Pesquisa Operacional (2020) 40: e220764 p.1-22
doi: 10.1590/0101-7438.2020.040.00220764

Printed version ISSN 0101-7438 / Online version ISSN 1678-5142
www.scielo.br/pope
ARTICLES

IMPROVING THE SHIFT-SCHEDULING PROBLEM USING NON-STATIONARY QUEUEING MODELS WITH LOCAL HEURISTIC AND GENETIC ALGORITHM

Caio Vitor Beojone[1*] and Regiane Máximo de Souza[2]

Received March 4, 2019 / Accepted October 27, 2019

ABSTRACT. We improve the shift-scheduling process by using nonstationary queueing models to evaluate schedules and two heuristics to generate schedules. Firstly, we improved the fitness function and the initial population generation method for a benchmark genetic algorithm in the literature. We also proposed a simple local search heuristic. The improved genetic algorithm found solutions that obey the delay probability constraint more often. The proposed local search heuristic also finds feasible solutions with a much lower computational expense, especially under low arrival rates. Differently from a genetic algorithm, the local search heuristic does not rely on random choices. Furthermore, it finds one final solution from one initial solution, rather than from a population of solutions. The developed local search heuristic works with only one well-defined goal, making it simple and straightforward to implement. Nevertheless, the code for the heuristic is simple enough to accept changes and cope with multiple objectives.

Keywords: nonstationary queues, genetic algorithm, local search heuristic.

1 INTRODUCTION

Emergency service systems (ESSs) still see workforce scheduling as a challenge and have even been studied within operations research/management science for more than fifty years (Green & Kolesar, 2004; Simpson & Hancock, 2009). The main objective of workforce scheduling is to minimize operational costs while maintaining adequate service level. However, the daily operations of ESSs are surrounded by randomness and require studies to consider variations in the morning, afternoon, and night operations (Schmid, 2012; Souza et al., 2015).

Due to the nature of the events they answer, ESSs usually must have a low workload through the day. This is a characteristic of systems with a low event frequency (Chiyoshi et al., 2011). The literature is thin for such systems. The majority of the literature studies high event frequency systems such as call centers.

*Corresponding author.
[1]Department of Production Engineering, São Paulo State University – UNESP, Bauru, Brazil, 17033-360 – E-mail: caio.beojone@epfl.ch – https://orcid.org/0000-0002-6491-7104
[2]Department of Production Engineering, São Paulo State University – UNESP, Bauru, Brazil, 17033-360 – E-mail: regiane@feb.unesp.br – https://orcid.org/0000-0002-4695-2678

The majority of the queueing models used on employee scheduling are stationary or piecewise constant (Green et al., 2001). This approximation creates distortions related to the user arrival process and, more rarely, to the service process (Kim & Whitt, 2014). There are also distortions to model the process of servers leaving the system at the end of their shifts. (Ingolfsson et al., 2007).

The use of time-dependent queueing models makes employee scheduling a larger but more realistic challenge by revealing its nonlinear and dynamic behavior. These characteristics are associated with problems involving difficult-to-find optimal solutions, so the use of heuristics becomes necessary, as raising server requirements is often unrealistic (Ingolfsson et al., 2010). We can enumerate some efforts on how the genetic algorithm is used (Ingolfsson et al., 2002), as an integer programming heuristic (Ingolfsson et al., 2010), and as the heuristic used in Chung and Min (2014).

Shift-scheduling problems still face limitations. As mentioned before, the use of stationary approximations is extremely common (Defraeye and Van Nieuwenhuyse, 2016). Also, Ingolfsson et al. (2002, 2010) pointed out that improved heuristics with lower computational efforts are desired, as the use of genetic algorithm is computationally expensive. Also, Feldman et al. (2008) and Defraeye and Van Nieuwenhuyse (2016) stated that systems with low arrival rates are a challenge for many optimization methods, turning small-scale systems into avenues for further research. Finally, Defraeye and Van Nieuwenhuyse (2016) indicated that the development of easy-to-compute problems could benefit the study of more complex performance metrics and their relations in queueing systems.

Hence, this paper aims to improve the shift-scheduling process revisiting the police precinct study from Kolesar et al. (1975) and Ingolfsson et al. (2002). Firstly, the genetic algorithm used in Ingolfsson et al. (2002) is analyzed regarding its convergence, fitness function, number of iterations, and computational times. The purpose of the genetic algorithm here is to determine how many servers will work on each possible shift. This answer must satisfy all constraints and have adequate cost and delay probability. This application of the genetic algorithm for shift-scheduling problems is based on the application found in Ingolfsson et al. (2002). We also propose a new fitness function and initial population generation method based on previous results, to make the genetic algorithm more reliable. A local search heuristic is also proposed for the same purpose. Such a simple heuristic is passed through an initial test, and its results are compared to that of the genetic algorithm through a sensitivity analysis decreasing and increasing arrival rates in a small-scale system. Therefore, this paper contributes to the literature by improving the genetic algorithm for shift scheduling problems, by presenting a simple and fast local search heuristic, and by evaluating these methods in a benchmark case study.

The remaining paper is structured as follows. Section 2 presents the related literature to the topic. Section 3 presents the methodology of our study, including: data used for the New York police precinct; the queueing model and its performance measures used to evaluate schedules; the base genetic algorithm and the proposed improvements; and the proposed local search heuristic. Section 4 shows the results of the proposed methods in the benchmark case and a brief analysis

about both the genetic algorithm and proposed local search heuristic under demand changes. Finally, Section 5 shows some final considerations about this study and perspectives for further research.

2 RELATED LITERATURE

Dantzig (1954) and Edie (1954) mark the advent of the personal scheduling problems back in the 1950s. Firstly, Edie (1954) collected data on tolls to obtain hourly personnel requirements and suggested that a linear programming problem could make use of such requirements to calculate an optimal shift-scheduling. The calculations for the requirements used stationary queueing models as the M/M/s (Erlang C) for each day hour. Green et al. (2001) called this approach as Stationary Independent Period-by-Period (SIPP). Dantzig (1954) presented an integer programming solution for Edie's (1954) suggestion. The combined formulation of Dantzig (1954) and Edie (1954) are known as SIPP-IP (SIPP solved with Integer Programing) method (Ingolfsson et al., 2002).

Baker (1976) classified personal scheduling in three categories: shift-scheduling, days-off scheduling, and tour-scheduling (combination of the first two categories). Our focus is on shift-scheduling problem, where planning considers a daily planning horizon for the scheduling problem (Van den Bergh et al., 2013).

The literature on shift-scheduling evolved a lot since its advent in the 1950s. A quick search in a Scopus database revealed 121 published papers about shift-scheduling in the last 20 years. Half of them were published after 2012. Preferred journals include (but are not limited to) the European Journal of Operational Research, Journal of Scheduling, Computers and Operations Research, and Annals of Operations Research.

Building a shift schedule has some steps (Ingolfsson et al. 2002). One can find papers on the literature with different orders and method to get the schedules (see Thompson (1997) and the references therein). Typically, it is necessary to (Buffa et al., 1976):

1. Forecast period-by-period demand,
2. Convert demand forecasts into staffing requirements,
3. Determine set of possible shifts,
4. Find a shift schedule that obey all requirements raised and minimize a goal as number of employees or labor costs.

Demand forecasts are made using historical data (Gillard and Knight 2014; Angelo et al., 2017); converting calls (arrivals) into arrival rates obeying a nonhomogeneous Poisson process (Ingolfsson et al. 2002). Kim and Whitt (2014) shows statistical tests to evaluate if an arrival rate is well-modeled using nonhomogeneous Poisson processes.

The requirements often respect a service level measure, which we consider as the probability that there is going to be at least one idle server when a new user arrives in the system. Gillard and Knight (2014) uses Singular Spectrum Analysis (a technique to decompose a time series into oscillatory components and noise) to generate demand forecasts and requirements because they consider that other means do not capture seasonal variations properly. Defraeye and Van Nieuwenhuyse (2016) and Schwarz et al. (2016) presented comprehensive literature reviews on this topic.

Possible shifts depend on the problem modeled. For instance, typically, industrial companies have non-overlapping shifts (Van den Bergh et al., 2013). Most of the problems are related to systems with a great number of events, such as call centers (Brown et al., 2005; Dietz, 2011; Kim & Ha, 2012). Some other papers are related to hospital emergency units (Creemers et al., 2012; Green & Kolesar, 1991; Green, et al., 2001; Gillard & Knight, 2014; Sungur et al., 2017). One can find other services in Ko and Gautam (2010), Chung and Min (2014), Thompson (1993, 1997), and Jennings et al. (1996). Among papers about ESS, one can cite Kolesar et al. (1975), using data from a police precinct.

As the problems themselves, finding a solution (schedule) that obeys all requirements and optimizes the goal is a great challenge and many methods arise in the literature. Alfares (2004) categorized solutions of scheduling problems in 10 groups: manual solutions, integer programing, implicit modeling, decomposition, goal programming, working set generation, linear-programing based solution, construction/improvement, metaheuristics, and others.

Ingolfsson et al. (2002) developed a method for workforce scheduling. The approach modeled the structure of possible shifts and the dynamic nature of demand. A genetic algorithm solved a benchmark case (shown in Section 3.1) and a numerical solution of non-stationary queueing system evaluated service levels. The authors found that the SIPP-IP method can significantly overestimate the service level that results from a schedule.

The non-stationary queueing model used in Ingolfsson et al. (2002) is a set of the Chapman-Kolmogorov equations (Kleinrock, 1975). However, Ingolfsson (2005) and Ingolfsson et al. (2007) pointed flaws on the direct use of Chapman-Kolmogorov equations without considering what happens the moment a busy server is scheduled to leave. The literature identified two possibilities for this moment, which are called end-of-shift disciplines. Firstly, the service is preempted and the user rejoins the head of the queue (preemptive discipline). Secondly, the server completes the job before leaving (exhaustive discipline). Ingolfsson (2005) points that the exhaustive discipline is often more realistic when users are humans. However, the exhaustive discipline still requires more advances to represent mealtime breaks (Beojone, 2017). Those interested on non-stationary queueing models may find detailed explanation for them and their numerical solutions in Schwarz et al. (2016), Defraeye and Van Nieuwenhuyse (2016) and the references therein.

3 METHODOLOGY

This section presents the studied case, a benchmark problem from the literature. It also presents the schedule evaluator (non-stationary queueing model). Further in this section we present the genetic algorithm and the proposed improvements for a shift-scheduling problem. Finally, we present the proposed local search heuristic.

A portion of our study is inspired on Ingolfsson et al. (2002). Therefore, it is imperative to clarify the points of convergence between this paper and Ingolfsson et al. (2002). Firstly, we used the same benchmark problem from New York's police. Secondly, we used the same queueing model. Finally, we use a similar construction for the genetic algorithm, which we improve in the next subsections. Besides the improvements on the genetic algorithm, this paper also contributes with a new local search heuristic, compared to Ingolfsson et al. (2002).

3.1 Data and studied case

Kolesar et al. (1975) studied a police precinct in New York to develop a shift-scheduling problem based on stationary queueing models. Ingolfsson et al. (2002) revisited this case to implement a shift-schedule problem using a genetic algorithm, which is explained in more detail ahead. Table 1 summarizes the main information about the New York police case seen in Kolesar et al. (1975) that will be used in our study, such as the arrival rates and the working hours for each possible tour. Note that there are three major shifts, with four mealtime breaks between 2 and 5 hours after the shift's beginning, for a total of 12 possible shifts, as used in Ingolfsson et al. (2002). Service time is considered constant throughout the day, with a constant rate μ equal to 2 calls/hour. This information is the same as used in both Kolesar et al. (1975) and Ingolfsson et al. (2002).

3.2 Schedule evaluator (non-stationary queueing model)

According to Gillard and Knight (2014), it is usual to ESSs to operate under stochastic conditions, so it is absolutely important to be able to plan for such situations. Also, these stochastic conditions are time-varying, and it is reasonable to approximate an ESS with a $M(t)/M/s(t)$ queueing model. For those unfamiliar with Kendall's notation, $M(t)/M/s(t)$ represents a queueing system with a nonhomogeneous Poisson arrival process $M(t)$, a negative exponential service time M, and a time-varying number of working servers $s(t)$. Arrival and service processes are represented by the $\lambda(t)$ and μ rates, respectively, in calls/hour.

As servers have defined shifts, it is necessary to define what happens in case a server is providing service the moment he is scheduled to leave. The behavior assumed by servers in this circunstance is called end-of-shift discipline. As Ingolfsson et al. (2010) did, we adopted preemptive end-of-shift discipline to facilitate the comparison of our computational results. Hence the non-stationary queueing model used in this paper is the same as the one seen in Ingolfsson et al. (2002), Green and Soares (2007), and Ingolfsson et al. (2010). The remaining of this subsection presents a brief explanation on the queueing model.

Table 1 – Summary of arrival rates and working hours of each tour for the New York police case.

t	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
$\lambda(t)$	9.8	9.6	8.7	7.6	6.7	5.3	4.1	3.2	2.5	2.5	2.9	3.8	4.3	5.0	5.9	6.6	7.8	8.6	9.4	9.8	10.2	10.4	10.2	10.0
Tour presence	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1	0	0	0	0	0	0	0	0
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	1	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	0	1	1	1	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	0	1	1	1
	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	0	1	1

Let $\pi_n(t)$ denote the transient state probability that n customers are in the system at time t. We can write the Chapman–Kolmogorov forward differential equations for a time-dependent multiserver queue as seen in Equation (1) to describe the exact behavior of such a system. The Chapman–Kolmogorov forward differential equations were solved, numerically, using the Runge–Kutta method.

$$\begin{aligned}
&\pi_0'(t) = -\lambda(t)\pi_0(t) + \mu\pi_1(t) \\
&\pi_n'(t) = -(\lambda(t) + n\mu)\pi_n(t) + \lambda(t)\pi_{n-1}(t) + (n+1)\mu\pi_{n+1}(t), for n = 1,2,\ldots s(t)-1 \\
&\pi_n'(t) = -(\lambda(t) + s(t)\mu)\pi_n(t) + \lambda(t)\pi_{n-1}(t) + s(t)\mu\pi_{n+1}(t), for n = s(t), s(t)+1,\ldots,K-1 \\
&\pi_K'(t) = -s(t)\mu\pi_K(t) + \lambda(t)\pi_{K-1}(t)
\end{aligned} \tag{1}$$

In this paper, nonstationary queueing models are firstly aimed at calculating instantaneous delay probabilities and hourly mean delay probabilities. To do so, we define delay probability $\pi_D(t)$ as the probability that an arriving new user must wait before being served. Our objective is to guarantee a lower $\pi_D(t)$ than a given probability P for all values of time t. Equation (2) shows the delay probability as calculated in nonstationary queueing models.

$$\pi_D(t) = 1 - \sum_{i=0}^{s(t)-1} \pi_i(t) \tag{2}$$

We also need to calculate the hourly mean service level, defined as follows in Equation (3):

$$\acute{\pi}_D(t) = \int_t^{t+1} \lambda(t)\pi_D(t)\,dt \Big/ \int_t^{t+1} \lambda(t)\,dt \tag{3}$$

As $\lambda(t)$ is constant on the interval $[t, t+1)$, $t \in \{0,1,2,3,\ldots,23\}$, we can simplify the Equation (3) into Equation (4):

$$\acute{\pi}_D(t) = \int_t^{t+1} \pi_D(t)\,dt, \; for\; t \in \{0,1,2,\ldots,23\} \tag{4}$$

We use the schedule evaluator (non-stationary queueing model) to support the decision process of both heuristics from subsections 3.3 (genetic algorithm) and 3.4 (local search).

3.3 Genetic Algorithm

Many characteristics of shift-scheduling problems lead us to the use of heuristics to solve them, such as the size and complexity of the search space and the non-linear behavior of delay probabilities, along with its discontinuities. We started this application with a standard genetic algorithm.

Goldberg (1989) presented the genetic algorithm. Its main idea is based on Darwin's evolutionary theory, so it needs a population of individuals that will compete to select the most adapted individual. Each individual might be selected to breed with another one. The most adapted individuals will have higher chances to breed and will thus pass its characteristics to the next generations. The least adapted individuals will have smaller chances to breed and will thus tend to disappear. Individuals are also susceptible to random mutations, altering their chances to survive and breed (for more information, see Goldberg, 1989).

The algorithm uses chromosomes to represent individuals, and each individual is a potential solution for the problem. The most adapted individual to have passed through the population at any time will be chosen as the final solution. These solutions are encoded in the form of binary vectors, as shown in Figure 1. With a fixed number of individuals (called the population), a new generation is created as an offspring of the previous population (no members of the previous generation are included in the next generation). To select individuals (chromosomes – Figure 1) for breeding, we measure their adaptation by calculating a fitness function. Then, two distinct individuals are selected to breed with replacement. Higher fitness leads to higher chances of being selected. Regarding the breeding process, selected individuals have their chromosomes pass through a process called crossover, in which their chromosomes are combined to create two new chromosomes (individuals). After crossover, these new chromosomes may suffer "mutation" at any point, where a bit may change from 0 to 1 and vice-versa. When a new generation emerges, we have an iteration, and the selection process begins once more until a number of iterations is reached or another stop criterion becomes true.

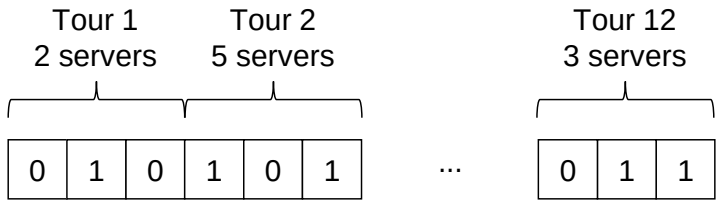

Figure 1 – Example of a chromosome encoded as a binary vector.
Based on Figure 2 from Ingolfsson et al. (2002, p. 591).

The seed for the initial population is the solution of the scheduling problem using SIPP method and integer programing (Dantzig, 1954). This solution method is referred in the remaining of the paper as "SIPP-IP method" and the solution itself (shift schedule for servers) from this method is referred as "SIPP-IP solution".

The process for generating the initial population used here was proposed by Ingolfsson et al. (2002). The initial population size is 60 individuals. The initial population was generated starting from the SIPP-IP solution. The SIPP-IP was one of the members of the initial population, and the remaining 59 individuals were generated randomly as follows: (1) we calculated the mean and variance of the SIPP-IP solution (number of servers in each shift is the piece of data for this calculation); (2) we sampled from a normal distribution with the calculated mean and variance

for each shift (defining new shift schedule for all shifts); and (3) sampled numbers were rounded to the nearest integer between 0 and 7 (limits of the adopted chromosome binary encoding; see Figure 1). The mutation probability for each bit was set to 1%, and the maximum number of iterations was set to 100.

Minimizing labor costs here means fewer vehicles, and maintaining an adequate service level means to keep the delay probability lower than 10%. Thus, our fitness functions were built using the same three parameters as in Ingolfsson et al. (2002): (1) CARS, or the sum of all vehicles from each tour; (2) MAXPROB, or the maximum delay probability along the time horizon; and (3) NUMHOURS, or the number of hours with a higher delay probability than the desired 10%.

The first fitness function used here is based on the third fitness function presented in Ingolfsson et al. (2002), and it is presented as Equation (5). M indicates a number large enough that the fitness function will not become negative. This fitness function is referred as FITNESS0 in the remainder of the paper. The second stop criterion (besides the number of iterations) for FITNESS0 was a difference between the fitness of the current population's best individual and average fitness of the population lower than 0.5.

$$FITNESS\,0 = M - (CARS * (1 + MAXPROB) + NUMHOURS * MAXPROB) \tag{5}$$

The fitness values of individuals were linearly scaled to ensure that the average fitness value would not change (Ingolfsson et al., 2002). Hence, the selection of individuals for breeding used the scaled value. The purpose of scaling fitness values was to avoid premature convergence during the first iterations and to avoid a random walk among the mediocre individuals in late iterations (Goldberg, 1989). Scaling process is also useful to avoid some bias caused by using a value of M in the fitness function.

3.3.1 FITNESS function choice

However, FITNESS0 puts too much attention to MAXPROB, which cannot indicate whether the service level constraint is respected or not. In the other hand, NUMHOURS is an interesting option, because it only assumes values different from 0 if the service level is higher than the constraint limit. Therefore, as an option for FITNESS0, we worked on a fitness function that tries to obey the service-level constraint at all costs. Equation (6) shows FITNESS1, built using the same parameters as before, except for MAXPROB.

$$FITNESS\,1 = M - CARS * (1 + NUMHOURS) \tag{6}$$

3.3.2 Initial population generation

When evaluating the population along the iterations, the breeding process comes across several individuals that are certainly not optimal. However, due to scaling and the nature of genetic al-

gorithms, the chromosomes of these individuals might pass through generations. In a Markovian process, as the queueing models we use, it is very unlikely that a situation in the beginning of the first shift will interfere in the third shift, and even in the second shift. For this reason, we propose another method to generate the initial population, focused in the shifts that do not obey the service level constraint. Before generating any initial population members, we ran the non-stationary queueing model to find the tours in which the delay probability constraint was not obeyed. For example, if the constraint was not obeyed somewhere between 2 and 3 a.m., then all tours beginning at midnight and ending at 8 a.m. were chosen because police officers are in the middle of their shift (mealtime breaks were considered as part of their shifts). This strategy was used to modify only tours that did not obey the service level constraint. Equation (7) shows the proposed method, where $CARS_i^0$ is the number of vehicles assigned to tour i for iteration 0 (initial population); U_i and U_j are random numbers generated according to a uniform distribution (0 as minimum and 1 as maximum); l is the set of tours chosen; and $CARS_j^{SIPP-IP}$ is the number of vehicles assigned to chosen tours in the SIPP-IP solution:

$$CARS_i^0 = \frac{U_i \sum_{j \in l} CARS_j^{SIPP-IP}}{\sum_{j \in l} U_j} \tag{7}$$

3.4 Local Search Heuristic

If we consider that the SIPP-IP solution is a reasonably good initial guess to the problem, it is straightforward to think about using a local search heuristic to find a solution. The module uses the SIPP-IP solution and tries to satisfy the delay probability constraints through minor changes to the current solution (for an introduction to local search algorithms, see Luke, 2013). The main idea is to try to change servers' tours to satisfy the delay probability constraint. In case it is not possible to fulfill the constraints only changing servers' tours, we add one new server to a chosen tour and try to manage it until the delay probability constraint is obeyed. In the other hand, if the SIPP-IP solution fulfills the delay probability constraint, it will be replaced by a poorer SIPP-IP solution, which is calculated after relaxing the delay probability constraint. Table 2 shows the notation used for the local search heuristic.

Table 3 illustrates the proposed local search heuristic through a pseudocode. It starts on iteration $k \leftarrow 0$ and calculates a solution from the SIPP-IP. At this point, the SIPP-IP considers a rC_D to ensure that C_D is still active on the nonstationary queueing model (measured on the schedule evaluator). Then, the code set $maxt$ as 24 hours. Note that the primary stop condition is to make C_D inactive. The code verifies whether $maxt$ is acceptable (capable of identifying a mt_w^s inside $[0, maxt)$). Then, it seeks the mt_w^s in which $\pi_D^k(mt_w^s) > C_D$, defining the tour w as one reference. In the next step, the code looks for y, where $y \in WS_w$ and with the lowest hourly mean delay probability ($\acute{\pi}_D$) on its mealtime break, y_{sched}^k. Then, it takes one server from tour w and adds one server to tour y, if possible. If this exchange turns $\pi_D(t_y^k)$ or $\pi_D^k(t_w^k)$ worse than $\pi_D^{k-1}(t_w^{k-1})$, then the script will cancel the server exchange and will not allow the same shift w to be chosen,

Table 2 – Notation and description for the local search heuristic.

Notation	Description
W	Set of all tours.
w, y	Tours of work (one of the rows from Table 1).
k	Current iteration.
n_w^k	Number of servers scheduled to tour w at iteration k.
mt_w^s	Starting time of the mealtime break of tour w.
WS_w	Set of tours with the same starting time as tour w.
rC_D	Relaxed delay probability constraint.
C_D	Delay probability constraint.
$maxt$	Observable time horizon.
$\pi_D^k(t)$	Delay probability on iteration k at epoch t.
$\acute{\pi}_D^k(t)$	Mean delay probability on iteration k at period $[t, t+1)$.

by limiting the $maxt$ to $mt_w^s - 1$. If there is no acceptable time horizon, the tour y receives a new server and starts a new iteration, resetting $maxt$ to 24 hours.

4 COMPUTATIONAL RESULTS

This section presents the results computed for the aforementioned studied case with the genetic algorithm and the proposed improvements. Ahead in this section we present the results for the local search and some comparisons between both solution methods in a sensitivity analysis. All computations ran in a Windows 10 Home, with a processor Intel Core® i5 7300HQ 2.50GHz and 8GB of RAM memory.

4.1 Genetic algorithm

Figure 2 shows the first results for configuration explained here. To obtain more consistent data, we ran the genetic algorithm and calculated an average value for each measure. Hence, Figure 2 does not show actual values for any run of the algorithm but an average value for all of them. Max., min., and avg. are the highest, lowest, and average FITNESS0 values for the population in each iteration, respectively. In observing the reference values of Figure 2, one can see a slight improvement to the average and minimum values for FITNESS0. However, the highest FITNESS0 value is in the initial population (iteration 0) and then falls and increases again, but it never reaches the initial solution value for FITNESS0. On 25 out of 30 runs, the best solution found was the SIPP-IP solution. In four runs, the genetic algorithm found a solution with 28 CARS, and in one run, it found a solution with 27 CARS. None of the 30 solutions obeyed the delay probability constraint. It is important to note that the SIPP-IP solution found (number of servers for each shift) was different from that shown in Ingolfsson et al. (2002) but with the same CARS value. This SIPP-IP solution had a MAXPROB of 0.12, against 0.14 for the SIPP-IP solution from Ingolfsson et al. (2002).

Algorithm 1 Proposed pseudocode for the local search heuristic.

start
$k \leftarrow 0$;
$C_D \leftarrow 0.1$;
$rC_D \leftarrow rC_D$;
Run SIPP-IP;
Schedule Evaluator;
while $max\pi_D(t) < rC_D$, **do**
$k \leftarrow k+1$;
$rC_D \leftarrow rC_D + 0.05$;
Run SIPP-IP using rC_D as delay probability constraint to obtain server requirements;
Schedule Evaluator;
end while
while $max\pi_D^k(t) > rC_D$, **do**
$maxt \leftarrow 24$ hours;
while $maxt > mt_1^e$, **do**
Identify last mt_w^s, such that $mt_w^s \leq maxt$ and $\pi_D^k(mt_w^s) > C_D$;
Identify tour mt_y^s such that $\pi_D^k\left(mt_y^s\right) \leq \pi_D^k(mt_x^s), \forall x \in WS_w$;
$k \leftarrow k+1$;
$n_x^k \leftarrow n_x^{k-1}, \forall x \in W$;
if $n_w^k - 1 \geq 0$, **then**
$n_w^k \leftarrow n_w^{k-1} - 1$;
$n_y^k \leftarrow n_y^{k-1} + 1$;
Schedule Evaluator;
if $\pi_D^k(mt_w^s) > \pi_D^{k-1}(mt_w^s)$ **Or** $\pi_D^k\left(mt_y^s\right) > \pi_D^{k-1}(mt_w^s)$, **then**
$n_w^k \leftarrow n_w^{k-1}$;
$n_y^k \leftarrow n_y^{k-1}$;
$maxt \leftarrow mt_w^s - 1$;
elseif $max\pi_D^k(t) > C_D$
return
end if
else
$maxt \leftarrow mt_w^s - 1$;
end if
end while
$k \leftarrow k+1$;
$n_y^k \leftarrow n_y^k + 1$;
Schedule Evaluator;
end while
end
return

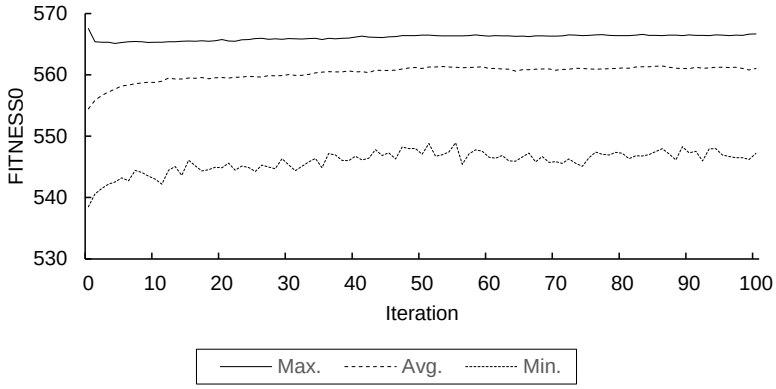

Figure 2 – Average reference values for FITNESS0 at each iteration.

Figure 3 shows the average results for the genetic algorithm after using FITNESS1 for 30 runs. Differently from FITNESS0, one can see a greater gap between the reference values in the first iterations and greater improvements to the average and minimum values. However, the upper reference values show a slight increase on the first iterations and an even slighter decrease until the final iterations. Only four solutions were found after the 20 first iterations, and 19 were found until iteration 10.

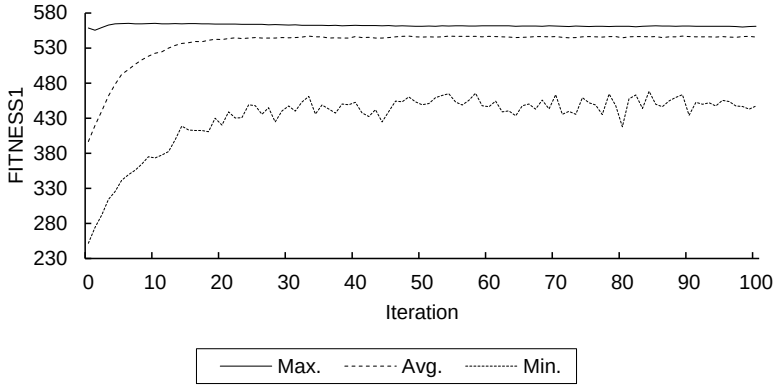

Figure 3 – Average reference values for FITNESS1 at each iteration.

Figure 4 shows the results for both fitness functions. The circle sizes represent the number of times that a solution was the solution found. Firstly, FITNESS0, as mentioned before, found only three solutions, none of which obeyed the delay probability constraint. Secondly, FITNESS1 found 18 different solutions, but it did not obey the delay probability constraint just twice. The most common solution had 31 CARS and a MAXPROB of 0.09; this solution was found eight times.

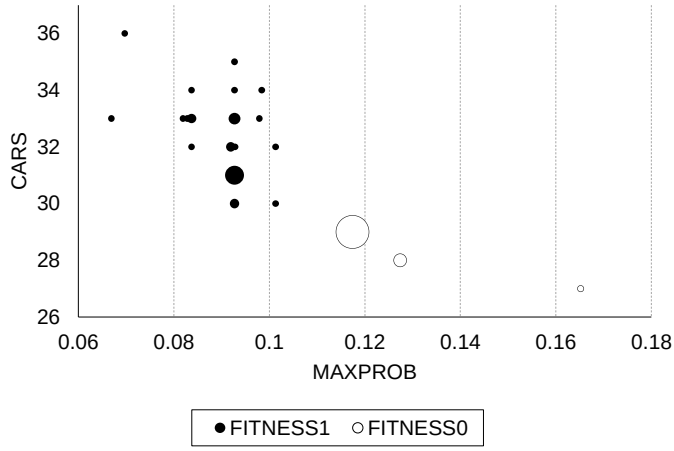

Figure 4 – Experimentation with two fitness functions in the genetic algorithm.

It is also important to note that changing the fitness function had an effect on runtimes. While FITNESS0 had an average runtime close to 400 seconds, FITNESS1 had 600-second runtimes. There was also a difference in variation, with FITNESS0 having greater variance and FITNESS1 having a smaller one. We stored previous solutions in a database, which was erased at each new run. FITNESS0 found its solutions after looking at 2,500 solutions, while FITNESS1 looked at 3,200 solutions, on average.

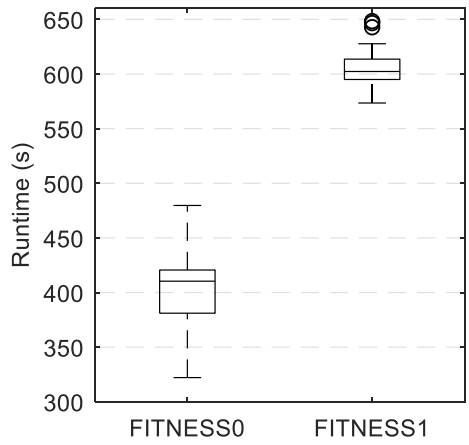

Figure 5 – Boxplot for runtimes in both fitness functions.

Figure 6 shows the average reference results for FITNESS1 after using the alternative method for generating the initial population. Note that the initial population had good reference values, even for the poorest solution. In 20 out of 30 runs, the best solution was found among the initial

population and the first iteration. This solution uses 30 CARS and has 0.09 MAXPROB. For comparison, using the previous initial population generation method, this solution was found only eight times, versus 26 times using the new method. Also, no solutions were found after the eighth iteration. The upper solutions decreased their FITNESS1 along the iterations, even though the average solutions had only slight improvements.

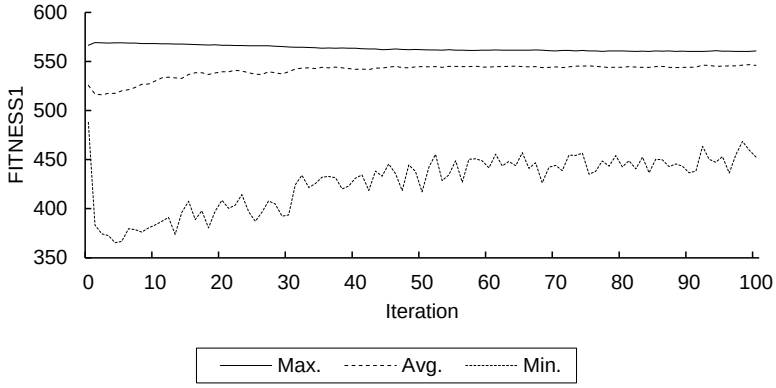

Figure 6 – Average reference values for FITNESS1 at each iteration using the new method to generate the initial population.

4.2 Local search heuristic

Figure 7 presents the MAXPROB and CARS results for each iteration of the proposed local search heuristic under different delay probabilities. Each marker represents the results of a single iteration. Larger markers indicate the final solution, and the numbers linked to them denote at which iteration the final solution was found. Hence, no more than 6 iterations separate the SIPP-IP solution (iteration 1) from the final solutions. This solution is final, and the local search will not find any other solution for this problem because there were no random choices or steps. Note that, although the algorithm tries to change servers from tours in the best way (taking a server with a mealtime break during a busy period and put it into a tour with a calm mealtime break), it always found worse solutions than the SIPP-IP solution. In the other hand, after adding new servers, rearranging them among tours decreased MAXPROB. Moreover, it is interesting to note that the simple idea of adding servers based on their mealtime breaks is a good strategy to reduce delays.

4.3 Comparisons

This final set of tests was aimed at comparing the results from the genetic algorithm and the proposed local search algorithm. The comparisons focused on increases and decreases in demand. The service times remained the same.

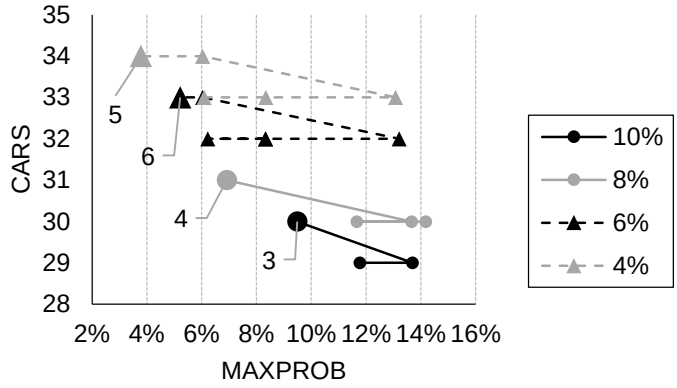

Figure 7 – Results for the local search heuristic under different delay probability constraints. Large markers indicate the final solution and the number linked to them the number of iterations.

Figure 8 presents the number of vehicles for both the genetic algorithm and the proposed local search heuristic as well as the solution from the SIPP-IP, for illustrational purposes. The genetic algorithm is represented by circles. The larger the circles are, the more times the genetic algorithm found that number of vehicles in the final solution. This does not mean that these solutions are the same; rather, it only means that they have the same number of vehicles. One can observe that the local search heuristic usually found the same solution as the genetic algorithm while the SIPP-IP overestimated the performance, allocating fewer vehicles.

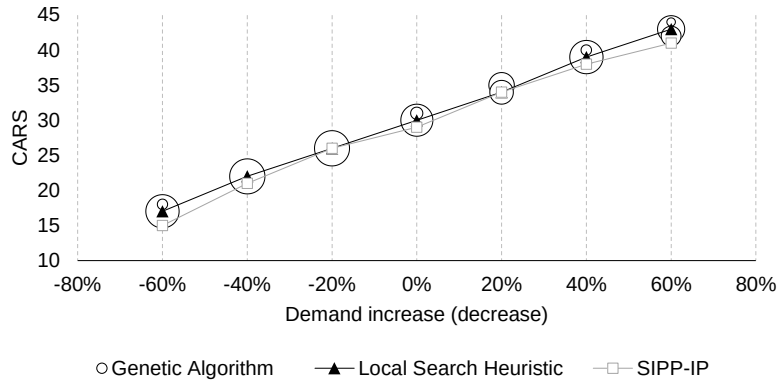

Figure 8 – Comparison of the genetic algorithm, proposed local search heuristic, and SIPP-IP for CARS.

Figure 9 explores the delay probability calculated for final solutions from the genetic algorithm, the proposed local search heuristic, and the SIPP-IP. In none of the cases did the SIPP-IP solution have acceptable performance. Also, as the demand increased, the genetic algorithm found more non-acceptable solutions (delay probability higher than 0.10). The genetic algorithm only bested the proposed local search heuristic, for a 60% demand increase, in which the proposed local

search needed 43 vehicles to guarantee an adequate delay probability, the genetic algorithm found a solution with only 42 vehicles.

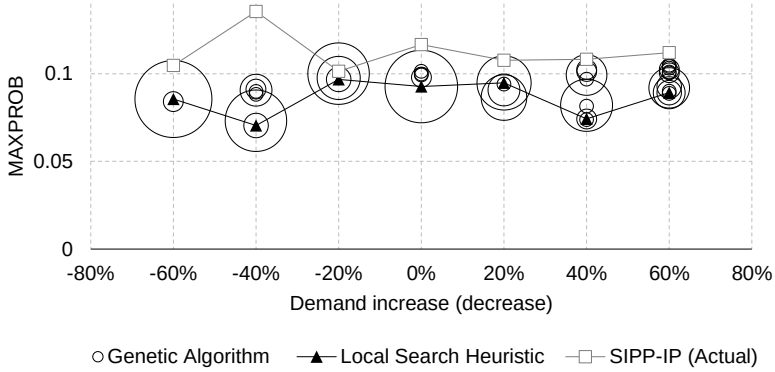

Figure 9 – Comparison of results from the genetic algorithm, proposed local search heuristic, and SIPP-IP (actual performance) for MAXPROB.

Another important point regards runtimes. Figure 10 shows the correlation between the average runtimes and the number of iterations for the proposed local search heuristic. The proposed local search heuristic had almost constant runtimes, much smaller than those seen for the genetic algorithm (about 600 seconds, as seen in Figure 5).

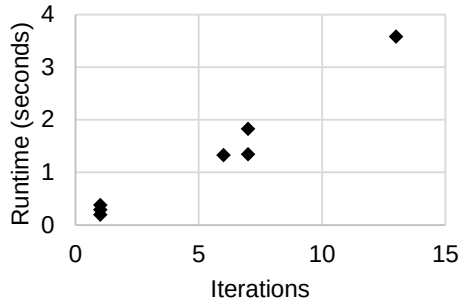

Figure 10 – Correlation between the number of iterations and the runtime for the proposed local search heuristic.

5 FINAL CONSIDERATIONS

We revisited the police precinct study from Kolesar et al. (1975) and Ingolfsson et al. (2002) to improve the shift-scheduling process using nonstationary queueing models to evaluate schedules. The genetic algorithm used in Ingolfsson et al. (2002) received improvements over the fitness function and initial population generation method. We also proposed a local search heuristic, in the form of a hill-climb heuristic. This simple heuristic passed through an initial test, and

its results were compared to those of the genetic algorithm through a sensitivity analysis by decreasing and increasing arrival rates on a small-scale system.

Our work has important implications for managers to improve the service quality and efficiency of an ESS. Firstly, the improved genetic algorithm found solutions that obey the delay probability constraint more often and have close to the optimum number of vehicles. Two important features were responsible for these improvements: the new fitness function, which forced the choice of schedules that obey the delay probability constraint, and the new initial population generation method. We illustrated through a series of tests how these new features affect convergence and the genetic algorithm's final solutions. In addition, results from the improved genetic algorithm led us to the proposed local search heuristic. This heuristic also finds close to optimum schedules with much lower computational expense, especially under low arrival rates. For comparison, the improved genetic algorithm found solutions after 600 seconds, on average, while the proposed local search heuristic found them in less than 5 seconds (even less than 1 second, in some tests). Both the genetic algorithm and local search heuristic determined the appropriate number of vehicles allocated in each possible tour, while considering 1-hour-long mealtime breaks.

Differently from a genetic algorithm, the developed heuristic will always come to the same results, as it does not rely on random choices. Furthermore, it finds one final solution from one initial solution (the SIPP-IP solution), rather than from a population of solutions. As the proposed local search heuristic does not use chromosomes, its solutions are not limited to the limits that the chromosome encoding might impose on the problem.

The developed heuristic works with only one well-defined goal, making it simple and straightforward to implement. It also relies on the intuitive idea of managing servers' tours based on their mealtime breaks. Nevertheless, the code for the heuristic is simple enough to accept changes and cope with multiple objectives.

There is a wide field for further research. Moreover, using Chapman–Kolmogorov equations is computationally expensive, so we encourage the use of approximate methods like randomization/uniformization (Grassmann, 1977) and the stationary backlog-carryover approach (Stolletz, 2008). The search for better and more efficient heuristics and algorithms is encouraged. Also, there is still room for improvement to the genetic algorithm, such as by using approximations instead of exact models or better fitness functions using aggregate measures. We suggest the use of the proposed local search heuristic for small-scale systems, when considering the use of complex performance measures, to take advantage of the simplicity of our method.

ACKNOWLEDGEMENTS

Research conducted at São Paulo State University – UNESP, but Caio Vitor Beojone is now a member of Urban Transport Systems Laboratory (LUTS), School of Architecture, Civil and Environmental Engineering (ENAC), École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, CH-1015.

The authors acknowledge the three anonymous reviewers for their comments, which greatly improved this paper.

References

[1] ALFARES HK. 2004. Survey, Categorization, and Comparison of Recent Tour Scheduling Literature. *Annals of Operations Research,* **127**: 145-175.

[2] ANGELO SA, ARRUDA EF, GOLDWASSER R, LOBO MS, SALLES A & SILVA JR. 2017. Demand forecast and optimal planning of intensive care unit (ICU) capacity. *Pesquisa Operacional,* **37**(2): 229-245.

[3] BAKER KR. 1976. Workforce Allocation in Cyclical Scheduling Problems: A Survey. *Operational Research Quarterly,* **27**(1): 155-167.

[4] BEOJONE CV. 2017. *Performance and alternative scenarios evaluation on a SAMU using the stationary and nonstationary hypercube model.* (Master's dissertation), UNESP, Bauru. Retrieved from https://repositorio.unesp.br/

[5] BROWN L, GANS N, MANDELBAUM A, SAKOV A, SHEN H, ZELTYN S & ZHAO L. 2005. Statistical Analysis of a Telephone Call Center: A Queueing-Science Perspective. *Journal of the American Statistical Association*, **100**(469): 36-50. doi:10.1198/016214504000001808

[6] BUFFA ES, COSGROVE MJ & LUCE BJ. 1976. An integrated work shift scheduling system. *Decision Sciences,* **7**: 620-630.

[7] CHIYOSHI F, IANNONI AP & MORABITO R. 2011. A tutorial on hypercube queueing models and some practical applications in emergency service systems. *Pesquisa Operacional,* **31**(2): 271-299.

[8] CHUNG K & MIN D. 2014. Staffing a service system with appointment based customer arrival. *Journal of the Operational Research Society,* **65**: 1533-1543.

[9] CREEMERS S, BELIËN J & LAMBRECHT M. 2012. The optimal allocation of server time slots over different classes of patients. *European Journal of Operational Research,* **219**: 508-521.

[10] DANTZIG GB. 1954. Letter to the Editor - A Comment on Edie's "Traffic Delays at Toll Booths". *Journal of the Operations Research Society of America,* **2**(3): 339-341.

[11] DEFRAEYE M & VAN NIEUWENHUYSE I. 2016. Staffing and Scheduling under nonstationary demand for service: A literature review. *Omega,* **58**: 4-25.

[12] DIETZ D. 2011. Practical scheduling for call center operations. *Omega,* **39**: 550-557.

[13] EDIE LC. 1954. Traffic Delays at Toll Booths. *Journal of the Operations Research Society of America,* **2**(2): 107-138.

[14] FELDMAN Z, MANDELBAUM A, MASSEY WA & WHITT W. 2008. Staffing of Time-Varying Queues to Achieve Time-Stable Performance. *Management Science,* **54**(2): 324-338.

[15] GILLARD J & KNIGHT V. 2014. Using Singular Spectrum Analysis to obtain staffing level requirements in emergency units. *Journal of the Operational Research Society,* **65**: 735-746.

[16] GOLDBERG DE. 1989. *Genetic Algorithms in Search, Optimization & Machine Learning.* Amsterdam: Addison-Wesley.

[17] GRASSMANN W. 1977. Transient solutions in Markovian queues: an algorithm for finding them and determining their waiting-time distributions. *European Journal of Operational Research,* **1**(6): 396-402.

[18] GREEN LV & KOLESAR PJ. 1991. The Pointwise Stationary Approximation for Queues with Nonstationary Arrivals. *Management Science,* **37**(1): 84-97.

[19] GREEN LV & KOLESAR PJ. 2004. Improving Emergency Responsiveness with Management Science. *Management Science,* **50**(8): 1001-1014.

[20] GREEN LV, KOLESAR PJ & SOARES J. 2001. Improving the Sipp Approach for Staffing Service Systems That Have Cyclic Demands. *Operations Research,* **49**(4): 549-564.

[21] INGOLFSSON A. 2005. Modeling the M(t)/M/s(t) Queue with Exhaustive Discipline. Retrieved from http://www.bus.ualberta.ca/aingolfsson/working_papers.htm

[22] INGOLFSSON A, AKHMETSHINA E, BUDGE S, LI Y & WU X. 2007. A Suvey and Experimental Comparison of Service-Level-Approximation Methods for Nonstationary M(t)/M/s(t) Queueing Systems with Exhaustive Discipline. *INFORMS Journal on Computing,* **19**(2): 201-214.

[23] INGOLFSSON A, CAMPELLO F, WU X & CABRAL E. 2010. Combining integer programming and the randomization method to schedule employees. *European Journal of Operational Research,* **202**: 153-163.

[24] INGOLFSSON A, HAQUE A & UMNIKOV A. 2002. Accounting for time-varying queueing effects in workforce scheduling. *European Journal of Operational Research,* **139**: 585-597.

[25] JENNINGS OB, MANDELBAUM A, MASSEY W & WHITT W. 1996. Server Staffing to meet time-varying demanda. *Management Science,* **42**(10): 1383-1394.

[26] KIM JW & HA SH. 2012. Advanced workforce management for effective customer services. *Quality & Quantity,* **46**: 1715-1726.

[27] KIM, S-H & WHITT W. 2014. Are Call Centers and Hospital Arrivals Well Modeled by Nonhomogeneous Poisson Processes. *Manufacturing & Service Operations Management,* **16**(3): 464-480.

[28] KLEINROCK L. 1975. *Queueing systems. Volume I: theory.* New York: Wiley.

[29] KO YM & GAUTAM N. 2010. Transient Analysis of Queues for Peer-based Multimedia Content Delivery. *IIE Transactions,* **42**(12): 881-896.

[30] KOLESAR PJ, RIDER KL, CRABILL TB & WALKER WE. 1975. A Queuing-Linear Programming Approach to Scheduling Police Patrol Cars. *Operations Research,* **23**(6): 1045-1062.

[31] LUKE S. 2013. *Essentials of Metaheuristics* (2nd ed.). Lulu. Retrieved from http://cs.gmu.edu/\~{}sean/book/metaheuristics/

[32] SCHMID V. 2012. Solving the dynamic ambulance relocation and dispatching problemusing approximate dynamic programming. *European Journal of Operational Research,* **219**: 611-621.

[33] SCHWARZ JA, SELINKA G & STOLLETZ R. 2016. Performance analysis of time-dependent queueing systems: Survey and classification. *Omega,* **63**: 170-189.

[34] SIMPSON N & HANCOCK P. 2009. Fifty years of operational research and emergency response. *Journal of the Operational Research Society*, **60**: S126-S139. doi:10.1057/jors.2009.3

[35] SOUZA RM, MORABITO R, CHIYOSHI F & IANNONI AP. 2015. Incorporating priorities for waiting customers in the hypercube queuing model with application to an emergency medical service system in Brazil. *European Journal of Operational Research*, **242**: 274-285. doi:10.1016/j.ejor.2014.09.056

[36] STOLLETZ R. 2008. Approximation of the non-stationary M(t)/M(t)/c(t)-queue using stationary queueing models: the stationary backlog-carryover approach. *European Journal of Operational Research,* **190**: 478-493.

[37] SUNGUR B, ÖZGÜVEN C & KARIPER Y. 2017. Shift scheduling with break windows, ideal break periods, and ideal waiting times. *Flexible Services and Manufacturing Journal,* **29**: 203-222.

[38] THOMPSON GM. 1997. Labor Staffing and Scheduling Models for Controlling Service Levels. *Naval Research Logistics,* **44**: 719-740.

[39] VAN DEN BERGH J, BELIËN J, DE BRUECKER P, DEMEULEMEESTER E & DE BOECK L. 2013. Personnel scheduling: A literature review. *European Journal of Operational Research,* **226**: 367-385.

How to cite

BEOJONE CA & SOUZA RM. 2020. Improving the shift-scheduling problem using non-stationary queueing models with local heuristic and genetic algorithm. *Pesquisa Operacional*, **40**: e220764. doi: 10.1590/0101-7438.2020.040.00220764.