# Network flow Optimization through Monte Carlo Simulation

Sayaji Hande
*PhD, Freelance Analyst*

Prasoon Patidar
*B. Tech, IIT Delhi*

Sachin Meena
*B. Tech, IIT Delhi*

Saurabh Banerjee
*Newgen Software Technology Ltd.*

***Abstract*—We encounter network flows in day to day life. They are the backbone of logistics, city planning, processes etc. In order to study these networks, domain specific connectivity graphs along with their historical observations are used. Traditionally, birth & death process, Little's law, Burke's theorem, etc. have been applied to analyze various network flow scenarios. In this paper, we approach similar problems using Monte Carlo simulations, Markov Chain and Queuing Theory, which provide an edge over traditional methods in case of high dimensionality of multiple nodes. The methodologies described in paper can be applied to various situations of business formulations: Load/traffic balancing, Queue reduction, Network Anomaly detection, etc. This paper provides an effective tool for designing, diagnosing, monitoring & predictions of process of networks. Networks flows are the backbone of logistics, city planning, processes, etc.**

***Index Terms*—Networks, Monte Carlo, Birth & Death, SLAs, Queuing Theory, Little's Law, Markov Process, Transition Matrix.**

## I. Introduction

Scenarios involve a set of processes, jumping from one node to another. Some industrial situations may have deterministic paths while others probabilistic. Arrival of a 'work items' and their service at various nodes is common phenomenon in many scenarios. Designing, assessing scenarios, etc. are of paramount importance. Sample business examples are visitors (malls, cities), bank counters, production lineups, network security, road traffic (site) management, load balancing, etc.

While in some industry, reduction of customer *wait time* along with *costs* is the prime objective, in other situations, increasing customer *life span* within the network is important.

As indicated earlier, solving this 'algebraically' is computationally very expensive as it is an NP-hard problem due to high dimensionality. This paper proposes usage of Monte Carlo simulations [1] to make computational complexity independent of dimensions. The paper integrates Graph Theory [2], Markov Theory [3] & Queuing Theory [4] to effectively provide an ability to design, optimize and review scenarios.

The primary *input* parameters of network flows are arrival rates, service times at nodes and feasible paths. **Rate:** Inflow (arrival) rate or its distribution; **Service Time:** Service rate at each station (node). This in turn depends on 'resources' loaded, its productivity & costs; **Graph Connectivity:** Activities/Nodes connectivities, may be a probabilistic transition matrix.

To arrive at the optimized tangible parameters, i.e. *waiting time*, *transition population size*, *resource utilization* etc. Markov Chain Monte Carlo Simulation [5] has been laid out. It effectively combines theories described above and utilizes them for a multi-node system. The parameters described above can be changed to analyze any scenario.

Following is the layout of the paper with respect to the sections: Effective Resource Loading at node level (Little Law [6]); Convergence rate & effective number of simulations; Simulation Algorithm; Empirical Studies & Demo. A few examples of above mentioned businesses follow.

- Delivery/Customer Services
- Traffic Management
- Anomaly Detection & Alerts

## II. Model Formulation

A network flow is parameterized by: arrival rate ($\lambda$), service rates at different nodes ($\mu_i$'s), and transition matrix ($T_m$).
*Without loss of generality & for sake of simplicity,*

*assume the network is deterministic (fixed no. of nodes, say $k$), let $\lambda$ be given arrival rate per time interval, let $T$ be the transition matrix, indicating probabilities of transition between nodes, let $r_1$, $r_2$, . . . , $r_k$ be the $k$ resources with respective productivity $p_1$,$p_2$, . . . , $p_k$. Here $p_i$[1] is time taken by $i^{th}$ resource at $i^{th}$ node to process the work item.*

Using the above parameters, the system proposed in this paper provides distribution of *waiting time (W), sojourn time (S), resource utilization & transition load*

**Assumptions** in this paper: *Service Time* distribution is provided by user (eg: Exponential, with parameter $\mu$); *Service type:* is taken to be first come first serve; also the network is deterministic, i.e. the no. of nodes and productivities remain fixed throughout the simulation.

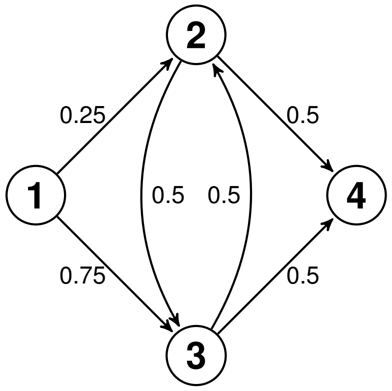
### A. Proposed Solution: Theoretical foundation

The subsections below elaborate various related theories and their integration to achieve the objectives described.

*1) Node Level Load Estimation:* A single node represents a work station for any network flow. Let us consider the different counters in a restaurant - bill counter, water dispenser etc. Essentially, each node in graph represents a M/M/c queue. Per unit arrival to the system, realized arrival at a particular node can be fraction of it or more based on transition matrix, $T$. For example, billing counter has same arrivals as in the system, water counter may have more actual arrivals and coffee counter may have a fraction. These actual arrivals are needed to be derived.

This paper provides you with a method to find arrival rates at every node. Estimation of workload will help us to optimise our resources, get a cost estimate for effective system. Also, It is possible to deploy different kinds of resources at different nodes, Thus gives us insight on most critical resources in our system. Moreover,at individual level, it helps us find anomalous behaviour in real life system monitoring. Following is transition graph that are being used to demonstrate the intuition.

[1]For sake of simplicity, it is assumed that $p_i$ are continuous variables, whereas in real-life scenarios, $p_i$ would be integers.



Now, the respective transition matrices of above figures, excluding "*exit / death*" node are following.

$$\begin{array}{c|ccc} & 1 & 2 & 3 \\ \hline 1 & 0 & 0.25 & 0.75 \\ 2 & 0 & 0 & 0.5 \\ 3 & 0 & 0.5 & 0 \end{array}$$

For this flow, the possible load per unit of an arrival at node 2 is $a_{12}$ and at $node3$ is $a_{13}$ where $A = (I-T)^{-1}$. Expected arrival at a node is same as sum total of arrivals in each transition of the element in the network.

$$\begin{bmatrix} 1 & 0.83 & 1.167 \\ 0 & 1.33 & 0.67 \\ 0 & 0.67 & 1.33 \end{bmatrix}$$

Hence in general, if $T$ is the transition matrix of a network, $T_{ij}$ provides probability of transitioning from $i$ to $j$ in 1 step. Similarly element of $T^2$, say $T^2_{ij}$, provides similar probabilities in 2 steps and so forth. This can be extended to infinite number of steps.

Hence, the effective arrivals to $j^{th}$ node from a birth node $i$, is $a_j = \sum_k T^k_{ij}$. The load on $j^{th}$ node can be derived as sum of similar entities for respective birth nodes. Keeping this concept in mind following has been derived.

Let $A = (I - T)^{-1}$, and $\lambda_i$ be arrival at node $i$, then total or effective load or arrivals on node $j$, say $L_j$ would be

$$L_j = \sum_i \lambda_i a_{ij}$$

Hence, if $L = (L_1, L_2, \dots)$ and $\lambda = (\lambda_1, \lambda_2, \dots)$ then, it is very clear that

$$\begin{aligned} L &= \lambda A \\ &= \lambda(I - T)^{-1} \end{aligned}$$

Above discussion might be summarized into the following theorems.

**Theorem II.1.** *The node level load, represented by vector $L$ is provided by arrival vector $\lambda$ as*

$$L = \lambda(I-T)^{-1}$$

*where $T$ is transition matrix from where the death/exit node is removed.*

The above theorem provides a starting point towards minimal resources required that would prevent blowing up of queues. Of course, arrival vector $\lambda$ and transition matrix $T$ is assumed to be known. A similar version of this theorem has been discussed in *Jacksonian Networks* [7].

**When does the system blow?**

It is known from earlier research on queuing models that a node with arrival rate $\lambda$, processing rate $\mu$ and allotted resource $r$ will blow up if the ratio $\frac{\lambda}{r*\mu} > 1$. This subsection extends this idea to a multi-node system as a theorem.

**Theorem II.2.** *Any markovian system does not blow up iff for all intermediate nodes, the ratio of arrival rate $\lambda$ to service rate $\mu$ is less than 1.*

**Proof:** Following is a proof by induction on the number of nodes in the system.

- Base case: if k = 1, the above argument holds and the theorem trivially holds
- **IH**: let the system hold for k nodes ($k \leq n$)
- **IS**: for the (k+1) node system, pick any subsystem of k nodes. By IH, this subsystem has for each node i, $\frac{\lambda_i}{r_i*\mu_i}$ and hence doesn't blow up. As long as the $(k+1)^{th}$ node satisfies $\frac{\lambda_{k+1}}{r_{k+1}*\mu_{k+1}} < 1$, it node does not blow, and hence the (k+1) node system does not blow.

Customers who enter the process flow often have different key characteristics. For example age, income, marital status, etc. are features that are considered for approval of a credit card. Given these features, the customers can be classified into multiple types where each customer type follows the corresponding transitions. However, in practical scenarios the number of customer groups formed can be huge which can lead to further complexities. The lemma mentioned below addresses this problem combining the transition matrices of different customer types into single matrix.

**Lemma II.1.** *A markovian model $M$ with finite customer types can be simulated with customer type dependent transition matrices using 1-step markovian model.*

**Proof:** Let $C_1, C_2, C_3...C_n$ be the n types of customers, let $T_1, T_2, T_3...T_n$ be corresponding transition matrices, and let $M_1, M_2, M_3...M_n$ be the process flow models corresponding to these transition matrices. Each model $M_i$ is a markovian model. Let $p_i$ be the probability of arrival of customer $C_i$ in given model $M$. The following claim & its proof complete our theorem.

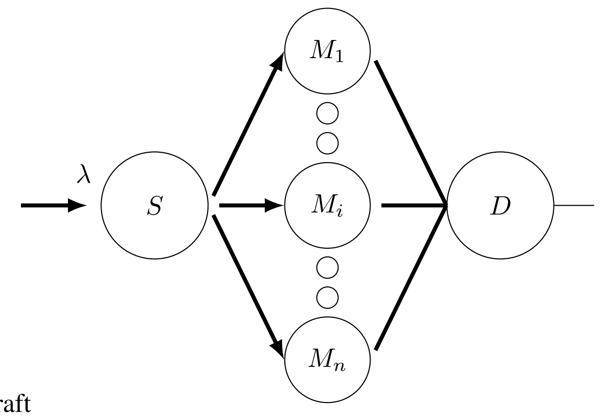


Figure: Constructed Model

**Claim:** The above constructed model with $S$ & $D$ as birth and death nodes, is a markovian system.
**Proof:** As each $S_i$ is a markovian system, the above model can be converted to a markovian system.

Following is another theorem towards Little's Law applicability at the node level.

**Theorem II.3.** *Let $\lambda_k$ be the arrival rate at node k, $W_k$ be the average waiting time at node k, then average length of the queue at node k ($L_k$) using Little's law is*

$$L_k = \lambda_k * W_k$$

*2) Multi-Skilled Resource Loading & Optimization:* Organizational resource acquisition is often multi-skilled. These multi-skilled resources may have different salary bands. Having multi-skilled resources is useful as one can have flexibility to change between different tasks (nodes).

The discussions in subsection 3.1 provide minimum resource requirement per node. Let us consider a process flow of n nodes. The load on $i^{th}$ node is $L_i$, let $\mu_i$ is mean service rate at node $i$ and $R = (r_1, r_2, \ldots, r_k)$ be a column vector of $k$ resource groups, where $r_j$ is resource count at $j^{th}$ group . Let $S$ be a $k \times n$ resource allocation matrix where $s_{ji}$ denotes the allocation ratio of $r_j$ at node $i$. Therefore,

$$0 \leq s_{ji} \leq 1$$

and for each group $r_i$,

$$\sum_{i=1}^{n} s_{ji} = 1$$

Using the condition for node equilibrium in Theorem 2,

$$\sum_{j=1}^{k} s_{ji} r_j \geq \rho_i, \text{ where } \rho_i = \frac{\lambda_i}{\mu_i}$$

In order to reduce the resource cost optimal resource loading is the key aspect. Assuming $C = (c_1, c_2, \ldots, c_k)$ represent the cost of each resource in the group. Therefore, the total resource cost for the process flow becomes

$$\psi = \sum_{j=1}^{k} c_j r_j$$

$S^T R \geq \rho$ and $R \geq 0$, gives a set of equations. The optimum resource loading at each node is obtained by solving these equations using *Linear Programming* [8] to minimize cost ($\psi$).

### *B. Proposed solution: Through Simulation*

In practical process flows, the number nodes and their interconnectivity can be huge. Moreover, the arrival rate and service rate may follow non-exponential distributions. The resource availability criteria and seasonality in the arrival and the service pattern increases the complexity of a process flow. Given the theoretical foundations described in Section 2.1, arriving to the optimum solution algebraically, if not impossible, is definitely highly complex and computationally expensive. The distributions of key parameters of interest i.e. waiting time, sojourn time, queue length,etc. which may not be obtained theoretically for every scenario, are needed to be derived to extract deeper insights of the process flow.

Considering the above problems in using the algebraic equations directly for extracting vital information from the process flow, the Monte Carlo simulation methodology provides effective solution. Convergence of outcomes of *what-if* scenarios run for a given process is linear and highly efficient using Monte Carlo simulations.

The process flow is simulated with customers generated as per arrival distribution and randomly generated service time for every node the customers visit according to the service rate distribution. In general, the customer arrivals follow *Poisson Process* and service rate follows *Exponential Distribution*. Since the transitions are probabilistic in nature, Gibbs Sampling technique [9] through a uniform random generator(URG) is used to decide which node should a customer move to next.

Moreover, the simulation is discrete event-based [10], instead of continuous time-based. Following are the events that cause the system state to change:

1) Arrival of a customer.
2) Any node finishes servicing its customer.

As stated earlier, our simulation methodology can be used for **designing, diagnosing, monitoring and prediction** of process flow networks.

- **Design**: Users get to provide all the specifications of the system, ranging from the number of nodes to parameter values. The results of simulations then help the user in redesigning the process flow.
- **Diagnose**: A model diagnosis process is a powerful consciousness raising activity in its own right, its main usefulness lies in the action that it induces. Diagnosis might suggest the possible problems in a model, and why they were not identified during monitoring. Users may run various simulations - keeping their design fixed - in search for anomalies.
- **Monitor**: Given a system of n-nodes, the simulator can predict the loads at individual nodes and monitor traffic at micro level(at each node). This may help an operations manager to identify critical nodes, either in terms of waiting time explosion or cost of resources deployed. Following earlier work on selecting top seeds to have maximum 'combined' value [11], one can select certain top nodes in a model that would be most probable causes for a glitch in the system design. Thus, (s)he would be able to monitor most critical k-nodes where k depends on the convenience of manager.
- **Prediction**: Once the design is specified, the

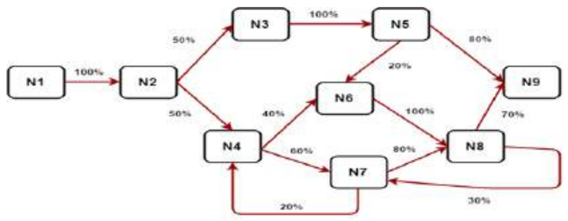


Fig. 1. State Diagram of our model

engine runs accordingly to predict plots of quantities of interest, like the waiting time of customers, utilization of the resources, the busyness of this system, etc. The simulator can also predict the time a customer has to wait in the queue or additional time that a customer's service will take among other things.

## III. Empirical Study & Results

### A. Experimentation Summary

In order to capture modeling for the process flows using Monte-Carlo simulations, we did an empirical study on essentially three different contention levels in system. Following characteristics of a system will be studied to draw inference about system.

- Waiting/Processing /Sojourn Time Distributions - These are frequency plots of no. of customers vs time, i.e waiting time distribution suggests how many customers had to wait how long to enter the system.
- Resource Utilization - This tells us about usability of resources by showing for what fraction of time, a particular resource(or Node) was active in simulation.
- Customer Count plots - This tell us for what fraction of time, how many customers were waiting/inside the system.

The transition probabilities can be fetched from figure 1.

### B. Observations

Figure 2 & 3 show a low contention system. As expected, 2(a) shows that the average waiting time for most customers stays around 0 units, but as $\lambda$ increases, number of such customers decreases. 2(b) tells that processing time per customer increases as $\lambda$ increases. Moving on to figure 2(d), we see that utilization of every resource increases with the number of people in the system, i.e., with increase in $\lambda$. We also observe that node 0 (N1) and node 7 (N8) are the busiest one. Since node 7 is the only node where a backward transition is possible (apart from node 6, which is the third most busiest node), it seems that customers are keeping it busy by bouncing back to it multiple times. Figures 2(e) and 2(f) tell us with some exceptions that number of people in a system and people waiting to enter at any point of time the system increases as we increase $\lambda$.

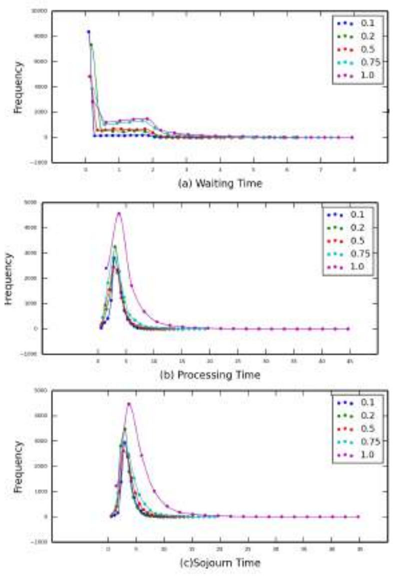


Fig. 2. Low Contention System(Distributions)

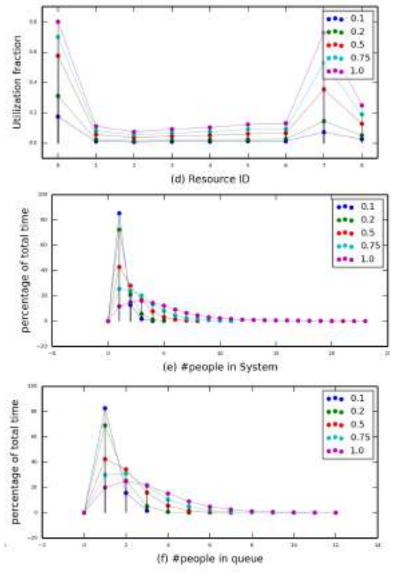


Fig. 3. Low Contention System(Resource Utilization)

Figure 4 & 5 is a normal contention system. Note that we have not provided results for a $\lambda = 1$ as the system explodes for this ($\lambda$,$\mu$) combination. 3(a)-(f) have explanations similar to their figure 2 counterparts. Figure 6 & 7 has high contention in the queues. Note that we have not provided results even for $\lambda = 0.75$ as the system explodes for this ($\lambda$,$\mu$) combination, let alone a $\lambda$ value of 1. Explanation of all figures remains in line with previous figures.

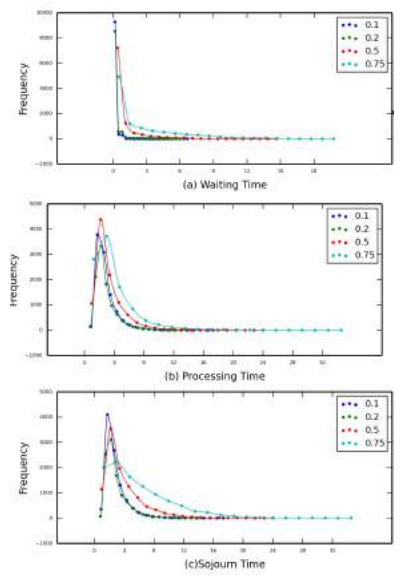


Fig. 4. Normal Contention System(Distributions)

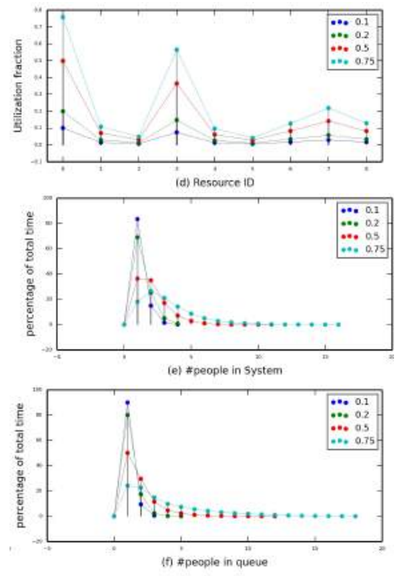


Fig. 5. Normal Contention System(Resource Utilization)

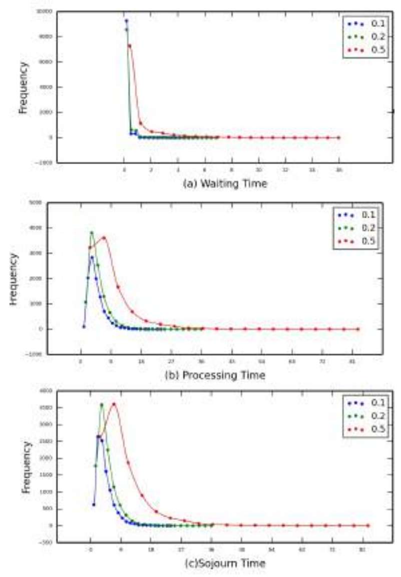


Fig. 6. High Contention System(Distributions)

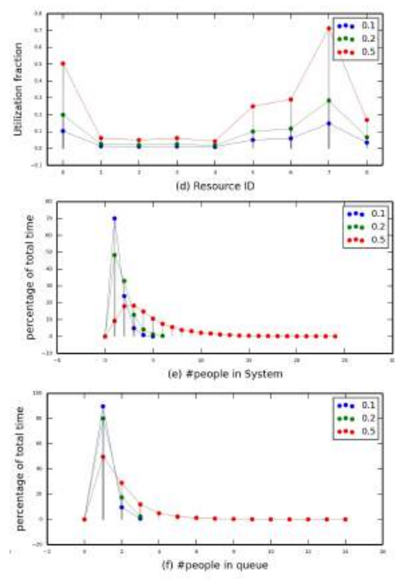


Fig. 7. High Contention System(Resource Utilization)

One interesting observation is that utilization of nodes follows a pattern similar to that of figure 2, and not figure 3. From these observations and our experimentation, it may be concluded that resource utilization, except for a few cases like in figure 3, usually remains consistent irrespective of the $\mu$ vectors that we use. Another important observation that we make is that the waiting time distribution remains similar between figures 3 and 4, which suggests that it only depends on the $\mu$ value for node 0.

## C. Validation

*1) Condition for Queue Blowing:* We can relate that the queue which is overflowing doesn't satisfy per node condition for arrival rate at that node and its service rate.

*2) Measurement of closeness:* Let us take a very loose bound on error compared to actual distribution considering the observations which are binary in nature(less than/more than some value). Thus, deviation (variance) from original value is maximum in binary observation. Let $w_i'$ is a binary valued random variable corresponding to observation of waiting time for $i^{th}$ customer. Let W corresponds to random variable for summed and down scaled waiting time.

$$W = \frac{1}{N} * \sum_{i=1}^{n} [w_i']$$

Now, given $w_i's$, if W corresponds to a binomial distribution, and thus, deviation from actual value is

$$Var(W) = pqN/N^2$$

Thus standard deviation of W is

$$sd(W) = \sqrt{pq/N} \leq 1/2 * \sqrt{1/N}$$

as standard deviation is maximum if p=q=1/2. Thus, this gives us an idea on how to generate an apparent distribution with bounded standard deviation. For ex., if one wishes to generate distribution which is less than 1% deviated from actual distribution, i.e

$$1/2 * \sqrt{1/N} \leq 0.01, N \geq 2500$$

then error in observed distribution is bounded to 1%.

## IV. Ongoing Research

Though our simulations are very close to real life systems, there may be cases when a service token (customer) is sent to multiple nodes by a split node, served at different nodes in parallel and then assembled in the joining node. After that the service token takes the succeeding path according to the transition probability. For example, a car manufacturing company gets orders for some cars. The engine parts and body parts are build simultaneously. The assembling of the cars is done in a particular station and after joining the cars go for painting, sale and so on. Although the load analysis of different nodes can be obtained by Theorem 1 with scaling vector at each node, complex simulation is needed for the waiting time and sojourn time analysis. Moreover, there can be nested split-join pairs in a very complex process flow.

## References

[1] S. Raychaudhuri, "Introduction to monte carlo simulation," in *2008 Winter Simulation Conference*, Dec 2008, pp. 91–100.

[2] N. Deo, *Graph theory with applications to engineering and computer science*. Courier Dover Publications, 2017.

[3] D. A. Levin and Y. Peres, *Markov chains and mixing times*. American Mathematical Soc., 2017, vol. 107.

[4] A. Komashie, A. Mousavi, P. J. Clarkson, and T. Young, "An integrated model of patient and staff satisfaction using queuing theory," *IEEE Journal of Translational Engineering in Health and Medicine*, vol. 3, pp. 1–10, 2015.

[5] R. Y. Rubinstein and D. P. Kroese, *Simulation and the Monte Carlo method*. John Wiley & Sons, 2016, vol. 10.

[6] R. Tolosana-Calasanz, J. Diaz-Montes, O. F. Rana, M. Parashar, E. Xydas, C. Marmaras, P. Papadopoulos, and L. Cipcigan, "Computational resource management for data-driven applications with deadline constraints," *Concurrency and Computation: Practice and Experience*, vol. 29, no. 8, p. e4018, 2017.

[7] U. N. Bhat, "Queueing networks," in *An Introduction to Queueing Theory*. Springer, 2015, pp. 159–176.

[8] G. Dantzig, *Linear programming and extensions*. Princeton university press, 2016.

[9] C. D. Sa, V. C.-H. Chen, and W. Wong, "Minibatch gibbs sampling on large graphical models," in *ICML*, 2018.

[10] J. S. Morgan, S. Howick, and V. Belton, "A toolkit of designs for mixing discrete event simulation and system dynamics," *European Journal of Operational Research*, vol. 257, no. 3, pp. 907 – 918, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221716306312

[11] S. Hande, R. Sinha, V. Gupta, and S. Z. George, "Seed group selection in a probabilistic network to increase content dissemination," Dec. 20 2016, uS Patent 9,524,527.