Buffer and throughput trade-offs in $M/G/1/K$ queueing networks: A bi-criteria approach

F. R. B. Cruz*† frc@cs.nott.ac.uk
T. Van Woensel‡§ t.v.Woensel@tue.nl
J. MacGregor Smith¶ jmsmith@ecs.umass.edu

February 28, 2010

Abstract — **The optimal design of real-life systems modeled as finite queueing networks is a difficult stochastic optimization problem. Besides being nonlinear and including integer variables, different objectives often conflict with each other. For example, one conflicting pair of objectives includes first minimizing the overall number of buffers and then maximizing throughput. In this paper, we present an original methodology to solve a buffer allocation and throughput trade-off problem in single server general queueing networks. An approximation of the complete set of all best solutions, known as the Pareto optimal or non-inferior set, is derived by a special version of a multi-objective genetic algorithm (MOGA). The applied MOGA proves to be suitable for the stochastic trade-off problem. A comprehensive set of computational results attest to the efficiency and efficacy of the proposed methodology. We were able to show from a medium-sized mixed network that the squared coefficient of service time variation plays an important role in buffer allocation, which indicates the importance of using a general service model. Moreover, after the analysis of the solutions in the decision variable space, we confirm that the buffer allocation is highly dependent on the target throughput, which sometimes can be sacrificed in favor of using fewer of the usually expensive buffer spaces.**

Keywords — Buffer allocation; queueing networks; optimization; genetic algorithms.

1 Introduction

Finite queues configured in networks, as seen in Fig. 1, arise in many real-life contexts and are not limited to manufacturing systems (Tan and Gershwin, 2009). They extend to other areas of interest, including service production, such as health care activities (Osorio and Bierlaire, 2009) or call centers (Jouini et al., 2009). The continued growth in the importance and complexity of this area requires more accurate models. The problem tackled in this paper focuses on one of the vital topics in production system design: the trade-off between buffer allocation (Faria et al., 2006; Nahas et al., 2006) and the corresponding level of quality for a given finite single-server queueing network. The performance measure considered to assess the level of quality is the throughput, Θ, or the average number of users served in a certain amount of time.

The throughput is one of the most popular performance measures, but other measures could be considered as well, including sojourn time, work-in-process, percentage utilization of resources and so on. There is a crucial trade-off between the overall amount of buffer space and its resulting throughput. Because buffer space can be expensive, it is desirable to minimize the total amount of buffers used. Another motivation for minimizing the number of buffers is to minimize the work-in-process, WIP, and thus the delay. On the other hand, one would also like to increase the network throughput. The throughput is directly related to the number of buffers allocated. More buffers will lead to a throughput that is generally higher than before; the monotonicity and concavity of the throughput with respect to buffer size (for instance, see Shi and Gershwin, 2009, and references therein) is well known and is an assumption supported by extensive numerical experiments. One might argue that in many applications the throughput is usually equal to the sum of the external arrivals with very few rejections because the probability of a full queue is usually small. However, this is not true when buffers are costly because this could lead to

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

sacrificing some of the throughput in favor of using less expensive buffer spaces.

Fig. 2 shows a typical surface for the throughput as a function of the total buffer size K and the external arrival rate Λ in a single $M/G/1/K$ queue (service rate is $\mu = 10$) which, in Kendall notation, stands for Markovian arrivals, generally distributed service times (with a known service rate μ and squared coefficient of variation s^2), a single server, and a total capacity of K items included that in service. For complex networks, such as $M/G/1/K$ queues, a similar behavior is observed in Fig. 2. The focus in this paper is on buffer allocation and the throughput trade-off problem in queueing networks, where each node follows an $M/G/1/K$ configuration in any arbitrary acyclic series-parallel topology, *e.g.*, Fig. 1.

This paper is more than a collection of results from various sources because there are many novel developments presented. We outline a number of these contributions here:

1. The buffer allocation problem is addressed with a multi-objective approach. To the best of the authors' knowledge, this has not been done before because the problem is exclusively tackled in a single-objective fashion. Rather than multiplying individual objectives by different weights and obtaining a standard maximization problem with a single objective function, obtaining Pareto optimal solutions is far less arbitrary. If the decision maker is not able to assign priorities to the individual objectives, a multi-objective algorithm allows them to choose among a large number of different trade-off solutions. That is, for those areas where one cannot merely consider the cost as a bottom-line in a single objective (say, for instance, in the emerging area of health care and similar system queue modeling, Osorio and Bierlaire, 2009), a decision maker should make this choice with the most complete information of their options (for more discussion on multi-objective decision making, the interested reader is referred to Chankong and Haimes, 1983).

2. Both the decision variables, namely the buffer space and the throughput, are optimized simultaneously by means of an approximate performance evaluation algorithm, the Generalized Expansion Method (GEM), and a problem-specific multi-objective genetic algorithm (MOGA). We show that a cross-over and mutation scheme, specifically designed for the problem, which is associated with a peeling-off scheme to select the Pareto set members (using the well-known fast non-dominating sorting), will make it possible to solve the problem efficiently. We demonstrate that solutions are obtained within a reasonable amount of computational effort and that they are accurate. Indeed, the approximate Pareto sets are proved to contain optimal solutions previously obtained for the pure buffer allocation problem.

3. Finally, some interesting insights emerge from the topologies studied. First, we were able to show that a simple and well-known methodology may be used to calibrate the parameters of the MOGA, which proved to be pretty effective. Second, we show that the MOGA performs well (properly converges) under a broad range of configurations (series, merges, splits, and mixed) when we control the convergence by means of the stopping criterion $\sigma_L < \delta_{\text{lim}}$, defined in Subsec. 3.1. This is relevant because one does not have to go into more intricate convergence measures and the algorithm performances are not expected to be significantly different for slightly different topologies. Third, using a medium-sized mixed network, we show that the squared coefficient of variation plays an important role in buffer allocation using the approximate Pareto curve, which indicates the importance of using general service models that are more accurate than purely Markovian models such as in the $M/M/1/K$ queues. Finally, after analyzing the solutions in the decision variable space, we confirm that the buffer allocation is highly dependent on the target throughput. Thus, we might want to sacrifice the throughput somewhat in favor of using fewer of the (usually expensive) buffer spaces. The approximate Pareto suggests that the MOGA provided may be helpful in making such a tough decision. These are crucial insights from a managerial perspective.

This paper is organized as follows. In Sec. 2, we present previous results in the area and formulate the buffer allocation and throughput trade-off problem as a multi-objective mathematical programming problem. A MOGA specially developed for such a trade-off problem is presented in detail in Sec. 3, along with the performance evaluation tool used to approximate the throughput. In Sec. 4, results from a comprehensive set of computational experiments are presented to show the efficiency of the approach and show some new insights. Finally, we summarize the paper in Sec. 5 with concluding remarks and topics for future research in the area.

2 The Buffer Allocation and Throughput Trade-off Problem

We first discuss the previous results from the literature and then give our multi-objective mathematical programming problem formulation. In general, we note that the throughput is not available in closed-form to relate it to the system parameters and other design variables, such as the network topology.

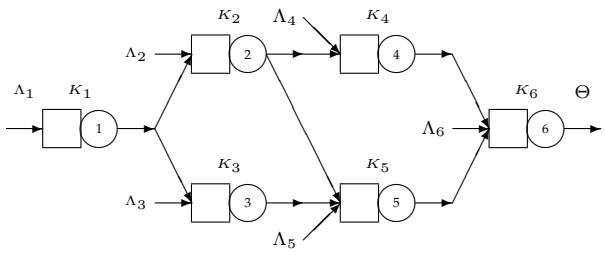


Figure 1: Queueing network in an arbitrary topology

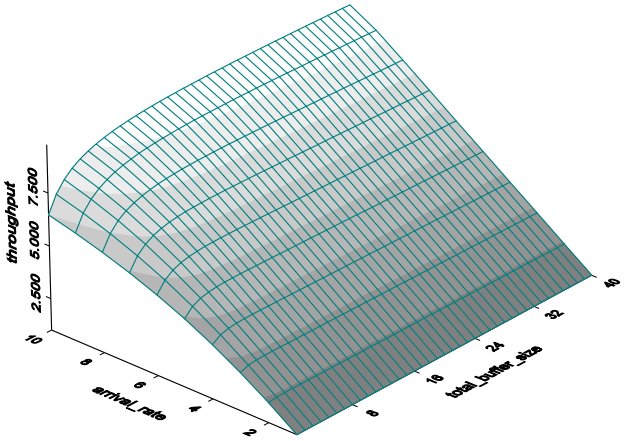


Figure 2: Throughput *versus* buffer size and arrival rate

2.1 *Classical Formulation*

From a modeling point of view, the buffer allocation and throughput trade-off problem can be defined as a buffer allocation problem (BAP) in which the total buffer costs are minimized subject to a minimum throughput threshold constraint. In fact, successful results were reported (Smith and Cruz, 2005; Cruz et al., 2008) when solving this problem with $M/G/1/K$ queues. In Smith and Cruz (2005) and Cruz et al. (2008), an integer mathematical programming model defined on a graph $G = (N, A)$, where N is the set of nodes and A is the set of directed arcs, is formulated as follows.
(BAP):

$$\min_{\mathbf{x}} \sum_{i \in N} c_i x_i, \tag{1}$$

subject to:

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

$$x_i \in \{1, 2, \ldots\} \; \forall \; i \in N, \tag{3}$$

which minimizes the total cost for buffer allocation to the network, where $\sum_{i \in N} c_i x_i$ are subject to a minimum total throughput $\Theta^{\min}$. In this formulation, $\Theta^{\min}$ is some threshold throughput, which is less than the total external arrival rate, $\Lambda = \sum_{i \in N} \Lambda_i$, where Λ_i is the external arrival rate to node i and x_i is the buffer allocation K_i to the ith $M/G/1/K_i$ queue, including those in service. It should be clear that the throughput is modeled as a constraint rather than as an objective in the above formulation. To solve the above problem, we relax constraint (2) with a Lagrangian relaxation technique (Lemaréchal, 2007) and solve the relaxed problem with a derivative-free search algorithm. Some factors, such as the threshold throughput $\Theta^{\min}$, must be determined beforehand. In particular, correctly setting this threshold throughput may not be a trivial task (Cruz et al., 2008; Smith and Cruz, 2005).

For a deeper discussion on this and other alternative formulations and algorithms for buffer allocation problems in queueing networks, the interested reader is referred to Smith and Cruz (2005) and Gershwin and Schor (2000).

2.2 *Multi-objective Approach*

In the BAP formulation given by Eq. (1) and subject to (2)–(3), the buffer-throughput trade-off is not evident because it is difficult to define the weights $\boldsymbol{\omega}$ to compose a single-objective function over different objectives along with other parameters, such as the minimum throughput Θ. In these situations, multi-objective formulations and solution techniques widen the set of feasible and optimal solutions. In this paper, we aim to determine the whole Pareto optimal set, that is, the set of solutions for which there is no other better solution in at least one objective without being worse with respect to any other objective (for details, see Chankong and Haimes, 1983). This means that the decision maker will be able to better evaluate the effect of replacing one solution by another. The multi-objective approach allows the loss to be evaluated with one objective (*e.g.*, throughput) compared to a simultaneous enhancement in another objective (*e.g.*, buffer allocation).

This trade-off multi-objective buffer allocation problem (MOBAP) can be formulated as the following mathematical programming formulation.
(MOBAP):

$$\underset{\mathbf{x}}{\text{minimize}} \, F(\mathbf{x}) = \Big(f_1(\mathbf{x}), -f_2(\mathbf{x}) \Big)^{\mathrm{T}}, \tag{4}$$

subject to:

$$x_i \in \{1, 2, \ldots\}, \; \forall i, \tag{5}$$

where $f_1(\mathbf{x}) = \sum_i c_i x_i$ is the total cost for buffer allocation and the c_i's are assumed to be unitary for the rest of this paper, $f_2(\mathbf{x}) = \Theta(\mathbf{x})$ is the throughput, and the

integer decision variables $x_i \equiv K_i$ are the total capacity (including those in service) for the ith $M/G/1/K_i$ queue.

Note that this MOBAP formulation generalizes the single objective formulation, given Eq. (1) subject to Eq. (2) and Eq. (3). Indeed, rather than working with objectives $f_1(\mathbf{x})$ and $f_2(\mathbf{x})$ simultaneously, Smith and Cruz (2005) suggested algorithms to generate an approximation to the Pareto set of solutions for the two objectives starting from an initial hyperplane h whose inclination is defined by some given weight ω_i for each objective i (Fig. 3). That is, defining the objective function $z(\mathbf{x}) = \omega_1 f_1(\mathbf{x}) + \omega_2 f_2(\mathbf{x})$, Smith and Cruz's (2005) single-objective optimization algorithm found a sequence of parallel hyperplanes to h that converge to the support hyperplane h^* with only one intersection point y^* that belongs to the Pareto set Y^*.

3 Optimization algorithms

Recent successful applications of genetic algorithms (GA) were reported for manufacturing problems, for instance, a recent paper by Ramli et al. (2009). A wealth of references is provided by these authors. Ramli et al. (2009) reiterate that GA works both in linear and nonlinear analytical expressions and very complex statements can easily be included in the objective expression, which is a perfect fit for the MOBAP formulation we are addressing. Then, we use a multi-objective version of a genetic algorithm (MOGA) in combination with the generalized expansion method (GEM), which accurately approximates the performance measures of a queueing network (Kerbache and Smith, 1987, 1988). MOGA's are particularly suitable for the multi-objective problems because they have performed well with these types of problems (*e.g.*, Purshouse and Fleming, 2007, and many references therein). In the following subsection, we describe the algorithms in depth.

3.1 Multi-objective Genetic Algorithm

Multi-objective genetic algorithms (MOGA's) are suitable for the trade-off problem because of their well established efficiency for dealing with multi-objective problems in general ((Fonseca and Fleming, 1995; Coello, 2000). Genetic algorithms (GA's) are optimization algorithms that perform an approximate global search that relies on the information obtained from the evaluation of several points in the search space and obtaining a population of these points that converges to the optimum by applying genetic operators: *mutation*, *crossover*, *selection*, and *elitism*. Each one of these operators may be implemented in several different ways and each characterizes a specific instance of GA. Additionally, convergence of GA's is guaranteed by assigning fitness to each population member and simultaneously preserving diversity (for details on GA, see Goldberg, 1989).

The MOGA implementation developed in this paper is shown in Fig. 4, which is loosely adapted from the elitist non-dominated sorting genetic algorithm (NSGA-II) described by Deb et al. (2002). This implementation is described in detail in the following sections. First, the algorithm reads the problem setting, that is, the graph $G(N, A)$, where N is the set of nodes and A is the set of directed arcs, and the arrival rate λ_n and service rate μ_n, $\forall\ n \in N$. Then, for `numGen` generations, offspring are generated by crossover and mutation; parents and offspring are combined and selected by a non-dominated sorting scheme, to generate a better-fitting population. Many pieces of the multi-objective genetic algorithm presented here are based on the literature. Of course, many adaptations had to be made in the crossover and mutation schema, which is typical when using GA's. However, to the best of the authors' knowledge, this is the first time that the NSGA-II, the reflection operator, and the stopping criterion were used together in one algorithm, to produce a better-fitted algorithm, as we show in the computational results section.

Selection and Elitism Operators

In the special case of multi-objective optimization problems, the *selection* and *elitism* operators must be structured to correctly identify the best individuals. Elitism is based on the concept of dominance, which is illustrated in Fig. 5. Point $\mathbf{x}_i = (x_{i_1}, x_{i_2}, \ldots, x_{i_n})$ dominates point $\mathbf{x}_j = (x_{j_1}, x_{j_2}, \ldots, x_{j_n})$ if it is better in one objective $f_1(\mathbf{x}_i)$ (lower, if minimization) without being worse in any other objective $f_2(\mathbf{x}_j)$ (higher, if maximization). For instance, in Fig. 5, point V is dominated by point I, but not by point II, and point VI is dominated by points I, II and III. Points I through IV compose the best front and may be seen as an approximation for the Pareto set, which is the set of points not dominated by any other set.

The algorithm presented in Fig. 6 is a peeling-off scheme that separates the individuals of a certain solution into several layers, $\mathcal{F}_i$, of non-dominated points. It can be implemented to run in polynomial time, $\mathcal{O}(n \log n)$, with efficient sorting algorithms. Selection is performed by picking points from each non-dominated point sequentially, $\mathcal{F}_1, \mathcal{F}_2, \ldots$, up to the point that the required individuals for the next iteration is reached.

Of course, some criteria must be applied if, after adding a group of individuals from a certain level $\mathcal{F}_i$, the maximum number of individuals is exceeded. The algorithm presented in Fig. 7 describes how to compute a measure of diversity, the crowding distance, as defined by Deb et al. (2002). A graphical interpretation for the crowding distance for a generic point i is presented in Fig. 8. In the figure, the crowding distance of the ith solution is the average side length of the cuboid (shown as a box connecting the $(i-1)$th and $(i+1)$th solutions in the immediate neighborhood of the ith solution). To ensure the greatest diversity, only the points with the largest crowding distance are kept for the next

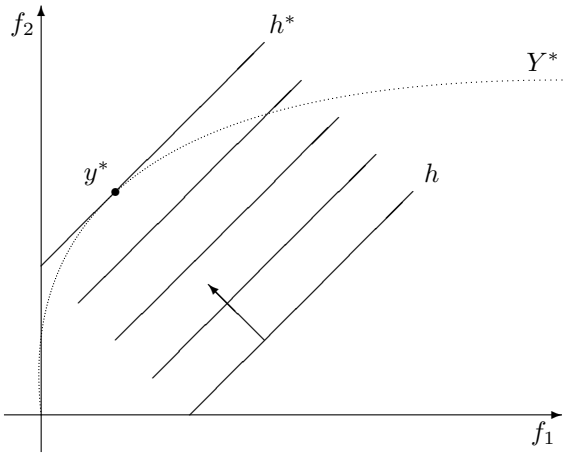


Figure 3: Pareto set Y^* and approximate solution y^*

algorithm
read graph, arrival, service rates, $G(N, A), \lambda_n, \mu_n, \ \forall\, n \in N$
$P_1 \leftarrow$ **GenerateInitialPopulation**(`popSize`)
for $i = 1$ **until** `numGen` **do**
/ generate offspring by crossover and mutation */*
$Q_i \leftarrow$ **MakeNewPop**(P_i)
/ combine parent and offspring */*
$R_t \leftarrow P_t \cup Q_t$
/ find non-dominated fronts* $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \dots)$ **/*
$\mathcal{F} \leftarrow$ **FastNonDominatedSort**(R_t)
/ find new population by means of*/*
/ the crowding-distance-assignment */*
$P_{t+1} \leftarrow$ **GenerateNewPopulation**(R_t)
end for
$P_{\texttt{numGen}+1} \leftarrow$ **ExtractParetoSet**$(P_{\texttt{numGen}})$
write $P_{\texttt{numGen}+1}$
end algorithm

Figure 4: Elitist non-dominated sorting genetic algorithm - NSGA-II

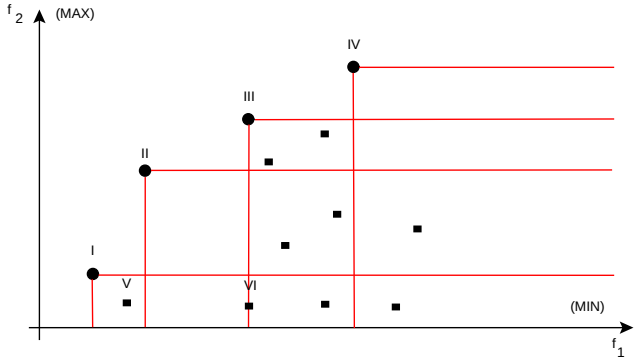


Figure 5: Dominated (■) and non-dominated (●) points

algorithm FastNonDominatedSort(P)
for each $p \in P$ **do**
$S_p \leftarrow \emptyset$
$n_p = 0$
for each $q \in P$ **do**
if p **dominates** q **then**
$S_p \leftarrow S_p \cup \{q\}$
else if q **dominates** p **then**
$n_p \leftarrow n_p + 1$
end for
if $n_p = 0$ **then**
$p_{\text{rank}} \leftarrow 1$
$\mathcal{F}_1 \leftarrow \mathcal{F}_1 \cup \{p\}$
end for
repeat for all other fronts, $\mathcal{F}_i, i = 2,$ `max`
end algorithm

Figure 6: Fast non-dominating sorting

algorithm Crowding-distance-assignment$(\mathcal{I})$
/ set number of solutions in set* $\mathcal{I}$ **/*
$l \leftarrow |\mathcal{I}|$
for $i = 1$ **until** l **do**
$\mathcal{I}_{\text{dist}_i} \leftarrow 0$
end for
for ***each objective*** m **do**
$\mathcal{I} \leftarrow \text{sort}(\mathcal{I}, m)$
$\mathcal{I}_{\text{dist}_1} \leftarrow \mathcal{I}_{\text{dist}_l} \leftarrow \infty$
for $i = 2$ **until** $l - 1$ **do**
$\mathcal{I}_{\text{dist}_i} \leftarrow \mathcal{I}_{\text{dist}_i} + \frac{(\mathcal{I}_{\text{dist}_{i+1}} - \mathcal{I}_{\text{dist}_i})}{(f_m^{\max} - f_m^{\min})}$
end for
end for
end algorithm

Figure 7: Crowding distance assignment

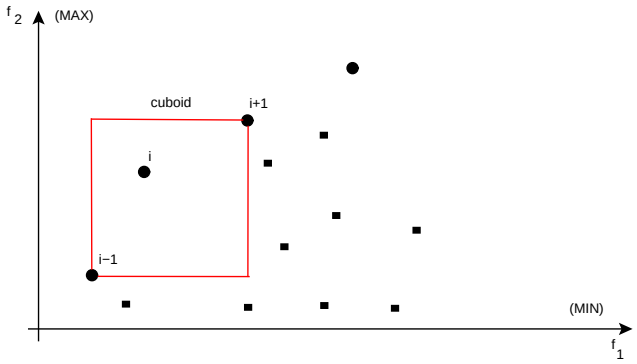


Figure 8: Crowding distance interpretation (Deb et al., 2002)

iterations.

Crossover and Mutation Operators

The *crossover* and *mutation* operators are somewhat independent of the multi-objective nature of the problem

but are highly dependent on the application. For the problem at hand, we implemented a crossover mechanism known as *uniform crossover* because of its efficiency in identifying, inheriting, and protecting common genes, as well as re-combining non-common genes (Sywerda, 1989; Hu and Di Paolo, 2007). Basically, crossover is performed for each variable in this scheme, with probability `rateCro`, according to a crossover operator. Fig. 9 illustrates the chromosome representation and the crossover used in this paper. Each gene $x_{i,(\bullet,t)}$ represents the buffer size at the ith queue of the selected parents to crossover at iteration t.

The crossover operator we choose to use here is known as *simulated binary crossover* (SBX) because of its convenience for real-coded GA's (Deb and Agrawal, 1995; Deb and Beyer, 1999), as we have here. The SBX operator simply simulates a binary crossover operator and all of its advantages, but without coding variables. In other words, with the SBX operator, the children solutions, $x_{i,(\bullet,t+1)}$, are calculated from the parents, $x_{i,(\bullet,t)}$, as follows:

$$x_{i,(1,t+1)} = 0.5\Big[(1+\beta_q)x_{i,(1,t)} + (1-\beta_q)x_{i,(2,t)}\Big],$$
$$x_{i,(2,t+1)} = 0.5\Big[(1-\beta_q)x_{i,(1,t)} + (1+\beta_q)x_{i,(2,t)}\Big],$$

where β_q is a random variable obtained from the following probability distribution

$$f(\beta) = \begin{cases} 0.5(\eta+1)\beta^{\eta}, & \text{if } \beta \leq 1; \\ 0.5(\eta+1)\frac{1}{\beta^{\eta+2}}, & \text{otherwise,} \end{cases} \quad (6)$$

that is specifically designed to create a child solution that has a similar search power as that in a single-point crossover in binary-coded GA's (Deb and Agrawal, 1995). For instance, a histogram of weights β_q randomly generated under the above probability distribution is presented in Fig. 10 with $\eta = 16$ to create children solutions from parent solutions, *e.g.*, $x_{i,(1,t)} = 5$ and $x_{i,(2,t)} = 10$. As η increases, the children will be closer to their parents.

Mutation occurs at a specific rate, `rateMut`, for each gene (the decision variables, K_i). As suggested by Deb and Agrawal (1995), a Gaussian perturbation, $\varepsilon_i \sim \mathcal{N}(0,1)$, is introduced in each K_i. The mutation scheme is presented in Fig. 11.

Note that constraint (5) may not hold any longer after crossover and mutation. To guarantee feasibility, the values are rounded and adjusted accordingly by means of the reflection operator

$$x_{i,r} = x_\ell + |x_i - x_\ell|,$$

where x_ℓ is the lower limit for buffer allocation (for the problem treated here, $x_\ell = 1$), x_i is the resulting buffer allocation after crossover and mutation and $x_{i,r}$ is the result after reflection, with $|x|$ being the absolute value of x. In other words, the proposed scheme is guaranteed to generate only feasible solutions without avoiding or privileging any particular solution.

Stopping Criterion

The stopping criterion for multi-objective optimization evolutionary algorithms is an issue that only recently has been addressed by the researchers (Rudenko and Schoenauer, 2004; Martí et al., 2007). Of course, the maximum number of generations, `numGen` (Fig. 4), plays an important role in the total computational effort but it may not be the best stopping criterion. As remarked by Rudenko and Schoenauer (2004), a much better stopping criterion is when a fixed number of iterations are performed without improvement to avoid wasting precious computation time. Rudenko and Schoenauer (2004) demonstrate that this stopping criterion is difficult to define using a comprehensive set of computational experiments. Alternatively, the authors propose a local stopping criterion that computes the stability measure of the spread of the non-dominated solutions after each iteration. Another criterion is a global stopping criterion proposed by Martí et al. (2007), which is a sophisticated method that views the population evolution as a dynamic system whose state may be estimated by a Kalman filter. For simplicity, we use the criterion by Rudenko and Schoenauer (2004), which is based on stabilizing the maximal crowding distance, d_l, measured over L generations by calculating the following standard deviation

$$\sigma_L \equiv \sqrt{\frac{1}{L}\sum_{l=1}^{L}(d_l - \bar{d}_L)^2}, \quad (7)$$

where $\bar{d}_L$ is the average of d_l over L generations and the MOGA stops when $\sigma_L < \delta_{\lim}$. The authors remark that σ_L should not depend on the actual values of the objective functions because all crowding distances are normalized. The authors suggest $L = 40$, and $\delta_{\lim} = 0.02$, which leads to the stopping criteria $\sigma_{40} \leq 0.02$, which will work pretty well for our problem.

In summary, MOGAs move the whole population toward the Pareto set, instead of a single point; thus, in a single run, the whole Pareto set, or a large portion of it, will be found. This demonstrates the superiority of MOGAs, in some sense, over deterministic algorithms, which are able to find only one Pareto point for each run (Fonseca and Fleming, 1995; Coello, 2000). The population size, `popSize`, and the mutation ratio, `rateMut`, among other parameters are critical to the convergence of MOGA, as we will show in our computational experiments. The whole NSGA-II process is illustrated in Fig. 12. The population P_t is merged with the offspring population Q_t generated by crossover and mutation and divided into layers $\mathcal{F}_i$ until a new population of at least `popSize` is reached. The most recent $\mathcal{F}_i$ layer is reduced by the smallest crowding distance criterion to produce a new population P_{t+1} of size `popSize` to be used for selection, crossover, and mutation to create a new population of same size and so on.

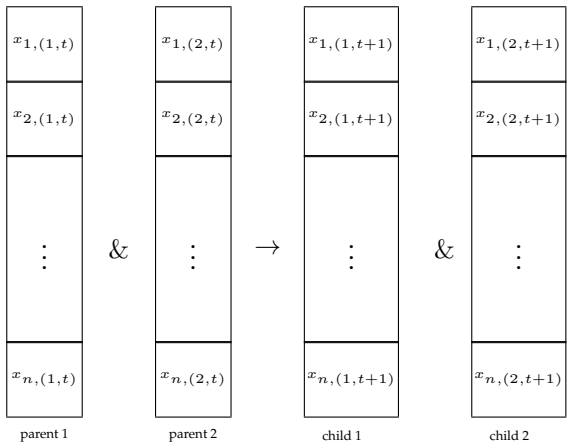


Figure 9: Chromosome representation and simulated binary crossover (SBX)

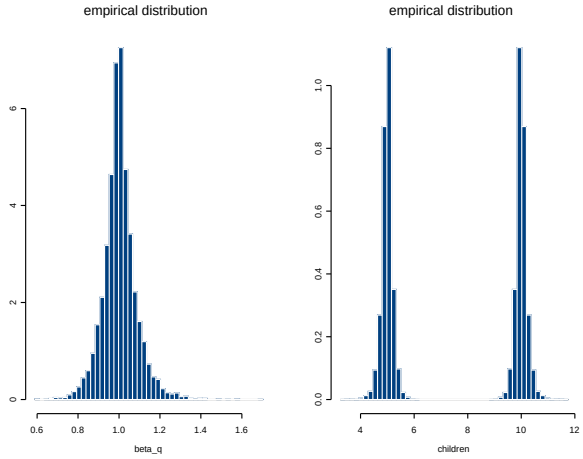


Figure 10: Distribution of weights β_q and children for SBX operator

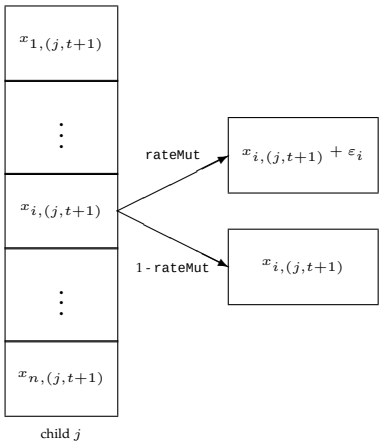


Figure 11: Mutation scheme

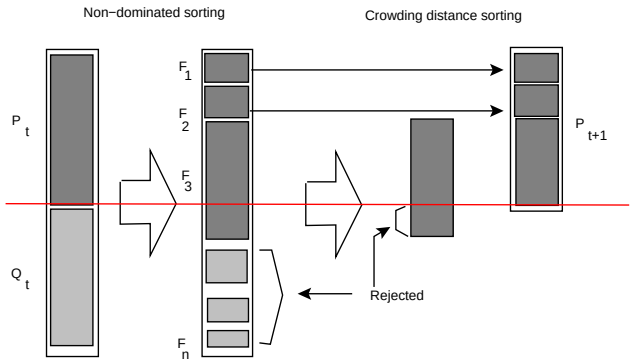


Figure 12: Non-dominated sorting scheme (Deb et al., 2002)

3.2 *Generalized Expansion Method*

To solve the trade-off problem, we need a good estimate for the throughput, $\Theta(\mathbf{x})$. The Generalized Expansion Method (GEM) has been used successfully in the past (Kerbache and Smith, 1987, 1988) to estimate performance measures for arbitrarily configured finite queueing networks. The GEM is a combination of node-by-node decomposition and repeated trials, in which each queue is analyzed separately and then corrections are made to take into account the relationships between the queues in the network. The GEM uses type I blocking, that is, the upstream node is blocked if the service on a customer is completed, but it cannot move downstream because the queue at the downstream node is full. This is sometimes referred to as *blocking after service* and is prevalent in most production and manufacturing, transportation, and similar systems. Of course, arrivals from outside that find a full buffer are rejected. With the help of Fig. 13 we now describe the GEM.

We remark that the exponential distribution is a good approximation for the interdeparture times of entities leaving an $M/G/1/K$ node. In fact, it is possible to show the quasi-reversibility of a broader class of finite queues, which are the state-dependent $M/G/C/C$ queues (Cheah, 1990; Cheah and Smith, 1994). When those entities that are lost are included, the output stream is Poisson. This assumption is supported by several empirical results (Smith and Cruz, 2005; Cruz et al., 2008, 2010; Smith et al., 2009, 2010; Andriansyah et al., 2010; van Woensel et al., 2010).

The following three stages are involved in the GEM: *network reconfiguration*, *parameter estimation*, and *feedback elimination*.

Network Reconfiguration

The first step involves reconfiguring the network. An auxiliary vertex h_j is created, which is modeled as an $M/G/\infty$ queue with service rate μ_h and precedes each finite queue j. When an entity leaves i, vertex j may be blocked with probability p_{K_j} (that is, $p_{K_j} \equiv P(Y = K_j)$, where Y is the random number of entities in the queue), or unblocked with probability $(1 - p_{K_j})$. Under block-

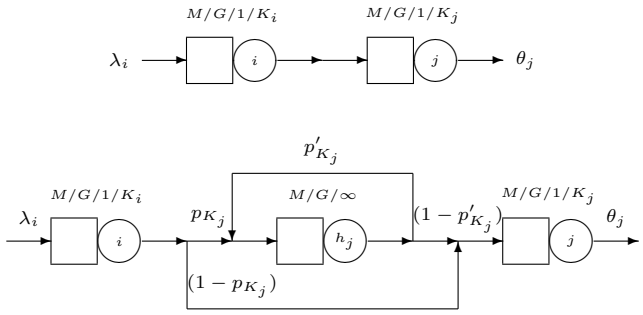


Figure 13: Generalized expansion method

ing, the entities are rerouted to vertex h_j for a delay while node j is busy. After this delay, the entity may be blocked again with probability p'_{K_j}, for a second delay period. Vertex h_j accumulates the time an entity has to wait before entering vertex j and the effective arrival rate to vertex j.

Parameter estimation

In the second stage, we essentially estimate the parameters p_K, p'_K, and μ_h, (for clarity, we omit the subscript for node j).

1. p_K is obtained by means of a two-moment approximation recently developed by Smith (2003)

$$p_K = \frac{\rho^{\left(\frac{2+\sqrt{\rho}s^2-\sqrt{\rho}+2K}{2+\sqrt{\rho}s^2-\sqrt{\rho}}\right)}(-1+\rho)}{\rho^{\left(2\frac{2+\sqrt{\rho}s^2-\sqrt{\rho}+K}{2+\sqrt{\rho}s^2-\sqrt{\rho}}\right)}-1},$$

where $\rho = \lambda/\mu$ is the traffic intensity at the node, that is, the ratio between the arrival rate λ and the service rate μ; and $s^2 = \mathrm{Var}(T_s)/\mathrm{E}(T_s)^2$ is the squared coefficient of variation of the service time T_s at the node. More details on two-moment approximations are provided by Choi et al. (2005).

2. p'_K is obtained with the following approximation from diffusion techniques given by Labetoulle and Pujolle (1980)

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

where r_1 and r_2 are the roots to the polynomial

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

with $\lambda = \lambda_j - \lambda_h(1-p'_K)$, λ_h is the actual arrival rate to the artificial holding node and λ_j is the actual arrival rate to the finite node j, given by

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

3. μ_h is calculated as follows using renewal theory

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

where σ_j^2 is the service time variance.

Feedback elimination

The repeated visits to the holding nodes (due to the feedback), create strong dependence in the arrival process. Therefore, the repeated immediate feedback is eliminated. This is accomplished 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

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

Summary

In summary, the ultimate goal of GEM is to provide an approximation scheme to update the service rates of upstream nodes that take into account all blocking after service caused by downstream nodes

$$\tilde{\mu}_i^{-1} = \mu_i^{-1} + p_K(\mu'_h)^{-1}.$$

Ultimately, for each finite node j in the network succeeding node i, we have simultaneous non-linear equations in variables p_K, p'_K, and μ_h, along with auxiliary variables such as λ and $\tilde{\lambda}_i$. Solving these equations simultaneously, we can compute all the performance measures of the network

$$\lambda = \lambda_j - \lambda_h(1-p'_K), \quad (8)$$

$$\lambda_j = \tilde{\lambda}_i(1-p_K), \quad (9)$$

$$\lambda_j = \tilde{\lambda}_i - \lambda_h, \quad (10)$$

$$p'_K = \left(\frac{\mu_j+\mu_h}{\mu_h} - \frac{\lambda\Big((r_2^K - r_1^K) - (r_2^{K-1} - r_1^{K-1})\Big)}{\mu_h\Big((r_2^{K+1} - r_1^{K+1}) - (r_2^K - r_1^K)\Big)}\right)^{-1}, \quad (11)$$

$$z = (\lambda+2\mu_h)^2 - 4\lambda\mu_h, \quad (12)$$

$$r_1 = \frac{[(\lambda+2\mu_h) - z^{\frac{1}{2}}]}{2\mu_h}, \quad (13)$$

$$r_2 = \frac{[(\lambda+2\mu_h) + z^{\frac{1}{2}}]}{2\mu_h}, \quad (14)$$

$$p_K = \frac{\rho^{\left(\frac{2+\sqrt{\rho}s^2-\sqrt{\rho}+2K}{2+\sqrt{\rho}s^2-\sqrt{\rho}}\right)}(-1+\rho)}{\rho^{\left(2\frac{2+\sqrt{\rho}s^2-\sqrt{\rho}+K}{2+\sqrt{\rho}s^2-\sqrt{\rho}}\right)}-1}. \quad (15)$$

Eq. (8) through Eq. (11) are related to the arrivals and feedback in the holding node. Eq. (12) through Eq. (14) are used to solve Eq. (11) with z used as a dummy parameter for simplicity. Lastly, Eq. (15) gives the blocking probability for the $M/G/1/K$ queue. Thus, we essentially have five equations to solve Eq. (8)–Eq. (11) and Eq. (15). An important point about this process is that we do not physically modify the networks; we only represent the expansion process through the software. For a slightly different implementation of the GEM, with flow-conservation traffic equations, the reader is referred to Cruz and Smith (2007).

4 COMPUTATIONAL RESULTS AND DISCUSSION

The GEM and the optimization algorithms were implemented in FORTRAN, and all code is available from the authors upon request. The first part of the experiments was designed to set up the MOGA parameters to optimize its convergence. Experiments were run for tandem, split, and merge topology queues. These networks are shown in Fig. 14.

4.1 MOGA parameter setup

The first set of experiments involves the study and discovery of the best values for the main parameters of MOGA: the population size, `popSize`; the mutation rate, `rateMut`; and parameter η, which controls the dispersion of random variable β_q used in the SBX operator, Eq. (6), to guarantee the fastest possible convergence rate. A strategy of experimentation extensively used in practice is the one-factor-at-a-time approach, which consists of selecting a starting point for each factor, then successively varying each factor over its range with the other factors kept constant. As noticed by Montgomery (2004), the major disadvantage of such a strategy is that it fails to account for any possible iteration between the factors, *i.e.*, the failure of one factor to produce the same effect on the response of another factor at different levels. One way to overcome this is to adopt the factorial experimental design (Montgomery, 2004). We propose a factorial experimental design configured with three fixed factors (factor A, the population size, `popSize`, factor B, the mutation rate, `rateMut`, and factor C, the crossover parameter, η). The possible interactions between factors A*B, A*C, B*C, and A*B*C are also of interest. All three factors will have three levels, respectively, factor A, `popSize` $= \{20, 40, 80\}$, factor B, `rateMut` $= \{0.01, 0.02, 0.10\}$, and factor C, $\eta = \{4, 8, 16\}$, and each experiment will be replicated 3 times, for 10-node series networks. There are a total of 81 experiments. We consider the three-factor analysis of variance model:

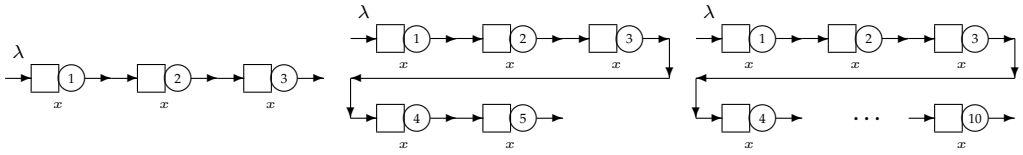
$$Y_{ijkl} = \mu + \tau_i + \beta_j + \gamma_k + (\tau\beta)_{ij} + (\tau\gamma)_{ik} + (\beta\gamma)_{jk} + (\tau\beta\gamma)_{ijk} + \epsilon_{ijkl},$$

in which $i, j, k = 1, 2, 3$ and

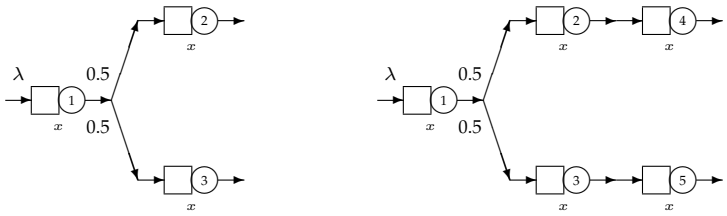
- μ is the overall mean effect;
- τ_i is the effect of the ith level of factor A, subject to $\sum_i \tau_i = 0$;
- β_j is the effect of the jth level of factor B, subject to $\sum_j \beta_j = 0$;
- γ_k is the effect of the kth level of factor C, subject to $\sum_k \gamma_k = 0$;
- $(\tau\beta)_{ij}$ is the effect of the interaction between τ_i and β_j, subject to $\sum_{i,j}(\tau\beta)_{ij} = 0$;
- $(\tau\gamma)_{ik}$ is the effect of the interaction between τ_i and γ_k, subject to $\sum_{i,k}(\tau\gamma)_{ik} = 0$;
- $(\beta\gamma)_{jk}$ is the effect of the interaction between β_j and γ_k, subject to $\sum_{j,k}(\beta\gamma)_{jk} = 0$;
- $(\tau\beta\gamma)_{ijk}$ is the effect of the interaction between τ_i, β_j, and γ_k, subject to $\sum_{ijk}(\tau\beta\gamma)_{ijk} = 0$; and
- ϵ_{ijkl} is a random error component which are random variables i.i.d. normally distributed with zero mean and variance σ^2.

The response variable is the number of generations until the convergence measure σ_L defined in Eq. (7) first reaches a certain threshold, 0.03. For this part of the experiments, the maximum number of generations was set to 10,000. The analysis of variance of the results, performed in MINITAB, is shown in Table 1. A logarithmic transformation was selected as a variance-stabilizing transformation for the response variable, which is the number of iterations until reaching $\sigma_L < 0.03$, to satisfy the assumptions of normality and homoscedasticity. Before the conclusions from the analysis of variance are adopted, the adequacy of the model was checked and confirmed (results not shown), *i.e.*, the errors are normally and independently distributed with mean zero and constant, but unknown, variance σ^2.

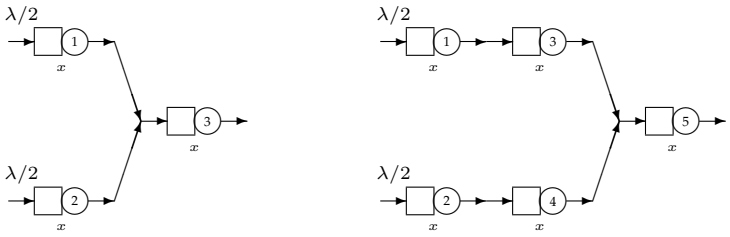
From Table 1, we can see that none of the possible iterations were significant, and that only the population size, `popSize` (P-value < 0.000), and the mutation rate, `rateMut` (P-value < 0.000), are important in the algorithm convergence speed. To assist in the practical interpretation of this experiment, Fig. 15 presents the plots of the three main effects. The main effect plots are merely graphs of the marginal response averages at the levels of the three factors. We note that not all the variables have positive main effects. In fact, the mutation rate reaches its best performance for the intermediate value 0.02. The population size somewhat stabilizes after 40, and we use 80. Finally, η reaches its best performance with the value 16, which is adopted from now on. From this optimal set of parameters (`popSize` $= 80$, `rateMut` $= 0.02$, and $\eta = 16$), it is clear that a maximum number of generations of 1,000 should suffice to achieve convergence and to ensure that the computation always finishes in a finite amount of time. Now we assess the performance of the algorithm in a variety of networks of queues.



(a) a three-node series topology (b) a five-node series topology (c) a ten-node series topology



(d) a three-node split topology (e) a five-node split topology



(f) a three-node merge topology (g) a five-node merge topology

Figure 14: Tested topologies

Table 1: Analysis of variance for the logarithm of the number of iterations before convergence

Source	DF	SS	MS	F	P
`popSize`	2	81.8120	40.9060	83.48	0.000
`rateMut`	2	0.9349	0.4674	0.95	0.392
η	2	10.6677	5.3338	10.89	0.000
`popSize` $*$ `rateMut`	4	1.2028	0.3007	0.61	0.655
`popSize` $*\ \eta$	4	2.5523	0.6381	1.30	0.281
`rateMut` $*\ \eta$	4	2.9694	0.7424	1.52	0.211
`popSize` $*$ `rateMut`$*\eta$	8	6.1057	0.7632	1.56	0.160
Error	54	26.4592	0.4900		
Total	80	132.7040			

4.2 Performance assessment

The complete results were generated using the following experimental design. The arrival rates considered were $\lambda = \{5.0, 7.0, 8.0\}$; the service rates, $\mu_i = 10.0$, $\forall\ i \in N$, result in a system utilization of $\rho = \lambda/\mu = \{0.5, 0.7, 0.8\}$ combined with several values for the squared coefficient of variation, $s^2 = \{0.5, 1.0, 2.0\}$, and number of nodes, $|N| = \{3, 5, 10\}$ for tandem networks, and $|N| = \{3, 5\}$ for merge and split topologies, which requires 64 different experiments. These experiments were designed to match some of the experiments reported by Smith and Cruz (2005).

The resulting boxplots of the number of generations until convergence are presented in Fig. 16 as a function of the topology, number of nodes, arrival rate, and s^2. Important conclusions that can be drawn from these experiments are that the topology and the squared coefficient of variation, s^2, do not seem to play an important role in the difficulty of the problem because the boxplots essentially show a similar average value and variability. On the other hand, as the number of nodes increases, the problem certainly requires more computation time for convergence. We note that some 10-node instances even reached the maximum number of generations (1000 generations) for the problem. We considered uniformly distributed initial solutions, $X \sim \mathcal{U}(0, x_{\max})$, with $x_{\max} = 25$, which put them close to the optimal solutions for $\lambda = 8.0$. When $x_{\max}$ was reduced to ten, for example, the initial solutions were closer to the optimal solution for $\lambda = 5.0$ (results not shown), and the boxplot for such a lambda changed accordingly to reflect the fact that now this was the case easiest to

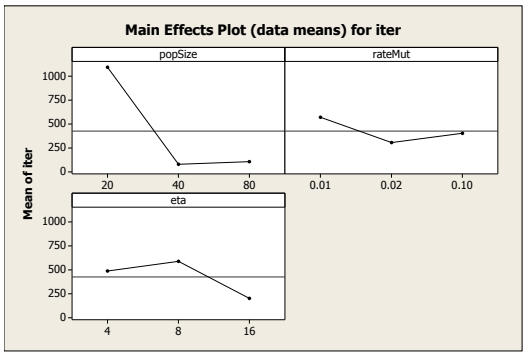


Figure 15: Main effects plots

solve. However, the boxplots for topology, number of nodes, and s^2 did not change significantly (results not shown).

As a final note on the performance of the algorithm, we remark that the stopping criterion (that is, $\sigma_{40} \leq 0.02$), seems to ensure quality solutions with a small amount of computational effort. Fig. 17 shows boxplots for the number of generations taken until the optimal solutions from Smith and Cruz (2005) first appeared in the Pareto-set approximation. Of course, there is no hard guarantee that the final approximations are accurate, but it certainly gives a good indication that we should not be far from the optimal Pareto set in all cases tested. Now, a more complete analysis will be performed for a more complex 16-node network.

4.3 Analysis of a large complex network

The 16-node complex network analyzed here is presented in Fig. 18 and extracted from Smith and Cruz (2005). For an arrival rate $\lambda = 5.0$, three different coefficients of variations were analyzed: $s^2 = \{0.5, 1.0, 2.0\}$. The results are presented in the following figures. In Fig. 19, we show the results in the objective function space. They make sense because the curves do not overlap and move smoothly as s^2 changes. For a given total buffer allocation, the throughput tends to be reduced as the squared coefficient of variation s^2 increases, as expected. We note that each point in this graph is a particular solution for a specific combination in the Pareto curve.

We also examined in detail some configurations in Fig. 19. Configurations 'A' to 'E' represent different overall buffer allocations and, consequently, different levels of congestion for the same squared coefficient of variation $s^2 = 0.5$ and for the same arrival rate $\lambda = 5$. In other words, different approximate non-dominated solutions were taken for different total buffer allocations, as seen in Table 2. Then, we examined how the overall allocation optimally spread out individually throughout the nodes of the network, that is, the approximate optimal buffer size associated with each one of the nodes, namely node #1, node #2, up to node #16.

Thus, the selected solutions 'A' to 'E' in Fig. 19 are illustrated in the decision variable space in Fig. 20. The results are encouraging because the algorithm was able to distribute the buffers evenly among the servers. We expected to see consistent allocations as the congestion increased (from solution 'E' to 'A'), which is confirmed by the results. It is interesting to see that, after the first node, a uniform pattern seems to occur for the different levels of congestion. This uniform buffer allocation should probably be expected, given the reasonably balanced topology presented in Fig. 18. The allocation pattern is kept fairly stable, as the throughput increases, which indicates a certain level of robustness in the solutions generated by the proposed methodology.

5 Conclusions and Final Remarks

In this paper, we developed a multi-objective approach for the buffer allocation and throughput trade-off problem for single server general queueing networks, which is useful for those cases when one does not want to merely consider cost as the bottom-line single objective. Combining the generalized expansion method (GEM), which is a performance evaluation tool, with a multi-objective genetic algorithm (MOGA) gives insightful Pareto curves. These curves explicitly show the trade-off between buffer spaces and throughput. Experimental results confirm optimal solutions found in earlier papers and show the merits of this approach.

Remaining questions include how well this methodology applies to different trade-off problems in finite queueing networks. Just to give a few examples, we mention server allocation/throughput trade-off problems and joint buffer-server allocation/throughput trade-off problems (for instance, in finite general multi-server queueing networks).

Possible future research directions include various applications of the algorithm to manufacturing and assembly problems, facility planning and layout design, telecommunication, and computer system network design problems. Another possible extension is to include networks with feedback loops because they model many important industrial systems, such as systems with captive pallets and fixtures or reverse streams of products due to re-work.

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

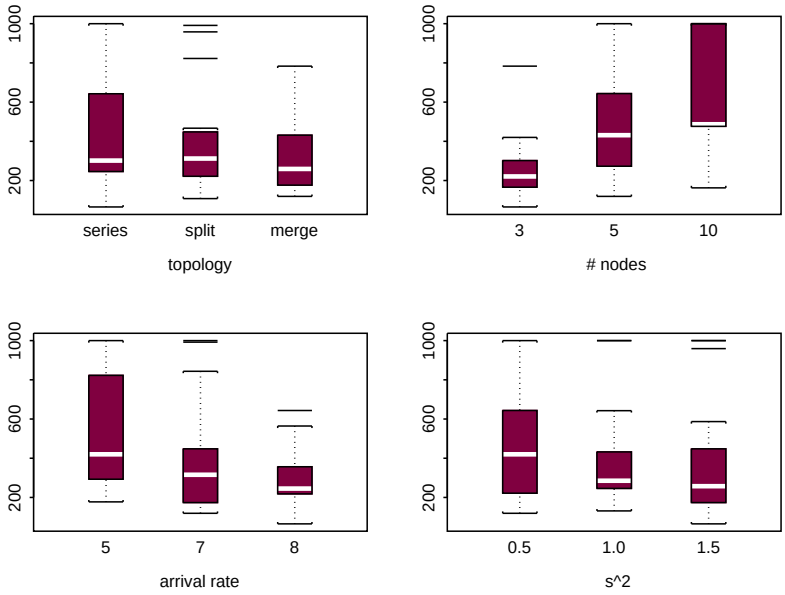


Figure 16: Number of generations until convergence

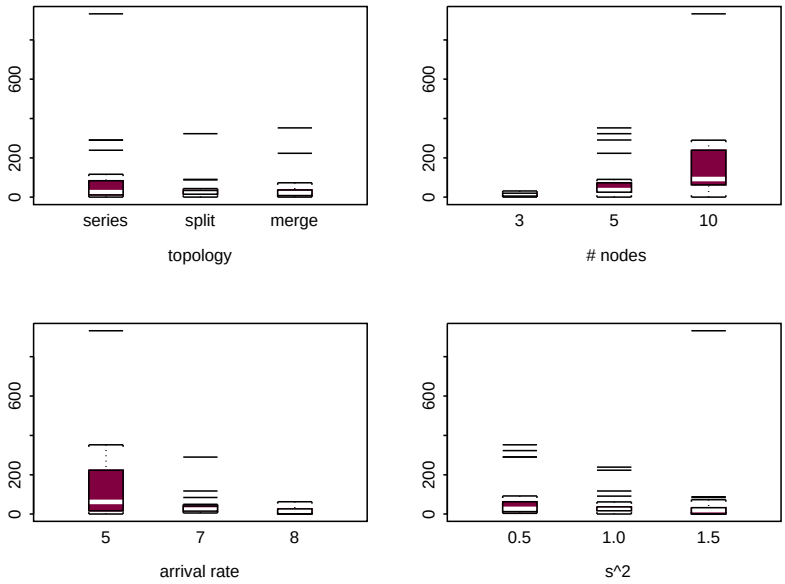


Figure 17: Number of generations until reaching optimality

Table 2: Configurations analyzed

Configuration	Total buffer allocation, $\sum_i c_i x_i$	Throughput, $\Theta(\mathbf{x})$
A	25	4.3365
B	39	4.8322
C	56	4.9630
D	78	4.9950
E	116	4.9999

(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).

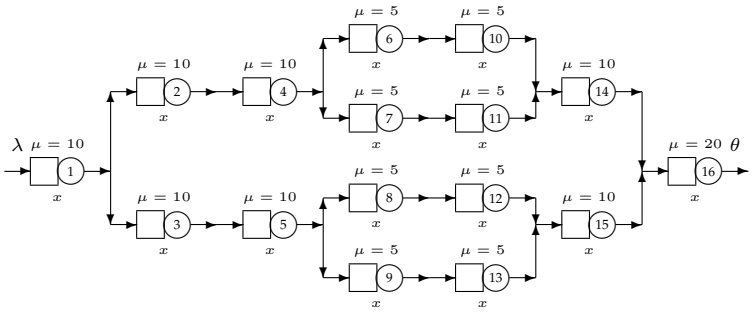


Figure 18: A 16-node complex network (Smith and Cruz, 2005)

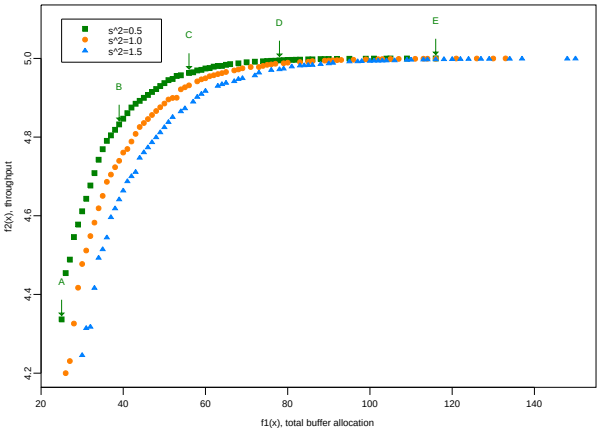


Figure 19: Profile analysis

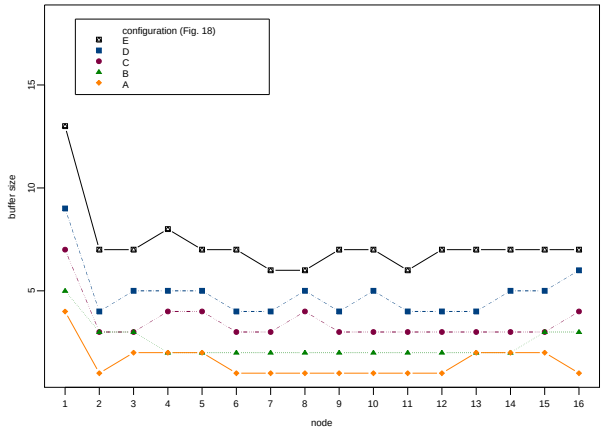


Figure 20: Allocation analysis

REFERENCES

Andriansyah, R., van Woensel, T., Cruz, F. R. B., and Duczmal, L., 2010. Performance optimization of open zero-buffer multi-server queueing networks. Computers & Operations Research, 37 (8), 1472–1487.

Chankong, V. and Haimes, Y. Y., 1983. Multiobjective Decision Making: Theory and Methodology. Amsterdam, The Netherlands: Elsevier.

Cheah, J., 1990. Modeling Traffic as $M/G/C/C$ State Dependent Queues. Master's thesis, University of Massachusetts, Amherst, MA.

Cheah, J. and Smith, J. M., 1994. Generalized $M/G/C/C$ state dependent queueing models and pedestrian traffic flows. Queueing Systems, 15, 365–386.

Choi, D. W., Kim, N. K., and Chae, K. C., 2005. A two-moment approximation for the $GI/G/c$ queue with finite capacity. INFORMS Journal on Computing, 17 (1), 75–81.

Coello, C. A. C., 2000. An updated survey of GA-based multiobjective optimization techniques. In: Proceedings of the ACM Computing Surveys, volume 32, 109–143.

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

Cruz, F. R. B. and Smith, J. M., 2007. Approximate analysis of $M/G/c/c$ state-dependent queueing networks. Computers & Operations Research, 34 (8), 2332–2344.

Cruz, F. R. B., van Woensel, T., Smith, J. M., and Lieckens, K., 2010. On the system optimum of traffic assignment in $M/G/c/c$ state-dependent queueing networks. European Journal of Operational Research, 201 (1), 183–193.

Deb, K. and Agrawal, R. B., 1995. Simulated binary crossover for continuous search space. Complex Systems, 9, 115–148.

Deb, K. and Beyer, H.-G., 1999. Self-adaptive genetic algorithms with simulated binary crossover. Technical report no. CI-61/99, Department of Computer Science/XI, University of Dortmund, 44221 Dortmund, Germany.

Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T., 2002. A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Transactions on Evolutionary Computation, 6 (2), 182–197.

Faria, J., Matos, M., and Nunes, E., 2006. Optimal design of work-in-process buffers. International Journal of Production Economics, 99 (1-2), 144–155.

Fonseca, C. M. and Fleming, P., 1995. An overview of evolutionary algorithms in multiobjective optimization. Evolutionary Computing, 3 (1), 1–16.

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

Goldberg, D. E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc..

Hu, X.-B. and Di Paolo, E., 2007. An efficient genetic algorithm with uniform crossover for the multiobjective airport gate assignment problem. In: IEEE Congress on Evolutionary Computation, CEC 2007, 55–62, Singapore, doi:10.1109/CEC.2007.4424454.

Jouini, O., Dallery, Y., and Aksin, Z., 2009. Queueing models for full-flexible multi-class call centers with real-time anticipated delays. International Journal of Production Economics, 120 (2), 389–399.

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

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

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

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

Martí, L., García, J., Berlanga, A., and Molina, J. M., 2007. A cumulative evidential stopping criterion for multiobjective optimization evolutionary algorithms. In: GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation, 2835–2842, New York, NY, USA: ACM, doi:http://doi.acm.org/10.1145/1276958.1277141.

Montgomery, D. C., 2004. Design and Analysis of Experiments. New York, NY, USA: John Wiley & Sons, Inc., 6th edition.

Nahas, N., Ait-Kadi, D., and Nourelfath, M., 2006. A new approach for buffer allocation in unreliable production lines. International Journal of Production Economics, 103 (2), 873–881.

Osorio, C. and Bierlaire, M., 2009. An analytic finite capacity queueing network model capturing the propagation of congestion and blocking. European Journal of Operational Research, 196 (3), 996–1007.

Purshouse, R. C. and Fleming, P. J., 2007. On the evolutionary optimization of many conflicting objectives. IEEE Transactions on Evolutionary Computation, 11 (6), 770–784.

Ramli, R., Yamamoto, H., and Qudeiri, J. A., 2009. Tool path of lathe machine in flexible transfer lines by using genetic algorithms. International Journal of Production Economics, 121 (1), 72–80.

Rudenko, O. and Schoenauer, M., 2004. 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.

Shi, C. and Gershwin, S. B., 2009. An efficient buffer design algorithm for production line profit maximization. International Journal of Production Economics, 122 (2), 725–740.

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

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

Smith, J. M., Cruz, F. R. B., and van Woensel, T., 2009. Optimal server allocation in general, finite, multi-server queueing networks. Applied Stochastic Models in Business & Industry (in press). doi:10.1002/asmb.813.

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

Sywerda, G., 1989. Uniform crossover in genetic algorithms. In: Proceedings of the third international conference on Genetic algorithms, 2–9, San Francisco, CA, USA: Morgan Kaufmann Publishers Inc..

Tan, B. and Gershwin, S. B., 2009. Analysis of a general Markovian two-stage continuous-flow production system with a finite buffer. International Journal of Production Economics, 120 (2), 327–339.

van Woensel, T., Andriansyah, R., Cruz, F. R. B., Smith, J. M., and Kerbache, L., 2010. Buffer and server allocation in general multi-server queueing networks. International Transactions in Operational Research, 17 (2), 257–286.