Buffer and Server Allocation in General Multi-Server Queueing Networks

T. Van Woensel[*†] R. Andriansyah[‡] F. R. B. Cruz[§¶]
J. MacGregor Smith[‖] L. Kerbache[**]
t.v.woensel@tue.nl r.andriansyah@tue.nl frc@cs.nott.ac.uk jmsmith@ecs.umass.edu kerbache@hec.fr

September 23, 2009

Abstract — **This paper deals with the joint optimization of the number of buffers and servers, an important issue since buffers and servers represent a significant amount of investment for many companies. The joint buffer and server optimization problem (BCAP) is a non-linear optimization problem with integer decision variables. The performance of the BCAP is evaluated by a combination of a two-moment approximation (developed for the performance analysis of finite general-service queues) and the generalized expansion method (a well-known method for performance analysis of acyclic networks of finite queues). A standard non-linear optimization package is used to optimize the BCAP for a large number of experiments. A comprehensive set of numerical results is presented. The results show that the methodology is capable of handling the trade-off between the number of servers and buffers, yielding better throughputs than previously published studies. Also, the importance of the squared coefficient of variation of the service time is stressed, since it strongly influences the approximate optimal allocation.**

Keywords — Joint buffer and server allocation; general service; queues; networks.

1 Introduction and Motivation

Optimization of large scale manufacturing systems and complex production lines has been the focus of numerous studies (Tempelmeier, 2003), given that both buffers and servers represent a significant amount of investment during the design phase of manufacturing systems. Obviously, the exact number of buffers and servers allocated to the different production steps will also lead to differences in the operational characteristics, such as throughput, work-in-process (WIP), cycle times, etc. It is thus of high importance to provide decision makers tools that allow not only for the evaluation of the performance of a given buffer and server combination, but also for the optimization of these combinations.

In this paper, we focus on the **joint buffer and server optimization problem** (BCAP) in a setting of restricted $M/G/c/K$ queueing networks. Queueing networks are commonly used to model such complex systems (Suri, 1985). Here we model the joint buffer and server optimization problem using queueing networks which are optimized for the allocated number of buffers and servers at each node in the network. These queueing networks are characterized by Markovian external arrivals M with rate Λ, general service times G with rate μ and a squared coefficient of variation s^2, multiple servers c, and total capacity K, *including* those entities in service (see Fig. 1).

Moreover, since we deal with restricted queueing networks, there is a finite capacity in each node, referred to as the total buffer capacity of size K_j. That is, a finite node j can only hold entities up to a certain quantity K_j *including* those entities in service. The buffer capacity at finite node j causes blocking to occur when the arriving quantity to node j exceeds its buffer capacity K_j ((see Buzacott and Shanthikumar, 1993). As a consequence, each node in the network might be affected by events at other nodes, leading to the phenomena of blocking and starvation (Perros, 1994). Because of this, finite buffer queueing networks might eventually suffer performance reduction, which can be measured via the overall throughput Θ of the network.

We obtain the performance measures of the manufacturing system via a two-moment approximation developed by Smith (2003) for performance analysis of finite general service time *single* queues and combined with the generalized expansion method (GEM), a well-known technique developed by Kerbache and

[*]Corresponding author.
[†]School of Industrial Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands.
[‡]Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands.
[§]School of Computer Science and Information Technology, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK.
[¶]On sabbatical leave from the Department of Statistics, Federal University of Minas Gerais, Belo Horizonte, MG, Brazil.
[‖]Department of Mechanical and Industrial Engineering, University of Massachusetts Amherst, Massachusetts 01003, USA.
[**]HEC School of Management, Paris, France.

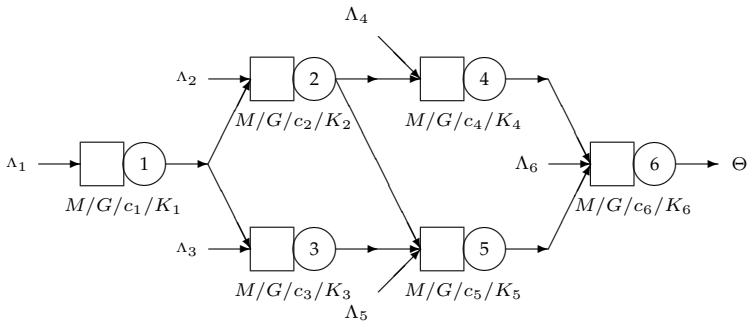

Figure 1: An arbitrarily configured queueing network

Smith (1987, 1988), for approximate performance analysis of *acyclic networks* of finite queues. The accuracy of the two-moment approximation combined with the GEM has previously been demonstrated (Smith, 2004) but to the authors' best knowledge this is the first time that these two performance evaluation tools have been conjointly used to simultaneously optimize buffer and server allocation in finite general-service queueing acyclic networks. In other words, the network configuration is not restricted to a tandem single-server production line; any possible acyclic multi-server configuration can be analyzed and optimized.

More specifically, we consider arbitrary acyclic configured queueing networks, including series, merge, split topologies, and any possible combination that can be evaluated using the GEM. For a given buffer and server combination, the two-moment approximation and the GEM thus provide the performance measure. Here the throughput, Θ , is the preferred measurement but other performance measures may also be of interest, such as the average WIP, cycle time, system utilization, and average queue lengths. However, in the literature, the throughput has been the most commonly used metric for evaluating the performance of finite buffer queueing networks (Down and Karakostas, 2008). The two-moment approximation and the GEM are combined with a non-linear optimization methodology based on Powell's search method (Himmelblau, 1972) and are used to optimize any large and arbitrarily acyclic configured networks including serial, merge, and split topologies, as well as the combination of any of these topologies, as we shall see shortly.

The optimization problem presented here is a difficult non-linear integer programming problem. To validate the optimization methodology, the results are compared both with complete enumeration and with simulation. These comparisons attest to the quality of the solutions provided by the proposed methodology, showing that the methodology provides sound allocations for both buffers and servers. Moreover, comparing the results to similar settings obtained from the literature allows quantification of the potential improvement over previously published results. The methodology provided in this paper can be used especially for system-level design of production/manufacturing lines or in other relevant environments (see examples and references in Suri et al., 1993). Reasonable estimates are obtained for the requirements and allocations for the optimal number of machines and waiting spaces for the WIP, given the demand, routing of jobs, machine processing rate, and variability.

Relevant literature

Although a great deal is known about improving production lines (for instance, Li and Meerkov, 2009, have considered simultaneous work and buffer allocation and obtained closed formulas for the Bernoulli machine reliability model), it is interesting to observe that little literature is available on the simultaneous optimization of both buffers and servers. To the authors' best knowledge, no papers exist that discuss the joint optimization of buffers and servers in acyclic networks of finite general-service time queues. Most papers dealing with the BCAP have focused on pure Markovian systems (both in the arrival distributions as well in the service distributions) and on serial or tandem lines (see for example Hillier and So, 1995, Spinellis et al., 2000).

Shanthikumar and Yao (1987) looked into the simultaneous optimal allocation of both servers and buffer capacity for a single station system (*i.e.*, no network was analyzed due to the computational complexity) considering *closed* queueing networks. Hillier and So (1995) and Spinellis et al. (2000) studied open networks of finite multi-server Markovian queues. Their experiments are limited to series in which the first station in the line is never starved (hence, infinite buffers for the first node) and the last node is never blocked. In their paper, Hillier and So (1995) considered joint optimization of finite multi-server Markovian queueing networks using a complete enumeration strategy. Using this methodology, however, they generated results only for small, relatively simple queueing networks. This is due to the fact that the possible buffer and server allocations increase exponentially with increasing number of nodes, maximum number of servers, and maximum number of buffers. Spinellis et al. (2000) extended the work of

Hillier and So (1995) by combining simulated annealing and the GEM to optimize the performance of production lines. Spinellis et al. (2000) were able to optimize a large production line (up to 70 stations with 140 buffers and servers) in reasonable computation times.

The BCAP can be reduced to a pure buffer allocation problem (BAP) or a pure server allocation problem (CAP). In both models, the servers for BAP and buffers for CAP, respectively, are assumed as as given. In the pure BAP **buffers** are inserted between servers to limit the propagation of disruptions, which increases the average throughput rate of the network (Hillier and Hillier, 2006). In practice, inclusion of buffers requires additional capital investment (floor space for manufacturing environments), which may be expensive. Buffering also increases in-process inventory. If the buffers are too large, the WIP inventory and capital costs incurred will outweigh the benefit of increased productivity. If the buffers are too small, the servers will be under-utilized or demand will not be met (Gershwin and Schor, 2000). It is thus essential to set the buffer sizes such that the desired performance can be realized (for a list of recently published papers on the BAP, see Smith et al., 2010).

The pure server allocation problem (CAP) has also been well researched. There is a vast amount of literature on the optimal allocation of **servers**. Many studies have been done considering single nodes, open and closed networks, infinite and finite buffer waiting room, and exponential service systems (for an overview, see Smith et al., 2009).

Structure of the paper

The paper is organized as follows. In Sec. 2, we mathematically define the BCAP. Next, the performance evaluation and the optimization tools are described in Sec. 3. We use a two-moment approximation and the GEM to obtain the relevant performance measures and a classical non-linear search method to optimize this type of system. In Sec. 4, we elaborate on the experimental results obtained for a large number of situations (*i.e.*, tandem, split, and merge cases). We analyze the performance and optimize the system both analytically and by simulation. Sec. 5 concludes this paper with final remarks and topics for future research.

2 Model Formulation and Implications

The BCAP is formulated as a single-objective optimization problem Smith et al. (rather than a multi-objective formulation; see also 2009). In our formulation, we assume that decision makers want to minimize their investment in buffers and servers, but want to assure a certain minimum service level for their customers. This service level is measured here as throughput, *i.e.*, how much of the arrivals has gone through the network. Obviously, many other formulations could be used, but the proposed model formulation has been used successfully in the past for both the CAP and the BAP models (see for example Smith and Cruz, 2005, for a similar model formulation).

2.1 Model Formulation

The model optimizes the number of buffers and servers such that the resulting throughput is greater than a predefined threshold throughput. We define a queueing network as a digraph $G = (N, A, P)$ in which N is the set of nodes, A is the set of arcs, and P is the set of respective routing probabilities. The BCAP is mathematically formulated as follows.
(BCAP):

$$Z = \min \left[\sum_{\forall i \in N} \omega_i c_i + \sum_{\forall i \in N} (1 - \omega_i) B_i \right], \qquad (1)$$

subject to:

$$\theta(\mathbf{c}, \mathbf{B}) \geq \theta^{\min}, \qquad (2)$$

$$c_i \in \{1, 2, \dots\}, \qquad \forall i \in N, \qquad (3)$$

$$B_i \in \{0, 1, \dots\}, \qquad \forall i \in N, \qquad (4)$$

in which c_i is the number of servers at node i, B_i is the number of pure buffers (*i.e.*, *excluding* those in service, and $K_i = c_i + B_i$ is the total capacity at node i), $\theta(\mathbf{c}, \mathbf{B})$ is the resulting throughput as a function of the server and buffer allocation ($\mathbf{c}$ and $\mathbf{B}$, respectively), $\theta^{\min}$ is a target threshold throughput, and ω_i is a relative cost variable ($0 \leq \omega_i \leq 1$).

Note that if $B_i = 0$, we have a zero-buffer node (*i.e.*, only including servers). Bufferless networks occur throughout a number of real-life physical systems in the semi-process and process industries (refer to Fransoo and Rutten, 1994, for more general information on process industries). A zero-buffer production environment might be necessary due to the processing technology of the product itself, or simply due to the absence of any intermediate storage capacity between two consecutive operations of a job. Examples of real-life zero-buffer networks are available in the literature. For instance, Hall and Sriskandarajah (1996) describe a steel production process and Ramudhin and Ratliff (1995) describe a condiment manufacturer producing mayonnaise and various types of salad dressing. These authors also present a long list of references for additional real-life zero-buffer examples.

Finally, observe that in the cost function we set the weights ω_i as *dependent* on the specific node i, $\forall i \in N$, because in many practical situations this is true, which leads to specific node weights. The methodology proposed is able to handle different weights per node, but we include numerical experiments in the paper only for equal weights for all nodes to avoid an explosion of possible experiments.

2.2 *Analysis and Implications of the proposed model*

In objective function Eq. (1), we assign a cost of ω_i to servers and $(1-\omega_i)$ to buffers. We can modify the value of ω_i, such that $0 < \omega_i < 1$, to reflect the relative cost of servers versus buffers. As ω_i is decreased, the cost of servers will become relatively lower than that of buffers. Alternatively, when the value of ω_i is increased, the servers become more costly relative to the buffers. In this way, we evaluate whether different pricing of servers and buffers results in a significantly different buffer and server allocation. It is worthwhile to mention that if $\omega_i = 0,\ \forall i \in N$, the above problem reduces to the pure BAP and if $\omega_i = 1,\ \forall i \in N$, the pure CAP is obtained.

One way to decrease the complexity of the BCAP is by incorporating the complicating constraints into the objective function via Lagrangian relaxation. A comprehensive overview of this method can be found in Lemaréchal (2001, 2007). Thus, for this particular problem the difficult constraint (2) may be relaxed by means of a Lagrangian variable $\alpha > 0$. The relaxed BCAP (RBCAP) is then formulated as follows:

$$Z_\alpha = \min \left[\sum_{\forall i \in N} \omega_i c_i + \sum_{\forall i \in N} (1-\omega_i) B_i + \alpha\Big(\theta^{\min} - \theta(\mathbf{c}, \mathbf{B})\Big) \right], \quad (5)$$

subject to constraints (3) and (4).

Note that in the relaxed formulation, RBCAP, the term $\alpha\Big(\theta^{\min} - \theta(\mathbf{c}, \mathbf{B})\Big)$ is always non-positive, for any feasible solution of the original formulation, BCAP. That is, if constraints (2), (3) and (4) are to be met, it must hold that $\alpha\Big(\theta^{\min} - \theta(\mathbf{c}, \mathbf{B})\Big) \leq 0$. Because $Z_\alpha \leq Z$, we will use Z_α as a lower bound on the optimal objective value of Z and therefore we would like to have this lower bound as tight as possible. To solve such a simultaneous server-buffer allocation problem, we set the threshold throughput $\theta^{\min}$ equal to the external arrival rate, Λ. A two-moment approximation and the GEM will then approximate the resulting throughput, $\theta(\mathbf{c}, \mathbf{B})$, given the particular server and buffer configuration. Note that $alpha$ gives the cost of not meeting the constraint. Following the rationale given by Cruz et al. (2008), we set the Lagrangian variable α equal to 10^3.

It is interesting to see how the objective function behaves when adding a server versus adding a buffer during the optimization. Let us assume that we start from a zero-buffer *single* node with one server (*i.e.*, $N = \{1\}$, $K = 1$, $B = 0$, and $c = 1$), submitted to an external arrival rate $\Lambda = 1.0$, service rate $\mu = 2.0$, squared coefficient of variation of the service time distribution $s^2 = \{0.5, 1.0, 1.5\}$, and equal relative costs, $\omega = 0.5$ (for the sake of simplicity, here we will not use index i from ω_i because we have here a single $M/G/c/K$ node).

Fig. 2 gives the percentage increase of adding either a server (adding one to four servers compared to the base case) or a buffer (adding one to seven buffers compared to the base case) to the zero-buffer base situation. In this way, we analyze the value of adding buffers and servers to a single node, *i.e.*, such as we would analyze when/if a multi-server node is better than a multi-buffer node.

It is clear that in this case the first added buffer or first added server gives the largest contribution to the throughput value, which is limited by the arrival rate λ. Note that the addition of the first extra server gives an increase in throughput of 13.4% to 18.6% depending upon the coefficient of variation s^2, while the first added buffer gives only 8.2% to 9.2% increase. It is important to mention that to achieve the same increase in throughput by using only buffers, we need five to six extra buffer spaces rather than only one server space. This example does not yet consider the price of the buffer versus the price of the server, which will also be an important driver in choosing between both. Based on the above analysis, if costs are equal ($\omega = 0.5$), a server will be preferred due to the higher increase in throughput. Clearly, depending upon the specific value for the cost ω it will be preferable to add either buffers or servers. Moreover, it is clear that if a buffer is equally as costly as a server, a server will always be preferred, since the latter will also act as a buffer but is also productive in terms of throughput. In the extreme (less realistic) case where servers are inexpensive and buffers expensive, *i.e.*, $\omega \to 0$, then only zero-buffer systems will be preferred.

Making use of the above insights with regards to the buffer/server trade-off and taking into account the relative prices of buffers versus servers, the following theorem holds:

Theorem: *If the marginal value of an extra buffer is less than*

$$\frac{(1-\omega)}{\alpha}$$

a zero-buffer system will be preferred.

Proof: *To show this theorem, let us compare the objective function of a zero-buffer system with a non-zero buffer system. The objective function value of a zero-buffer system F^{ZB} can be obtained as follows (in this case $B = 0$, and θ^{ZB} is the throughput of the zero-buffer system):*

$$\begin{aligned} F^{\mathrm{ZB}} &= \omega c + (1-\omega)B + \alpha\Big(\theta^{\min} - \theta^{\mathrm{ZB}}\Big) \\ &= \omega c + \alpha\Big(\theta^{\min} - \theta^{\mathrm{ZB}}\Big). \end{aligned} \quad (6)$$

The non-zero buffer system F^{NZB} is given as follows (in this case, θ^{NZB} is the throughput of the non-zero buffer system):

$$F^{\mathrm{NZB}} = \omega c + (1-\omega)B + \alpha\Big(\theta^{\min} - \theta^{\mathrm{NZB}}\Big). \quad (7)$$

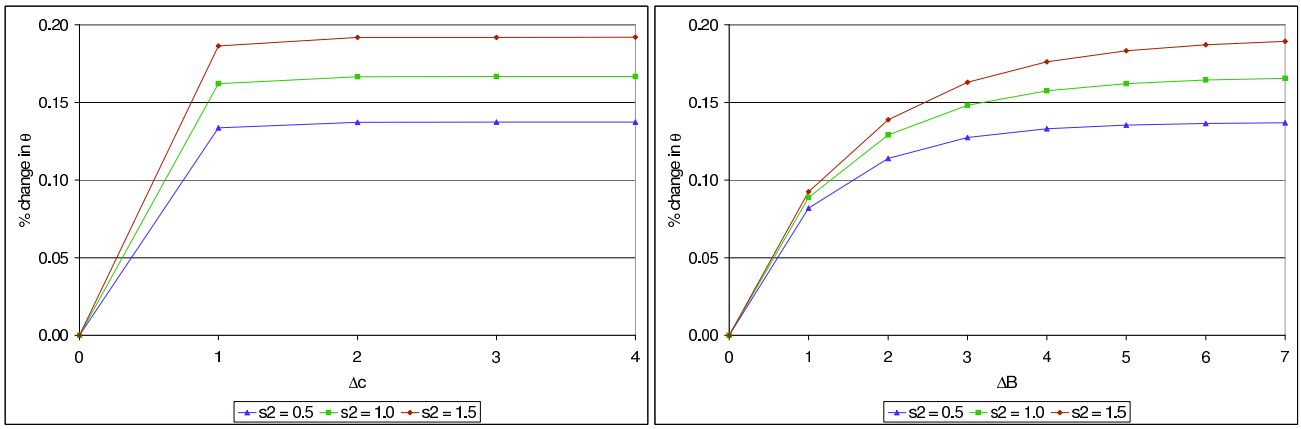

Figure 2: Throughput increase versus added number of buffers and servers

One is indifferent between a zero-buffer system or a non-zero buffer system if the difference between both objective functions values is equal to zero, i.e. $F^{\mathrm{ZB}} - F^{\mathrm{NZB}} = 0$. *Using Eqs. (6) and (7), we obtain the following expressions:*

$$\begin{aligned} F^{\mathrm{ZB}} - F^{\mathrm{NZB}} &= 0 \Rightarrow \\ \left[\omega c + \alpha\left(\theta^{\min} - \theta^{\mathrm{ZB}}\right)\right] - \left[\omega c + (1-\omega)B + \right. & \\ \left. \alpha\left(\theta^{\min} - \theta^{\mathrm{NZB}}\right)\right] &= 0 \Rightarrow \\ (1-\omega)B + \alpha\left(\theta^{\mathrm{ZB}} - \theta^{\mathrm{NZB}}\right) &= 0. \quad (8) \end{aligned}$$

Defining $\Delta B = B - 0$ *and* $\Delta\theta = \left(\theta^{\mathrm{NZB}} - \theta^{\mathrm{ZB}}\right)$, *Eq. (8) can then be rewritten as:*

$$\begin{aligned} (1-\omega)\Delta B - \alpha\Delta\theta &= 0 \Rightarrow \\ \Delta\theta &= \frac{(1-\omega)}{\alpha}\Delta B \Rightarrow \\ \frac{\Delta\theta}{\Delta B} &= \frac{(1-\omega)}{\alpha}. \end{aligned}$$

■

This theorem says that if the marginal gain on throughput of one extra unit of buffer, $\frac{\Delta\theta}{\Delta B}$, is greater than $\frac{(1-\omega)}{\alpha}$, a non-zero buffer system will be better; similarly, when $\frac{\Delta\theta}{\Delta B} < \frac{(1-\omega)}{\alpha}$, the zero- buffer system will be preferred. Interestingly, this cut-off point is equal to $\frac{(1-\omega)}{\alpha}$ (and not zero). This means that although the throughput might be higher, the other system might still be preferred due to the relative price relationship of ω versus α and depending upon the change in B. Consequently, the marginal gain on throughput of one extra unit of buffer should be significantly larger (compared to $\frac{(1-\omega)}{\alpha}$) to prefer a non-zero buffer system.

2.3 *A word on the complexity of the model*

It is worthwhile to state that the model described above is a difficult non-linear integer programming problem. For a capacity $K \leq 3$, we obtain the following possible (c, B) combinations which are all candidate solutions: $\{(1,0),(1,1),(1,2),(2,0),(2,1),(3,0)\}$. Solutions where $B = 0$ are the zero-buffer combinations; any other solution includes buffers. In general, it is easy to show that given the maximum capacity K at a node, the number of combinations that must be evaluated at that node is equal to $\sum_{i=1}^{K}(K - i + 1)$. After some rewriting this reduces to $K(K+1)/2$. With regards to these possible combinations, each node can be seen as independent of the other nodes. Consequently, for a network with N nodes, the number of combinations involved goes to $\left[K\left(K+1\right)/2\right]^N$.

Clearly, the solution space grows exponentially in the number of nodes, but not in the capacity of each node. The worst-case complexity of the solution space can thus be written as $\mathcal{O}(K^{2N})$. This shows that a complete enumeration, certainly for larger real-life networks, is infeasible. Note that one could reduce the number of combinations if the number of servers at each node is reduced (*i.e.*, not up to capacity K). This would, however, mean that some zero-buffer combinations would not be evaluated. Based on the analysis in the previous section, this could prove to be dangerous and lead to sub-optimal solutions. Finally, it is worth mentioning that the complexity of the solution space is exponentially increasing in the number of network nodes rather than in the characteristics of the node (*i.e.*, the capacity K). For an arbitrary topology, the optimal topology problem (*i.e.*, the design of the network) is shown to be $\mathcal{NP}$-hard ((Johnson et al., 1978), and the same is conjectured (see Smith and Daskalaki, 1988) for the resource allocation problems (*i.e.*, the optimal allocation of the scarce resources in the network).

3 Performance Evaluation and Optimization

In the relaxed problem, RBCAP, the performance of the network is measured via the throughput θ. This throughput is obtained with a two-moment approximation and the GEM. The optimization with regards to the

number of buffers, servers and throughput will be done via a standard non-linear search method.

3.1 Performance Evaluation with the GEM

The GEM is an effective and robust approximation technique to measure the performance of open finite queueing networks. Developed by Kerbache and Smith (1987, 1988), the GEM has become an appealing approximation technique for performance evaluation of queueing networks due to its accuracy and relative simplicity. Moreover, exact solutions to performance measurement are restricted only to very simple networks and simulation requires a considerable amount of time.

The GEM is basically a combination of two approximation methods, namely repeated trials and node-by-node decomposition. To evaluate the performance of a queueing network, the GEM first divides the network into single nodes with revised service and arrival parameters. Blocked customers are registered into an artificial 'holding node' and are repeatedly sent to this node until they are serviced. The addition of the holding node *expands* the network and transforms the network into an equivalent Jackson network, where each node can be solved independently. Generally, the GEM assumes a type I blocking that is commonly referred to as transfer blocking. This type of blocking occurs when the service of a job is completed at a certain node but cannot proceed to the next node because the queue is full. This condition is prevalent in most production and manufacturing, transportation, and other similar systems.

The effectiveness of GEM as a performance evaluation tool has been presented in many papers, including Kerbache and Smith (1987, 1988, 2000), Jain and Smith (1994), Smith (2003), and Andriansyah et al. (2009). The GEM, however, cannot handle feedback loops such as those found in semiconductor manufacturing, for example. This creates a dependency in the arrival patterns that is hard to tackle in this type of queueing network analysis. Here we will present only a high-level overview of the method. For more detailed information and applications of the GEM, the reader is referred to, *e.g.*, Kerbache and Smith (1987, 1988).

There are three main stages in the GEM, (I) a network reconfiguration, (II) a parameter estimation, and (III) a feedback elimination. The notation for the GEM presented in Table 1 will be used throughout the paper. The steps are described as follows.

Table 1: Basic network notation

Variable	Description
Λ	external Poisson arrival rate to the network
λ_j	Poisson arrival rate to node j
$\tilde{\lambda}_j$	effective arrival rate to node j
μ_j	exponential mean service rate at finite node j
$\tilde{\mu}_j$	effective service rate at finite node j due to blocking
p_K	blocking probability of finite queue of size K
p'_K	feedback blocking probability in the expansion method
h	the artificial holding node created in the GEM
c	number of servers
B_j	buffer capacity at node j *excluding* those in service
K_j	buffer capacity at node j *including* those in service
N	set of nodes in the network
ρ	$\lambda/(\mu c)$ = traffic intensity
θ	mean throughput rate
s^2	squared coefficient of variation of the service time distribution

Stage I: Network reconfiguration

For each finite node in the queueing network, an artificial node is created to register the blocked jobs. By introducing such artificial nodes, we also create new routing probabilities in the network. The result of network reconfiguration can be seen from Figure 3.

There are two possible states of the finite node, namely *saturated* and *unsaturated*. Arriving jobs will try to access the finite node j. With a probability of $(1-p_K)$, the job will find the finite node unsaturated, so that the job will enter the queue and eventually get serviced. However, if the finite node is saturated (with a probability of p_K), then the job will be directed to the artificial holding node h where it will get delayed. The delay at the artificial node is modeled using an $M/G/\infty$ queue, representing delay time without queueing. Afterward, the blocked job will try to re-enter the finite queue with a success probability of $(1-p'_K)$. There is a probability of p'_K that the blocked job still finds the finite node saturated and thus will be directed again to the artificial node h. This process repeats until the blocked job is able to enter the finite node.

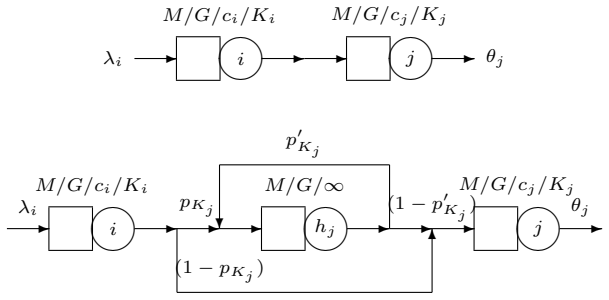

Figure 3: The Generalized Expansion Method (GEM)

Stage II: Parameter estimation

At this stage the values for important parameters are determined, namely, the blocking probability, p_K, the feedback blocking probability, $p'K$, and the service rate of the holding node, μ_h.

- To determine p_K, exact analytical formulas should be used whenever possible (Kerbache and Smith, 2000). Unfortunately, an exact p_K formula is unavailable for $M/G/c/K$ queues, the case of interest in this paper. However, approximations

for p_K were provided recently by Smith (2003) and they can be used here. These approximations are based on a closed-form expression derivable from the finite capacity exponential queue $M/M/c/K$ using Kimura's two-moment approximation (Kimura, 1996). The following p_K formula for $M/G/2/K$ is presented as an example:

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

Other expressions are readily available for $c \in \{3,\ldots,10\}$ (Smith, 2003) and in principle it is possible to extend the approximation approach to $c \gg 10$. The formulas are likely to become extremely complex, but they should also be directly translatable into numerical formulas that would be useful. This will be done at a future date.

- Since no exact method is available to calculate p'_K, an approximation from Labetoulle and Pujolle (1980) based on diffusion techniques is used:

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

in which r_1 and r_2 are the roots of the polynomial

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

in which $\lambda = \lambda_j - \lambda_h(1-p'_K)$, and λ_j and λ_h are the actual arrival rates to the finite and artificial holding notes, respectively. Furthermore, it can be argued that

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

- The delay distribution at the holding node h is actually nothing but the remaining service time of the finite node j. Based on the renewal theory, one can formulate the remaining service time distribution as the following rate μ_h, where

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

in which σ_j^2 is the service time variance of the finite node. At this point, one should notice that if the service time of the finite node is exponentially distributed with rate μ_j, then the memoryless property of exponential distribution will hold such that

$$\mu_h = \mu_j.$$

Stage III: Feedback elimination

As a result of the feedback loop at the holding node, a strong dependency on the arrival process is created. To eliminate such dependency, the service rate at the holding node must be adjusted as follows,

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

As a consequence, the service rate at node i preceding the finite node j is affected as well. One can see that the mean service time at node i is μ_i^{-1} when the finite node is unsaturated, and $\mu_i^{-1} + \mu_h'^{-1}$ when the finite node is saturated. Thus, on average, the mean service time of node i preceding the finite node j is

$$\tilde{\mu}_i^{-1} = \mu_i^{-1} + p_K\mu_h'^{-1}. \tag{9}$$

Bringing it all together

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

$$\lambda = \lambda_j - \lambda_h(1-p'_K), \tag{10}$$
$$\lambda_j = \tilde{\lambda_i}(1-p_K), \tag{11}$$
$$\lambda_j = \tilde{\lambda_i} - \lambda_h, \tag{12}$$
$$p'_K = \left[\frac{\mu_j+\mu_h}{\mu_h} - \frac{\lambda\left[(r_2^K - r_1^K) - (r_2^{K-1} - r_1^{K-1})\right]}{\mu_h\left[(r_2^{K+1} - r_1^{K+1}) - (r_2^K - r_1^K)\right]}\right]^{-1} \tag{13}$$
$$z = (\lambda+2\mu_h)^2 - 4\lambda\mu_h, \tag{14}$$
$$r_1 = \frac{[(\lambda+2\mu_h) - z^{\frac{1}{2}}]}{2\mu_h}, \tag{15}$$
$$r_2 = \frac{[(\lambda+2\mu_h) + z^{\frac{1}{2}}]}{2\mu_h}, \tag{16}$$
$$p_K = f(\rho, s^2, c). \tag{17}$$

Note that Eq. (17) for p_K is a function of the traffic intensity ρ, the squared coefficient of variation of the (general) service time, s^2, and it is mainly a function of the number of servers c. Expressions for p_K for $M/G/c/K$ queues, with $c \in \{1,2,\ldots,10\}$, have been developed by Smith (2003) and are used in the above set of equations. Following a similar approach as in Smith (2003), expressions for $c \gg 10$ can be straightforwardly developed.

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 that 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. An important point about this process is that we do not physically modify the networks, only represent the expansion process through the software.

The actual throughput at a node i that is succeeded by a finite node j, denoted as θ_i, is then obtained as follows,

$$\theta_i = \lambda_i (1 - p_K).$$

Note that the blocking probability p_K at node i is a complex function of the traffic intensity ρ, among other parameters, which is dependent on its *corrected* mean service time $\tilde{\mu}_i$, given by the last step of the GEM, Eq. (9). Finally, the throughput of the overall queueing network is the sum of all throughput(s) obtained at the last node(s) of the network.

3.2 Optimization Algorithm

To optimize the relaxed problem RBCAP, given by Eq. (6), subject to constraints (3) and (4), we shall use Powell's method (a classical non-linear derivative-free optimization algorithm) while a two-moment approximation and the GEM compute the performance measure of interest (the throughput). In a few words, Powell's method, well detailed in Himmelblau (1972), locates the minimum of a non-linear function $f(\mathbf{x})$ by successive unidimensional searches from an initial starting point $\mathbf{x}^{(0)}$ 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, an appropriate step is made in each direction. The overall step from the beginning to the p^{th} step is conjugated to all of the p sub-directions of the search. Fig. 4 describes Powell's unconstrained optimization algorithm, as used in our experiments. Implementations of the algorithm in FORTRAN and C are common.

Although we have had remarkable success in the past with coupling Powell's algorithm and the GEM (Smith, 2004), extra care must be taken here with the buffer and server allocation problem because Powell's algorithm is an unconstrained search process. Thus, unless we control the search, certain solution vectors violating the constraints could be considered and we do not want to consider these solutions. Additionally, because the search process could be trapped in local optima, we restarted the algorithm several times (20 was sufficient) with different random starting solutions.

Another important point is that the algorithm relies on the blocking probabilities that are actually implemented. For instance, we used for the tests given here an implementation that included only the blocking probabilities p_K for the cases $c = \{1, 2, \ldots, 10\}$. That means if one tries the algorithm for cases with $\rho = \lambda/(10 \times \mu) > 1.0$, then an error message should be given, because there is such a thing as a stable queue under $\rho > 1.0$. However, this should not be a problem from a practical point of view because it is possible to extend p_K for values $c \gg 10$, as noted by Smith (2003).

algorithm
input $G(N, A, P)$, $\mathbf{\Lambda}$, $\boldsymbol{\mu}$, *and* $\mathbf{x}^{(0)}$
/* *choose a set of linear-independent 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 uni-dimensional search* */
/* *computing* $f(\bullet)$ *by the GEM* */
$\mathbf{x}^{(i+1)} \leftarrow \arg\min_{\gamma \in \mathcal{R}} f\left(\mathbf{x}^{(i)} + \gamma \mathbf{d}^{(i)}\right)$
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
$\mathbf{x}^{(\text{opt})} \leftarrow \arg\min_{\gamma \in \mathcal{R}} f\left(\mathbf{x}^{(n+1)} + \gamma(\mathbf{x}^{(n+1)} - \mathbf{x}^{(1)})\right)$
choose new search direction $\mathbf{d}^{(1)}, \ldots, \mathbf{d}^{(n)}$
end if
until $\|\mathbf{x}^{(\text{opt})} - \mathbf{x}^{(1)}\| < \epsilon$
print $\mathbf{x}^{(\text{opt})}$
end algorithm

Figure 4: Powell's algorithm

As a final remark, the reader should be aware that usually the quality of the approximations deteriorates with the increase of the number of nodes in the network. Thus, where possible, the user should try to reduce the number of nodes by aggregation so as to retain only the nodes that are very important. However, we notice that quite long lines have been successfully analyzed by the GEM, as reported by Spinellis et al. (2000).

4 Numerical Results

First, complete enumeration results are shown for a simple network and detailed results generated by our optimization methodology are given. Subsequently, we describe the different network structures used and their input data. Next, we elaborate on the results for these network structures. Finally, we analyze more complex queueing network structures. The reader should bear in mind that since the range of possible experiments is exponential, we have presented a select sample.

4.1 Complete Enumeration

To evaluate the quality of the solutions given by the algorithm, we compare the optimal allocation of buffers and servers generated via our approach with the results obtained via a complete enumeration. The purpose is to verify whether the optimal solution obtained is indeed (close to) the true optimal solution. We focus here on the objective function where price $\omega_i = 0.5,\ \forall i \in N$. The reason is that based on the analysis presented in Sec. 2.2, we know that the zero-buffer system is preferred. The

situation with price $\omega_i = 0.5$ is thus appreciated as a convenient configuration for testing the proposed approach developed for the BCAP.

We evaluate a 3-node series topology (see Fig. 6), in which the arrival rate Λ is fixed at 1.0, the processing rate at each node μ_i is fixed at 10, and an s^2 of 1.5 is used. Table 2 gives the results for this experiment and Fig. 5 shows the graphical representation of the buffer and server combinations and their respective objective function values.

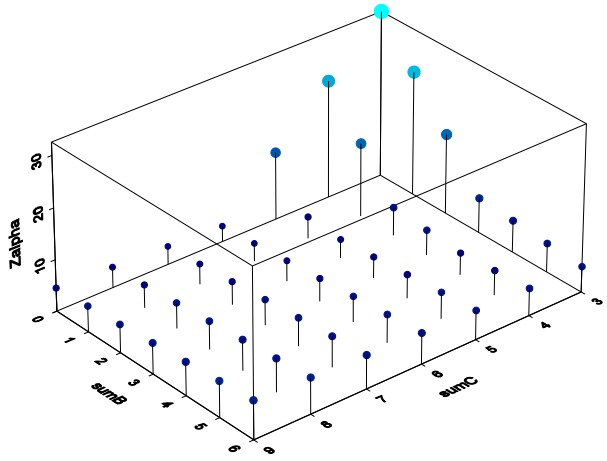

Figure 5: Complete enumeration results

Although the network analyzed is small, the performance measures for 49 possible combinations of buffers and servers were required. Each of these possible combinations was evaluated using the GEM to obtain the related throughput. The objective function value was then minimized to obtain the optimal allocation, for a given server and buffer allocation. The number of buffers, the number of servers, and the resulting throughput for this minimized objective function value are given in Table 2. Based on the complete enumeration, we see that the lowest objective function value Z_α, is obtained with the zero-buffer configuration $\mathbf{c} = (2, 2, 2)$ and $\mathbf{B} = (0, 0, 0)$ (Table 2, line #22). The solution generated by the proposed methodology also gives the same zero-buffer network configuration. Other settings (not shown) were also tested and these results confirmed the ability of our methodology to find sound server and buffer allocations.

As an ultimate check, we also set up a simulation experiment for this case. We used an observation period of 200,000 time units and a warm-up period of 2,000 time units (Robinson, 2007), for 20 independent replications. The simulation was carried out using ARENA (Kelton et al., 2001). Using simulation, we find that the simulated throughput equals 0.999 (the half-width of the 95% confidence interval equals 0.001). This accuracy is similar to that reported by Smith (2003), Andriansyah et al. (2009), and others.

4.2 *Results for three basic network structures*

In this section, we analyze the buffer and server allocations for three basic configurations (series, merge and splits). First, we review the different network structures analyzed. Subsequently, we elaborate on the results for these different network structures.

Network Structures

For each topology, we use networks with number of nodes $|N| = \{3, 7, 15\}$. Jobs arrive into the network with an arrival rate of $\Lambda \in \{2.0, 4.0, 8.0\}$. It is possible to have more than one arrival node in the network. The jobs then flow following the predefined routes. At each node i, there are c_i available servers that process the jobs, each with a processing rate of μ_i. For all settings of serial, split, and merge topologies, we will use a processing rate $\mu_i = 10, \ \forall i \in N$. The variability of the servers is reflected by their squared coefficient of variation s^2, which we set to $\{0.5, 1.0, 1.5\}$ for each node. There is a capacity of K_i (buffers and servers) at each node. Due to the finite buffer capacity, blocking may occur and some arrivals will be lost. In such cases, the resulting throughput θ will be lower than the arrival rate Λ.

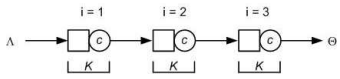

Figure 6: Series topology, $|N| = 3$

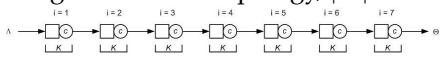

Figure 7: Series topology, $|N| = 7$

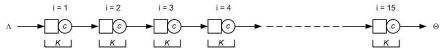

Figure 8: Series topology, $|N| = 15$

The series topology is shown in Figs. 6, 7, and 8, which present networks of 3, 7, and 15 nodes, respectively. This topology has a simple flow structure where the finished jobs from one node are moved to the next node downstream. The routing probability from one node to another is simply 1.0.

The split topology is shown in Figs. 9, 10, and 11, which present networks of 3, 7, and 15 nodes, respectively. This topology represents alternative routings to be chosen by the incoming arrival after being processed at a node. The likelihood of choosing a route is represented by the routing probability.

The merge topology is shown in Figs. 12, 13, and 14, which present networks of 3, 7, and 15 nodes, respectively. This topology combines more than one incoming source of arrival stream into a finite node. In all figures from these three topologies, the position of each node in the network is also depicted. These positions will be referred to in all experiments and analysis that involve such topologies.

Table 2: Complete enumeration of buffer-server combinations ($N = 3$, $\Lambda = 1.0$, and $s^2 = 1.5$)

#	$\mathbf{c}$	$\mathbf{K}$	$\sum_i c_i$	$\sum_i K_i$	$\sum_i B_i$	$\theta(\mathbf{c},\mathbf{B})$	Z_α	#	$\mathbf{c}$	$\mathbf{K}$	$\sum_i c_i$	$\sum_i K_i$	$\sum_i B_i$	$\theta(\mathbf{c},\mathbf{B})$	Z_α
1	(1 1 1)	(1 1 1)	3	3	0	0.96896	32.5	26	(2 2 2)	(4 3 3)	6	10	4	1.00000	5.00
2	(1 1 1)	(1 1 2)	3	4	1	0.97783	24.2	27	(2 2 2)	(4 4 3)	6	11	5	1.00000	5.50
3	(1 1 1)	(1 2 2)	3	5	2	0.98692	15.6	28	(2 2 2)	(4 4 4)	6	12	6	1.00000	6.00
4	(1 1 1)	(2 2 2)	3	6	3	0.99623	6.77	29	(3 2 2)	(3 2 2)	7	7	0	0.99997	3.53
5	(1 1 1)	(2 2 3)	3	7	4	0.99734	6.16	30	(3 2 2)	(3 3 2)	7	8	1	0.99999	4.01
6	(1 1 1)	(2 3 3)	3	8	5	0.99844	5.56	31	(3 2 2)	(3 3 3)	7	9	2	1.00000	4.50
7	(1 1 1)	(3 3 3)	3	9	6	0.99955	4.95	32	(3 2 2)	(3 4 3)	7	10	3	1.00000	5.00
8	(1 1 2)	(1 1 2)	4	4	0	0.97898	23.0	33	(3 2 2)	(3 4 4)	7	11	4	1.00000	5.50
9	(1 2 1)	(1 2 2)	4	5	1	0.98811	14.4	34	(3 2 2)	(4 4 4)	7	12	5	1.00000	6.00
10	(1 1 2)	(2 2 2)	4	6	2	0.99747	5.53	35	(3 2 2)	(5 4 4)	7	13	6	1.00000	6.50
11	(1 1 2)	(2 3 2)	4	7	3	0.99858	4.92	36	(3 3 2)	(3 3 2)	8	8	0	0.99999	4.01
12	(2 1 1)	(2 3 3)	4	8	4	0.99969	4.31	37	(3 3 2)	(3 3 3)	8	9	1	1.00000	4.50
13	(2 1 1)	(3 3 3)	4	9	5	0.99970	4.80	38	(3 3 2)	(3 3 4)	8	10	2	1.00000	5.00
14	(2 1 1)	(4 3 3)	4	10	6	0.99970	5.30	39	(3 3 2)	(4 3 4)	8	11	3	1.00000	5.50
15	(1 2 2)	(1 2 2)	5	5	0	0.98931	13.2	40	(3 3 2)	(5 3 4)	8	12	4	1.00000	6.00
16	(2 1 2)	(2 2 2)	5	6	1	0.99871	4.29	41	(3 3 2)	(5 4 4)	8	13	5	1.00000	6.50
17	(2 2 1)	(2 2 3)	5	7	2	0.99982	3.68	42	(3 3 2)	(5 5 4)	8	14	6	1.00000	7.00
18	(2 2 1)	(3 2 3)	5	8	3	0.99984	4.16	43	(3 3 3)	(3 3 3)	9	9	0	1.00000	4.50
19	(2 2 1)	(3 3 3)	5	9	4	0.99985	4.65	44	(3 3 3)	(4 3 3)	9	10	1	1.00000	5.00
20	(2 2 1)	(4 3 3)	5	10	5	0.99985	5.15	45	(3 3 3)	(5 3 3)	9	11	2	1.00000	5.50
21	(2 2 1)	(4 4 3)	5	11	6	0.99985	5.65	46	(3 3 3)	(5 4 3)	9	12	3	1.00000	6.00
22	**(2 2 2)**	**(2 2 2)**	**6**	**6**	**0**	**0.99996**	**3.04**	47	(3 3 3)	(5 5 3)	9	13	4	1.00000	6.50
23	(2 2 2)	(3 2 2)	6	7	1	0.99997	3.53	48	(3 3 3)	(5 5 4)	9	14	5	1.00000	7.00
24	(2 2 2)	(3 3 2)	6	8	2	0.99998	4.02	49	(3 3 3)	(5 5 5)	9	15	6	1.00000	7.50
25	(2 2 2)	(3 3 3)	6	9	3	1.00000	4.50								

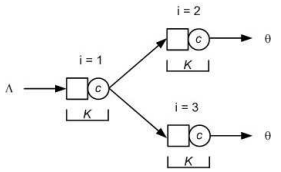

Figure 9: Split topology, $|N| = 3$

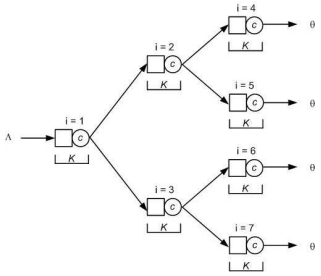

Figure 10: Split topology, $|N| = 7$

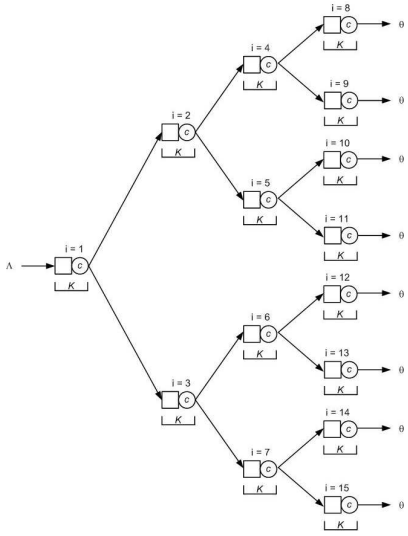

Figure 11: Split topology, $|N| = 15$

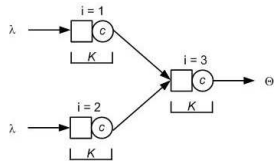

Figure 12: Merge topology, $|N| = 3$

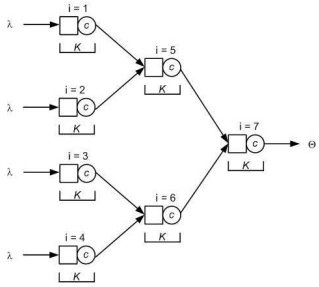

Figure 13: Merge topology, $|N| = 7$

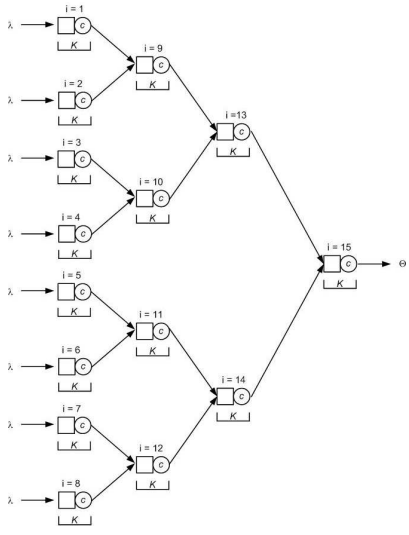

Figure 14: Merge topology, $|N| = 15$

Detailed results

Tables 3, 4 and 5 give the results for the basic network structures outlined in the previous section. The c/B price ratio gives an indication of the relative cost of servers compared to buffers. A price ratio of 8:1, for

example, means that servers are 8 times more expensive than buffers. For the small networks given in Table 3, simulation results are added to verify the quality of the solutions.

Some interesting observations can be made from these tables.

- Zero-buffer solutions are generated where it is expected, *i.e.*, if the c/B cost ratio favors adding servers. This is a good indication of the quality of our heuristic. We observe that in some cases it is harder to identify the zero-buffer solution. Fortunately, if the zero-buffer system is not identified, the error made is small and is maximally off by one buffer unit per node.

- As the price of servers is increased (and the price of buffers is reduced), the resulting allocation shows that there is a decrease in the number of servers, $\sum_i c_i$. Then the number of buffers, $\sum_i B_i$, will increase compared to the allocation for the case with equal prices. The reduction of servers happens due to the fact that the price of servers is becoming higher and hence it is better to reduce the servers and add more buffers to the network. However, reducing the number of servers affects the throughput significantly due to the large reduction of the processing rates and therefore many buffers are needed to compensate for even a slight decrease in the number of servers. This confirms our analysis as discussed in Sec. 2.

- It is interesting to note that the number of servers is sometimes severely reduced, especially when the server cost is high, leading to single server systems. To compensate, the buffer sizes need to be dramatically increased to guarantee the target throughput (see in Tables 3 to 5, the buffer size columns, $\sum_i B_i$). This is an important observation in cases where servers are relatively expensive compared to buffers (*e.g.*, in the semiconductor industry).

- The observations made are consistent over the different topologies (comparing Tables 3, 4 and 5). Series, merges and split topologies behave similarly with regard to the c/B cost ratio.

- Focusing on the small sets, we see that the throughput results are confirmed by the simulations. More specifically, the estimated throughput falls in the confidence interval for all simulated experiments. Note that the simulation of larger-sized networks may be possible but it certainly would require programming a simulation code from scratch instead of using ARENA, as was done here. We remark that the aim here is only to give some idea of the accuracy of the performance evaluation methodology since a more detailed study has been done elsewhere (for instance, see Smith, 2003).

4.3 A complex topology

We consider a combination of all three basic topologies, as shown in Fig. 15. This network consists of 16 nodes with the processing rate of servers in each node given in Fig. 15. The network is originally from Smith and Cruz (2005). We use exactly the same values for Λ, μ, s^2, and routing probabilities for the splitting node (#1 and #2). Note that the routing probability #1 refers to the upper tier of the node, while #2 refers to the lower tier (see Fig. 15 for the position of each node in the network). For the purpose of comparison, we reproduce in Table 6 the results from Smith and Cruz (2005) for this network structure (Table 29 in their paper). Note that they considered an $M/G/1/K$ setting and therefore the number of servers in all nodes is set to 1 while optimizing the buffer allocation. Using our methodology, we run experiments with all settings similar to those listed in Table 6. The results are presented in Table 7.

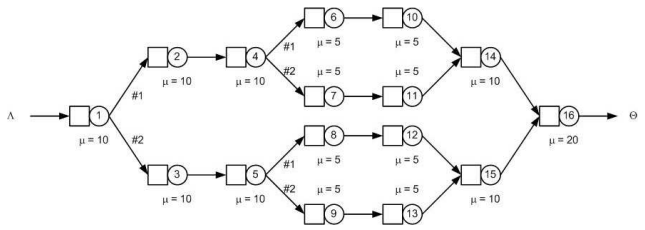

Figure 15: Combined topology

The results from our methodology, given in Table 7, show a higher throughput than for the pure Buffer Allocation Problem (see Table 6) for every setting. As expected, we found that the optimal server allocation in the BCAP is different from the server settings in the pure BAP. However, this depends strongly upon the price ratio of servers to buffers. We found that $M/G/1/K$ is not an optimal configuration for this particular queueing network structure, except when servers are becoming relatively expensive. For these cases, we found that single servers are indeed optimal (see rows where the c/B ratio is 8:1).

We observe that (near) zero-buffer configurations are identified where appropriate, *i.e.*, where the servers are relatively cheaper compared to buffers. Varying the coefficient of variation does result in some changes in the optimal server and buffer allocation, which highlights the importance of models dealing with general service times. The results show that the number of buffers seem to be larger with higher variability, which could be expected, since the increase in the squared coefficient of variation means a high variability. The extra buffers are there to handle this increased variability.

Of course, no equal allocation is obtained. This is intuitively acceptable since there are some nodes that receive more arrivals (*i.e.*, merging nodes) and other nodes that receive fewer arrivals (*i.e.*, nodes preceding the splitting nodes). Furthermore, the processing rates of servers are not the same at each node. Extra

Table 3: Results for $N = 3$

SERIES									Simulation		
	c/B ratio	$\mathbf{c}$	$\mathbf{K}$	$\sum_i c_i$	$\sum_i K_i$	$\sum_i B_i$	$\theta(\mathbf{c},\mathbf{B})$	Z_α	$\theta(\mathbf{c},\mathbf{B})^s$	δ	Z_α^s
$\Lambda = 2$	1:8	(3 3 3)	(3 3 3)	9	9	0	2.0000	1.00	2.000	0.002	1.20
$s^2 = 1.0$	1:4	(3 3 3)	(3 3 3)	9	9	0	2.0000	1.80	2.000	0.002	2.00
	1:2	(2 2 2)	(2 2 2)	6	6	0	1.9990	2.98	2.000	0.001	2.10
	1:1	(2 2 2)	(2 2 2)	6	6	0	1.9990	3.98	2.000	0.001	3.10
	2:1	(2 2 2)	(2 2 2)	6	6	0	1.9990	4.98	2.000	0.001	4.10
	4:1	(1 1 1)	(5 5 5)	3	15	12	1.9997	5.11	1.999	0.001	5.60
	8:1	(1 1 1)	(5 5 5)	3	15	12	1.9997	4.31	1.999	0.001	4.80
SPLIT									Simulation		
	c/B ratio	$\mathbf{c}$	$\mathbf{K}$	$\sum_i c_i$	$\sum_i K_i$	$\sum_i B_i$	$\theta(\mathbf{c},\mathbf{B})$	Z_α	$\theta(\mathbf{c},\mathbf{B})^s$	δ	Z_α^s
$\Lambda = 4$	1:8	(3 3 3)	(3 3 3)	9	9	0	3.9999	1.07	3.999	0.002	2.50
$s^2 = 1.0$	1:4	(3 3 3)	(3 3 3)	9	9	0	3.9999	1.87	3.999	0.002	3.30
	1:2	(3 2 2)	(3 2 2)	7	7	0	3.9993	3.06	3.998	0.002	4.63
	1:1	(3 3 3)	(3 3 3)	9	9	0	3.9999	4.57	3.999	0.002	6.00
	2:1	(3 2 2)	(3 2 2)	7	7	0	3.9993	5.39	3.998	0.002	6.97
	4:1	(2 1 1)	(5 5 5)	4	15	11	3.9997	5.67	3.999	0.002	6.40
	8:1	(1 1 1)	(10 5 5)	3	20	17	3.9997	4.86	3.999	0.002	5.86
MERGE									Simulation		
	c/B ratio	$\mathbf{c}$	$\mathbf{K}$	$\sum_i c_i$	$\sum_i K_i$	$\sum_i B_i$	$\theta(\mathbf{c},\mathbf{B})$	Z_α	$\theta(\mathbf{c},\mathbf{B})^s$	δ	Z_α^s
$\Lambda = 8$	1:8	(4 4 4)	(4 4 4)	12	12	0	7.9999	1.43	7.999	0.004	2.53
$s^2 = 1.0$	1:4	(4 4 4)	(4 4 4)	12	12	0	7.9999	2.50	7.999	0.004	3.60
	1:2	(4 4 4)	(4 4 4)	12	12	0	7.9999	4.10	7.999	0.004	5.20
	1:1	(3 3 3)	(3 3 5)	9	11	2	7.9995	6.05	8.000	0.003	5.30
	2:1	(3 3 4)	(3 3 4)	10	10	0	7.9998	6.90	7.999	0.003	7.57
	4:1	(2 2 3)	(5 5 6)	7	16	9	7.9998	7.65	7.998	0.003	9.00
	8:1	(1 1 3)	(10 10 6)	5	26	21	7.9997	7.09	8.000	0.002	7.08

Table 4: Results for $N = 7$

SERIES								
	c/B ratio	$\mathbf{c}$	$\mathbf{K}$	$\sum_i c_i$	$\sum_i K_i$	$\sum_i B_i$	$\theta(\mathbf{c},\mathbf{B})$	Z_α
$\Lambda = 4$	1:8	(3 3 3 3 3 3 3)	(3 3 3 3 3 3 3)	21	21	0	3.9995	2.81
$s^2 = 1.0$	1:4	(3 3 3 3 3 3 3)	(3 3 3 3 3 3 3)	21	21	0	3.9995	4.67
	1:2	(3 3 3 3 3 3 3)	(3 3 3 3 3 3 3)	21	21	0	3.9995	7.47
	1:1	(3 3 3 3 3 3 3)	(3 3 3 3 3 3 3)	21	21	0	3.9995	11.0
	2:1	(3 3 3 3 3 3 3)	(3 3 3 3 3 3 3)	21	21	0	3.9995	14.5
	4:1	(2 2 2 2 2 2 2)	(5 5 5 5 5 5 5)	14	35	21	3.9995	15.9
	8:1	(1 1 1 1 1 1 1)	(10 10 10 10 10 10 10)	7	70	63	3.9993	13.9
SPLIT								
	c/B ratio	$\mathbf{c}$	$\mathbf{K}$	$\sum_i c_i$	$\sum_i K_i$	$\sum_i B_i$	$\theta(\mathbf{c},\mathbf{B})$	Z_α
$\Lambda = 8$	1:8	(4 4 4 4 4 4 4)	(4 4 4 4 4 4 4)	28	28	0	7.9999	3.21
$s^2 = 1.0$	1:4	(4 4 4 4 4 4 4)	(4 4 4 4 4 4 4)	28	28	0	7.9999	5.70
	1:2	(3 3 3 3 3 3 3)	(5 3 3 3 3 3 3)	21	23	2	7.9995	8.88
	1:1	(3 3 3 3 3 3 3)	(5 3 3 3 3 3 3)	21	23	2	7.9995	12.0
	2:1	(4 3 3 2 2 2 2)	(4 3 3 2 2 2 2)	18	18	0	7.9985	13.5
	4:1	(4 2 2 1 1 1 1)	(4 5 5 5 5 5 5)	12	34	22	7.9994	14.6
	8:1	(2 1 1 1 1 1 1)	(9 10 10 5 5 5 5)	8	49	41	7.9991	12.6
MERGE								
	c/B ratio	$\mathbf{c}$	$\mathbf{K}$	$\sum_i c_i$	$\sum_i K_i$	$\sum_i B_i$	$\theta(\mathbf{c},\mathbf{B})$	Z_α
$\Lambda = 2$	1:8	(2 2 2 2 2 2 4)	(2 2 2 2 2 2 4)	16	16	0	2.0000	1.80
$s^2 = 1.0$	1:4	(2 2 2 2 2 2 2)	(2 2 2 2 2 2 2)	14	14	0	1.9996	3.15
	1:2	(2 2 2 2 2 2 2)	(2 2 2 2 2 2 2)	14	14	0	1.9996	5.02
	1:1	(2 2 2 2 2 2 2)	(2 2 2 2 2 2 2)	14	14	0	1.9996	7.35
	2:1	(1 1 1 1 2 2 2)	(2 2 2 2 2 2 2)	10	14	4	1.9994	8.59
	4:1	(1 1 1 1 1 1 1)	(2 2 2 2 3 3 5)	7	19	12	1.9995	8.52
	8:1	(1 1 1 1 1 1 1)	(2 2 2 2 3 3 5)	7	19	12	1.9995	8.08

servers allocated to some nodes in the network are compensated by a significant reduction in the number of buffers. In all settings, our methodology identifies systems with a very low buffer size. Even so, the resulting throughput is higher than for the single-server case. These encouraging results show the importance of considering both buffers and servers in the optimization of complex queueing networks.

Sensitivity analysis

Table 8 shows the results of varying the arrival rate to the complex 16-node network, $\Lambda = \{2.0, 8.0\}$. The results suggest that changing the arrival rate results in a change to the buffer and server allocation. Extra arrivals will require more servers and buffers to minimize blocking. As such, there is a notable increase in the number of servers at each node. Again, the zero-buffer allocations

Table 5: Results for $N = 15$

	c/B ratio	$\mathbf{c}$	$\mathbf{K}$	$\sum_i c_i$	$\sum_i K_i$	$\sum_i B_i$	$\theta(\mathbf{c},\mathbf{B})$	Z_α
SERIES								
$\Lambda = 8$	1:8	(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 4 4 4 4 4 4 4)	60	60	0	7.9985	8.14
$s^2 = 1.0$	1:4	(4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4)	(4 4 4 4 4 4 4 4 4 4 4 4 4 4 4 4)	60	60	0	7.9985	13.5
	1:2	(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 4 4 4 4 4 4 4)	60	60	0	7.9985	21.5
	1:1	(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 4 4 4 4 4 4 4)	60	60	0	7.9985	31.5
	2:1	(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 4 4 4 4 4 4 4)	60	60	0	7.9985	41.5
	4:1	(3 3 3 3 3 3 3 3 3 3 3 3 3 3 3)	(6 6 6 6 6 6 6 6 6 6 6 6 6 6 6)	45	90	45	7.9984	46.6
	8:1	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(10 10 10 10 10 10 10 10 10 10 10 10 10 10 10)	30	150	120	7.9983	41.7
SPLIT								
	c/B ratio	$\mathbf{c}$	$\mathbf{K}$	$\sum_i c_i$	$\sum_i K_i$	$\sum_i B_i$	$\theta(\mathbf{c},\mathbf{B})$	Z_α
$\Lambda = 2$	1:8	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	30	30	0	1.9996	3.68
$s^2 = 1.0$	1:4	(4 2 2 2 2 2 2 1 1 1 1 1 1 1 1)	(4 2 2 2 2 2 2 1 1 1 1 1 1 1 1)	24	24	0	1.9988	6.04
	1:2	(4 2 2 2 2 2 2 1 1 1 1 1 1 1 1)	(4 2 2 2 2 2 2 1 1 1 1 1 1 1 1)	24	24	0	1.9988	9.24
	1:1	(4 2 2 2 2 2 2 1 1 1 1 1 1 1 1)	(4 2 2 2 2 2 2 1 1 1 1 1 1 1 1)	24	24	0	1.9988	13.2
	2:1	(2 2 2 1 1 1 1 1 1 1 1 1 1 1 1)	(2 2 2 2 2 2 2 1 1 1 1 1 1 1 1)	18	22	4	1.9982	15.1
	4:1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(5 3 3 2 2 2 2 1 1 1 1 1 1 1 1)	15	27	12	1.9983	16.1
	8:1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(5 3 3 2 2 2 2 2 2 2 2 2 2 2 2)	15	35	20	1.9994	16.1
MERGE								
	c/B ratio	$\mathbf{c}$	$\mathbf{K}$	$\sum_i c_i$	$\sum_i K_i$	$\sum_i B_i$	$\theta(\mathbf{c},\mathbf{B})$	Z_α
$\Lambda = 4$	1:8	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	31	31	0	3.9992	4.21
$s^2 = 1.0$	1:4	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 4)	31	32	1	3.9993	7.71
	1:2	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	31	31	0	3.9992	11.1
	1:1	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 4)	31	32	1	3.9993	16.7
	2:1	(1 1 1 1 1 1 1 1 2 2 2 2 2 2 3)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	23	31	8	3.9988	19.2
	4:1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 2)	(2 2 2 2 2 2 2 2 3 3 3 3 5 5 5)	16	43	27	3.9989	19.3
	8:1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(2 2 2 2 2 2 2 2 3 3 3 3 5 5 10)	15	48	33	3.9989	18.1

Table 6: Results for complex topology (taken from Smith and Cruz, 2005)

#1	#2	Λ	s^2	$\mathbf{c}$	$\mathbf{B}$	$\sum_i c_i$	$\sum_i B_i$	$\theta(\mathbf{c},\mathbf{B})$
0.5	0.5	5.0	0.5	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(8 4 4 4 4 4 4 4 4 4 4 4 4 4 5)	16	69	4.9899
			1.0	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(10 5 5 5 5 4 4 4 4 4 4 4 4 5 5 5)	16	77	4.9879
			1.5	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(11 5 5 5 5 5 5 5 5 5 5 5 5 5 5 6)	16	87	4.9877

Table 7: Results for the complex topology optimized on both buffers and servers

#1	#2	Λ	s^2	c/B ratio	$\mathbf{c}$	$\mathbf{K}$	$\sum_i c_i$	$\sum_i K_i$	$\sum_i B_i$	$\theta(\mathbf{c},\mathbf{B})$	Z_α
0.5	0.5	5.0	0.5	1:8	(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 3 3 3 3 3)	48	48	0	4.9996	5.76
				1:4	(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 3 3 3 3 3)	48	48	0	4.9996	10.0
				1:2	(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 3 3 3 3 3)	48	48	0	4.9996	16.4
				1:1	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	44	44	0	4.9998	22.2
				2:1	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	32	44	12	4.9989	26.5
				4:1	(3 1 1 1 1 1 1 1 1 1 1 1 1 1 3)	(3 5 5 5 5 3 3 3 3 3 3 3 5 5 3)	20	60	40	4.9974	26.6
				8:1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(11 6 6 6 6 4 4 4 4 4 4 4 6 6 11)	16	90	74	4.9994	23.0
			1.0	1:8	(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 3 3 3 3 3)	48	48	0	4.9994	5.94
				1:4	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	44	44	0	4.9997	9.09
				1:2	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	44	44	0	4.9997	15.0
				1:1	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	44	44	0	4.9997	22.3
				2:1	(3 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(3 3 3 3 3 2 2 2 2 2 2 2 2 3 3)	34	40	6	4.9984	26.2
				4:1	(2 2 2 2 2 1 1 1 1 1 1 1 1 2 3)	(6 3 3 3 3 4 4 4 4 4 4 4 3 3 4)	25	60	35	4.9989	28.1
				8:1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(13 6 6 6 6 4 4 4 4 4 4 4 6 6 13)	16	94	78	4.9987	24.1
			1.5	1:8	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	44	44	0	4.9996	5.24
				1:4	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	44	44	0	4.9996	9.15
				1:2	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	44	44	0	4.9996	15.0
				1:1	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	44	44	0	4.9996	22.4
				2:1	(3 2 2 2 2 2 2 2 2 2 2 2 2 2 3)	(3 3 3 3 3 2 2 2 2 2 2 2 2 3 3)	34	40	6	4.9979	26.8
				4:1	(2 2 2 2 2 1 1 1 1 1 1 1 1 2 3)	(6 3 3 3 3 4 4 4 4 4 4 4 3 3 4)	25	60	35	4.9983	28.7
				8:1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(15 7 7 7 7 4 4 4 4 4 4 4 7 7 15)	16	104	88	4.9986	25.4

occur when buffers are relatively more expensive than servers, regardless of the arrival rate.

Table 9 shows the results of varying the routing probability of the complex 16-node network. We changed the routing probability in such a way that two mirrored networks are evaluated: first, #1 = 0.2 and #2 = 0.8, and second, #1 = 0.2 and #2 = 0.8. Based on the results, we see that the resulting allocations are also mirrored in most cases. Some deviations are observed, but these are within an acceptable range. This again suggests that the methodology is performing as it should. Similarly, depending upon the relative price of buffers versus servers, the allocation changes accordingly, giving preference to either buffers or servers.

4.4 Managerial insights

Based on the results provided above, we list here the main insights and the important managerial contributions of the proposed approach:

Table 8: Complex topology: different Λ

#1	#2	Λ	s^2	c/B ratio	$\mathbf{c}$	$\mathbf{K}$	$\sum_i c_i$	$\sum_i K_i$	$\sum_i B_i$	$\theta(\mathbf{c},\mathbf{B})$	Z_α
0.5	0.5	2.0	0.5	1:8	(4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4)	(4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4)	36	36	0	1.9999	4.06
				1:1	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	32	32	0	1.9994	16.6
				8:1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	16	44	28	1.9993	18.0
			1.5	1:8	(4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4)	(4 2 2 2 2 2 2 2 2 2 2 2 2 2 2 4)	36	36	0	1.9999	4.09
				1:1	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	(2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2)	32	32	0	1.9991	16.9
				8:1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(6 4 4 4 4 2 2 2 2 2 2 2 2 4 4 6)	16	52	36	1.9991	19.1
		8.0	0.5	1:8	(4 4 4 4 4 4 4 4 4 4 4 5 4 4 4 4)	(4 4 4 4 4 4 4 4 4 4 4 5 4 4 4 4)	65	65	0	7.9999	7.34
				1:1	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	(5 3 3 3 3 2 2 2 2 2 2 2 2 3 3 5)	44	44	0	7.9976	24.4
				8:1	(3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3)	(6 8 8 8 8 5 5 5 5 5 5 5 5 8 8 6)	20	100	80	7.9986	28.0
			1.5	1:8	(4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4)	(4 3 3 3 3 3 3 3 3 3 3 3 3 3 3 4)	50	50	0	7.9991	6.41
				1:1	(6 3 3 3 3 3 3 3 3 3 3 3 3 3 3 6)	(6 3 3 3 3 3 3 3 3 3 3 3 3 3 3 6)	54	54	0	7.9994	27.6
				8:1	(3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 3)	(7 11 11 11 11 6 6 6 6 6 6 6 6 11 11 7)	20	128	108	7.9985	31.3

Table 9: Complex topology: different routing probability

#1	#2	Λ	s^2	c/B ratio	$\mathbf{c}$	$\mathbf{K}$	$\sum_i c_i$	$\sum_i K_i$	$\sum_i B_i$	$\theta(\mathbf{c},\mathbf{B})$	Z_α
0.2	0.8	5.0	0.5	1:8	(5 2 3 2 3 2 2 2 3 2 2 2 3 2 3 5)	(5 2 3 2 3 2 2 2 3 2 2 2 3 2 3 5)	43	43	0	4.9998	4.98
				1:1	(5 2 3 2 3 2 2 2 3 2 2 2 3 2 3 5)	(5 2 3 2 3 2 2 2 3 2 2 2 3 2 3 5)	43	43	0	4.9998	21.7
				8:1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(11 3 8 3 8 1 3 3 7 1 3 3 7 3 8 11)	16	83	67	4.9988	22.9
			1.5	1:8	(5 2 3 2 3 2 2 2 3 2 2 2 3 2 3 5)	(5 2 3 2 3 2 2 2 3 2 2 2 3 2 3 5)	43	43	0	4.9996	5.16
				1:1	(5 2 3 2 4 1 2 2 3 1 2 2 3 2 4 5)	(5 2 3 2 4 1 2 2 3 1 2 2 3 2 4 5)	43	43	0	4.9996	21.9
				8:1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(15 4 11 4 11 1 3 3 9 1 3 3 9 4 11 15)	16	107	91	4.9986	25.8
0.8	0.2	5.0	0.5	1:8	(5 3 2 3 2 3 2 2 2 3 2 2 2 3 2 5)	(5 3 2 3 2 3 2 2 2 3 2 2 2 3 2 5)	43	43	0	4.9998	4.98
				1:1	(5 3 2 3 2 3 2 2 2 3 2 2 2 3 2 5)	(5 3 2 3 2 3 2 2 2 3 2 2 2 3 2 5)	43	43	0	4.9998	21.7
				8:1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(11 8 3 8 3 7 3 3 1 7 3 3 1 8 3 11)	16	83	67	4.9988	22.9
			1.5	1:8	(5 3 2 3 2 3 2 2 2 3 2 2 2 3 2 5)	(5 3 2 3 2 3 2 2 2 3 2 2 2 3 2 5)	43	43	0	4.9996	5.16
				1:1	(5 4 2 4 2 4 2 2 1 3 2 2 1 4 2 5)	(5 4 2 4 2 4 2 2 1 3 2 2 1 4 2 5)	45	45	0	4.9997	22.8
				8:1	(1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1)	(15 11 4 11 4 9 3 3 1 9 3 3 1 11 4 15)	16	107	91	4.9986	25.8

1. The theorem in Sec. 2.2 is confirmed by all results. This shows that the marginal throughput gain of one extra unit of buffer must be sufficiently large to prefer a non-zero buffer system. If this is not the case, it is always better to choose a zero-buffer system. An important factor in this trade-off is the relationship between the cost of a server, ω, and the cost of not meeting the throughput constraint, α. In addition, it is important to recognize that a server is simultaneously acting as a buffer, but has the advantage that the server also adds value. Consequently, one extra buffer has to improve the throughput much more than a server to be added.

2. In this paper, networks up to size $N = 16$ are analyzed. Theoretically, networks of any size N can be handled by the algorithm. In many practical applications, we see that although N is large at first sight, a reduction in N is possible by combining machines, states *etc.* Note that the number of nodes N can be small, but each node i can have a large number of servers. Specifically, we have not hit the boundary of our methodology yet for practical industrial applications.

3. Variability in the service rates clearly results in performance deterioration and increases the need for buffers and servers, as deduced from the different results using different coefficients of variation. Programs that reduce variability are thus important to minimize the total number of buffers and servers needed.

5 Conclusions

In this paper, we discussed the joint buffer and server allocation problem (BCAP). We considered arbitrary configured networks that consist of series, split, and merge topologies, and combinations of all three topologies. The optimization methodology that we developed utilizes a Lagrangian relaxation so as to incorporate multiple objectives (minimizing the number of buffers and minimizing the number of servers) and constraints (*i.e.*, a threshold throughput) into a single objective function. This methodology identifies the best buffer and server allocation for a given queueing network structure using the Powell's search method.

The results from this optimization methodology were compared with complete enumeration and with simulation. These comparisons demonstrated the good solution quality of the proposed methodology. The results obtained for different network structures show that the methodology provides sound allocations for both buffers and servers. Comparing the results to similar settings obtained from the literature, we quantify the possible improvements over these published results. For instance, the throughput could be improved with the increase of the number of servers, disclosing that sometimes we cannot go any further with the throughput with increasing only the number of buffers. Previously developed methodologies were not able to handle this trade-off between the number of servers and buffers. Another important conclusion is that it is important to consider general service times when they occur, because the squared coefficient of variation of the

service time strongly influences the allocation that will ensure optimum performance.

A potential research direction is using multi-objective algorithms as the search method in combination with the GEM for the problem on-hand. This powerful class of search method would allow one to consider a *multi objective buffer-server allocation problem* with multiple distinct objectives.

ACKNOWLEDGMENTS

The research of Frederico Cruz has been partially funded by CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico; grants 201046/1994-6, 301809/1996-8, 307702/2004-9, 472066/2004-8, 304944/2007-6, 561259/2008-9, 553019/2009-0), by CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior; grant BEX-0522/07-4), and by FAPEMIG (Fundação de Amparo à Pesquisa do Estado de Minas Gerais; grants CEX-289/98, CEX-855/98, TEC-875/07, and CEX-PPM-00401/08).

REFERENCES

Andriansyah, R., van Woensel, T., Cruz, F. R. B. and Duczmal, L., 2009. Performance optimization of open zero-buffer multi-server queueing networks. Manuscript under review. **URL:** *http://www.est.ufmg.br/ftp/fcruz/publics/zbuff.pdf*

Buzacott, J. A. and Shanthikumar, J. G., 1993. *Stochastic Models of Manufacturing Systems*, Prentice-Hall, Englewood Cliffs, NJ.

Cruz, F. R. B., Duarte, A. R. and van Woensel, T., 2008. Buffer allocation in general single-server queueing network, *Computers & Operations Research* **35**(11): 3581–3598.

Down, D. G. and Karakostas, G., 2008. Maximizing throughput in queueing networks with limited flexibility, *European Journal of Operational Research* **187**(1): 98–112.

Fransoo, J. C. and Rutten, W. G. M. M., 1994. A typology of production control situations in process industries, *International Journal of Operations & Production Management* **14**(12): 47–57.

Gershwin, S. B. and Schor, J. E., 2000. Efficient algorithms for buffer space allocation, *Annals of Operations Research* **93**: 117–144.

Hall, N. G. and Sriskandarajah, C., 1996. A survey of machine scheduling problems with blocking and no-wait in process, *Operations Research* **44**(3): 510–525.

Hillier, F. S. and So, K. C., 1995. On the optimal design of tandem queueing systems with finite buffers, *Queueing Systems* **21**: 245–266.

Hillier, M. S. and Hillier, F. S., 2006. Simultaneous optimization of work and buffer space in unpaced production lines with random processing times, *IIE Transactions* **38**: 39–51.

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

Jain, S. and Smith, J. M., 1994. Open finite queueing networks with $M/M/C/K$ parallel servers, *Computers & Operations Research* **21**(3): 297–317.

Johnson, D. S., Lenstra, J. K. and Rinnooy Kan, A. H. G., 1978. The complexity of the network design problem, *Networks* **8**(4): 279–285.

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

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

Kerbache, L. and Smith, J. M., 1988. Asymptotic behavior of the expansion method for open finite queueing networks, *Computers & Operations Research* **15**(2): 157–169.

Kerbache, L. and Smith, J. M., 2000. Multi-objective routing within large scale facilities using open finite queueing networks, *European Journal of Operational Research* **121**(1): 105–123.

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

Labetoulle, J. and Pujolle, G., 1980. Isolation method in a network of queues, *IEEE Transactions on Software Engineering* **SE-6**(4): 373–381.

Lemaréchal, C., 2001. Lagrangian relaxation, *Computational Combinatorial Optimization* **1**: 112–156.

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

Li, J. and Meerkov, S. M., 2009. *Production Systems Engineering*, Springer, New York, NY.

Perros, H. G., 1994. *Queueing networks with blocking*, Oxford University Press, Inc., New York, NY.

Ramudhin, A. and Ratliff, H. D., 1995. Generating daily production schedules in process industries, *IIE Transactions* **27**(5): 646–656.

Robinson, S., 2007. A statistical process control approach to selecting a warm-up period for a discrete-event simulation, *European Journal of Operational Research* **176**(1): 332–346.

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

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

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

Smith, J. M. and Cruz, F. R. B., 2005. The buffer allocation problem for general finite buffer queueing networks, *IIE Transactions* **37**(4): 343–365.

Smith, J. M., Cruz, F. R. B. and van Woensel, T., 2009. Optimal server allocation in general, finite, multi-server queueing networks, *Applied Stochastic Models in Business & Industry* (in press).

Smith, J. M., Cruz, F. R. B. and van Woensel, T., 2010. Topological network design of general, finite, multi-server queueing networks, *European Journal of Operational Research* **201**(2): 427–441.

Smith, J. M. and Daskalaki, S., 1988. Buffer space allocation in automated assembly lines, *Operations Research* **36**(2): 343–358.

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

Suri, R., 1985. An overview of evaluative models for flexible manufacturing systems, *Annals of Operations Research* **3**: 13–21.

Suri, R., Sanders, J. L. and Kamath, M., 1993. *Logistics of Production and Inventory*, North Holland, Amsterdam, chapter Performance evaluation of production networks, pp. 199–286.

Tempelmeier, H., 2003. Practical considerations in the optimization of flow production systems, *International Journal of Production Research* **41**(1): 149–170.