bankers algorithm. In *Selected Writings on Computing: A Personal Perspective*, pages 308–312. Springer, 1982.

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

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

[8] 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.

[9] 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.

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

[11] 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.

[12] 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.

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

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