Optimal Server Allocation in General, Finite, Multi-Server Queueing Networks

J. MacGregor Smith* jmsmith@ecs.umass.edu
F. R. B. Cruz† fcruz@est.ufmg.br
T. van Woensel‡ T.v.Woensel@tm.tue.nl

May 6, 2009

Abstract — **Queueing networks with finite buffers, multiple servers, arbitrary acyclic, series-parallel topologies, and general service time distributions are considered in this paper. An approach to optimally allocate servers to series, merge, and split topologies and their combinations is demonstrated. The methodology builds upon two-moment approximations to the service time distribution embedded in the generalized expansion method for computing the performance measures in complex finite queueing networks and Powell's algorithm for optimally allocating servers to the network topology. Convexity of the objective function along with results from computational experiments are presented for showing the efficacy of the methodology.**

Keywords — Multi-server; finite buffer; queueing networks; optimal server allocation.

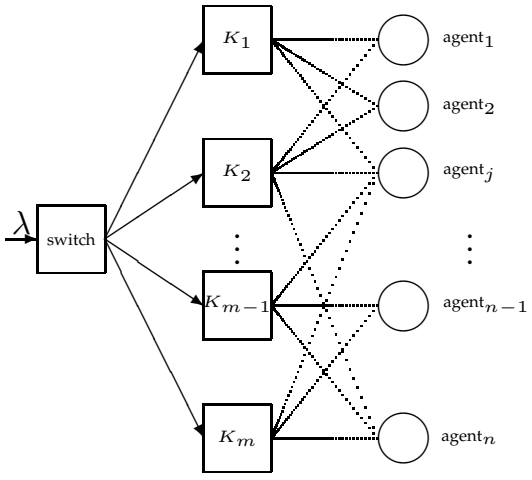


Figure 1: Call Center Topology

1 INTRODUCTION

The design of finite buffer queueing networks with multiple servers is a difficult, challenging problem. Determining the optimal number of servers within the network is the central focus of this paper. Not only is the problem extremely complex, it is critically important to many industries and service sector activities such as: manufacturing, retail, transportation, and telecommunications. Suppose we have the task of designing a new network for a process which involves multiple nodes in a complex acyclic, series-parallel topology much as in Figure 1. Recent important examples include the design of call centers that normally employ hundreds of servers (agents), where finite buffer K_i queues exist to hold the customers, and customer types can be handled by many different skilled servers. The dotted lines in the diagram indicate the necessary special skills available with the agents for handling the different customer types.

Not only must we deal with the pattern of arrival and service processes as affected by the topology, let's argue that one key decision variable is to determine the number of servers at each node so as to effectuate the overall throughput performance of the network. We don't want to arbitrarily assign the number of servers, otherwise the desired throughput and queueing performance measures of the network will not be realized. How should we tackle the server allocation problem? This is the key design issue of this paper. We need a sensible methodology and a set of tools to carry out this task.

Our paper is unique in that no one to our knowledge has contributed a paper on the optimal allocation of servers in general service time distribution, finite, and series-parallel topology networks.

1. We tackle here a complex problem of optimal server allocation in acyclic networks of general service-time finite queues with a unique and practical optimization approach;

2. A comprehensive literature review is carried out to better pose the optimal server allocation problem among correlated problems already published;

*Department of Mechanical and Industrial Engineering, University of Massachusetts Amherst, Massachusetts 01003, USA
†Department of Statistics, Federal University of Minas Gerais, Belo Horizonte, MG, Brazil
‡Department of Operations, Planning Accounting and Control, Eindhoven University of Technology, Eindhoven, The Netherlands

3. The convexity of the objective functions is examined in great detail consequently assessing the the suitability of the optimization algorithm proposed;

4. We explore thoroughly the relationship between the number of server and the number of buffers and other important parameters, such as the squared coefficient of variation of the service time, the service rate, and the arrival rate at the node.

Our previous papers only have focused on buffer allocation in general networks with single [33] and more recently multiple servers [34], however, the problems and the literature concerning these two problems are different.

1.1 Outline of Paper

§2 presents a review of the problem and its difficulty, known results, along with the various references appropriate to its analysis. §3 presents the mathematical models and convexity properties appropriate to the optimization process. §4 delivers the performance and optimization algorithms, while §5 discusses the experimental design and results for series, merge, and split network topologies. One of the differences and contributions of our paper compared to previous ones on this topic of server optimization is our ability to model split, merge, and many other complex topologies. §6 rounds out the paper with conclusions and open questions.

2 Problem Background

The determination of the number and the allocation of servers in an arbitrary topology queueing network is a complex problem. Many people have tackled this problem for single nodes, exponential service and infinite buffer queueing networks, yet not as much literature exists for the case when there are finite buffers, complex topologies, and general service time distributions in the network. The reason for this is basically due to the intractability of the problem for assessing the exact performance of a finite buffer queueing network of arbitrary topology, let alone optimizing them. While simulation could be a method of choice, employing simulation for large complex networks becomes prohibitive in terms of solution time, so we seek analytical approximations. We shall array some of the seminal analytical works in the area as well as outline the approaches to the problem utilized in the past.

2.1 Search for a Simple Formula

Within the infinite buffer queueing literature, there are certain simple formulas for determining the number of servers that are quite effective. One such formula and its derivatives is often referred to as the square root rule (it actually goes back to Erlang) where in this case, $\rho = \lambda/\mu$, is the proportion of time each server is busy, c is the number of servers, and γ is a constant giving the rough grade of service [42, 2]:

$$c = \rho + \gamma\sqrt{\rho} \tag{1}$$

We shall argue that the type of formula above becomes a useful bounding mechanism in the network topology search process and we will show that there is a reliable way to bound the optimal number of servers in finite queueing network topologies with the following expression. If the arrival rate to node i is λ_i, then the term on the left is a reliable lower bound and the term on the right provides an upper bound on $\mathbf{c}^*$:

$$\left\lceil \frac{\lambda_i}{\mu_i} \right\rceil \leq c_i^* \leq \left\lceil \frac{\lambda_i}{\mu_i} + \gamma\sqrt{\frac{\lambda_i}{\mu_i}} \right\rceil \tag{2}$$

Approaches for optimal server allocation are normally based on marginal allocation algorithms, convexity of the queueing performance parameters, and product-form properties of the queueing network system under study. In essence, as we shall argue, all these properties and concepts will be integrated in our solution methodology.

2.2 Literature Review

As mentioned earlier, there is a vast amount of literature for the optimal allocation of servers. Many studies have been done considering single nodes, open and closed networks, infinite and finite buffer waiting room, and exponential service systems. Of course, the latter topic of finite buffer, arbitrary topology, and general service networks is the most complex, and fewer studies have been accomplished.

Much of the earliest attempts to optimize the number of servers in infinite buffer exponential systems occurred in the late 50's and early 60's. The 70's, 80's and 90's also experienced a number of efforts where general service was considered.

Interest in finite queueing systems did not really take off until the 80's when networks of finite queues began to be considered. Since then, there have been a number of publications in the 80's and 90's and beyond on finite queueing systems both for single nodes and serial/tandem networks. The work of Kerbache and Smith [17], and the two-moment approximation of Smith [32], along with Powell's optimization algorithm [15] will be crucial to the optimization algorithms used in this paper. Figure 2 illustrates a sample view of the literature for this problem. Notice that there is little, if any, literature on network approaches treating general service and more general topologies. What we contribute in this paper is a multi-server optimization approach for these most general service acyclic, series-parallel networks.

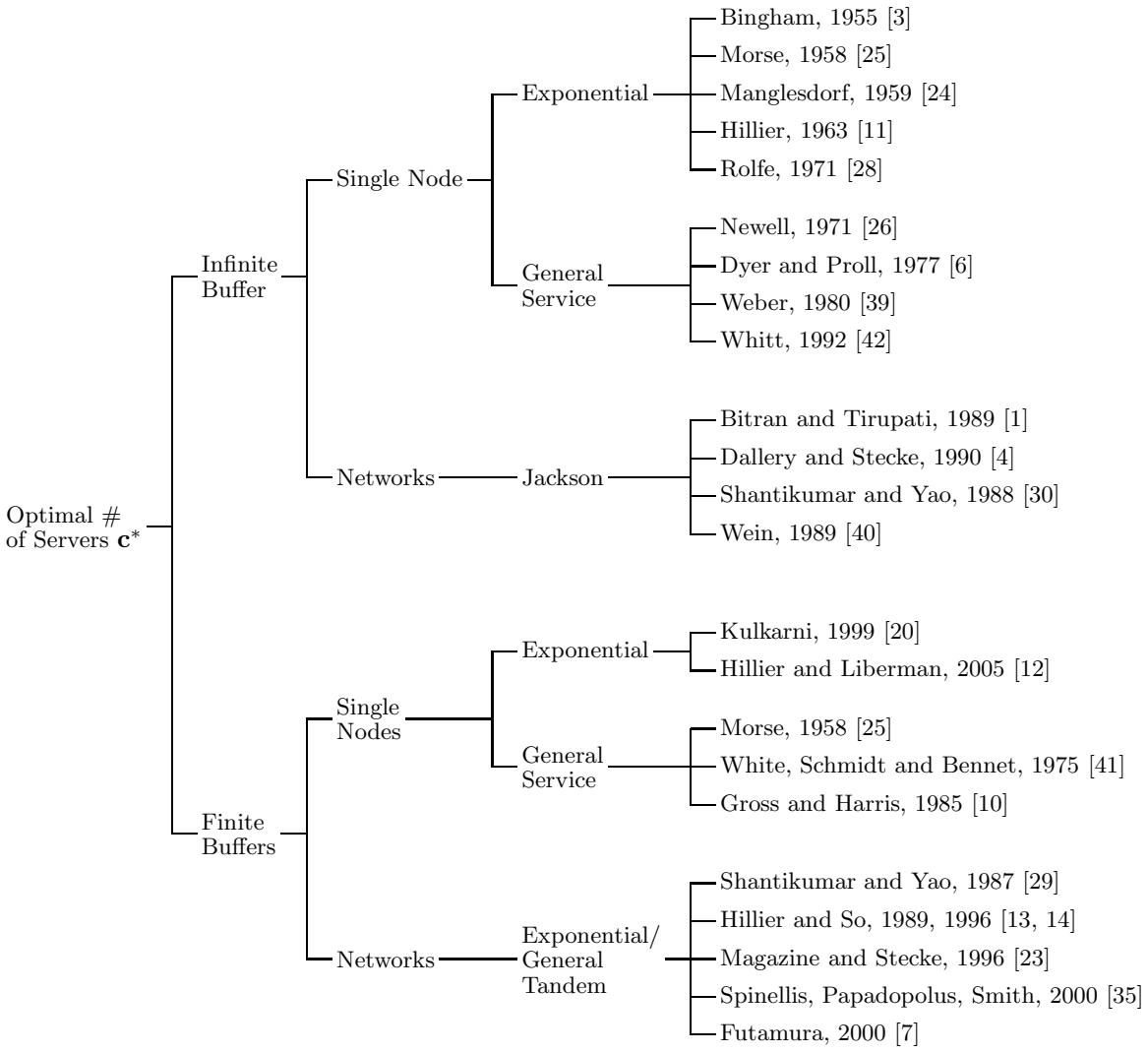


Figure 2: **Optimal Server Finite Queues and Networks References**

3 Mathematical Models

The mathematical models underlying our algorithmic procedure conjoin models based on two-moment approximations of the performance measures of finite queueing network systems and optimal search strategies for allocating the servers where closed form expression of the objective functions and constraints are unavailable. The assumptions and notation used in the model development follow.

3.1 Assumptions

We assume here that the time between arrivals is exponentially distributed with rate λ (that is, the number of arrivals in a fixed amount of time is a Poisson process) and that the service times are generally distributed with mean μ and squared coefficient of variation s^2. We also assume blocking after service (BAS) (sometimes referred to as *production or transfer blocking*) which is a typical protocol in manufacturing and facility planning applications. However, it is worthwhile mentioning that quite important systems would *not* follow BAS, such as communication networks, which would rather assume blocking before service (BBS), also known as *service blocking*, or even, sometimes, repetitive blocking (RBS), know as *rejection blocking*. The notation used in the model development follow.

3.2 Notation

This section presents most all of the notation we need for the paper. Figure 3 is a useful reference for the notation.

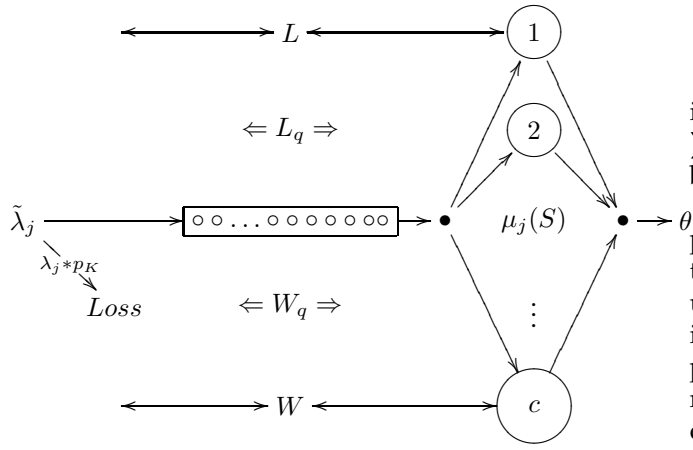


Figure 3: $M/G/c_j/K_j$ Queue Diagram

$\tilde{\lambda_j}$:= Effective arrival rate to node $j = \lambda_j(1 - p_K)$;

Λ := External Poisson arrival rate to the network;

$\mu_j(S)$:= Mean service rate at node j;

$\mathbf{c}$:= Vector number of servers;

$\epsilon \in (0, 1)$:= Threshold for the blocking probability;

B_j := Buffer capacity at node j *excluding* those in service;

K_j := Buffer capacity at node j *including* those in service;

N := Number of stations in the network

p_K := Blocking probability of finite queue of size K;

p_0^j := Unconditional probability that there is no customer in the service channel at node j (either being served or being held after service);

$\rho = \tilde{\lambda}/(\mu c)$:= Proportion of time each server is busy;

$s^2 = \mathrm{V}ar(T_s)/\mathrm{E}(T_s)^2$:= Squared coefficient of variation of the service time, T_s;

$\mathbf{x}$:= Server allocation vector of decision variables in the optimization routine

θ := Mean throughput rate.

θ^τ := Threshold Mean throughput rate.

3.3 *Optimization Problem*

In this paper, we consider the following type of optimization problem which also was the central objective used in our buffer allocation paper [32]:

$$\min \sum_i c_i, \tag{3}$$

subject to:

$$\Theta(\mathbf{c}) \geq \Theta^{\min}, \tag{4}$$

$$c_i \in \{1, 2, \ldots\}, \ \forall i, \tag{5}$$

in which we seek to minimize the total server allocation, $\sum_i c_i$, subject to providing a threshold throughput $\Theta^{\min}$ by mean of a set of integer numbers of servers c_i.

In the above formulation $\Theta^{\min}$ is a threshold throughput value and $c_i \equiv x_i$ is the decision variable, which is the server allocation at the i-th queue. In order to better understand the properties of the optimization problem, it is fruitful at this point to explain the method for computing the throughput, since this is critical to the optimization search process. This estimate of $\Theta(\mathbf{c})$ is based on a two-parameter approximation of this performance measure.

If one starts with the blocking probability of the $M/M/1/K$ system and treats K continuously, one can generate an expression for the continuous optimal buffer size as a function of p_K and s^2. If one fixes the number of servers, one can solve for the blocking probability of the system. In the case of $c = 1$, the following expression is obtained for the blocking probability:

$$p_K = \frac{\rho^{\left(\frac{-\sqrt{\rho\, e^{-s^2}} + 2\,B + \sqrt{\rho\, e^{-s^2}}\, s^2}{2 + \sqrt{\rho\, e^{-s^2}}\, s^2 - \sqrt{\rho\, e^{-s^2}}}\right)} (\rho - 1)}{\rho^{\left(2\,\frac{-\sqrt{\rho\, e^{-s^2}} + B + \sqrt{\rho\, e^{-s^2}}\, s^2 + 1}{2 + \sqrt{\rho\, e^{-s^2}}\, s^2 - \sqrt{\rho\, e^{-s^2}}}\right)} - 1}$$

This expression of the blocking probability is especially useful when $0 \leq s^2 \leq 1$. This process can be extended for $c > 1$, in fact, expressions for p_K of up to $c = 500$ have been found. Please see some of the other references for further details, [31, 32].

Analytical results from the $M/M/c/K$ model provide the following expression for p_K:

$$p_K = \frac{\left(\frac{\lambda}{\mu}\right)^K (c!)^{-1} \left(c^{K-c}\right)^{-1}}{\left(e^{\frac{\lambda}{\mu}} \Gamma(c, \frac{\lambda}{\mu}) \left(\Gamma(c)\right)^{-1} + \frac{\left(\frac{\lambda}{\mu}\right)^c \left(1 - \left(\frac{\lambda}{c\mu}\right)^{K-c+1}\right)}{(c!)\left(1 - \frac{\lambda}{c\mu}\right)}\right)}.$$

Since the blocking probability function is very complex, one cannot express the value of K without fixing c. If one fixes $c = 1$, the function becomes the same as the one which was derived from the previous formula for the $M/M/1/K$ system.

If one fixes $c = 2$, the following closed form expression for p_K can be developed from the $M/M/c/K$ formula:

$$p_K = \frac{2\,\rho^{\,2\frac{\sqrt{\rho\, e^{-s^2}}\, s^2 - \sqrt{\rho\, e^{-s^2}} + K}{2 + \sqrt{\rho\, e^{-s^2}}\, s^2 - \sqrt{\rho\, e^{-s^2}}}} (-2\,\mu + \lambda)}{2\,\rho^{\,2\frac{\sqrt{\rho\, e^{-s^2}}\, s^2 - \sqrt{\rho\, e^{-s^2}} + K}{2 + \sqrt{\rho\, e^{-s^2}}\, s^2 - \sqrt{\rho\, e^{-s^2}}}} \lambda - 2\,\mu - \lambda} \tag{6}$$

As we shall argue and show, we can continue this process for $c \geq 3$. The effectiveness of this blocking probability approximation was shown in [32].

3.4 *Bounds on ρ*

As stated in the introductory section, we can obtain a bound on ρ for allocating c servers, by examining the performance measures of $M/G/c/K$ systems. For example, if we focus on an $c = 1$, the performance graph of L the number in the system as a function of s^2 reveals a very interesting sigmoid(s-shaped) function, see Figure 4. At $\rho = 1$, there is an inflection point and the performance curves below $\rho < 1$ are convex increasing. After

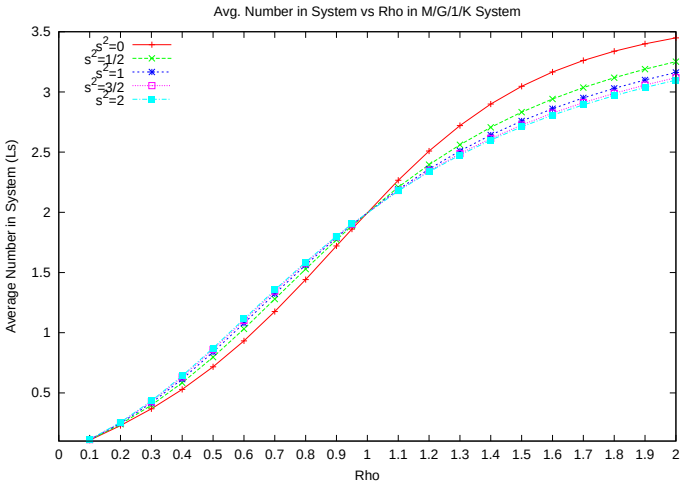


Figure 4: Avg. # in System for $M/G/1/4$ System

$\rho \geq 1$ the curves are concave. These s-shaped functions similar to Figure 4 become more pronounced as $K \to \infty$. These s-shaped functions occur for most all finite buffer multi-server systems of $c > 1$. Thus, in what follows, we will restrict our search for optimal $c^* \geq 1$ and for those systems where $\rho < 1$.

3.5 *Properties of the Objective Function*

In order to understand a fundamental property of our objective function, let's relax our objective function so that we have an unconstrained optimization problem in the number of servers and the throughput. We will focus on a single node, so that our objective becomes

$$\min Z_\theta = \mathbf{C}_c\mathbf{x} - C_\theta\theta(\mathbf{x}),$$

where $x \in \{1, 2, 3, \ldots\}$ is the number of integer servers. We have a linear term in the number of servers and the nonlinear term in the throughput function. We have added a cost term vector on the number of serves $\mathbf{C}_c$ and a profit term C_θ for the throughput so that we can examine the tradeoffs for the two performance measures. We wish to show that the unconstrained optimization problem where s^2 varies is a piecewise convex function.

Here is the proof sketch. Since the first term of the objective function is linear and therefore convex, it remains to show that the second term involving the $\theta(\mathbf{x})$ is concave and since $-\theta(\mathbf{x})$ is convex, then the summation of these two terms reveals an overall convex function. The key to showing the concavity of $\theta(\mathbf{x})$ is to show that the blocking probability which is a function of the multiple servers p_K is convex.

3.6 *Derivation of Concave Approximation Function*

The approach we will take is to examine the convexity of the blocking probability within the range of the parameters felt to be important in the research under study. We shall assume that $\rho < 1$ and that the range of the number of servers is $c = [2, 10]$. The fact that $\rho < 1$ is related to the fact that beyond $\rho \geq 1$, it appears that the general performance measures may no longer be convex.

Let us identify the threshold buffer size based upon the blocking probability for the $M/M/c/K$ system, where we subtract out the number of servers. This term is used for both the exponential approximation and the deterministic approximation in the two-moment approximation schema. Tijm's two-moment approximation [5, 36, 38] relies on a weighted combination of an exact (if available) expression of the $M/D/c/K$ blocking probability as well as the blocking probability of the $M/M/c/K$ formula.

Let's carry out the process for $c = 2$. The process can be carried out in a similar way for all $c \geq 2$. Equation 7 is the basic weighted average equation from Tijms [36]:

$$B_\epsilon^T(s^2) = s^2 B_\epsilon(M) + (1 - s^2)B_\epsilon(D) \tag{7}$$

in which $B_\epsilon^T(s^2)$ is Tijm's approximate buffer size, $B_\epsilon(M)$ is the exact buffer size for a Markovian system, and $B_\epsilon(D)$ is the exact buffer size for a deterministic service time system.

Of course, if exact expressions are available for both formulas, then Tijm's approximation is exact for the two extreme cases. Kimura, on the other hand, also has a two-moment approximation that turns out to be a little simpler and is the one built upon since it utilizes Markovian approximations as its basis. His expression is:

$$\tilde{B}_\epsilon(s^2) = B_\epsilon(M) + \text{NINT}\left\{\frac{(s^2 - 1)\sqrt{\rho}}{2} B_\epsilon(M)\right\} \tag{8}$$

in which NINT(x) is the nearest integer to x.

We then start with an expression for the threshold capacity of the queue as a function of the blocking probability formula.

$$K = \frac{\ln\left(\frac{1}{2}\frac{pk\,(2\,\mu+\lambda)}{2\,\mu-\lambda+pk\,\lambda}\right)}{\ln(\rho)}$$

The linking term between the deterministic and exponential terms is given by the following, see [31] for more details.

$$\frac{1}{2}\left(s^2 - 1\right)\sqrt{\frac{\rho}{e^{s^2}}}$$

Now, we combine the two expressions as a function of the buffer size to yield:

$$B = \frac{\ln\left(\frac{1}{2}\frac{pk\,(2\,\mu+\lambda)}{2\,\mu-\lambda+pk\,\lambda}\right)}{\ln(\rho)} - 2 + \frac{1}{2}\left(s^2-1\right)\sqrt{\frac{\rho}{e^{s^2}}}\left(\frac{\ln\left(\frac{pk\left(\frac{1}{2}\mu+\lambda\right)}{2\,\mu-\lambda+pk\,\lambda}\right)}{\ln(\rho)}\right) - 2$$

Then, we can simplify this expression into:

$$\frac{-1}{2}\frac{\left(2+\sqrt{\frac{\rho}{e^{s^2}}}s^2-\sqrt{\frac{\rho}{e^{s^2}}}\right)\left(-\ln\left(\frac{1}{2}\frac{pk\left(\frac{1}{2}\mu+\lambda\right)}{2\,\mu-\lambda+pk\,\lambda}\right)+2\,\ln(\rho)\right)}{\ln(\rho)}$$

Finally, after some more algebra, we arrive at the blocking probability, p_K, described in Equation (6). All of these transformations for $c \geq 1$ include the exact expression for the $M/M/c/K$ blocking probability when $s^2 = 1$.

We show here two examples of the p_K function, one for $c = [2, 10]$ and $s^2 = \{0, 2\}$ to reinforce the above properties. If we examine the blocking probability function for $c = 2$ and set $\lambda = 1, s^2 = 0$, then the p_K function is given in the following graphic Figure 5 over a range of K, μ. A similar graph for $c = 2, s^2 = 2$ occurs but

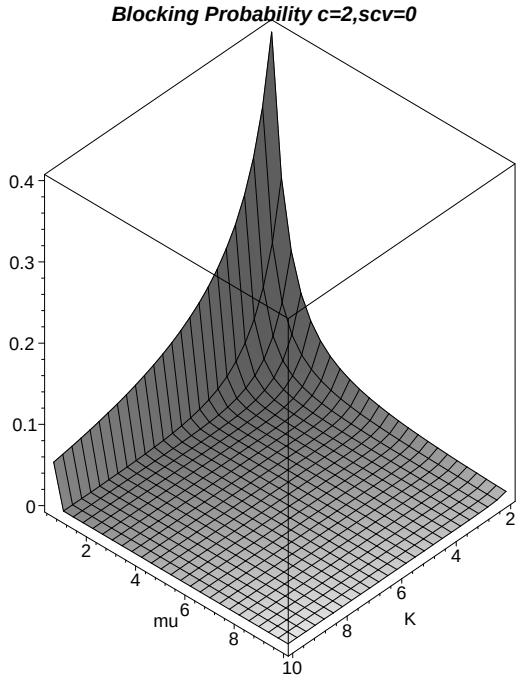


Figure 5: Convex Blocking Probability, $c = 2, s^2 = 0$

it is very similar to Figure 5. To complete our coverage of $c = 10$, the graph of our blocking probability for $c = 10, s^2 = 2$ occurs below in Figure 6. The derivation of the blocking probability formula for $c = 10$ is similar to the above, while the closed form expression of the blocking probability is very complicated.

With the above expressions for p_K we see that they are fundamentally based upon the blocking probability

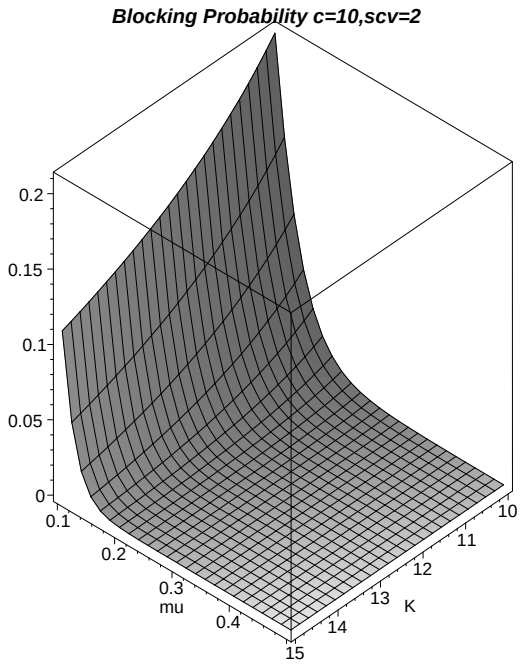


Figure 6: Convex Blocking Probability, $c = 10, s^2 = 2$

of the $M/M/c/K$ system. In fact, the expression for p_K we arrived at is based upon a convex combination of the blocking probabilities of the $M/M/c/K$ system. It is well-known that:

Fact: *A linear combination with positive coefficients of convex functions is a convex function.*

Köchel [19] has shown a number of properties of $M/M/c/K$ systems among which the following one on the convexity of the blocking probabilities is important.

Proposition 1: [19] *For each buffer waiting space $B = K - c$, the average number of per time unit rejected customers of the $M/M/c/K queue$, and the rejection probability p_K are decreasing convex functions of c.*

Since our expression for p_K is based on the $M/M/c/K$ system, we can conclude that:

Proposition 2: *The blocking probability based upon our two-moment expression is a convex function.*

3.7 Concavity of Throughput Function

Figure 7 illustrates in two dimensions the various bounds we are examining and μ_B represents the bound from the service rate. We studied this type of bound and its implications for exponential tandem networks in an earlier paper [9] but did not explore general service time distributions in any detail as we are doing in this current paper. While we will show that for single nodes in this section of the paper, the objective function is piecewise convex, our real interest is how this result

will translate into controlling the search for optimal c^* in network topologies.

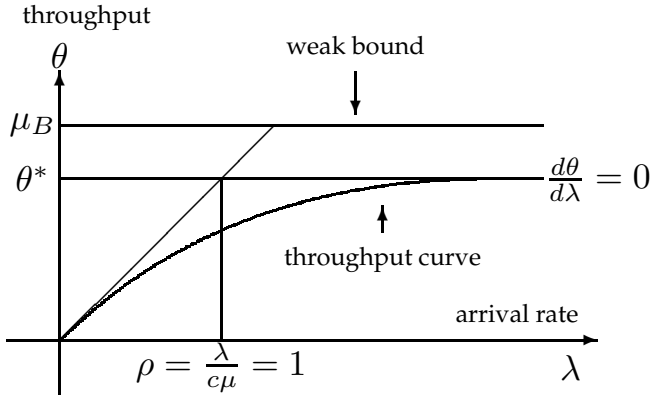


Figure 7: Threshold Throughput θ^* Bound

The final expression for the p_K, Equation (6) can then be used to compute the throughput for any squared coefficient of variation s^2. For example with $c = 2$, the throughput functions for $s^2 \in \{0, 2\}$ are given below.

$$\theta(s^2 = 0) = \lambda \left(1 - \frac{2\,\rho^{\frac{-2\sqrt{\rho}+2\,K}{2-\sqrt{\rho}}}\,(-2\,\mu+\lambda)}{\left(2\,\rho^{\frac{-2\sqrt{\rho}+2\,K}{2-\sqrt{\rho}}}\,\lambda - 2\,\mu - \lambda\right)} \right)$$

$$\theta(s^2 = 2) = \lambda \left(1 - \frac{2\,\rho^{\frac{6\sqrt{\rho\,e^{-4}}+2\,K}{2+3\sqrt{\rho\,e^{-4}}}}\,(-2\,\mu+\lambda)}{\left(2\,\rho^{\frac{6\sqrt{\rho\,e^{-4}}+2\,K}{2+3\sqrt{\rho\,e^{-4}}}}\,\lambda - 2\,\mu - \lambda\right)} \right)$$

Each one of these formulas is a convex combination of the $M/M/c/K$ blocking probability. In addition to Köchel, many authors have shown that the throughput function is stochastically monotone for a single server with general service time distributions, see Glasserman and Yao, [8]. Since the throughput function based upon our approximation is differentiable, in principle, we could exhaustively test the Hessian of our approximation to examine whether the function is negative semi-definite. This would be very tedious and not reveal anything significant. However, for all $s^2 \in [0, 2]$ and assuming K is continuous, the throughput function is differentiable. Therefore, since our throughput function is differentiable and can be generalized for all servers $c > 1$ [31, 32], we have:

$$\lim_{\lambda\to\infty} \frac{d\theta}{d\lambda} = 0.$$

Also, since the throughput function has a monotone linear segment starting from the origin at 45^o which intersects the segment of the derivative at θ^*, see Figure 7, then the overall throughput function is a concave function.

Proposition 3: *The throughput function θ based upon the two-moment approximation schema for a single node with multiple servers $c \geq 1$ and general service times in the range of $s^2 \in [0, 2]$ is a concave function.*

Explicitly constructing the concave function can be done with our approximation. As an illustration, Figure 8 shows the throughput function θ vs. λ for a single queue with $c = 2, s^2 = \{0, 1, 2\}, \mu = 2$. For the lower variability systems, the throughput function rises faster to the asymptotic limit. In all three cases the intersection point of the piecewise linear concave function corresponds to the threshold $\rho = \frac{\lambda}{c\mu} = 1$.

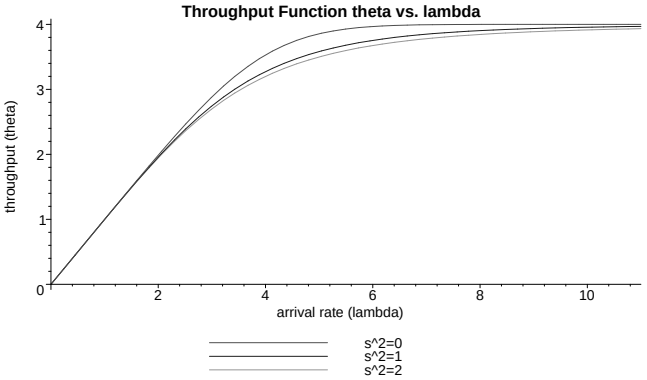


Figure 8: Throughput Function θ vs. λ

Because θ is concave and since $-\theta$ is convex, then the overall sum of these convex functions yields:

Proposition 4: *For a single node with multiple servers and assuming $s^2 \in [0, 2]$, the objective function Z_θ is a piecewise linear convex function.*

The piecewise nature of the objective function occurs because of the integer values for the servers. In an experiment to show the piecewise convexity of our objective function, we have a single queue where $\lambda = 10, \mu = 6, C_c = 15, C_\theta = 10$ and with a small fixed buffer with $K = 5$ and we wish to see how the objective cost function varies as a function of the squared coefficient of variation s^2, and the number of servers c. Notice in Figure 9 the clear separation of the convex curves for the different values of s^2. They are also non-overlapping. This non-overlapping of the curves also appears to be an important property in any approximation method.

What we also see is that for the low variability situations $s^2 = 0, 1/2$ (the two lower curves), the optimal buffer value is $c^* = 2$, while for the higher variability situations $s^2 = 1, 3/2$, (the two upper curves) the optimal number of servers is $c^* = 3$. Since in this experiment, the cost of the servers is higher than the throughput profit, we have more emphasis on the number of servers. The third curve for $s^2 = 1$ is exact while the others are approximations.

We also argue (but do not prove) that for general service time finite buffer networks, indeed the throughput

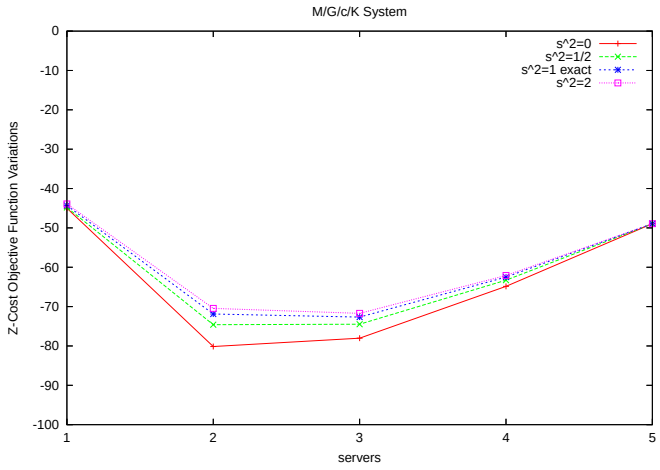
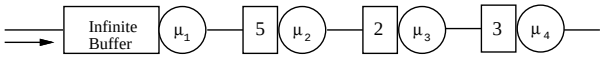


Figure 9: Piecewise Convex Objective Function, $K = 5$

function can be approximated by a piecewise linear concave function. Consider a tandem topology shown in Table 1. In an experimental analysis of this network, three resulting asymptotic throughput curves with various service rate distributions and parameters (exponential, normal, and uniform) generate identical, but shifted, piecewise concave throughput functions Z_θ as for the single node, see Figure 10.



A. Exponential	B. Normal	C. Uniform
$\mu_1 = 4$	$\mu_1 = 4, \sigma = 1.2$	$\mu_1^L = 3,\ \mu_1^U = 5$
$\mu_2 = 3$	$\mu_2 = 3, \sigma = 1.5$	$\mu_2^L = 2,\ \mu_2^U = 4$
$\mu_3 = 3.5$	$\mu_3 = 3.5, \sigma = 0.6$	$\mu_3^L = 2,\ \mu_3^U = 5$
$\mu_4 = 3.2$	$\mu_4 = 3.2, \sigma = 2.0$	$\mu_4^L = 3.2,\ \mu_4^U = 4.4$

Table 1: Service Parameters for Simulation Expts.

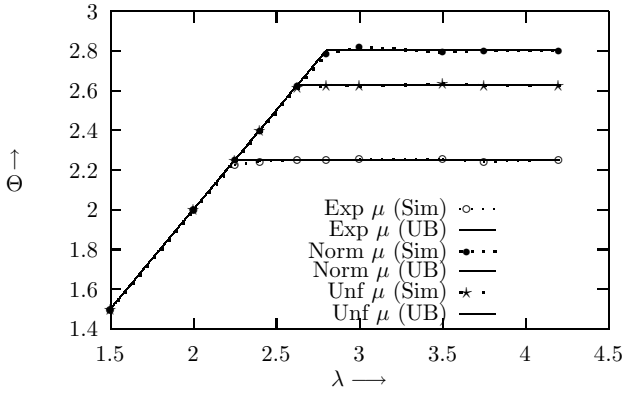


Figure 10: Piecewise θ for General Tandem Network

Additional experiments for more general series-topology systems and the existence of the piecewise concavity of the throughput function occur in [9]. Theoretical results which also support the concavity of the throughput function occur in [27] which ultimately would show that the objective function $Z(\theta)$ for these acyclic, series-parallel networks is convex. For another instance, Figure 11 illustrates a complex acyclic, series-parallel topology with general service time distributions and finite buffers. Figure 12 illustrates the concave piecewise linear function for the throughput of this network. Simulation experiments were used to generate the curve. If there is feedback in the network topology, the throughput function may be very difficult to isolate and control, so this is why we consider only acyclic, series-parallel network topologies.

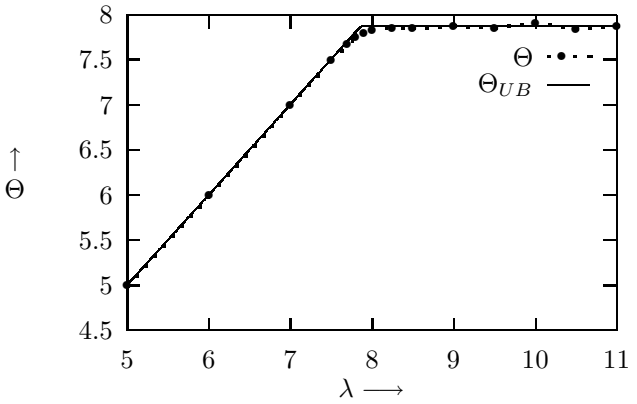


Figure 12: λ versus θ

Conjecture: *The objective function $Z_\theta = \mathbf{C}_c\mathbf{x} - C_\theta\theta(\mathbf{x})$ for multi-server acyclic, series-parallel queueing networks with general service times is a piecewise linear convex function.*

4 Algorithms

This section illustrates briefly, the key component algorithmic parts of the optimization methodology, namely the Expansion method for computing the throughput in the network topologies, and the derivative-free optimization search algorithm, Powell's Method, for determining the number of servers. Reference to some of our previous papers, gives further details of the steps of the methodology but we feel it is important to understand the basics of our methodology so that one will understand the results of the experiments in the next section of the paper.

4.1 General Expansion Method

We will need an accurate estimate for the throughput $\Theta(\mathbf{c})$, given a server allocation vector $\mathbf{c} = (c_1, c_2, \ldots, c_n)^{\mathrm{T}}$, in order to solve the optimal server allocation problem. The general expansion method (GEM) is an effective and robust technique to approximately estimate performance measure of finite queueing networks as we have here. The method, developed by Kerbache and Smith [17], was successful in the past coupled with optimization algorithms. Essentially, the

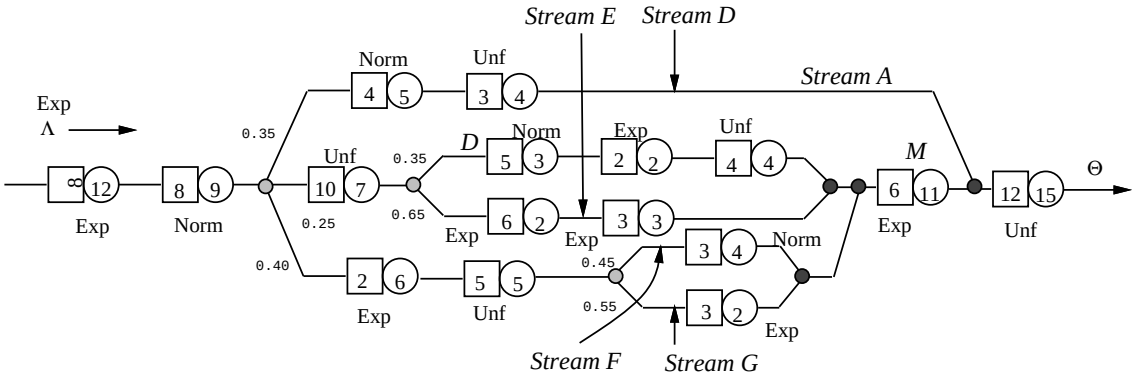


Figure 11: Complex Series-Parallel Topology

GEM is a combination of node-by-node decomposition and repeated trials. Each finite node is analyzed separately and corrections are made to take into account the interrelation between two adjacent finite queues. The GEM adopts blocking after service (BAS) protocol, which is prevalent in most production, manufacturing, transportation, and similar systems. Of course, external arrivals that find a full front buffer are rejected. Three stages are involved in the method, namely *network reconfiguration*, *parameter estimation*, and *feedback elimination*.

Network reconfiguration

In the fist step of the GEM, an artificial node h is created preceding each finite node to register blocked jobs. The resulting reconfigured network can be seen in Figure 13. Node h is modeled as an $M/G/\infty$ queue with service rate μ_h. Each job that leaves node i may be blocked with probability p_K or unblocked with probability $(1-p_K)$. If blocked, the entities are rerouted to vertex h, for a delay while server j is busy. After this delay, the entity may experience a second delay, with probability p'_K. Node h helps accounting the time an entity has to wait before finally joining vertex j and computing the effective arrival rate to vertex j.

Parameter estimation

In this stage, we essentially estimate the parameters p_K, p'_K, and the service rate for the holding node, μ_h. The blocking probability p_K is obtained by a two-moment approximation recently developed [31]. For $M/G/c/K$, $c = 2$, this approximation provides:

$$p_K = \frac{2\rho^{2(\frac{(2+\sqrt{\rho/e}s^2-\sqrt{\rho/e}+B)}{(2+\sqrt{\rho/e}s^2-\sqrt{\rho/e})}}(2\mu-\lambda)}{-2\rho^{2(\frac{(2+\sqrt{\rho/e}s^2-\sqrt{\rho/e}+B)}{(2+\sqrt{\rho/e}s^2-\sqrt{\rho/e})}}\lambda+2\mu+\lambda},$$

in which $\rho = \lambda/\mu$ is the traffic intensity and $s^2 = \mathrm{Var}(T_s)/\mathrm{E}(T_s)^2$ is the squared coefficient of variation of the service time T_s at the node. Other expressions are available for $c \in \{1, 2, \ldots, 10\}$. The feedback blocking

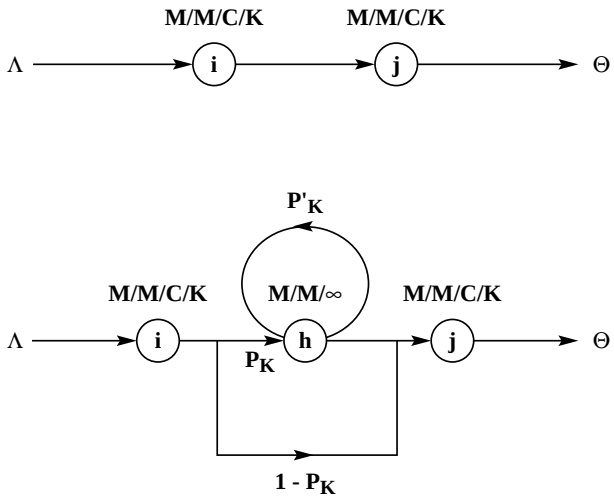


Figure 13: BAS protocol in Finite Queues

probability, p'_K, is obtained by an approximation from Labetoulle and Pujolle [21] based on diffusion techniques. Finally, the service rate of the holding node, μ_h, is given by renewal theory [18]. Further details will not be given here but the interested reader can find more information in the literature [17, 16].

Feedback elimination

Finally, the third stage is designed to deal with the strong dependences in the arrival process that are created by the repeated visits to the holding nodes due to the feedback. To eliminate this feedback, the customer is given an extra time during the first passage through the holding node, being the service rate corrected to [17, 16]:

$$\mu'_h = (1 - p'_K)\mu_h.$$

Summary

The GEM ultimate goal is to provide an update for the service rate of nodes that are preceded by finite nodes:

$$\tilde{\mu}_i^{-1} = \mu_i^{-1} + p_K(\mu'_h)^{-1}.$$

Similarly, the process can be extended to merge and split networks, as illustrated in Figures 14 and 15. Important to stress is that we do not physically modify the networks. The holding nodes are only artificial nodes included in the performance evaluation software.

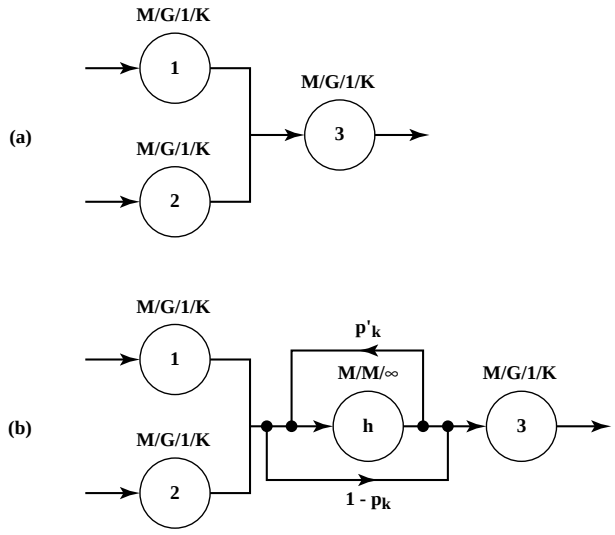


Figure 14: Merge Topologies

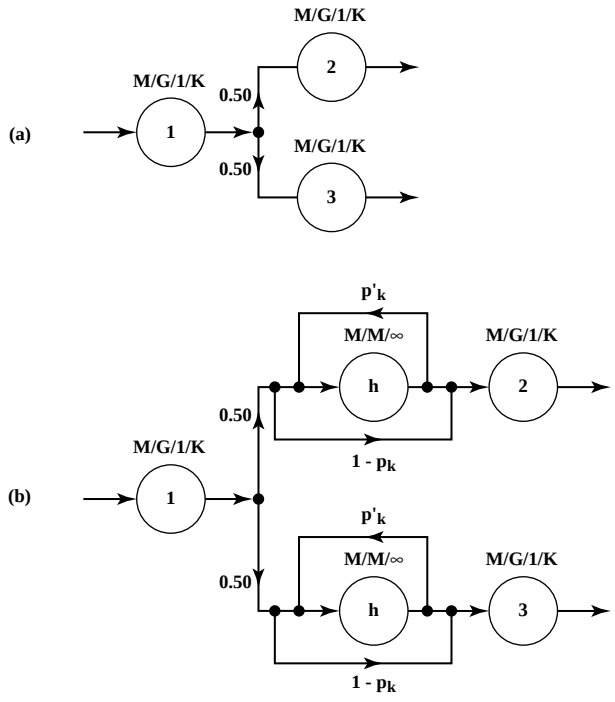


Figure 15: Split Topologies

4.2 *Optimization Algorithm*

One way to simplify the optimal server allocation problem is through a penalty function approach, similar to the Lagrangian relaxation (see details in [22]). Thus defining a dual variable α and relaxing constraint (4), the following penalized problem arises:

$$Z_\alpha = \min \left[\sum_{i=1}^{N} c_i + \alpha \Big(\Theta^{\min} - \Theta(\mathbf{c}) \Big) \right], \qquad (9)$$

subject to

$$c_i \in \{1, 2, \ldots\}, \ \forall i, \qquad (10)$$

$$\alpha \geq 0. \qquad (11)$$

Notice that if we relax the integrality constraints for c_i, the resulting problem is a classical unconstrained optimization problem, for a given α. Then, we can set α accordingly and use some classical nonlinear programming solver. Although variables c_i are integer, they will be approximated very reasonably by round off from the nonlinear programming solver, as we shall see. The algorithm we choose to use here is Powell algorithm, successful in the past for optimizing similar stochastic optimization problems [32]. Powell method [15] locates the minimum of a non-linear function by successive unidimensional searches from an initial starting point $\mathbf{c}^{(0)}$ along a set of conjugate directions, which are generated within the procedure itself.

In light of the server optimization problem, a number of concerns arose as to how to make sure the implementation of the algorithm would accommodate the bounds on $\rho < 1$. We had to put some special tests in the code to ensure that the search process would not consider vectors where $\rho \geq 1$ would occur and also not to violate the bound $c \leq K$.

Also, the integer nature of $\mathbf{c}$ was of some concern as the algorithm only treats real variable values. While a full implementation of a branch-and-bound type algorithm was ruled out since the worst case time complexity is exponential, the search process generally proceeds in integer increments, so that the integer rounding process implemented with the algorithm works pretty well.

4.3 *Algorithm Complexity Analysis*

The optimization algorithm we have developed is not polynomially bounded in N or $\mathbf{c}$. This is because the performance algorithm is dependent on solving a set of nonlinear equations, which are highly dependent on the starting solution to show convergence. This is typical of a Newton-type algorithm which is the underlying methodology used to solve the set of nonlinear equations. Even though Powell's method exhibits quasi-quadratic convergence, our algorithm is exponential in running time. While, it is exponential in running time, the solutions achieved in the example cases $N \in [3, 20]$ were very fast as we will show.

5 Experimental Design

We have developed experimental results for testing the algorithm for series, merge, split topology and their natural combinations. To exhaustively treat the issues in

this first part of the experiment section, we have focused on small networks, where we could explore the key relationships between c, K, and s^2, μ, λ.

5.1 2-node Series Topologies

The 2-node series topology is the simplest type of queueing network, other than a feedback loop network involving one node. Figure 16 illustrates the arrangement of nodes we examined. We decided to carefully examine the interplay between the number of servers **c** and the buffer vector **K** and also the variability factor through the effect of the squared coefficient of variation s^2.

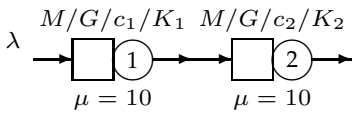


Figure 16: Series 2-node topology

5.1.1 Uniform Buffers $K = \{(5,5),(10,10),(25,25)\}$

A series of three experiments were conducted where uniform buffers of $K = \{5, 10, 25\}$ were utilized and experiments run with the coefficient of variation ranging between $s^2 = \{0, \frac{1}{2}, 1, 2\}$. A number of interesting patterns were created as can be seen in the following tables. As one would intuitively suspect that the smaller buffers would be allocated more servers and also in proportion to the variability in the system and this essentially turns out to be the case. For the higher variability case, more servers are needed but at a higher cost, in order to achieve the threshold throughput requirements.

Table 2: 2-node Optimal Server Results $K = (5,5)$

					Simulation		
λ	s^2	c_1, c_2	θ_α	Z_α	θ_s	δ	Z_s
5	0	2,2	4.999	4.17	4.993	.00190	11.0
	$\frac{1}{2}$	2,2	4.999	4.32	4.992	.00260	12.0
	1	2,2	4.999	4.57	4.994	.00227	10.0
	2	2,2	4.999	5.42	4.990	.00236	14.0
9	0	3,3	8.999	6.52	8.989	.00335	17.0
	$\frac{1}{2}$	3,3	8.999	6.99	8.987	.00319	19.0
	1	4,3	8.999	7.01	8.989	.00315	18.0
	2	5,2	8.999	7.01	8.981	.00293	26.0
	2*	4,3			8.984	.00358	23.0

We noticed that in the experiment with $\lambda = 9, s^2 = 2$, the number of servers was approaching the fixed value of $K = 5$, so we limited the λ value to 9 so as to not violate the bound $c \leq K$. Also, notice, that for the last two experiments in Table 2 that we re-ran the last experiment to check the optimal allocation $(5,2)$ to see if an equivalent number of servers in the allocation $(4,3)$ was better or not and we see that $(4,3)$ is slightly better in the simulated throughput, so it remains a toss-up whether the algorithm's $(5,2)$ result is optimal or not for the last experiment, even though there is more variability in the simulated throughput with the $(4,3)$ allocation.

In contrast to Table 2, Table 3 with more capacity illustrates fewer servers are needed to achieve the same threshold throughput with the increased buffer constraint $K = (10,10)$. Notice that at $\lambda = 9, s^2 = 2$ the largest number of servers $(3,3)$ are allocated but not nearly as many as required for this worst case situation in Table 2.

Table 3: 2-node Optimal Server Results $K = (10,10)$

					Simulation		
λ	s^2	c_1, c_2	θ_α	Z_α	θ_s	δ	Z_s
5	0	1,1	4.999	2.04	4.993	.00169	9.0
	$\frac{1}{2}$	1,1	4.999	2.43	4.993	.00248	9.0
	1	2,1	4.999	3.69	4.993	.00244	10.0
	2	2,2	4.999	4.00	4.995	.00200	9.0
9	0	2,2	8.999	4.10	8.992	.00284	12.0
	$\frac{1}{2}$	2,2	8.999	4.31	8.992	.00264	12.0
	1	2,2	8.999	4.76	8.990	.00289	14.0
	2	3,3	8.999	6.02	8.992	.00338	14.0

Table 4 illustrates the allocation for a situation where $K = (25,25)$. Again, fewer servers are required, and a more uniform allocation results.

Table 4: 2-node Optimal Server Results $K = (25,25)$

					Simulation		
λ	s^2	c_1, c_2	θ_α	Z_α	θ_s	δ	Z_s
5	0	1,1	5.000	2.00	4.993	.00180	9.0
	$\frac{1}{2}$	1,1	5.000	2.00	4.993	.00251	9.0
	1	1,1	5.000	2.00	4.994	.00268	8.0
	2	1,1	4.999	2.01	4.994	.00241	8.0
9	0	2,2	9.000	4.00	8.991	.00236	13.0
	$\frac{1}{2}$	2,2	9.000	4.00	8.992	.00242	12.0
	1	2,2	9.000	4.00	8.992	.00358	12.0
	2	2,2	9.000	4.00	8.990	.00237	14.0

5.1.2 Unbalanced Buffers

A similar set of experiments were conducted where now the buffers are unbalanced to test if the server allocation would be inversely proportional to the variability in the system. In fact, upon examining the tabular outputs, this indeed, is the case.

In Table 5 one sees an unbalanced allocation across the experiments reflecting a bottleneck type of problem since the buffer allocation is $K = (10,5)$. What is interesting is that the first station receives more servers than the second even though the second station has fewer buffers. It appears to be the fact that the first station

is buffering the arrival process, so it is better to allocate the extra servers at the initial station. This results seems, however, to be a function of the buffer size as we shall show. In the last experiment in Table 5, another simulation experiment was run to test the optimal allocation from the algorithm. The $(3,2)$ allocation by the algorithm was contrasted with the $(2,3)$ on and the simulation indicated that our server allocation had higher throughput and thus was better.

Table 5: 2-node Optimal Server Results $K = (10, 5)$

					Simulation		
λ	s^2	c_1, c_2	θ_α	Z_α	θ_s	δ	Z_s
5	0	1,1	4.999	3.04	4.993	.00169	10.0
	$\frac{1}{2}$	1,2	4.999	3.39	4.993	.00269	10.0
	1	1,2	4.999	4.00	4.995	.00268	8.0
	2	2,2	4.999	4.00	4.992	.00244	12.0
9	0	2,2	8.999	4.10	8.992	.00284	12.0
	$\frac{1}{2}$	2,2	8.999	4.31	8.991	.00395	13.0
	1	2,2	8.999	4.76	8.992	.00348	12.0
	2	3,2	8.999	5.24	8.990	.00246	15.0
	2*	2,3			8.984	.00247	21.0

In Table 6 which is the mirror opposite of the previous experiment, some interesting results occur. The optimal solutions are not truly symmetric as one might expect when the traffic is high. Especially, the last four experiments, it is more expensive to allocate the servers in this configuration with $K = (5, 10)$ than in the previous $K = (10, 5)$. Also an additional experiment in the heavy traffic case was run to compare with the algorithm's allocation $(4, 2)$ and once again, the algorithm's allocation is better than $(3, 3)$.

Table 6: 2-node Optimal Server Results $K = (5, 10)$

					Simulation		
λ	s^2	c_1, c_2	θ_α	Z_α	θ_s	δ	Z_s
5	0	2,1	4.999	3.04	4.993	.00190	10.0
	$\frac{1}{2}$	2,1	4.999	3.36	4.994	.00236	9.0
	1	2,1	4.999	4.05	4.992	.00216	11.0
	2	2,2	4.999	4.00	4.991	.00255	13.0
9	0	3,2	8.999	5.10	8.990	.00335	15.0
	$\frac{1}{2}$	3,2	8.999	5.31	8.989	.00326	16.0
	1	3,2	8.999	5.45	8.980	.00302	25.0
	2	4,2	8.999	6.24	8.984	.00397	22.0
	2*	3,3			8.974	.00326	32.0

The next pair of experiments compare the buffer allocation with $K = (25, 10)$ and $K = (10, 25)$. The results in Table 7 are pretty consistent with the previous experiments, with fewer servers needed to meet the threshold throughput. The experiment with $\lambda = 5, s^2 = 2$ is interesting because the algorithm allocated the extra server to the second node, rather than the first node, probably because the buffer at the first node is much larger than the second. Also, notice again in the last experiment $\lambda = 9, s^2 = 2$, the server allocation is $(2, 2)$ and again we tested another alternative allocation $(1, 3)$ but it is not as good as the algorithm's and the alternative has more variability in the throughput output.

Table 7: 2-node Optimal Server Results $K = (25, 10)$

					Simulation		
λ	s^2	c_1, c_2	θ_α	Z_α	θ_s	δ	Z_s
5	0	1,1	4.999	2.02	4.993	.00169	9.0
	$\frac{1}{2}$	1,1	4.999	2.15	4.993	.00211	9.0
	1	1,1	4.999	2.69	4.993	.00268	9.0
	2	1,2	4.999	3.00	4.987	.00237	16.0
9	0	2,2	9.000	4.00	8.991	.00236	13.0
	$\frac{1}{2}$	2,2	9.000	4.00	8.989	.00265	15.0
	1	2,2	9.000	4.00	8.991	.00243	13.0
	2	2,2	9.000	4.00	8.990	.00217	14.0
	2*	1,3			8.842	.00384	162.0

Finally, Table 8 illustrates the symmetric option for our previous experiment. The server allocation is more expensive for this buffer allocation, especially in the $\lambda = 9$ range. The experiment with $\lambda = 5, s^2 = 2$ is also interesting because it allocated the extra server to the first node, rather than the second node, as in the previous experiment. Finally, in the last experiment $\lambda = 9, s^2 = 2$, the allocation by the algorithm $(3, 2)$ is better than the alternative $(2, 3)$. So from our 2-node experiments, the server allocation from the algorithm seems pretty consistent, yet the allocation depends on how the K, μ, p_K, λ and s^2 interact, so no simple design rules always emerge. Nonetheless, this is the very reason why some type of algorithm is necessary.

Table 8: 2-node Optimal Server Results $K = (10, 25)$

					Simulation		
λ	s^2	c_1, c_2	θ_α	Z_α	θ_s	δ	Z_s
5	0	1,1	4.999	2.02	4.993	.00169	9.0
	$\frac{1}{2}$	1,1	4.999	2.28	4.995	.00250	7.0
	1	2,1	5.000	3.00	4.992	.00241	11.0
	2	2,1	4.999	3.00	4.993	.00244	10.0
9	0	2,2	9.000	4.00	8.992	.00284	12.0
	$\frac{1}{2}$	2,2	9.000	4.00	8.991	.00296	13.0
	1	2,2	9.000	4.00	8.990	.00304	14.0
	2	3,2	9.000	5.00	8.993	.00265	12.0
	2*	2,3			8.987	.00276	18.0

5.2 *3-node Series Topologies*

To examine the issue of network topology, three node experiments will be examined: i) Series, ii) Split, and iii) Merge. First, let's assume we have a series topology with 3-nodes, finite buffers $K_i = 25\ \forall i$, service rates $\mu_i = 10\ \forall i$, and we shall vary the arrival rate

$\lambda \in \{5, 7, 8\}$, see Figure 17. For the sake of the argument, there is no cost for the buffers, only the tradeoff between the number of servers and meeting the throughput threshold accounts for the objective function values. The results compared with simulation of the same systems with Arena simulation models are included in Table 9. The results indicate that as ρ increases and there is more variability in s^2, more servers are added to the system. In this first set of experiments, $\rho < 1$, and a uniform pattern of service allocations results.

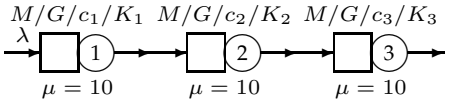


Figure 17: Series 3-node topology

As a simplified simulation comparison, twenty replications of each experiment were conducted with Arena, with the first 2000 time units discarded, and $100,000$ time units carried out for each simulation run to yield the 95% confidence intervals.

Table 9: 3-node Optimal Server Results

					Simulation		
s^2	λ	$\mathbf{c}$	$\theta(\mathbf{x})$	Z_α	$\theta(\mathbf{x})^s$	δ	Z^s_α
0	5	(1,1,1)	5.000	3.000	4.993	.00202	9.90
	7	(1,1,1)	6.999	3.001	6.992	.00253	10.80
	8	(1,1,1)	7.999	3.159	7.992	.00325	11.50
1	5	(1,1,1)	5.000	3.000	4.993	.00314	9.80
	7	(1,1,1)	6.999	3.591	6.992	.00362	11.20
	8	(2,2,2)	8.000	6.000	7.993	.00326	12.80
2	5	(1,1,1)	4.999	3.010	4.994	.00263	8.80
	7	(2,2,2)	7.000	6.000	6.993	.00399	12.70
	8	(2,2,2)	8.000	6.000	7.993	.00329	12.70

While the throughput calculations of the analytical approach perhaps seems overly optimistic, compared with simulation, even when the pattern of allocation is $(1, 1, 1)$ to augment the servers, does not seem justified, unless the variability in the system is increased. In order to further elucidate this issue, let's examine a worst case scenario with the service variability $s^2 = 2$, the last experiment in Table 9. Let's fix the total number of servers to six, and examine exhaustively the server allocation for this topology. We will generate seven possible configurations in the neighborhood of our supposed optimal allocation $(2, 2, 2), s^2 = 2$. Table 10 illustrates the simulation output of all these neighborhood configurations. All experiments were done with Arena according to the same parameter restrictions on the runs as before.

Thus as we can see, in Table 10 the $(2, 2, 2)$ server allocation is optimal. This is not a proof, but convincing evidence that our algorithm is pretty accurate, even when there is a large degree of server variability.

Table 10: 3-node Optimal Neighborhood Server Results

$\lambda = 8, s^2 = 2$		Simulation		
Experiment	$\mathbf{c}$	θ_s	δ	Z^s_α
1	(2,2,2)	7.9933	.00329	12.70
2	(3,2,1)	7.9911	.00301	14.90
3	(1,2,3)	7.9695	.00407	36.50
4	(1,3,2)	7.9683	.00435	37.70
5	(4,1,1)	7.9920	.00282	14.00
6	(1,1,4)	7.9601	.00322	45.90
7	(1,4,1)	7.9663	.00349	39.70

5.3 Splitting Topologies

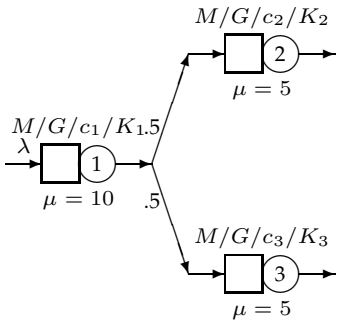


Figure 18: Split 3-node topology

For splitting topologies of 3-nodes seen in Figure 18, we will also assume $K_i = 25$, service rates $\mu_i = 10, 5, 5$ and we shall vary the arrival rate $\lambda \in \{5, 7, 8\}$. The reduction in the service rate for the two split nodes is to equalize the load on these nodes since the split flows are $50 - 50$. Table 11 illustrates our results, which for the sake of the argument also provide uniform server allocations, with more servers for the higher variable situations. As a check on our optimal results, the next-to-last experiment with $s^2 = 2$ was re-run in Arena with a six-server allocation of $(4, 1, 1)$ as indicated in the last experiment in Table 11. The simulation of $(4, 1, 1)$ has a lower throughput and consequently the Z value is also worse in comparison to the $(2, 2, 2)$ solution from our algorithm. So again, this strengthens our algorithm results.

5.4 Merging Topologies

For merging topologies of 3-nodes seen in Figure 19, we will also assume $K_i = 25$, service rates $\mu_i = 5, 5, 10$ and we shall vary the arrival rate $\lambda \in \{5, 7, 8\}$. Again the two streams merging into the 3rd have slower arrival rates so as not to overly load the merge node. Table 12 illustrates our results, which for the sake of the argument also provide uniform server allocations, with more servers for the higher variable situations.

Table 11: 3-node Optimal Server Results

					Simulation		
s^2	λ	$\mathbf{c}$	$\theta(\mathbf{x})$	Z_α	$\theta(\mathbf{x})^s$	δ	Z^s_α
0	5	(1,1,1)	5.000	3.000	4.995	.00356	8.50
	7	(1,1,1)	6.999	3.001	6.993	.00306	9.90
	8	(2,2,2)	7.999	3.105	7.991	.00387	12.10
1	5	(1,1,1)	5.000	3.000	4.995	.00302	8.00
	7	(1,1,1)	6.999	3.394	6.991	.00410	12.50
	8	(2,2,2)	8.000	6.000	7.991	.00411	15.50
2	5	(1,1,1)	4.999	3.007	4.995	.00289	8.10
	7	(2,2,2)	7.000	6.000	6.992	.00432	14.60
	8	(2,2,2)	8.000	6.000	7.993	.00416	13.30
	8	(4,1,1)			7.987	.00290	18.70

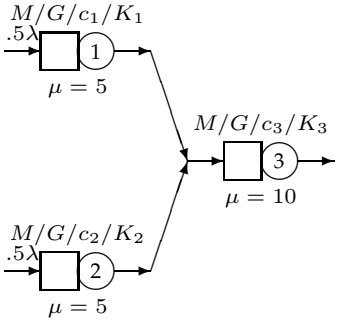


Figure 19: Merge 3-node topology

Again as a test on the performance of our optimization algorithm, the next-to-last experiment with $s^2 = 2$ was re-run with a six server allocation of $(1, 1, 4)$ to compare with our $(2, 2, 2)$ allocation. The resulting throughput of the $(1, 1, 4)$ last experiment was dramatically smaller and correspondingly, the Z value was greatly inflated, thus, giving more confidence to our algorithm results.

5.5 *Complex Topologies*

As one example of a complex network topology, let us examine the four-node topology presented in Figure 20. Here, the arrival and service rates are shown on the Figure 20.

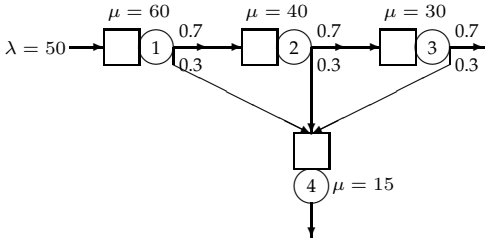


Figure 20: Complex-4-node Topology

In the first instance all buffers were all assumed to be

Table 12: 3-node Optimal Server Results

					Simulation		
s^2	λ	$\mathbf{c}$	$\theta(\mathbf{x})$	Z_α	$\theta(\mathbf{x})^s$	δ	Z^s_α
0	5	(1,1,1)	5.000	3.000	4.993	.00312	10.00
	7	(1,1,1)	6.999	3.001	6.989	.00298	13.80
	8	(1,1,1)	7.999	3.105	7.991	.00302	11.70
1	5	(1,1,1)	5.000	3.000	4.992	.00267	11.00
	7	(1,1,1)	6.999	3.394	6.989	.00404	14.00
	8	(2,2,2)	8.000	6.000	7.990	.00354	16.50
2	5	(1,1,1)	4.999	3.007	4.990	.00256	12.70
	7	(2,2,2)	7.000	6.000	6.990	.00348	16.50
	8	(2,2,2)	8.000	6.000	7.988	.00331	17.80
	8	(1,1,4)			7.964	.00302	42.20

$K_i = 25$. Even with the large number of buffers, there is a great deal of blocking as one might expect, since the split processes of the three queues into node #4 are all competing for the same buffer. In Table 13, simulation runs of the different experiments are included. Each simulation experiment took around 5 minutes time and 20 replications were carried out for $102,000$ time units with a warm-up period of 2000 time units. Notice that the last experiment with $s^2 = 2$ had a very large half-width compared to the other experiments. This indicates a great deal of variability in this experiment, probably due to the heavy blocking and resource utilization. Another run of the third experiment was done (resuls not shown) with a run length of $202,000$ time units, but no essential difference was recorded in the mean throughput value while the half-width was reduced somewhat.

Table 13: 4-node Complex Optimal Server Results $K_i = 25$

					Simulation		
s^2	λ	$\mathbf{c}$	$\theta(\mathbf{x})$	Z_α	$\theta(\mathbf{x})^s$	δ	Z^s_α
0	50	(2,2,1,3)	49.999	9.328	49.977	.00717	31.0
1	50	(2,2,2,4)	49.999	11.004	49.983	.00685	27.0
2	50	(2,2,2,4)	49.999	11.092	49.177	.01563	833.0

In the second set of runs, buffers were increased to $K_i = 50$ to see what effect they had on the server allocation process. There is a significant decrease in the number of servers at each of the queues, since the additional buffer resources begins to better handle the traffic. Again, the last experiment with $s^2 = 2$ indicates a great deal of variability in this run.

5.6 *Production Line Example*

We will model a five stage production line which is a push regime as opposed to a pull system. Processing

Table 14: 4-node Complex Optimal Server Results $K_i = 50$

					Simulation		
s^2	λ	$\mathbf{c}$	$\theta(\mathbf{x})$	Z_α	$\theta(\mathbf{x})^s$	δ	Z^s_α
0	50	(1,1,1,3)	49.999	7.004	49.979	.00873	27.0
1	50	(1,2,1,3)	49.999	8.851	49.983	.01097	37.0
2	50	(2,2,2,3)	50.000	10.000	49.775	.01321	234.0

rates are $\mu_1 = 6.5, \mu_2 = 5.0, \mu_3 = 8.0, \mu_4 = 5.0, \mu_5 = 6.0$ and all processing times are deterministic $s^2 = 0$. The finite capacity of the production line has four buffers for each machine, so $K = 5$ for all machines. We wish to see whether single servers can handle the traffic.

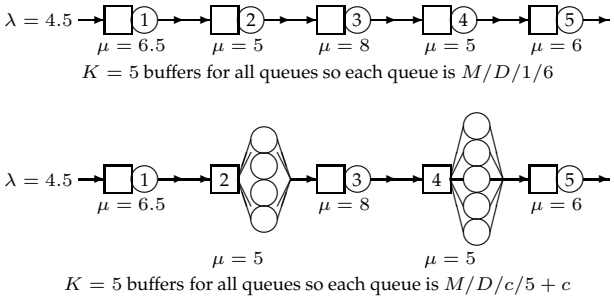


Figure 21: 5-station Production Line

To see the significance of the processing time assumption for the production line over say an exponential model of the production line, let's examine a single workstation with $K = 4$ and compare the probability distribution of the number in the system with an $M/M/1/K$ assumption. We will compare the workstations with $\rho = \{\frac{1}{4}, \frac{19}{20}, \frac{3}{2}\}$ traffic intensities. These probability distributions were generated by the two-moment approximation schema and they are quite accurate. Notice that in the low traffic intensity, Figure 22,

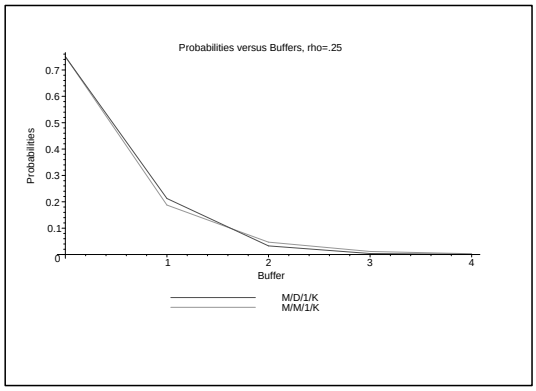


Figure 22: M/D/1/K and M/M/1/K Comparison

there is very little difference in the probability distributions between the M/D/1/K and M/M/1/K systems. However, when the traffic intensity approaches $\rho = 1$, Figure 23, there is a significant difference in the probability distributions. Finally, when $\rho = \frac{3}{2}$, see Figure 24, there still is a significant difference remaining.

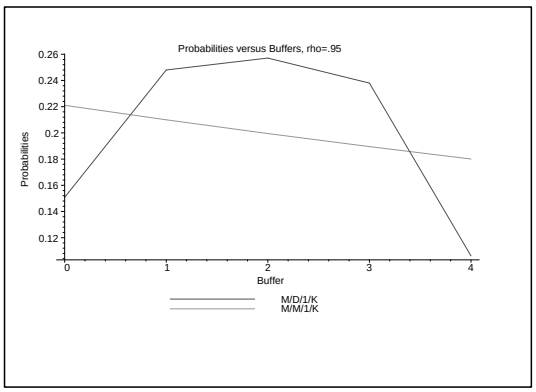


Figure 23: M/D/1/K and M/M/1/K Comparison

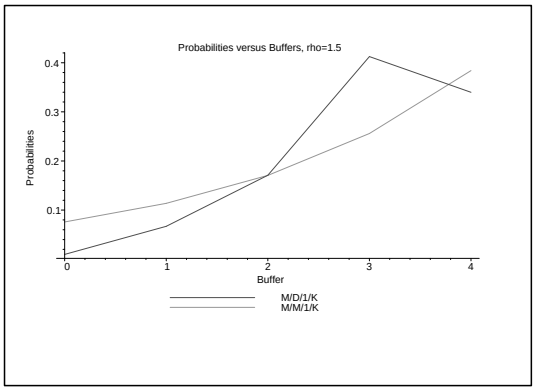


Figure 24: M/D/1/K and M/M/1/K Comparison

Table 15: 5-node Production Line Optimal Server Results K varies

					Simulation		
λ	K	$\mathbf{c}$	$\theta(\mathbf{x})$	Z_α	$\theta(\mathbf{x})^s$	δ	Z^s_α
4	(5,5,5,5,5)	(2,2,2,2,2)	3.999	10.189	3.9944	0.00204	15.60
4	(10,10,10,10,10)	(1,2,1,2,1)	3.999	7.880	3.9932	0.00188	13.80
4	(25,25,25,25,25)	(1,1,1,1,1)	3.999	5.032	3.9932	0.00188	11.80
4	(50,50,50,50,50)	(1,1,1,1,1)	4.000	5.000	3.9932	0.00188	11.80
5	(5,5,5,5,5)	(2,2,3,2,2)	4.999	11.026	4.9932	0.00203	17.80
5	(10,10,10,10,10)	(2,2,1,2,2)	4.999	9.218	4.9934	0.00186	15.60
5	(25,25,25,25,25)	(1,2,1,2,1)	4.999	7.103	4.9932	0.00176	13.80
5	(50,50,50,50,50)	(1,2,1,2,1)	5.000	7.000	4.9934	0.00180	13.60

Since we will be operating the line at a high $\rho \to 1$, the discrete time distribution will have a major effect on the server allocation. In the experiments which follow, the two middle machines M_2, M_4 are the bottlenecks, so the allocation of servers will be largely affected by their μ, K combinations. The results are rather interesting. In the first instance, with $\lambda = 4$, a uniform allocation with $\mathbf{K} = 5$ seems plausible, however, one might think that a server allocation of $(1, 1, 1, 1, 1)$ might be better. Doing one run of this latter configuration with the simulation

model (30 runs), yielded a throughput and corresponding half-width of $\theta = 3.9793, \delta = (.00195)$ so this latter allocation is clearly not optimal. In the second experiment, a rather intuitive allocation occurred, which actually buffers the bottlenecks. Again, in order to reach the threshold throughput while minimizing the number of servers, this allocation was suggested. The final two experiments with $\lambda = 4$ showed that $\mathbf{K} = (25, 50)$ was enough to reduce the number of servers to single values.

In the $\lambda = 5$ experiments, even more interesting results occurred. When single servers were used, a throughput of $\theta = 4.7492$ resulted which is clearly unacceptable. For the first experiment, with $\mathbf{K} = 5$, the algorithm allocation favored the middle server which seems to support the bowl phenomenon [13]. While in the second experiment, the phenomenon of allocating servers to the bottlenecks occurred when $\mathbf{K} = 10$ with an interesting solution of $\mathbf{c} = (2, 2, 1, 2, 2)$. Finally, for the latter two experiments $\mathbf{K} = (25, 50)$, this buffering option continued to be applied with a slight reduction $\mathbf{c} = (1, 2, 1, 2, 1)$. To emphasize this point once again, it is very difficult to predict the server allocation patterns *a* priori. One needs some type of algorithm because with general service time distributions, predicting the server patterns becomes quite complex. Futamura [7], pp 83-87, performs similar experiments with series systems and his results roughly parallel ours with respect to the unpredictability of the patterns found. In other words, the inverted bowl effect [13] sometimes happens, but other times, it does not occur.

To finally round out the set of general complex manufacturing experiments, a larger series network with ten queues and variations in buffer sizes was carried out with a range of arrival rate fluctuations and s^2 variations. The service time at all the queues had a mean of 10 while the buffers where the embolden values represent the increased buffer size

$$(25, 25, 25, \mathbf{50}, \mathbf{50}, \mathbf{50}, 25, 25, 25, 25)$$

so queues $4, 5, 6$ had more buffers than the other queues. The optimal algorithm yielded interesting server allocation patterns where more servers are allocated to the lower buffer sizes as indicated in Table 16 instead of the queues #4,5,6. This is intuitively understandable, given the conjoint effects of the buffers and service time changes. Finally, an extension of the above tandem system was carried out for twenty-one queues in order to show that our algorithm is capable of solving larger networks. Again, the buffer allocation was nonsymmetrical and is as follows, where the embolden values represent the increased buffer size:

(25, 25, 25, **50, 50, 50**, 25, 25, 25, 25, 25, 25, 25, **50, 50, 50**, 25, 25, 25, 25, 25)

Looking at Table 17 we see that the server allocation is pretty uniform until $\lambda = 17$ and then because of the buffer differential, the number of servers at the larger buffer ($K = 50$) is smaller than the other locations. This indicates that our algorithm is working pretty well. The results of these last experiments again underly the importance of the buffer and server allocation interaction process along with the service time variability.

Table 16: 10-node Complex Series Server Results

s^2	λ	**c**	θ	Z
0	9	(2,2,2,1,1,1,2,2,2,2)	8.999	17.066
1	9	(2,2,2,2,2,2,2,2,2,2)	9.000	20.000
2	9	(2,2,2,2,2,2,2,2,2,2)	9.000	20.000
0	12	(2,2,2,2,2,2,2,2,2,2)	12.000	20.000
1	12	(2,2,2,2,2,2,2,2,2,2)	11.999	20.000
2	12	(2,2,2,2,2,2,2,2,2,2)	11.999	20.000
0	15	(2,2,2,2,2,2,2,2,2,2)	14.999	20.981
1	15	(3,3,3,2,2,2,3,3,3,3)	14.999	27.003
2	15	(3,3,3,2,2,2,3,3,3,3)	14.999	27.089
0	21	(3,3,3,3,3,3,3,3,3,3)	20.999	30.147
1	21	(3,3,3,3,3,3,3,3,3,3)	20.997	32.909
2	21	(3,3,3,3,3,3,3,3,3,3)	20.999	30.019

Table 17: 21-node Complex Series Server Results

s^1	λ	**c**	θ	Z
1	7	(1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1)	6.998194	22.806
1	11	(2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2,2)	10.999986	42.014
1	17	(3,3,3,2,2,2,3,3,3,3,3,3,2,2,2,3,3,3,3,3)	16.996452	60.547
1	21	(3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3)	20.993780	69.219

6 Summary and Conclusions

We have presented a viable approach to the design of allocating an optimal number of servers to complex finite queueing networks with general service time distributions. We have shown that the objective function is essentially a piecewise linear convex function even under general service and demonstrated an implementation of an algorithm which is quite effective for a variety of arbitrary topologies, some with heavy blocking.

While we have not solved extremely large networks of size $N > 21$ at this time, because we wanted to make sure that the results were accurate for small lines, this extension to larger lines should not be a major problem and will be demonstrated in future papers.

References

[1] G. R. Bitran and D. Tirupati. Tradeoff curves, targeting and balancing in manufacturing queueing networks. *Operations Research*, 37(4):547–564, 1989.

[2] S. Borst, A. Mandelbaum, and M. I. Reiman. Dimensioning large call centers. *Operations Research*, 52(17-34):1, 2004.

[3] G. Brigham. On a congestion problem in an aircraft factory. *Journal of the Operations Research Society of America*, 3(4):412–428, 1955.

[4] Y. Dallery and K. E. Stecke. On the optimal allocation of servers and workloads in closed queueing networks. *Operations Research*, 38(4):694–703, 1990.

[5] A.G. De Kok and H. Tijms, 1985. "A Two-moment approximation for a Buffer Design Problem Requiring a Small Rejection Probability," *Performance Evaluation* **5**, 77-84.

[6] M. E. Dyer and L. G. Proll. On the validity of marginal analysis for allocating servers in $M/M/c$ queues. *Management Science*, 23(9):1019–1022, 1977.

[7] K. Futamura. The multiple server effect: Optimal allocation of servers to stations with different service-htime distributions in tandem queueing networks. *Annals of Operations Research*, 93(1-4):71–90, 2000.

[8] P. Glasserman and D. D. Yao. Monotonicity in generalized semi-Markov processes. *Mathematics of Operations Research*, 17(1):1–21, 1992.

[9] H. D. Gosavi and J. M. Smith. Asymptotic bounds of throughput in series-parallel queueing networks. *Computers & Operations Research*, 22(10):1057–1073, 1995.

[10] D. Gross and C. Harris. *Fundamentals of Queueing Theory*. John Wiley & Sons, New York, 1985.

[11] F. S. Hillier. Economic models for industrial waiting line problems. *Management Science*, 10(1):119–130, 1963.

[12] F. S. Hillier and G. J. Lieberman. *Introduction to Operations Research*. McGraw Hill Higher Education, 8th edition, 2005.

[13] F. S. Hillier and K. C. So. The assignment of extra servers to stations intandem queueing systems with small or no buffers. *Performance Evaluation*, 10:219–231, 1989.

[14] F. S. Hillier and K. C. So. On the simultaneous optimization of server and work allocations in production line systems with variable processing times. *Operations Research*, 44(3):435–443, 1996.

[15] D. M. Himmelblau. *Applied Nonlinear Programming*. McGraw-Hill Book Company, New York, 1972.

[16] L. Kerbache and J. MacGregor Smith, 1988. "Asymptotic Behaviour of the Expansion Method for Open Finite Queueing Networks." *Computers and Operations Research,* **15** (2), 157-169.

[17] L. Kerbache and J. M. Smith. The generalized expansion method for open finite queueing networks. *European Journal of Operational Research*, 32:448–461, 1987.

[18] L. Kleinrock. *Queueing Systems, Volume I: Theory*. John Wiley and Sons, first edition, 1975.

[19] P. Köchel, 2004. "Finite Queueing Systems-Structural Investiagions and Optimal Design." *Int. Jour. of Prod. Economics.***88**, 157-171.

[20] V. G. Kulkarni. *Modeling, Analysis, Design, and Control of Stochastic Systems*. Springer-Verlag, New York, 1982.

[21] Labetoulle, J. and Pujolle, G. "Isolation Method in a network of queues," *IEEE Trans.on Software Eng., Vol. SE-6,* 4, 373-380, 1980.

[22] C. Lemaréchal. The omnipresence of Lagrange. *Annals of Operations Research*, 153(1):9–27, 2007.

[23] M. J. Magazine and K. E. Stecke. Throughput for production lines with serial workstations and parallel service facilities. *Performance Evaluation*, 25:211–232, 1996.

[24] T. M. Manglesdorf. Waiting line theory applied to manufacturing problems. In E. H. Bowmand and R. B. Fetter, editors, *Analysis of Industrial Operations*, Illinois, 1959. Richard D. Irwin: Homewood.

[25] P. M. Morse. *Queues, Inventories and Maintenance: The Analysis of Operational Systems with Variable Demand and Supply*. John Wiley & Sons, John Wiley & Sons, Inc., New York, 1958.

[26] G. F. Newell. *Applications of Queueing Theory*. Chapman & Hall, London, 2nd edition, 1982.

[27] R. Rajan and R. Agrawal. Concavity of queueing systems with NBU service times. *Advances in Applied Probability*, 30(2):551–567, 1998.

[28] A. J. Rolfe. A note on marginal allocation in multiple-server service systems. *Management Science*, 17(9):656–658, 1971.

[29] J. G. Shanthikumar and D. D. Yao. Optimal server allocation in a system of multi-server stations. *Management Science*, 33(9):1173–1180, 1987.

[30] J. G. Shanthikumar and D. D. Yao. On server allocation in multiple center manufacturing systems. *Operations Research*, 36(2):333–342, 1988.

[31] J. M. Smith. $M/G/c/K$ blocking probability models and system performance. *Performance Evaluation*, 52(4):237–267, 2003.

[32] J. M. Smith. Optimal design and performance modelling of $M/G/1/K$ queueing systems. *Mathematical and Computer Modelling*, 39(9-10):1049–1081, 2004.

[33] Smith, J. MacGregor and Cruz, F.R. 2005. "The Buffer Allocation Problem for General Finite Buffer Queueing Networks," *IIE Transactions on Design and Manufacturing*

[34] Smith, J. MacGregor, Cruz, F.R., and Van Woensel, T. 2009. "Topological Network Design of General Finite Multi-Server Queueing Networks," acceprted in *European Journal of Operationsla Research.*

[35] D. Spinellis, C. T. Papadopoulos, and J. M. Smith. Large production line optimization using simulated annealing. *International Journal of Production Research*, 38(3):509–541, 2000.

[36] H. Tijms, 1986. *Stochastic Modeling and Analysis*. New York:Wiley.

[37] H. Tijms, 1992. "Heuristics for Finite-Buffer Queues," *Probability in the Engineering and Informational Sciences* **6**, 277-285.

[38] H. Tijms, 1994. **Stochastic Models: An Algorithmic Approach.** New York:Wiley

[39] R. R. Weber. On the marginal benefit of adding servers to $G/GI/m$ queues. *Management Science*, 26(9):946–951, 1980.

[40] L. M. Wein. Capacity allocation in generalized Jackson networks. *Operations Research Letters*, 8(3):143–146, 1989.

[41] J. A. White, J. W. Schmidt, and G. K. Bennett. *Analysis of Queueing Systems*. Academic Press, Inc., London, 1975.

[42] W. Whitt. Understanding the efficiency of multi-server service systems. *Management Science*, 38(5):708–723, 1992.