# Performance Evaluation and Optimization of Open Zero-Buffer Multi-Server Queueing Networks

R. Andriansyah [a], T. van Woensel [b,*], F. R. B. Cruz [c], L. Duczmal [c]

[a] *Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands*

[b] *Department of Technology Management, Eindhoven University of Technology, Eindhoven, The Netherlands*

[c] *Department of Statistics, Federal University of Minas Gerais, Belo Horizonte, MG, Brazil*

**Abstract**

Open zero-buffer multi-server general queueing networks occur throughout a number of physical systems in the semi-process and process industries. In this paper, we first analyze and quantify the performance of these systems in terms of throughput using the generalized expansion method (GEM). We compare our results with simulation, and show that the GEM gives an effective approximation for the performance measures of the systems under study. Secondly, we embed the performance evaluation via the GEM in a multi-objective optimization setting. This multi-objective optimization approach results in the Pareto efficient curves optimizing both the number of servers used and the throughput. Experiments for a large number of settings and network topologies are presented in detail.

*Key words:* Zero Buffer Systems, Generalized Expansion Method, Performance Evaluation and Optimization, Genetic Algorithms

## 1. Introduction and Motivation

Bufferless networks occur throughout a number of real-life physical systems in the semi-process and process industries. A zero-buffer production environment might be created due to the processing technology of the product itself, or simply due to the absence of any storage capacity between two consecutive operations of

* Corresponding author.

*Email addresses:* r.andriansyah@tue.nl (R. Andriansyah), t.v.woensel@tm.tue.nl (T. van Woensel), fcruz@est.ufmg.br (F. R. B. Cruz), duczmal@est.ufmg.br (L. Duczmal).

a job. More specifically, the system of interest is a zero-buffer multi-server general queueing network, which operates as follows: jobs arrive to the first node in the system with a certain arrival rate $\lambda$. The job is then processed by an available server with service rate $\mu$. After the service is finished, the job proceeds to the next connected node in the system only if a server is available at this node. If this is not the case, then the finished job has to wait at the previous server until there is an available server at the downstream node. There is no queueing space between servers: the only buffer space available in each node are the servers themselves. Due to the possibility of blocking, the throughput rate of the network might be smaller than the arrival rate.

Hall and Sriskandarajah (1996) describe a steel production process. Here, molten steel goes through a series of operations ranging from ingots, un-molding, reheating, soaking, and preliminary rolling. In this process, the steel must pass through one operation to another continuously, without any waiting or buffering of work in process, since such waiting would result in cooling down the steel to a temperature that is not acceptable for the next process. Hence, either a job is finished and transferred directly to the next process, or it is buffered in the machine itself until the downstream process is ready to receive another job. In food-processing , no buffer space is allowed between the cooking operation and before the canning operation. This is due to the requirement that the product should still be fresh when it is canned. Similar issues can be found in producing juice and beer (see *e.g.*, Fey (2000)). In this examples, restrictions in the processing technology creates zero-buffer production system. Ramudhin and Ratliff (1995) studied a condiments manufacturer, with mayonnaise and various types of salad dressing as the product. The nature of the product dictates hygienic consideration as one of the critical factor in production. Hence, there is no space for work-in-process inventory and the product must never wait between two operations. As a last example, third generation mobile communication networks are characterized by a multi-server zero-buffer queueing system (Tsybakov, 2002). In such systems, arrivals are represented by requests of audio, data, and video messages, whereas the service time is the message transmission time. Here, zero-buffers are caused due to absence of storage capacity between operations.

Despite the high industrial relevance of zero-buffer networks particularly in process and semi-process industries, only scant literature is available quantifying the performance or optimizing these networks. We refer to Fransoo and Rutten (1994) for more information on process industries. This paper suggests a solution to this challenging problem by providing a methodology for the performance evaluation and optimization of zero-buffer systems in an arbitrary network configuration. The major contributions of this paper are:

(i) In the literature, few papers are found on the specific zero buffer systems analyzed in the present paper. Moreover, these papers are mainly limited to single server tandem line settings. Here, we examine the performance of open zero-buffer multi-servers with arbitrarily configured queueing networks;

(ii) We compare our results with simulation, and show that the generalized expansion method (GEM) proposed by Kerbache and Smith (1988) gives an excellent approximation for the performance measures

of the systems studied.

(iii) We present a multi-objective optimization methodology for zero-buffer systems, which provide particularly insightful results. It is the first time to our knowledge that a multi-objective approach is considered for the optimization of bufferless systems.

(iv) We demonstrate the usefulness of the developed tools in the system design phase of zero-buffer systems. More specifically, we elaborate on several network topologies, including a complex, arbitrarily configured zero-buffer system, and provide answers to some difficult managerial questions regarding the system's performance and its optimal configuration.

The paper is structured as follows. First, a brief literature review on zero-buffer queueing networks is presented. Then, we describe how we obtain the blocking probability, one of the most important performance measures in the analysis. Next, the performance evaluation methodology applied is described. In this paper, we use the GEM to obtain the relevant performance measures. After this, we use a genetic algorithm approach to optimize this kind of queueing networks. In the following section, we elaborate on the experimental results obtained for a large number of situations (*i.e.*, tandem, split, merge cases, and large complex topologies). We analyze the performance of these systems both analytically and by simulation. The last section concludes this paper, with final remarks and topics for future research in the area.

## 2. Literature Review

The research in the area of queueing networks is very active, resulting in a vast amount of journal and conference papers, books, reports etc. For a general and complete classification of queueing networks, the reader is referred to, *e.g.*, Walrand (1988). Queueing networks can be divided into two categories (Perros, 1994): unrestricted and restricted queueing network. Unrestricted queueing networks include cases where all nodes within the network have an unlimited capacity. Restricted queueing networks have limited (finite) capacity for all nodes in the network. These queueing networks often represent real-life systems, where there normally exist finite spaces for holding entities. In this paper, we focus on investigating restricted queueing networks.

### 2.1. *Restricted Queueing Networks*

Restricted queueing networks have a finite capacity in each node, referred to as the total buffer capacity of with 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 $K_j$ (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). In general, three blocking mechanisms can be distinguished, blocking-after-service (BAS), blocking-before-service (BBS), and repetitive-service (RS). These three types of blocking mechanisms are presented in Figure 1. Most production lines operate under the BAS system. Moreover, in the literature it is the most common assumption regarding buffer behavior (Dallery and Gershwin, 1992).

Figure 1 goes around here

BAS systems occur when a finished job in node $i$ attempts to move downstream to node $j$ whose buffers are still being occupied by other jobs. In this sense, buffers at downstream nodes are full and hence cannot receive the finished job from the upstream node. It is referred to as BAS due to the fact that the blocked job is already finished with service in the previous node. It is therefore blocked while occupying the upstream server. In the literature, this type of blocking is also known as type-1 blocking (Perros, 1994).

BBS systems are characterized by situations in which jobs in upstream node $i$ have already determined its destination downstream node $j$ even before they are serviced. If a job at upstream node $i$ finds full its downstream destination node $j$, the server at node $i$ becomes blocked and thus cannot process the job. In this sense, the blocking occurs before the job is processed in upstream node $i$. Jobs in upstream node will have to wait until there is available space in its downstream destination node, either in the queue or server. In the literature, this type of blocking is known as type-2 blocking (Perros, 1994).

RS systems occur when a finished job at upstream node $i$ cannot continue to downstream node $j$ because the downstream node is full, and the job immediately receives another service at node $i$. This repeating process continues until downstream node $j$ is not full. In the literature, this type of blocking is known as type-3 blocking (Perros, 1994).

### 2.2. *Zero-Buffer Queueing Networks*

A particular case where a queueing network has a finite capacity but no buffers before servers is denoted as a zero-buffer queueing network (also denoted in the literature as bufferless, no intermediate buffers etc.). In this specific case, the buffer space at node $j$ is equal to the number of server $c$ in that node, that is, $K_j = c$. Given that there is no space to queue, a job in the upstream node can only enter the downstream node if the servers have finished processing their jobs (see Figure 2).

Figure 2 goes around here

For small tandem line zero-buffer networks, exact solutions are available. Small asynchronous single server tandem lines with zero buffer and reliable machines were partly analyzed by several authors (Hildebrand, 1968; Hillier and Boling, 1967; Muth and Alkaff, 1987; Rao, 1976a,b). Most of this exact analysis is based on Continuous Time Markov processes. Here, we will rely on simulations and on approximation techniques for analyzing more complex queueing networks (including splits, merges, and multiple servers per node and symetrical/asymmetrical settings). More specifically, for the system on-hand in this paper, we will obtain

the performance measures via the GEM (discussed in detail in section 3). In Kendall notation, we look at $M/M/c/K$ queueing models and set the buffer size equal to the number of servers, $K = c$, resulting in $M/M/c/c$ queueing models.

### 2.3. *Blocking probabilities*

For $M/M/c/c$ models, it is known that the probability of the system being full at any time in steady state, given that the maximum number of jobs in the system equals to the number of servers in the system, corresponds to the Erlang's loss formula:

$$p_c = \frac{\left(\frac{\lambda}{\mu}\right)^c / c!}{\sum_{i=0}^{c} \left(\frac{\lambda}{\mu}\right)^i / i!} \tag{1}$$

with $\lambda$ being the arrival rate, $\mu$ the service rate and $c$, the number of servers.

It turns out that Eq. (1) also holds for any service-time distribution (other than Markovian). More specifically, this means that for $M/G/c/c$ queueing systems with no waiting space, the above equation is also valid regardless of the service time variability or the distribution itself. The expressions for the steady-state probabilities for this $M/G/c/c$ case are identical to those for the corresponding $M/M/c/c$ system (see Gross and Harris, 1998). This means that all results presented here, also immediately hold for the general service-time distribution case.

## 3. Performance Evaluation of Zero Buffer Networks

The GEM is a robust and effective approximation technique developed by Kerbache and Smith (1987). It has been successfully used to estimate performance measures for finite queueing networks. As described in previous papers, this method is basically a combination of repeated trials and node-by-node decomposition in which each queue is analyzed separately and then corrections are made in order to take into account the interrelation between the queues in the network. The GEM uses BAS, which is prevalent in most production and manufacturing, transportation, and other similar systems.

In this section, we will only present a high-level overview of the method. For more detailed information and applications of the GEM, the reader is referred to the papers by Kerbache and Smith (1988), Spinellis et al. (2000), Smith and Cruz (2005), Cruz and Smith (2007), and Cruz et al. (2007a). The GEM involves three stages: network reconfiguration, parameter estimation, and feedback elimination. Before discussing each stage in detail, the variables used in this section are defined in Table 1.

Table 1 goes around here

### 3.1. *Network reconfiguration*

The first step in the GEM involves reconfiguring the network: an artificial node is added for each finite node in the network. The artificial node is added to register the blocked customers at the finite node (Kerbache and Smith, 1988).

Following Kerbache and Smith (2000), the GEM thus creates for each finite queue, represented by vertex $j$, an auxiliary vertex $h_j$, modeled as an $M/G/\infty$ queue (see Figure 3). When an entity arrives to the system, vertex $j$ may be blocked with probability $p_{c_j}$, or unblocked, with probability $(1 - p_{c_j})$. Under blocking, the entities are rerouted to vertex $h_j$ for a delay while node $j$ is busy. Vertex $h_j$ helps to accumulate the time an entity has to wait before entering vertex $j$ and to compute the effective arrival rate to vertex $j$. In other words, the GEM ultimate goal is to provide an approximation scheme to update the service rates of upstream nodes that takes into account all blocking after service in there, caused by downstream nodes:

$$\tilde{\mu}_i^{-1} = \mu_i^{-1} + p_{c_j}(\mu'_{h_j})^{-1}.$$

Figure 3 goes around here

### 3.2. *Parameter estimation*

In the second stage, the parameter estimation of all unknowns for the system described above are determined. These unknowns are in this case (Kerbache and Smith, 1987):

(i) The probability of a customer being blocked, $p_{c_j}$;

(ii) The probability that a customer is forced back to the holding node given he was rejected at the previous trial, $p'_{c_j}$;

(iii) The service rate at the holding node, $\mu_{h_j}$;

The reader is referred to the paper by Jain and Smith (1994) for more details.

### 3.3. *Feedback elimination*

The repeated visits to the holding nodes (due to the feedbacks), create strong dependence in the arrival process. Therefore, the repeated immediate feedback is eliminated. This is done by giving the customer enough service time during the first passage through the holding node. The adapted service rate for the holding node $\mu'_{h_j}$ then becomes:

$$\mu'_{h_j} = (1 - p'_{c_j})\mu_{h_j}. \tag{2}$$

And the adapted squared coefficient of the service rate for the holding node $(c^2_{sh_j})'$ becomes:

$$(c^2_{sh_j})' = p'_{c_j} + (1 - p'_{c_j})c^2_{sh_j}. \tag{3}$$

The effective service rate in node 2 will be the average of $\mu_2^{-1}$ when there is no blocking and $\mu_2^{-1}+(\mu'_{h_j})^{-1}$ when there is blocking or:

$$\mu_{2\text{eff}}^{-1} = \mu_2^{-1} + p_{c_j}(\mu'_{h_j})^{-1}. \tag{4}$$

## 4. Multi-Objective Optimization of Zero Buffer Networks

In this section, we consider the optimization of zero-buffer systems where the GEM is incorporated in a genetic algorithm (GA) approach. Since there are no buffers in the network, the optimization problem on-hand is essentially a server allocation problem for zero-buffer general service queueing network. As servers are expensive, one would like to minimize the amount of servers as much as possible. On the other hand, one would also like to maximize the throughput of the network. This throughput is affected by the allocated servers, *i.e.* more servers leads to a higher throughput.

### 4.1. *Model Formulation*

The problem described can be formulated as a multi-objective optimization problem with two objectives: minimizing the total number of servers and maximizing the throughput. The following mathematical formulation represents the multi-objective optimization problem.

$$\text{optimize } Z = \Big\{ f_1(\mathbf{x}), f_2(\mathbf{x}) \Big\} \tag{5}$$

s.t.:

$$x_i \in \{1, 2, \ldots\},\ \forall i, \tag{6}$$

in which

$f_1(\mathbf{x}) = \sum_i c_i \times x_i$, where $x_i$ is the server allocation to node $i$ and $c_i$ is the cost of a server at node $i$,

$f_2(\mathbf{x}) = \Theta(\mathbf{x})$, the throughput profit.

In the above formulation, we trade-off throughput versus total number of servers. Of course, we could have used Work-In-Progress (WIP) or cycle time. We feel, however, that the first criterion in the network optimization should be assuring a certain throughput compared to the arrival rate. As such, we consider the other variables WIP or cycle time as secondary criteria. Of course, the methodology can be extended taking into account these variables. For the sake of presentation of the results and the insights, we choose not to do this (see *e.g.* Smith and Cruz, 2005, for a similar approach).

### 4.2. *Genetic Algorithms*

To solve the above problem formulation, we use a powerful class of optimization heuristics, called Genetic Algorithms (GA). One particular quality of a GA is that it is capable of generating an entire set of multi-objective solutions called the *Pareto-optimal set*. As such, a GA searches for a set of optimal solutions of a problem with more than one objective function in a single run. A GA finds the set of *non-dominated* solutions, that is, the solutions that are better off than another solution with regards to at least one objective, while not being worse off in any other objectives than another solution. In a minimization problem, for example, solutions $x \in F$ that are not a part of the Pareto-optimal set are said to be dominated because there are other solutions $z \in F$ such that $f(z) < f(x)$, and $f(z) \neq f(x)$. As such, $f(z)$ is better off than $f(x)$ in at least one objective function, without being worse off in the other objective functions. The resulting Pareto curve allows one to see the tradeoff among the objective function values from different solutions. That is, a GA allows for evaluating the loss in one objective (*e.g.* the throughput) compared to a simultaneous enhancement in another objective (*e.g.* the server allocation).

The optimization procedure in a GA starts with the evaluation of several points along the search space. This set of individual points is called the population. The initial population converges to the optimum population through several operators as follows (Carrano et al., 2006):

(i) **Mutation:** Randomly modify some individuals to reach other points in the search space.

(ii) **Crossover:** Combine the randomly organized pairwise individuals such that each former pair of individuals creates a new pair.

(iii) **Selection:** Evaluate the individuals after mutation and crossover. Using a probabilistic rule, these individuals are chosen or not chosen to be included in the new population. The rule is organized so that individuals with better objective value have greater probability to be selected into the new population.

(iv) **Elitism:** The "best" individuals from the former population are deterministically inserted into the new population.

Particularly important for a multi-objective optimization problem is defining the selection and the elitism operators to determine the best individual from the search space. One of the most widely used selection and elitist operator is the non-dominated sorting scheme, known as the NSGA-II (see *e.g.* Deb et al. (2002)). It has been shown that the NSGA-II requires less computational effort as compared to other GA algorithms. For a complete elaboration on this algorithm, see Deb et al. (2002), and for details on the version of the particular implementation used in this article, see Cruz et al. (2007b). We will use a GA as the search method in combination with the GEM to obtain the performance measures of the network.

## 5. Experimental Results

We first discuss the results of using the GEM as a performance evaluation tool. Secondly, we use a GA heuristic in combination with the GEM to solve the multi-objective optimization problem as described previously. Finally, we provide a complex topology to demonstrate the performance of our approach in the design of large and complex zero buffer networks. All algorithms described are coded in Fortran and are available from the authors upon request.

### 5.1. *Performance Evaluation*

To evaluate the performance of the GEM in our network settings, we conducted experiments using three different topologies: series, split, and merge topologies. We examine these topologies for both symmetrical as asymmetrical settings. Please bear in mind that the range of possible experiments is exponential itself, so we have determined a select sample to present. The configurations tested are shown in Figures 4, 5, and 6.

Figure 4 goes around here

Figure 5 goes around here

Figure 6 goes around here

#### 5.1.1. *Symmetrical Networks*

In each topology, we analyze several number of nodes, $N \in \{3, 5, 9\}$, and servers, $c \in \{2, 4, 10\}$. We set different values for the arrival rates, $\lambda \in \{2, 4, 8, 16\}$. In all settings a service rate with a mean of $\mu = 10$ was used. With these combinations of parameters, we ended up with 36 different systems for each of the topologies (series, merge, and split). The series topology consists of a series of nodes ($N$), in which each node has $c$ number of servers (see Figure 4). In the split topology (see Figure 5), we divide the arriving jobs to subsequent nodes after departing from the first node in the system. That is, the splitting node is positioned directly after the first node. The routing probability for both routes is set to be equal to each other (0.5). Every case in this topology has two ending nodes in the system ($N - 1$ and $N$). In the merge topology (see Figure 6), the jobs arrive from two different source nodes. The overall arrival rate is then divided equally for the two source nodes. The last node merges the two streams.

Table 2 shows the results for all different sets and topologies. The results reported are the overall throughput at the network. Table 2 is read as follows: the 3-node experiment with an arrival rate $\lambda = 2$ and the number of servers $c$ equal to $(2, 2, 2)$ results in a throughput of 1.996. In general, we observe that the blocking effect becomes more severe for high arrival rates relative to the service rates. For those cases, the throughput rate is reduced more compared to the arrival rate than the less congested settings. This rule holds true for all three types of topologies involved in the experiment.

Table 2 goes around here

In order to evaluate the solution quality of the GEM, we set up a simulation experiment of four selected cases in the series, merge, and split topologies. We used an observation period of 200,000 time unit and a warm-up period of 2,000 time units for 20 independent replications. The simulation was carried out using ARENA (Kelton et al., 2001). Table 3 is based on the simulation results. In the column labeled analytical, we give the throughput result from the GEM for each of the cases. We then compare this analytical result with the average result obtained via the simulation. The column $\delta$ refers to the half-width of the 95% confidence interval. Also included in the tables is the % deviation for the analytical results on the throughput, $\Delta\%\theta \stackrel{\text{def}}{=} \frac{(\theta-\theta^s)}{\theta^s} \times 100\%$. We see that the analytical results for all cases fall within the 2% error range, which is very reasonable and acceptable (see column $\Delta\%\theta$). Based on the simulation output, we conclude that the analytical results are reliable.

Table 3 goes around here

5.1.2. *Asymmetrical Networks*

In the previous section, we considered cases with symmetrical settings in the service rates for the different nodes. In this section, we set up some experiments unbalancing the routing probabilities (for the split topologies) and the arrival rates (for the merge topologies), and assuming different service rates, $\mu$'s, and different number of servers, $c$'s, along the queueing network. The results are shown in Table 4. Simulation results from selected cases are presented in Table 5. Based on the asymmetrical results, we can see that the main conclusions, as observed for the symmetrical results, hold being the highest error below 5%. We see that the blocking effects becomes more important with the increase of the arrival rates. With regards to the simulations, we note that the GEM tends to both overestimate and sometimes underestimate the throughput as compared to the "true" throughput from simulation (as reflected in the values for $\Delta\%\theta$). This is in line with what should be expected when comparing simulation with analytical results.

Table 4 goes around here

Table 5 goes around here

5.2. *Network Optimization*

In this section, we identify the optimal server configuration given the network structure, arrival rate, and processing rates similar to that of the previous sections. We consider both symmetrical as well as asymmetrical settings. In this section, we will again focus on the earmarked networks from Tables 2 and 4. Using the GA in conjunction with the GEM, we identify the Pareto-optimal set, which is the set of optimal solutions for both objectives in the objective function, *i.e.* the minimal number of servers and the maximal throughput. After much experimentation, the population size was set to 20 and the number of generations

to 100, 1,000, and 10,000, for 3, 5, and 9 nodes, respectively, in order to reach a good trade-off between a fair approximation of the Pareto-optimal set and low computational times.

The resulting Pareto-optimal sets for the earmarked symmetrical settings are provided in Figure 7. The figures show that the throughput of all the earmarked *symmetrical* settings lie on the Pareto-optimal set (pointed with an arrow). As such, these settings are optimal given the corresponding network structure, arrival rate, and processing rates. A valuable insight from the Pareto-optimal set generated by the GA is that we are able to quantify the additional throughput gained by adding one or more servers into the network and optimizing the allocation of the additional servers. For example, the Pareto graphs in 7 (c), (f), and (i) show that too many servers are utilized and that less servers could have been used without reducing the throughput significantly. Eventually, one may use this information to evaluate whether the cost of adding extra servers into the network is justified by the benefit gained from the additional throughput. This information is critical given the fact that there are cases where additional servers only give marginal increase to the throughput, and other cases where additional servers give significant increase to the throughput.

Figure 7 goes around here

Figure 8 shows the GA optimization results for the earmarked *asymmetrical* settings. We observed that the throughput from all the earmarked settings is below the Pareto-optimal set (see in Figure 8, pointed with an arrow), which means that our settings for the network structure, arrival rates, and processing rates were not optimal in the previous section. More specifically, for the same total number of servers in the network, there exists another server allocation that results in a higher throughput. For example, in the case of asymmetrical serial topology with 3 nodes and processing rates of $(12, 11, 10)$ for nodes 1, 2, and 3 respectively, the GA found the optimal server allocation as $(7, 6, 3)$ for node 1, 2 and 3. This configuration results in a throughput of 2.000 as opposed to a throughput of 1.998 when the server allocation is 2,4,10, as in the earmarked asymmetrical setting. Note that both results have a total number of 16 servers, but that the actual configuration can be improved such that the throughput can be increased. Depending upon the managerial preference, one can thus decide to increase the throughput by using the same number of servers but in a different configuration (*i.e.* find vertically the nearest configuration on the Pareto-optimal set). Alternatively, one can also accept the current throughput but achieve this throughput with less servers used (*i.e.* find horizontally the nearest configuration on the Pareto curve).

Figure 8 goes around here

### 5.3. *A large and complex topology*

In this section, we analyze a large and complex topology. This topology is inspired by an example fruit juice blending and packaging plant, of which a detailed description can be found in Fey (2000). Using this complex topology, we elaborate in detail on the merits of the methodology, by applying the GEM and the

multi-objective approach to this large design problem. More specifically, the queueing network structure given in Figure 9 is analyzed.

Figure 9 goes around here

It should be clear that this queueing network combines all three topologies that have been investigated separately in the previous section (namely series, merge, and split). It is interesting to see whether the proposed methodology performs well in such a complicated queueing network setting. In this section, three critical decisions are investigated in designing and optimizing this zero buffer production system:

(i) The effect of different arrival rates to the throughput of the system.

(ii) The effect of different service rates of the servers to the throughput of the system.

(iii) The effect of a different network structure to the throughput of the system

We run the GEM using the parameter values given in Table 6. We set the arrival rate to 30.215 units/h; we consider two situations, namely a slow server setting and a fast server setting (see Table 6). The slow server setting is created by using low processing rates of machines as compared to the arrival rate. On the contrary, the fast server setting is characterized by high processing rates of machines as compared to the arrival rate. As such, we would like to see the performance of the GEM in the presence of both low and high blocking situations.

Table 6 goes around here

We compare the results from simulation with that of the GEM and obtain the results depicted in Table 7. The GEM performs quite well given such a complex queueing network setting (that is, queueing network with serial, merge and split topologies with a total of 137 servers involved). There is a notable reduction of the GEM's accuracy under a queueing network setting with high blocking probabilities. However, it is clear that the GEM is able to approximate the throughput of a very complex queueing setting in a matter of seconds. This eliminates the necessity to conduct lengthy simulations, as indicated in Table 7 for the required CPU time.

Table 7 goes around here

Again using the GA in conjunction with the GEM, we derive the Pareto-optimal set for the minimal number of servers and the maximal throughput. We focus on the results for the slow server case discussed in Table 6. Figure 11 gives the results for the Pareto set for increasing populations sizes. Clearly, the Pareto-set converges for a reasonable size of the population (from 100). In the figure, the white dot corresponds to the throughput for the server vector as used in the higher experiments shown in Tables 7 (without the GA optimization). The Figure clearly shows that it is possible to either increase the throughput with the same number of servers (vertical line from white dot), or to reduce the total number of servers with the same throughput (horizontal line from white dot). Similar results are obtained for the fast server case.

Figure 11 goes around here

The fact that the methodology is able to approximate the throughput with high accuracy and within

a short amount of time allows us to analyze diverse system design problems, which otherwise would be extremely time consuming to do by using simulation. Following these encouraging results, we will next consider three particular system design problems of queueing networks, namely (1) evaluating the effect of the arrival rate, (2) evaluating the effect of different processing rates, and (3) evaluating the network structure itself.

*Different arrival rates*

We consider the case in which it is possible to change the arrival rate into the queueing network, while holding the service rate of each server constant. This situation typically denotes the case where a plant manager wants to investigate the threshold of arrival rate beyond which the current server setting is incapable of delivering the desirable throughput. To tackle this issue, we vary the arrival rate into the system and analyze the percentage of blocking compared to the arrivals. Using the same service rate as provided in Table 6, we obtain the throughput for different arrival rates.

In Figure 10, the results from different arrival rates give us insight about the system's capability of handling different arrival volumes. We can see, for example, that the slow server configuration of the system is able to handle an arrival rate up to 10 units/h with a resulting blocking of less than 5%, while the fast server configuration would handle much more. Higher arrival rates will cause a significant increase in blocking percentage, as it can be seen from Figure 10. As a practical implication, the plant manager can use this insight to determine the appropriate arrival rate such that the blocking probability is under the predefined blocking probability threshold. This example shows how the GEM can be used effectively to predict system performance given different arrival rates, and suggest the threshold arrival rate that will seriously impair the performance of the system in terms of blocking.

Figure 10 goes around here

*Different processing rates*

Another common case in system design, involves the choice of the facilities used in the network. In particular, we assume that there are several choices of facilities characterized with different processing rates. We treat the arrival rate as an external variable that is difficult or impossible to change. Given this setting, we would like to evaluate the effect on throughput of changing the service rates, while holding the arrival rate and all other parameters constant. We again set the arrival rate fixed at 30.215 units/h, and use three alternative server processing rates as given in Table 8. The number of servers for each facility is that of Table 6. Given the fixed arrival rate, each alternative configuration is evaluated on the throughput as provided in the last line of Table 8.

Table 8 goes around here

We assume that configuration #1 is the default configuration of the queueing network. Note that this configuration is similar to the previous case where we consider high utilization setting. The resulting throughput from the GEM is 17.422 units/h, as before. Given the processing rates and the number of available servers, one can easily notice that *Phase 2* acts as the bottleneck in the network. As such, one may increase the processing rate of this particular node, as this could be advantageous for the throughput. Furthermore, the last part of the system (*Phase 3*) also has a relatively high utilization, creating heavy imbalance in the system. Table 8 gives the resulting throughput when the processing rates of these phases are increased. Using this configuration, the throughput is increased by 6.1%. As a last example, we increase the processing rates of all last three nodes in *Phase 3* in the system and the bottleneck node (see configuration #3). Consequently, the throughput increases further to 16% compared to the default configuration.

This example shows how the GEM can be used effectively to predict system performance under varying processing rates of servers in the zero buffer network. The result from this analysis can be used as a basis for *i.e.* benefit/cost analysis in order to justify the potential investment for increasing the servers' capacity. Following this line of analysis, one can argue whether or not the cost of increasing the capacity of the servers is justified by the benefit from the additional throughput gained.

*Different network structures*

We now consider the case where it is possible to manipulate the structure of the queueing network in order to increase the throughput of the network. One way to reduce congestion (and thus increase the throughput) of a queueing network is by combining multiple sources of variability, which is known as the concept of *variability pooling* (Hopp and Spearman, 2000). This can be achieved, in one way, by sharing the queue through the use of more flexible machines.

Let us assume that by using more flexible machines it is possible to reduce the number of lines in *Phase 3* from twelve to only three lines. As such, we assume that the new machines are capable of performing multiple tasks. The new queueing network structure is given in Figure 12. Note that for the three-line structure, we will use the data of configuration #1 in Table 8.

Figure 12 goes around here

Using the same total number of servers as in configuration #1, we obtain a throughput of 22.887 units/h, higher than before for this three-line structure, as compared to the throughput of 17.422 units/h for the twelve-line structure (see Table 8, configuration #1). The additional throughput gain can be the used to evaluate whether it is beneficial to use more flexible packaging machines in the network.

## 6. Conclusions and Future Research

Throughout this paper, we have demonstrated the use of the generalized expansion method (GEM) to both evaluate and optimize the performance of a zero-buffer queueing network. We considered three topologies of queueing networks, namely series, split, and merge topologies, taking into account both symmetrical as well as asymmetrical composition of the topologies. We obtained the throughput of the networks using the GEM and compared the results with simulation. The insights from the multi-objective optimization of zero-buffer queueing networks are proven to be invaluable particularly in the design phase of such systems.

We provided an large scale complex topology for which the GEM was used to approximate the throughput rate. We argued that in addition to its accuracy both in low and high blocking probability settings, the expansion method provides a relatively simple means to see how changes in arrival rate and/or processing rate affects the performance of a zero buffer queueing networks. All in all, we found the GEM as a fast and accurate approximation method to measure the throughput rate of zero-buffer queueing networks as well as for optimization purposes of such networks.

*Acknowledgements*

The research of prof. Frederico Cruz has been partially funded by the CNPq (*Conselho Nacional de Desenvolvimento Científico e Tecnológico*) of the Ministry for Science and Technology of Brazil, grants 201046/1994-6, 301809/1996-8, 307702/2004-9, and 472066/2004-8, the FAPEMIG (*Fundação de Amparo à Pesquisa do Estado de Minas Gerais*), grants CEX-289/98, CEX-855/98, and TEC-875/07, and PRPq-UFMG, grant 4081-UFMG/RTR/FUNDO/PRPq/99.

## References

Buzacott, J., Shanthikumar, J. G., 1993. Stochastic Models of Manufacturing Systems. Prentice-Hall.

Carrano, E. G., Soares, L. A. E., Takahashi, R. H. C., Saldanha, R. R., Magela Neto, O., April 2006. Electric distribution network multiobjective design using a problem-specific genetic algorithm. IEEE Transactions on Power Delivery 21 (2), 995–1005.

Cruz, F. R. B., Duarte, A. R., van Woensel, T., 2007a. Buffer allocation in general single-server queueing network. Computers & Operations Research. URL `http://dx.doi.org/10.1016/j.cor.2007.03.004`

Cruz, F. R. B., Duczmal, L., van Woensel, T., Smith, J. MacGregor., 2007b. A multi-objective approach for buffer allocation in general queueing networks. In: Sixth Conference on the Analysis of Manufacturing Systems. BETA and EURANDOM, TU/e, Lunteren, The Netherlands.

435 Cruz, F. R. B., Smith, J. MacGregor, 2007. Approximate analysis of $M/G/c/c$ state-dependent queueing
436 networks. Computers & Operations Research 34 (8), 2332–2344.
437 Dallery, Y., Gershwin, S. B., 1992. Manufacturing flow line systems: A review of models and analytical
438 results. Queueing Systems 12, 3–94.
439 Deb, K., Agrawal, S., Pratap, A., Meyarivan, T., 2002. A fast elitist non-dominated sorting genetic algorithm
440 for multi-objective optimization: NSGA-II. IEEE Transactions on on Evolutionary Computation 6, 182–
441 197.
442 Fey, J., 2000. Design of a fruit juice blending and packaging plant. Ph.D. thesis, Eindhoven University of
443 Technology.
444 Fransoo, J. C., Rutten, W. G. M. M., 1994. A typology of production control situations in process industries.
445 International Journal of Operations and Production Management 14 (12), 47–57.
446 Gross, D., Harris, C. M., 1998. Fundamentals of Queueing Theory, 3rd Edition. Wiley Series in Probability
447 and Statistics, New York.
448 Hall, N. G., Sriskandarajah, C., 1996. A survey of machine scheduling problems with blocking and no-wait
449 in process. Operations Research 44, 510–525.
450 Hildebrand, D., 1968. On the capacity of tandem server, finite queue, service systems. Operations Research
451 16, 72–82.
452 Hillier, F., Boling, R., 1967. Finite queues in series with exponential or Erlang service times: A numerical
453 approach. Operations Research 16, 286–303.
454 Hopp, W.J. and Spearman, M.L., 2000. Factory physics. MacGraw Hill International Editions, New York,
455 NY, USA.
456 Jain, S., Smith, J. MacGregor, 1994. Open finite queueing networks with $M/M/C/K$ parallel servers. Com-
457 puters & Operations Research 21 (3), 297–317.
458 Kelton, D., Sadowski, R. P., Sadowski, D. A., 2001. Simulation with Arena. MacGraw Hill College Div.,
459 New York, NY, USA.
460 Kerbache, L., Smith, J. MacGregor, 1987. The generalized expansion method for open finite queueing net-
461 works. European Journal of Operational Research 32, 448–461.
462 Kerbache, L., Smith, J. MacGregor, 1988. Assymptotic behavior of the expansion method for open finite
463 queueing networks. Computurs & Operations Research 15 (2), 157–169.
464 Kerbache, L., Smith, J. MacGregor, 2000. Multi-objective routing within large scale facilities using open
465 finite queueing networks. European Journal of Operational Research 121, 105–123.
466 Muth, E., Alkaff, A., 1987. The throughput rate of three-station production lines: A unifying solution.
467 International Journal of Production Research 25, 1405–1413.
468 Perros, H. G., 1994. Queueing Networks with Blocking. Oxford University Press.
469 Ramudhin, A., Ratliff, H. D., 1995. Generating daily production schedules in process industries. IEE Trans-

actions 27, 646–656.

Rao, N., 1976a. A generalization of the bowl phenomenon in series production systems. International Journal of Production Research 14 (4), 437–443.

Rao, N., 1976b. A viable alternative to the method of stages solution of series production systems with Erlang service times. International Journal of Production Research 14 (6), 699–702.

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

Spinellis, D., Papodopoulos, C., Smith, J. MacGregor, 2000. Large production line optimisation using simulated annealing. International Journal of Production Research 38 (3), 509–541.

Tsybakov, B., 2002. Optimum discarding in a bufferless system. Queueing Systems 41, 165–197.

Walrand, J., 1988. An Introduction to Queueing Networks. Prentice-Hall, Englewoord Cliffs.

## Figure Legends

**Figure 1:** The three different blocking mechanisms
**Figure 2:** A zero-buffer queueing network representation
**Figure 3:** The generalized expansion method
**Figure 4:** Series topology
**Figure 5:** Split topology
**Figure 6:** Merge topology
**Figure 7:** GA optimization for earmarked symmetrical settings
**Figure 8:** GA optimization for earmarked asymmetrical settings
**Figure 9:** Queueing network structure
**Figure 11:** GA optimization for the fruit juice plant
**Figure 10:** Plot of throughput for different arrival rates
**Figure 12:** New queueing network structure

## Table Legends

**Table 1:** Variables used in this paper
**Table 2:** Throughput $\theta$ for the symmetrical cases
**Table 3:** Simulation results for the symmetrical cases
**Table 4:** Throughput $\theta$ for the asymmetrical cases.
**Table 5:** Simulation results for the asymmetrical cases
**Table 6:** Processing rates used in the complex topology
**Table 7:** Data used in the complex topology
**Table 8:** Throughput rates for different configurations

## Figures

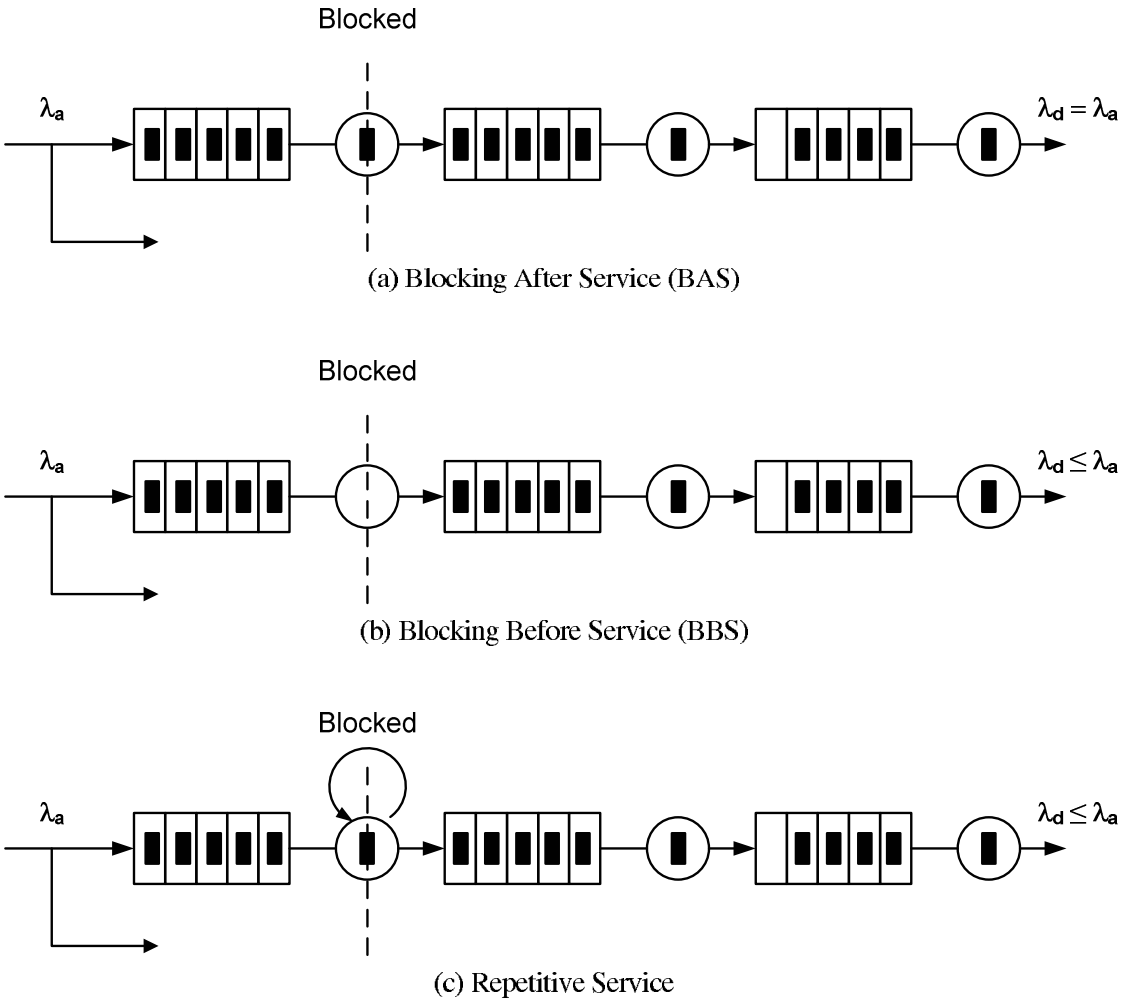



Fig. 1.

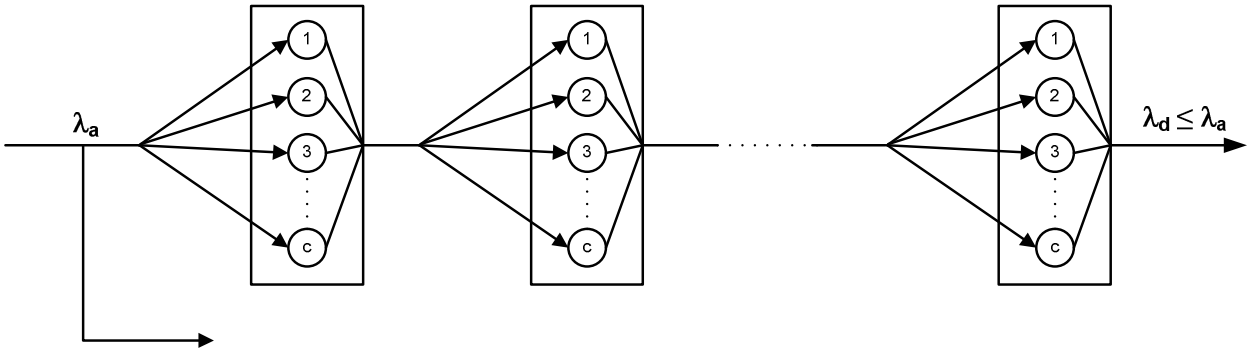



Fig. 2.

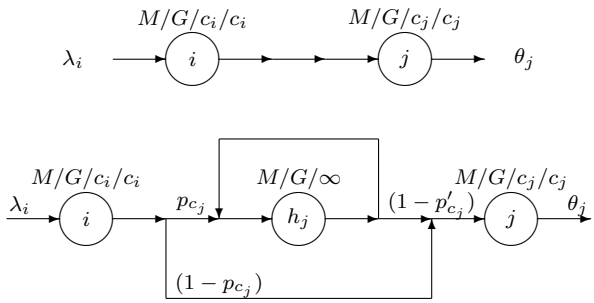



Fig. 3.

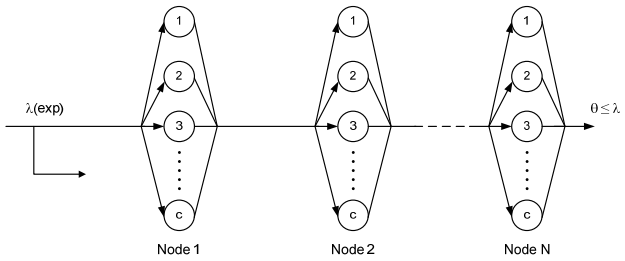



Fig. 4.

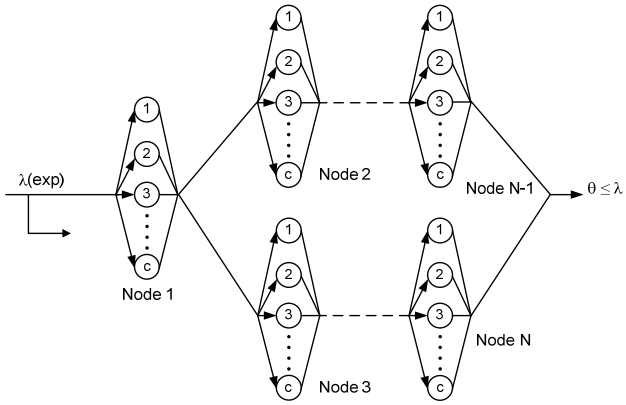



Fig. 5.

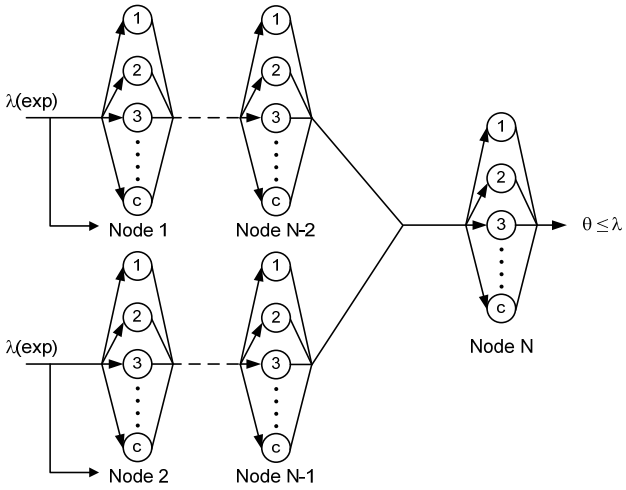
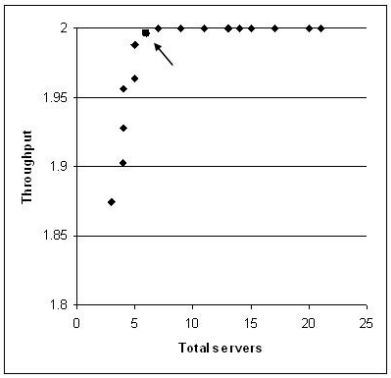
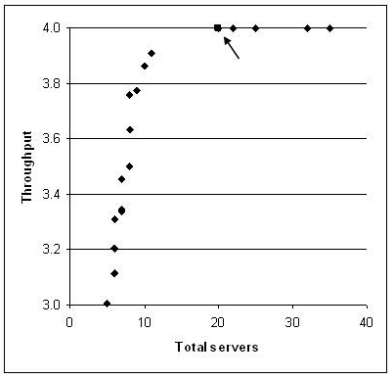


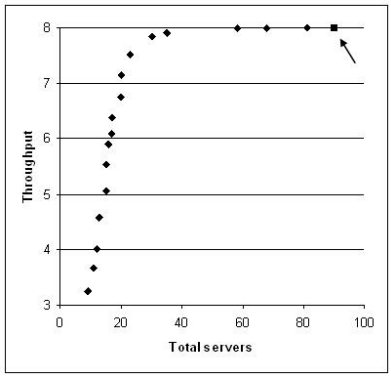

Fig. 6.



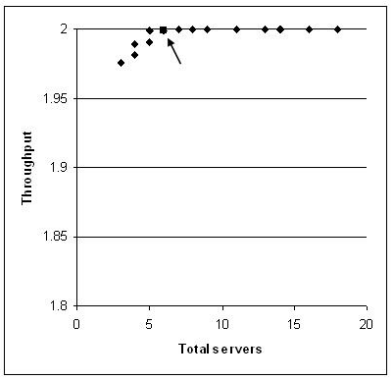

(a) Serial 3 nodes



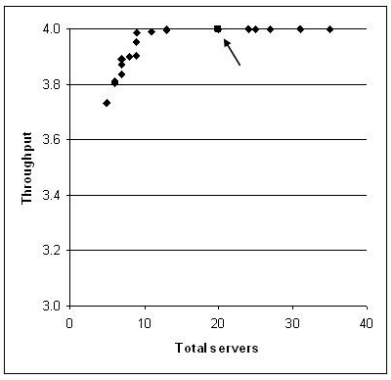

(b) Serial 5 nodes



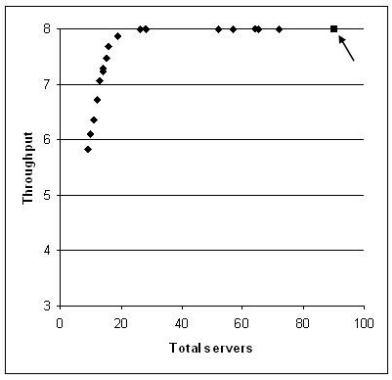

(c) Serial 9 nodes



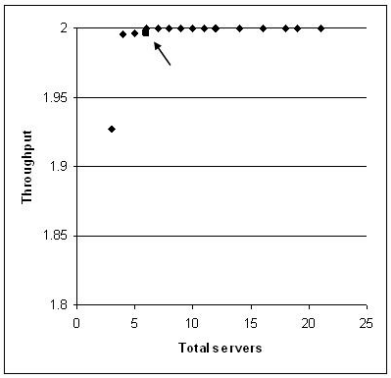

(d) Merge 3 nodes



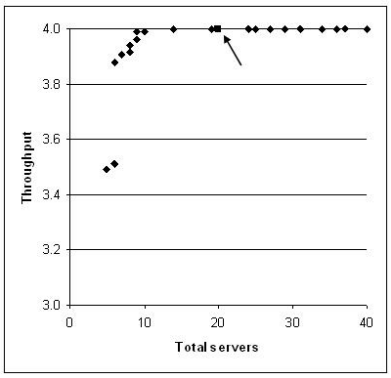

(e) Merge 5 nodes



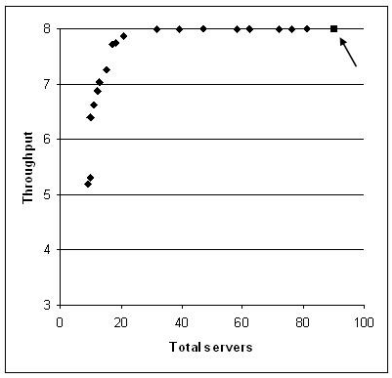

(f) Merge 9 nodes




(g) Split 3 nodes

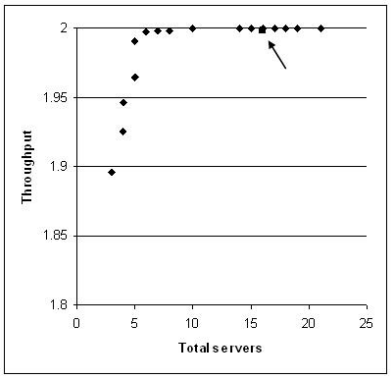



(h) Split 5 nodes

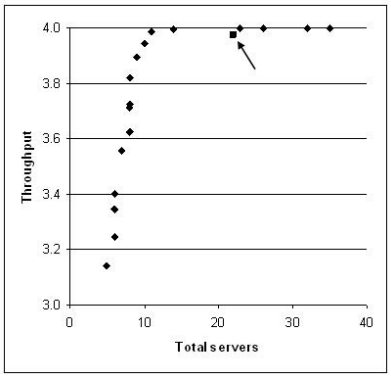



(i) Split 9 nodes

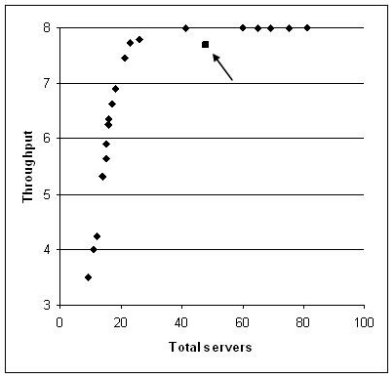
Fig. 7.



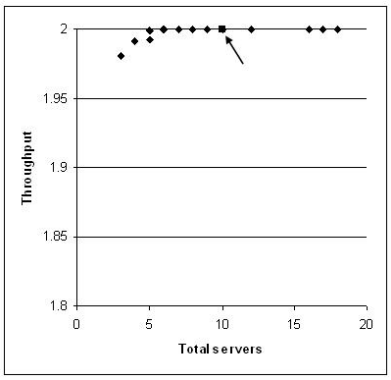

(a) Serial 3 nodes



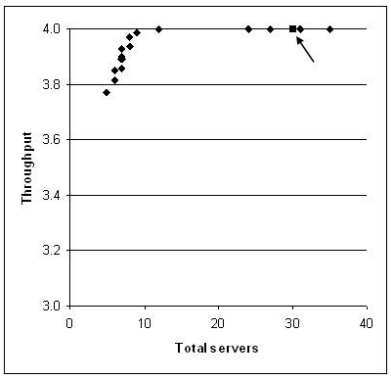

(b) Serial 5 nodes



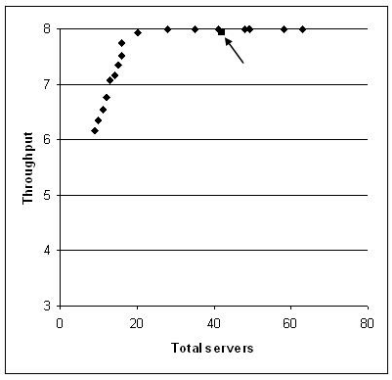

(c) Serial 9 nodes



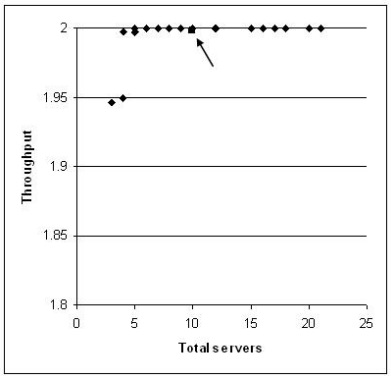

(d) Merge 3 nodes



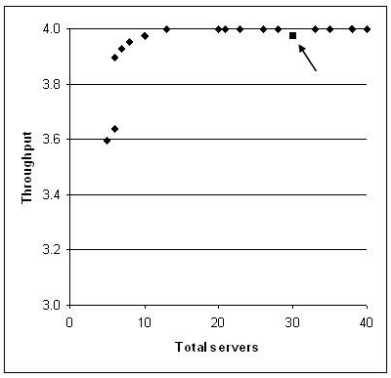

(e) Merge 5 nodes



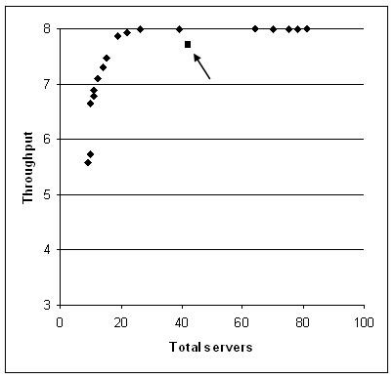

(f) Merge 9 nodes




(g) Split 3 nodes




(h) Split 5 nodes




(i) Split 9 nodes

Fig. 8.

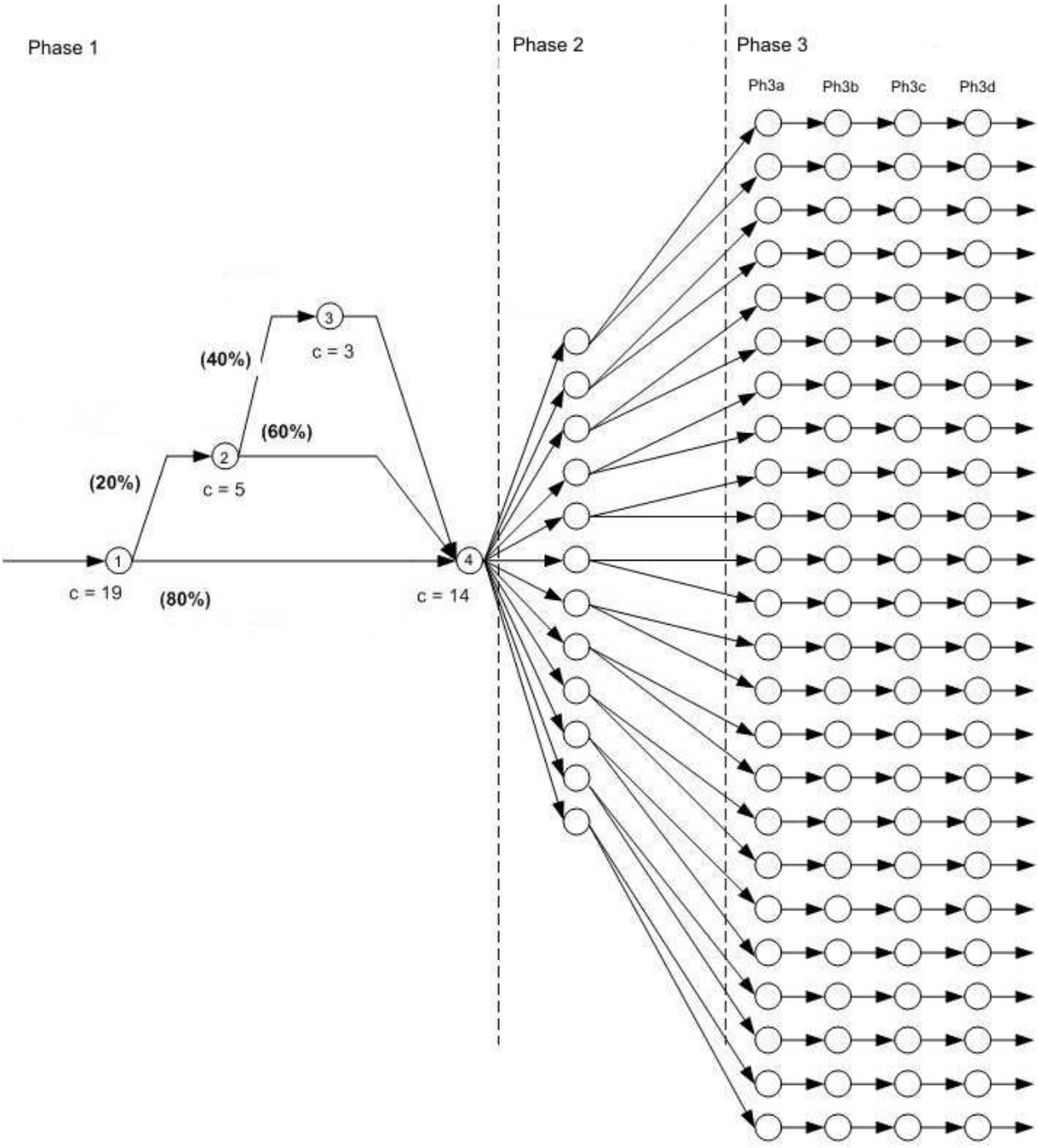

Phase 1
Phase 2
Phase 3
Ph3a
Ph3b
Ph3c
Ph3d
1
2
3
4
c = 19
c = 5
c = 3
c = 14
(20%)
(80%)
(40%)
(60%)

Fig. 9.

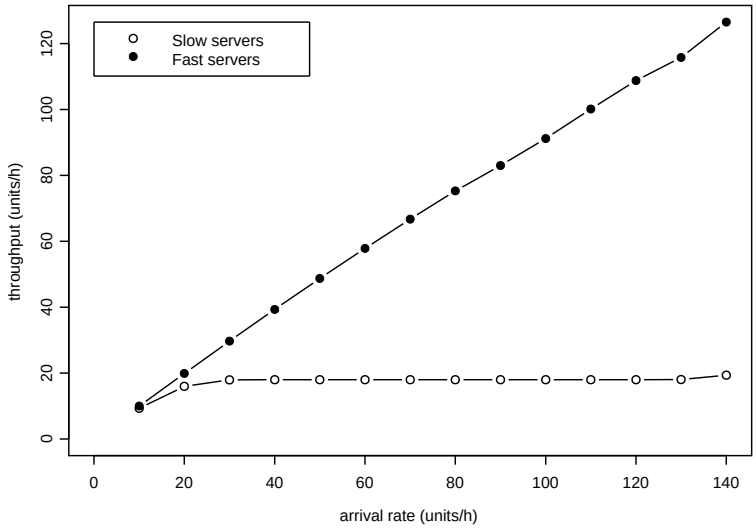

Slow servers
Fast servers
throughput (units/h)
arrival rate (units/h)
0
20
40
60
80
100
120
140

Fig. 10.

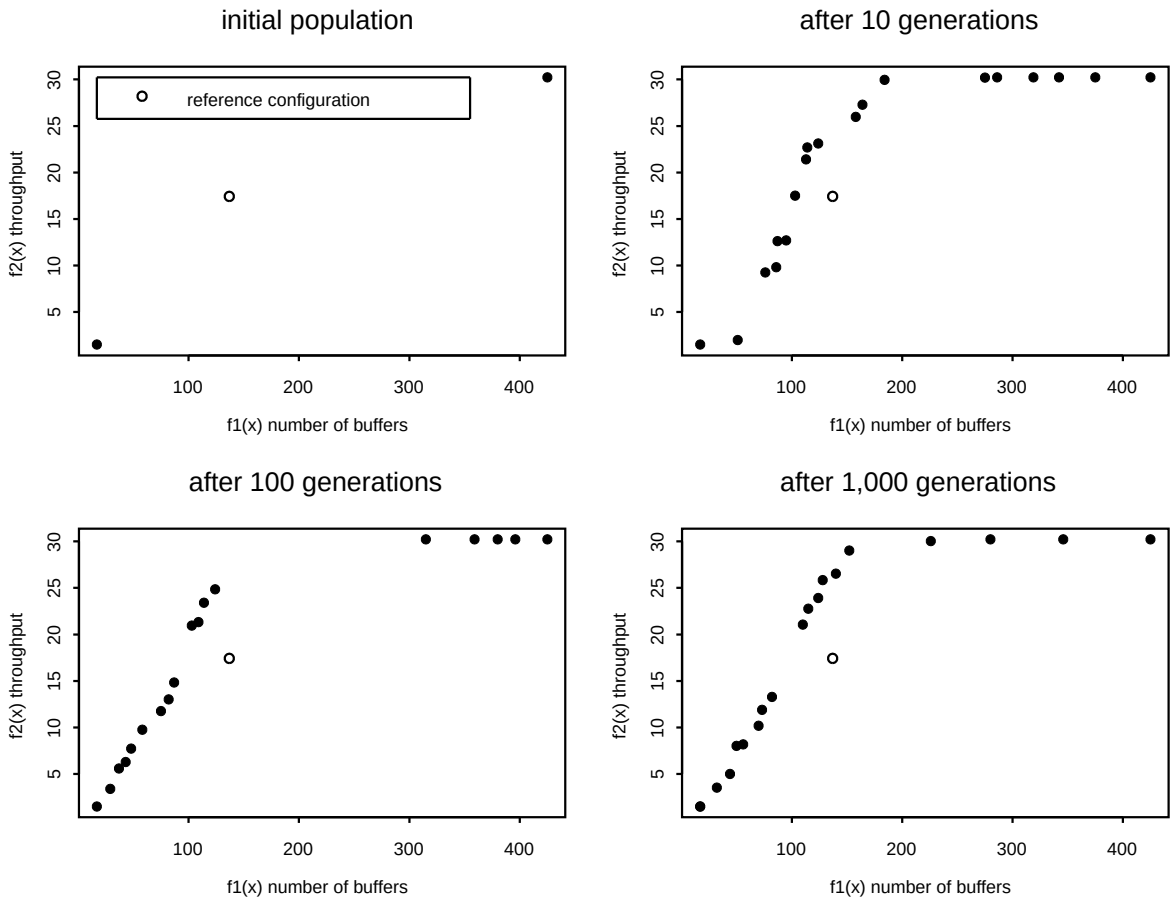

initial population
reference configuration
f2(x) throughput
f1(x) number of buffers
after 10 generations
f2(x) throughput
f1(x) number of buffers
after 100 generations
f2(x) throughput
f1(x) number of buffers
after 1,000 generations
f2(x) throughput
f1(x) number of buffers

Fig. 11.

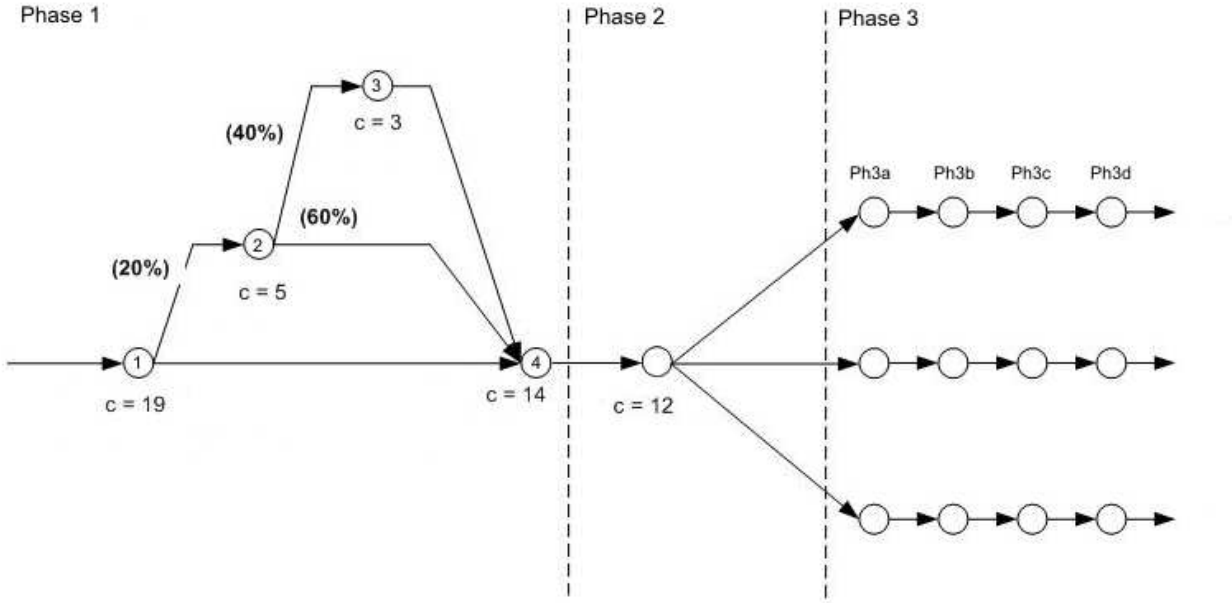

Phase 1
Phase 2
Phase 3
3
c = 3
(40%)
(60%)
2
(20%)
c = 5
1
c = 19
4
c = 14
c = 12
Ph3a
Ph3b
Ph3c
Ph3d

Fig. 12.

## Tables

Table 1

| Variable | Description |
|---|---|
| $h_j$ | The holding node added to the network, preceding each finite node $j$ |
| $\lambda_j$ | The arrival rate to node $j$ |
| $\widetilde{\lambda_j}$ | The effective arrival rate to node $j$ |
| $\mu_j$ | The service rate at node $j$ |
| $\widetilde{\mu_j}$ | The effective service rate at node $j$ |
| $p_{c_j}$ | Blocking probability of finite queue of size $c_j$ |
| $p'_{c_j}$ | Feedback blocking probability in the GEM |
| $\mu_{h_j}$ | The service rate at the holding node $h$ |
| $c^2_{Dj}$ | The squared coefficient of variation of the departure process from node $j$ |
| $c^2_{ai,j}$ | The squared coefficient of variation of the arrival process to the blocked node $j$ |
| $c^2_{ah_j}$ | The squared coefficient of variation of the arrival process at the holding node $h_j$ |
| $c^2_{sh_j}$ | The squared coefficient of variation of the service process at the holding node $h_j$ |

Table 2

| Series | $N=3$ | | | $N=5$ | | | $N=9$ | | |
|---|---|---|---|---|---|---|---|---|---|
| **c** | (2,2,2) | (4,4,4) | (10,10,10) | (2,...,2) | (4,...,4) | (10,...,10) | (2,...,2) | (4,...,4) | (10,...,10) |
| $\lambda=2$ | $1.996^{(*)}$ | 2.000 | 2.000 | 1.996 | 2.000 | 2.000 | 1.995 | 2.000 | 2.000 |
| $\lambda=4$ | 3.949 | 4.000 | 4.000 | 3.940 | $4.000^{(*)}$ | 4.000 | 3.924 | 4.000 | 4.000 |
| $\lambda=8$ | 7.394 | 7.988 | 8.000 | 7.255 | 7.988 | 8.000 | 7.009 | 7.987 | $8.000^{(*)}$ |
| $\lambda=16$ | $11.49^{(*)}$ | 15.63 | 16.00 | 10.676 | 15.60 | 16.00 | 9.572 | 15.56 | 16.00 |
| Merge | $N=3$ | | | $N=5$ | | | $N=9$ | | |
| **c** | (2,2,2) | (4,4,4) | (10,10,10) | (2,...,2) | (4,...,4) | (10,...,10) | (2,...,2) | (4,...,4) | (10,...,10) |
| $\lambda=2$ | $2.000^{(*)}$ | 2.000 | 2.000 | 2.000 | 2.000 | 2.000 | 2.000 | 2.000 | 2.000 |
| $\lambda=4$ | 3.993 | 4.000 | 4.000 | 3.993 | $4.000^{(*)}$ | 4.000 | 3.992 | 4.000 | 4.000 |
| $\lambda=8$ | 7.900 | 7.999 | 8.000 | 7.892 | 7.999 | 8.000 | 7.875 | 7.999 | $8.000^{(*)}$ |
| $\lambda=16$ | 14.52 | 15.98 | 16.00 | 14.40 | $15.98^{(*)}$ | 16.00 | 14.16 | 15.98 | 16.00 |
| Split | $N=3$ | | | $N=5$ | | | $N=9$ | | |
| **c** | (2,2,2) | (4,4,4) | (10,10,10) | (2,...,2) | (4,...,4) | (10,...,10) | (2,...,2) | (4,...,4) | (10,...,10) |
| $\lambda=2$ | $1.997^{(*)}$ | 2.000 | 2.000 | 1.997 | 2.000 | 2.000 | 1.997 | 2.000 | 2.000 |
| $\lambda=4$ | 3.957 | 4.000 | 4.000 | 3.957 | $4.000^{(*)}$ | 4.000 | 3.956 | 4.000 | 4.000 |
| $\lambda=8$ | 7.538 | 7.988 | 8.000 | 7.531 | 7.988 | 8.000 | 7.517 | 7.988 | $8.000^{(*)}$ |
| $\lambda=16$ | 12.59 | 15.65 | 16.00 | 12.52 | 15.65 | 16.00 | 12.38 | 15.65 | $16.00^{(*)}$ |

$^{(*)}$Earmarked experiments confirmed by simulation (see Table 3).

Table 3

| | | | Analytical | Simulation | | | |
|---|---|---|---|---|---|---|---|
| Topology | $\lambda$ | **c** | $\theta$ | $\theta^s$ | $\delta$ | CPU(min) | $\Delta\%\theta$ |
| Series | 2.0 | (2,2,2) | 1.996 | 1.966 | 0.001 | 2.0 | 1.51% |
| | 4.0 | (4,4,4,4,4) | 4.000 | 3.999 | 0.002 | 5.2 | 0.03% |
| | 8.0 | (10,...,10) | 8.000 | 8.002 | 0.003 | 36 | -0.02% |
| | 16.0 | (2,2,2) | 11.49 | 11.32 | 0.002 | 22 | 1.50% |
| Merge | 2.0 | (2,2,2) | 2.000 | 1.991 | 0.002 | 1.9 | 0.44% |
| | 4.0 | (4,4,4,4,4) | 4.000 | 3.999 | 0.002 | 5.8 | 0.03% |
| | 8.0 | (10,...,10) | 8.000 | 7.998 | 0.003 | 20 | 0.03% |
| | 16.0 | (4,4,4,4,4) | 15.98 | 15.88 | 0.003 | 24 | 0.63% |
| Split | 2.0 | (2,2,2) | 1.997 | 1.967 | 0.002 | 2.1 | **1.55%** |
| | 4.0 | (4,4,4,4,4) | 4.000 | 3.996 | 0.002 | 5.7 | 0.09% |
| | 8.0 | (10,...,10) | 8.000 | 7.999 | 0.003 | 25 | 0.02% |
| | 16.0 | (10,...,10) | 16.00 | 16.00 | 0.004 | 150 | 0.00% |

Table 4

| Series | $N=3$ | $N=5$ | $N=9$ |
|---|---|---|---|
| $\mathbf{c}$ | (2,4,10) | (2,4,10,2,4) | (2,4,10,2,4,10,2,4,10) |
| $\boldsymbol{\mu}$ | (12,11,10) | (12,11,10,12,11) | (12,11,10,12,11,10,12,11,10) |
| $\lambda=2$ | $1.998^{(*)}$ | 1.998 | 1.998 |
| $\lambda=4$ | 3.974 | $3.974^{(*)}$ | 3.974 |
| $\lambda=8$ | 7.698 | 7.697 | $7.696^{(*)}$ |
| $\lambda=16$ | $13.50^{(*)}$ | 13.47 | 13.45 |
| Merge | $N=3$ | $N=5$ | $N=9$ |
| $\mathbf{c}$ | (4,4,2) | (10,10,4,4,2) | (2,2,4,4,10,10,4,4,2) |
| $\boldsymbol{\mu}$ | (11,11,12) | (10,10,11,11,12) | (12,12,11,11,10,10,11,11,12) |
| $\lambda=2$ | $2.000^{(*)}$ | 2.000 | 2.000 |
| $\lambda=4$ | 4.000 | $4.000^{(*)}$ | 3.996 |
| $\lambda=8$ | 8.000 | 8.000 | $7.947^{(*)}$ |
| $\lambda=16$ | 15.92 | $15.93^{(*)}$ | 15.35 |
| Split | $N=3$ | $N=5$ | $N=9$ |
| $\mathbf{c}$ | (2,4,4) | (2,4,4,10,10) | (2,4,4,10,10,4,4,2,2) |
| $\boldsymbol{\mu}$ | (12,11,11) | (12,11,11,10,10) | (12,11,11,10,10,11,11,12,12) |
| $\lambda=2$ | $1.998^{(*)}$ | 1.998 | 1.998 |
| $\lambda=4$ | 3.974 | $3.974^{(*)}$ | 3.974 |
| $\lambda=8$ | 7.698 | 7.698 | $7.698^{(*)}$ |
| $\lambda=16$ | 13.51 | 13.51 | $13.51^{(*)}$ |

$^{(*)}$Earmarked experiments confirmed by simulation (see Table 5).

Table 5

| | | | | Analytical | Simulation | | | |
|---|---|---|---|---|---|---|---|---|
| Topology | $\lambda$ | $\mathbf{c}$ | $\boldsymbol{\mu}$ | $\theta$ | $\theta^s$ | $\delta$ | CPU(min) | $\Delta\%\theta$ |
| Series | 2.0 | (2,4,10) | (12,11,10) | 1.998 | 1.977 | 0.001 | 3.0 | 1.07% |
| | 4.0 | (2,4,10,2,4) | (12,11,10,12,11) | 3.974 | 3.839 | 0.002 | 9.9 | 3.52% |
| | 8.0 | (2,4,10,2,4,10,2,4,10) | (12,11,10,12,11,10,12,11,10) | 7.969 | 7.759 | 0.002 | 35 | 2.70% |
| | 16.0 | (2,4,10) | (12,11,10) | 13.50 | 13.50 | 0.002 | 43 | 0.00% |
| Merge | 2.0 | (4,4,2) | (11,11,12) | 2.000 | 2.001 | 0.002 | 1.9 | -0.04% |
| | 4.0 | (10,10,4,4,2) | (10,10,11,11,12) | 4.000 | 3.999 | 0.002 | 5.8 | 0.02% |
| | 8.0 | (2,2,4,4,10,10,4,4,2,2) | (12,12,11,11,10,10,11,11,12) | 7.947 | 7.652 | 0.003 | 20 | 3.86% |
| | 16.0 | (10,10,4,4,2) | (10,10,11,11,12) | 15.93 | 16.00 | 0.005 | 29 | -0.44% |
| Split | 2.0 | (2,4,4) | (12,11,11) | 1.998 | 1.973 | 0.001 | 2.1 | 1.27% |
| | 4.0 | (2,4,4,10,10) | (12,11,11,10,10) | 3.974 | 3.980 | 0.002 | 6.1 | -0.15% |
| | 8.0 | (2,4,4,10,10,4,4,2,2) | (12,11,11,10,10,11,11,12,12) | 7.698 | 7.654 | 0.002 | 22 | 0.57% |
| | 16.0 | (2,4,4,10,10,4,4,2,2) | (12,11,11,10,10,11,11,12,12) | 13.51 | 12.88 | 0.030 | 38 | **4.89%** |

Table 6

| Server | Number of servers | Slow servers (units/h) | Fast servers (units/h) |
|---|---|---|---|
| Phase 1 server 1 | 19 | 5.833 | 15.42 |
| Phase 1 server 2 | 5 | 3.400 | 11.05 |
| Phase 1 server 3 | 3 | 3.400 | 11.05 |
| Phase 1 server 4 | 14 | 3.400 | 20.00 |
| Phase 2 servers | 12 | 2.000 | 26.25 |
| Ph3a servers | 21 | 7.500 | 23.63 |
| Ph3b servers | 21 | 2.213 | 22.13 |
| Ph3c servers | 21 | 2.213 | 22.13 |
| Ph3d servers | 21 | 2.213 | 22.13 |

Table 7

| | | Analytical | Simulation | | | |
|---|---|---|---|---|---|---|
| | $\lambda$ | $\theta$ | $\theta^s$ | $\delta$ | CPU (min) | $\Delta\%\theta$ |
| Slow servers | 30.215 | 17.422 | 18.464 | 0.02 | 120 | -5.6% |
| Fast servers | 30.215 | 29.907 | 30.213 | 0.006 | 70.1 | -1.0% |

Table 8

| Server | Configuration #1 (units/h) | Configuration #2 (units/h) | Configuration #3 (units/h) |
|---|---|---|---|
| Phase 1 server 1 | 5.833 | 5.833 | 5.833 |
| Phase 1 server 2 | 3.400 | 3.400 | 3.400 |
| Phase 1 server 3 | 3.400 | 3.400 | 3.400 |
| Phase 1 server 4 | 3.400 | 3.400 | 3.400 |
| Phase 2 servers | 2.000 | 6.000 | 6.000 |
| Ph3a servers | 7.500 | 7.500 | 7.500 |
| Ph3b servers | 2.213 | 2.213 | 3.320 |
| Ph3c servers | 2.213 | 2.213 | 3.320 |
| Ph3d servers | 2.213 | 3.320 | 3.320 |
| **Throughput** | **17.422** | **18.480** | **20.124** |
| **Improvement** | | **6.1%** | **16%** |