The Buffer Allocation Problem for General Finite Buffer Queueing Networks

J. MacGregor Smith*
e-mail: jmsmith@ecs.umass.edu

F. R. B. Cruz†
e-mail: fcruz@ufmg.br

May 18, 2004

Abstract — **The Buffer Allocation Problem (BAP) is a difficult stochastic, integer, nonlinear programming problem. In general, the objective function and constraints of the problem are not available in closed-form. An approximation formula for predicting the optimal buffer allocation is developed based upon a two-moment approximation formula involving the expressions for $M/M/1/K$ systems. The closed form expressions of $M/M/1/K$ and $M/G/1/K$ systems are utilized for the BAP in series, merge, and splitting topologies of finite buffer queueing networks. Extensive computational results demonstrate the efficacy of the approach.**

Keywords — Convexity,Blocking Probabilities, Buffer Allocation

1 Introduction

In almost all real physical systems, finite buffers exist. For example, in manufacturing systems, there is limited waiting room between workstations in assembly lines, material handling systems, and cellular manufacturing cells. In telecommunication systems, there are finite capacity telephone lines, computer networks, and capacitated ATM switches. Finally, in service systems such as facilities, there are limited circulation systems (elevators, stairways, and corridors), capacitated activity spaces, and finite storage areas. In all these system applications, it is crucial to compute the performance measures of these systems with finite buffers. It is also important to determine the optimal configuration of these systems with these finite buffers in mind. Any way in which this optimization process can be facilitated would be a great boost to the applications.

1.1 Motivation

One of the principal reasons for this paper is to develop closed form expressions for the blocking probability in general finite queueing networks. These closed form expressions would then be very useful in both analytical and simulation modelling of these general finite queueing systems. Finally, these closed form expressions can quickly generate buffer allocation solutions to complex topologies involving series, merge, and splitting topologies such as those in Figure 1.

1.2 Outline of Paper

In Section 2 an overview of the BAP is given and the focus is presented with which this paper is concerned. In Section 3, the mathematical model of the closed form expression for the optimal buffer size in $M/M/1/K$ systems is examined and the approximation formula for $M/G/1/K$ systems is developed. In Section 4, the Expansion Method is explained and its use in computing optimal buffer allocations for series, merge, and splitting topologies is shown in Section 5. Finally, Section 6 concludes the paper.

*Department of Mechanical and Industrial Engineering, University of Massachusetts Amherst, Massachusetts 01003

†Department of Statistics, Federal University of Minas Gerais, Belo Horizonte, MG, Brazil. Partially funded by the Brazilian Agencies CNPq, FAPEMIG, and PRPq-UFMG.

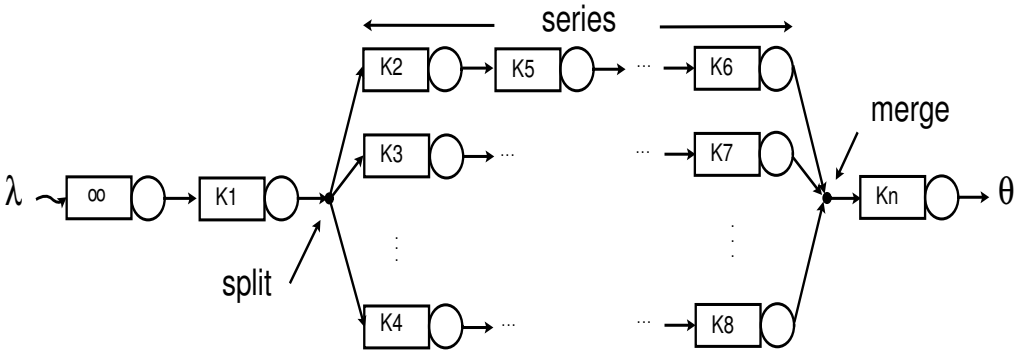


Figure 1: Example queueing network topology.

2 Buffer Allocation Problem (BAP)

For the sake of the argument, the BAP is concerned with how many buffer spaces must be provided so that the loss/delay blocking probability will be below a specific threshold. In particular, it can be argued that the BAP in its simplest form is to find the smallest integer $K \geq 0$ for which the blocking probability $p_K \leq \epsilon$ for any acceptable threshold level $\epsilon \in (0, 1)$. For the most part, we will assume that the system utilization $\rho < 1$ since if $\rho \geq 1$ there may not exist an optimal value of K [24]. This aspect will be demonstrated in the experimental part of this paper.

2.1 Problem Formulation

If we have more than one finite queue in a network of queues, the topology of the queueing network can become a difficult problem to model because of the interdependence of the blocking of one workstation upon another. For the most general case in a network of queues, the BAP is perhaps best formulated as a nonlinear multiple-objective programming problem where the decision variables are the integers. Not only is the BAP an $\mathcal{NP}$-hard combinatorial optimization problem [8], it is made all the more difficult from a practical point-of-view by the fact that the objective function is not obtainable in closed form to inter-relate the integer decision variables $\mathbf{x}$ and the performance measures such as throughput Θ, work-in-process $\mathbf{L}$, total buffers allocated $\sum_i x_i$, and other system performance measures such as system utilization ρ for any but the most trivial situations. Because we cannot obtain the problem in closed form, then derivative based methods would go through numerical computation of gradients and Hessians which is beyond the scope of this paper.

In this paper, we will consider the following type of optimization problem:

primal:

$$\min f(\mathbf{x}) = \sum_i c_i x_i \tag{1}$$

s.t.:

$$\Theta_i(\mathbf{x}) \geq \Theta_i^{\min}, \ \forall i, \tag{2}$$

$$x_i \geq 0, \ \forall i, \tag{3}$$

that minimizes the allocation cost $\sum_i c_i x_i$, constrained to provide minimum throughput $\Theta_i^{\min}$. In the above formulation $\Theta_i^{\min}$ is a threshold throughput value and x_i is the buffer decision variable. While the above formulation resembles a linear programming problem, the buffer allocation in the objective function is not explicitly modelled, since $\Theta_i(\mathbf{x})$ is a complex function of the arrival, service rates, traffic intensity, and other parameters and variables in the queueing system.

Another problem which is essentially the dual of the primal is given by:

dual:

$$\max g(\mathbf{w}) = \sum_i \Theta_i^{\min} w_i \tag{4}$$

s.t.:

$$\sum_j \Theta_j(\mathbf{x}^*) w_j \leq \sum_i c_i x_i^*, \tag{5}$$

$$w_j \geq 0, \ \forall j, \tag{6}$$

which finds the dual prices w_j such that the overall income is maximized subject to the constraint that no more money $\sum_j \Theta_j(\mathbf{x}^*) w_j$ is spent in the design $\sum_j c_i x_i^*$. This will result in the most economical buffer allocation $\mathbf{x}^*$ which will balance the design cost c_i and the per unit throughput w_j. We certainly could view w_j also as fixed prices instead and the threshold throughput $\Theta_j^{\min}$ as a decision variable. In such a case, the problem would be of maximizing the profit $\sum_j w_j \Theta_j^{\min}$ subject to a maximum budget $\sum_j w_j \Theta_j^{\min} \leq \sum_j w_j \Theta(\mathbf{x}^*) = B$. In both

cases again, one must remember that $\Theta(\mathbf{x})$ is not available in closed-form to relate $\mathbf{x}$ with the system parameters and other design variables.

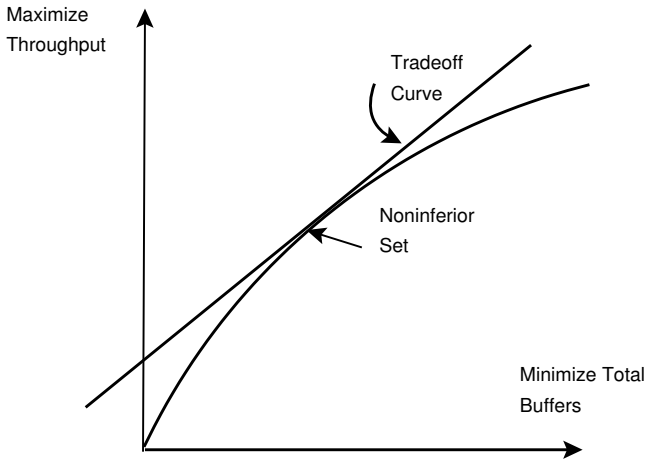


Figure 2: Pareto tradeoff curve.

Rather than select either the primal or the dual problem, what we shall do in this paper is generate an approximation to the non-inferior set of solutions for the two objectives which are maximizing throughput and minimizing the total buffer allocation. It is felt that examining the non-inferior set of solution through tradeoffs of the two objectives is both insightful and practical. Figure 2 gives one an indication of the approach we shall take.

2.2 Literature Review

The literature on the BAP can be generally broken down into one of four general methodological approaches: *Dynamic Programming, Search Methods, Metaheuristics* and *Simulation Methods*. The area of metaheuristics in the BAP is just beginning to bear fruit.

Dynamic Programming is a logical and very powerful approach to the BAP problem as it takes an essentially complex, non-closed form objective function and proceeds to allocate buffers to the stages of the network topology in the very natural way in which dynamic programming is designed to perform. The performance measures utilized in dynamic programming, however, may have to make certain restrictive assumptions in order to effectively compute the performance measure of the network topology.

The disadvantage with dynamic programming is in the exponential growth in the number of solution stages and states which thus requires an exponential amount of memory and consequently limits its applicability to small network topologies with few buffer alternatives.

Search Methods on the other hand tend to resolve the exponential explosion in the number of alternative buffer vectors by quickly sifting through the many alternative buffer vectors to discover those which yield close to optimal results.

Their main disadvantage is that often very restrictive assumptions must be made with the performance measures, and, even then, approximate performance models must be used in order to make the search process effective, thus, trading off accuracy in the performance measures for searching the buffer alternatives.

Metaheuristics are related to search methods but use a series of more general rules to search for feasible solutions to problems and eventually close in on the optimal solution. Typical solution techniques in this area include simulated annealing, tabu search, and genetic algorithms. Their chief advantage over traditional search methods is that they can jump over local optimal solutions in search of the global optimal ones. Their main disadvantage is that they often do not utilize the special structure of the problem which may be available in the objective function and constraints to guide their search and thus, have to "tune-up" to produce solutions to a specific problem type.

Simulation Methods on the other hand represent an attempt to capture the performance measures for a wide range of robust assumptions (probability distributions) which allow it to be a very general method indeed.

However, its generality may make the search process for the optimal vector either impossible or severely limiting because the computation times become prohibitive. Actually, it is the very nature of the inter-dependencies in the multi-variable optimization process of the BAP which creates difficulties in the use of simulation methods.

Figure 3 illustrates a sampling of some of the approaches to the BAP over the years, with appropriately cited references. If some references have been left out in this diagram, it is certainly not for exclusionary reasons, but for the sake of brevity and conciseness.

For a recent compendium of some of the newest literature on this topic, one is encouraged to review the new material in Smith, Gershwin, and Papadopoulos [34].

3 Mathematical Models

In this section of the paper, we will develop our closed form expression for $M/M/1/K$ and $M/G/1/K$ systems. We will first develop the closed form expression for relating the optimal buffer size as a function of the blocking probability and the system utilization. Then we will develop an approximation for the blocking probability in $M/G/1/K$ systems that builds upon the formula for the $M/M/1/K$ system. This is claimed to be a novel approach since most approximations are based upon infinite queueing systems rather than finite ones. First, some convenient notation to guide the reader.

3.1 Notation

The following section presents some of the notation we need for the paper.

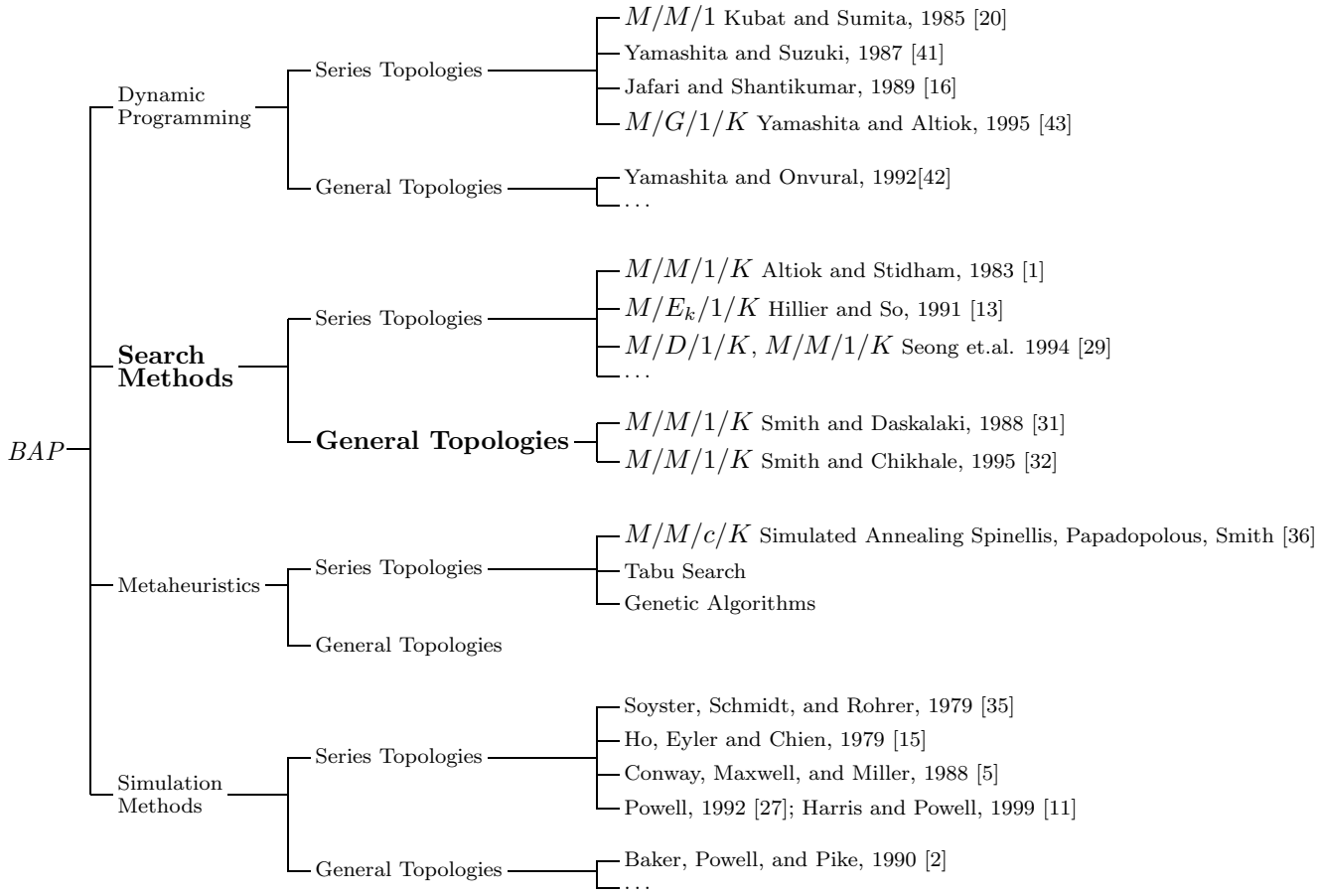


Figure 3: Morphological diagram of BAP approaches.

a^2 := squared coefficient of variation of the arrival process.

λ_i := external Poisson arrival rate at node i.

μ_j := exponential mean service rate at node j.

$\epsilon \in (0,1)$:= threshold for the blocking probability.

$\rho_i = \lambda_i/\mu_i$:= the traffic intensity at node i.

K_j := buffer capacity at node j including those in service.

p_K := blocking probability of finite queue of size K.

s^2 := squared coefficient of variation of the service process.

Θ := mean throughput rate.

3.2 $M/M/1/K$ Systems

The blocking probability for an $M/M/1/K$ system with $\rho < 1$ is well-known:

$$\frac{(1-\rho)\rho^K}{1-\rho^{K+1}} = p_K$$

If we relax the integrality of K, we can express K in terms of ρ and p_K and arrive at a closed-form expression for the buffer size which is the largest integer as follows:

$$K = \left\lceil \frac{\ln(\frac{p_K}{1-\rho+p_K\rho})}{\ln(\rho)} \right\rceil$$

This is a very useful formula for determining the buffer size K of an individual queue.

What also is interesting about the formula is illustrated in Figure 4 which shows the smooth monotonic nature of the function over the range of ρ and p_K. The general range of the function for all values of ρ, p_K is in Figure 4 and the specific function values for K in the range of most practical interest is in Figure 4 on the bottom.

In a previous paper [33], we showed that the closed form expression for the optimal buffers was not a convex function over the entire range of the variation of p_K and ρ. This implies that only local optimal solutions are possible. This local optimality feature is reflected in the results we are able to achieve in this paper as we shall demonstrate in Section 5.

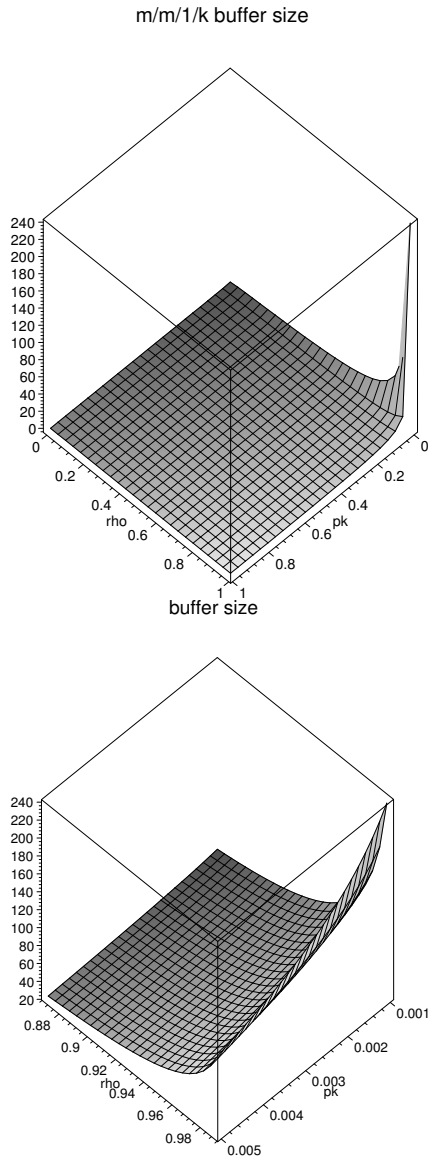


Figure 4: $M/M/1/K$ function properties.

3.3 Non-Markovian Systems

In principle, for the $M/G/1/K$ system, one could develop the blocking probability for fixed K and all the other probabilities, but this would be tedious and not as useful when it comes to modelling the design of queueing networks with varying buffer sizes, see Chapter 5 §5.1.8 in Gross and Harris [10].

3.4 Different approximations

There are numerous other approximations for the blocking probability possible for $M/G/1/K$ systems. One survey article by Springer and Makens [37] analyzes in some detail five approximation formula and concludes that the formula by Gelenbe is the most accurate and robust. In the following analysis, we shall examine this closed-form approximation formula, and, as in the previous paper [33], examine the monotonic nature of the blocking probability. There are other buffer allocation approximations [23, 24, 38, 44] that could be analyzed, and we will include in particular the approximations of Tijms and Kimura, and compare then with Gelenbe's in Subsection 3.7 of the paper. The Tijms and Kimura's approximations are not closed-form but are very accurate.

3.5 Gelenbe's Formula

Gelenbe's formula is based on approximating the discrete queueing process as a continuous diffusion process. The blocking probability from Gelenbe's equation with squared arrival and service process coefficient of variation is given by the following equation [9] where a^2 and s^2 are respectively the squared coefficient of variations of the arrival and service processes:

$$\frac{\lambda\,(\mu-\lambda)\,e^{-2\,\frac{(\mu-\lambda)(k-1)}{\lambda\,a^2+\mu s^2}}}{\left(\mu^2-\lambda^2 e^{-2\,\frac{(\mu-\lambda)(k-1)}{\lambda\,a^2+\mu s^2}}\right)}=p_K.$$

The closed-form expression derivable for K from Gelenbe's formula is the following:

$$K=1/2\frac{2\lambda-2\mu+\ln(\frac{p_K\mu^2}{\lambda(-\lambda+\mu+p_K\lambda)})\lambda a^2+\ln(\frac{p_K\mu^2}{\lambda(-\lambda+\mu+p_K\lambda)})\mu s^2}{(\lambda-\mu)}.$$

If we assume we have a Markovian system, then $a^2=s^2=1$ and we get the function whose graph is illustrated at the top in Figure 5. Further, if we set $a^2=1, s^2=0$ we get a monotonic function that is indicated on the bottom in Figure 5. The Markovian approximation is very accurate, while the $M/D/1/K$ is not as accurate an approximation, even though the order of magnitude reduction in the buffer size which is roughly 2 is quite accurate. This assessment of Gelenbe's formula essentially agrees with the assessment of Springer and Makens [37].

The smooth monotonic nature of the $M/G/1/K$ approximation is an important property that should be shared by all approximations.

3.6 M/G/1/K Approximation Development

Let us now develop our closed form expression for the blocking probability which is based on Kimura's formula for the buffer size. In order to describe Tijm's and Kimura's approximations, an additional bit of notation is needed at this point in the paper. Let us define $K_\epsilon(M)$ as the Markovian expression for the optimal buffer size as a function of the blocking probability and the threshold, $p_K(\epsilon)$. Also, $K_\epsilon(D)$ is the expression of the optimal buffer size as a function of a deterministic service process.

3.7 Two-moment Approximation Schema

Tijm's two-moment approximation [6, 38, 40] relies on a weighted combination of an exact (if available) expression of the $M/M/1/K$ blocking probability as well as the blocking probability of the $M/D/1/K$ formula:

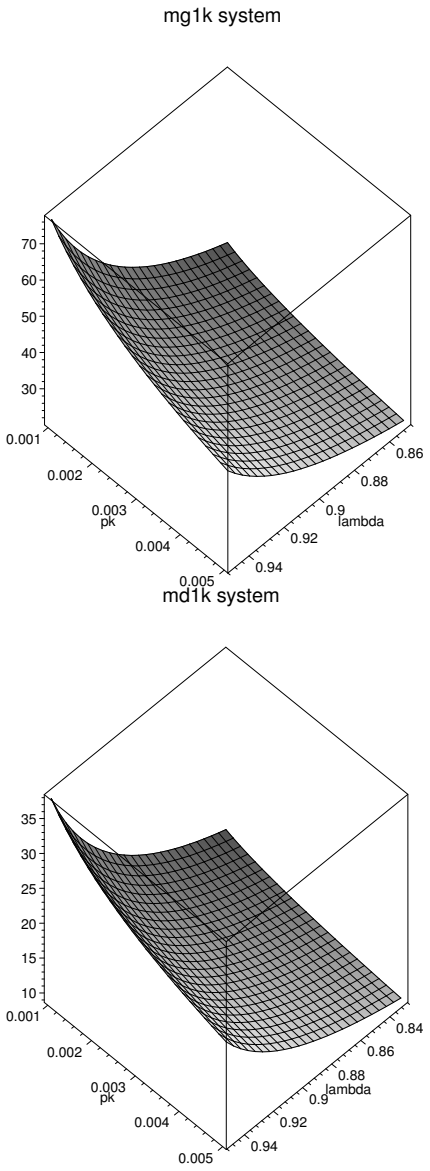


Figure 5: Gelenbe's $M/M/1/K$ on top and $M/D/1/K$ expression on bottom.

$$K_\epsilon^T(s^2) = s^2 K_\epsilon(M) + (1-s^2)K_\epsilon(D).$$

Of course, if exact expressions are available for both formulas, then Tijm's approximation is exact for the two extreme cases. His approximation has been shown to be very good, and we shall corroborate his results.

Kimura, on the other hand, has also a two-moment approximation that turns out to be a little simpler and is the one we shall build upon since it utilizes Markovian approximations as its basis. His expression is:

$$\tilde{K}_\epsilon(s^2) = K_\epsilon(M) + \mathrm{NINT}\Big(1/2(s^2-1)\sqrt{\rho}K_\epsilon(M)\Big).$$

where NINT is the nearest integer. An important observation about Kimura's expression is that his formula estimates the pure buffer without the space for the customers in service, while Tijm's formula is more general and includes those in service.

Rather than simply use Kimura's expression, we shall substitute the formula for the optimal buffer size of the $M/M/1/K$ system and relax the integrality of K to yield the following closed-form expression for the optimal buffer size for $M/G/1/K$ systems.

Let us repeat for clarity the formula for the optimal buffer size for the $M/M/1/K$ formula:

$$K = \frac{\ln(\frac{p_K}{1-\rho+p_K\,\rho})}{\ln(\rho)}.$$

Here, we add Kimura's expression for the approximation of the optimal buffer size based on his two-moment approximation formula. It is important to note here that we subtract the space for the server in each of the two terms of Kimura's formula in order to estimate the true buffer space in the queue:

$$\Big(\ln(\frac{p_K}{1-\rho+p_K\,\rho})\,(\ln(\rho))^{-1}-1\Big)$$

$$+1/2\,\left(s^2-1\right)\sqrt{\rho}\left(\ln(\frac{p_K}{1-\rho+p_K\rho})(\ln(\rho))^{-1}-1\right).$$

Now we factor the terms of the above expression to give the following simplified expression for the optimal buffer size in $M/G/1/K$ formulas:

$$\frac{\Big(\ln(\frac{p_K}{1-\rho+p_K\,\rho})-\ln(\rho)\Big)\left(2+\sqrt{\rho}s^2-\sqrt{\rho}\right)}{2\ln(\rho)}.$$

If $s^2=1$, then the formula yields the same expression as for the $M/M/1/K$ formula, when we subtract the space for the server.

As an added side benefit for developing the closed form expression for the optimal buffer, if we invert the last expression we can obtain the blocking probability for the $M/G/1/K$ system as:

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

In order to test the efficacy of the blocking probability formula a small set of experiments are performed comparing our approximation with Gelenbe's and the $M/M/1/K$ model and exact results from Seelen, Tijms, and Van Hoorn [28] for an $M/G/1/K$ model with $s^2=0.50$, see Tables 1–5 and the resulting Figures 6–8. The results from Table 1 are due to the fact that this first experiment is just an Erlang loss model, so all the approaches should yield the same results which they indeed do. The results of our new formula in the other tables when compared to Gelenbe's are surprisingly accurate, especially in the $\rho<1$ values and in relation to the growth of the blocking probability values over the range of the parameters. Gelenbe's formula does better in $\rho \geq 1.50$, however, this is beyond the range of ρ we feel is most appropriate to consider.

Table 1: Comparison of p_K formulas.

	K=1, s^2=0.50			
ρ	new	$M/M/1/K$	gelenbe	exact
0.10	0.090909	0.090909	0.090909	0.090909
0.20	0.166667	0.166667	0.166667	0.166667
0.30	0.230769	0.230769	0.230769	0.230769
0.40	0.285714	0.285714	0.285714	0.285714
0.50	0.333333	0.333333	0.333333	0.333333
0.60	0.375000	0.375000	0.375000	0.375000
0.70	0.411765	0.411765	0.411765	0.411765
0.80	0.444444	0.444444	0.444444	0.444444
0.90	0.473684	0.473684	0.473684	0.473684
1.10	0.523810	0.523810	0.523810	0.523810
1.20	0.545455	0.545455	0.545455	0.545455
1.30	0.565217	0.565217	0.565217	0.565217
1.50	0.600000	0.600000	0.600000	0.600000
1.70	0.629630	0.629630	0.629630	0.629630
1.90	0.655172	0.655172	0.655172	0.655172

Table 2: Comparison of p_K formulas.

	K=2, s^2=0.50			
ρ	new	$M/M/1/K$	gelenbe	exact
0.10	0.00739	0.009009	0.00053	0.00698
0.20	0.02630	0.032258	0.00458	0.02576
0.30	0.05323	0.064748	0.01659	0.05316
0.40	0.08543	0.102564	0.03886	0.08629
0.50	0.12072	0.142857	0.07055	0.12281
0.60	0.15746	0.183673	0.10894	0.16087
0.70	0.19446	0.223744	0.15102	0.19917
0.80	0.23089	0.262295	0.19425	0.23676
0.90	0.26619	0.298893	0.23687	0.27306
1.10	0.33214	0.365559	0.31637	0.34047
1.20	0.36251	0.395604	0.35238	0.37132
1.30	0.39111	0.423559	0.38577	0.40032
1.50	0.44312	0.473684	0.44503	0.45251
1.70	0.48873	0.516995	0.49537	0.49804
2.00	0.54678	0.571429	0.55728	0.55556

Table 3: Comparison of p_K formulas.

	K=3, s^2=0.50			
ρ	new	$M/M/1/K$	gelenbe	exact
0.10	0.00061	0.000900	0.00000	0.00049
0.20	0.00427	0.006410	0.00013	0.00378
0.30	0.01297	0.019054	0.00129	0.01210
0.40	0.02772	0.039409	0.00600	0.02677
0.50	0.04867	0.066667	0.01768	0.04811
0.60	0.07523	0.099265	0.03865	0.07551
0.70	0.10635	0.135413	0.06893	0.10777
0.80	0.14074	0.173442	0.10650	0.14342
0.90	0.17711	0.211980	0.14852	0.18098
1.10	0.25132	0.286792	0.23583	0.25699
1.20	0.28750	0.321908	0.27776	0.29369
1.30	0.32232	0.355099	0.31735	0.32880
1.50	0.38675	0.415385	0.38838	0.39323
1.70	0.443553	0.467771	0.44870	0.44955
2.00	0.51508	0.533333	0.52206	0.52000

Table 4: Comparison of p_K formulas.

	K=6, s^2=0.50			
ρ	new	$M/M/1/K$	gelenbe	exact
0.10	0.00000	0.000001	0.00000	0.00000
0.20	0.00002	0.000051	0.00000	0.00001
0.30	0.00020	0.000510	0.00000	0.00013
0.40	0.00104	0.002462	0.00002	0.00081
0.50	0.00373	0.007874	0.00032	0.00322
0.60	0.01026	0.019200	0.00218	0.00949
0.70	0.02322	0.038462	0.00912	0.02245
0.80	0.04484	0.066342	0.02633	0.04456
0.90	0.07596	0.101867	0.05711	0.07653
1.10	0.16039	0.186732	0.15002	0.16255
1.20	0.20778	0.231187	0.20198	0.21023
1.30	0.25484	0.274518	0.25232	0.25723
1.50	0.34150	0.354055	0.34207	0.34329
1.70	0.41457	0.422050	0.41574	0.41569
2.00	0.50059	0.503937	0.50147	0.50108

As one final set of experiments, Figure 8 illustrates the performance of the new probability model when the squared coefficient of variation $s^2 = 2$ and the buffer sizes range from $K = 2, 3, 6, 11$. For the buffer size $K = 1$ and $s^2 = 1/2$, all models have the same results as $K = 1$ and $s^2 = 1/2$. Again, as can be seen in Figure 8, the new model does remarkably well as compared with Gelenbe's formula, the $M/M/1/K$ lower bound, and simulation. Since we had no exact blocking probability results, we simulated the system with a Gamma random variable and 200,000 simulated time units to approach steady state. ARENA was the simulation language employed.

Given confidence in the accuracy of the blocking probability formula, we want to embed it in an optimization process to search for the optimal buffer sizes. Let us now discuss the algorithmic process for doing this.

4 Algorithm

The Expansion Method is a robust and effective approximation technique developed by Kerbache and Smith [18]. As described in previous papers, this method is characterized as a combination of repeated trials and node-by-node decomposition solution procedures. Methodologies for computing performance measures for a finite queueing network use primarily the following two kinds of blocking:

1. *Type I:* The upstream node i gets blocked if the *service on a customer is completed* but it cannot move downstream due to the queue at the downstream node j being full. This is sometimes referred to as blocking after service (BAS) [25].

2. *Type II:* The upstream node is blocked when the downstream node becomes saturated and service must be suspended on the upstream customer regardless of whether service is completed or not. This is sometimes referred to as blocking before service (BBS) [25].

The Expansion Method uses *Type I* blocking, which is prevalent in most production and manufacturing, transportation and other similar systems.

Consider a single node with finite capacity K (including service). This node essentially oscillates between two states — the saturated phase and the unsaturated phase. In the unsaturated phase, node j has at most $K-1$ customers (in service or in the queue). On the other hand,

Table 5: Comparison of p_K formulas.

	K=11, s^2=0.50			
ρ	new	$M/M/1/K$	gelenbe	exact
0.10	0.00000	0.000000	0.00000	0.00000
0.20	0.00000	0.000000	0.00000	0.00000
0.30	0.00000	0.000001	0.00000	0.00000
0.40	0.00001	0.000025	0.00000	0.00000
0.50	0.00006	0.000244	0.00000	0.00004
0.60	0.00043	0.001454	0.00002	0.00034
0.70	0.00232	0.006015	0.00038	0.00208
0.80	0.00937	0.018448	0.00356	0.00901
0.90	0.02840	0.043732	0.01843	0.02828
1.10	0.11762	0.133421	0.11194	0.11848
1.20	0.17663	0.187721	0.17434	0.17743
1.30	0.23430	0.241118	0.23365	0.23486
1.50	0.33376	0.335922	0.33382	0.33394
1.70	0.41182	0.412473	0.41187	0.41187
2.00	0.50000	0.500122	0.50002	0.50001

when the node is saturated no more customers can join the queue. Refer to Figure 9 for a graphical representation of the two scenarios.

The Expansion Method has the following three stages:

- *Stage I: Network Reconfiguration.*
- *Stage II: Parameter Estimation.*
- *Stage III: Feedback Elimination.*

The following additional notation defined by Kerbache and Smith [18, 17] shall be used in further discussion regarding this methodology :

$h :=$ the holding node established in the Expansion Method.

$\tilde{\lambda_j}$:= effective arrival rate to node j.

$\tilde{\mu_j}$:= effective service rate at node j due to blocking.

p'_K:=feedback blocking probability in the Expansion Method.

4.1 Stage I: Network Reconfiguration

Using the concept of two phases at node j, an artificial node h is added for each finite node in the network to register blocked customers. Figure 9 shows the additional delay, caused to customers trying to join the queue at node j when it is full, with probability p_K. The customers successfully join queue j with a probability $(1 - p_K)$. Introduction of an artificial node also dictates the addition of new arcs with p_K and $(1 - p_K)$ as the routing probabilities.

The blocked customer proceeds to the finite queue with probability $(1 - p'_K)$ once again after incurring a delay at the artificial node. If the queue is still full, it is rerouted with probability p'_K to the artificial node where it incurs another delay. This process continues till it finds a space in the finite queue. A feedback arc is used to model the repeated delays. The artificial node is modelled as an $M/M/\infty$ queue. The infinite number of servers is used simply to serve the blocked customer a delay time without queueing.

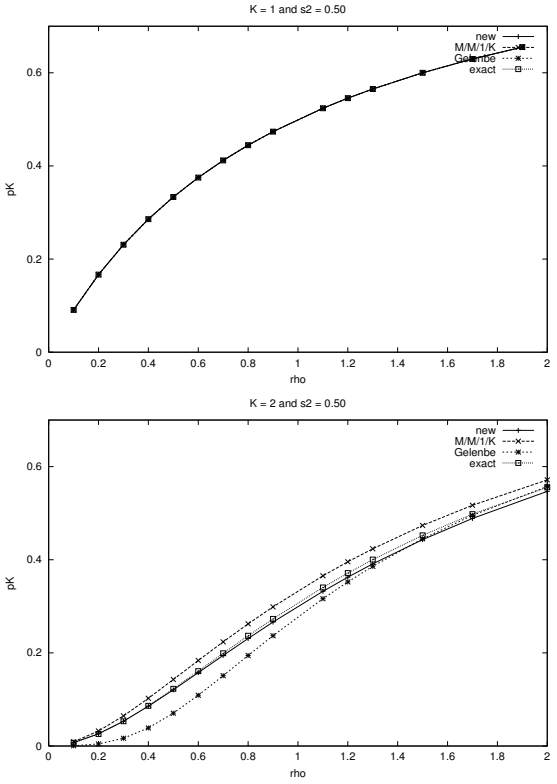


Figure 6: p_K comparisons $K = 1, 2$, $s^2 = 1/2$.

4.2 Stage II: Parameter Estimation

This stage essentially estimates the parameters p_K, p'_K and μ_h utilizing known results for the $M/M/c/K$ model.

- p_K: Utilizing our analytical results for the $M/G/1/K$ model provides the following expression for p_K:

$$p_K = \frac{\rho^{\left(\frac{2+\sqrt{\rho}s^2-\sqrt{\rho}+2K}{2+\sqrt{\rho}s^2-\sqrt{\rho}}\right)}(-1+\rho)}{\left(\rho^{\left(2\frac{2+\sqrt{\rho}s^2-\sqrt{\rho}+K}{2+\sqrt{\rho}s^2-\sqrt{\rho}}\right)}-1\right)}.$$

- p'_K: Since there is no closed form solution for this quantity an approximation is used given by Labetoulle and Pujolle obtained using diffusion techniques [22]:

$$p'_K = \left[\frac{\mu_j+\mu_h}{\mu_h} - \frac{\lambda[(r_2^K - r_1^K) - (r_2^{K-1} - r_1^{K-1})]}{\mu_h[(r_2^{K+1} - r_1^{K+1}) - (r_2^K - r_1^K)]}\right]^{-1},$$

where r_1 and r_2 are the roots to the polynomial:

$$\lambda - (\lambda + \mu_h + \mu_j)x + \mu_h x^2 = 0,$$

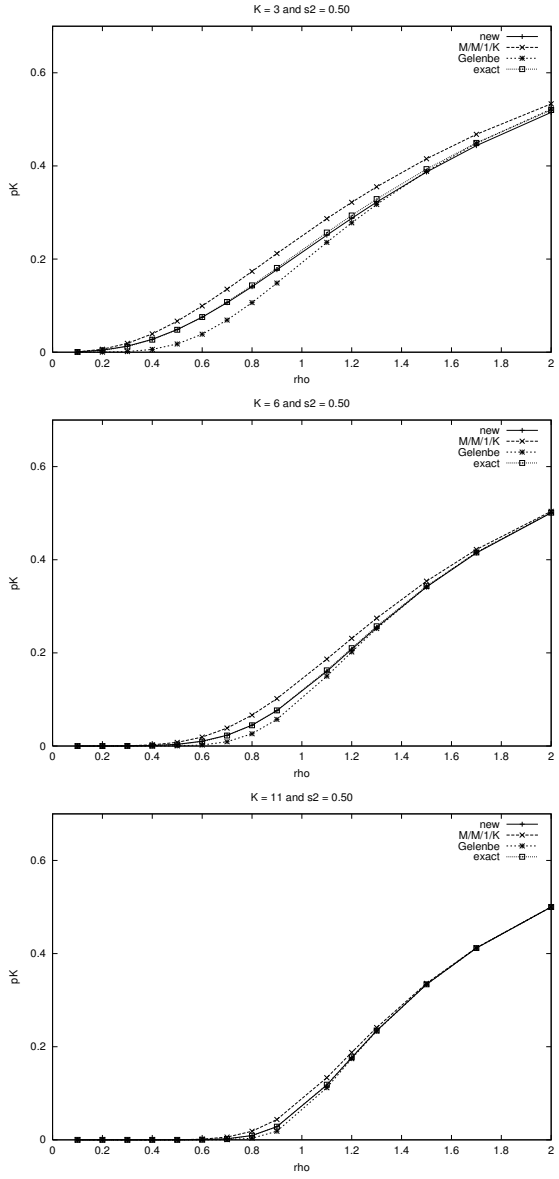


Figure 7: p_K comparisons $K = 3, 6, 11$, $s^2 = 1/2$.

while, $\lambda = \lambda_j - \lambda_h(1-p'_K)$ and λ_j and λ_h are the actual arrival rates to the finite and artificial holding nodes respectively. Labetoulle and Pujolle [22] illustrate in their paper a comparison of their method for computing p'_K with an Erlang service system and an hyperexponential system and it is shown that the calculation for p'_K is very reasonable for these general service systems. Given these results, we felt comfortable in applying p'_K for the general service situation.

In fact, the arrival rate to the finite node λ_j is given by:

$$\lambda_j = \tilde{\lambda_i}(1 - p_K) = \tilde{\lambda_i} - \lambda_h.$$

Let us examine the following argument to determine the service time at the artificial node. If an arriving customer is blocked, the queue is full and thus a

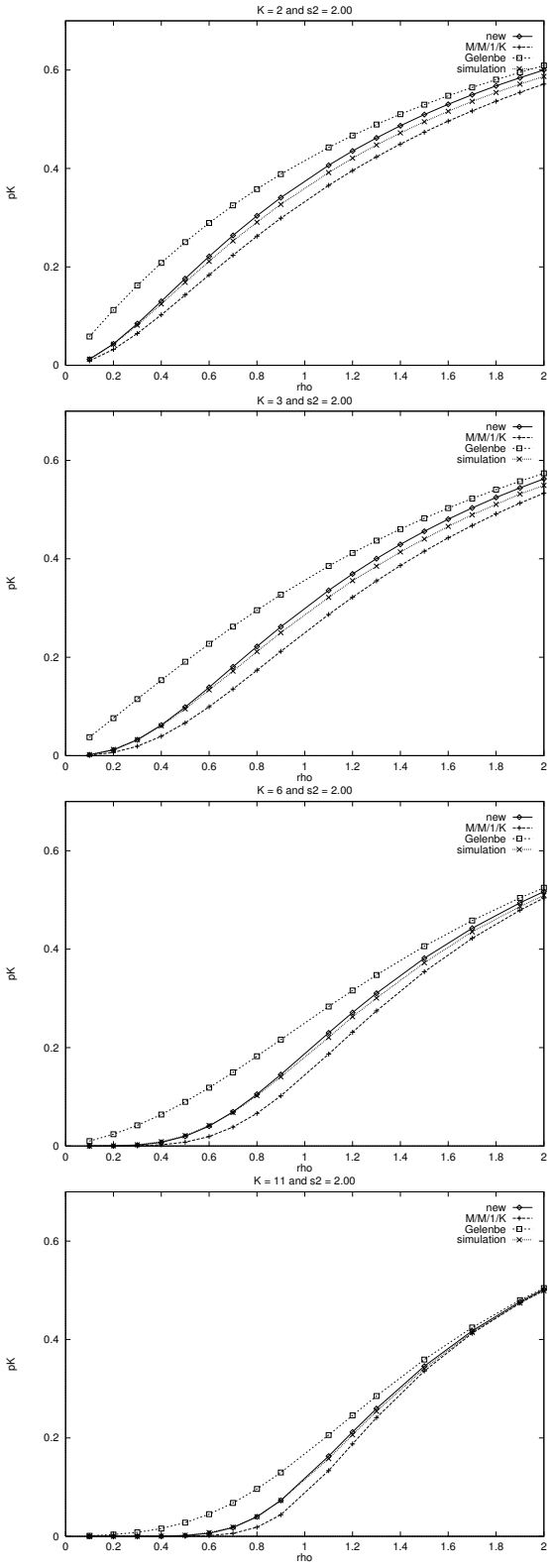


Figure 8: p_K comparisons $K = 2, 3, 6, 11$, $s^2 = 2$.

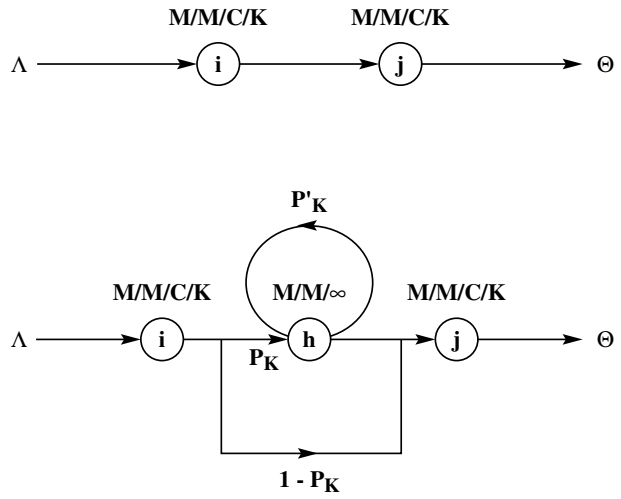


Figure 9: Type I blocking in finite queues.

customer is being serviced, so the arriving customer to the holding node has to remain in service at the artificial holding node for the remaining service time interval of the customer in service. The delay distribution of a blocked customer at the holding node has the same distribution as the remaining service time of the customer being serviced at the node doing the blocking. Using renewal theory, one can show that the remaining service time distribution has the following rate μ_h:

$$\mu_h = \frac{2\mu_j}{1+\sigma_j^2\mu_j^2},$$

where, σ_j^2 is the service time variance given by Kleinrock [19]. Notice that if the service time distribution at the finite queue doing the blocking is exponential with rate μ_j, then:

$$\mu_h = \mu_j,$$

i.e. the service time at the artificial node is also exponentially distributed with rate μ_j. When the service time of the blocking node is not exponential, then μ_h will be affected by σ_j^2.

4.3 Stage III: Feedback Elimination

Due to the feedback loop around the holding node, there are strong dependencies in the arrival processes. Elimination of these dependencies requires reconfiguration of the holding node which is accomplished by recomputing the service time at the node and removing the feedback arc. The new service rate is given by

$$\mu_h' = (1-p_K')\mu_h.$$

The probabilities of being in any of the two phases (saturated or unsaturated) are p_K and $(1-p_K)$. The mean service time at a node i preceding the finite node is μ_i^{-1} when in the unsaturated phase and $(\mu_i^{-1}+\mu_h'^{-1})$ in the saturated phase. Thus, on an average, the mean service time at the node i preceding a finite node is given by

$$\tilde{\mu}_i^{-1} = \mu_i^{-1} + p_K\mu_h^{-1}.$$

Similar equations can be established with respect to each of the finite nodes. Ultimately, we have simultaneous non-linear equations in variables p_K, p_K', μ_h^{-1} along with auxiliary variables such as μ_j and $\tilde{\lambda_i}$. Solving these equations simultaneously we can compute all the performance measures of the network:

$$\lambda = \lambda_j - \lambda_h(1-p_K') \quad (7)$$

$$\lambda_j = \tilde{\lambda_i}(1-p_K) \quad (8)$$

$$\lambda_j = \tilde{\lambda_i} - \lambda_h \quad (9)$$

$$p_K' = \left[\frac{\mu_j+\mu_h}{\mu_h} - \frac{\lambda[(r_2^K - r_1^K) - (r_2^{K-1} - r_1^{K-1})]}{\mu_h[(r_2^{K+1} - r_1^{K+1}) - (r_2^K - r_1^K)]}\right]^{-1} \quad (10)$$

$$z = (\lambda+2\mu_h)^2 - 4\lambda\mu_h \quad (11)$$

$$r_1 = \frac{[(\lambda+2\mu_h) - z^{\frac{1}{2}}]}{2\mu_h} \quad (12)$$

$$r_2 = \frac{[(\lambda+2\mu_h) + z^{\frac{1}{2}}]}{2\mu_h} \quad (13)$$

$$p_K = \frac{\rho^{\left(\frac{2+\sqrt{\rho}s^2-\sqrt{\rho}+2\,K}{2+\sqrt{\rho}s^2-\sqrt{\rho}}\right)}(-1+\rho)}{\left(\rho^{\left(2\,\frac{2+\sqrt{\rho}s^2-\sqrt{\rho}+K}{2+\sqrt{\rho}s^2-\sqrt{\rho}}\right)}-1\right)} \quad (14)$$

Equations 7 to 10 are related to the arrivals and feedback in the *holding* node. The equations 11 to 13 are used for solving equation 10 with z used as a dummy parameter for simplicity of the solution. Lastly, equation 14 gives the approximation to the blocking probability for the $M/G/1/K$ queue. Hence, we essentially have five equations to solve, *viz.* 7 to 10 and 14.

To recapitulate, we first expand the network; followed by approximation of the routing probabilities, due to blocking, and the service delay in the holding node and finally the feedback arc at the holding node is eliminated. Once these three stages are complete, we have an expanded network which can then be used to compute the performance measures for the original network. As a decomposition technique this approach allows successive addition of a holding node for every finite node, estimation of the parameters and subsequent elimination of the holding node.

Figure 10 and 11 illustrate the process of expanding the network topologies for the merge and split topologies in the Expansion Method. An important point about this process is that we do not physically modify the networks, only represent the expansion process through the software.

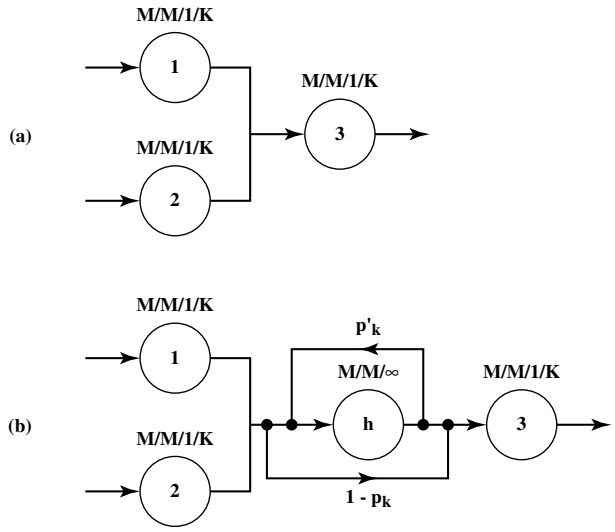


Figure 10: Merge topologies.

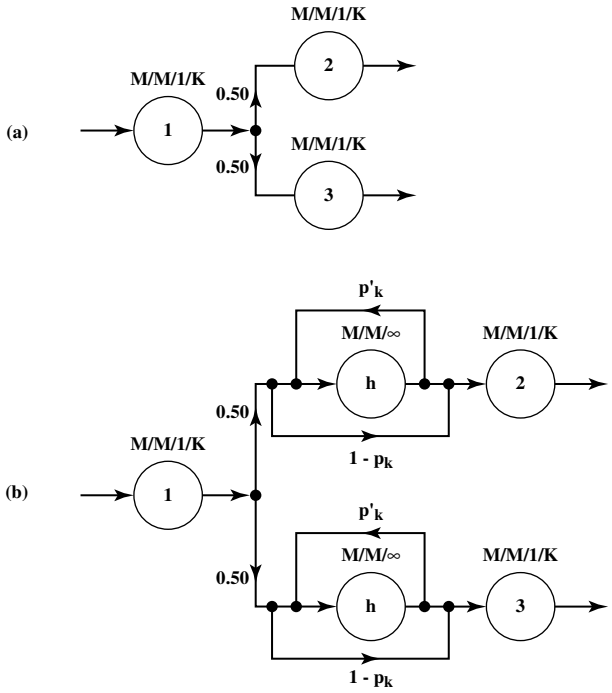


Figure 11: Split topologies.

4.4 Optimization Problem

The primal optimization problem with $M/M/1/K$ and $M/G/1/K$ systems that will be examined here is essentially the following:

$$\min Z = \sum_{i=1}^{N} x_i \tag{15}$$

s.t.:

$$\Theta(\mathbf{x}) \geq \Theta^{\min}, \tag{16}$$

$$x_i \in \{0, 1, 2, \ldots\}, \ \forall i, \tag{17}$$

in which the x_i is the buffer space of the ith queue system.

One way to incorporate the throughput constraint is through a penalty function approach. Defining a dual variable α, the penalized problem is the following:

$$\min Z = \sum_{i=1}^{N} x_i - \alpha\Big(\Theta(\mathbf{x}) - \Theta^{\min}\Big) \tag{18}$$

s.t.:

$$x_i \in \{0, 1, 2, \ldots\}, \ \forall i, \tag{19}$$

$$\alpha \geq 0. \tag{20}$$

$\Theta^{\min}$ can be pre-specified and then serve as the input λ to an approximate performance measure program such as the Expansion Method program [17], that will compute the corresponding throughput. In the particular formulation of the problem the $x_i, \forall i$, become the decision variables under optimization control. While these are essentially integer variables, they can be reasonably approximated by round off from the nonlinear programming solver.

As we shall demonstrate, solving the above problem will afford us a method to generate an approximation to the non-inferior set of solutions for the two objectives. In order to couple the optimization problem with the Expansion Method, Powell's algorithm is used to search for the optimal buffer vector(s) while the Expansion Method computes the performance measure of throughput. In the next subsection, we describe briefly Powell's algorithm.

4.5 Powell's Algorithm

Powell's method, as presented in Himmelblau [14], locates the minimum of $f(\mathbf{x})$ of a non-linear function by successive unidimensional searches from an initial starting point $\mathbf{x}^{(k)}$ along a set of conjugate directions. These conjugate directions are generated within the procedure itself. Powell's method is based on the idea that if a minimum of a non-linear function $f(\mathbf{x})$ is found along p conjugate directions in a stage of the search, and an appropriate step is made in each direction, the overall step from the beginning to the p^{th} step is conjugate to all of the p subdirections of the search.

Figure 12 describes Powell's unconstrained optimization algorithm used in our experiments. Implementations of the algorithm in FORTRAN and C are common.

We have had remarkable success in the past with coupling Powell's algorithm and the Expansion Method. Let us examine how well it does with the addition of the new general blocking probability formula.

5 Experimental Results

In the following experimental results, we have analyzed series, merge, and splitting topologies of $N = 3, 5, 7, 10$

algorithm
input $G(N, A, P)$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, *and* $\mathbf{x}^{(0)}$
/* *chose a set of l.i. search directions* */
choose $\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(n)}$
$\mathbf{x}^{(\text{opt})} \leftarrow \mathbf{x}^{(0)}$
repeat
$\mathbf{x}^{(1)} \leftarrow \mathbf{x}^{(\text{opt})}$
for $i = 1$ **to** n **do**
/* *perform unidimensional search* */
/* *compute* $f(\bullet)$ *by the Expansion Method* */
$\mathbf{x}^{(i+1)} \leftarrow \left\{\mathbf{x}^* | f(\mathbf{x}^*) = \min_{\gamma \in \mathcal{R}} f\left(\mathbf{x}^{(i)} + \gamma \mathbf{d}^{(i)}\right)\right\}$
end for
$\mathbf{x}^{(n+2)} \leftarrow 2\mathbf{x}^{(n+1)} - \mathbf{x}^{(1)}$
if $f(\mathbf{x}^{(n+2)}) \geq f(\mathbf{x}^{(1)})$ **then**
$\mathbf{x}^{(\text{opt})} \leftarrow \mathbf{x}^{(n+1)}$
else
$\mathbf{x}^{(\text{opt})} \leftarrow \left\{\mathbf{x}^* | f(\mathbf{x}^*) = \min_{\gamma \in \mathcal{R}} f\left(\mathbf{x}^{(n+1)} + \gamma(\mathbf{x}^{(n+1)} - \mathbf{x}^{(1)})\right)\right\}$
choose new $\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(n)}$
end if
until $\|\mathbf{x}^{(\text{opt})} - \mathbf{x}^{(1)}\| < \epsilon$
print $\mathbf{x}^{(\text{opt})}$
end algorithm

Figure 12: Powell's algorithm.

nodes. We are interested in the patterns of the buffer allocation as we vary the arrival rate and the squared coefficient of variation of the service processes. We will also examine at the end of this section some larger networks, a 9-node and a 16-node series-merge-split topology, to show the effectiveness and reasonableness of our approach. All the experiments were carried out on a Dell Dimension 266Mhz machine with a Windows NT 4.0 operating system with a Compaq Visual Fortran code, version 6.5.

We shall also be interested in comparing whether the series, merge or splitting topologies is a better layout alternative to achieve the given levels of performance.

As we shall demonstrate, the squared coefficient of variation of the service time process s^2 is critical in the buffer allocation patterns and in the generation of the smallest set of buffers and maximizing the throughput. This critical nature of the coefficient of variation is as expected [13].

In all the other analytical experiments which follow, the optimization process is highly dependent on the starting solution. This is due to the Newton-type approach for solving the sets of nonlinear equations in the Expansion Method. In most cases, we started with a uniform buffer allocation, then perturbed the starting solution and re-started the optimization process to continue the search. The results we have then represent the best solution after twenty random starts. Even though there are many starts, the algorithm works very quickly.

5.1 Series Topologies

As we can see in the buffer allocation for the series topologies, the pattern found in the smallest network essentially becomes the pattern for the larger networks. If we take the optimization methodology and solve for a series network system of $N = 3, 5, 7$, and, 10 nodes respectively with an exponential service rate of $\mu_i = 10$, $\forall i$, we get the results presented in Table 6.

Table 6: Results for series topologies, $s^2 = 1$.

	N			
$s^2 = 1$	3	5	7	10
$\lambda = 5$				
$\bar{x}$	$\mathbf{10 \ldots 10^{**(1)}}$	$\mathbf{10 \ldots 10}$	$\mathbf{10 \ldots 10}$	$\mathbf{10 \ldots 10}$
Θ	4.9964	4.9939	4.9915	4.9879
Z	33.6535	56.0725	78.4791	112.065
$\lambda = 7$				
$\bar{x}$	$\mathbf{18 \ldots 18}$	$\mathbf{18 \ldots 18}$	$\mathbf{17 \ldots 17}$	$\mathbf{17 \ldots 17}$
Θ	6.9929	6.9882	6.9766	6.9670
Z	61.1439	101.8356	142.3955	203.0329
$\lambda = 8$				
$\bar{x}$	$\mathbf{25 \ldots 25^{**(2)}}$	$\mathbf{25 \ldots 25}$	$\mathbf{25 \ldots 25}$	$\mathbf{25 \ldots 25}$
Θ	7.9857	7.9764	7.9674	7.9544
Z	89.3423	148.5815	207.5785	295.6448

Notice that the buffer allocation is uniform across the series topology. While for $\lambda = 7$ in the last 2 series, the solution was not the same as the others, this is due to the objective function and the tradeoffs that are being made during the optimization with $\sum x_j$ and Θ since as the line increases with the number of stations, the $\sum x_j$ dominates the Θ threshold.

This type of result is similar to the uniform buffer allocation results of De Kok [7]. We would like to explain these results as a function of our approximation for the $M/G/1/K$ queue.

5.2 Explanation

In $M/M/1/K$ queueing systems, if we have a series network of exponential servers, and the blocking probability is *essentially zero,* then the output from one station in the line will be the input rate to the subsequent station, and this is approximately Poisson [12].

If we have a series of general servers and *essentially zero* blocking probabilities, then it is conjectured the outputs from each station and inputs to the subsequent stations in the series will also be approximately Poisson. This is the case for $M/G/c/c$ systems (including state dependent ones [4]).

Therefore, if we have a closed form expression of the buffer size for an $M/G/1/K$ system, then we can utilize it to predict the buffer size as a function of the traffic intensity ρ in the network and the threshold blocking probability p_K. We do not need a $GI/G/1/K$ formula.

Using the results of our $M/G/1/K$ model, we can generate the following formula to predict the buffer size as a function of ρ, p_K, s^2 the coefficient of variation of the service times.

$$K = \frac{\ln\left(\frac{p_K}{1-\rho+pk\,\rho}\right)}{(\ln\rho)} + \frac{1}{2}\frac{\left(s^2-1\right)\sqrt{\rho}\ln\left(\frac{pk}{1-\rho+pk\,\rho}\right)}{(\ln\rho)}.$$

This formula allows us to compute K^* as a function of ρ, p_K, and s^2. If one looks at the graphs for $s^2 = 1$, see Figure 13, one will notice that the results predicted by the graph are very close to the results we achieved with our optimization methodology in Table 6. In the generation of the five separate curves for the graph, $p_K = 0.0005$. As seen in Eq. (18)–(20), p_K and α in our search methodology are closely related.

What is interesting about the curves is that they do not overlap. Also the curve for $s^2 = 1$ is exact. The curves also indicate that as $\rho \to 1$ the buffer size explodes. This latter result is due to the denominator in the closed form expression.

While the expression and the graph assumes the p_K threshold value is constant for all values of ρ and s^2, in the actual network computations, the Θ and p_K will vary and also be affected by the objective function since we are trading off the buffer size against the Θ threshold. So we should not expect that the graph values and the result returned by the computer program will coincide exactly. However, the graph and closed form expression yield a very close approximation for the different values of ρ and s^2.

To substantiate these results further, an additional set of experiments is shown for the situations where $s^2 = 1/2$ and $3/2$, see Tables 7 and 8. Inspecting the outputs from the experiments and the graph shows close agreement.

Table 7: Results for series topologies, $s^2 = 1/2$.

$s^2 = 1/2$	N = 3	N = 5	N = 7	N = 10
$\lambda = 5$				
$\bar{x}$	$\mathbf{8}\ldots\mathbf{8}^{**(3)}$	$\mathbf{8}\ldots\mathbf{8}$	$\mathbf{8}\ldots\mathbf{8}$	$\mathbf{8}\ldots\mathbf{8}$
Θ	4.9956	4.9926	4.9897	4.9854
Z	28.4413	47.3800	66.3016	94.6522
$\lambda = 7$				
$\bar{x}$	$\mathbf{14}\ldots\mathbf{14}$	$\mathbf{14}\ldots\mathbf{14}$	$\mathbf{14}\ldots\mathbf{14}$	$\mathbf{14}\ldots\mathbf{14}$
Θ	6.9921	6.9869	6.9817	6.9741
Z	49.9412	83.1355	116.3020	165.9080
$\lambda = 8$				
$\bar{x}$	$\mathbf{21}\ldots\mathbf{21}^{**(4)}$	$\mathbf{20}\ldots\mathbf{20}$	$\mathbf{20}\ldots\mathbf{20}$	$\mathbf{20}\ldots\mathbf{20}$
Θ	7.9909	7.9800	7.9723	7.9611
Z	72.1147	119.9804	167.7015	238.8670

5.3 Series Simulation Experiments

In order to see how close to optimal are the generated patterns, it is interesting to compare our results with those of simulation. Experiments with ARENA with 102,000 time units, 2000 time units warmup and 10 replications were found to yield fairly stable results and low standard deviation. In the following series of tables, we sample the earmarked experiments $**(\bullet)$ from Tables 6 to 8 to compare with simulation. For all the non-exponential service times, a 2-stage gamma distribution was used to capture the general service times with non-unit s^2.

Table 8: Results for series topologies, $s^2 = 3/2$.

$s^2 = 3/2$	N = 3	N = 5	N = 7	N = 10
$\lambda = 5$				
$\bar{x}$	$\mathbf{11}\ldots\mathbf{11}^{**(5)}$	$\mathbf{11}\ldots\mathbf{11}$	$\mathbf{11}\ldots\mathbf{11}$	$\mathbf{11}\ldots\mathbf{11}$
Θ	4.9943	4.9905	4.9867	4.9812
Z	38.7344	64.5181	90.2709	128.8427
$\lambda = 7$				
$\bar{x}$	$\mathbf{21}\ldots\mathbf{21}$	$\mathbf{20}\ldots\mathbf{20}$	$\mathbf{20}\ldots\mathbf{20}$	$\mathbf{20}\ldots\mathbf{20}$
Θ	6.9911	6.9802	6.9726	6.9613
Z	71.9407	119.7906	167.7906	238.6796
$\lambda = 8$				
$\bar{x}$	$\mathbf{30}\ldots\mathbf{30}^{**(6)}$	$\mathbf{30}\ldots\mathbf{30}$	$\mathbf{30}\ldots\mathbf{30}$	$\mathbf{29}\ldots\mathbf{29}$
Θ	7.9841	7.9738	7.9639	7.9400
Z	105.9512	176.1874	246.1261	350.0453

In reading the simulation tables, Θ_a refers to the analytical throughput, Θ_s refers to the simulation throughput, the next column is the 95% [c.i.], and Z_s refers to the optimal objective function value from the simulation.

Table 9: Experiments #1 and #2, $\lambda = (5, 8)$, $s^2 = 1$.

$\bar{x}$	Θ_a	Θ_s	95% [c.i.]	Z_s	Z_α
(9,9,9)		4.9908	[4.9849,4.9967]	36.20*	
(10,10,10)	4.9964	4.9928	[4.9864,4.9992]	37.20	33.65
(11,11,11)		4.9946	[4.9910,4.9982]	38.40	
(24,24,24)		7.9883	[7.9829,7.9937]	83.70	
(25,25,25)	7.9857	7.9917	[7.9867,7.9967]	83.30*	89.34
(26,26,26)		7.9901	[7.9848,7.9954]	87.90	

In these first two experiments, Table 9, the optimal solution suggested by the methodology is very close to the results for the simulation model for $\lambda = 5$ and apparently optimal for $\lambda = 8$. The throughput predicted by the analytical model is covered by the 95% confidence interval on the first but not the second experiment.

Table 10: Experiments #3 and #4, $\lambda = (5, 8)$, $s^2 = 1/2$.

$\bar{x}$	Θ_a	Θ_s	95% [c.i.]	Z_s	Z_α
(7,7,7)		4.9960	[4.9924,4.9996]	25.00*	
(8,8,8)	4.9956	4.9946	[4.9899,4.9993]	29.40	28.44
(9,9,9)		4.9921	[4.9862,4.9981]	34.90	
(20,20,20)		7.9892	[7.9835,7.9950]	70.80*	
(21,21,21)	7.9909	7.9836	[7.9788,7.9884]	79.40	72.12
(22,22,22)		7.9863	[7.9801,7.9925]	79.70	

In this middle set of experiments, Table 10, for $\lambda = 5, \Theta_\alpha$ is well within the 95% c.i. and the suggested optimal solution by our heuristic is close to the one for the simulation. For $\lambda = 8, \Theta_\alpha$ is not within the 95% c.i. but the values of θ_s for the neighboring solutions would seem to indicate that the heuristic solution $\mathbf{x} = (\mathbf{21}, \mathbf{21}, \mathbf{21})$ should have a higher throughput value.

In these last two experiments, Table 11, Θ_α is well within the 95% c.i. for both sets of experiments and just

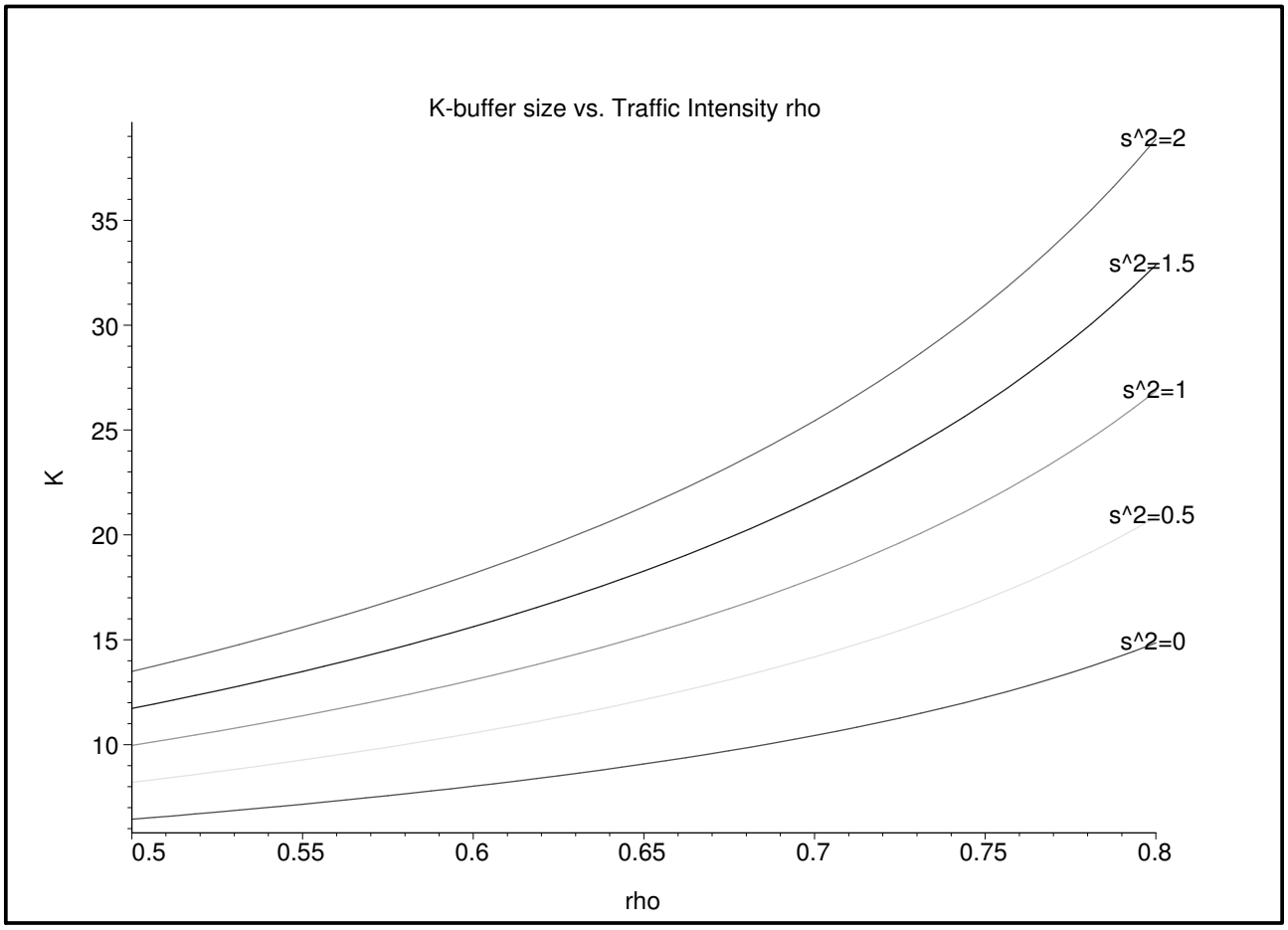


Figure 13: Graph of K vs. ρ.

Table 11: Experiments #5 and #6, $\lambda = (5, 8)$, $s^2 = 3/2$.

$\bar{x}$	Θ_a	Θ_s	95% [c.i.]	Z_s	Z_α
(10,10,10)		4.9920	[4.9867,4.9973]	38.00*	
(11,11,11)	4.9943	4.9940	[4.9867,4.9993]	39.00	38.73
(12,12,12)		4.9938	[4.9904,4.9972]	42.20	
(29,29,29)		7.9909	[7.9865,7.9953]	96.10*	
(30,30,30)	7.9841	7.9871	[7.9832,7.9910]	102.90	105.95
(31,31,31)		7.9865	[7.9800,7.9930]	106.50	

misses being optimal for $\lambda = 5$.

Table 12: Results for split topologies, $N = 3$.

	s^2		
$N = 3$	0.5	1.0	1.5
$\lambda = 5$			
$\bar{x}$	$\mathbf{8}\nearrow^{\mathbf{7}^{**(9)}}_{\mathbf{7}}$	$\mathbf{10}\nearrow^{\mathbf{9}^{**(7)}}_{\mathbf{9}}$	$\mathbf{11}\nearrow^{\mathbf{10}^{**(11)}}_{\mathbf{10}}$
Θ	4.9951	4.9963	4.9946
Z	26.9233	31.6578	36.3663
$\lambda = 7$			
$\bar{x}$	$\mathbf{14}\nearrow^{\mathbf{13}}_{\mathbf{13}}$	$\mathbf{18}\nearrow^{\mathbf{16}}_{\mathbf{16}}$	$\mathbf{21}\nearrow^{\mathbf{18}}_{\mathbf{18}}$
Θ	6.9932	6.9927	6.9898
Z	46.8231	57.2627	67.2410
$\lambda = 8$			
$\bar{x}$	$\mathbf{21}\nearrow^{\mathbf{18}^{**(10)}}_{\mathbf{18}}$	$\mathbf{25}\nearrow^{\mathbf{22}^{**(8)}}_{\mathbf{22}}$	$\mathbf{30}\nearrow^{\mathbf{26}^{**(12)}}_{\mathbf{26}}$
Θ	7.9897	7.9860	7.9835
Z	67.2873	83.2221	98.4819

5.4 Splitting Topologies

In the splitting topologies, the arrival flow is split evenly between the two resulting downstreams, and thus a balanced buffer allocation should result. In fact, see Table 12 and 13, the symmetric pattern for the series topology is very similar for the splitting topology, except the buffer at the first node is much larger than the buffers required at the downstream nodes. Also, the objective function is smaller than the series topology, since there is less blocking in the downstream nodes.

5.5 Splitting Simulation Experiment

For the first set of splitting experiments, Table 14, we are very close on the first run in minimizing Z and in agreement with the optimal solution value for the simulation model on the second run. Both values for Θ_α are within the 95% c.i.

In Table 15, we achieve the optimal solution for $\lambda = 5$ and both Θ_α are within the 95% c.i. but we miss the

Table 13: Results for split topologies, $N = 5$.

	s^2		
$N = 5$	0.5	1.0	1.5
$\lambda = 5$			
$\bar{x}$	$\mathbf{8}^{\mathbf{7,7}}_{\mathbf{7,7}}$	$\mathbf{10}^{\mathbf{9,9}}_{\mathbf{9,9}}$	$\mathbf{11}^{\mathbf{10,10}}_{\mathbf{10,10}}$
Θ	4.9916	4.9939	4.9912
Z	44.3408	52.0825	59.7900
$\lambda = 7$			
$\bar{x}$	$\mathbf{14}^{\mathbf{13,13}}_{\mathbf{13,13}}$	$\mathbf{18}^{\mathbf{16,16}}_{\mathbf{16,16}}$	$\mathbf{21}^{\mathbf{18,18}}_{\mathbf{18,18}}$
Θ	6.9891	6.9879	6.9826
Z	76.9448	94.0769	110.3673
$\lambda = 8$			
$\bar{x}$	$\mathbf{21}^{\mathbf{18,18}}_{\mathbf{18,18}}$	$\mathbf{25}^{\mathbf{22,22}}_{\mathbf{22,22}}$	$\mathbf{30}^{\mathbf{26,26}}_{\mathbf{26,26}}$
Θ	7.9826	7.9766	7.9727
Z	110.3755	136.3779	161.2632

Table 14: Experiments #7 and #8, $\lambda = (5, 8)$, $s^2 = 1$.

$\bar{x}$	Θ_a	Θ_s	95% [c.i.]	Z_s	Z_α
(9^{9}_{9})		4.9914	[4.9878,4.9951]	35.60	
$(\mathbf{10}^{\mathbf{9}}_{\mathbf{9}})$	4.9963	4.9945	[4.9919,4.9971]	33.50*	31.66
(11^{9}_{9})		4.9948	[4.9898,4.9992]	34.50	
(24^{22}_{22})		7.9822	[7.9758,7.9886]	85.80	
$(\mathbf{25}^{\mathbf{22}}_{\mathbf{22}})$	7.9858	7.9897	[7.9842,7.9952]	79.30*	83.22
(26^{22}_{22})		7.9830	[7.9761,7.9899]	87.00	

optimal solution for $\lambda = 8$.

For the final set of splitting comparisons $s^2 = 3/2$, in Table 16, the Θ_α for $\lambda = 5$ is not within the 95% c.i. but for $\lambda = 8$ we are within and very close to being optimal.

5.6 Merging Topologies

Again, the results of the merging topologies in Table 17 and 18 are symmetric with respect to the splitting topologies, with similar throughputs and slightly reduced objective function values compared to the series and splitting topologies.

5.7 Merging Simulation Experiment

In the merging experiments, Table 19, $s^2 = 1$, we see that all the throughputs for the simulation model are similar to those of the $N = 3$ series and splitting models. The objective function values are slightly less for the merge topologies than either the series or splitting

Table 15: Experiments #9 and #10, $\lambda = (5, 8)$, $s^2 = 1/2$.

$\bar{x}$	Θ_a	Θ_s	95% [c.i.]	Z_s	Z_α
(7^{6}_{6})		4.9888	[4.9835,4.9941]	30.20	
$(\mathbf{8}^{\mathbf{7}}_{\mathbf{7}})$	4.9951	4.9924	[4.9860,4.9988]	29.60*	26.92
(9^{8}_{8})		4.9949	[4.9870,4.9978]	30.10	
(20^{17}_{17})		7.9934	[7.9892,7.9977]	60.60*	
$(\mathbf{21}^{\mathbf{18}}_{\mathbf{18}})$	7.9897	7.9881	[7.9813,7.9949]	68.90	67.29
(22^{19}_{19})		7.9909	[7.9846,7.9972]	69.10	

Table 16: Experiments #11 and #12, $\lambda = (5, 8)$, $s^2 = 3/2$.

$\bar{x}$	Θ_a	Θ_s	95% [c.i.]	Z_s	Z_α
(10^{9}_{9})		4.9894	[4.9849,4.9939]	38.60*	
$(\mathbf{11}^{\mathbf{10}}_{\mathbf{10}})$	4.9946	4.9881	[4.9842,4.9920]	42.90	36.37
(12^{11}_{11})		4.9919	[4.9883,4.9955]	42.10	
(29^{25}_{25})		7.9824	[7.9788,7.9860]	96.60*	
$(\mathbf{30}^{\mathbf{26}}_{\mathbf{26}})$	7.9835	7.9830	[7.9770,7.9890]	99.00	98.48
(31^{27}_{27})		7.9864	[7.9821,7.9907]	98.60	

Table 17: Results for merge topologies, $N = 3$.

	s^2		
$N = 3$	0.5	1.0	1.5
$\lambda = 5$			
$\bar{x}$	$^{7}_{7}\searrow\mathbf{8}^{**(15)}$	$^{9}_{9}\searrow\mathbf{10}^{**(13)}$	$^{10}_{10}\searrow\mathbf{11}^{**(17)}$
Θ	4.9951	4.9963	4.9946
Z	26.9219	31.6573	36.3654
$\lambda = 7$			
$\bar{x}$	$^{13}_{13}\searrow\mathbf{14}$	$^{16}_{16}\searrow\mathbf{18}$	$^{18}_{18}\searrow\mathbf{21}$
Θ	6.9932	6.9927	6.9898
Z	46.8216	57.2594	67.2339
$\lambda = 8$			
$\bar{x}$	$^{18}_{18}\searrow\mathbf{21}^{**(16)}$	$^{22}_{22}\searrow\mathbf{25}^{**(14)}$	$^{26}_{26}\searrow\mathbf{30}^{**(18)}$
Θ	7.9897	7.9858	7.9835
Z	67.2778	83.2059	98.4585

ones. This underscores the fact that this is generally the best topology of the three alternatives. The throughput of the analytical model does very well in light traffic, although the simulated value of the objective is lower, but this is due to the high throughput of the simulation. The analytical throughput is outside of the 95% confidence interval in the heavier traffic case but apparently the optimal solution for the $\lambda = 8$ case.

Inspecting Table 20, for $\lambda = 5$, the Θ_α is within the 95% c.i. while for the $\lambda = 8$, the Θ_α is within the 95% c.i. and both heuristic solutions are relatively close to the optimal solution.

For the results in Table 21, the analytical throughput

Table 18: Results for merge topologies, $N = 5$.

	s^2		
$N = 5$	0.5	1.0	1.5
$\lambda = 5$			
$\bar{x}$	$^{\mathbf{7,7}}_{\mathbf{7,7}}\searrow\mathbf{8}$	$^{\mathbf{9,9}}_{\mathbf{9,9}}\searrow\mathbf{10}$	$^{\mathbf{10,10}}_{\mathbf{10,10}}\searrow\mathbf{11}$
Θ	4.9917	4.9939	4.9912
Z	44.3385	52.0816	59.7881
$\lambda = 7$			
$\bar{x}$	$^{\mathbf{13,13}}_{\mathbf{13,13}}\searrow\mathbf{14}$	$^{\mathbf{16,16}}_{\mathbf{16,16}}\searrow\mathbf{18}$	$^{\mathbf{18,18}}_{\mathbf{18,18}}\searrow\mathbf{21}$
Θ	6.9891	6.9879	6.9827
Z	76.9415	94.0707	110.3531
$\lambda = 8$			
$\bar{x}$	$^{\mathbf{18,18}}_{\mathbf{18,18}}\searrow\mathbf{21}$	$^{\mathbf{22,22}}_{\mathbf{22,22}}\searrow\mathbf{25}$	$^{\mathbf{26,26}}_{\mathbf{26,26}}\searrow\mathbf{30}$
Θ	7.9826	7.9767	7.9728
Z	110.3569	136.3469	161.2188

Table 19: Experiments #13 and #14, $\lambda = (5, 8)$, $s^2 = 1$.

$\bar{x}$	Θ_a	Θ_s	95% [c.i.]	Z_s	Z_α
(9 9 9)		4.9916	[4.9868,4.9964]	35.40*	
(9 **10** 9)	4.9963	4.9916	[4.9865,4.9967]	36.40	31.66
(9 11 9)		4.9881	[4.9822,4.9940]	40.90	
(22 24 22)		7.9800	[7.9754,7.9846]	88.80	
(22 **25** 22)	7.9858	7.9811	[7.9742,7.9880]	87.90	83.21
(22 26 22)		7.9848	[7.9796,7.9898]	85.20*	

Table 20: Experiments #15 and #16, $\lambda = (5, 8)$, $s^2 = 1/2$.

$\bar{x}$	Θ_a	Θ_s	95% [c.i.]	Z_s	Z_α
(6 7 6)		4.9907	[4.9894,4.9921]	28.30*	
(7 **8** 7)	4.9951	4.9903	[4.9834,4.9972]	31.70	26.9219
(8 9 8)		4.9895	[4.9838,4.9953]	35.50	
(17 20 17)		7.9841	[7.9777,7.9001]	69.90*	
(18 **21** 18)	7.9897	7.9835	[7.9733,7.9937]	73.50	67.2778
(19 22 19)		7.9880	[7.9838,7.9922]	72.00	

solution for $\lambda = 5$ is outside the 95% c.i. while it appears that in the $\lambda = 5$ case we have bracketed the optimal solutions and in the $\lambda = 8$ case we are within the 95% c.i. and seem to be very close to the optimal solution.

5.8 Comparison of Series, Merge, and Splitting Topologies

If we examine the set of tables for the different network topologies, one can see that the merging topologies actually dominate the splitting and series topologies when we inspect the objective function values and throughput measures.

Why does this occur? Well, in principle, in the merging and splitting topologies, the arrival and service processes are sub-divided into smaller units, so that the traffic flows are more uniformly distributed and thus the buffer allocation should be slightly reduced and this results in a smaller objective function value and larger throughputs.

Table 21: Experiments #17 and #18, $\lambda = (5, 8)$, $s^2 = 3/2$.

$\bar{x}$	Θ_a	Θ_s	95% [c.i.]	Z_s	Z_α
(9 10 9)		4.9849	[4.9816,4.9882]	43.10*	
(10 **11** 10)	4.9946	4.9859	[4.9794,4.9924]	45.10	36.3654
(11 12 11)		4.9890	[4.9817,4.9963]	45.00	
(25 29 25)		7.9781	[7.9723,7.9839]	100.90*	
(26 **30** 26)	7.9835	7.9798	[7.9731,7.9865]	102.20	98.4585
(27 31 27)		7.9793	[7.9723,7.9863]	105.70	

5.9 Bottleneck Topologies

As another set of experiments, it is interesting to compare the optimal buffer allocation when there are bottlenecks in the line. In this part of the experiments, we analyze a three-node series topology where the bottleneck is either in the front, middle, or end of the line. By a bottleneck, we mean that the service rates are unbalanced, with the bottleneck server at a rate of $\mu = 9$ compared to the other two servers with $\mu = 10$. Tables 22, 23, and 24 illustrate the resulting allocations and performance measures.

It is very difficult to generalize here because the variability in service times is not amenable to deterministic rules of thumb. On the other hand, as one can see inspecting Tables 22, 23, and 24, the bottleneck at the end of the series system appears to be better than the other two configurations. While the objective function is only slightly lower than the other two situations, it seems to indicate that having the bottleneck at the end of the line is slightly better. This makes sense since the previous two queues act as a buffer for the entire line, thus, increasing the throughput and decreasing the overall buffer cost. This depends, however, on the variability in the system as seen in the following detailed simulation comparisons, so some caution is advised here.

Table 22: Results for bottleneck at 1st node, $N = 3$.

	s^2		
$N = 3$	0.5	1.0	1.5
$\lambda = 5$			
$\bar{x}$	$\mathbf{9, 8, 8^{**(21)}}$	$\mathbf{11, 10, 10^{**(19)}}$	$\mathbf{13, 11, 11^{**(23)}}$
Θ	4.9952	4.9957	4.9942
Z	29.8079	35.3502	40.7845
$\lambda = 7$			
$\bar{x}$	**18,14,14**	**23,18,18**	**27,20,20**
Θ	6.9911	6.9915	6.9874
Z	54.9235	67.4753	79.6100
$\lambda = 8$			
$\bar{x}$	$\mathbf{31, 20, 20^{**(22)}}$	$\mathbf{38, 25, 25^{**(20)}}$	$\mathbf{45, 30, 30^{**(24)}}$
Θ	7.9853	7.9815	7.9787
Z	85.6732	106.4550	126.3394

Table 23: Results for bottleneck at 2nd node, $N = 3$.

	s^2		
$N = 3$	0.5	1.0	1.5
$\lambda = 5$			
$\bar{x}$	**8,9,8**	**10,11,10**	**11,13,11**
Θ	4.9952	4.9957	4.9942
Z	29.8074	35.3502	40.7836
$\lambda = 7$			
$\bar{x}$	**14,18,14**	**18,23,18**	**20,27,20**
Θ	6.9911	6.9915	6.9874
Z	54.9173	70.4701	79.5980
$\lambda = 8$			
$\bar{x}$	**20,31,20**	**25,38,25**	**30,45,30**
Θ	7.9854	7.9816	7.9787
Z	85.6365	106.4045	126.2798

Table 24: Results for bottleneck at 3rd node, $N = 3$.

	s^2		
$N = 3$	0.5	1.0	1.5
$\lambda = 5$			
$\bar{x}$	**8,8,9**	**10,10,11**	**11,11,13**
Θ	4.9912	4.9957	4.9942
Z	29.8075	35.3497	40.7826
$\lambda = 7$			
$\bar{x}$	**14,14,18**	**18,18,23**	**20,20,27**
Θ	6.9940	6.9915	6.9874
Z	55.0058	67.4653	79.5866
$\lambda = 8$			
$\bar{x}$	**20,20,31**	**25,25,38**	**30,30,45**
Θ	7.9854	7.9817	7.9788
Z	85.5993	106.3549	126.2212

5.10 Bottleneck Simulation Experiment

Table 25: Experiments #19 and #20, $\lambda = (5, 8)$, $s^2 = 1$.

$\bar{x}$	Θ_a	Θ_s	95% [c.i.]	Z_s	Z_α
(10,9,9)		4.9903	[4.9858,4.9948]	37.70*	
(11,10,10)	4.9952	4.9924	[4.9883,4.9965]	38.60	35.35
(12,11,11)		4.9931	[4.9885,4.9977]	40.90	
(37,24,24)		7.9851	[7.9802,7.9900]	99.90*	
(38,25,25)	7.9815	7.9861	[7.9787,7.9935]	102.90	106.46
(39,26,26)		7.9885	[7.9822,7.9948]	102.50	

In this first set of experiments with exponential service, Table 25, the Θ_α is within the 95% c.i. for $\lambda = 5$ and is within the 95% c.i. for $\lambda = 8$. Both heuristic solutions seem to be very close to the optimal solution.

Table 26: Experiments #21 and #22, $\lambda = (5, 8)$, $s^2 = 1/2$.

$\bar{x}$	Θ_a	Θ_s	95% [c.i.]	Z_s	Z_α
(7,8,7)		4.9934	[4.9886,4.9992]	28.60*	
(8,9,8)	4.9952	4.9933	[4.9888,4.9978]	31.70	29.81
(9,10,9)		4.9930	[4.9860,5.0000]	35.00	
(19,30,19)		7.9862	[7.9821,7.9903]	81.80*	
(20,31,20)	7.9854	7.9892	[7.9813,7.9972]	81.80*	85.64
(21,32,21)		7.9883	[7.9805,7.9961]	85.70	

In Table 26 we do fairly well in the lower traffic and in the higher traffic. Both analytical results are within the 95% c.i. confidence interval and seem to be close to the optimal solution for $\lambda = 5$ and tied for it in the $\lambda = 8$ case.

Finally, in Table 27, Θ_α is within the 95% c.i. for $\lambda = 5$ but not for $\lambda = 8$ and we appear to have achieved the optimal solutions in both cases.

5.11 Final Coda

To round out the optimization approach, a 9-node split-series topology and a 16-node series-merge-split topology will be examined. These are complex networks since they maintain an intricate blocking pattern embedded within them. They will also demonstrate the generality of the approach which is the main point of the paper in the first place. In both experiments, an arrival rate of $\lambda = 5$ from an exponential distribution was utilized. The simulation experiments for both networks were run for 10 replications for $102,000$ time units, with the first 2000 time units as a warm-up period.

Table 27: Experiments #23 and #24, $\lambda = (5, 8)$, $s^2 = 3/2$.

$\bar{x}$	Θ_a	Θ_s	95% [c.i.]	Z_s	Z_α
(10,10,12)		4.9898	[4.9836,4.9960]	42.20	
(11,11,13)	4.9942	4.9931	[4.9896,4.9967]	41.90*	40.78
(12,12,14)		4.9903	[4.9852,4.9954]	47.70	
(29,29,44)		7.9831	[7.9773,7.9889]	118.90	
(30,30,45)	7.9787	7.9914	[7.9852,7.9976]	113.60*	126.28
(31,31,46)		7.9896	[7.9843,7.9949]	118.40	

Figure 14 also shows the expected buffer sizes for lower traffic levels needed for some of the nodes in these complex network topologies to follow, since the optimization procedure needs a very good starting solution. This is just the other half of the curve of Figure 13. Again, the middle curve, $s^2 = 1$ is an exact solution.

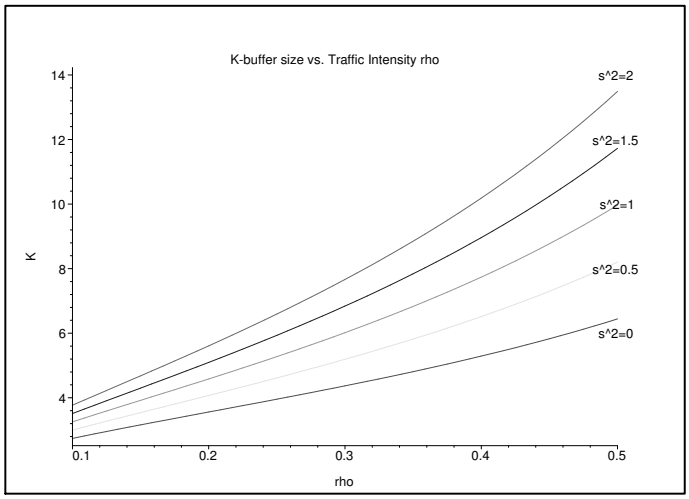


Figure 14: Graph of K vs. ρ.

5.11.1 9-node network

Figure 15 depicts the split-series topology with 9-nodes. First an experiment with exponential service times is carried out, then gamma distributions with $s^2 = 1/2$ are run, then finally, gamma distributions with $s^2 = 3/2$ are generated. At this point the patterns of the buffer allocation are compared as a function of the changes in the general service distributions.

The results are very encouraging for this series-split configuration. Inspecting Table 28 it is apparent that the throughput errors are extremely accurate [0.0180% − 0.0481%] and the objective function values range between 3.85% and 5.42%.

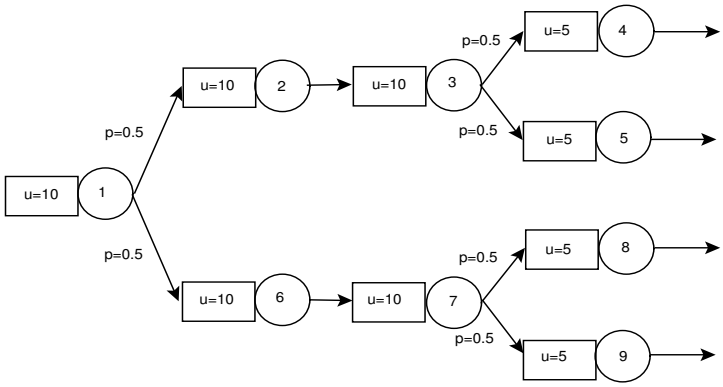


Figure 15: 9-node queueing network topology.

Table 28: 9-node experimental comparison.

experiment	Θ_α	Θ_s	% dev.	buffer pattern	Z_α	Z_s	% dev.
$s^2 = \frac{1}{2}$	4.9936	4.9917 [4.9875,4.9960]	0.0375%	**8** $4\to4$ **4** **4**; $4\to4$ **3** **3**	46.44	48.30	**3.851%**
$s^2 = 1$	4.9930	4.9906 [4.9869,4.9943]	**0.0481%**	**10** $5\to5$ **4** **4**; $5\to5$ **4** **4**	52.40	55.40	**5.42%**
$s^2 = \frac{3}{2}$	4.9922	4.9913 [4.9854,4.9973]	**0.0180%**	**11** $5\to5$ **5** **5**; $5\to5$ **5** **5**	57.10	59.70	4.36%

5.11.2 16-node network

Figure 16 illustrates the 16-node network series-merge-split topology. Table 29 illustrates the buffer allocation for a series of experiments. The final set of experiments illustrate different routings of customers along the top tier of workstations. The % split at the first node is indicated in the experiments and ranges from 50 : 50 to 70 : 30 with the higher percentage travelling to the upper tier of nodes. Other than the variation at the node #1, the other split nodes had a 50 : 50 percentage. In all cases, proportional and symmetric buffer allocations were derived by the algorithms for these different routing probabilities. All the optimization run times were very fast.

Also notice that the % deviations for the throughput value are very close in many instances and less than 0.1903% in the worst case while in the objective function value we are off. One other point that is interesting, is that the blocking probabilities in the simulation models at most all the nodes were not zero as one might expect, but ranges on average from $0 \sim 63\%$ for most of the nodes in the experiments $s^2 \in [1/2, 3/2]$ respectively. This is surprising but is probably due to the fact that we are trading off throughput for the buffer space in the objective function. The highest blocking probabilities occur at the principal split node of the 9 and 16-node topologies respectively.

All in all, the optimal "simulated" results are very sensitive to the average throughput value, and as one can see, there is a great deal of variation in this value. Although the solutions generated by the design methodology are often close to the local optimum, it is difficult to precisely say how the heuristic does compared to the global optimum since one would need an exact value of throughput and this seems terribly difficult to achieve.

6 Summary and Conclusions

In this paper, we have examined an approximation technique which utilizes a closed form expression for the blocking probability in allocating the buffers in $M/G/1/K$ systems for series, merge, and splitting topologies. The closed form expression of the blocking probability was shown to be an effective and robust approximation for generating the optimal buffer allocation. We have utilized an approach that generates an approximation to the non-inferior set of solutions for maximizing throughput and minimizing total buffers. That our approach is very general should be of some importance to many of the applications where $M/G/1/K$ systems predominate.

Future efforts will examine $M/G/c/K$ systems and related optimization issues.

References

[1] Altiok, T. and S. Stidham, 1983. "The allocation of interstage buffer capacities in production lines," *IIE Transactions* **15**, 251–261.

[2] Baker, K.R., S.G. Powell and D.F. Pyke, 1990. "Buffered and Unbuffered Assembly Systmes with Variable Processing Times," *J. Mfg. Oper. Mgmt.* **3**, 200-223.

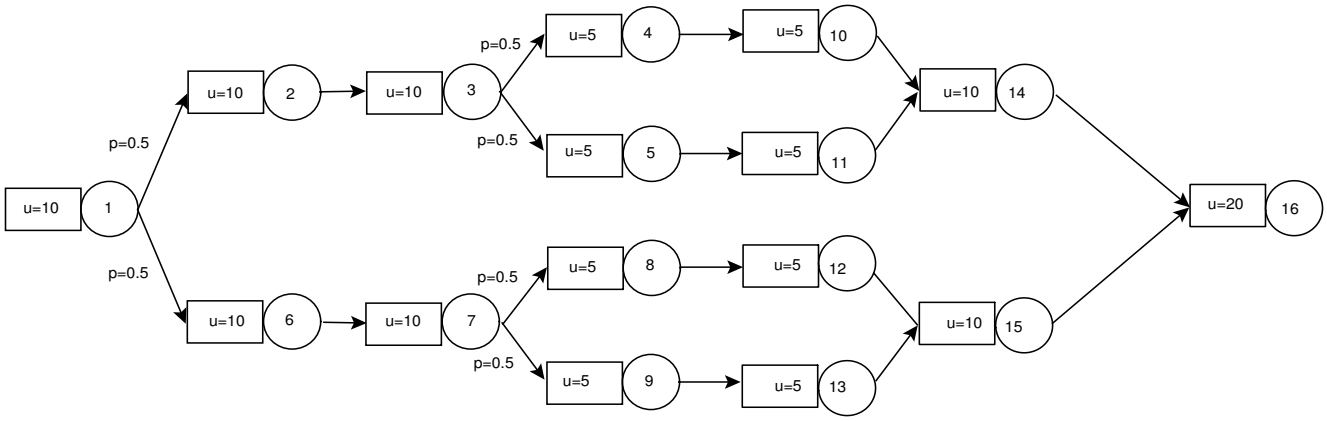
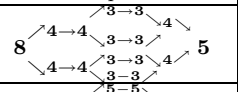
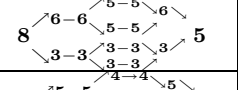
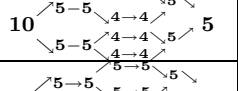
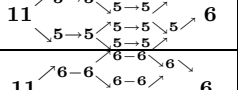
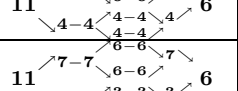
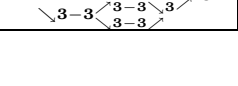


Figure 16: 16-node queueing network topology.

Table 29: 16-node experimental comparison.

experiment	Θ_α	Θ_s	% dev.	buffer pattern	Z_α	Z_s	% dev.
$s^2 = \frac{1}{2}$	4.9899	4.9941 [4.9882,5.0000]	0.0841%		79.07	66.90	18.19%
$s^2 = \frac{1}{2}\nearrow^{70}_{30}\searrow$	4.9916	4.9937 [4.9897,4.9978]	0.0421%		80.38	78.30	2.66%
$s^2 = 1$	4.9879	4.9903[4.9863,4.9944]	0.0481%		89.14	86.70	**2.81%**
$s^2 = \frac{3}{2}$	4.9877	4.9896[4.9857,4.9935]	0.0381%		99.30	97.40	1.95%
$s^2 = \frac{3}{2}\nearrow^{60}_{40}\searrow$	4.9877	4.9900[4.9879,4.9921]	0.0461%		99.78	96.00	3.94%
$s^2 = \frac{3}{2}\nearrow^{70}_{30}\searrow$	4.9818	4.9913[4.9869,4.9957]	**0.1903%**		101.21	91.70	10.37%

[3] Bazaraa, M., H.D. Sherali and C.M. Shetty, 1993. *Nonlinear Programming.* Wiley.

[4] Cruz, F. R. B. and J. MacGregor Smith, 2004. "Algorithms for analysis of generalized $M/G/c/c$ state dependent queueing networks." *Manuscript.* URL: ftp://ftp.est.ufmg.br/pub/fcruz/publics/ana.pdf

[5] Conway, R., W.L. Maxwell, J.O. McClain, and L.J. Thomas, 1988. "The Role of Work-in-process Inventories in Serial Production Lines," *Operations Research* **36**, 229-241.

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

[7] De Kok, A. G., 1990. "Computationally efficient approximations for balanced flowlines with finite intermediate buffers," *Int. Journal of Prod. Res.* **28**, 401-419.

[8] Garey, M. and D. Johnson, 1979. *Computers and Intractability* Freeman: San Francisco.

[9] Gelenbe, E., 1975. "On approximate Computer System Models," *JACM* **22**(2), 261-269.

[10] Gross, D. and C. Harris, 1985. *Fundamentals of Queueing Theory.* Wiley.

[11] Harris, J.H. and S.G. Powell, 1999. "An algorithm for optimal buffer placement in reliable serial lines." *IIE Transactions* **31**, 287-302.

[12] Hillier, F.S. and R.W. Boling, 1967. "Finite Queues in Series with Exponential or Erlang Service Times– a Numerical Approach," *Opns. Res.* **15**, 286-303.

[13] Hillier, F. and K. So, 1991. "The Effect of the Coefficient of Variation of Operation Times on the Allocation of Storage Space in Production Line Systems," *IIE Trans.* **23**-2, 198-206.

[14] Himmelblau, D.M., 1972. *Applied Nonlinear Programming.* Mc-Graw-Hill.

[15] Ho, Y.C., M.A. Eyler, and T.T. Chien, 1979. "A Gradient Technique for General Buffer Storage Design in a Production Line," *Int. J. of Prod. Res.* **17**, 557-580.

[16] Jafari, M.A. and J.G. Shantikumar, 1989, "Determination of Optimal Buffer Storage Capacities and Optimal Allocation in Multisatage Automatic Transfer Lines," *IIE Trans.* **21**, 130-135.

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

[18] Kerbache, L. and J. MacGregor Smith, 1987. "The Generalized Expansion Method for Open Finite Queueing Networks," *The European Journal of Operations Research* **32**, 448–461.

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

[20] Kubat, P. and U. Sumita, 1985. "Buffers and Backup Machines in Automatic Transfer Lines," *Int. J. Prod. Res.* **23**-6, 1259-1280.

[21] Kuehn, P.J., 1979. "Approximate Analysis of General Queueing Networks by Decomposition," *IEEE Trans. on COMM* **27**, 113-126.

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

[23] Kimura, T., 1996. "A Transform-Free Approximation for the Finite Capacity $M/G/s$ Queue," *Operations Research* **44**(6), 984-988.

[24] Kimura, T., 1996. "Optimal Buffer Design of an $M/G/s$ Queue with Finite Capacity," *Commun. Statist. - Stochastic Models* **12**(1), 165-180.

[25] Onvural, R., 1990. "Survey of Closed Queueing Networks with Blocking," *ACM Computing Surveys* **22**(2), 83-121.

[26] Powell, M.J.D., 1964. "An efficient method for finding the minimum of a function of several variables without calculating derivatives," *Comput. J.* **(7)**, 155–162.

[27] Powell, S.G., 1992. "Buffer Allocation in Unbalanced Serial Lines," Working Paper #289, Amos Tuck School of Business Administration, Dartmouth College, Hanover, N.H. 03755.

[28] Seelen, L.P., H. Tijms, and M.H. VanHoorn, 1985. *Tables for Multi-Server Queues*, North-Holland.

[29] Seong, D., S.Y. Chang, and Y. Hong, 1994. "Heuristic Algorithms for Buffer Allocation in a Production Line with Unreliable Machines," Working Paper, Management Sciences Research Laboratory, Pohang, Korea.

[30] Singh, A. and J. MacGregor Smith 1997, "Buffer Allocation for an Integer Nonlinear Network Design Problem," *Computers and Operations Research* **24**(5), 453-472.

[31] Smith, J. MacGregor and S. Daskalaki, 1988. "Buffer Space Allocation in Automated Assembly Lines,' *Operations Research* **36**(2), 343-358.

[32] Smith, J. MacGregor and N. Chikhale, 1995. "Buffer Allocation for a class of Nonlinear Stochastic Knapsack Problems," *Annals of Operations Research* **58**, 323-360.

[33] Smith, J. MacGregor, 2003. "$M/G/c/K$ blocking probability models and system performance," *Performance Evaluation* **52**, 237-267.

[34] Smith, J. MacGregor, S.B. Gershwin SB, and C.T. Papadopoulos CT (Eds.), 2000. "Performance Evaluation and Optimization of Production Lines," *Annals of Operations Research* **93**.

[35] Soyster, A.L., W.J. Schmidt, and M.W. Rohrer, 1979. "Allocation of Buffer Capacities for a Class of Fixed Cycle Production Lines," *AIIE Trans.* **11**, 140-146.

[36] Spinellis, D., C.T. Papadopoulos, J. MaCgregor Smith, 2000. "Large production line optimization using simulated annealing," *International Journal of Production Research* **38**(3), 509-541.

[37] Springer, M. and P. Makens, 1991. "Queueing Models for performance analysis and Selection of Single Station Models," *EJOR* **58**, 123-145.

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

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

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

[41] Yamashita, H. and S. Suzuki, 1987. "An Approximate Solution Method for Optimal Buffer Allocation in Serial n-stage Automatic Production Lines," *Trans. Japan Soc. Mech. Engin.* **53-C**, 807-814 (in Japanese).

[42] Yamashita, H. and R. Onvural, 1992. "Allocation of Buffer Capacities in Queueing Networks with Arbitrary Topologies," Working Paper, Department of Mechanical Engineering, Sophia University, Tokyo, Japan.

[43] Yamashita, H. and T. Altiok, 1995. "Buffer Capacity Allocation for a Desired Throughput in Production Lines," Working Paper, Department of Mechanical Engineering, Sophia University, Tokyo, Japan.

[44] Yao, D. and Buzacott, J.A., 1985. "Queueing Models for a Flexible Machining Station Part 1: The Diffusion Approximation," *EJOR* **19**, 233-240.