Computers & Operations Research 37 (2010) 1472–1487

Contents lists available at ScienceDirect

Computers & Operations Research

journal homepage: www.elsevier.com/locate/caor

Performance optimization of open zero-buffer multi-server queueing networks

R. Andriansyah [a], T. Van Woensel [b,*], F.R.B. Cruz [c,1], L. Duczmal [d]

[a] Department of Mechanical Engineering, Eindhoven University of Technology, Eindhoven, The Netherlands
[b] Department of Technology Management, Operations, Planning, Accounting and Control, Eindhoven University of Technology, Eindhoven, The Netherlands
[c] School of Computer Science and Information Technology, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham NG8 1BB, UK
[d] Department of Statistics, Federal University of Minas Gerais, 31270-901 Belo Horizonte, MG, Brazil

ARTICLE INFO

Available online 3 December 2009

Keywords:
Zero buffer systems
Pareto efficiency
Performance evaluation and optimization
Genetic algorithms

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 evaluate the performance of these systems in terms of throughput using the generalized expansion method (GEM) and compare our results with simulation. Secondly, we embed the performance evaluation in a multi-objective optimization setting. This multi-objective optimization approach results in the Pareto efficient curves showing the trade-off between the total number of servers used and the throughput. Experiments for a large number of settings and different network topologies are presented in detail.

1. Introduction and motivation

Bufferless networks occur throughout a number of real-life physical systems in the semi-process and process industries (refer to Fransoo and Rutten [1], 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.

In their paper, Hall and Sriskandarajah [2] describe a steel production process. Molten steel goes through a series of operations ranging from ingots, un-molding, reheating, soaking, and preliminary rolling. In this production process, the steel must pass 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. Similarly, in food-processing environments 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 (e.g., see Fey [3]). In these cited examples, *restrictions in the processing technology* and its characteristics creates zero-buffer production system.

Ramudhin and Ratliff [4] 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 [5]. 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 *simple 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 focusing exclusively on these types of networks. Our paper reviews the current work in the field, completes and complements the literature where necessary.

The system of interest in this paper is a *zero-buffer multi-server general queueing network* (as shown in Fig. 1 for a queueing network representation for a tandem line). Zero-buffer systems are a special case of a restricted queueing network. Restricted queueing networks have 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 [6]. As a consequence, each node in the network might be affected by events at other nodes, leading to the

* Corresponding author.
E-mail addresses: r.andriansyah@tue.nl (R. Andriansyah), t.v.woensel@tm.tue.nl (T. Van Woensel), frc@cs.nott.ac.uk (F.R.B. Cruz), duczmal@est.ufmg.br (L. Duczmal).
[1] On sabbatical leave.

0305-0548/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2009.11.004

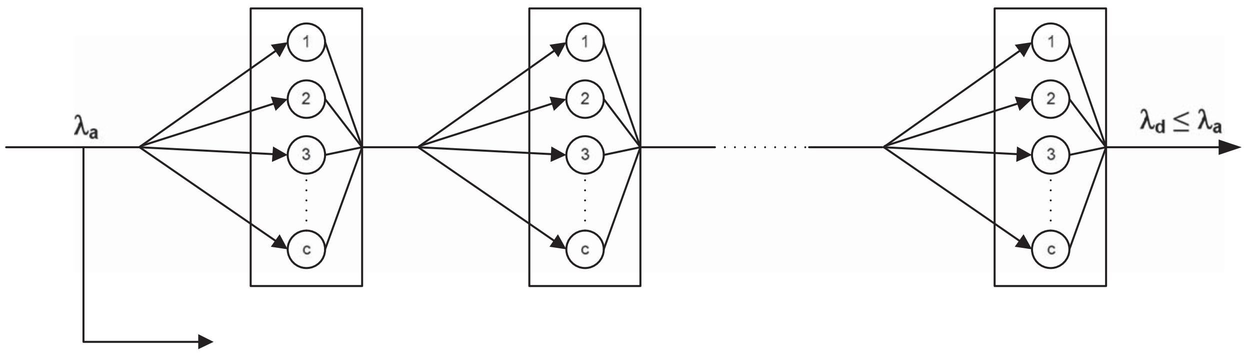

Fig. 1. A zero-buffer queueing network representation.

phenomena of blocking and starvation [7]. 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_j in that node, that is, $K_j = c_j$. 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 Fig. 1).

This type of system operates as follows. Jobs arrive to the first node in the system with a certain arrival rate λ_a. The job is then processed by an available server with service rate μ. 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 thus might be smaller than the arrival rate (i.e., $\lambda_d \le \lambda_a$). Note that if the first node is full, (part of) the arrivals are lost for the system. In this paper, we both evaluate and optimize this type of zero-buffer systems. The networks used in this paper are characterized by combinations of split systems (where one flow splits into two flows), tandem lines, and merge systems (where two flows merge into a single flow). Moreover, the networks analyzed are assumed to be acyclic, i.e., no cycling or feedback loops are allowed as this creates a difficult to handle dependency in the network. A number of methods are available to evaluate the performance of restricted queueing networks, as we will see in Section 2.

In this paper, we will use the generalized expansion method, described in Section 3, as the performance evaluation tool. The optimization is done via a genetic algorithm, a heuristic procedure specially designed for the problem on-hand. 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 networks. 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 lead to a higher throughput. In Section 4, we model this trade-off as a multi-objective optimization problem with two conflicting objectives, minimize the total number of servers allocated in the network and maximize the throughput. This results in highly insightful Pareto sets which explicitly show the trade-off between all objectives.

The main contributions of this paper are the following.

1. The paper on-hand deepens the knowledge on zero-buffer systems. More specifically, this paper *reviews, completes and extends* the current state of the literature. Reviewing the literature, we show that 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 and only few zero-buffer cases were evaluated without much details analyzed (see e.g., Jain and Smith [8]). We complete this scarce literature by executing a very large number of experiments for these zero-buffer systems and derive relevant generic insights based on this performance evaluation. As such, this paper is very complementary with the current literature. Finally, we extend the current state of the literature developing a multi-objective optimization method designed for the zero-buffer system.
2. We compare our results with simulation, and show that the generalized expansion method (GEM) proposed by Kerbache and Smith [9] gives an excellent approximation for the performance measures of the zero-buffer systems studied. We show that the GEM delivers results with an accuracy mostly less than 5% (with maxima up to 16% for heavy congested systems, similar as in [8]), for basic series, merge, and split topologies, both symmetrical and asymmetrical.
3. We present a multi-objective optimization methodology for zero-buffer systems, which provides particularly insightful results. It is the first time to our knowledge that a multi-objective approach is considered for the optimization of networks of bufferless systems. The proposed methodology is proven to be successful to derive Pareto sets, which among other things showed that although some optimal configurations could be correctly 'guessed' (mainly for symmetrical networks), this is not always the case, as seen for asymmetrical networks.
4. 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 acyclic configured zero-buffer system. Based on this analysis, we see that our proposed optimization method is a feasible alternative to optimize real-life multi-objective zero-buffer queueing networks and can provide answers to 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 [10]. Queueing networks can be divided into two categories [7], unrestricted and restricted queueing networks. 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. Restricted queueing networks often represent real-life systems, where there normally exist finite spaces for holding entities. In this paper, we focus on a special case of restricted queueing networks where the finite space is zero and thus blocking might occur.

In general, three blocking mechanisms can be distinguished in restricted queueing networks, blocking-after-service (BAS), blocking-before-service (BBS), and repetitive-service (RS). Most production lines operate under the BAS system, which will be the only blocking mechanism assumed here. Moreover, in the literature it is the most common assumption regarding buffer behavior [11].

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 [12–16]. 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 generalized expansion method (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.

There are some other papers dealing with a similar type of networks but with different server settings. For example, Cruz et al. [17] describe in their paper a performance optimization methodology for open finite single-server queueing networks, focusing on networks of single-server queues, $M/G/1/K$, and a single-objective optimization derivative-free search algorithm. Cheah and Smith [18], Jain and Smith [19], Cruz et al. [20], and Cruz and Smith [21] choose to deal with $M/G/c/c$ state-dependent queueing networks, that is, queues with Markovian arrivals, general state-dependent service rates, c parallel servers, and the total capacity c, including the servers. It was observed that the service rates in (pedestrian) traffic flow network applications were exhibiting a certain state-dependent behavior, i.e., congestion has an influence on the service rate. Finally, it is worthwhile mentioning the paper from Jain and Smith [8], in which few experiments were reported for zero-buffer multi-server queueing networks.

3. Performance evaluation of zero buffer networks

The GEM is a robust and effective approximation technique developed by Kerbache and Smith [22]. 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 blocking after service, which is prevalent in most production and manufacturing, transportation, and other similar systems, as said earlier. In this section, we present an overview of the method. For more detailed information and applications of the GEM, the reader is referred to the papers by Kerbache and Smith [9,22,23], Jain and Smith [8], Spinellis et al. [24], and Smith and Cruz [25]. 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.

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 [9].

Following Kerbache and Smith [23], 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 Fig. 2). 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:

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

Table 1
Variables used in this paper.

Variable	Description
h_j	The holding node added to the network, preceding each finite node j
λ_j	The arrival rate to node j
$\tilde{\lambda}_j$	The effective arrival rate to node j
μ_j	The service rate at node j
$\tilde{\mu}_j$	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
μ_{h_j}	The service rate at the holding node h_j

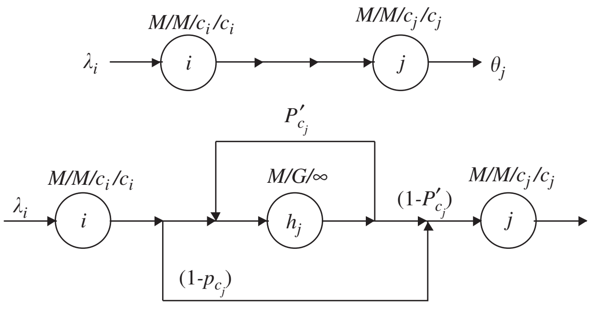

Fig. 2. The generalized expansion method.

3.2. Parameter estimation

In the second stage, the parameter estimation of all unknowns for the system described above are determined. More specifically, the following equations and the relevant references are used.

1. Utilizing the analytical results for the $M/M/c/c$, the probability of a customer being blocked, p_c (for clarity, we omit the subscript for node j), is provided by the following expression (also known as Erlang B formula):

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

2. The probability that customers are forced back to the holding node given that they were rejected at the previous trial, p_c', is not available in closed form. Then, we use an approximation given by Labetoulle and Pujolle [26] obtained using diffusion techniques:

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

in which r_1 and r_2 are the roots to the polynomial:

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

with $\lambda=\lambda_j-\lambda_h(1-p_c')$ and λ_j and λ_h are the actual arrival rates to the finite and artificial holding nodes respectively. Labetoulle and Pujolle [26] illustrate in their paper a comparison of their method for computing p_c' with an Erlang service system and an hyper-exponential system and it is shown that the calculation for p_c' is very reasonable for general service systems. Given these results, we felt comfortable in applying p_c' for the general service situation.
In fact, the arrival rate to the finite node j is given by:

$$\lambda_j = \tilde{\lambda}_i(1-p_c) = \tilde{\lambda}_i - \lambda_h.$$

Let us examine the following argument to determine the service time at the artificial node. If an arriving customer is blocked, the queue is full and thus a customer is being serviced, so the arriving customer to the holding node has to remain in service at the artificial holding node for the remaining service time interval of the customer in service. The delay distribution of a blocked customer at the holding node has the same distribution as the remaining service time of the customer being serviced at the node doing the blocking. Using renewal theory, one can show that the remaining service time distribution has the following rate μ_h:

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

where σ_j^2 is the service time variance given by Kleinrock [27]. Notice that if the service time distribution at the finite queue doing the blocking is exponential with rate μ_j, then:

$$\mu_h = \mu_j,$$

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

For more details, the reader is referred to the paper by Jain and Smith [8].

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 μ_h' then becomes:

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

3.4. 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', and μ_h^{-1}, along with auxiliary variables such as μ and $\tilde{\lambda}_i$. Solving these equations simultaneously we can compute all the performance measures of the network:

$$\lambda = \lambda_j - \lambda_h(1-p_c'), \tag{1}$$

$$\lambda_j = \tilde{\lambda}_i(1-p_c), \tag{2}$$

$$\lambda_j = \tilde{\lambda}_i - \lambda_h, \tag{3}$$

$$p_c'^{\,-1} = \left[\frac{\mu_j+\mu_h}{\mu_h} - \frac{\lambda[(r_2^c-r_1^c)-(r_2^{c-1}-r_1^{c-1})]}{\mu_h[(r_2^{c+1}-r_1^{c+1})-(r_2^c-r_1^c)]}\right], \tag{4}$$

$$z = (\lambda+2\mu_h)^2 - 4\lambda\mu_h, \tag{5}$$

$$r_1 = \frac{[(\lambda+2\mu_h)-z^{1/2}]}{2\mu_h}, \tag{6}$$

$$r_2 = \frac{[(\lambda+2\mu_h)+z^{1/2}]}{2\mu_h}, \tag{7}$$

$$p_c = \frac{\left(\frac{\lambda}{\mu}\right)^c \Big/ c!}{\sum_{i=0}^{c} \left(\frac{\lambda}{\mu}\right)^i \Big/ i!}. \tag{8}$$

Eqs. (1)–(4) are related to the arrivals and feedback in the *holding* node. Eqs. (5)–(7) are used for solving Eq. (4) with z used as a dummy parameter for simplicity of the solution. Lastly, Eq. (8) gives the blocking probability for the $M/M/c/c$ queue. Hence, we essentially have five equations to solve viz. Eqs. (1)–(4) and Eq. (8).

Recapitulating, we first expand the network. Then, the blocking probabilities and the service delay in the artificial holding node are approximated. Finally the feedback arc at the holding node is eliminated and the actual service rate of each node preceded by a finite node is corrected. Once these three stages are complete, we have an expanded network which can then be used to compute the performance measures for the original network. As a decomposition technique this approach allows successive addition of a holding node for every finite node, estimation of the parameters, and subsequent elimination of the holding node. 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, preceded by a finite node, is then obtained as follows (for clarity, we omit the subscripts):

$$\theta = \lambda(1-p_c) = \lambda\left(1-\frac{(\lambda/\tilde{\mu})^c/c!}{\sum_{i=0}^{c}(\lambda/\tilde{\mu})^i/i!}\right). \tag{9}$$

The throughput of the overall queueing network is the sum of all throughput(s) obtained at the last node(s) of the network.

As a final note, 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, Eq. (8). Note that simple approximations such as the one recently developed by Adelman [28] could also be used. It turns out that Eq. (8) 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 [29]). This means that all results presented here, also immediately hold for the general service-time distribution case.

4. Multi-objective optimization of zero-buffer networks

In this section, we consider the multi-objective optimization of zero-buffer systems where the GEM is incorporated in a genetic algorithm (GA) approach.

4.1. Model formulation

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

(MOO):

$$\text{minimize } F(\mathbf{x}) = (f_1(\mathbf{x}), -f_2(\mathbf{x}))^{\mathrm{T}}, \tag{10}$$

subject to

$$x_i \in \{1, 2, \ldots\}, \ \forall i \in \{1, 2, \ldots, N\}, \tag{11}$$

in which $f_1(\mathbf{x}) = \sum_{i=1}^{N} x_i$, for $x_i \equiv c_i$, is the server allocation to node i, $f_2(\mathbf{x}) = \Theta(\mathbf{x})$ is the throughput profit, and N is the number of nodes in the network.

In the above formulation, we make the trade-off between throughput versus total number of servers. Of course, instead of the throughput, we could have used the cycle time or other performance measure. 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, such as 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 [25], for a similar approach).

4.2. A genetic algorithm

To solve the MOO problem formulation, Eqs. (10)–(11), we use a powerful class of optimization heuristics called genetic algorithm (GA). GAs were chosen because they are a good fit for multi-objective optimization, as demonstrated by many articles recently published in the area (e.g., see Purshouse and Fleming [30], and references therein). Of course, any other multi-objective optimization approach could be used for the same purpose, but the comparison and evaluation of different optimization approaches for our problem on-hand is not the subject of this paper.

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, also known as *efficient* solutions. The resulting Pareto curve allows one to see the trade-off among the objective function values from different solutions. That is, a GA allows for evaluating the change in one objective (e.g., the throughput) compared to a simultaneous change in another objective (e.g., the total server allocation).

The instance of GA used in this article is presented in Fig. 3. In short, GAs are optimization algorithms to perform an approximate global search relaying on the information obtained from the evaluation of several points, usually in a high dimensional search space, and obtaining a population of points that converges to the optimum through the application of the so-called genetic operators, *mutation*, *crossover*, *selection*, and *elitism*. For each implementation of these operators, a particular instance of GA is derived.

Following the steps presented in the algorithm from Fig. 3, the setting of the problem to be solved is read and an initial population is generated, composed by (i) the extreme solution $\mathbf{c} = (1, 1, \ldots, 1)$, (ii) the other extreme solution $\mathbf{c} = (M, M, \ldots, M)$, with M being a big enough number, and (iii) arbitrary solutions uniformly distributed between these two extremes. Once two parents are chosen from the population, an offspring is generated.

Specifically for the problem on hand, we find that a simple fitness-oriented *crossover* scheme does what is necessary for a satisfactory convergence speed, although, sometimes, the crossover step is not even required (that is, the genetic algorithm performs simply a pure random search; for instance, see the case of a bi-level linear programming successfully solved by Mathieu et al. [31], by means of a pure random search GA). The crossover used here starts by selecting a pair of individuals (solutions) with a probability proportional to their fitness to function $f_2(\mathbf{x})$, that is, the throughput delivered by the respective solution, $\Theta(\mathbf{x})$. The crossover position is then selected by uniform distribution and the two individuals are mixed accordingly.

Mutation happens at a specific rate, `RateMut`, for each one of the gens of the individuals, and represents and increment or decrement in c_i, that is, ± 1, uniformly distributed. Notice that after crossover and mutation, constraint (11) must hold, in order to guarantee feasibility. If not, the values are adjusted accordingly. In other words, the scheme proposed is guaranteed to generate

```
algorithm
  readgraph, arrival, servicerates, $G(V, A)$, $\lambda_v$, $\mu_v$, $\forall_v \in V$
  /*generate initial population*/
  $P_1 \leftarrow$ GetInit Population (pop Size)
  for $i = 1$ until numGen do
    /*generate offspring by crossover and mutation*/
    $Qi \leftarrow$ Make New Population (pop Size, $Pi$)
    /*combine parent and offspring*/
    $R_t \leftarrow Pt \cup Q_t$
    /*select new population*/
    $P_{t+1} \leftarrow$ Select New Population (pop Size, $Rt$)
  end for
  write $P_{\text{numGen}+1}$
end algorithm
```

Fig. 3. A genetic algorithm.

only feasible (i.e., integer) solutions, if it is fed with initial feasible solutions.

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 NSGA-II [32]. It has been shown that NSGA-II requires less computational effort as compared to other GA algorithms. For a complete elaboration on this algorithm, see Deb et al. [32]. We will use a GA as the search method, in combination with the GEM as the performance evaluation method. As a final remark, it is well-known that critical to the convergence of GAs are the mutation rate, `RateMut`, the population size, `popSize`, and the number of generations, `numGen`. The effect and iterations of these parameters were empirically tested and will be presented shortly in the following section.

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. Quality assessment of the performance evaluation

To assess the quality of the GEM as an approximate performance evaluation tool of our queueing networks, we conducted experiments using four different topologies, namely series, split, merge, and mixed topologies. We examine some of 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 Figs. 4–7.

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 Fig. 4). In the split topology (see Fig. 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 probabilities for both routes are 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 Fig. 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.

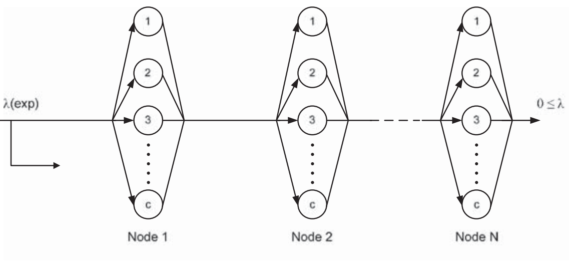

Fig. 4. Series topology.

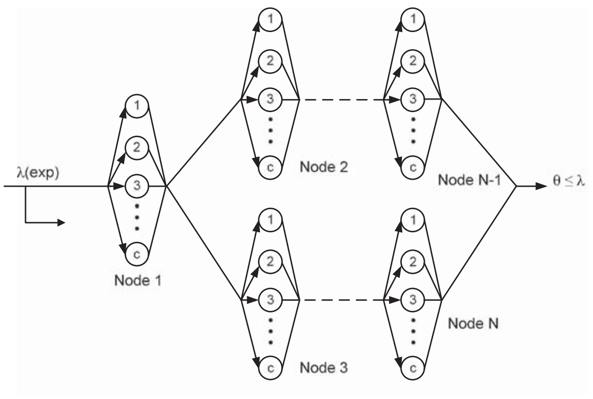

Fig. 5. Split topology.

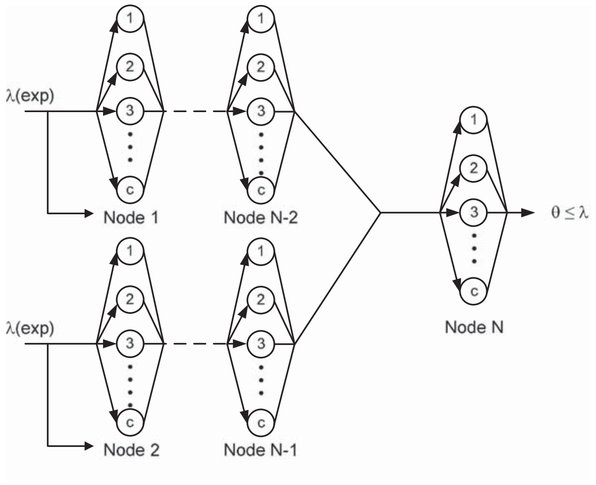

Fig. 6. Merge topology.

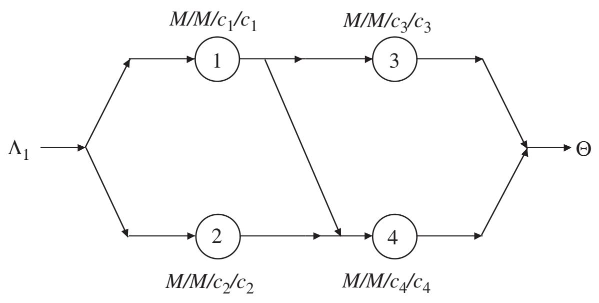

Fig. 7. Yet another network structure.

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 rates are significantly lower than the arrival rates, in opposition to less congested settings. This rule holds true for all three types of topologies involved in the experiment. Similarly, the lower the

Table 2
Throughput θ for the symmetrical cases.

	$N=3$			$N=5$			$N=9$		
c	(2,2,2)	(4,4,4)	(10,10,10)	$(2,\ldots,2)$	$(4,\ldots,4)$	$(10,\ldots,10)$	$(2,\ldots,2)$	$(4,\ldots,4)$	$(10,\ldots,10)$
Series									
$\lambda=2$	1.996[a]	2.000	2.000	1.996	2.000[a]	2.000	1.995	2.000	2.000[a]
$\lambda=4$	3.949[a]	4.000	4.000	3.940	4.000[a]	4.000	3.924	4.000	4.000[a]
$\lambda=8$	7.394[a]	7.988	8.000	7.257	7.988[a]	8.000	7.009	7.987	8.000[a]
$\lambda=16$	11.50[a]	15.63	16.00	10.70	15.61[a]	16.00	9.572	15.56	16.00[a]
Merge									
$\lambda=2$	2.000[a]	2.000	2.000	2.000	2.000[a]	2.000	2.000	2.000	2.000[a]
$\lambda=4$	3.993[a]	4.000	4.000	3.993	4.000[a]	4.000	3.992	4.000	4.000[a]
$\lambda=8$	7.900[a]	7.999	8.000	7.892	7.999[a]	8.000	7.875	7.999	8.000[a]
$\lambda=16$	14.52[a]	15.98	16.00	14.40	15.98[a]	16.00	14.16	15.98	16.00[a]
Split									
$\lambda=2$	1.997[a]	2.000	2.000	1.997	2.000[a]	2.000	1.997	2.000	2.000[a]
$\lambda=4$	3.957[a]	4.000	4.000	3.957	4.000[a]	4.000	3.956	4.000	4.000[a]
$\lambda=8$	7.538[a]	7.988	8.000	7.531	7.988[a]	8.000	7.517	7.988	8.000[a]
$\lambda=16$	12.59[a]	15.65	16.00	12.52	15.65[a]	16.00	12.38	15.65	16.00[a]

[a] Earmarked experiments confirmed by simulation (see Table 3).

Table 3
Simulation results for the symmetrical cases.

Topology	N	**c**	λ	Analytical	Simulation			$\Delta\%\theta$
				θ	θ^s	δ	CPU (min)	
Series	3	(2,2,2)	2	1.996	1.966	0.001	2.0	1.53
			4	3.949[a]	3.952	0.002	5.8	−0.09
			8	7.394	7.424	0.003	12	−0.41
			16	11.50	11.32	0.002	22	1.59
	5	(4,4,4,4,4)	2	2.000	1.999	0.002	5.0	0.05
			4	4.000	3.999	0.002	5.2	0.03
			8	7.988[a]	7.987	0.003	20	0.02
			16	15.61	15.61	0.004	40	0.03
	9	$(10,\ldots,10)$	2	2.000	2.000	0.001	9.0	0.02
			4	4.000	4.000	0.002	18	0.00
			8	8.000	8.002	0.003	36	−0.02
			16	16.00[a]	16.00	0.003	73	−0.01
Merge	3	(2,2,2)	2	2.000	1.991	0.002	1.9	0.45
			4	3.993	3.930	0.001	3.8	1.61
			8	7.900	7.473	0.002	8.2	5.71
			16	14.52	12.54	0.003	14	15.8
	5	(4,4,4,4,4)	2	2.000	1.999	0.001	3.0	0.04
			4	4.000	3.999	0.002	5.8	0.03
			8	7.999	7.993	0.003	12	0.08
			16	15.98	15.88	0.003	24	0.63
	9	$(10,\ldots,10)$	2	2.000	1.999	0.001	5.3	0.04
			4	4.000	4.000	0.003	10	0.05
			8	8.000	7.998	0.003	20	0.03
			16	16.00	16.00	0.005	45	0.00
Split	3	(2,2,2)	2	1.997	1.967	0.002	2.1	1.53
			4	3.957	3.783	0.002	4.8	4.59
			8	7.538	6.785	0.002	8.9	11.1
			16	12.59	10.65	0.002	15	18.2
	5	(4,4,4,4,4)	2	2.000	2.000	0.001	4.5	0.00
			4	4.000	3.996	0.002	5.7	0.10
			8	7.988	7.936	0.003	17	0.65
			16	15.65	15.09	0.004	28	3.69
	9	$(10,\ldots,10)$	2	2.000	2.000	0.002	12	0.00
			4	4.000	3.999	0.002	13	0.02
			8	8.000	7.999	0.003	25	0.01
			16	16.00	16.00	0.004	150	0.00

[a] Earmarked results presented in Fig. 10.

Table 4
Simulation results for the asymmetrical cases.

Topology	**c**	$\boldsymbol{\mu}$	λ	Analytical	Simulation			$\Delta\%\theta$
				θ	θ^s	δ	CPU (min)	
Series	(2,4,10)	(12,11,10)	2	1.998	1.977	0.001	3.0	1.06
			4	3.974[a]	3.973	0.002	6.3	0.02
			8	7.698	7.698	0.002	12	0.00
			16	13.50	13.50	0.002	43	0.00
	(2,4,10,2,4)	(12,11,10,12,11)	2	1.998	1.997	0.001	5.0	0.04
			4	3.974	3.839	0.002	9.9	3.52
			8	7.697[a]	7.700	0.003	20	−0.04
			16	13.47	13.50	0.002	37	−0.21
	(2,4,10,2,4,10,2,4,10)	(12,11,10,12,11,10,12,11,10)	2	1.998	1.999	0.001	11	−0.03
			4	3.974	3.973	0.002	18	0.03
			8	7.969	7.759	0.002	35	2.71
			16	13.45[a]	13.36	0.003	68	0.71
Merge	(4,4,2)	(11,11,12)	2	2.000	2.000	0.002	1.9	−0.01
			4	4.000	4.000	0.002	4.2	0.00
			8	8.000	7.987	0.003	8.2	0.16
			16	15.92	15.472	0.003	17	2.90
	(10,10,4,4,2)	(10,10,11,11,12)	2	2.000	2.000	0.002	3.0	−0.01
			4	4.000	3.999	0.002	5.8	0.03
			8	8.000	7.998	0.002	12	0.03
			16	15.93	16.00	0.005	29	−0.44
	(2,2,4,4,10,10,4,4,2)	(12,12,11,11,10,10,11,11,12)	2	2.000	1.994	0.001	5.1	0.33
			4	3.996	3.952	0.002	10	1.12
			8	7.947	7.652	0.003	20	3.86
			16	15.35	14.12	0.002	41	8.73
Split	(2,4,4)	(12,11,11)	2	1.998	1.973	0.001	2.1	1.27
			4	3.974	3.839	0.002	2.5	3.51
			8	7.698	7.057	0.003	4.7	9.08
			16	13.51	11.59	0.003	8.1	16.6
	(2,4,4,10,10)	(12,11,11,10,10)	2	1.998	1.977	0.002	1.8	1.09
			4	3.974	3.980	0.002	6.1	−0.15
			8	7.698	7.058	0.002	6.7	9.07
			16	13.51	11.59	0.003	11	16.62
	(2,4,4,10,10,4,4,2,2)	(12,11,11,10,10,11,11,12,12)	2	1.998	1.976	0.001	3.0	1.12
			4	3.974	3.839	0.002	5.8	3.51
			8	7.698	7.654	0.002	22	0.57
			16	13.51	12.88	0.030	38	4.89

[a] Earmarked results presented in Fig. 11.

number of servers, for a given arrival rate, the higher the blocking (and lower the throughput).

In order to evaluate the solution quality of the GEM, we set up simulation experiments for 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 (for details on how to select a warm-up period, see Robinson [33]). The simulation was carried out using ARENA [34]. Table 3 presents 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 δ 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, $\Delta\%\theta \overset{\text{def}}{=} ((\theta\text{-}\theta^s)/\theta^s) \times 100\%$.

We see that the analytical results are very reasonable and acceptable (see column $\Delta\%\theta$) although they are not always as accurate as desirable. For instance, under extreme high utilization rates, that is, heavy traffic and quite few servers, the error may be as high as $\approx 20\%$. As we seek optimal server allocation, such a high utilization rate is not expected. Thus based on the simulation output, we conclude that the analytical results are reliable, for our optimization purposes.

Table 5
Simulation results for mixed topology.

λ	**c**	Analytical	Simulation			$\Delta\%\theta$
		θ	θ^s	δ	CPU (min)	
2.0	(2,2,2,2)	1.9995	1.991	0.002	1.6	0.42%
	(4,4,4,4)	2.0000	2.001	0.002	1.6	−0.03%
	(10,10,10,10)	2.0000	2.000	0.002	1.6	0.03%
4.0	(2,2,2,2)	3.9934	3.933	0.002	3.1	1.53%
	(4,4,4,4)	3.9999	4.000	0.002	3.1	0.00%
	(10,10,10,10)	4.0000	4.000	0.002	3.1	0.01%
8.0	(2,2,2,2)	7.9119	7.541	0.003	6.0	4.92%
	(4,4,4,4)	7.9994	7.992	0.003	6.2	0.10%
	(10,10,10,10)	8.0000	7.999	0.003	6.3	0.01%
16.0	(2,2,2,2)	14.966	13.237	0.003	11	13.1%
	(4,4,4,4)	15.975	15.871	0.003	12	0.66%
	(10,10,10,10)	16.000	15.998	0.004	13	0.01%

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 asymmetrical experiments unbalancing the routing probabilities, for the split topologies (routing probabilities 0.4–0.6, in the splitting nodes), and the arrival rates, for the merge topologies (external arrivals 0.4λ and 0.6λ, in the front nodes), and assuming different service rates, μ's, and different number of servers, c's, along the networks. The analytical results are shown in Table 4, along with simulation results. Based on the asymmetrical results, we can see that the main conclusions

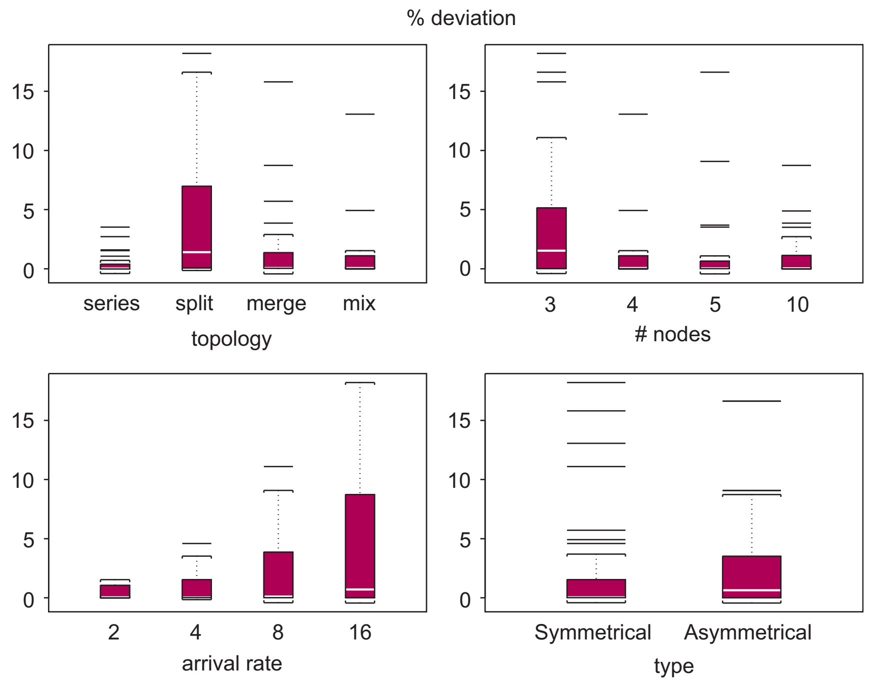

Fig. 8. Boxplots of the errors observed.

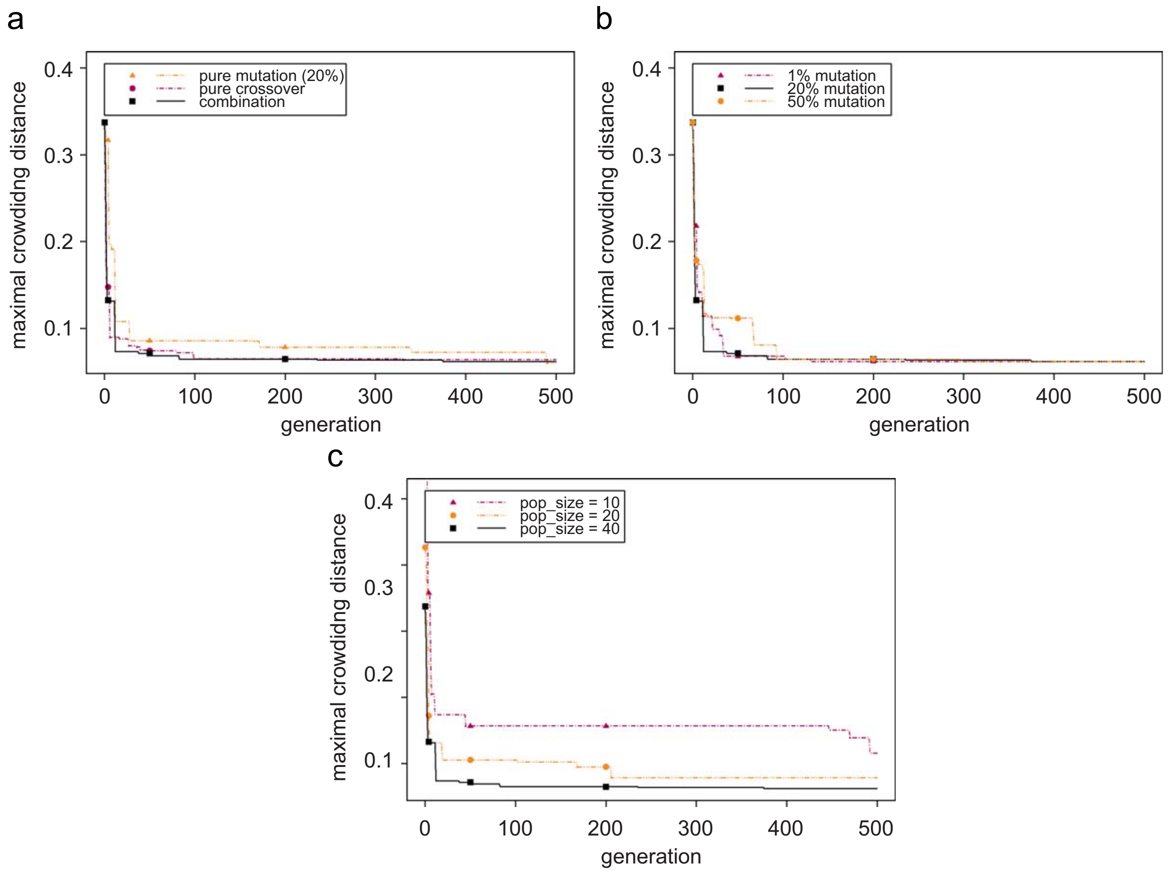

Fig. 9. Performance assessment of GA algorithm: (a) mutation and crossover iteration, (b) mutation effect and (c) population size effect.

hold. As observed for the symmetrical results, the errors may be quite high under high utilization. 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.

5.1.3. *Yet another network structure*

As a final demonstration of the methodology we evaluate the network structure depicted in Fig. 7. Interestingly, this is a network with different paths which are linked to each other (e.g., 1-4; 2-4). This is distinct from the previous topologies, where we had articulation vertices in all instances. The results may be seen in Table 5.

5.1.4. *A final word on the GEM accuracy*

Fig. 8 shows a number of boxplots of the % deviation, $\Delta\%\theta$, as function of the chosen topology, the number of nodes, the different arrival rates, and type of network (series, split, merge, or the mixed type).

Concerning the topologies, series presented the lowest errors and the lowest variabilities, while splits give the highest errors and variabilities. The number of nodes in the network and their type, symmetrical or asymmetrical, do not seem to have a significant effect in the % deviations and their variabilities. Clearly the arrival rate plays a important role in the % deviations and variabilities, mainly because of the increase of the congestion in the network with the increase of the arrival rate. The GEM is well-known for its inaccuracy in highly congested queueing networks.

5.2. *Network optimization*

In this section, we identify the optimal server configuration, changing the network structure, arrival rate, and processing rates, similarly to that of the previous sections. We consider both symmetrical and 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, as explained previously.

After extensive experimentation, we came to some interesting conclusions concerning the GA. We observe in Figs. 9(a)–(c) how evolves the maximal crowding distance from the Pareto set approximation, along with the number of generations, for the 9-node symmetrical series network earmarked in Table 2. The crowding distance is a measure of the spread of the non-dominated solutions (more details, on Deb et al. [32]) and the stabilization of the maximal crowding distance may give us some indication of convergence [35]. From Fig. 9(a) we clearly see that a combined scheme including mutation and crossover operators is effective for speeding up the convergence. In Fig. 9(b) we see that, unfortunately, an arbitrarily high mutation probability cannot guarantee a fast convergence. For the problems tested, the mutation rate was fixed to 20%, as we believe that this is the value around which the maximal convergence speed will be. Finally, observing Fig. 9(c), we conclude that the population size does not change dramatically the convergence speed beyond a certain point. Thus, in the light of the computational experiments, we use here a population of 40 individuals, for the small nets (up to nine nodes), and 80, for the large and complex network. The number of generations is set to 500, for the three- and five-node configurations, and 1,000, for the nine-node and the large complex networks.

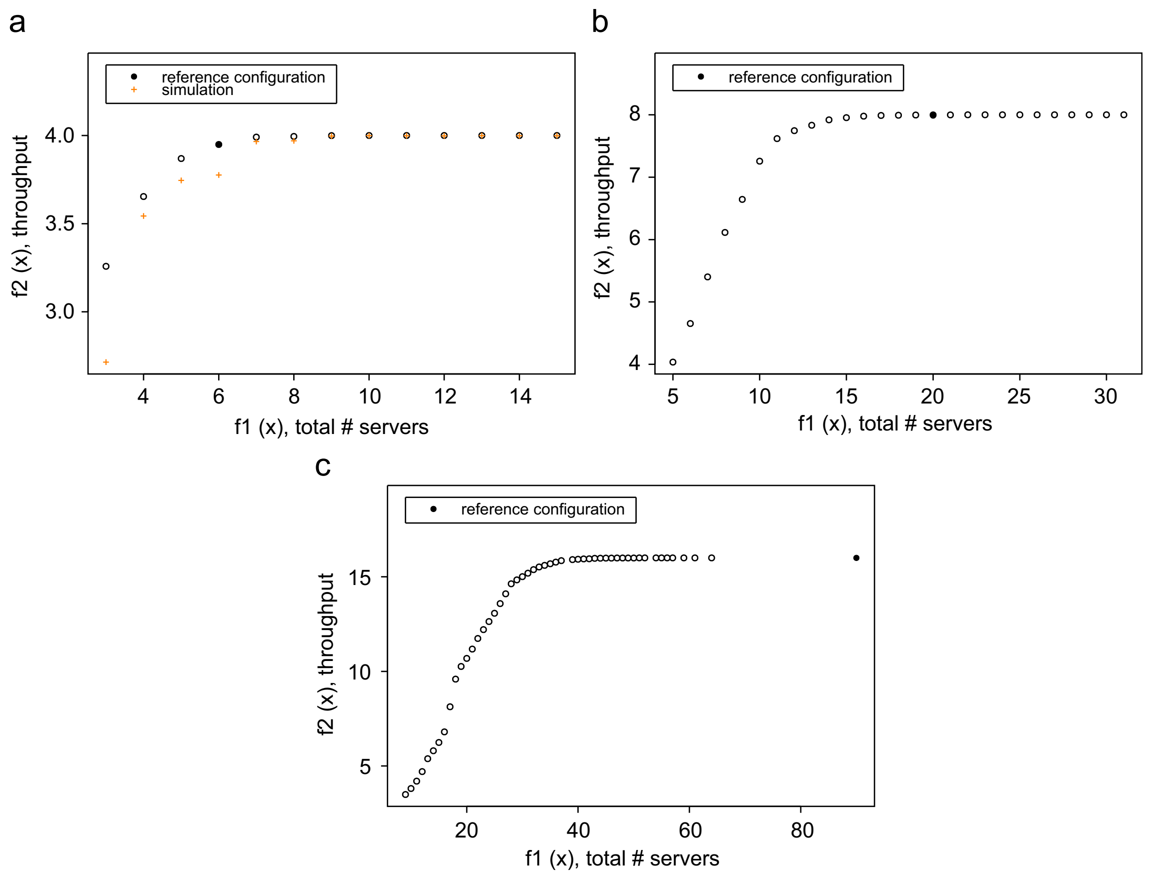

Fig. 10. GA optimization for some symmetrical settings: (a) 3-node series, $\lambda = 4$, (b) 5-node series, $\lambda = 8$ and (c) 9-node series, $\lambda = 16$.

The resulting Pareto-optimal sets for some series symmetrical settings are provided in Fig. 10. The other topologies presented similar results (not shown). It is worthwhile noticing that when applying the GA to find the Pareto sets, some of the cases included heavy traffic situations (for instance, in Fig. 10(a), for a total of 3 servers, the throughput is as low as 3.258, while the arrival rate equals 4.0). Since the GEM notably deteriorates its performance under heavy traffic, as noticed in the last paragraph of Section 5.1, we have also provided some comparisons with simulated results, seen in Fig. 10(a), in order to attest for the quality and robustness of the Pareto sets. Based on the simulations, we run for 200,000 time units and replicated 20 times, we found the half-width of the 95% confidence intervals too small (≈ 0.002) to be noticeable in the graphs. Comparing analytical and simulated results, we notice a somewhat larger disagreement between these two values for heavier traffic in the network (i.e., the total number of servers reduces). In conclusion, the methodology may not find the best solutions under heavy traffic because the actual throughput is not so accurately estimated by the GEM in these cases. When the target throughput is considerably lower than the arrival rate, i.e., the system operates under heavy traffic, a lower number of servers than what is actually necessary is indicated by the algorithm to achieve a pre-specified throughput.

Figs. 10(a)–(c) show that the throughput of all the *symmetrical* setting (identified in the graphs as *reference configuration*) lie on the Pareto-optimal set. 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 graph in Fig. 10(c) shows 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.

Fig. 11 shows the GA optimization results for some series *asymmetrical* settings (again, similar results, not shown, were found for the other topologies). We also presented comparisons with simulations, seen in Fig. 11(b), because some heavy traffic situations are present. In fact, as seen in Fig. 11(b), for a total number of servers of 5, for instance, the approximate throughput given from the GEM is only 4.984, for an external arrival rate of 16. Confirming what we have observed earlier, the GEM tends to estimate the throughput less accurately when the network is under such heavy traffic situations and the approximate Pareto set may not be as accurate for such cases.

Analyzing Fig. 11 in more detail, we observe that the throughput from all settings is below the Pareto-optimal set, which means that our settings for the network structure, arrival rates, and processing rates were not optimal, as previously for the symmetrical networks. More specifically, for the same total number of servers in the network, there exists another server allocation that results in a better throughput. For example, in the case of asymmetrical serial topology with 3 nodes, Fig. 11(a), processing rates of $(12, 11, 10)$ for nodes 1, 2, and 3 respectively, and arrival rate $\lambda = 16$, the GA found a much better server allocation, $(7, 4, 5)$, with the same total allocation as $(2, 4, 10)$. In other words, configuration $(7, 4, 5)$ results in a throughput of 15.997 as opposed to a throughput of only 13.50, when the server

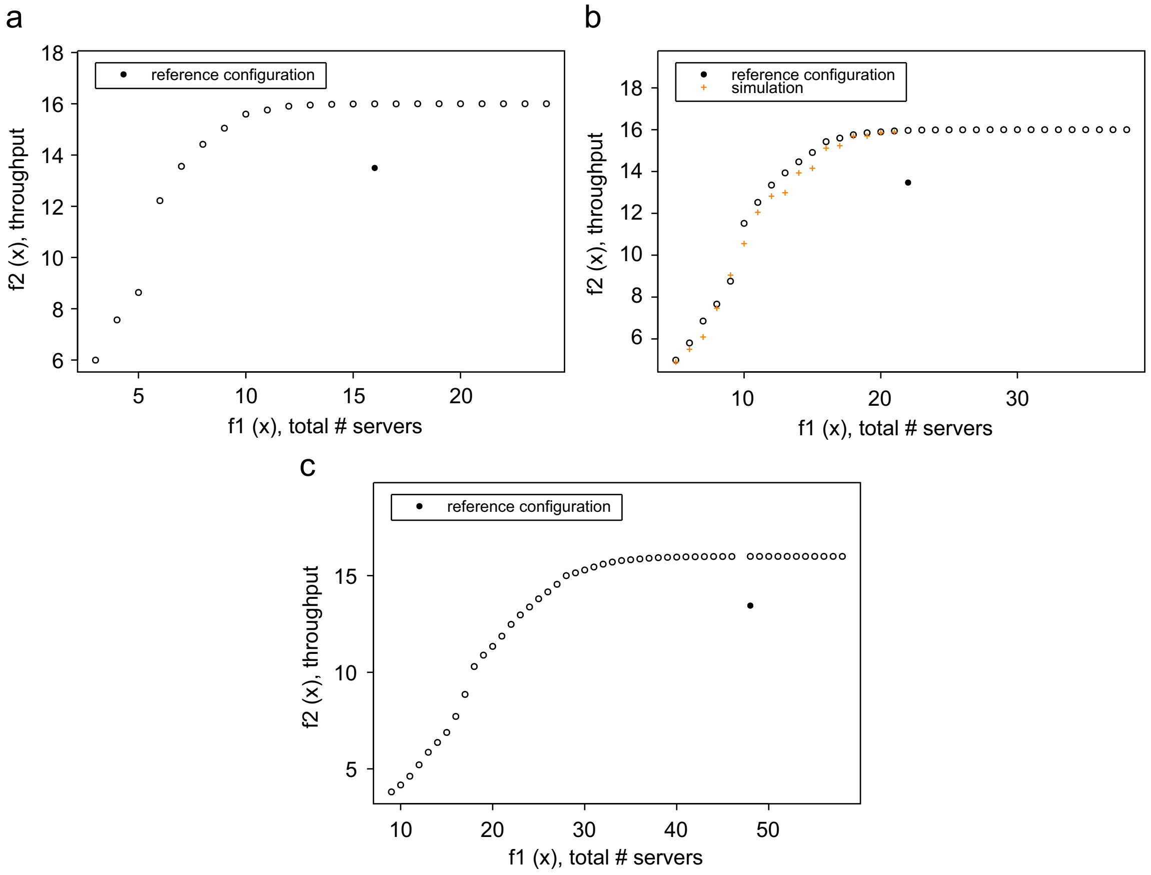

Fig. 11. GA optimization for some asymmetrical settings: (a) 3-node series, $\lambda = 16$, (b) 5-node series, $\lambda = 16$ and (c) 9-node series, $\lambda = 16$.

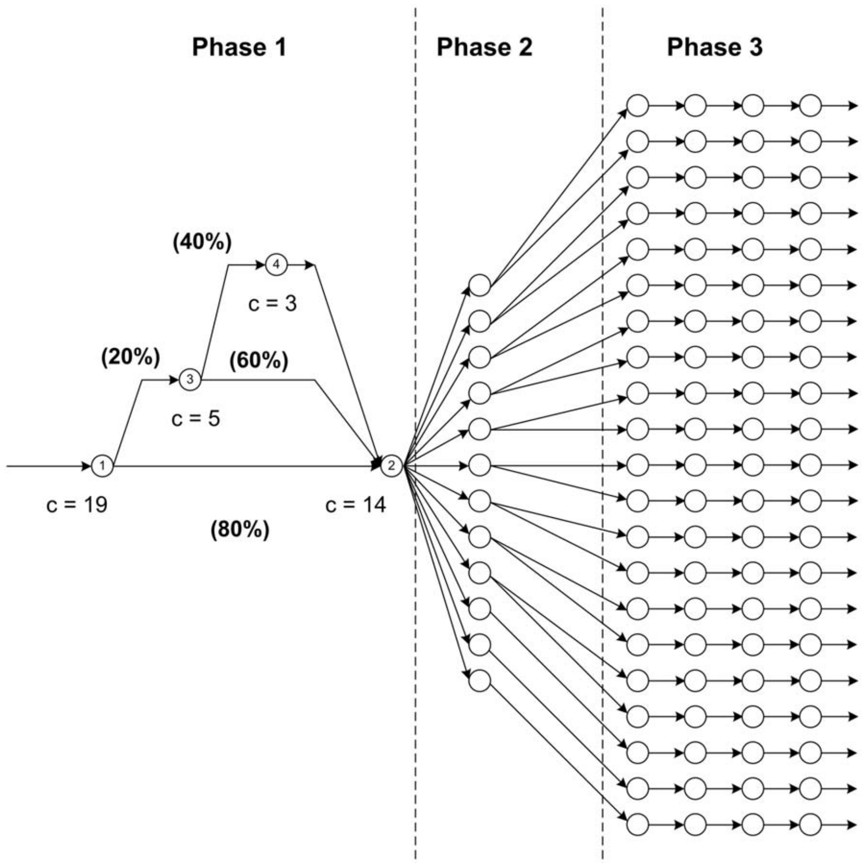

Fig. 12. Queueing network structure.

Table 6
Processing rates used in the complex topology.

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	14	3.400	20.00
Phase 1 server 3	5	3.400	11.05
Phase 1 server 4	3	3.400	11.05
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

allocation was (2, 4, 10), as seen in Table 4. 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).

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. A detailed description can be found in Fey [3].

Table 7
Results for the large and complex topology.

	λ	Analytical	Simulation			$\Delta\%\theta$
		θ	θ^s	δ	CPU (min)	
Slow servers	30.215	17.939	18.464	0.02	120	−2.8%
Fast servers	30.215	29.909	30.213	0.006	70.1	−1.0%

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 Fig. 12 is analyzed.

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

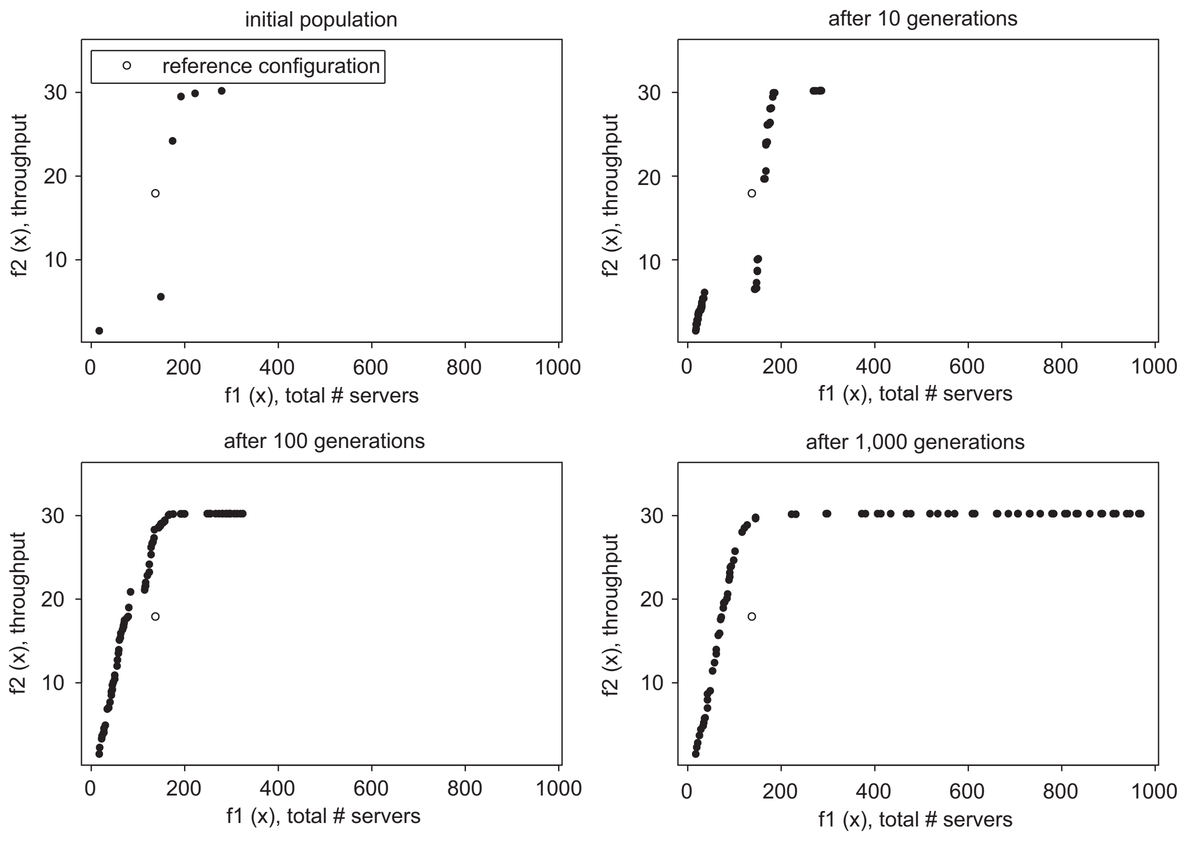

Fig. 13. GA optimization for the complex topology. (a) initial population. (b) after 10 generations. (c) after 100 generations. (d) after 1000 generations.

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:

1. The effect of different arrival rates to the throughput of the system;
2. The effect of different service rates of the servers to the throughput of the system;
3. 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.

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.

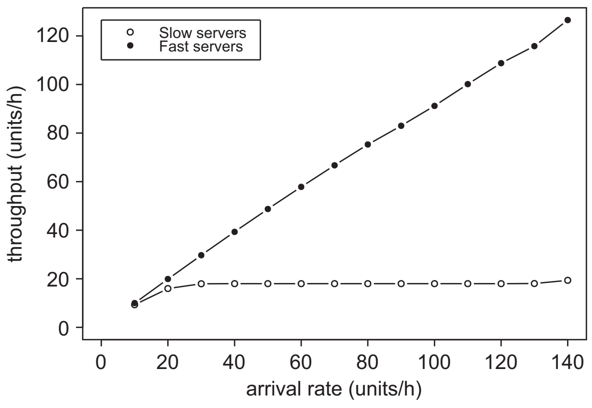

Fig. 14. Plot of throughput for different arrival rates.

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. Fig. 13 gives the results for the Pareto set as the population evolves. Clearly, the Pareto-set converges for a reasonable number of generations (from 100). The white dots in Fig. 13 corresponds to the total server allocation, 137, and corresponding throughput, 17.939, for the slow-server case (see in Tables 6 and 7). Fig. 13 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 were obtained for the fast server case.

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

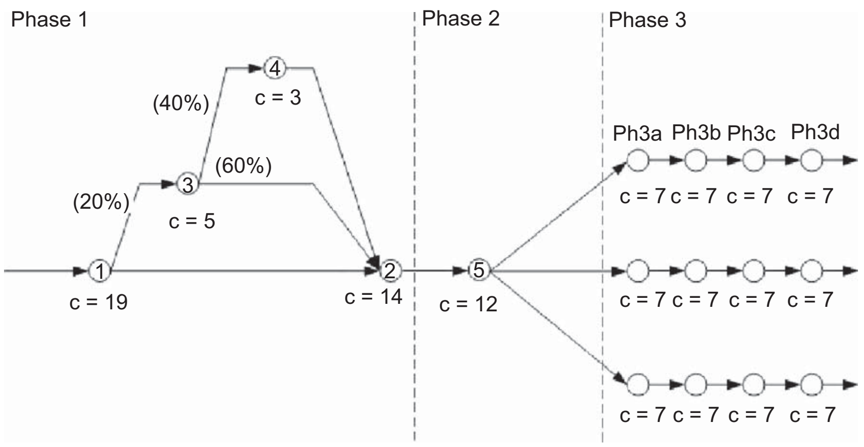

Fig. 15. New queueing network structure.

Table 8
Throughput rates for different configurations.

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	**17.939**	**21.191**	**24.030**
Improvement	-	**18%**	**34%**

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.

5.3.1. *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 Fig. 14, 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 Fig. 14. 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.

5.3.2. *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.

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 17.939 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 18%. 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 34% 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.

5.3.3. *Different network structures*

We now consider the case where it is possible to manipulate the structure of the queueing network in order to increase its throughput. 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* [36]. 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 21 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 Fig. 15. Note that for the three-line structure, we will use the data of configuration #1 in Table 8.

Using the same total number of servers as in configuration #1, we obtain a throughput of 23.613 units/h, higher than before for this three-line structure, as compared to the throughput of 17.939 units/h for the 21-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 demonstrated the use of the generalized expansion method (GEM) to both evaluate and optimize the performance of a zero-buffer queueing network. Concerning the performance evaluation algorithm employed, the GEM, we have shown that it typically delivers results within 5% of error, for basic series, merge, split, and mixed topologies, both symmetrical and asymmetrical. The maximum error observed may be higher, around $\approx 20\%$, for configurations under very heavy traffic. These are new results, since the GEM has never been used to evaluate $M/M/c/c$ queueing networks neither has its efficacy assessed as thoroughly as done here.

The proposed optimization methodology was successful to derive Pareto sets which, among other things, showed that although some optimal configurations could be correctly 'guessed', mainly for symmetrical networks, this is not always the case, as seen for asymmetrical networks. GAs are a feasible alternative to optimize real life multi-objective zero-buffer queueing networks, as shown in our example with a large and complex topology. We considered several simple topologies of queueing networks, taking into account both symmetrical and asymmetrical arrival and service rates. The insights from the multi-objective optimization of zero-buffer queueing networks were proved to be invaluable, particularly in the design phase of such systems.

We also provided a realistic 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 moderate blocking probability settings, the GEM provides a relatively simple mean to see how changes in arrival rate or processing rate affects the performance of a zero buffer queueing networks. All in all, we found the GEM as a fast (runs typically in a split second) and accurate approximation method to measure the throughput rate of zero-buffer queueing networks as well as for optimization purposes of such networks.

Topics for future research include the analysis and optimization of networks with cycles, e.g., to model many important industrial systems that have loops, such as systems with reverse streams of products due to re-work, or even the extension to networks of $GI/G/c/c$ queues, i.e., including generally distributed and independent arrivals [37,38]. Concerning the optimization tool, efforts could be made in order to speed up its convergence by mean of even more specialized crossover operators. In another direction, successful results were reported in the past using Powell coupled with the GEM to buffer allocation in certain types of queueing networks [25]. Whether or not a single-objective tool as Powell could be successfully embedded in a multi-objective methodology, possibly drawing inspiration from Hughes [39], is another interesting topic for future research in the area.

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

[1] Fransoo JC, Rutten WGMM. A typology of production control situations in process industries. International Journal of Operations and Production Management 1994;14(12):47–57.
[2] Hall NG, Sriskandarajah C. A survey of machine scheduling problems with blocking and no-wait in process. Operations Research 1996;44:510–25.
[3] Fey J. Design of a fruit juice blending and packaging plant. PhD thesis, Eindhoven University of Technology; 2000.
[4] Ramudhin A, Ratliff HD. Generating daily production schedules in process industries. IIE Transactions 1995;27:646–56.
[5] Tsybakov B. Optimum discarding in a bufferless system. Queueing Systems 2002;41:165–97.
[6] Buzacott J, Shanthikumar JG. Stochastic models of manufacturing systems. Englewood Cliffs, NJ: Prentice-Hall; 1993.
[7] Perros HG. Queueing networks with blocking. Oxford: Oxford University Press; 1994.
[8] Jain S, Smith JM. Open finite queueing networks with $M/M/C/K$ parallel servers. Computers & Operations Research 1994;21(3):297–317.
[9] Kerbache L, Smith JM. Asymptotic behavior of the expansion method for open finite queueing networks. Computers & Operations Research 1988;15(2):157–69.
[10] Walrand J. An introduction to queueing networks. Englewood Cliffs, NJ: Prentice-Hall; 1988.
[11] Dallery Y, Gershwin SB. Manufacturing flow line systems: a review of models and analytical results. Queueing Systems 1992;12:3–94.
[12] Hildebrand D. On the capacity of tandem server, finite queue, service systems. Operations Research 1968;16:72–82.
[13] Hillier F, Boling R. Finite queues in series with exponential or Erlang service times: a numerical approach. Operations Research 1967;16:286–303.
[14] Muth E, Alkaff A. The throughput rate of three-station production lines: a unifying solution. International Journal of Production Research 1987;25:1405–13.
[15] Rao N. A generalization of the bowl phenomenon in series production systems. International Journal of Production Research 1976;14(4):437–43.
[16] Rao N. A viable alternative to the method of stages solution of series production systems with Erlang service times. International Journal of Production Research 1976;14(6):699–702.

[17] Cruz FRB, Duarte AR, van Woensel T. Buffer allocation in general single-server queueing network. Computers & Operations Research 2008;35(11):3581–98.
[18] Cheah J, Smith JM. Generalized $M/G/C/C$ state dependent queueing models and pedestrian traffic flows. Queueing Systems 1994;15:365–86.
[19] Jain R, Smith JM. Modeling vehicular traffic flow using $M/G/C/C$ state dependent queueing models. Transportation Science 1997;31(4):324–36.
[20] Cruz FRB, Smith JM, Queiroz DC. Service and capacity allocation in $M/G/C/C$ state dependent queueing networks. Computers & Operations Research 2005;32(6):1545–63.
[21] Cruz FRB, Smith JM. Approximate analysis of $M/G/c/c$ state-dependent queueing networks. Computers & Operations Research 2007;34(8): 2332–44.
[22] Kerbache L, Smith JM. The generalized expansion method for open finite queueing networks. European Journal of Operational Research 1987;32: 448–61.
[23] Kerbache L, Smith JM. Multi-objective routing within large scale facilities using open finite queueing networks. European Journal of Operational Research 2000;121:105–23.
[24] Spinellis D, Papodopoulos C, Smith JM. Large production line optimisation using simulated annealing. International Journal of Production Research 2000;38(3):509–41.
[25] Smith JM, Cruz FRB. The buffer allocation problem for general finite buffer queueing networks. IIE Transactions 2005;37(4):343–65.
[26] Labetoulle J, Pujolle G. Isolation method in a network of queues. IEEE Transactions on Software Engineering 1980;SE-6(4):373–81.
[27] Kleinrock L. Queueing systems. Theory, vol. I. New York: Wiley; 1975.
[28] Adelman D. A simple algebraic approximation to the Erlang loss system. Operations Research Letters 2008;36(4):484–91.
[29] Gross D, Harris CM. Fundamentals of queueing theory, third ed. New York: Wiley Series in Probability and Statistics; 1998.
[30] Purshouse RC, Fleming PJ. On the evolutionary optimization of many conflicting objectives. IEEE Transactions on Evolutionary Computation 2007;11(6):770–84.
[31] Mathieu R, Pittard L, Anandalingam G. Genetic algorithms based approach to bi-level linear programming. Recherche Opérationnelle/Operations Research 1994;28(1):1–21.
[32] Deb K, Agrawal S, Pratap A, Meyarivan T. A fast elitist non-dominated sorting genetic algorithm for multi-objective optimization: NSGA-II. IEEE Transactions on Evolutionary Computation 2002;6:182–97.
[33] Robinson S. A statistical process control approach to selecting a warm-up period for a discrete-event simulation. European Journal of Operational Research 2007;176(1):332–46.
[34] Kelton D, Sadowski RP, Sadowski DA. Simulation with arena. New York: McGraw Hill College Div.; 2001.
[35] Rudenko O, Schoenauer M. A steady performance stopping criterion for Pareto-based evolutionary algorithms. In: Proceedings of the 6th international multi-objective programming and goal programming conference, Hammamet, Tunisia; 2004.
[36] Hopp WJ, Spearman ML. Factory physics. New York: McGraw Hill International Editions; 2000.
[37] Choi DW, Kim NK, Chae KC. A two-moment approximation for the $GI/G/c$ queue with finite capacity. INFORMS Journal on Computing 2005;17(1): 75–81.
[38] Kim NK, Chae KC. Transform-free analysis of the $GI/G/1/K$ queue through the decomposed Little's formula. Computers & Operations Research 2003;30(3): 353–65.
[39] Hughes EJ. Multiple single objective Pareto sampling. In: I. Press, editor. Congress on evolutionary computation CEC'03. Piscataway, NJ; 2003. p. 2678–84. doi: 10.1109/CEC.2003.1299427.