

1

2 Buffer Allocation in General Single-Server Queueing 3 Networks

4 F. R. B. Cruz^{a,*}, A. R. Duarte^a, and T. van Woensel^b

5 ^a*Department of Statistics, Federal University of Minas Gerais, 31270-901 - Belo Horizonte - MG, Brazil*

6 ^b*Department of Operations, Planning Accounting and Control, Eindhoven University of Technology, Eindhoven, The
7 Netherlands*

8 **Abstract**

9 The optimal buffer allocation in queueing network systems is a difficult stochastic, non-linear, integer mathematical
10 programming problem. Moreover, the objective function, the constraints or both are usually not available in closed-
11 form, making the problem even harder. A good approximation for the performance measures is thus essential for
12 a successful buffer allocation algorithm. A recently published two-moment approximation formula to obtain the
13 optimal buffer allocation in general service time single queues is examined in detail, based on which a new algorithm
14 is proposed for the buffer allocation in single-server general service time queueing networks. Computational results
15 and simulation results are shown to evaluate the efficacy of the approach in generating optimal buffer allocation
16 patterns.

17 *Key words:* Buffer allocation, queues, networks.

18 **1. Introduction**

19 Manufacturing, telecommunication, and material handling systems are just few examples of practical
20 interest that may be represented by finite buffer queueing networks. Because of the critical costs for buffer
21 space, it is crucial to optimally determine the buffer spaces in order to ensure maximum performance at the

* Corresponding author. Phone: (+55 31) 3499 5929. Fax: (+55 31) 3499 5924.

Email addresses: fcruz@ufmg.br (F. R. B. Cruz), andersonrd@ufmg.br (A. R. Duarte), T.v.Woensel@tm.tue.nl (T. van Woensel).

22 lowest possible cost. The buffer allocation problem (BAP) is computationally hard to solve as the BAP is
 23 usually formulated as a stochastic, non-linear, integer mathematical programming problem. The BAP is to
 24 find optimal values for the buffer sizes K such that the blocking probability p_K is below some pre-specified
 25 threshold ε for all queues in the network. In this article, the buffer allocation problem will focus on networks
 26 of $M/G/1/K$ queues, which in Kendall's notation considers Markovian arrivals, generally distributed service
 27 times, one single server, and the total capacity of K items, including the item in service. In this case, the
 28 BAP is a complicated problem as it involves general service time queues configured in arbitrary networks
 29 [19] as seen in Fig. 1. No closed-form objective functions are available for these types of networks. Thus, one
 30 needs to rely on approximations.

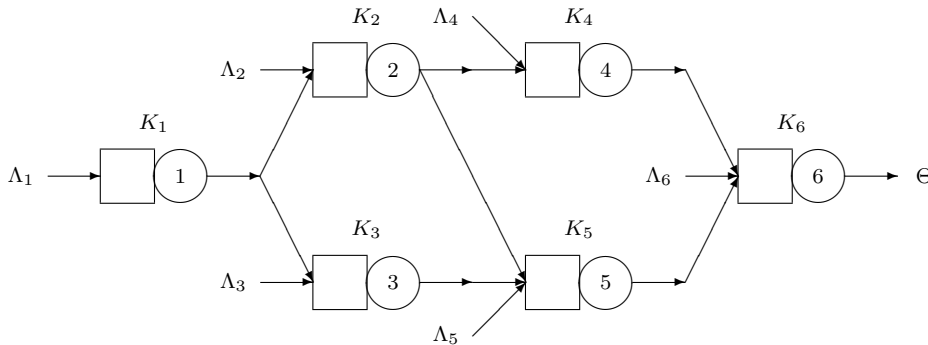


Fig. 1. Queueing network in an arbitrary topology.

31 One of the objectives of this article is to extensively compare approximations for the blocking probability,
 32 p_K , the probability that an arriving entity finds the queueing system at its capacity, in general single-server
 33 queues. One of the most regarded performance measures of queueing systems, the blocking probability is
 34 a building block for buffer allocation formulations. Another objective is to assess the accuracy of a novel
 35 implementation [3] for the Generalized Expansion Method (GEM), a well-know method for performance
 36 evaluation of finite queueing networks, applied to networks of $M/G/1/K$ queues. Finally, the third objective
 37 is to propose a simple algorithm for the buffer allocation problem in arbitrarily configured, finite buffer,
 38 single-server queueing networks. This will then result in insights into this challenging network design problem.

39 This article is organized as follows. In Sec. 2 the BAP is defined as a non-linear mathematical programming
 40 formulation and a short literature review is presented on the different algorithms developed in the past for
 41 similar problems. Some of the most effective approximations for p_K are extensively compared in Sec. 3. In
 42 Sec. 4 the performance evaluation algorithm for finite queueing networks is described and its accuracy is
 43 tested. Then, Sec. 5 describes the proposed algorithm to solve the BAP. Computational results evaluating
 44 the efficacy of the new buffer allocation algorithm are discussed in Sec. 6. Finally, Sec. 7 closes the article
 45 with final comments and topics for future research in the area.

46 2. Buffer Allocation Problem

47 2.1. Problem formulation

48 The BAP is concerned with how much space needs to be allocated in order to guarantee that the probability
49 of loosing clients (or delaying them) is below a certain threshold. In its simplest definition the BAP seeks
50 the lowest integer $K > 0$ such that $p_K \leq \varepsilon$ for some acceptable threshold $\varepsilon \in (0, 1)$. It is assumed that the
51 system utilization ρ (that is, the ratio between the arrival rate and the service rate, $\rho = \lambda/\mu$) is below 1.0,
52 because an optimum may not exist for K if $\rho \geq 1.0$ (see Kimura [12]).

53 The BAP may be defined by a multi-objective non-linear mathematical programming formulation with
54 integer decision variables $x_i \equiv K$, for the i th $M/G/1/K$ queue. However, in this article the following single
55 objective formulation will be considered:

$$56 \quad Z = \min \sum_i x_i, \tag{1}$$

57 s.t.:

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

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

58 which minimizes the total buffer allocation to the network, $\sum_i x_i$, subject to providing a minimum total
59 throughput Θ^{\min} . In this formulation, Θ^{\min} is some threshold throughput, not superior to the total external
60 arrival rate, $\Lambda = \sum_i \Lambda_i$, and x_i is the buffer K allocation to the i th $M/G/1/K$ queue, including those in
61 service. Although similar to a linear integer mathematical programming problem, the formulation does not
62 model directly the buffer allocation because $\Theta(\mathbf{x})$ is a function hard to define, involving the arrival rates,
63 the service rates, and other parameters and variables in the queueing network.

64 2.2. Literature Overview

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

68 The *simulation* methods aim to represent the actual systems by means of robust assumptions. In other
69 words, general probability distributions are used to model the various aspects of the system, such as inter-
70 arrival times, batch size of the arrivals, service times, among others. Simulation methods are usually very
71 general and efficient but the price paid is a great computational effort that may reduce the size of treatable
72 instances. However, successful uses of simulation methods have been reported by researchers, such as, for
73 instance, Soyster et al. [21], for series queueing networks, and Baker et al. [2], for general topologies.

74 *Metaheuristics* are very popular methods nowadays, mainly because of the increasing computational
75 capacity available. Typical techniques that fall into this area include simulated annealing, tabu search, and
76 more recently, generic algorithms. The advantages of metaheuristics are the absence of all those restrictive
77 assumptions usually required by the traditional methods and the ability of avoiding local optima traps in
78 the seek of the global optimum. The disadvantage is that usually the metaheuristics must be tailored to the
79 special structure of the problem. Among others, a successful case of use was reported by Spinellis et al. [22],
80 for buffer allocation in tandem networks of $M/M/c/K$ queues.

81 *Dynamic programming* is another powerful and reasonable approach for the BAP. Usually the exponential
82 space complexity of dynamic programming methods reduces their applicability to very small size instances.
83 However, the approach has been proved successful in many cases. For instance, Kubat and Sumita [14] and
84 Yamashita and Altiok [24] reported results for networks of $M/M/1$ queues in series and Yamashita and
85 Onvural [25], for general topologies.

86 Finally, there are the *search methods*, which try to solve the problems avoiding the combinatorial explosion
87 of possible solutions by choosing those solutions that are close to the optimum results. Their main disad-
88 vantage is their restrictive assumptions, such as concavity and convexity, that may limit the applicability. In
89 the past, search methods were also successful in solving the BAP. In series topologies, the BAP was solved
90 by search methods by Altiok and Stidham [1], for networks of $M/M/1/K$ queues, and by Hillier and So [8],
91 for $M/E_k/1/K$ queues. In general topologies, for instance, there are results reported by Smith and Chikhale
92 [18] and Smith and Daskalaki [20], for $M/M/1/K$ queues.

93 For networks of single-server general service time queues configured in general topologies, which is the
94 main object of this article, there are not many results besides those reported by Smith and Cruz [19] with a
95 methodology based on Powell's algorithm. The algorithm proposed in this article also falls into this category
96 of search methods but is considerably simpler and easier to implement, compared e.g. to Smith and Cruz
97 [19].

98 **3. Blocking Probability**

99 Accurate approximations for the blocking probability for $M/G/1/K$ systems, p_K , will be presented in the
100 following paragraphs. They are based on finite Markovian systems, $M/M/1/K$, but approximations based
101 on infinite queueing systems are also common. The relevance of accurate approximations for the BAP is
102 apparent when we take into account that the throughput in a single queue is the following function of the
103 arrival rate and the blocking probability

$$\theta = \lambda(1 - p_K).$$

104 3.1. *Markovian Systems*

105 The blocking probability expression for a finite Markovian system is well-know [7]

$$106 \quad p_K = \frac{(1 - \rho)\rho^K}{1 - \rho^{K+1}}, \quad (4)$$

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

$$109 \quad K = \left\lceil \frac{\ln\left(\frac{p_K}{1 - \rho + p_K \rho}\right)}{\ln(\rho)} \right\rceil, \quad (5)$$

110 in which $\lceil x \rceil$ is the lowest integer not inferior to x , and the blocking probability may be expressed in terms
111 of the threshold throughput, Θ^{\min} , and the arrival rate, λ , as

$$p_K \leq 1 - \Theta^{\min}/\lambda.$$

112 Expression (5) is not only useful for the optimal buffer allocation for individual Markovian queues but it
113 is also useful for networks of general service time queueing systems, as shortly it will be apparent.

114 3.2. *Gelenbe's Approximation*

115 Generally speaking, approximations developed in the past for the blocking probability are based on infinite
116 queues. Actually, many of them could be adapted for $M/G/1/K$ queues. A survey by Makens [16], for
117 instance, analyzed five different approximations and concluded that Gelenbe's formula [6] was efficient for
118 the majority of the cases tested. Gelenbe's approximation is based on an approximation of the discrete
119 queueing process by a continuous diffusion process. The blocking probability is given by

$$120 \quad p_k = \frac{\lambda(\mu - \lambda)e^{-2\frac{(\mu-\lambda)(k-1)}{\lambda c_a^2 + \mu c_s^2}}}{\left(\mu^2 - \lambda^2 e^{-2\frac{(\mu-\lambda)(k-1)}{\lambda c_a^2 + \mu c_s^2}}\right)}, \quad (6)$$

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

$$124 \quad K = \frac{2\lambda - 2\mu + \ln\left(\frac{p_K \mu^2}{\lambda(-\lambda + \mu + p_K \lambda)}\right) \lambda c_a^2 + \ln\left(\frac{p_K \mu^2}{\lambda(-\lambda + \mu + p_K \lambda)}\right) \mu c_s^2}{2(\lambda - \mu)}. \quad (7)$$

125 Taking both the squared coefficients of variation of the inter-arrival time and the service time equal to
126 one, $c_a^2 = c_s^2 = 1$, Eq. (7) results in the optimal buffer allocation of Markovian single queues, $M/M/1/K$.
127 Notice that the resulting expression will not be exactly the $M/M/1/K$ formula, Eq. (5), because Gelenbe's

128 expression is an approximation. However, as noticed by Makens [16] and by Smith and Cruz [19], Gelenbe's
 129 expression is accurate for Markovian system, while is not accurate for deterministic service time systems,
 130 $M/D/1/K$.

131 3.3. Two-moment Approximation

132 The two-moment approximation scheme is based on a weighted combination of some approximation for
 133 the optimal buffer expressions for Markovian systems, $M/M/1/K$, denoted by K_ϵ^M , and for deterministic
 134 service time systems, $M/D/1/K$, denoted by K_ϵ^D . Tijm's formula [23] is one two-moment approximation
 135 that has been shown to be very good in practice. It is given by

$$136 \quad K_\epsilon^{\text{Tijms}}(c_s^2) = c_s^2 K_\epsilon^M + (1 - c_s^2) K_\epsilon^D, \quad (8)$$

137 for $c_s^2 \geq 0$. Clearly, Tijm's formula is exact for the extreme cases, i.e., $c_s^2 = 0$ and $c_s^2 = 1$, if exact expressions
 138 are known for K_ϵ^M and K_ϵ^D .

139 Kimura's formula [13] is another good two-moment approximation, which is a little simpler as it uses as
 140 a basis only an approximation for the optimal pure buffer expression of Markovian systems

$$141 \quad B_\epsilon^{\text{Kimura}}(c_s^2) = B_\epsilon^M + \text{NINT} \left[\frac{(c_s^2 - 1)}{2} \sqrt{\rho} B_\epsilon^M \right], \quad (9)$$

142 in which $\text{NINT}[x]$ is the nearest integer to x . Important to say about Kimura's formula is that it estimates
 143 the pure buffer without the space for the customers in service (that is, $B = K - 1$, for $M/G/1/K$ systems),
 144 while Tijm's formula includes those in service.

145 Recently, Smith [17] proposed the following two-moment approximation for $M/G/1/K$ queues, based on
 146 Kimura's formula

$$B_\epsilon^{\text{Smith}}(c_s^2) = \underbrace{\left[\frac{\ln \left(\frac{pK}{1 - \rho + pK\rho} \right)}{\ln(\rho)} - 1 \right]}_{B_\epsilon^M = K^M - 1} + \frac{(c_s^2 - 1)}{2} \sqrt{\rho} \underbrace{\left[\frac{\ln \left(\frac{pK}{1 - \rho + pK\rho} \right)}{\ln(\rho)} - 1 \right]}_{B_\epsilon^M = K^M - 1}. \quad (10)$$

147 in which expression (5), subtracted by the space for the single server, is used as the estimate for the
 148 optimal pure buffer allocation of Markovian systems, B_ϵ^M . Now, factoring the terms of the approximation,
 149 the following simplified expression for the optimal buffer size in $M/G/1/K$ is given

$$150 \quad B_\epsilon^{\text{Smith}}(c_s^2) = \frac{\left[\ln \left(\frac{pK}{1 - \rho + pK\rho} \right) - \ln(\rho) \right] (2 + \sqrt{\rho} c_s^2 - \sqrt{\rho})}{2 \ln(\rho)}. \quad (11)$$

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

$$p_K = \frac{\rho \left(\frac{2 + \sqrt{\rho} c_s^2 - \sqrt{\rho} + 2(K-1)}{2 + \sqrt{\rho} c_s^2 - \sqrt{\rho}} \right) (-1 + \rho)}{\rho \left(\frac{2 + \sqrt{\rho} c_s^2 - \sqrt{\rho} + (K-1)}{2 + \sqrt{\rho} c_s^2 - \sqrt{\rho}} \right) - 1}. \quad (12)$$

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 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, $c_s^2 = 1.0$, hypoexponential, $c_s^2 = 0.5$, and hyperexponential service time systems, $c_s^2 = 2.0$, were tested. Because no exact blocking probabilities are available, the results are compared with simulation, using Gamma distributions for the service times and 20,000 simulated time units to approach steady state. ARENA was the simulation system employed (for details, see Kelton et al. [10]). The simulation results presented are averages from 30 replications. The mean standard errors are too small to be noticed in the graphs presented in Fig. 2–Fig. 4. Some studies have been published for $M/G/1/K$ single queues [17, 19] but not at the extend seen in this article.

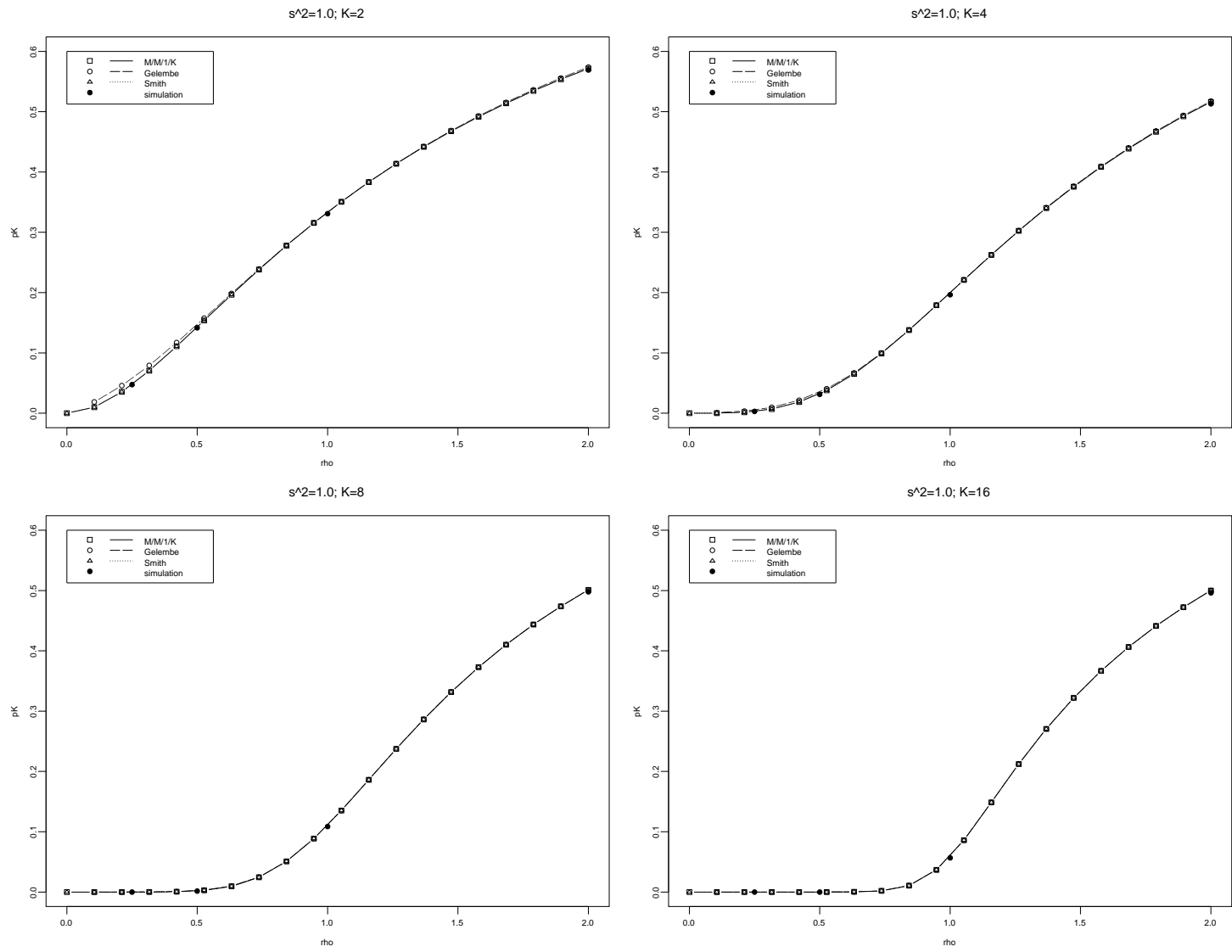
Markovian Systems

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 performed 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 $\rho < 1.0$ divergences were noticed involving Gelenbe's formula. Notice that as K increases the blocking probabilities get close to zero when $\rho < 1.0$, which is a logical and expected behavior.

Hypoexponential Systems

The results for hypoexponential systems, with $c_s^2 = 0.5$, are available in Fig. 3. For hypoexponential 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 hypoexponential systems one will tend to allocate larger buffer spaces than necessary.

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



8

Fig. 2. Comparisons for p_K for Markovian systems.

6

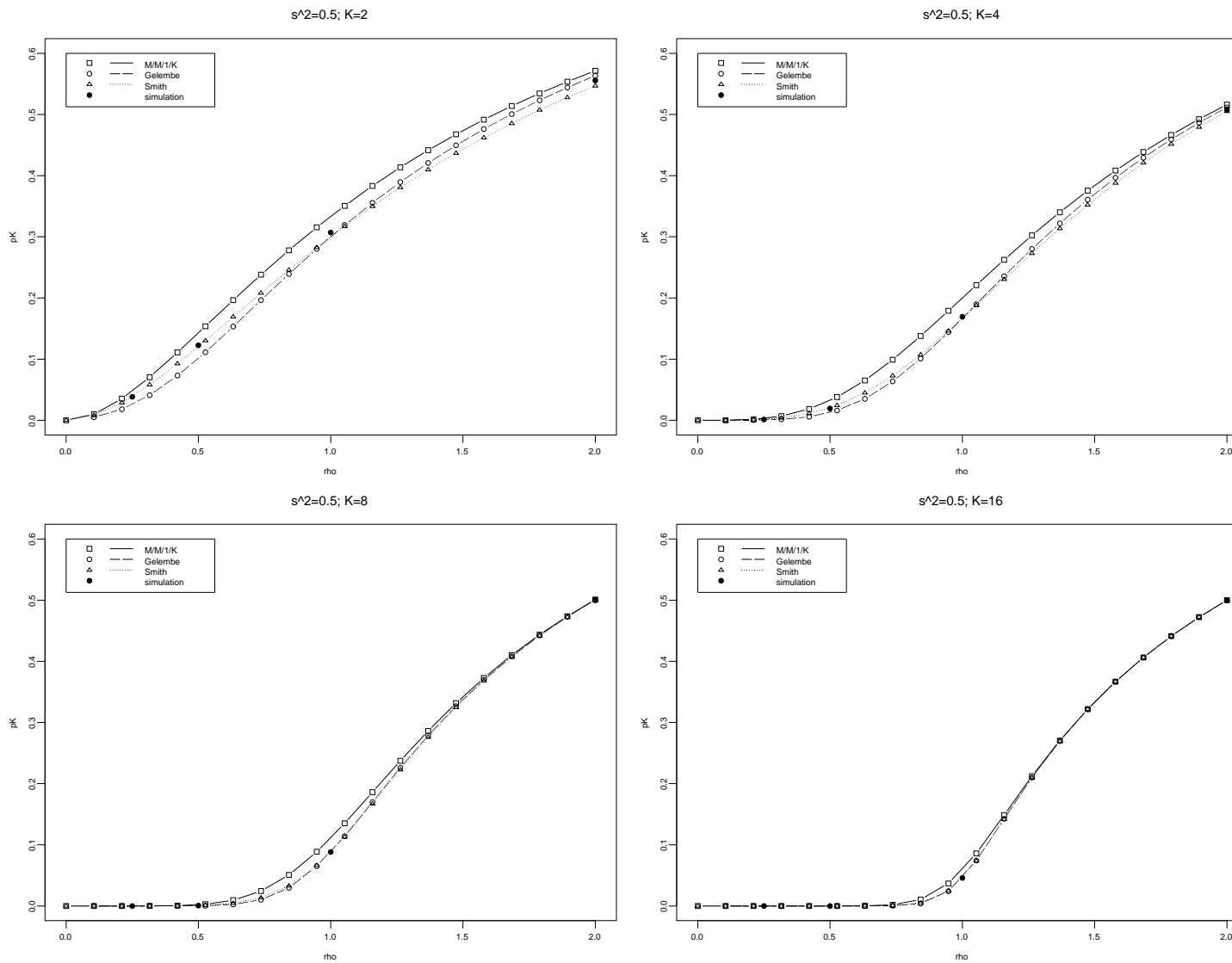


Fig. 3. Comparisons for p_K for hypoexponential systems with $c_s^2 = 0.5$.

182 on ρ . For large buffer sizes, the blocking probabilities tend to be zero for those cases in which the system
183 utilization is below unity, $\rho < 1.0$. However, it is noticeable that although the approximations may disagree
184 considerably for small buffer sizes they all tend to produce similar estimates as the buffer size increases.

185 *Hyperexponential Systems*

186 Results for hyperexponential systems, with $c_s^2 = 2.0$, are presented in Fig. 4. The Markovian approxima-
187 tions may be seen as a lower bound for the blocking probabilities, as their values always underestimate the
188 simulation results, taken here as references. The inadequacy of Markovian approximations for hyperexpo-
189 nential systems for optimal buffer allocation purposes is confirmed. In this case, one will allocate less buffer
190 space than necessary.

191 In comparison with the simulations results, Gelenbe's approximations overestimate the blocking proba-
192 bilities just in the most appropriate range of ρ (for system utilization less than the unity, $\rho < 1.0$). By its
193 side, Smith's approximation presents estimates close to the simulation results independent of ρ . As observed
194 for hypoexponential systems, the blocking probabilities tend to be close to zero for system utilization below
195 the unity as the buffer size increases. Also similarly to hypoexponential systems, all approximations tend to
196 agree for larger buffer size systems.

197 **4. Performance Evaluation Algorithm**

198 *4.1. Generalized Expansion Method*

199 Notice that, in order to solve the optimization problem given by Eq. (1), (2), and (3), one will need an
200 estimate for the throughput, $\Theta(\mathbf{x})$. An algorithm available is the Generalized Expansion Method (GEM),
201 successfully used in the past to estimate performance measures for arbitrarily configured finite queueing
202 networks.

203 Well described in many articles, in particular in the recently published article by Kerbache and Smith
204 [11], the GEM is basically a combination of node-by-node decomposition and repeated trials, in which each
205 queue is analyzed separately and then corrections are made in order to take into account the interrelation
206 between the queues in the network. The GEM uses type I blocking, that is, the upstream node gets blocked if
207 the service on a customer is completed but it cannot move downstream due to the queue at the downstream
208 node being full. This is sometimes referred to as blocking after service, which is prevalent in most production
209 and manufacturing, transportation, and similar systems. The implementation used here in this work is based
210 on a recently proposed implementation of the GEM by Cruz and Smith [3], suitable for light to moderate
211 traffic, and, for the first time, used in networks of $M/G/1/K$ queues. The algorithm may be seen in Fig. 5.

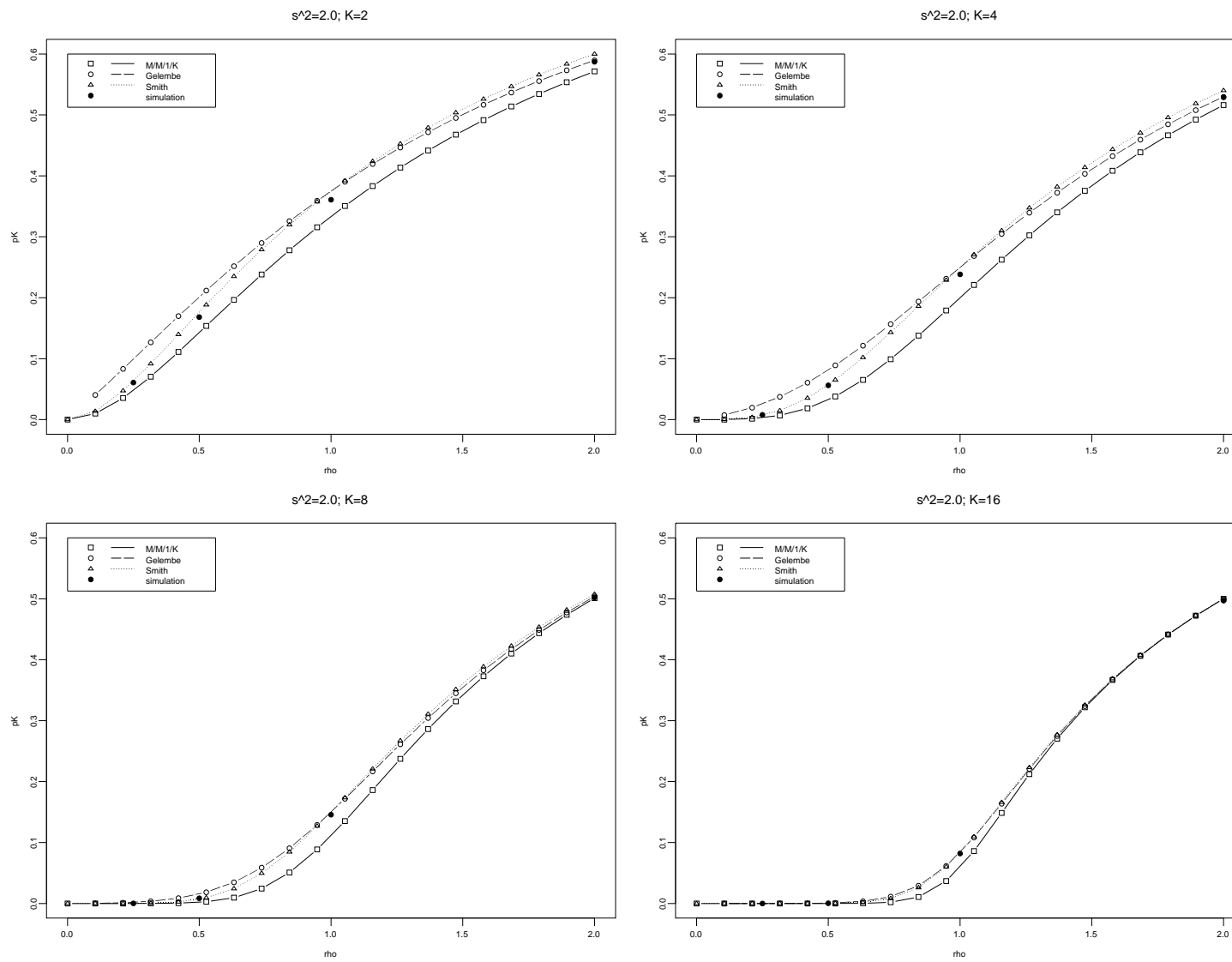


Fig. 4. Comparisons for p_K for hyperexponential systems with $c_s^2 = 2.0$.

```

algorithm
read  $G(V, A, P)$ ,  $\Lambda$ ,  $\mu$ ,  $c_s^2$ 
/* preevaluate all nodes */
 $\lambda_l \leftarrow \Lambda_l \forall l \in V$ 
 $Q \leftarrow \emptyset$ 
while  $Q \neq V$ 
  choose  $j \in (V \setminus Q)$ 
  if  $i \in Q, \forall (i, j) \in A$  then
    /* evaluate performance for node  $j$  */
    compute  $p_{K_j}$ 
    compute  $\theta_j = \lambda_j \times (1 - p_{K_j})$ 
    /* 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 */
 $Q \leftarrow \emptyset$ 
 $\theta_i^{\max} \leftarrow \infty, \forall i \in V$ 
while  $Q \neq V$ 
  choose  $i \in (V \setminus Q)$ 
  if  $j \in Q, \forall (i, j) \in A$  then
    /* update performance measure */
     $E[T_s]_i^* \leftarrow \min E[T_s]_i$ 
    s.t.:  $\theta_i \leq \theta_i^{\max}$ ,
     $E[T_s]_i \geq 1/\mu_i$ 
     $p_{K_i} \leftarrow f(\rho_i = \lambda_i/\mu_i^*, c_s^2)$ , Eq. (12)
     $\theta_i \leftarrow \lambda_i(1 - p_{K_i})$ 
    /* backpropagate to predecessors */
    for  $\forall k \in \{k' | (k', i) \in A\}$  then
      update  $\theta_k^{\max}$ 
    end for
    /* label node as evaluated */
     $Q \leftarrow Q \cup \{i\}$ 
  end if
end while
/* write final results */
write  $p_{K_i}, \theta_i, \forall i \in V$ 
end algorithm

```

Fig. 5. Algorithm for performance evaluation.

212 The GEM starts by reading all relevant information from the network under analysis, including the set
 213 of vertexes in the networks, V , the set of arcs, A , and the routing matrix, $P \equiv [p_{(i,j)}]$, which defines the
 214 probabilities of an entity to choose one or another path. Following Kerbache and Smith [11], the GEM
 215 consists of creating for each finite queue following another finite queue (see Fig. 6), represented by vertex j ,
 216 an auxiliary vertex h_j , modeled as an $M/G/\infty$ queue. When an entity arrives at the system, vertex j may
 217 be blocked with probability p_{K_j} , or unblocked, with probability $(1 - p_{K_j})$. Under blocking, the entities are
 218 rerouted to vertex h_j for a delay while node j is busy. Vertex h_j helps to accumulate the time an entity has
 219 to wait before entering vertex j and to compute the effective arrival rate to vertex j .

220 Thus, the algorithm chooses an arbitrary node, j , from set V but not from set Q (in which Q is the set of

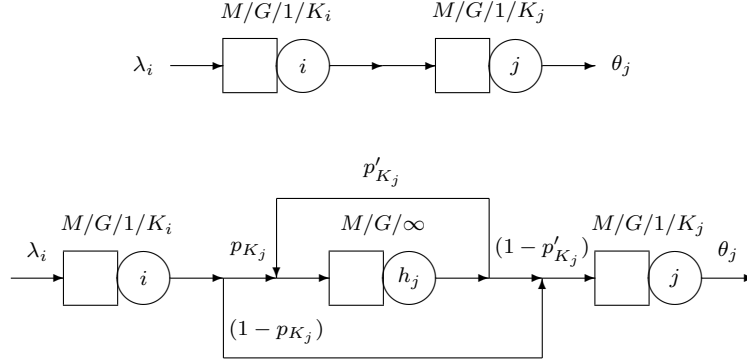


Fig. 6. Generalized expansion method.

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 $\theta_j = \lambda_j \times (1 - p_{K_j})$, $\lambda_j = \Lambda_j + \sum_i \lambda_{ij}$. These service rates are then forwarded as arrival rates to the downstream nodes (if they exist), and vertex j is included into set Q . For instance, in the network illustrated in Figure 1, a possible valid sequence to perform pre-evaluations is $1 \rightarrow 2 \rightarrow 4 \rightarrow 3 \rightarrow 5 \rightarrow 6$.

Notice that the GEM also includes a re-evaluation step, designed to guarantee flow conservation, that is, $\theta_j \leq \lambda_j + \sum_{i|(i,j) \in A} \theta_i p_{ij}$, for all $j \in V$. The re-evaluation step is a labeling algorithm working in reverse. For the network presented in Figure 1, a possible valid sequence to perform the reevaluations is $6 \rightarrow 5 \rightarrow 4 \rightarrow 3 \rightarrow 2 \rightarrow 1$, because a node can only be re-evaluated if all of its successors were re-evaluated already. The re-evaluation algorithm corrects the estimates by means of adjustments in the expected service time of each node i , $E[T_s]_i$. Further details will not be given in this article. The interested reader is referred to Cruz and Smith [3].

4.2. Computational Experiments

A series of computational experiments was performed to attest for the accuracy of the proposed implementation of the GEM for networks of $M/G/1/K$ queues. However, only networks of $M/G/1/2$ queues were considered because those are the most critical cases, since all approximations tend to agree for larger buffer systems, as shown in Section 3.

We run experiments in three basic topologies, series, merge, and split, combined into several values for the system utilization, ρ , and for the squared coefficient of variation, c_s^2 . When no exact results are available for the configurations tested, the results are compared with simulation. ARENA was the simulation system employed (for details, see Kelton et al. [10]). For the non-Markovian service times (that is, $c_s^2 = \{0.5, 2.0\}$), we used a two-stage gamma distribution, with convenient settings for the shape and scale parameters. In order to assure the steady-state regime, 200,000 time units were used as the simulation times with a warm-up

244 period of 2,000 time units. The simulation results presented are the averages over 20 replications, to get
245 reduced mean standard errors. Slightly longer and shorter simulations and replications were tested but the
246 results (not shown) did not change significantly.

247 Table 1 presents the results for the experiments, obtained from a PC Pentium(R) 4 CPU 3.00GHZ, 960
248 MB RAM running the Microsoft(R) Windows XP. In the column labeled ‘analytical’, we give the throughput
249 result from the GEM for each of the cases. We then compare this analytical result with the average result
250 obtained via the simulation. The column δ refers to the half-width of the 95% confidence interval. Notice that
251 the Monte Carlo errors were quite small. Also included in these tables is the % deviation for the analytical
252 results on the throughput, column $\Delta\% \theta$, from which we can see that the analytical results may be quite far
253 away from the simulation (exact) results (see results in boldface in Table 1). Mainly the results from this
254 GEM implementation get worse as the system gets overloaded. Notice that for the split topology the results
255 are quite good, because the flow in excess is rejected right away in the first node being then divided into
256 two nodes with equal service rate. It appears that the quality of the approximations is mainly dependent on
257 the squared coefficient of variation of the service time. In fact, the % deviation finds its highest values with
258 the highest c_s^2 .

259 Concluding, from the simulation CPU times reported for a single evaluation of the performance measures
260 for the queueing networks, it is apparent that simulation based techniques may not be quite effective for
261 optimization purposes of these queueing networks, unless the system is very small, because typically hundreds
262 or thousands of performance evaluations may be required for the optimization algorithms. On the other hand,
263 the analytical results are shown to be quite reliable and satisfactory. Additionally, as this new implementation
264 of the GEM is primarily for optimization purposes, we should not worry too much about high deviations
265 under overloaded traffic.

266 5. Buffer Allocation Algorithm

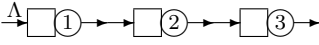
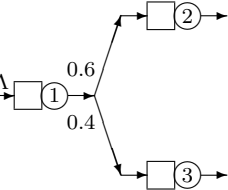
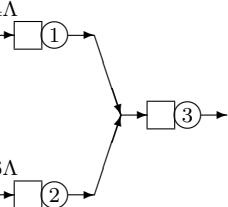
267 5.1. A Lagrangean Relaxation Approach

268 The optimization problem that will be examined here is given by Eq. (1), (2), and (3). In the formulation,
269 x_i becomes the decision variable under optimization control, that is, $x_i \equiv K$, for the i th queue.

270 A possible way to solve the problem is through Lagrangean relaxation, a technique that consists in relaxing
271 the complicating constraints and including them in the objective function as a penalty. Among the classical
272 references to the Lagrangean relaxation, the article by Fisher [5] should be cited. A recently published
273 tutorial about the Lagrangean relaxation by Lemaréchal [15] is another reference for the technique. One way
274 to incorporate the throughput constraint is then through a penalty function. Defining a dual variable α and
275 relaxing the constraint (2), the following penalized problem is given:

Table 1

Results for the generalized expansion method.

series topology															
		node #1				node #2				node #3					
ρ	c_s^2	analytical	simulation			analytical	simulation			analytical	simulation			CPU*	
		θ	θ	δ	$\Delta\% \theta$	θ	θ	δ	$\Delta\% \theta$	θ	θ	δ	$\Delta\% \theta$		
0.1	0.5	0.9783	0.9928	0.0011	1.5	0.9783	0.9928	0.0011	1.5	0.9783	0.9928	0.0011	1.5	1.63	
	1.0	0.9737	0.9910	0.0008	1.7	0.9737	0.9910	0.0008	1.7	0.9737	0.9910	0.0008	1.7	1.68	
	2.0	0.9643	0.9873	0.0008	2.3	0.9643	0.9873	0.0008	2.3	0.9643	0.9873	0.0008	2.3	1.68	
0.2	0.5	1.8530	1.9477	0.0018	4.9	1.8530	1.9477	0.0018	4.9	1.8530	1.9477	0.0018	4.9	3.38	
	1.0	1.8225	1.9348	0.0013	5.8	1.8225	1.9348	0.0013	5.8	1.8225	1.9348	0.0013	5.8	3.42	
	2.0	1.7675	1.9072	0.0011	7.3	1.7675	1.9072	0.0011	7.3	1.7675	1.9072	0.0011	7.3	3.38	
0.4	0.5	3.1667	3.6389	0.0014	13.0	3.1667	3.6389	0.0014	13.0	3.1667	3.6389	0.0014	13.0	6.75	
	1.0	3.0333	3.5522	0.0014	14.6	3.0333	3.5522	0.0014	14.6	3.0333	3.5522	0.0014	14.6	6.62	
	2.0	2.8324	3.3890	0.0019	16.4	2.8324	3.3890	0.0019	16.4	2.8324	3.3890	0.0019	16.4	6.42	
split topology															
		node #1				node #2				node #3					
ρ	c_s^2	analytical	simulation			analytical	simulation			analytical	simulation			CPU*	
		θ	θ	δ	$\Delta\% \theta$	θ	θ	δ	$\Delta\% \theta$	θ	θ	δ	$\Delta\% \theta$		
1.0	0.5	0.9904	0.9930	0.0010	0.3	0.5939	0.5959	0.0008	0.3	0.3965	0.3970	0.0006	0.1	1.37	
	1.0	0.9884	0.9912	0.0010	0.3	0.5926	0.5946	0.0008	0.3	0.3958	0.3966	0.0007	0.2	1.43	
	2.0	0.9842	0.9873	0.0009	0.3	0.5899	0.5925	0.0008	0.4	0.3943	0.3948	0.0007	0.1	1.40	
2.0	0.5	1.9322	1.9479	0.0017	0.8	1.1569	1.1682	0.0011	1.0	0.7754	0.7797	0.0010	0.6	3.17	
	1.0	1.9173	1.9352	0.0014	0.9	1.1474	1.1608	0.0009	1.2	0.7699	0.7744	0.0009	0.6	2.85	
	2.0	1.8892	1.9121	0.0012	1.2	1.1296	1.1469	0.0009	1.5	0.7596	0.7653	0.0009	0.7	2.83	
4.0	0.5	3.5683	3.6475	0.0020	2.2	2.1269	2.1884	0.0013	2.8	1.4414	1.4591	0.0010	1.2	5.90	
	1.0	3.4851	3.5778	0.0016	2.6	2.0747	2.1467	0.0011	3.4	1.4105	1.4310	0.0011	1.4	5.50	
	2.0	3.3506	3.4542	0.0017	3.0	1.9904	2.0725	0.0012	4.0	1.3602	1.3817	0.0009	1.6	5.48	
merge topology															
		node #1				node #2				node #3					
ρ	c_s^2	analytical	simulation			analytical	simulation			analytical	simulation			CPU*	
		θ	θ	δ	$\Delta\% \theta$	θ	θ	δ	$\Delta\% \theta$	θ	θ	δ	$\Delta\% \theta$		
1.0	0.5	0.3995	0.3990	0.0007	-0.1	0.5910	0.5986	0.0011	1.3	0.9904	0.9976	0.0015	0.7	1.05	
	1.0	0.3994	0.3995	0.0008	0.0	0.5890	0.5977	0.0009	1.4	0.9884	0.9972	0.0011	0.9	1.07	
	2.0	0.3992	0.3991	0.0006	0.0	0.5850	0.5966	0.0011	1.9	0.9842	0.9957	0.0012	1.2	1.08	
2.0	0.5	0.7961	0.7956	0.0010	-0.1	1.1361	1.1875	0.0010	4.3	1.9322	1.9831	0.0014	2.6	2.28	
	1.0	0.7953	0.7948	0.0013	-0.1	1.1220	1.1843	0.0012	5.3	1.9173	1.9792	0.0021	3.1	2.25	
	2.0	0.7936	0.7920	0.0010	-0.2	1.0955	1.1763	0.0013	6.9	1.8891	1.9683	0.0016	4.0	2.23	
4.0	0.5	1.5718	1.5682	0.0012	-0.2	1.9962	2.3034	0.0014	13.3	3.5681	3.8716	0.0019	7.8	4.68	
	1.0	1.5655	1.5575	0.0010	-0.5	1.9191	2.2737	0.0013	15.6	3.4845	3.8312	0.0017	9.0	4.60	
	2.0	1.5530	1.5334	0.0012	-1.3	1.7960	2.2165	0.0012	19.0	3.3490	3.7499	0.0012	10.7	4.52	

*simulation CPU time in minutes.

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

s.t:

$$\begin{aligned} x_i &\in \mathbb{N}, \forall i, \\ \alpha &\geq 0. \end{aligned}$$

Note that for any feasible vector \mathbf{x} — that is, a vector \mathbf{x} for which the constraints (2) and (3) hold — the term $\alpha(\Theta^{\min} - \Theta(\mathbf{x}))$ must be non-positive and will be a penalty of the objective function related to the difference between the threshold throughput, Θ^{\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, since removing the constraint (2) cannot increase the optimal value Z (for a detailed discussion on this issue, the reader is referred to the article by Fisher [5]).

As a way to approximately solve the BAP we propose to set the threshold throughput Θ^{\min} exactly to the total external arrival rate Λ , which will then serve as the input to the performance evaluation algorithm, described in the earlier section, which will compute the corresponding throughput $\Theta(\mathbf{x})$. Thus, under the assumption that the threshold throughput Θ^{\min} is exactly the total external arrival rate Λ , the best (highest) possible inferior limit is given by Theorem 1.

Theorem 1 *If Θ^{\min} is exactly the total external arrival rate Λ , then the highest inferior limit, $L(\alpha^*) = \max_{\alpha \geq 0} L(\alpha)$, is achieved for $\alpha^* \rightarrow \infty$.*

Proof: *It follows from $\Theta(\mathbf{x})$ being a non-decreasing function of \mathbf{x} , as it is seen in Fig. 7, and also from the Lagrangean function, $L(\alpha)$, which is the minimum of linear functions of α ,*

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

with non-negative intercepts and non-negative slopes with

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

which results in a non-decreasing convex envelopment, as it is seen in Fig. 8.

■

The best Lagrangean multiplier α , as defined by Theorem 1, is not practical because one would need that

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

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

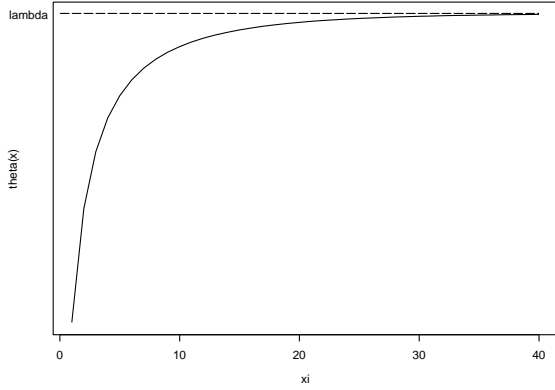


Fig. 7. Throughput *versus* buffer size.

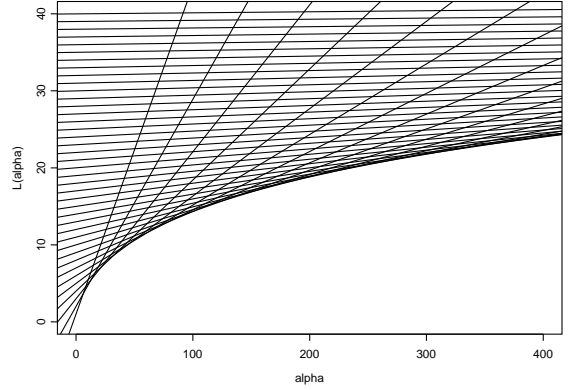


Fig. 8. Lagrangian function $L(\alpha)$.

302 $\alpha(\Theta^{\min} - \Theta(\mathbf{x})) \leq 1,$

303 because, otherwise, it would be better to spend one more unity of buffer space to some i th queue, x_i ,
 304 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
 305 corresponding α_ε as follows

306 $\alpha_\varepsilon \leq 1/(\Theta^{\min} - \Theta(\mathbf{x})),$

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

308 *5.2. Search Algorithm*

309 The Lagrangean relaxation of the BAP, $L(\alpha)$, plus an additional relaxation of the integrality constraints
 310 for x_i , is a classical unconstrained optimization problem. Among all possible algorithms to solve the BAP,
 311 a derivative free search algorithm was used, which is shown in Fig. 9, for its simplicity, and also efficiency,
 312 as it will be seen.

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

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

318 which is optimized only in relation to the first coordinate of vector \mathbf{x} , keeping fixed the remaining coordinates.

319 The process is repeated for the second coordinate and so on, until the last coordinate is reached. A completely

320 new vector $\mathbf{x}^{(n+1)}$ is obtained and compared with the previous vector $\mathbf{x}^{(1)}$. If the Euclidean distance between
 321 these two vectors is less than a pre-specified value ϵ , the algorithm stops. Otherwise, the whole process keeps
 322 running until the convergence is reached. Actually, the algorithm is a classical derivative-free direct search
 323 method.

```

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

```

Fig. 9. Algorithm for optimal buffer allocation.

324 6. Experimental Results

325 All algorithms were implemented in FORTRAN, taking advantage of the subroutines already developed
 326 for similar problems [17, 19] and are available upon request. The experiments were run for tandem, split,
 327 and merge queues, as presented in Fig. 10.

328 The arrival rates considered were $\Lambda = \Theta^{\min} = \{1.0, 2.0, 4.0\}$, the service rates, $\mu_i = 10.0, \forall i$, resulting in
 329 a system utilization $\rho = \{0.1, 0.2, 0.4\}$, combined with several values for the squared coefficient of variation,
 330 $c_s^2 = \{0.5, 1.0, 2.0\}$, and number of nodes, $|V| = \{3, 7, 15\}$. The results are presented in Tab. 2.

331 From Tab. 2, it is seen that the pattern found in the small networks essentially becomes the pattern
 332 for the large networks. Also, the optimal allocation clearly depends on the coefficient of variation, c_s^2 . The
 333 results make sense, are stable, and are symmetrical for the split and merge topologies. Note that the buffer
 334 allocation is uniform across the series topology. This type of result is similar to the uniform buffer allocation
 335 results of de Kok [4] but the well-known bowl phenomenon [9] was not obtained here. In turns out that the
 336 bowl phenomenon is only present in optimal buffer allocations constrained to a maximal number of total
 337 buffer allocated, as, for instance, in the model presented in the article by Hillier and So [9].

338 In order to see how close to optimal the generated patterns are, it is interesting to compare these results
 339 with simulation. Experiments with ARENA with 200,000 time units, 2,000 time units warm-up and 20
 340 replications were found to yield fairly stable results and short 95% confidence intervals. For all the non-

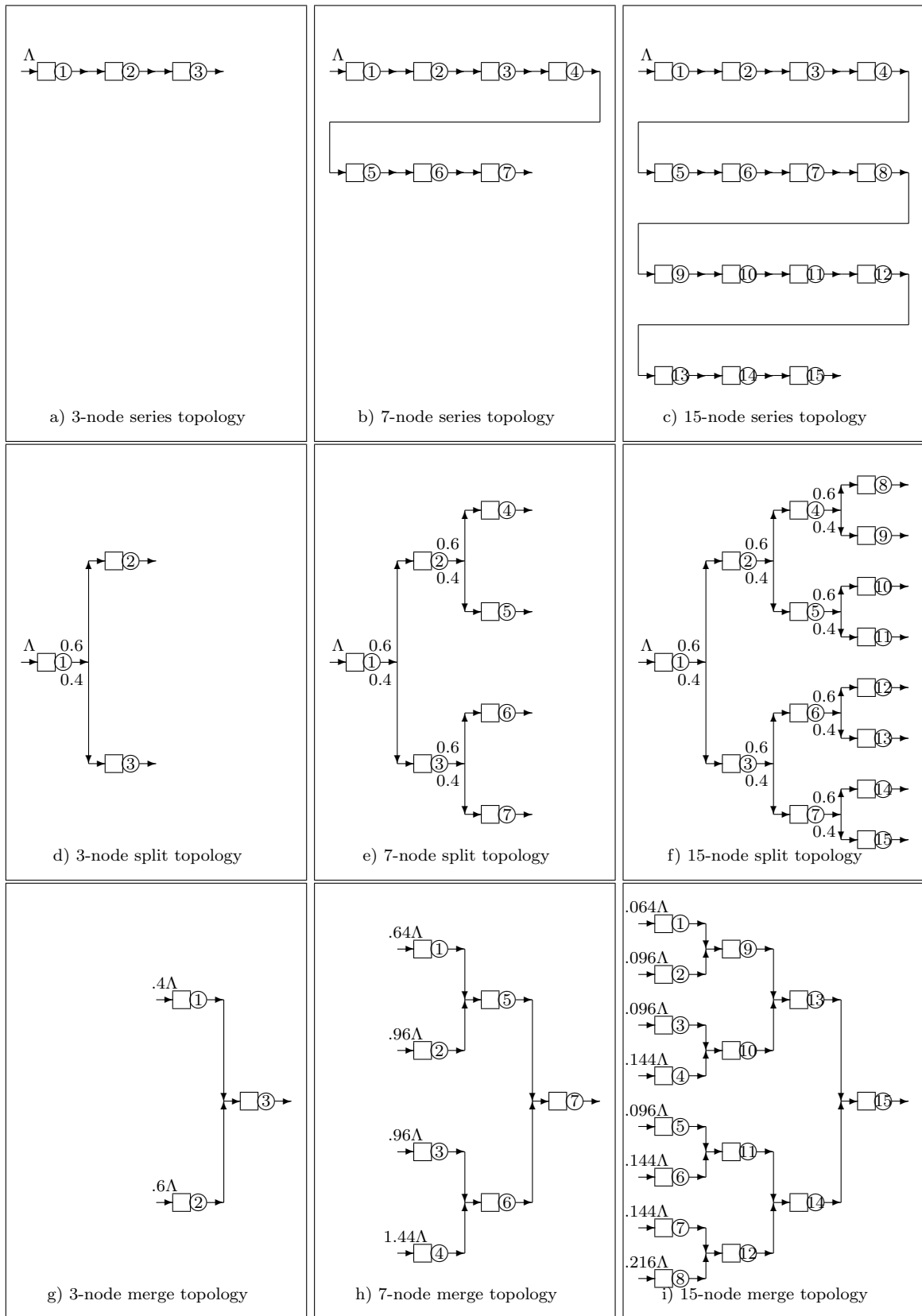


Fig. 10. Topologies tested.

Table 2

Buffer allocation results.

series topology				
ρ	$ V $	3	7	15
	c_s^2	K	K	K
0.1	0.5	(3 3 3) ^(*)	(3 3 3 3 3 3 3)	(3 3 3 3 3 3 3 3 3 3 3 3 3 3 3)
	1.0	(3 3 3)	(3 3 3 3 3 3 3)	(3 3 3 3 3 3 3 3 3 3 3 3 3 3 3)
	2.0	(4 4 4)	(4 4 4 4 4 4 4)	(4 4 4 4 4 4 4 4 4 4 4 4 4 4 4)
0.2	0.5	(5 5 5)	(5 5 5 5 5 5 5)	(5 5 5 5 5 5 5 5 5 5 5 5 5 5 5)
	1.0	(5 5 5)	(5 5 5 5 5 5 5) ^(*)	(5 5 5 5 5 5 5 5 5 5 5 5 5 5 5)
	2.0	(6 6 6)	(6 6 6 6 6 6 6)	(6 6 6 6 6 6 6 6 6 6 6 6 6 6 6)
0.4	0.5	(7 7 7)	(7 7 7 7 7 7 7)	(7 7 7 7 7 7 7 7 7 7 7 7 7 7 7)
	1.0	(8 8 8)	(8 8 8 8 8 8 8)	(8 8 8 8 8 8 8 8 8 8 8 8 8 8 8)
	2.0	(10 10 10)	(10 10 10 10 10 10 10)	(10 10 10 10 10 10 10 10 10 10 10 10 10 10 10)
split topology				
ρ	$ V $	3	7	15
	c_s^2	K	K	K
0.1	0.5	(3 3 2)	(3 3 2 2 2 2 2)	(3 3 2 2 2 2 2 2 2 2 2 1 2 1 1 1)
	1.0	(3 3 2)	(3 3 2 2 2 2 2)	(3 3 2 2 2 2 2 2 2 2 2 1 2 1 1 1)
	2.0	(4 3 2)	(4 3 2 2 2 2 2)	(4 3 2 2 2 2 2 2 2 2 2 1 2 1 1 1)
0.2	0.5	(5 4 3)	(5 4 3 3 2 2 2)	(5 4 3 3 2 2 2 2 2 2 2 2 2 2 2)
	1.0	(5 4 3)	(5 4 3 3 3 3 2)	(5 4 3 3 2 2 2 2 2 2 2 2 2 2 2)
	2.0	(6 4 3)	(6 4 3 3 3 3 2)	(6 4 3 3 3 3 2 2 2 2 2 2 2 2 2)
0.4	0.5	(7 5 4)	(7 5 4 4 3 3 3)	(7 5 4 4 3 3 3 3 3 3 3 2 3 2 2 2)
	1.0	(8 6 4)	(8 6 4 4 3 3 3)	(8 6 4 4 3 3 3 3 3 3 3 2 3 2 2 2)
	2.0	(10 6 5)	(10 6 5 5 4 4 3)	(10 6 5 5 4 4 3 3 3 3 3 2 3 2 2 2)
merge topology				
ρ	$ V $	3	7	15
	c_s^2	K	K	K
0.1	0.5	(2 3 3)	(2 2 2 2 2 3 3)	(1 1 1 2 1 2 2 2 2 2 2 2 2 3 3)
	1.0	(2 3 3)	(2 2 2 2 2 3 3)	(1 1 1 2 1 2 2 2 2 2 2 2 2 3 3)
	2.0	(2 3 4)	(2 2 2 2 2 3 4)	(1 1 1 2 1 2 2 2 2 2 2 2 2 3 4)
0.2	0.5	(3 4 5)	(2 2 2 3 3 4 5)	(2 2 2 2 2 2 2 2 2 2 2 3 3 4 5)
	1.0	(3 4 5)	(2 2 3 3 3 4 5)	(2 2 2 2 2 2 2 2 2 2 2 3 3 4 5)
	2.0	(3 4 6)	(2 3 3 3 3 4 6)	(2 2 2 2 2 2 2 2 2 3 3 3 3 4 6)
0.4	0.5	(4 5 7)	(3 3 3 4 4 5 7)	(2 2 2 3 2 3 3 3 3 3 3 4 4 5 7)
	1.0	(4 6 8)	(3 3 3 4 4 6 8)	(2 2 2 3 2 3 3 3 3 3 3 4 4 6 8)
	2.0	(5 6 10)	(3 4 4 5 5 6 10)	(2 2 2 3 2 3 3 3 3 4 4 5 5 6 10)

(*) Earmarked experiments checked by simulation (see Table 3).

341 exponential service times, a two-stage gamma distribution was used to capture the general service times
342 with non-unit c_s^2 . The results for some of the series queues are seen in Tab. 3.

343 The result for a three-node series topology was analyzed in more detail. Note that the best solutions
344 correspond closely to the lowest $L(\alpha)$ (see Tab. 3, in boldface). The general conclusion is that optimization

Table 3

Simulation results for tandem queues.

Λ	c_s^2	$ V $	\mathbf{x}	$\Theta(\mathbf{x})$		$L(\alpha)$	CPU*
				average	δ		
1.0	0.5	3	(2 2 2)	0.9928	0.0011	13.19	1.68
			(2 2 3)	0.9928	0.0011	14.18	1.67
			(2 3 3)	0.9928	0.0011	15.17	1.68
			(3 3 3)	0.9994	0.0012	9.59 ^{†,‡}	1.47
			(3 3 4)	0.9999	0.0009	10.07	1.70
			(3 4 4)	1.0000	0.0009	11.00	1.70
			(4 4 4)	1.0000	0.0013	12.00	1.70
2.0	1.0	7	(3 3 3 3 3 3 3)	1.9861	0.0014	34.90	7.27
			(4 4 4 4 4 4 4)	1.9966	0.0010	31.40 [†]	7.28
			(5 5 5 5 5 5 5)	1.9994	0.0013	35.60 [‡]	7.28
			(6 6 6 6 6 6 6)	1.9996	0.0016	42.40	7.30
			(7 7 7 7 7 7 7)	2.0001	0.0021	48.90	7.65

*CPU time for simulation in Arena in minutes.

[†]Best solution via simulation.[‡]Best solution via optimization algorithm.

345 algorithm tends to allocate more space than necessary to ensure the desired performance. Also noticeable
346 is that the CPU time for the simulations grows quickly as the number of node in the network increases
347 indicating that the simulation may not be the most efficient tool for optimizing but it is certainly useful for
348 assessing the quality of solutions via other methods.

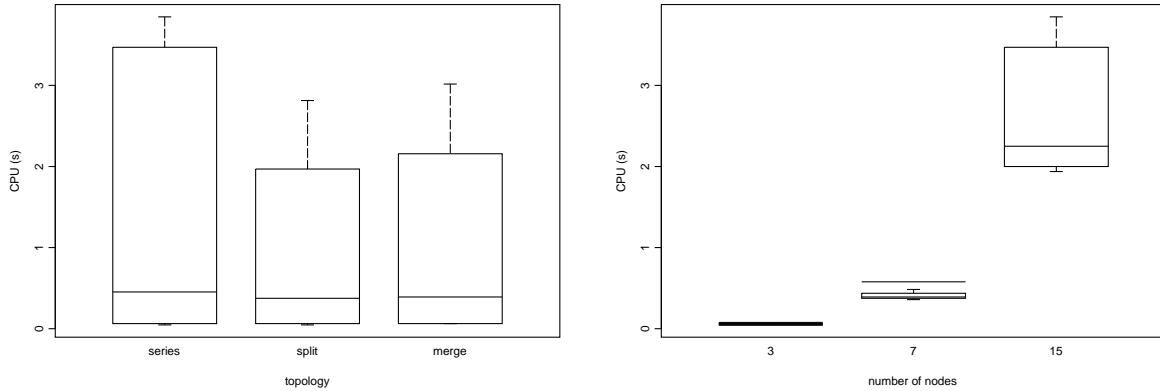


Fig. 11. Running times in function of the topology and number of nodes.

349 As a final note on the computational performance of the algorithm, it is important to know how it will
350 behave with small changes in the input. The running times, in seconds, for the cases studied in Tab. 2

351 are presented in Figure 11, reporting boxplots in function of the topology and the number of nodes in the
352 network.

353 The running times seem to be independent on the topology but they certainly will increase with the
354 number of nodes. This increase, although not too drastic, is followed by an increase in the variability, which
355 indicates that the running times may be less predictable for large networks.

356 7. Summary and Conclusions

357 One major difficulty in dealing with the buffer allocation problem (BAP), in general, and for $M/G/1/K$
358 queues, in particular, is to find good approximations for the performance measures of interest. The BAP is
359 much more difficult when queues are configured in networks, in which blocking after service frequently occurs.
360 In this paper, some of the most effective approximations for the blocking probability, a crucial performance
361 measure for the BAP treated here, were extensively compared. The approximation by Smith [17] seemed to
362 be the most accurate for the cases tested and was used here to solve the BAP.

363 The algorithm proposed is based on a Lagrangean relaxation, a technique that has been proved efficient in
364 solving optimization problems with complicated constraints. The Lagrangean relaxation enables one to avoid
365 hard optimization formulations by relaxing complicating constraints and including them into the objective
366 function as a penalty. Important properties of the relaxed problem were derived, which made possible the
367 development of a search algorithm, considerably simpler than the algorithm previously published for the
368 same problem by Smith and Cruz [19]. In comparison with the exact simulation results, the algorithm seemed
369 to produce very fast and accurate solutions and can be used in the design of production systems.

370 Topics for future research in the area include extensions to systems that have loops, such as systems with
371 captive pallets and fixtures. Also of interest is the study of algorithms for multi-server general service time
372 queueing networks.

373 *Acknowledgements*

374 The research of prof. Frederico Cruz has been partially funded by the CNPq (*Conselho Nacional de*
375 *Desenvolvimento Científico e Tecnológico*) of the Ministry for Science and Technology of Brazil, grants
376 201046/1994-6, 301809/1996-8, 307702/2004-9, 472066/2004-8, and 472877/2006-2, by the FAPEMIG (*Fun-*
377 *dação de Amparo à Pesquisa do Estado de Minas Gerais*), grants CEX-289/98 and CEX-855/98, and PRPq-
378 UFMG, grant 4081-UFMG/RTR/FUNDO/PRPq/99.

379 **References**

- 380 [1] Altioik, T., Stidham, S., 1983. The allocation of interstage buffer capacities in production lines. IIE
381 Transactions 15, 251–261.
- 382 [2] Baker, K. R., Powell, S., Pyke, D., 1990. Buffered and unbuffered assembly systems with variable
383 processing times. Journal of Manufacturing and Operations Management 3, 200–223.
- 384 [3] Cruz, F. R. B., Smith, J. MacGregor, 2006. Approximate analysis of $M/G/c/c$ state-dependent queueing
385 networks. Computers & Operations Research (in press).
386 URL <http://dx.doi.org/10.1016/j.cor.2005.09.006>
- 387 [4] De Kok, A. G., 1990. Computationally efficient approximations for balanced flowlines with finite inter-
388 mediate buffers. International Journal of Production Research 28, 401–419.
- 389 [5] Fisher, M. L., 1985. An application oriented guide to lagrangean relaxation. Interfaces 15, 10–21.
- 390 [6] Gelenbe, E., 1975. On approximate computer system models. Journal of the ACM 22 (2), 261–269.
- 391 [7] Gross, D., Harris, C. M., 1998. Fundamentals of Queueing Theory, 3rd Edition. John Wiley & Sons,
392 New York, NY, USA.
- 393 [8] Hillier, F., So, K., 1991. The effect of the coefficient of variation of operation times on the allocation of
394 storage space in production line systems. IIE Transactions 23 (2), 198–206.
- 395 [9] Hillier, F. S., So, K. C., 1995. On the optimal design of tandem queueing systems with finite buffers.
396 Queueing Systems 21, 245–266.
- 397 [10] Kelton, D., Sadowski, R. P., Sadowski, D. A., 2001. Simulation with Arena. MacGraw Hill College Div.,
398 New York, NY, USA.
- 399 [11] Kerbache, L., Smith, J. MacGregor, 2000. Multi-objective routing within large scale facilities using open
400 finite queueing networks. European Journal of Operational Research 121 (1), 105–123.
- 401 [12] Kimura, T., 1996. Optimal buffer design of an $M/G/s$ queue with finite capacity. Communications in
402 Statistics - Stochastic Models 12 (1), 165–180.
- 403 [13] Kimura, T., 1996. A transform-free approximation for the finite capacity $M/G/s$ queue. Operations
404 Research 44 (6), 984–988.
- 405 [14] Kubat, P., Sumita, U., 1985. Buffers and backup machines in automatic transfer lines. International
406 Journal of Production Research 23 (6), 1259–1280.
- 407 [15] Lemaréchal, C., 2003. The omnipresence of Lagrange. 4OR 1, 7–25.
- 408 [16] Makens, M. C. S. P. K., 1992. Queueing models for performance analysis: Selection of single station
409 models. European Journal of Operational Research 58 (1), 123–145.
- 410 [17] Smith, J. MacGregor, 2002. $M/G/c/k$ blocking probability models and system performance. Perfor-
411 mance Evaluation 52, 237–267.
- 412 [18] Smith, J. MacGregor, Chikhale, N., 1995. Buffer allocation for a class of nonlinear stochastic knapsack

- 413 problems. *Annals of Operations Research* 58, 323–360.
- 414 [19] Smith, J. MacGregor, Cruz, F. R. B., 2005. The buffer allocation problem for general finite buffer
415 queueing networks. *IIE Transactions on Design & Manufacturing* 37 (4), 343–365.
- 416 [20] Smith, J. MacGregor, Daskalaki, S., 1988. Buffer space allocation in automated assembly lines. *Opera-*
417 *tions Research* 36 (2), 343–358.
- 418 [21] Soyster, A. L., Schmidt, J. W., Rohrer, M. W., 1979. Allocation of buffer capacities for a class of fixed
419 cycle production lines. *AIIE Transactions* 11 (2), 140–146.
- 420 [22] Spinellis, D., Papadopoulos, C. T., Smith, J. MacGregor, 2000. Large production line optimization using
421 simulated annealing. *International Journal of Production Research* 38 (3), 509–541.
- 422 [23] Tijms, H. C., 1986. *Stochastic Modeling and Analysis: A Computational Approach*. John Wiley & Sons,
423 New York, MA, USA.
- 424 [24] Yamashita, H., Altiok, T., 1998. Buffer capacity allocation for a desired throughput in production lines.
425 *IIE Transactions* 30 (10), 883–892.
- 426 [25] Yamashita, H., Onvural, R., 1994. Allocation of buffer capacities in queueing networks with arbitrary
427 topologies. *Annals of Operations Research* 48, 313–332.