Journal of the Brazilian Computer Society, 2026, 32:1, doi: 10.5753/jbcs.2026.5178

Optimal resource allocation in networks of general single-server finite queues

Gabriel L. Souza [**Universidade Federal de Ouro Preto** | ***gabriel.souza@ufop.edu.br***]
Anderson R. Duarte [**Universidade Federal de Ouro Preto** | ***anderson.duarte@ufop.edu.br***]
Frederico R. B. Cruz [**Universidade Federal de Minas Gerais** | ***fcruz@est.ufmg.br***]
Gladston J. P. Moreira [**Universidade Federal de Ouro Preto** | ***gladston@ufop.edu.br***]

Computing Department, Universidade Federal de Ouro Preto, Campus Morro do Cruzeiro, Ouro Preto-MG, 35402-136, Brazil.

Received: 11 November 2024 • **Accepted:** 02 June 2025 • **Published:** 24 April 2026

Abstract This study simultaneously considers minimizing the total buffer allocation and the overall service rates in the network while maximizing its throughput. Some algorithms have already been proposed in the literature, but the discussion of efficient alternatives is relevant. We develop a novel approach to multi-objective particle swarm optimization (MO-PSO). We apply this approach to an acyclic, general single-server finite queueing network to optimize throughput. This algorithm was specifically tailored to address the problem, which involves mixed-integer variables and constraints that depend on the current solution, since service rates cannot fall below arrival rates. The proposed approach simultaneously decreases the total buffer allocation and the overall service rate. Consequently, our method yields a suboptimal Pareto set for these conflicting objectives. We conducted a computational and experimental study to verify the effectiveness of the proposed approach and to compare it with previously proposed solutions. The insights gained can enhance the design of queue networks.

Keywords: Multi-objective optimization, Particle swarm optimization, Buffer allocation, Service rate allocation, Queueing networks

1 Introduction

Several studies emphasize throughput as a crucial performance metric in queueing networks [Cruz *et al*., 2010, 2012; Cruz, 2009]. The primary focus of this study is to maximize throughput (Θ) in an acyclic general single-server finite queueing network. Finding the most efficient solution to queueing network problems involves allocating the smallest total capacity and the lowest overall service rates to achieve the highest Θ. Notably, many everyday scenarios are interconnected with general single-server finite queueing networks. Therefore, this study specifically targets the optimization of such queueing network problems. In other words, this study primarily focuses on networks of queues featuring Poisson (Markovian) arrivals, single servers, service times that are generally independent and identically distributed, and a maximum total capacity of k customers (including those in service). These queues are called $M/G/1/k$ in Kendall notation.

It is worth emphasizing that the methodology described herein draws inspiration from several potential real-world applications in the existing literature. Here are some examples. In the work of Kose and Kilincci [2020] and Zennaro *et al*. [2022], queueing models were applied to production lines, illustrating the adaptability and practical relevance of developing efficient algorithms to optimize these models. Research by MacGregor Smith [2018] and Hernández-Vázquez *et al*. [2019] details improvements to industrial processes using queues, further underscoring the practical utility of algorithms for their optimization. The works by Xi *et al*. [2022] and Kassoul *et al*. [2022] exemplify queueing-based methodologies, such as the one described here, seamlessly integrated into production-system optimization, thereby enhancing operational efficiency. The healthcare sector may also benefit from our methods, as evidenced by the comprehensive applications discussed by Almehdawe *et al*. [2019] and Ingolfsson *et al*. [2020], which demonstrate the effectiveness of queues in addressing challenges within health systems.

Furthermore, in the domain of vehicle and pedestrian traffic modeling, the articles by Khalid *et al*. [2020] and Liu *et al*. [2021] underscore the versatility of queue networks for understanding and improving traffic dynamics. It is also worth emphasizing the applicability of queues in computer and communication systems, as suggested by Inzillo *et al*. [2019] and Pourjavad and Almehdawe [2022]. Additionally, there is a compelling application detailed by Cruz *et al*. [2018] revolving around a manufacturing example involving a vehicle assembly system conceptual design. This particular case highlights the necessity of finite buffers and the optimization of service rates.

An example of a queueing network addressed in this study is depicted in Figure 1. In particular, our research aims to maximize throughput and minimize overall service rates and total buffer allocation. An essential consideration is the trade-off among evaluating overall service rates, total buffer allocation, and resulting throughput. These objective functions conflict because increased throughput requires higher capacities and service rates. In the queueing network illustrated in Figure 1, the parameters λ, Θ, and π_{ij} represent, respectively, the Markovian arrival rate to the network, the system throughput, and the probability that an entity departing from the i-th queue

is routed to the j-th queue.

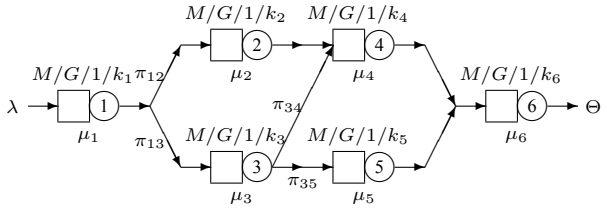


Figure 1. Complex network of 6 queues, adapted from MacGregor Smith and Cruz [2005].

Cruz *et al*. [2012, 2018] and Souza *et al*. [2020, 2023] have presented various approaches using a multi-objective evolutionary algorithm (MOEA), particularly the well-known elitist non-dominated sorting genetic algorithm (NSGA-II), to provide efficient solutions to this problem. While these studies offer different problem-solving strategies, they all employ the same MOEA. In our study, we adopted a mathematical formulation similar to that presented by Cruz *et al*. [2012, 2018]. On the other hand, Souza *et al*. [2020, 2023] introduced a post-processing strategy that uses the classic particle swarm optimization algorithm (PSOA) for another mathematical programming formulation. In their approach, the initial solutions generated by the MOEA are further refined by the PSOA, yielding improvements. Despite the different mathematical formulations, these previous studies have motivated us to consider the PSOA as a potentially suitable solution for our problem. Among derivative-free heuristic optimization algorithms, multi-objective PSOA (MO-PSO) is widely used. This category of algorithms has proven highly effective in addressing complex optimization problems with non-linear functions, where derivative-based optimization techniques become impractical. Our research problem aligns perfectly with this scenario, further justifying our choice of MO-PSO as the preferred optimization approach.

This article develops a MO-PSO to solve a difficult stochastic multi-objective optimization problem (see Section 2). Although the MO-PSO is well explored in the literature, the algorithm has been adapted specifically to address unique features of the stochastic model, including mixed-integer variables and tight boundary constraints. This context demands a thorough evaluation of the MO-PSO structure for this problem. Despite these difficulties, we found that the MO-PSO provided better solutions to the problem than before. Furthermore, the contribution to producing this fine-tuning for the algorithm is essential since overly complex experiments were not required to obtain an adequate configuration. An extensive computational experiment is performed to report the efficiency of this approach.

Furthermore, comparisons with the NSGA-II are presented in Section 3, along with important insights into the solutions produced. Findings on the fact that MO-PSO better distributes resources across networks, as solutions with the same amount of resources were able to provide better performance measures. Finally, Section 4 addresses concluding remarks and outlines possible future research.

2 Material and Methods

The methodological content was divided into two stages. Firstly, the multi-objective mathematical programming formulation was presented. Subsequently, we present the two algorithms used: the performance evaluation algorithm, which estimates the queueing network's throughput given a total buffer allocation and service rates. Afterward, the optimization algorithm that yields optimal solutions to the proposed multi-objective mathematical formulation was detailed.

2.1 Problem formulation

Several formulations have already been proposed in the literature, most of them in a single-objective setting. Typically, problems involving queueing networks are modeled as a graph $\mathcal{G}(V, A)$, where V is a finite set of m queues (vertices) and A is a finite set of arcs (connections) between queues.

We can cite some classic formulations: the server allocation problem (CAP) [MacGregor Smith, 2015; Duarte, 2024], the buffer allocation problem (BAP) [MacGregor Smith and Cruz, 2005; Cruz *et al*., 2008], and the simultaneous buffer and server allocation problem (BCAP) [Martins *et al*., 2019]. All these formulations have single objectives. Their main purpose is to minimize resource use in buffer allocation, service rates, or both, while ensuring a minimum threshold of network performance, usually measured by throughput. For all these formulations, it is possible to propose a dual formulation to maximize throughput, subject to constraints on the maximum resource levels for buffers and service rates.

However, it is worth noting that before these successful multi-objective formulations, such formulations were relatively scarce. Only in the recent past have we seen the emergence of successful multi-objective formulations for this problem. In this context, a noteworthy contribution is made by Cruz *et al*. [2012], who introduce a comprehensive multi-objective formulation that integrates various possibilities related to the problem. This formulation is defined as follows:

$$\text{minimize } F(\mathbf{K}, \boldsymbol{\mu}) = \Big[f_1(\mathbf{K}), f_2(\boldsymbol{\mu}), f_3(\mathbf{K}, \boldsymbol{\mu})\Big], \quad \text{(1a)}$$

s.t.:

$$\begin{aligned} &k_i \in \mathbb{N}, \\ &\mu_i \geqslant 0. \end{aligned} \quad \text{(1b)}$$

for all $i \in \{1, 2, \ldots, m\}$.

Several key variables and functions are introduced to represent important aspects of the problem in the formulation above. Firstly, the variable k_i represents the overall capacity of the i-th queue, taking into account both items currently in service and those waiting in the queue, *i.e.*, $k_i = b_i + 1$, where b_i denotes the allocated buffers. The function $f_1(\mathbf{K}) = \sum_{i=1}^{m} k_i$ aggregates these individual queue capacities to calculate the total capacity allocation across all queues, analogously to the sum of the individual queue buffers with the m servers. Secondly, $f_2(\boldsymbol{\mu}) = \sum_{i=1}^{m} \mu_i$ represents the overall service rate allocation. It sums the service rates for each queue to provide a measure of the total service-rate allocation across all queues. Thirdly, the function $f_3(\mathbf{K}, \boldsymbol{\mu}) = -\Theta(\mathbf{K}, \boldsymbol{\mu})$ is introduced to quantify the throughput. The negative sign in

$f_3(\mathbf{K}, \boldsymbol{\mu})$ is applied because the goal is to maximize throughput. Throughput is a critical metric in queueing networks, as it measures the rate at which items are successfully processed or served.

It is worth emphasizing that estimating throughput in general service queueing networks is a complex task that requires a specialized approach. This estimate is obtained through repeated trials of a node-by-node decomposition method called the Generalized Expansion Method (GEM), which we will elaborate on shortly. Understanding this estimation process is vital for effectively applying the formulation we have presented.

In essence, this formulation provides a framework for capturing the intricate trade-offs among total buffer allocation, service rate allocation, and throughput in queueing networks. It provides the groundwork for optimizing these critical aspects, enhancing the overall performance of such networks.

2.2 Algorithms

2.2.1 Generalized Expansion Method

We use the GEM to evaluate the performance of a queueing network for a specific parameter set. This method, originally introduced by Kerbache and MacGregor Smith [1987, 1988], is further elaborated upon in this article section.

First, consider a finite $M/G/1/k$ single queue. We can estimate Θ (throughput) using a two-moment approximation proposed by Kimura [1996] for the blocking probability, P_k, which is a suitable approximation for a wide range of values [MacGregor Smith and Cruz, 2005; Cruz *et al*., 2008]:

$$P_k = \frac{\rho^{\left(\frac{2+\sqrt{\rho}\mathrm{cv}^2-\sqrt{\rho}+2(k-1)}{2+\sqrt{\rho}\mathrm{cv}^2-\sqrt{\rho}}\right)}(\rho-1)}{\rho^{\left(2\frac{2+\sqrt{\rho}\mathrm{cv}^2-\sqrt{\rho}+(k-1)}{2+\sqrt{\rho}\mathrm{cv}^2-\sqrt{\rho}}\right)}-1}, \tag{2}$$

in which the ratio $\rho = \lambda/\mu$, known as the traffic intensity, must satisfy $\rho < 1$ for queue stability (λ and μ denote the arrival and service rates, respectively). Additionally, $\mathrm{cv}^2 = \mathbb{V}(T_s)/\mathbb{E}^2(T_s)$ refers to the squared coefficient of variation of the service time, which is the ratio of the variance to the square of the expectation of the service time T_s.

When calculating Θ in a finite $M/G/1/k$ single queue, it is crucial to emphasize that adjustments are required for the effective arrival rate experienced by the queue. The blocking probability P_k, as defined in Eq. (2), represents the fraction of arrivals unable to enter the system due to a temporary lack of waiting space. Due to the PASTA property of Markovian arrivals (Poisson Arrivals See Time Averages), the effective arrival rate observed by the servers can be described as [Gross *et al*., 2009]:

$$\lambda_{\text{eff}} = \lambda(1 - P_k), \tag{3}$$

and Θ may be obtained by

$$\Theta = \lambda_{\text{eff}} = \lambda(1 - P_k). \tag{4}$$

In contrast, estimating throughput in queue networks presents a more intricate challenge. We employ the GEM, a computationally efficient technique for estimating network queue performance, as proposed by Kerbache and MacGregor Smith [1987, 1988]. GEM is an algorithm that evaluates the performance of networks of finite queues under arbitrary acyclic topological configurations. This approach combines repeated trials of a node-by-node decomposition method, analyzing each queue individually while accounting for interrelationships among network queues, as detailed below.

In the estimation process, if a customer attempts to enter a downstream node that is full, the customer must wait in the upstream node and therefore block it. This occurrence is usual in manufacturing, production, and transportation systems and is generally called *blocking after service*. For each finite vertex j, GEM creates an auxiliary node (h_j) modeled as an $M/G/\infty$ queue (see Fig. 2). Considering each of the customers in the network queueing, vertex j may be unblocked (with probability $1 - P_{k_j}$) or blocked (with probability P_{k_j}). When blocking occurs, the customers are redirected to node h_j and delayed while node j remains busy. The wait time at node h_j is recorded with a service rate μ'_h, provided by the GEM, before entering node j, and updated according to the effective arrival rate coming from node i to node j, $\lambda_{\text{eff}} = \lambda_i(1 - P_{k_i})$.

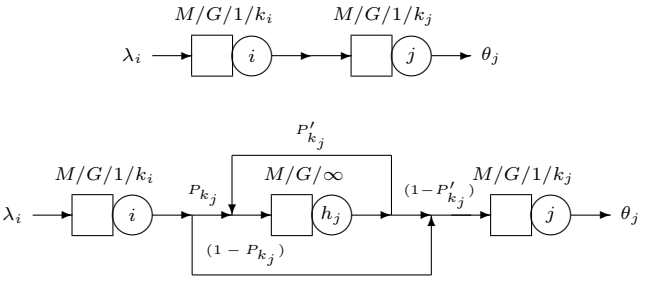


Figure 2. The GEM applied to a series network configuration.

The main target of the GEM is to provide the updated service rate of the i-th queue, μ_i, as follows:

$$\tilde{\mu}_i^{-1} = \mu_i^{-1} + P_{K_j}(\mu'_h)^{-1}. \tag{5}$$

In summary, as described earlier, the GEM accounts for queue delays due to potential blockages and iteratively updates system performance measures.

2.2.2 Multi-objective Optimization

Kennedy and Eberhart [1995] originally introduced the PSOA, an optimization technique inspired by the collective movement of various species. This algorithm emulates the behavior of flocks or swarms of animals in their search for food and their movements toward warmer or colder regions, all aimed at improving their living conditions. Typically, a leader guides the swarm's movement while the entire group collaborates to find a more favorable habitat for the species. We introduce a MO-PSO designed to simultaneously optimize the buffer sizes and service rates of networks consisting of general single-server finite queues.

According to these observations, the MO-PSO uses mathematical equations to replicate the swarm's movements and move a set of points (swarm particles) in search of optimal

positions. The MO-PSO is based on the concept of cooperation. In addition, it exhibits individuality and sociability. It can exchange information with neighbors, memorize a previous position, and use this information to improve search optimality. The particles are initially randomly positioned in the space of feasible solutions. At each iteration, these points move according to the information contained in the particles and the objective function. For each point, the movement depends on the direction and length of that movement. This operation is treated as the speed of a particle in that swarm.

Numerous details regarding the implementation of MO-PSO are available in the literature. Trivedi *et al*. [2020] offer simplified versions of MO-PSO, while Fan *et al*. [2017] introduce more complex variants. Additionally, Zhao *et al*. [2013] presents adapted versions tailored for mixed-integer mathematical programming formulations. Here, we propose a MO-PSO based on the classic implementation by Coello Coello and Lechuga [2002].

We proposed an MO-PSO, an extension of the single-objective PSOA by Kennedy and Eberhart [1995]. Each resource allocation configuration (buffers and service rates) determines a possible solution and is represented by a particle that aims to optimize the queueing network. In our proposal, the decision variables $(x_1, \ldots, x_\ell)$ correspond to $(k_1, k_2, \ldots, k_m, \mu_1, \mu_2, \ldots, \mu_m)$, where $\ell = 2m$ for a queueing network with m queues, representing each particle. Here, $k_i = b_i + 1$ is used for all queues, since they are single-server queues, and b_i denotes the allocated buffers. A mixed-integer problem with both integer and real variables is the multi-objective optimization problem addressed in this study. Therefore, defining a particle adaptation strategy is necessary to ensure its feasibility. These adaptations are part of the algorithm's fine-tuning to the problem under study.

The procedure changes the buffer allocation for each queue in the network, and since $k_i = b_i + 1 \geqslant 1$, $b_i \geqslant 0$ must always be ensured. Also, constraints related to service rates must be satisfied to ensure that $\rho_i < 1$. For all queues, the arrival rate λ_i needs to be less than the service rate μ_i. Thus, to ensure feasibility associated with the service rates, a reflection operator was used:

$$\mu_{\texttt{rfl}_i} = \lambda_i + |\mu_i - \lambda_i|, \tag{6}$$

in which λ_i represents the arrival rate at the i-th queue. Referring to Figure 1, note that λ_i is only known in advance at the entrance queue ($i = 1$). For all other queues ($i \neq 1$), the arrival rates depend on the flow from the preceding queues. To obtain these rates, the application of GEM (as described in Subsec. 2.2.1) is essential.

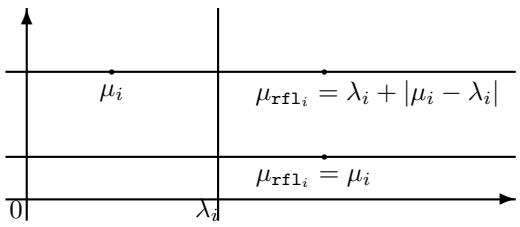


Figure 3. Schematics of the reflection operator for the service rate μ_i.

Figure 3 illustrates the effect of the reflection operator on the service rate. If $\mu_i < \lambda_i$, μ_i is updated to $\mu_{\texttt{rfl}_i}$. Otherwise, no change is made. This ensures that $\mu_i > \lambda_i$ and $\rho_i < 1$ are always maintained, preserving the feasibility of the solutions.

Resuming with the description of the developed MO-PSO, each particle, with $1 \leqslant i \leqslant s$ where s represents the swarm size (particle population), is determined with the following attributes:

- Particle position, $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,\ell})$;
- Particle speed, $v_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,\ell})$;
- Personal best particle position, p_i;
- Global best particle position, g_i.

The concepts of personal and global best position are intuitive for single-objective algorithms. In multi-objective algorithms, such as this adaptation of the PSOA, it is essential to have a well-defined dominance relation ($\prec$). Let $\mathbf{x}, \mathbf{x}'$ be two possible solutions in the decision variables space $\mathcal{X}$, then:

$$F(\mathbf{x}) \prec F(\mathbf{x}') \iff \begin{array}{l} F(\mathbf{x}) \leqslant F(\mathbf{x}') \text{ and} \\ F(\mathbf{x}) \neq F(\mathbf{x}'), \end{array} \tag{7}$$

where $F(\mathbf{x}) \leqslant F(\mathbf{x}')$ holds iff $f_j(\mathbf{x}) \leqslant f_j(\mathbf{x}')$, for all $j \in 1, 2, 3$, and $F(\mathbf{x}) \neq F(\mathbf{x}')$ holds iff there exists $j \in 1, 2, 3$ such that $f_j(\mathbf{x}) \neq f_j(\mathbf{x}')$. For the optimization problem described in Eq. (1a), a solution $\mathbf{x}^* \in \mathcal{X}$ is considered a *Pareto optimal solution* if there does not exist $\mathbf{x} \in \mathcal{X}$ such that $F(\mathbf{x}) \prec F(\mathbf{x}^*)$.

The optimization approach for the queueing network proposed using MO-PSO is presented in Algorithm 1. In this framework, the procedure **GenerateInitialSwarm** generates integer initial values for the variables k_i, and real values for μ_i, $\forall i$, according to a uniform distribution such that the values are > 0 and less than an arbitrarily large value (say 100), which defines the parameter bounds.

Algorithm 1 MO-PSO

Require: $\mathcal{G}(V, A), \lambda_i \; \forall i \in V$
$X^0 \leftarrow$ **GenerateInitialSwarm**(`swarmSize`)
$P \leftarrow X^0$
$\mathcal{F} \leftarrow$ **NonDominatedSort**(X^0) ▷ *find non-dominated fronts* $\mathcal{F} = (\mathcal{F}_1, \mathcal{F}_2, \ldots)$
for $t = 0$; $t <$ `maxIter`, t++ **do** ▷ *begin move swarm*
  $g_i \leftarrow$ **Rand**$(\mathcal{F})$
  for $i = 1$; $i \leqslant$ `swarmSize`; i++ **do**
    $v_i^{t+1} \leftarrow$ **Speed**(x_i^t, p_i, g_i)
    $x_i^{t+1} \leftarrow$ **NewPosition**(x_i^t, v_i^t)
    if x_i^{t+1} *dominates* p_i **then**
      $p_i \leftarrow x_i^{t+1}$
    else
      if p_i *does not dominate* x_i^{t+1} **then**
        $p_i \leftarrow$ **Rand**(x_i^{t+1}, p_i)
      end if
    end if
  end for
  $\mathcal{F} \leftarrow$ **NonDominatedSort**$(X^{t+1} \cup X^t)$
  $X^{t+1} \leftarrow$ **SelectFrom**$(\mathcal{F})$
end for ▷ *end move swarm*
write $X^{\texttt{maxIter}}$ ▷ *write final solution*

To describe the procedure **NonDominatedSort**$(\dot{})$, consider, for example, a bi-objective minimization problem. Figure 4 illustrates some possible solutions along with their dominance relationships. There are numbered points (I to VI) and unnumbered points. It is possible to verify that all unnumbered points are dominated points. In addition, points V and VI are also dominated. Note that point V is dominated by point I, while points I, II, and III dominate point VI. Note that point V is nondominated, for example, by point II. In the same way, point VI is nondominated by point IV. Points I to IV are nondominated by any of the other points. Therefore, points I-IV determine the best frontier. This best frontier approximates the Pareto set, the set of points that are nondominated by the other points.

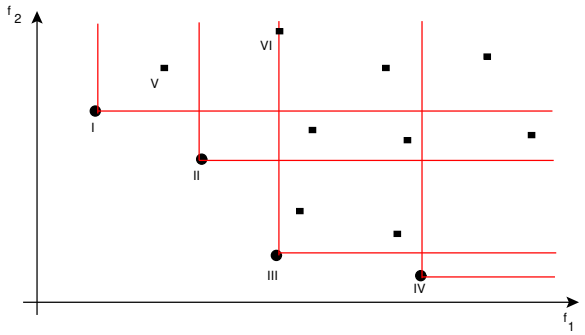


Figure 4. An example with nondominated (●) and dominated (■) points.

The procedure **NonDominatedSort**$(\dot{})$ is designed to partition the population into a series of dominance-based subsets called 'fronts'. It begins by identifying the first front, $\mathcal{F}_1$, which consists of all solutions that are not dominated by any other solution in the population; this set is often referred to as the empirical Pareto set. Following this, the solutions in $\mathcal{F}_1$ are temporarily excluded from the population, allowing the algorithm to identify the second front, $\mathcal{F}_2$, composed of solutions that are dominated only by members of $\mathcal{F}_1$. This iterative process continues, each time removing the solutions of the current front and then finding the next set of non-dominated solutions among the remaining population. As a result, every solution is assigned to a front that reflects its relative dominance. This ranking method effectively organizes the population into layers of optimality, with $\mathcal{F}_1$ representing the highest quality solutions, followed by $\mathcal{F}_2$, and so forth until all solutions are classified.

While not a common practice in most multi-objective particle swarm optimizers, the algorithm employs a selection process (referred to as step **SelectFrom**) by sequentially traversing the nondominated fronts ($\mathcal{F}_1, \mathcal{F}_2, \dots$) until the required number of individuals for the subsequent iteration to be performed. If adding a group of individuals of $\mathcal{F}_i$ surpasses the maximum allowable number of individuals, a specific criterion must be employed. The MO-PSO algorithm calculates the diversity metric introduced by Deb *et al.* [2002], commonly referred to as the *crowding distance*. Using the crowding distance aims to guarantee maximum diversity. The crowding distance is associated with the area of the cuboid defined in Figure 5. In higher dimensions, this is called the hyper-volume of a hyper-parallelepiped. Therefore, only the points with the highest crowding distance remain for subsequent iterations, as illustrated in Figure 5.

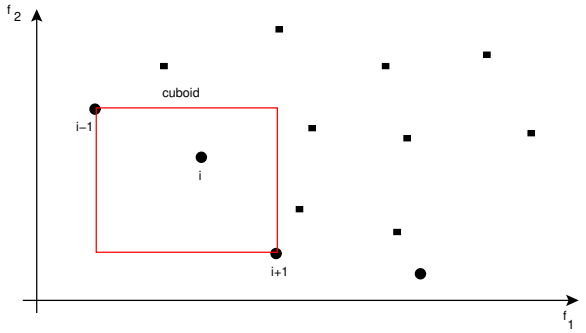


Figure 5. The rectangle for obtaining the crowding distance.

The particles' speed (**Speed**) and position (**NewPosition**) are updated in accordance with Eqs. (8a) and (8b), respectively:

$$v_i^{t+1} = w^t v_i^t + r_1\left(p_i - x_i^t\right) + r_2\left(g_i - x_i^t\right), \tag{8a}$$

$$x_i^{t+1} = x_i^t + v_i^{t+1}. \tag{8b}$$

Additionally, for the integer variables (buffers), the update of the position occurs as follows:

$$x_i^{t+1} = \text{int}\left(x_i^t + v_i^{t+1}\right); \tag{9}$$

For the real variables (service rates), the update of the position occurs as follows:

$$x_i^{t+1} = \left(x_i^t + v_i^{t+1}\right)_{\texttt{rfl}}. \tag{10}$$

reminding that the reflection operator must be applied, Eq. (6), to ensure feasibility, *i.e.*, that $\rho < 1$, as explained earlier.

This MO-PSO has three parameters. The parameters r_1 and r_2 are random values uniformly distributed over $[0, 1]$. In addition to them, w is the inertia weight. We experimentally evaluated different values for w, and although it had little influence on convergence, the best results were obtained with $w = 0.4$.

The speed and particle position updates are well established by adapting the MO-PSO for the problem under study. Figure 6 represents an analogy of the MO-PSO's operation to describe each particle's movement. It is possible to verify that

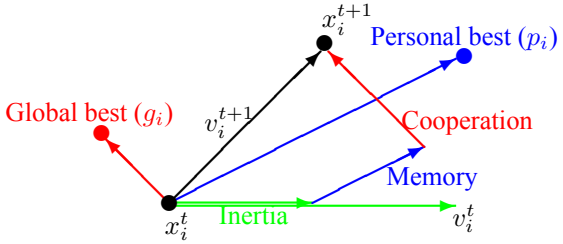
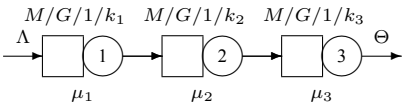
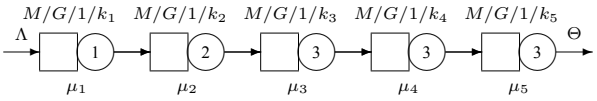


Figure 6. Schematics of particle movement in the MO-PSO.

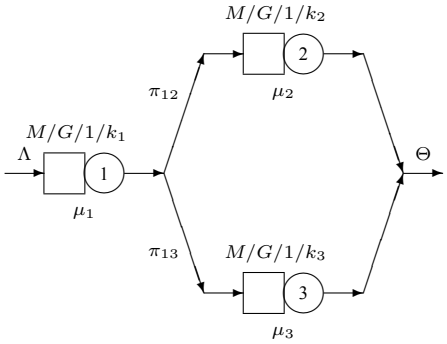
the movements are conducted by a contribution associated with inertia, a second contribution associated with memory (cognitive contribution), and finally, a third contribution associated with cooperation (social contribution).



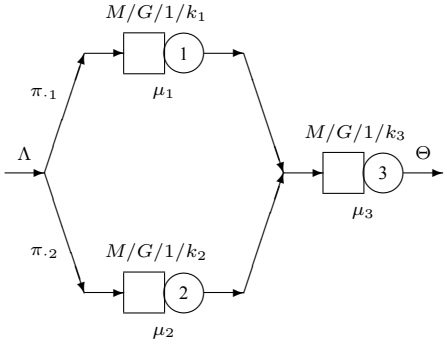
(a) 3-node series topology



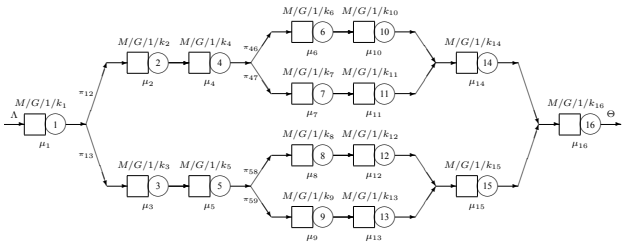
(b) 5-node series topology



(c) 3-node split topology



(d) 3-node merge topology



(e) 16-node mixed topology

Figure 7. Tested queueing networks.

2.3 Computational Experiments

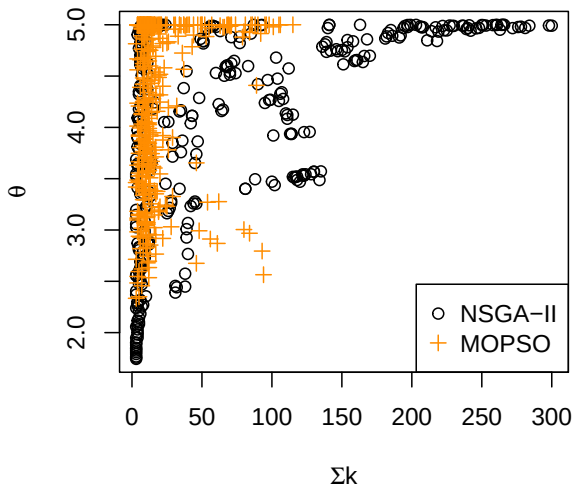
The MO-PSO version was implemented in FORTRAN for compatibility with previous implementations of the GEM [MacGregor Smith, 2003, 2004]. The codes implemented in this study are available for research and educational purposes upon prior request. We performed the computational experiments on the 11th-generation Intel(R) Core(TM) i7-1165G7 2.80 GHz processor with 8.00 GB of RAM.

Numerous computational experiments were conducted using various queueing networks. Figure 7 depicts the topologies employed, which have also been utilized in prior studies [MacGregor Smith and Cruz, 2005].

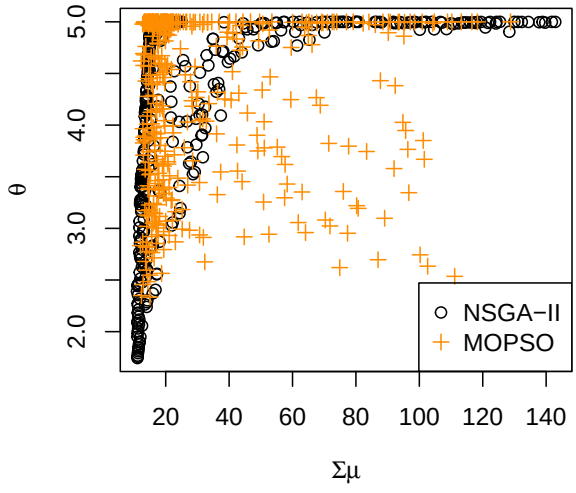
To explore the characterization of general service times, experiments used the following different values: $\mathrm{cv}^2 \in \{0.5, 1.0, 1.5\}$, representing hypo-exponential, exponential (Markovian), and hyper-exponential service times, respectively. All tested topologies used an external arrival rate $\Lambda = 5.0$. Furthermore, it is assumed that the routing vector, with components π_{ij} representing the probability of a customer departing from i-th queue and being routed to j-th queue in the mixed topologies, is known and assigns equal probabilities to all possible routes.

Regarding the comparative study, the computational experiments compared the results obtained using MO-PSO with those previously achieved with NSGA-II, which was graciously provided by Cruz *et al.* [2012]. The prior parameterization of the NSGA-II was preserved. As well, the swarm size employed in the MO-PSO matches the population size of the NSGA-II, with `maxIter` defined as $4,000$ for the MO-PSO and `numGen` defined as $4,000$ for the NSGA-II.

3 Numerical Results and Insights



(a) Total buffer allocation

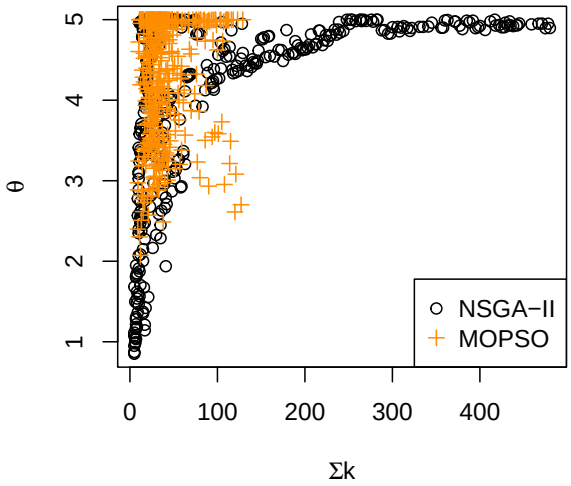


(b) Overall service rates

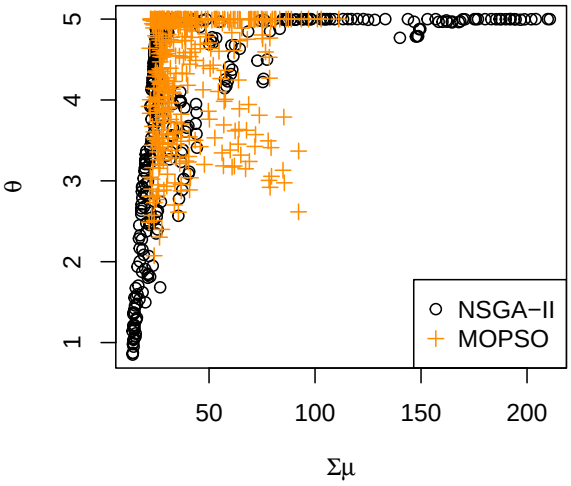
Figure 8. Solutions for a 3-node queueing network (series, $\mathrm{cv}^2 = 0.5$).

The presented results were obtained for a variety of network topologies (series, split, merge, and mixed, as shown in Figure 7), and different cv^2 (0.5, 1.0, and 1.5). These experiments were conducted with various network sizes (3, 5, 6, and 16 nodes), with the network size being the factor that exhibited the greatest influence on the solutions.

Figures 8 to 11 illustrate a comparison between typical solutions generated by the NSGA-II and MO-PSO, for various queueing network topological scenarios.



(a) Total buffer allocation



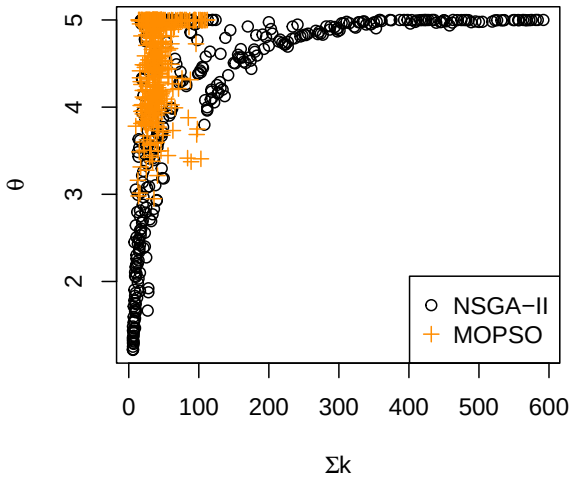
(b) Overall service rates

Figure 9. Solutions for a 5-node queueing network (series, $\mathrm{cv}^2 = 0.5$).

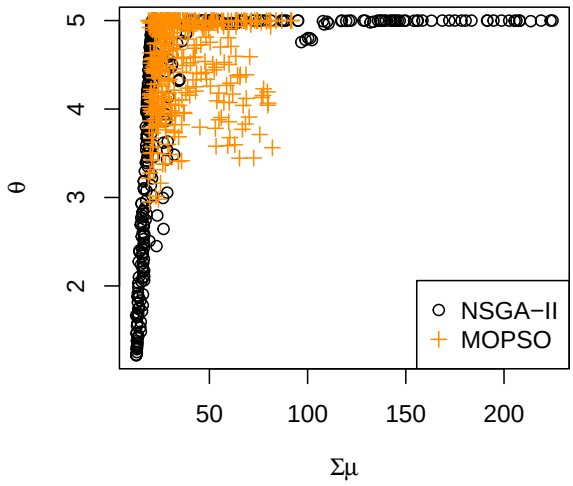
Each of these figures consists of two graphs. On the left, you can observe the throughput alongside the total buffer allocation, and on the right, the throughput of the overall service rates.

In the left graphs, it is evident that the MO-PSO consistently achieves higher throughput compared to the NSGA-II. This is demonstrated by the MO-PSO solutions being closer to the ordinate axis (Θ axis) than the NSGA-II solutions. Moreover, the total buffer allocations across the queueing network were notably lower, especially at high throughput.

Moving to the right-hand graphs, we conduct a similar analysis, focusing on overall service rates across the queueing network. Here, we observe that MO-PSO solutions exhibit similar resource-cost behavior for service rates as NSGA-II. However, MO-PSO consistently provides solutions with higher throughput and lower overall service rates than NSGA-II.



(a) Total buffer allocation



(b) Overall service rates

Figure 10. Solutions for a 6-node queueing network ($\mathrm{cv}^2 = 0.5$).

Finally, Figures 12 and 13 show 400 solutions obtained by running MO-PSO and NSGA-II, evaluated in terms of throughput.

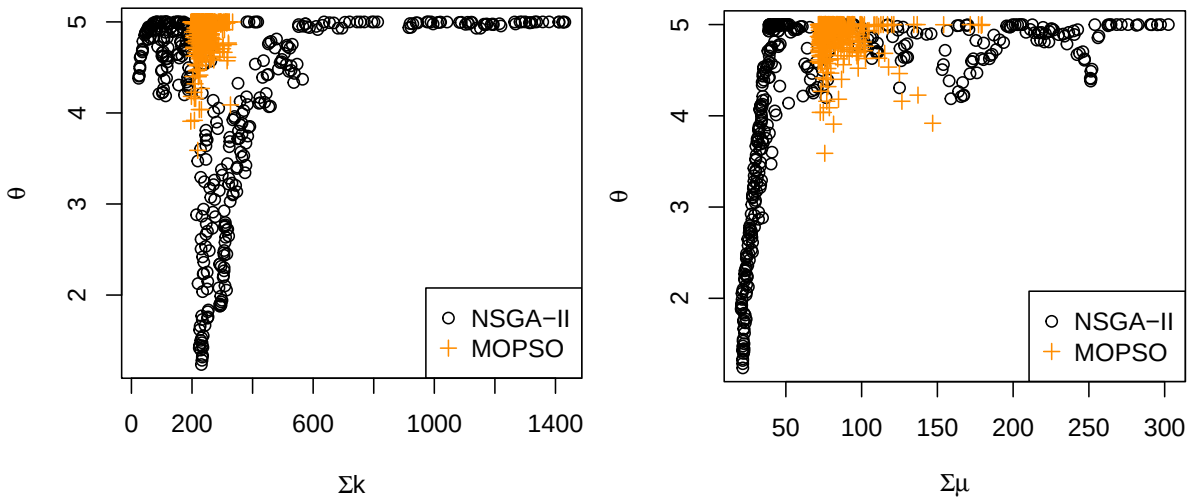
The x-axis, labeled 'index', represents the solutions ordered in ascending order based on their achieved throughput. It is evident that MO-PSO consistently outperforms NSGA-II, yielding higher-throughput solutions.

In addition to the qualitative results presented earlier, we conducted quantitative comparisons of the solutions.

Table 1 displays the results of the solutions obtained through NSGA-II and MO-PSO concerning the size of the queueing networks in terms of the number of nodes.

It is evident that for all types of queueing networks, the solutions generated by the MO-PSO algorithm outperformed those of NSGA-II. Specifically, MO-PSO solutions exhibited higher average throughput and required smaller allocations of capacity and service rate. Furthermore, these solutions displayed lower standard deviations, indicating not only that MO-PSO solutions are superior on average but also that they exhibit reduced variability (uncertainty).

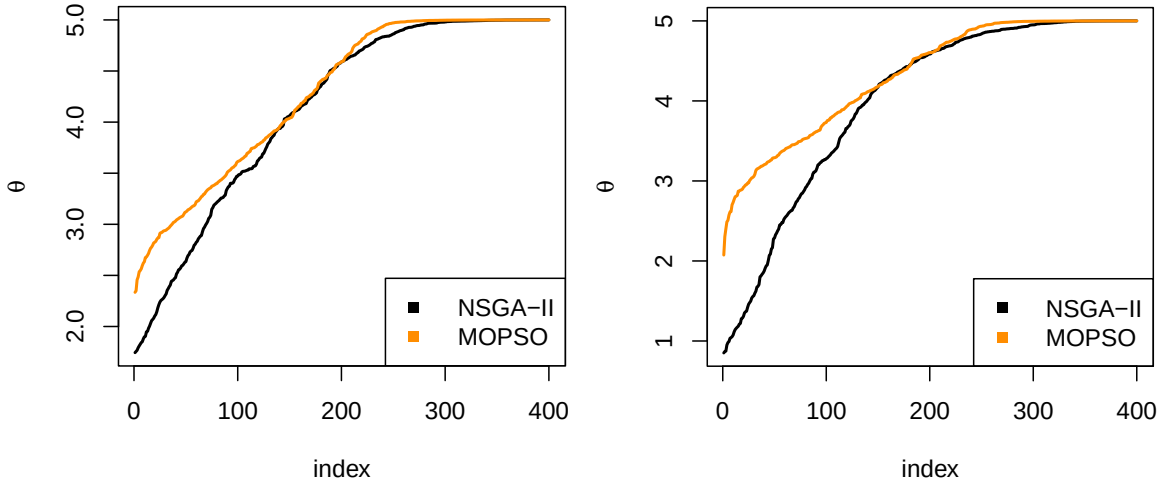
Table 2 presents several solutions in both the variable space



(a) Total buffer allocation

(b) Overall service rates

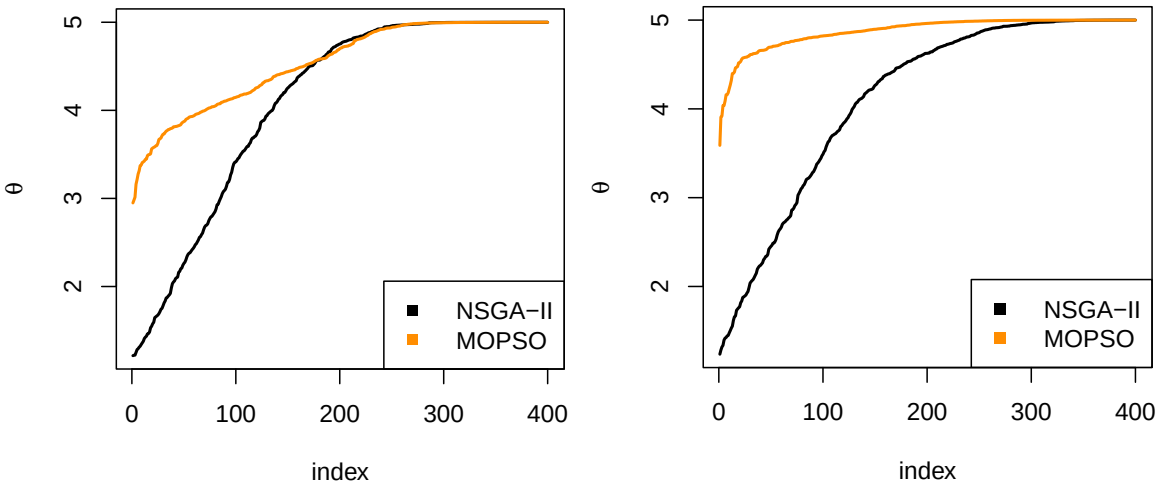
Figure 11. Solutions for a 16-node queueing network ($\mathrm{cv}^2 = 0.5$).



(a) 3-node queueing network (serie).

(b) 5-node queueing network (serie).

Figure 12. Solutions for queueing network ($\mathrm{cv}^2 = 0.5$), Throughput obtained.



(a) 6-node queueing network (mixed).

(b) 16-node queueing network (mixed).

Figure 13. Solutions for queueing network ($\mathrm{cv}^2 = 0.5$), Throughput obtained.

and objective space for series queues with varying cv^2 values. We selected non-dominated solutions based on two priority criteria: one that prioritizes overall service rates (priority μ_i, with $\sum k_i$ in descending order) and another that prioritizes

Table 1. Average performance of NSGA-II and MO-PSO. (Note: best results are in **bold**).

Number of nodes	Measure	NSGA-II			MO-PSO		
		Θ	$\sum k$	$\sum \mu$	Θ	$\sum k$	$\sum \mu$
3	average	4.2666	65.4000	30.1552	**4.3428**	**19.2725**	**28.9909**
	s.d.	0.8942	84.1356	34.8020	**0.7209**	**21.6376**	**23.9270**
5	average	3.9209	116.4333	47.6520	**4.2135**	**41.7117**	**44.2374**
	s.d.	1.2924	133.0205	49.0737	**0.8363**	**28.1054**	**22.5255**
6	average	4.0511	143.9458	42.9435	**4.4722**	**43.6950**	**39.2556**
	s.d.	1.2157	164.6712	48.6312	**0.5925**	**20.0907**	**19.0219**
16	average	4.0929	385.3467	97.9728	**4.8603**	**254.9808**	**79.5809**
	s.d.	1.0476	334.4950	90.8839	**0.2181**	**25.7793**	**18.9287**

total buffer allocation (priority k_i, with $\sum \mu_i$) in descending order. This approach allows researchers to prioritize one objective over the other or strike a balance between the two.

Initially, we observe a consistent pattern in service rate allocation, with higher rates assigned to the network's initial and final nodes (queues). This pattern is commonly referred to as the 'bowl effect'. When the priority is to optimize capacities, a similar pattern emerges for the k_i values. Given that the queues are in series, this outcome aligns with expectations and is promising.

Another noteworthy observation is the impact of increasing service rate variability. For hyperexponential systems ($\mathrm{cv}^2 = 1.5$), we observe the expected increase in resource requirements to achieve a certain throughput. This aligns with expectations, as higher variability in service rates necessitates additional resources to maintain performance. This underscores the importance of considering general queues rather than relying solely on Markovian approximations, especially when the required resource quantities (service rate or queue buffer) are underestimated, which can lead to suboptimal network throughput.

These are just a few of the analyses enabled by the results from MO-PSO. Equally insightful results could be obtained for networks with split, merge, or mixed configurations (results not shown). For instance, these networks often exhibit economies of scale, in which the service rate or node buffer at a split point is lower than the sum of the rates or capacities of the nodes that follow the split. Conversely, in merge networks, we observe the opposite effect: the merging node requires fewer allocated resources than the sum of the resources at the nodes prior to the merge.

Finally, we conducted a computational experiment to assess CPU time performance. We executed 50 runs of NSGA-II and MO-PSO, each set with parameters `numGen`=100 and `maxIter`=100. To put this in perspective, we previously conducted experiments with `numGen`=4,000 and `maxIter`=4,000, which made each run approximately 40 times slower. The main target of this investigation was to confront the CPU time requirements of the two algorithms. Figure 14 illustrates the average execution times from these 50 runs.

This experiment was repeated for queueing networks (series topology) with 2 to 16 nodes (queues) to check the evolution of CPU times. The experiment confirms that NSGA-II and MO-PSO do not require prohibitive computational times. The times taken by NSGA-II are slightly higher than those taken by MO-PSO. The graphical analysis shows that NSGA-II has a slightly exponential increase in the number of queues of the network. On the other hand, MO-PSO shows that the CPU time grows approximately linearly with the number of queues in the network.

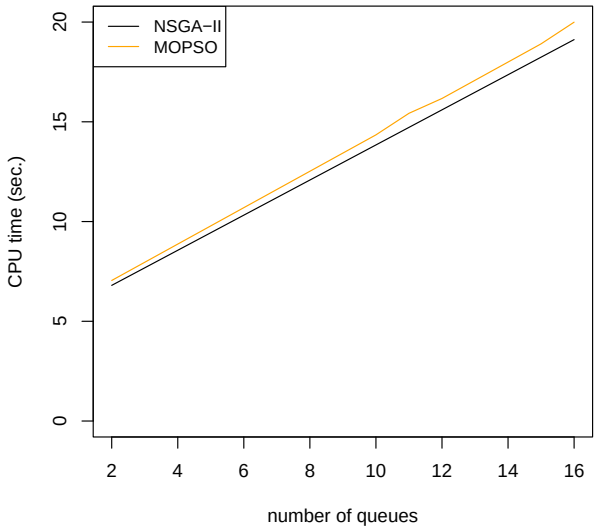


Figure 14. Average CPU time for 50 replications.

4 Conclusion and Final Remarks

Here, we describe a multi-objective particle swarm optimization algorithm (MO-PSO) developed to maximize network throughput for general single-server finite queues. The MO-PSO is coupled with the generalized expansion method to evaluate the performance of the queueing network and obtain approximate Pareto sets. It is adapted for an optimization problem with specific requirements. The combined strategy for generating the particles' initial positions and updating them is presented as a novelty in this work.

From the obtained solutions, it was possible to observe that a trade-off is reached between resource allocation (total

Table 2. Some nondominated solutions for 3-node series queueing networks.

		Decision variables		Objective space		
cv^2	Priority	(k_1, k_2, k_3)	(μ_1, μ_2, μ_3)	Θ	$\sum k_i$	$\sum \mu_i$
0.5	μ_i	(30, 49, 36)	(6.6124, 7.1083, 6.4078)	5.0000	115	20.1285
		(26, 33, 42)	(5.3332, 5.0186, 5.4649)	4.9797	101	15.8167
		(32, 23, 35)	(7.8532, 7.5925, 6.2181)	4.9996	90	21.6638
		(22, 25, 33)	(7.8237, 7.3573, 6.5133)	4.9328	80	21.6942
		(24, 24, 22)	(6.2017, 6.3245, 6.0772)	4.9979	70	18.6034
		(20, 17, 26)	(6.5855, 6.2001, 5.8261)	4.9912	63	18.6117
		(08, 31, 11)	(7.4785, 5.4574, 6.9282)	5.0000	50	19.8641
		(15, 11, 14)	(7.3954, 6.8143, 6.3954)	5.0000	40	20.6050
		(05, 13, 12)	(8.3870, 7.5205, 8.6679)	4.9976	30	24.5755
		(06, 10, 04)	(6.7477, 5.3985, 7.2274)	4.9999	20	19.3736
	k_i	(4, 5, 5)	(49.9398, 45.2361, 33.5937)	5.0000	14	128.7696
		(5, 4, 4)	(27.4333, 42.2275, 31.2565)	4.9997	13	100.9173
		(3, 4, 4)	(37.3310, 25.1066, 28.4142)	4.9971	11	90.8518
		(4, 4, 3)	(25.7844, 27.6134, 28.8907)	4.9858	11	82.2885
		(3, 4, 4)	(25.0001, 23.9007, 22.0889)	4.9949	11	70.9897
		(4, 3, 3)	(12.4656, 30.9877, 20.3698)	4.9474	10	63.8231
		(4, 5, 6)	(24.2847, 10.5510, 15.7831)	4.9805	15	50.6188
		(4, 6, 9)	(21.2812, 10.6808, 8.1724)	4.9999	19	40.1344
		(3, 4, 4)	(9.6435, 13.1736, 8.8215)	4.9752	11	31.6385
		(5, 7, 6)	(6.8400, 7.0270, 6.5932)	5.0000	18	20.4601
1.5	μ_i	(62, 38, 37)	(6.6301, 7.0718, 6.5716)	4.9998	137	20.2735
		(31, 49, 48)	(6.6629, 5.9591, 6.2057)	4.9999	128	18.8278
		(37, 48, 33)	(7.4494, 6.4559, 7.3052)	4.9963	118	21.2105
		(40, 26, 44)	(6.0277, 6.597, 6.1574)	4.9978	110	18.7821
		(23, 34, 42)	(7.9702, 6.8084, 6.6635)	4.9999	99	21.4421
		(18, 30, 42)	(8.35, 7.1127, 6.0131)	4.9995	90	21.4758
		(21, 34, 26)	(6.5858, 6.8856, 6.0525)	4.9996	81	19.5239
		(24, 17, 30)	(6.4903, 8.0076, 6.1354)	4.9848	71	20.6333
		(25, 14, 21)	(6.8088, 7.4083, 11.1408)	4.9604	60	25.3580
		(13, 10, 17)	(10.1766, 12.3478, 9.1402)	5.0000	40	31.6646
	k_i	(5, 7, 5)	(49.8954, 32.3009, 48.7662)	5.0000	17	130.9625
		(5, 5, 4)	(43.3624, 37.3563, 38.8292)	4.9985	14	119.5480
		(4, 5, 5)	(41.0926, 41.0954, 27.4999)	4.9796	14	109.688
		(3, 3, 4)	(31.2554, 36.1424, 35.4082)	4.9999	10	102.806
		(4, 6, 5)	(32.9313, 18.3520, 38.3947)	4.9996	15	89.6780
		(6, 7, 6)	(24.6307, 27.7002, 27.5599)	4.9999	19	79.8908
		(4, 5, 5)	(23.9200, 22.5130, 22.5378)	4.9994	14	68.9708
		(4, 5, 6)	(21.3278, 20.1988, 19.0922)	4.9998	15	60.6188
		(5, 5, 5)	(19.7623, 11.5747, 19.2951)	4.9901	15	50.6321
		(2, 4, 4)	(16.3560, 14.9944, 10.8123)	4.9960	10	42.1627

buffer and service rates) and throughput to yield efficient solutions, that is, with low implementation costs and high throughput. Notably, we verified that the MO-PSO can provide a lower-investment solution than the NSGA-II, *i.e.*, with lower buffer allocation and overall service rates. Discussions of the results are not restricted to the objective space. Analyzing the variables space allows conclusions about the structure of the solutions provided by the MO-PSO. Furthermore, these considerations enable evaluations of queueing network deployments by informing decisions about the feasibility of investing in buffers and service rates. Different topologies were tested to confirm the generality of such findings.

Future investigations can be conducted to optimize the throughput in networks with finite general multiserver queues or cyclic queues *e.g.*, networks with feedback for rework loops. Furthermore, future research can evaluate the MO-PSO in real-life situations.

Declarations

Funding

This work was funded by the Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq, grant 307151/2022-0, 305442/2022-8), Fundação de Amparo à Pesquisa do Estado de Minas Gerais (FAPEMIG, grant APQ-01647-22, PPM-00564-17).

Authors' Contributions

GLS: Conceptualization, Validation, Formal Analysis, Investigation, Writing – Original Draft, Visualization. **ARD:** Conceptualization, Methodology, Writing – Review & Editing, Supervision. **FRBC:** Conceptualization, Methodology, Writing – Review & Editing, Supervision. **GJPM:** Conceptualization, Writing – Review & Editing, Supervision. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Almehdawe, E., Jewkes, B., and He, Q.-M. (2019). Optimization in a two-stage multi-server service system with customer priorities. *Journal of the Operational Research Society*, 70(2):326–337. DOI: 10.1080/01605682.2018.1438762.

Coello Coello, C. A. and Lechuga, M. S. (2002). MOPSO: A proposal for multiple objective particle swarm optimization. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No.02TH8600)*, volume 2, pages 1051–1056. DOI: 10.1109/CEC.2002.1004388.

Cruz, F. R. B. (2009). Optimizing the throughput, service rate, and buffer allocation in finite queueing networks. *Electronic Notes in Discrete Mathematics*, 35:163–168. LAGOS'09 - V Latin-American Algorithms, Graphs and Optimization Symposium. DOI: 10.1016/j.endm.2009.11.028.

Cruz, F. R. B., Duarte, A. R., and Souza, G. L. (2018). Multi-objective performance improvements of general finite single-server queueing networks. *Journal of Heuristics*, 24(5):757–781. DOI: 10.1007/s10732-018-9379-8.

Cruz, F. R. B., Duarte, A. R., and van Woensel, T. (2008). Buffer allocation in general single-server queueing networks. *Computers & Operations Research*, 35(11):3581–3598. DOI: 10.1016/j.cor.2007.03.004.

Cruz, F. R. B., Kendall, G., While, L., Duarte, A. R., and Brito, N. C. L. (2012). Throughput maximization of queueing networks with simultaneous minimization of service rates and buffers. *Mathematical Problems in Engineering*, 2012(Article ID 692593):19 pages. DOI: 10.1155/2012/692593.

Cruz, F. R. B., van Woensel, T., and MacGregor Smith, J. (2010). Buffer and throughput trade-offs in $M/G/1/K$ queueing networks: A bi-criteria approach. *International Journal of Production Economics*, 125(2):224–234. DOI: 10.1016/j.ijpe.2010.02.017.

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. DOI: 10.1109/4235.996017.

Duarte, A. R. (2024). The server allocation problem for Markovian queueing networks. *International Journal of Services and Operations Management*, 48(2):256–271. DOI: 10.1504/IJSOM.2024.138931.

Fan, Z., Wang, T., Cheng, Z., Li, G., and Gu, F. (2017). An improved multiobjective particle swarm optimization algorithm using minimum distance of point to line. *Shock and Vibration*, 2017:1–16. DOI: 10.1155/2017/8204867.

Gross, D., Shortle, J. F., Thompson, J. M., and Harris, C. M. (2009). *Fundamentals of queueing theory*. Wiley - Interscience, New York, NY, fourth edition. DOI: 10.1002/9781119453765.

Hernández-Vázquez, J. O., Hernández-González, S., Jiménez-García, J. A., Hernández-Ripalda, M. D., and Hernández-Vázquez, J. I. (2019). Enfoque híbrido metaheurístico AG-RS para el problema de asignación del buffer que minimiza el inventario en proceso en líneas de producción abiertas en serie. *Revista Iberoamericana de Automática e Informática Industrial*, 16(4):447–458. DOI: 10.4995/riai.2019.10883.

Ingolfsson, A., Almehdawe, E., Pedram, A., and Tran, M. (2020). Comparison of fluid approximations for service systems with state-dependent service rates and return probabilities. *European Journal of Operational Research*, 283(2):562–575. DOI: 10.1016/j.ejor.2019.11.041.

Inzillo, V., De Rango, F., and Quintana, A. A. (2019). A self clocked fair queuing MAC approach limiting deafness and round robin issues in directional MANET. In *2019 Wireless Days (WD)*, pages 1–6. IEEE. DOI: 10.1109/WD.2019.8734263.

Kassoul, K., Cheikhrouhou, N., and Zufferey, N. (2022). Buffer allocation design for unreliable production lines using genetic algorithm and finite perturbation analysis. *International Journal of Production Research*, 60(10):3001–3017. DOI: 10.1080/00207543.2021.1909169.

Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of ICNN'95 - International Conference on Neural Networks*, volume 4, pages 1942–1948. DOI: 10.1109/ICNN.1995.488968.

Kerbache, L. and MacGregor Smith, J. (1987). The generalized expansion method for open finite queueing networks. *European Journal of Operational Research*, 32:448–461. DOI: 10.1016/S0377-2217(87)80012-7.

Kerbache, L. and MacGregor Smith, J. (1988). Asymptotic behavior of the expansion method for open finite queueing networks. *Computers & Operations Research*, 15(2):157–169. DOI: 10.1016/0305-0548(88)90008-1.

Khalid, R., Nawawi, M. K. M., Kawsar, L. A., Ghani, N. A., Kamil, A. A., and Mustafa, A. (2020). Optimal routing of pedestrian flow in a complex topological network with multiple entrances and exits. *International Journal of Systems Science*, 51(8):1325–1352. DOI: 10.1080/00207721.2020.1756524.

Kimura, T. (1996). A transform-free approximation for the finite capacity $M/G/s$ queue. *Operations Research*, 44(6):984–988. DOI: 10.1287/opre.44.6.984.

Kose, S. Y. and Kilincci, O. (2020). A multi-objective hybrid evolutionary approach for buffer allocation in open serial production lines. *Journal of Intelligent Manufacturing*, 31(1):33–51. DOI: 10.1007/s10845-018-1435-6.

Liu, J., Hu, L., Xu, X., and Wu, J. (2021). A queuing network simulation optimization method for coordination control of passenger flow in urban rail transit stations. *Neural Computing and Applications*, 33(17):10935–10959. DOI: 10.1007/s00521-020-05580-5.

MacGregor Smith, J. (2003). $M/G/c/k$ blocking probability models and system performance. *Performance Evaluation*, 52(4):237–267. DOI: 10.1016/S0166-5316(02)00190-6.

MacGregor Smith, J. (2004). Optimal design and performance

modelling of $M/G/1/k$ queueing systems. *Mathematical and Computer Modelling*, 39(9-10):1049–1081. DOI: 10.1016/S0895-7177(04)90534-1.

MacGregor Smith, J. (2015). Optimal workload allocation in closed queueing networks with state dependent queues. *Annals of Operations Research*, 231(1):157–183. DOI: 10.1007/s10479-013-1418-0.

MacGregor Smith, J. (2018). Simultaneous buffer and service rate allocation in open finite queueing networks. *IISE Transactions*, 50(3):203–216. DOI: 10.1080/24725854.2017.1300359.

MacGregor Smith, J. and Cruz, F. R. B. (2005). The buffer allocation problem for general finite buffer queueing networks. *IIE Transactions*, 37(4):343–365. DOI: 10.1080/07408170590916986.

Martins, H. S. R., Cruz, F. R. B., Duarte, A. R., and Oliveira, F. L. P. (2019). Modeling and optimization of buffers and servers in finite queueing networks. *OPSEARCH*, 56(1):123–150. DOI: 10.1007/s12597-019-00362-7.

Pourjavad, E. and Almehdawe, E. (2022). Optimization of the technician routing and scheduling problem for a telecommunication industry. *Annals of Operations Research*, 315(1):371–395. DOI: 10.1007/s10479-022-04658-8.

Souza, G. L., Duarte, A. R., Moreira, G. J. P., and Cruz, F. R. B. (2020). A novel formulation for multi-objective optimization of general finite single-server queueing networks. In *IEEE Congress on Evolutionary Computation (CEC)*, pages 1–8. DOI: 10.1109/CEC48606.2020.9185827.

Souza, G. L., Duarte, A. R., Moreira, G. J. P., and Cruz, F. R. B. (2023). Post-processing improvements in multi-objective optimization of general single-server finite queueing networks. *IEEE Latin America Transactions*, 21(3):381–388. DOI: 10.1109/TLA.2023.10068841.

Trivedi, V., Varshney, P., and Ramteke, M. (2020). A simplified multi-objective particle swarm optimization algorithm. *Swarm Intelligence*, 14(2):83–116. DOI: 10.1007/s11721-019-00170-1.

Xi, S., MacGregor Smith, J., Chen, Q., Mao, N., Zhang, H., and Yu, A. (2022). Simultaneous machine selection and buffer allocation in large unbalanced series-parallel production lines. *International Journal of Production Research*, 60(7):2103–2125. DOI: 10.1080/00207543.2021.1884306.

Zennaro, I., Finco, S., Aldrighetti, R., and Battini, D. (2022). Buffer size evaluation in a bottle plant production system: a comparison between different solving methods. *International Journal of Services and Operations Management*, 42(4):500–524. DOI: 10.1504/IJSOM.2022.124990.

Zhao, X., Jin, Y., Ji, H., Geng, J., Liang, X., and Jin, R. (2013). An improved mixed-integer multi-objective particle swarm optimization and its application in antenna array design. In *5th IEEE International Symposium on Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications*, pages 412–415. DOI: 10.1109/MAPE.2013.6689835.