Proceedings of the XXVI Iberian Latin-American Congress on Computational Methods in Engineering CILAMCE 2005
Brazilian Assoc. for Comp. Mechanics & Latin American Assoc. of Comp. Methods in Engineering
Guarapari, Espírito Santo, Brazil, 19^{th}–21^{st} October 2005

Paper CIL20-0662

BUFFER ALLOCATION IN GENERAL QUEUEING NETWORKS

F. R. B. Cruz
A. R. Duarte
C. J. Klechen
fcruz@ufmg.br
andersonrd@ufmg.br
klechen@ufmg.br
Department of Statistics, Federal University of Minas Gerais
31270-901 - Belo Horizonte - MG, Brazil

Abstract. *Optimal buffer allocation in queueing network systems is recognized as a difficult stochastic, nonlinear, integer mathematical programming problem. It is usual that the objective function, constraints or both are not available in closed-form making the problem harder. A good approximation for the performance measures is essential for a successful algorithm for buffer allocation. A recently published approximation formula for predicting the optimal buffer allocation in general service time single queues, which is based on a two-moment approximation formula, is examined in details, based on which a new algorithm is proposed for buffer allocation in general service time queueing networks. Computational results are shown to attest for the efficacy of the approach in generating nearly optimal buffer allocation patterns.*

Keywords: *Buffer allocation, Queues, Networks.*

1. INTRODUCTION

Manufacturing, telecommunication, and material handling systems are just few examples of practical interest that may be viewed as finite buffer queueing networks. Because of the critical costs for buffer space, it is crucial to define optimal buffer spaces in order to ensure maximum performance at the lowest possible cost. The buffer allocation problem (BAP) is computationally hard to solve as the BAP is usually formulated as a stochastic, non-linear, integer mathematical programming problem. Besides, most of the times there is not even available closed-form objective functions and constraints but only approximations. The BAP problem becomes much more complicated when it involves general service time queues configured in networks (MacGregor Smith & Cruz, 2005) in a generic topology as seen in Fig. 1.

One of the objectives of this paper is to compare approximations for the blocking probability, p_k, the probability that an arriving entity finds the queueing system at its capacity. One of the most regarded performance measures of queueing systems, the blocking probability is a building block for buffer allocation formulations. The BAP is to find optimal values for k such that p_k are below some pre-specified threshold ε for all queues in the network. In this paper, buffer allocation will be restricted to networks of $M/G/1/k$ queues, which in Kendall's notation consider Markovian arrivals, General distributed service times, a single server, and total capacity k, including the one in service.

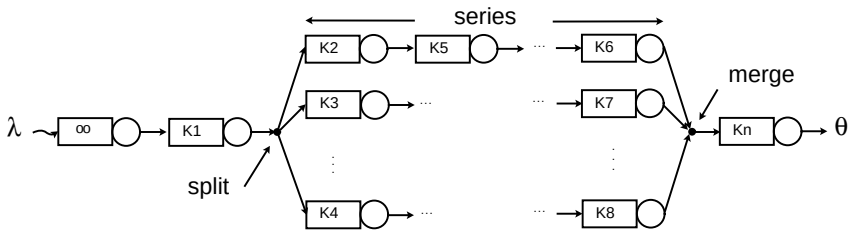


Figure 1: Queueing network in a general topology.

The paper is organized as follows. In Sec. 2 the BAP is defined as a non-linear mathematical programming formulation and a short literature review is presented about algorithms developed in the past for similar problems. Some of the most effective approximations for p_k are compared in Sec. 3. Then, Sec. 4 details the proposed algorithm to solve the BAP. Computational results that attests for the efficacy of the new algorithm are discussed in Sec. 5. Finally, Sec. 6 closes the paper with final comments and topics for future research in the area.

2. BUFFER ALLOCATION PROBLEM

2.1 Introduction

The BAP is concerned with how much space needs to be allocated in order to guarantee that the probability of loosing clients (or delaing them) is below a certain threshold. In its simplest definition the BAP seeks the lowest interger $k^* > 0$ such that $p_{k^*} \leq \varepsilon$ for some acceptable threshold $\varepsilon \in (0, 1)$. It is assumed that the system utilization, ρ (that is, the ratio of the arrival rate and the service rate, λ/μ) is below one, because an optimum for k may not exist for $\rho \geq 1$ (Kimura, 1996a).

2.2 Problem formulation

The BAP may be defined as a multi-objective non-linear mathematical programming formulation with integer decision variables $x_i \equiv k$, for the ith $M/G/1/k$ queue. However in this paper only the following single objective formulation will be considered

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

s.t.:

$$\Theta(\mathbf{x}) \geq \Theta^{\min}, \tag{2}$$

$$x_i \in \mathbb{N},\ \forall i, \tag{3}$$

which minimizes the total capacity allocated to the network, $\sum_i x_i$, subject to providing a minimum total throughput $\Theta^{\min}$. In this formulation, $\Theta^{\min}$ is the threshold throughput and x_i is the buffer k allocated to the ith $M/G/1/k$ queue. Although similar to a linear

integer mathematical programming problem, the formulation does not model directly the buffer allocation because $\Theta(\mathbf{x})$ is a function hard to define, involving the arrival rates, the service rates, and other parameters and variables in the queueing network.

2.3 Literature Overview

The literature of the BAP may be divided roughly into four methodological approaches: simulation methods, meta-heuristics, dynamic programming, and searching methods. In the following paragraphs, a short overview of these approaches will be presented.

The simulation methods aim to represent the actual systems by means of robust assumptions. In other words, general probability distributions are used to model the various aspects of the system, such as inter-arrival times, batch size of the arrivals, service times, among others. Simulation methods are usually very general and efficient but the price paid usually is a great computational effort that may reduce the size of treatable instances. However successful uses of simulation methods have been reported by researchers, such as, for instance, Soyster et al. (1979), for series queueing networks, and Baker et al. (1990), for general topologies.

Metaheuristics are very popular methods nowadays, mainly because of the increasing computational capacity available. Typical techniques that fall into the area include simulated annealing, taboo search, and more recently, generic algorithms. The advantages of metaheuristics are the absence of all those restrictive assumptions usually required by the traditional methods and the ability of avoiding local optima traps in the seek of the global optimum. The disadvantage is that usually the metaheuristics do not take into account the special structure of the problem. Among others, a successful case of use was reported by Spinellis et al. (2000) for buffer allocation in tandem networks of $M/M/c/k$ queues.

Dynamic programming is another powerful and reasonable approach for the BAP. Usually the exponential space complexity of dynamic programming methods reduces their applicability to very small sized instances. However, the approach has been proved successful in many cases. For instance, Kubat & Sumita (1985) and Yamashita & Altiok (1998) report results for networks of $M/M/1$ queues in series and Yamashita & Onvural (1994), for general topologies.

Finally, there are the search methods, which try to solve the problems avoiding the combinatorial explosion of possible solutions by choosing those close to the optimum results. Their main disadvantage is their restrictive assumptions, such as concavity and convexity, that may limit their applicability. In the past, the search methods were also successful in solving the BAP. In the series topology, the BAP was solved by searching methods for networks of $M/M/1/k$ queues (Altiok & Stidham, 1983) and $M/E_k/1/k$ queues (Hillier & So, 1991). In the general topology, for instance, there are results for $M/M/1/k$ queues (MacGregor Smith & Daskalaki, 1988; MacGregor Smith & Chikhale, 1995). For queueing networks configured in the general topology with general service times, $M/G/1/k$, the main object of this paper, there are not many results besides those reported by MacGregor Smith & Cruz (2005) with a methodology based on Powell's algorithm. The algorithm to be proposed here is considerably simpler and easier to implement.

3. BLOCKING PROBABILITY

Accurate approximations for the blocking probability for $M/G/1/k$ systems, p_k, will be presented in the following paragraphs. They are based on finite Markovian systems, $M/M/1/k$, but approximations based on infinite queueing systems are also common.

3.1 Markovian Systems

The blocking probability expression for a finite Markovian system is well-know

$$p_k = \frac{(1-\rho)\rho^k}{1-\rho^{k+1}}, \tag{4}$$

for $\rho \neq 1$, being possible then to express k in terms of ρ, the system utilization, and p_k, the blocking probability, as follows

$$k = \left\lceil \frac{\ln\left(\frac{p_k}{1-\rho+p_k\rho}\right)}{\ln(\rho)} \right\rceil, \tag{5}$$

in which $\lceil x \rceil$ is the lowest integer not inferior to x.

This is useful as an expression for the optimal buffer allocation in individual Markovian queues. Its usefulness for networks of general service time queueing systems will be apparent shortly.

3.2 Gelenbe's Approximation

Generally speaking, approximations developed in the past for the blocking probability are based on infinite queues. Actually, many of them could be adapted for $M/G/1/k$ queues. A survey by Makens (1992), for instance, analyzed five different approximations and concluded that Gelenbe's formula (Gelenbe, 1975) is efficient for most of the cases tested. Gelenbe's approximation is based on an approximation of the discrete queueing process by a continuous diffusion process. The blocking probability is given by

$$p_k = \frac{\lambda(\mu-\lambda)e^{-2\frac{(\mu-\lambda)(k-1)}{\lambda c_a^2+\mu c_s^2}}}{\left(\mu^2-\lambda^2 e^{-2\frac{(\mu-\lambda)(k-1)}{\lambda c_a^2+\mu c_s^2}}\right)}, \tag{6}$$

in which λ is the arrival rate, μ, the service rate, $c_a^2 = \mathrm{Var}(T_a)/\mathrm{E}(T_a)^2$ is the squared coefficient of variation of the inter-arrival time, T_a, and $c_s^2 = \mathrm{Var}(T_s)/\mathrm{E}(T_s)^2$ is the squared coefficient of variation of the service time, T_s. From Eq. (6), it is possible to get explicitly the optimal buffer allocation

$$k = \frac{2\lambda - 2\mu + \ln\left(\frac{p_k\mu^2}{\lambda(-\lambda+\mu+p_k\lambda)}\right)\lambda c_a^2 + \ln\left(\frac{p_k\mu^2}{\lambda(-\lambda+\mu+p_k\lambda)}\right)\mu c_s^2}{2(\lambda-\mu)}. \tag{7}$$

Taking unitary squared coefficients of variation of the inter-arrival time and service time, $c_s^2 = c_s^2 = 1$, Eq. (7) will lead to an optimal buffer allocation of Markovian single queues, $M/M/1/k$. Notice that the resulting expression will not be exactly the $M/M/1/k$ formula, Eq. (5), because Gelenbe's expression is an approximation. However, as noticed by Makens (1992) and MacGregor Smith & Cruz (2005), Gelenbe's expression is accurate for Markovian system, while is not accurate for deterministic service time systems $M/D/1/k$.

3.3 Two-moment Approximation

The two-moment approximation schema to be presented here are based on a weighted combination of the optimal buffer expressions of Markovian systems, $M/M/1/k$, denoted by k_ϵ^{M}, and deterministic service time systems, $M/D/1/k$, denoted by k_ϵ^{D}. Tijm's formula (Tijms, 1986) is one approximation that has been shown to be very good. It is given by

$$k_\epsilon^{\mathrm{Tijms}}(c_s^2) = c_s^2 k_\epsilon^{\mathrm{M}} + (1 - c_s^2) k_\epsilon^{\mathrm{D}}, \tag{8}$$

for $c_s^2 \geq 0$. Clearly, Tijm's formula is exact for the extreme cases if exact expressions are known for k_ϵ^{M} and k_ϵ^{D}.

Another good approximation is Kimura's formula (Kimura, 1996b), which is a little simpler as it uses only the optimal buffer expression of Markovian systems on its basis

$$k_\epsilon^{\mathrm{Kimura}}(c_s^2) = k_\epsilon^{\mathrm{M}} + \mathrm{NINT}\left[\frac{(c_s^2 - 1)}{2}\sqrt{\rho}\, k_\epsilon^{\mathrm{M}}\right], \tag{9}$$

in which $\mathrm{NINT}[x]$ is the nearest integer to x. Important to say about Kimura's formula is that it estimates the pure buffer without the space for the customers in service while Tijm's formula includes those in service.

Recently, MacGregor Smith (2002) proposed an approximation for $M/G/1/k$ queues based on Kimura's formula

$$k_\epsilon^{\mathrm{Smith}}(c_s^2) = \left[\underbrace{\frac{\ln\left(\frac{p_K}{1-\rho+p_K\rho}\right)}{\ln(\rho)} - 1}_{k_\epsilon^{\mathrm{M}}}\right] + \frac{(c_s^2 - 1)}{2}\sqrt{\rho}\left[\underbrace{\frac{\ln\left(\frac{p_K}{1-\rho+p_K\rho}\right)}{\ln(\rho)} - 1}_{k_\epsilon^{\mathrm{M}}}\right]. \tag{10}$$

in which Eq. (5), subtracted by the space for the single server, has been used as the estimate for the optimal buffer allocation of Markovian systems, k_ϵ^{M}. Now, factoring the terms of the approximation, the following simplified expression for the optimal buffer size in $M/G/1/k$ is given

$$k_\epsilon^{\mathrm{Smith}}(c_s^2) = \frac{\left[\ln\left(\frac{p_K}{1-\rho+p_K\,\rho}\right) - \ln(\rho)\right]\left(2 + \sqrt{\rho}c_s^2 - \sqrt{\rho}\right)}{2\ln(\rho)}. \tag{11}$$

Notice also that Eq. (11) yields the same expression as Eq. (5) if $c_s^2 = 1$ and the space for the server is subtracted. Additionally, as a side effect of Eq. (11), it is possible to obtain a closed-form approximate expression for the blocking probability of single $M/G/1/k$ queues

$$p_k = \frac{\rho^{\left(\frac{2+\sqrt{\rho}c_s^2-\sqrt{\rho}+2(k-1)}{2+\sqrt{\rho}c_s^2-\sqrt{\rho}}\right)}(-1+\rho)}{\rho^{\left(2\frac{2+\sqrt{\rho}c_s^2-\sqrt{\rho}+(k-1)}{2+\sqrt{\rho}c_s^2-\sqrt{\rho}}\right)}-1}. \tag{12}$$

As it will be seen in the following sections, Eq. (12) will be useful for computing performance measures of queueing networks of $M/G/1/k$ systems.

3.4 Computational Experiments

A series of computational experiments was performed to test the efficacy of the blocking probabilities given by the Markovian formula, Eq. (4), Gelenbe's formula, Eq. (6), and MacGregor Smith's formula, Eq. (12). For the buffer sizes the values $k = \{2, 4, 8, 16\}$ were considered. For each one of the buffer sizes, Markovian, $c_s^2 = 1.0$, hipoexponential, $c_s^2 = 0.5$, and hiperexponential service time systems, $c_s^2 = 2.0$, were tested. Because no exact blocking probabilities were available, the results were compared with simulations obtained with a Gamma random variable with convenient parameters, and 20,000 simulated time units to approach steady state. ARENA was the simulation system employed (see Kelton et al., 2001, for details). The simulation results presented are averages from 30 replications. The standard errors are too small to be noticed in the graphs presented in Fig. 2–Fig. 4.

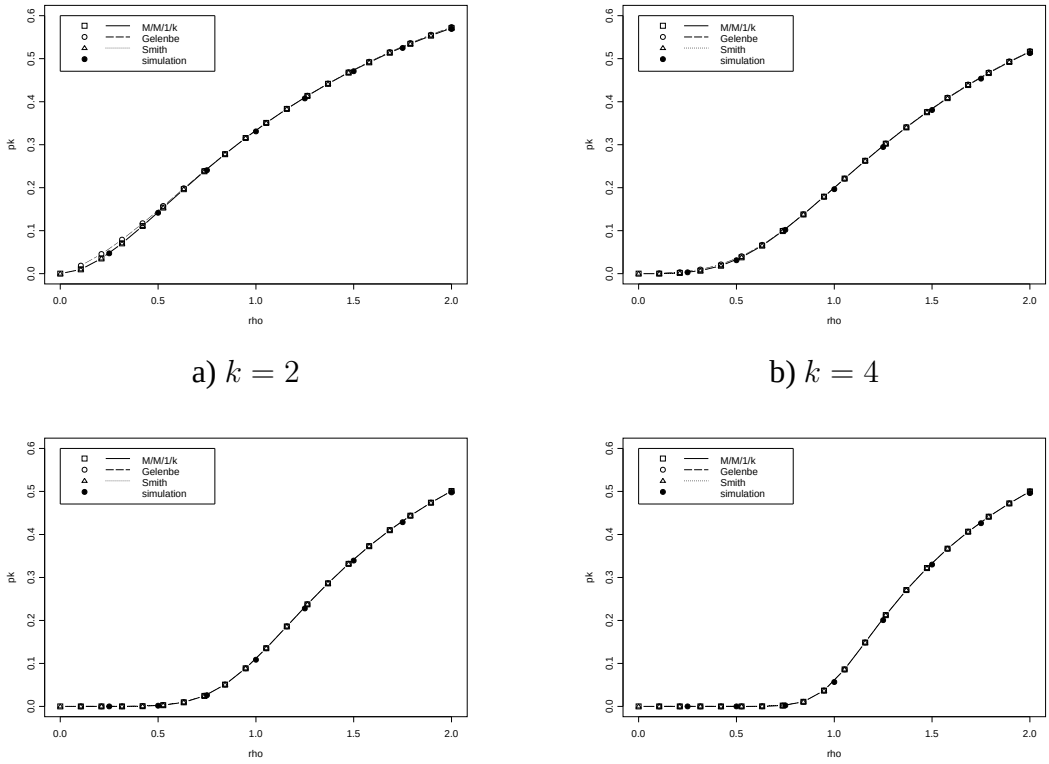
Markovian Systems

Results for the first set of experiments, done for Markovian systems, that is, $c_s^2 = 1.0$, are presented in Fig. 2. These experiments were planned just to validate the implementations as all of them should yield the same results which they indeed do in most of the cases. Actually, only for $k = 2$ and $\rho < 1.0$ some divergence was noticed involving Gelenbe's formula. It is noticeable that as k increases the blocking probabilities are close to zero when $\rho < 1.0$.

Hipoexponential Systems

Hipoexponential systems, with $c_s^2 = 0.5$, were also tested. The results are available in Fig. 3. For hipoexponential systems the Markovian approximation is an upper bound for the blocking probabilities as it always overestimates the simulation results, assumed here as reference. Thus, it is clear that by simply using Markovian approximations for hipoexponential systems one will tend to allocate larger buffer spaces than necessary.

Taking again the simulations as references, Gelenbe's approximation underestimates the blocking probabilities if the system utilization is below unity but tends to overestimate them otherwise. On the other hand, MacGregor Smith's approximation seems to be more accurate than Gelenbe's approximation and less dependent on the ρ. However, it is



a) $k = 2$ b) $k = 4$

c) $k = 8$ d) $k = 16$

Figure 2: Comparisons for p_k for Markovian systems.

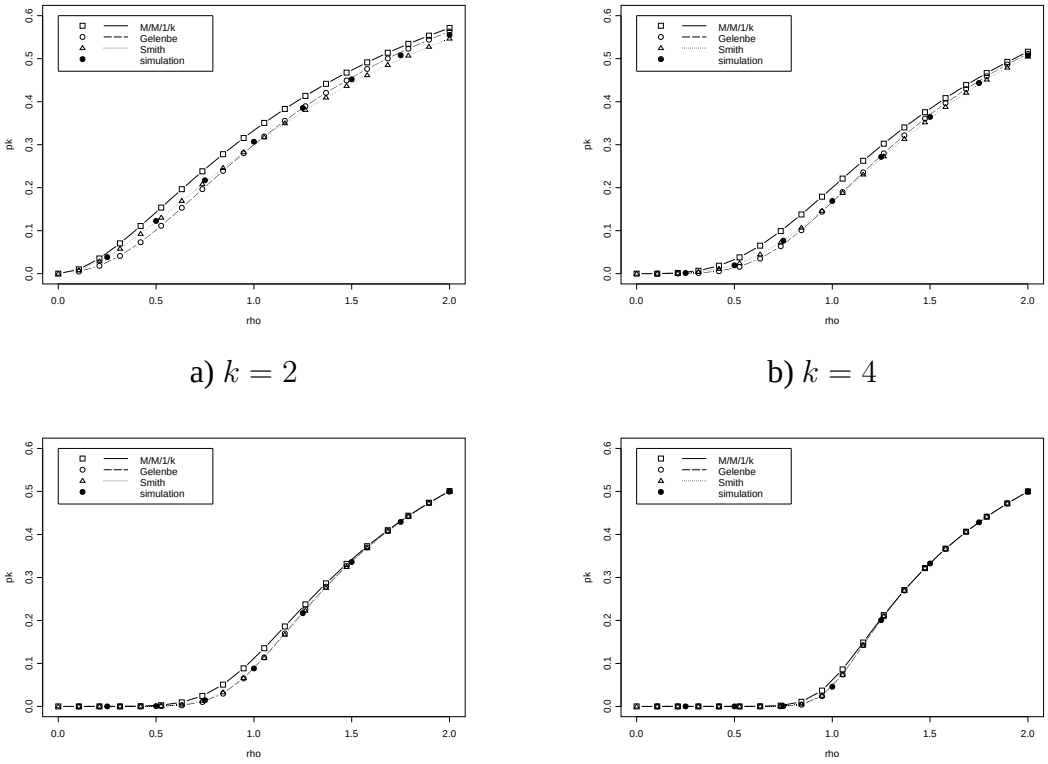
noticeable that although the approximations may disagree considerably for small buffer sizes they all tend to produce similar estimates as the buffer size increases. As a final remark, as the buffer size increases, the blocking probabilities tend to be zero for those cases in which the system utilization is below unity, $\rho < 1.0$.

Hiperexponential Systems

Results for hiperexponential systems, with $c_s^2 = 2.0$, are presented in Fig. 4. The Markovian approximations may be seen as a lower bound for the blocking probabilities as their values always underestimate the simulation results, taken here as references. It is confirmed here the inadequacy of Markovian approximations for hiperexponential systems for optimal buffer allocation purposes as one will tend to allocate less buffer space than necessary.

In comparison with the simulations results, Gelenbe's approximations overestimate the blocking probabilities just in the range of ρ that it is most appropriate to consider in practice, that is, for system utilization less than the unity, $\rho < 1.0$. By its side, MacGregor Smith's approximation presents estimates closer to the simulation results independent on the ρ.

As observed for hipoexponential systems, all approximations tend to agree for large buffer size systems. Also similarly to hipoexponential systems, the blocking probability tends to be close to zero for system utilization below the unity as the buffer size increases.



a) $k = 2$ b) $k = 4$

c) $k = 8$ d) $k = 16$

Figure 3: Comparisons for p_k for hipoexponential systems with $c_s^2 = 0.5$.

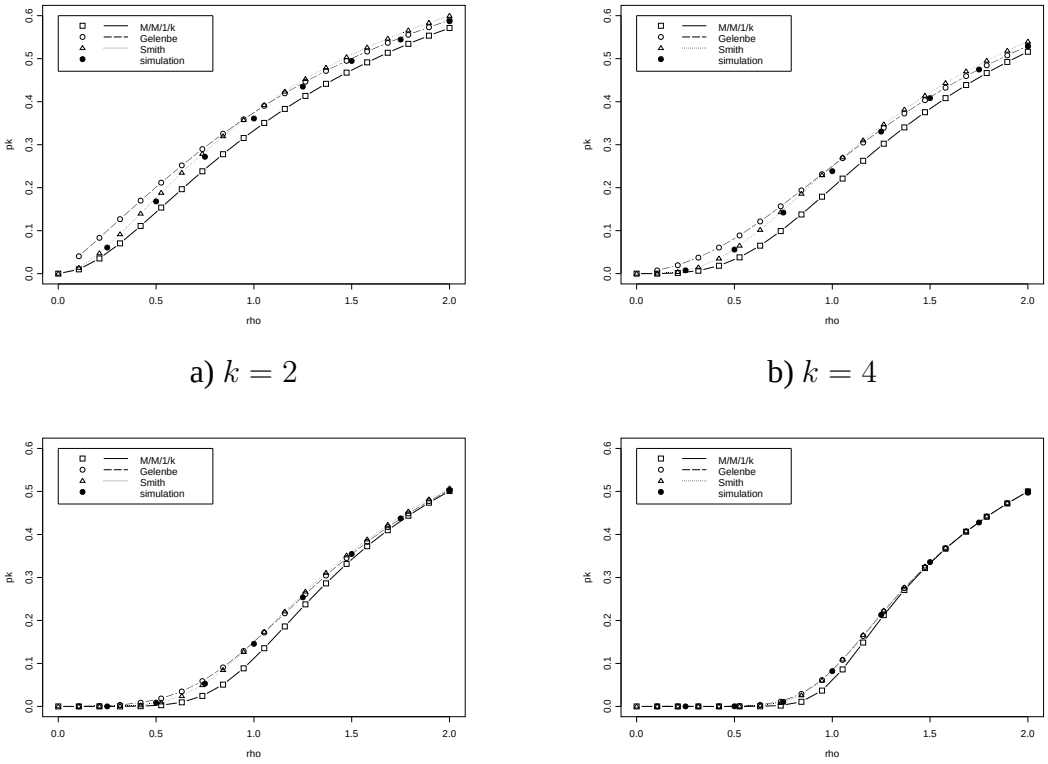
4. ALGORITHMS

The optimization problem that will be examined here is given by Eq. (1)–Eq. (3). Notice that $\Theta^{\min}$ can be pre-specified and then serve as the input λ to an approximate performance measure program that will also give $\Theta(\mathbf{x})$ for the $M/G/1/k$ queueing network system of interest, as explained in the following paragraphs. In the formulation, x_i become the decision variables under optimization control, that is, $x_i \equiv k$, for the ith queue.

A possible way to solve the problem is through the Lagrangean relaxation, a technique that consists in relaxing the complicating constraints and including then in the objective function as a penalty. Among the classical references to the Lagrangean relaxation, the paper by Fisher (1985) could be cited. A recently published tutorial about the Lagragean relaxation by Lemaréchal (2003) is another reference for the technique.

Thus, one way to incorporate the throughput constraint is through a penalty function. Defining a dual variable α and relaxing constraint (2), the following penalized objective function is given

$$L(\alpha) = \min \left[\sum_{i=1}^{N} x_i + \alpha \underbrace{(\Theta^{\min} - \Theta(\mathbf{x}))}_{\leq 0} \right] \tag{13}$$



a) $k = 2$ b) $k = 4$

c) $k = 8$ d) $k = 16$

Figure 4: Comparisons for p_k for hiperexponential systems with $c_s^2 = 2.0$.

s.t:

$$x_i \in \mathbb{N},\ \forall i \tag{14}$$

$$\alpha \geq 0. \tag{15}$$

Notice that the term $\alpha(\Theta^{\min} - \Theta(\mathbf{x}))$, always non-negative for any $\mathbf{x}$ feasible, is a penalty of the objective function related to the difference between the pre-specified throughput, $\Theta^{\min}$, and the effective throughput, $\Theta(\mathbf{x})$. Thus, it follows that $L(\alpha) \leq Z$, that is, $L(\alpha)$ is an inferior limit for Z, the optimal solution for the BAP, Eq. (1)–Eq. (3). The best possible (highest) inferior limit is given by the following Theorem 1.

Theorem 1 *The highest inferior limit, $L(\alpha^*) = max_{\alpha \geq 0} L(\alpha)$, is achieved for $\alpha^* \longrightarrow \infty$.*

Proof: *It follows from $\Theta(\mathbf{x})$, a non-decreasing function of $\mathbf{x}$, as it is seen in Fig. 5, and also from the Lagrangean function, $L(\alpha)$, which is the minimum of linear functions of α,*

$$L(\alpha) = \min \left(\underbrace{\sum_{i=1}^{N} x_i}_{\textit{intercept}} + \alpha \overbrace{(\Theta^{\min} - \Theta(\mathbf{x}))}^{\textit{slope}} \right),$$

with non-negative intercepts and slopes with

$$\lim_{\mathbf{x}\to\infty}(\Theta^{\min} - \Theta(\mathbf{x})) = 0,$$

which results in a non-decreasing convex envelopment, as it is seen in Fig. 6.

■

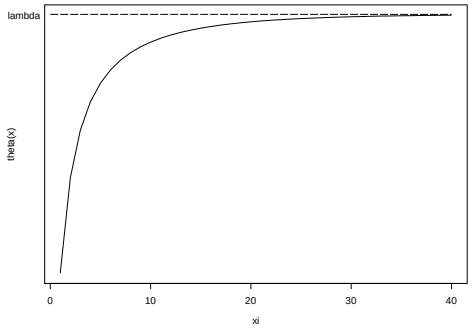


Figure 5: Throughput *versus* buffer size.

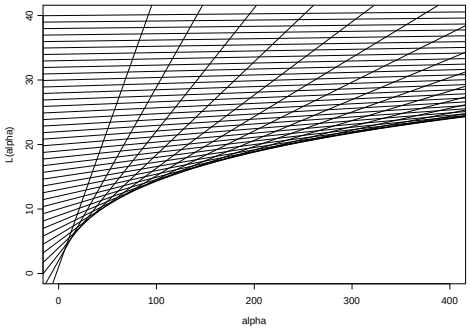


Figure 6: Lagrangian function $L(\alpha)$.

4.1 Note on the Lagrangean Multiplier

The Lagrangean relaxation of the BAP, $L(\alpha)$, plus an additional relaxation of the integrality constraints for x_i, is a classical unconstrained optimization problem. The best Lagrangean multiplier α, as defined by Theorem 1, is not practical because one would need that

$$\Big(\Theta^{\min} - \Theta(\mathbf{x})\Big) = 0,$$

which yields $x_i \to \infty, \forall\ i$. On the other hand, if a small difference, say $\Big(\Theta^{\min} - \Theta(\mathbf{x})\Big) = \varepsilon$, is acceptable, it must hold that

$$\alpha\Big(\Theta^{\min} - \Theta(\mathbf{x})\Big) \leq 1,$$

because, otherwise, it would be better to spend one more unity of buffer space to some ith queue, x_i, to increase $\Theta(\mathbf{x})$ (remind that $\Theta(\mathbf{x})$ is a non-decreasing function of $\mathbf{x}$). Thus, it is possible to define a corresponding α_ε as follows

$$\alpha_\varepsilon \leq 1/\Big(\Theta^{\min} - \Theta(\mathbf{x})\Big),$$

which, assuming $\Big(\Theta^{\min} - \Theta(\mathbf{x})\Big) \leq 10^{-3}$, yields $\alpha_\varepsilon = 10^3$.

4.2 Search Algorithm

Among the many possible algorithms to solve the BAP, a derivative free search algorithm was used, which is seen in Fig. 7, for its simplicity, and also efficiency, as it will

algorithm
read $G(V, A, P)$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, $\mathbf{c}_s^2$, $\mathbf{x}^{(0)}$
$\mathbf{x}^{(\mathrm{opt})} \leftarrow \mathbf{x}^{(0)}$
repeat
$\mathbf{x}^{(1)} \leftarrow \mathbf{x}^{(\mathrm{opt})}$
for $i = 1$ **until** n **do**
/* *unidirectional search* */
$\mathbf{x}^{(i+1)} \leftarrow \left\{\mathbf{x}^* | f(\mathbf{x}^*) = \min_{j \in \mathbb{N}} f(\mathbf{x}^{(i)} + j\mathbf{e}^{(i)})\right\}$
end for
if $f(\mathbf{x}^{(n+1)}) < f(\mathbf{x}^{(1)})$ **then**
$\mathbf{x}^{(\mathrm{opt})} \leftarrow \mathbf{x}^{(n+1)}$
end if
until $\|\mathbf{x}^{(\mathrm{opt})} - \mathbf{x}^{(1)}\| < \epsilon$
write $\mathbf{x}^{(\mathrm{opt})}$
end algorithm

Figure 7: Algorithm for optimal buffer allocation.

algorithm
read $G(V, A, P)$, $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, $\mathbf{c}_s^2$
/* preevaluate all nodes */
$Q \leftarrow \emptyset$
while $Q \neq V$
choose $j \in (V \setminus Q)$
if $\forall\ (i, j) \in A,\ i \in Q$ **then**
/* compute performance measures for node j */
compute $p_k^{(j)}$ *and* $\theta_j = \lambda_j \times (1 - p_k^{(j)})$
/* forward information */
for $\forall\ l$, *such that* $(j, l) \in A$ **do**
$\lambda_l \leftarrow \lambda_l + p_{(j,l)} \times \theta_j$
end for
/* update set Q */
$Q \leftarrow Q \cup \{j\}$
end if
end while
/* reevaluate all nodes */
(to be implemented)
/* write final results */
write $p_k^{(i)}$, $\theta_i, \forall\ i \in V$
end algorithm

Figure 8: Algorithm for performance evaluation.

be seen. The algorithm starts by reading the inputs, that is, the number of vertexes in the networks, V, the number of arcs, A, the routing matrix $P \equiv [p_{(i,j)}]$, which defines the probabilities of an entity to choose one or another path. Also read, are the vector of arrival rates, $\boldsymbol{\lambda}$, of service rates, $\boldsymbol{\mu}$, squared coefficient of variation of service rates, $\mathbf{c}_s^2$, and an initial buffer allocation vector, $\mathbf{x}^{(0)}$. With these values, the algorithm take the objective function

$$f(\mathbf{x}) = \sum_{i=1}^{N} x_i + \alpha\Big(\Theta^{\min} - \Theta(\mathbf{x})\Big), \tag{16}$$

which is optimized only in relation to the first coordinate of vector $\mathbf{x}$, keeping the remain fixed. The process is repeated for the second coordinate and so on, until the last coordinate is reached. A completely new vector $\mathbf{x}^{(n+1)}$ is obtained and compared with the previous vector $\mathbf{x}^{(1)}$. If the Euclidean distance between these two vectors is less than a pre-specified value ϵ, the algorithm stops. Otherwise, the hole process keeps running until the convergence is reached.

4.3 Performance Evaluation Algorithm

Notice that the algorithm presented in Fig. 7 needs an estimate for the objective function $f(\mathbf{x})$, Eq. (16), which implies that an estimate for $\Theta(\mathbf{x})$ must be sought. An algorithm available is the Generalized Expansion Method (GEM), successfully used in the past to estimate performance measures for finite queueing networks.

Well described in many papers, in particular in the recently published paper by Kerbache & MacGregor Smith (2000), the GEM is basically a combination of repeated trials and node-by-node decomposition in which each queue is analyzed separately and then corrections are made in order to take into account the interrelation between the queues in

the network. The GEM uses type I blocking, that is, the upstream node gets blocked if the service on a customer is completed but it cannot move downstream due to the queue at the downstream node being full. This is sometimes referred to as blocking after service, which is prevalent in most production and manufacturing, transportation, and similar systems. The implementation used in this work is seen in Fig. 8.

Similarly to the optimization algorithm, the GEM starts by reading all relevant information from the network under analysis. As it is seen in the paper by Kerbache & MacGregor Smith (2000), the GEM consist in creating for each finite queue, represented by vertex j, an auxiliary vertex h, modeled as an $M/M/\infty$ queue. When an entity arrives to the system, vertex j may be blocked with probability $p_k^{(j)}$, or unblocked, with probability $(1 - p_k^{(j)})$. Under blocking, the entities are rerouted to vertex h for a delay while node j is busy. Vertex h helps to accumulate the time an entity has to wait before entering vertex j and to compute the effective arrival rate to vertex j.

Thus, the algorithm chooses an arbitrary node, j, from set V but not from set Q (in which Q is the set of nodes already evaluated), such that for all arc $(i, j) \in A$, vertex i has been evaluated already. Then, vertex j has computed its blocking probability $p_k^{(j)}$, from Eq. (12), and its arrival rate, from $\theta_j = \lambda_j \times (1 - p_k^{(j)})$. These service rates are then forwarded as arrival rates to the downstream nodes (if they exist), and vertex j is included in set Q.

Notice that the GEM includes an reevaluation step that will not be implemented here. For the BAP defined here, the coupling between nodes will be minimum in the optimum, as a small difference between the arrival rate and the service rate is sought. Thus, the queueing network may be seen as a Jackson network in which all node may be analyzed separately, which is precisely the pre-evaluation step of the GEM presented in Fig. 8.

5. EXPERIMENTAL RESULTS

All algorithms were implemented in FORTRAN, taking advantage of all code already developed for similar problems (MacGregor Smith, 2002; MacGregor Smith & Cruz, 2005). For simplicity, all experiments were run for tandem queues (that is, series queues, see Fig. 9), even though the algorithm is ready to solve networks in general topologies too (see Duarte, 2005, for additional computational results for other topologies). Arrival rates considered were $\lambda = \Theta^{\min} = \{1.0, 2.0, 4.0\}$ users/s, homogeneous service rates $\mu_i = 10.0, \forall\ i$, with squared coefficient of variation $c_s^2 = \{0.5, 1.0, 2.0\}$, and number of nodes $N = \{2, 4, 8\}$. The results are seen in Tab. 1.

In order to see how close to optimal are the generated patterns, it is interesting to compare the results with those of simulation. Experiments with ARENA (see Kelton et al., 2001, for details) with 100.000 time units, 2000 time units warm-up and 30 replications were found to yield fairly stable results and acceptable 95% confidence intervals. For all the non-exponential service times, a 2-stage gamma distribution was used to capture the general service times with non-unit c_s^2.

From Tab. 1, it is possible to see in boldface the buffer allocation $\mathbf{x}$, obtained by the optimization algorithm. The pattern found in the small networks essentially becomes the pattern for the large networks. Notice that the buffer allocation is uniform across the series topology and that the throughput is essentially the arrival rate. This type of results

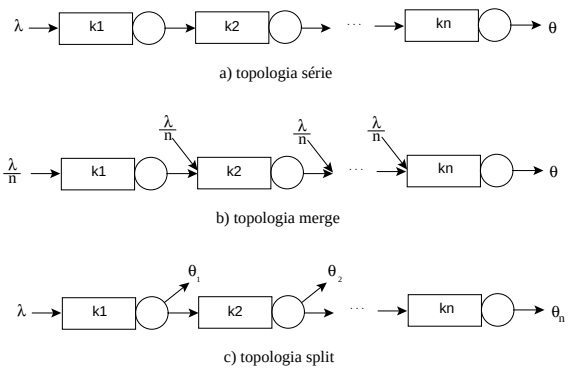


Figure 9: Networks of queues in various topologies.

Table 1: Simulation results for tandem queues.

				$\Theta(\mathbf{x})$					
c_s^2	λ	n	$\mathbf{x}$	average	MIN	MAX	CI[95%]	$L(\alpha)$	CPU*
0.5	1.0	2	2 2	0.99231	0.98945	0.99525	(0.98979;0.99483)	11.7	0.47
			2 3	0.99239	0.98957	0.99556	(0.98998;0.99480)	12.6	0.47
			3 2	1.00010	0.99825	1.00220	(0.99846;1.00174)	**4.9**†	0.45
			3 3	**1.00030**	**0.99833**	**1.00220**	(**0.99856**;**1.00204**)	**5.7**‡	0.45
			3 4	1.00030	0.99834	1.00220	(0.99856;1.00204)	6.7	0.45
			4 3	1.00150	0.99879	1.00460	(0.99930;1.00370)	5.5	0.45
			4 4	1.00160	0.99882	1.00460	(0.99935;1.00385)	6.4	0.47
1.0	2.0	4	3 3 3 3	1.98860	1.98390	1.99360	(1.98492;1.99228)	23.40	1.95
			4 4 4 4	1.99750	1.99370	2.00080	(1.99469;2.00031)	**18.50**†	1.87
			5 5 5 5	**1.99850**	**1.99380**	**2.00170**	(**1.99564**;**2.00136**)	**21.50**‡	1.90
			6 6 6 6	1.99970	1.99290	2.00380	(1.99532;2.00408)	24.30	1.90
2.0	4.0	8	9 9 9 9 9 9 9 9	3.9939	3.9880	4.0015	(3.98884;3.99896)	**78.10**†	7.98
			10 10 10 10 10 10 10 10	**3.9948**	**3.9904**	**3.9988**	(**3.99103**;**3.99857**)	**85.20**‡	8.05
			11 11 11 11 11 11 11 11	3.9983	3.9935	4.0021	(3.99504;4.00156)	89.70	9.83

*CPU time in minutes. †Best solution via simulation. ‡Best solution via optimization algorithm.

is similar to the uniform buffer allocation results of De Kok (1990).

Concerning the simulation results, only the 2-node network was analyzed in details because the time for setting up the experiments and running them was short enough to do so. Notice that the best solution from the optimization algorithm does not correspond exactly to the lowest $L(\alpha)$ (see Tab. 1, in boldface), but something quite close to it. The general conclusion is that optimization algorithm tends to allocate more space than necessary to ensure the desired performance. Also noticeable is that the CPU time for the simulations grows quickly as the number of node in the network increases indicating that the simulation may not be efficient as a tool for optimizing buffer allocation although it is certainly useful for assessing the quality of solutions via other methods.

6. SUMMARY AND CONCLUSIONS

One major difficulty in dealing with the buffer allocation problem (BAP) in general and for $M/G/1/k$ queues in particular is to find good approximate expressions for the performance measures of interest. The BAP is made much more difficult when queues are configured in networks, in which blocking after service frequently occurs and compli-

cates the analysis. In this paper, some of the most effective approximations for the blocking probability, a crucial performance measure for the BAP treated here, were extensively compared. The approximation by MacGregor Smith seemed to be the most accurate for the cases tested and was used for solving the BAP. The algorithm proposed is based on Lagrangean relaxation, a technique that has been proved efficient in solving optimization problems with complicate constraints. The Lagrangean relaxation enables one to avoid hard optimization formulations by relaxing complicate constraints and including then into the objective function as a penalty. Important properties of the relaxed problem were derived, which made possible the development of a search algorithm, considerably simpler than one previously published for the same problem (MacGregor Smith & Cruz, 2005). In comparison with exact simulation results, the algorithm seemed to produce very fast and accurate solutions and can be used in the design of production systems. Topics for future research in the area include extensions to systems that have loops, such as systems with captive pallets and fixtures. Also of interest is the study of algorithms for multi-server general service time queueing networks.

Acknowledgements

The research of Frederico R. B. Cruz has been partially funded by the CNPq, grants 201046/1994-6, 301809/1996-8, 307702/2004-9, and 472066/2004-8, the FAPEMIG (*Fundação de Amparo à Pesquisa do Estado de Minas Gerais*), grants CEX-289/98 and CEX-855/98, and PRPq-UFMG, grant 4081-UFMG/RTR/FUNDO/PRPq/99.

REFERENCES

Altiok, T. & Stidham, S., 1983. The allocation of interstage buffer capacities in production lines. *IIE Transactions*, vol. 15, pp. 251–261.

Baker, K. R., Powell, S., & Pyke, D., 1990. Buffered and unbuffered assembly systems with variable processing times. *Journal of Manufacturing & Operations Management*, vol. 3, pp. 200–223.

De Kok, A. G., 1990. Computationally efficient approximations for balanced flowlines with finite intermediate buffers. *International Journal of Production Research*, vol. 28, pp. 401–419.

Duarte, A. R., 2005. Buffer allocation in general queueing networks. Master's thesis, Department of Statistics - UFMG, Belo Horizonte, MG, Brazil (in Portuguese).

Fisher, M. L., 1985. An application oriented guide to Lagrangean relaxation. *Interfaces*, vol. 15, pp. 10–21.

Gelenbe, E., 1975. On approximate computer system models. *Journal of the ACM*, vol. 22, n. 2, pp. 261–269.

Hillier, F. & So, K., 1991. The effect of the coefficient of variation of operation times on the allocation of storage space in production line systems. *IIE Transactions*, vol. 23, n. 2, pp. 198–206.

Kelton, D., Sadowski, R. P., & Sadowski, D. A., 2001. *Simulation with Arena*. MacGraw Hill College Div., New York, NY, USA.

Kerbache, L. & MacGregor Smith, J., 2000. Multi-objective routing within large scale facilities using open finite queueing networks. *European Journal of Operational Research*, vol. 121, n. 1, pp. 105–123.

Kimura, T., 1996a. Optimal buffer design of an $M/G/s$ queue with finite capacity. *Communications in Statistics - Stochastic Models*, vol. 12, n. 1, pp. 165–180.

Kimura, T., 1996b. A transform-free approximation for the finite capacity $M/G/s$ queue. *Operations Research*, vol. 44, n. 6, pp. 984–988.

Kubat, P. & Sumita, U., 1985. Buffers and backup machines in automatic transfer lines. *International Journal of Production Research*, vol. 23, n. 6, pp. 1259–1280.

Lemaréchal, C., 2003. The omnipresence of Lagrange. *4OR*, vol. 1, pp. 7–25.

MacGregor Smith, J., 2002. $M/G/c/k$ blocking probability models and system performance. *Performance Evaluation*, vol. 52, pp. 237–267.

MacGregor Smith, J. & Chikhale, N., 1995. Buffer allocation for a class of nonlinear stochastic knapsack problems. *Annals of Operations Research*, vol. 58, pp. 323–360.

MacGregor Smith, J. & Cruz, F. R. B., 2005. The buffer allocation problem for general finite buffer queueing networks. *IIE Transactions on Design & Manufacturing*, vol. 37, n. 4, pp. 343–365.

MacGregor Smith, J. & Daskalaki, S., 1988. Buffer space allocation in automated assembly lines. *Operations Research*, vol. 36, n. 2, pp. 343–358.

Makens, M. C. S. P. K., 1992. Queueing models for performance analysis: Selection of single station models. *European Journal of Operational Research*, vol. 58, n. 1, pp. 123–145.

Soyster, A. L., Schmidt, J. W., & Rohrer, M. W., 1979. Allocation of buffer capacities for a class of fixed cycle production lines. *AIIE Transactions*, vol. 11, n. 2, pp. 140–146.

Spinellis, D., Papadopoulos, C. T., & MaCgregor Smith, J., 2000. Large production line optimization using simulated annealing. *International Journal of Production Research*, vol. 38, n. 3, pp. 509–541.

Tijms, H. C., 1986. *Stochastic Modeling and Analysis: A Computational Approach*. John Wiley & Sons, New York, MA, USA.

Yamashita, H. & Altiok, T., 1998. Buffer capacity allocation for a desired throughput in production lines. *IIE Transactions*, vol. 30, n. 10, pp. 883–892.

Yamashita, H. & Onvural, R., 1994. Allocation of buffer capacities in queueing networks with arbitrary topologies. *Annals of Operations Research*, vol. 48, pp. 313–332.