KATHOLIEKE UNIVERSITEIT
LEUVEN

Faculty of Business and Economics

Controlling excessive waiting times in emergency departments: an extension of the ISA algorithm

Mieke Defraeye and Inneke Van Nieuwenhuyse

DEPARTMENT OF DECISION SCIENCES AND INFORMATION MANAGEMENT (KBI)

KBI 1115

Controlling excessive waiting times in emergency departments: an extension of the ISA algorithm

Mieke Defraeye and Inneke Van Nieuwenhuyse
Research Center for Operations Management
Department of Decision Sciences and Information Management
K.U. Leuven, Belgium

mieke.defraeye@econ.kuleuven.be
inneke.vannieuwenhuyse@econ.kuleuven.be

June 28, 2011

Abstract

In an emergency department (ED), the demand for service is not constant over time. This cannot be accounted for by means of waiting lists or appointment systems, so capacity decisions are the most important tool to influence patient waiting times. Additional complexities result from the relatively small system size that characterizes an ED (i.e. a small number of physicians or nurses) and the presence of customer impatience. Assuming a single-stage multiserver $M(t)/G/s(t) + G$ queueing system with general abandonment and service times and time-varying demand for service, we suggest a method inspired by the simulation-based Iterative Staffing Algorithm (ISA) proposed by Feldman and others (2008) as a method to set staffing levels throughout the day. The main advantage of our extension is that it enables the use of performance measures based on the probability of experiencing an excessive waiting time, instead of the common focus on delay probability as a performance metric.

Keywords: emergency department, personnel planning, time-varying arrival rate

1 Introduction

In many service systems, the demand for service is not constant over time. Fluctuations on a daily, monthly, weekly or yearly basis can be present, which complicates the process of determining appropriate staffing levels. Our particular focus lies on emergency departments (ED's): in these systems, capacity is the main lever to control waiting times, as customer service cannot be guaranteed by means of waiting lists or appointment systems. In an ED, service is mainly related to the length of the customer's waiting time (in particular

the longest waits), and hence, controlling excessive waiting times is usually a primary goal. However, most approaches proposed in the literature focus on delay probabilities or expected waiting times. This paper presents a simulation-based staffing method that enables to stabilize the *probability of excessive waiting* (i.e., the probability that the waiting time exceeds a maximum acceptable value) throughout the day. The suggested method is inspired by the Iterative Staffing Algorithm (ISA), proposed by Feldman et al. [2], which focuses on stabilizing the *delay probability* throughout the day (note that this corresponds to a maximum acceptable wait of zero). The use of discrete-event simulation provides distinct advantages over analytical methods, such as increased flexibility in modeling assumptions and the ability to control the precision of the results. The downside is that evaluation through simulation tends to be more time-consuming. In addition, the focus on performance measures which are related to the length of the waiting times complicates the derivation of appropriate staffing levels based on the simulation's performance outcome.

This research proposes a computationally efficient way to evaluate the probability of excessive waiting, and adjusts the staffing update function of the original ISA algorithm to account for the relatively small system size that characterizes an ED. In our approach, we represent the ED by a single-stage multiserver system with customer abandonments. Our experiments indicate that the proposed method (which we call ISA(τ)) succeeds in finding a staffing vector that meets the performance constraint, in a variety of problem settings. Large systems (for which the number of servers required is in the order of 100) and extremely small instances (requiring only 1-2 servers) can be solved, although the computation time increases severely with the problem size. A solution can be obtained for exponential as well as general service and abandonment time distributions, and staffing intervals are taken into consideration.

The remainder of the paper is organized as follows: in Section 2 we zoom in on the problem that will be addressed. A limited overview of related literature is given in Section 3: Section 3.1 discusses how stationary models can be used to set staffing levels in a nonstationary system, whereas Section 3.2 describes the Iterative Staffing Algorithm as proposed in [2] (which can be considered as the starting point of the approach we suggest). A more detailed description of ISA(τ) follows in Section 4, which includes both a performance evaluation and a staffing updating procedure. Computational results of ISA(τ) are reported in Section 5 and compared to the staffing results obtained by applying stationary approximation techniques available in the literature.

2 Problem description

A single-stage multiserver $M(t)/G/s(t) + G$ queue[1] is considered, with time-varying arrivals and customer impatience (a schematic representation is given

[1]Note that in reality, an ED rather resembles a multiserver queueing *network* in which patients move through several process steps to receive treatment. However, here we focus on single-stage queues.

in Figure 1). Customers enter the system according to a Markovian time-varying pattern with arrival rate $\lambda(t)$ over the time horizon $[0, T]$. Only fluctuations on a daily basis are considered, as these tend to be most outspoken [11]. Customers may leave the system prematurely: the abandonment rate is denoted by θ. Let μ denote the service rate of the service process, i.e. the rate at which one server (or physician) can treat customers (or patients). Contrary to the arrival process, the service and abandonment rates are assumed to be time-invariant and no assumptions are placed on the distribution of these. The main goal is to determine an appropriate staffing requirement function $s(\tilde{t})$, which defines the number of servers (or physicians) to be assigned in each staffing interval $\tilde{t}$, in view of stabilizing the probability of excessive wait (defined as the probability that waiting time of a customer exceeds some "maximum acceptable wait"). The resulting staffing level function should guarantee that the excess wait probability is sufficiently low at all time instants. This performance requirement can be expressed as:

$$Pr(W(t) > \tau) \leq \alpha \qquad 0 \leq t < T \tag{1}$$

where $W(t)$ represents the virtual waiting time[2] of a customer arriving at time t, τ indicates the maximum acceptable waiting time, and α denotes the targeted (i.e. the maximum allowed) excess wait probability. We assume that τ is never larger than the expected time-to-abandon (otherwise τ would probably be mis-specified, especially in the context of an ED). Note that, by controlling excess wait probabilities, the number of abandoning customers is influenced as well (albeit implicitly). As in reality, the number of servers is only allowed to change at fixed points in time, the planning period over which the number of servers is assumed to remain unchanged will be referred to as the staffing interval (hence the distinction in notation between time t and staffing interval $\tilde{t}$).

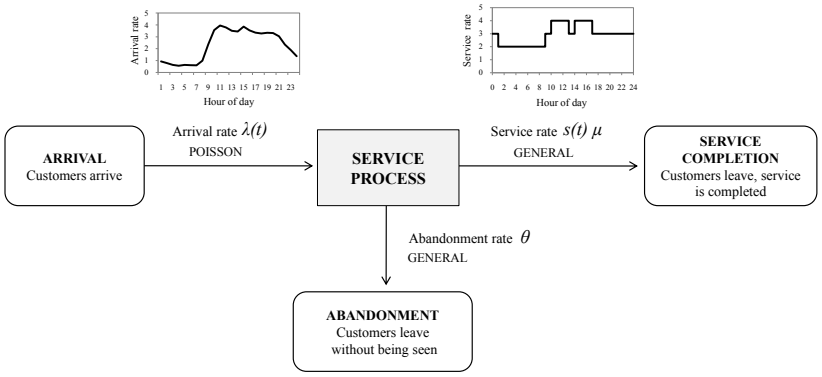


Figure 1: Schematic representation a single-stage queueing system with time-varying demand

[2]The virtual waiting time $W(t)$ is defined as the time that a customer would have to spend in queue if he were to arrive at time t (cf. [9], pp. 13-14).

Note that Expression 1 reduces to a constraint on the delay probability for $\tau = 0$. Consequently, our performance requirement is more general than the (commonly used) delay probability. Measuring performance based on excess wait probabilities gives more leeway to the staffing level function, while at the same time preserving patient service, as the decision maker can decide on the value of τ.

3 Related literature

3.1 Stationary approximations

When considering time-varying arrival rates, assessing system performance becomes severely more complex, as the available stationary results are no longer valid. Nonetheless, stationary models are often relied upon to *approximate* the performance on the nonstationary system. An overview of these so-called stationary approximations (and others) can be found in [8] and [22]. Here, a partial overview of the literature concerning stationary approximations is given, with a main emphasis on 4 easy-to-implement methods which are compared to the proposed algorithm in Section 5.2.3.

Firstly, the system characteristics of the nonstationary model need to be translated into appropriate input parameters for the stationary model(s). In doing so, one has to keep in mind that capacity remains constant within each staffing interval. Often, a stationary model is defined in each staffing interval and hence the continuously varying arrival rate has to be transformed into one single parameter per staffing interval. In the *SIPPmax* approach (introduced in [6]), the arrival rate is set equal to the maximum[3] arrival rate over the staffing interval. Equivalently, one can determine the required staffing level for each time instant based on a stationary model that uses the arrival rate prevailing at that moment, and set the staffing level in each staffing interval equal to the maximum required capacity over the considered staffing interval afterwards (this is referred to as a *Segmented PSA* approach [8]).

However, the insertion of the arrival rate (prevailing at each time or over an interval) into a stationary model often leads to poor results [8]. In this respect, Green and Kolesar [5] pointed out that due to the nonstationarity of the system, peaks in the arrival rate are usually not reflected immediately in the system's offered load. A *time lag* between the actual offered load and the arrival rate is present and the magnitude of this lag is often approximated by the expected service time. As a result, a lagged variant of the previously described SIPPmax approach can be defined (denoted as *lagSIPPmax*), by adding a lag to the arrival rate ($\lambda_{lagSIPPmax}(\tilde{t}) \equiv max\{\lambda(t - 1/\mu) : t \in \tilde{t}\}$).

Although in [7], it was shown that accounting for this time lag always leads to better staffing vectors, the lagged methods are outperformed by the *Modified Offered Load* (MOL) approach (introduced by Jagerman [15] and explored fur-

[3]The average arrival rate can be used instead of the maximum. However, we preferred the latter as this results in a higher amount of safety capacity (yet more elevated labor costs).

ther by Massey and Whitt [18], [19]). Better staffing vectors can be obtained if a *modified arrival rate* is plugged into each stationary model, which is derived from the infinite server offered load prevailing at that time (denoted as $m_\infty(t)$ in what follows, and equivalent to the number of customers in system if infinitely many servers would be available). The infinite server offered load usually reflects the time-dependence better than the stationary approximation of the offered load, and moreover, a tractable expression for $m_\infty(t)$ exists [1]. The arrival rate to be inserted into the stationary models then equals $\lambda_{MOL}(t) \equiv m_\infty(t)\mu$.

Once the nonstationary system has been transformed into one or more stationary models, the performance corresponding to a certain staffing vector can be assessed. For the $M/M/s$ model, the delay probability and excess wait probability can be calculated explicitly in a straightforward way, using the available closed form formulas (applied to each staffing interval, cf. [9] pp. 66-72):

$$Pr(W > 0) = \frac{\frac{(s\rho)^s}{s!(1-\rho)}}{\left(\frac{(s\rho)^s}{s!(1-\rho)} + \sum_{n=0}^{s-1} \frac{(\rho s)^n}{n!}\right)} \tag{2}$$

$$Pr(W > \tau) = Pr(W > 0)e^{-(s\mu-\lambda)\tau} \tag{3}$$

Where $\rho \equiv \lambda/(s\mu)$ represents the traffic intensity, with stationary arrival rate λ and s the number of available servers. The staffing level can then be chosen as the smallest amount of capacity satisfying the performance constraint.

Unfortunately, closed form expressions for the stationary performance are not always available and approximations are often necessary for obtaining the steady state performance measures that are needed to determine appropriate staffing requirements (this particularly holds for the difficult $M/G/s+G$ model, which we focus on). Approximations of performance measures for stationary models can be found in the literature: e.g. methods focusing on $M/M/s+M$ models can be found in Garnett et al. [4] and Mandelbaum and Zeltyn [17], whereas for the $M/G/s+G$ model the interested reader is referred to [14] and [21].

Explicit performance calculations can often be avoided, however, by using a simple rule of thumb: the *square-root staffing rule* (SRS), often also referred to as square root safety staffing (general background on SRS is provided in [3], [20] and [16]). The main benefit of SRS lies in its simplicity and robustness: at each time instant, staffing is determined as the offered load augmented with an amount of safety capacity, which is related to the performance target α (cf. Expression 4). The required amount of safety capacity is proportional to the offered load (denoted m), and depends on the desired quality of service (which is reflected in the quality of service parameter β).

$$s = m + \beta\sqrt{m} \tag{4}$$

However, a key difficulty lies in establishing the link between the desired performance target and the corresponding β that is needed to determine the desired staffing requirements. To obtain the appropriate β, the inverse of the *Halfin-Whitt delay function* (for $M/M/s$ models, cf.[10]) and the *Garnett delay function* (for $M/M/s+M$ models, cf. [4]) are used most commonly; an extension towards $M/M/s+G$ models can be found in [23]. These functions are derived for many-server heavy-traffic regimes, but should work for small system sizes as well [8]. Important to note, however, is that the SRS rule is a rule of thumb and therefore provides no guarantee for the performance constraint being met.

In Section 5.2.3, the proposed ISA(τ) approach is compared to staffing vectors obtained through 4 stationary approximations: SIPPmax, lagSIPPmax and MOL are used to obtain the input parameters for the stationary models, and capacity levels are determined based on the closed form $M/M/s$ formula as well as the SRS rule for $M/M/s+M$ models (for a more elaborate discussion we refer to Section 5.2.3).

3.2 The Iterative Staffing Algorithm (ISA)

In [2], a promising simulation-based technique for determining staffing requirements in time-varying queues is proposed. As the name suggests, the *Iterative Staffing Algorithm* (ISA) repeatedly evaluates and alters several staffing functions, until the desired performance is attained. For each staffing function considered, system performance is evaluated by means of simulation and the staffing level is updated based on the observed performance. This sequence of evaluating performance and updating staffing levels is called an *iteration*.

Performance is expressed in terms of a constraint on the delay probability, that is, the delay probability must lie below a target value α at all time instants:

$$Pr(W(t) > 0) \leq \alpha \qquad 0 \leq t \leq T \tag{5}$$

Equivalently, the delay probability equals the probability that the number of customers in the system $N(\tilde{t})$ surpasses the available capacity $s(\tilde{t})$, leading to the following constraint:

$$Pr(N(\tilde{t}) \geq s(\tilde{t})) \leq \alpha \qquad \forall \tilde{t} \tag{6}$$

The ISA does not explicitly account for the length of staffing intervals and assumes staffing changes can be made almost continuously (or, equivalently, staffing intervals are very small). The planning horizon T is divided into small intervals: staffing changes are only allowed at the start of each interval and the number in system is evaluated once every staffing interval. ISA then proceeds as follows. Initially, all staffing levels are set equal to an arbitrarily large number. Subsequently, system performance is simulated by performing a fixed number of independent replications, which results in a distribution of the number of customers in system at each moment in time. Then, staffing levels are improved

(simultaneously for all staffing intervals) such that at each staffing interval $\tilde{t}$, the staffing level corresponds to the smallest value of $s(\tilde{t})$ satisfying the performance requirement in Expression 6. Formally, the evaluation of the distribution of the number of patients in system at the start of staffing interval $\tilde{t}$ in iteration i ($N_i(\tilde{t})$) determines the staffing level in interval $\tilde{t}$ in iteration $i+1$ ($s_{i+1}(\tilde{t})$):

$$s_{i+1}(\tilde{t}) = \arg\min\{k \in \mathbb{N} : Pr(N_i(\tilde{t}) \geq k) \leq \alpha\} \qquad \forall \tilde{t} \tag{7}$$

The algorithm stops when the staffing changes in subsequent iterations become sufficiently small for all staffing intervals (i.e., staffing levels differ by at most 1, for all $\tilde{t}$).

In [2], the performance of ISA is illustrated by means of two examples: a theoretical example using a sinusoidal arrival pattern and a more realistic example, using the arrival rate of a medium-sized call center (with hourly call volumes varying between less than 100 to over 2000). The service rate equals the abandonment rate in both examples, and consequently the distribution of the number in system in the $M(t)/M/s(t)+M$ is identical to the infinite server equivalent [22]. In addition, both examples are characterized by relatively high average arrival rates (e.g. 100 customers per hour for the sinusoidal example). The performance corresponding to the final ISA staffing levels is simulated using 5000 independent replications. Some other performance measures, such as abandonment probabilities, average queue lengths and average waiting times were evaluated as well. Performance measures (in each staffing interval) were measured and averaged over all replications; no confidence intervals w.r.t. these variables were considered. The results reveal that ISA performs well for both examples, as all delay probabilities are close to the targeted value and more or less stable over time.

The major advantage of the ISA lies in the use of simulation to evaluate system performance. As a result, the appropriateness of the staffing function generated by ISA is validated automatically (that is, under the assumption that the simulation model is adequate). Moreover, this method has potential to be applied to much more general settings, for which analytical results are no longer available. However, some aspects make the traditional ISA less appropriate in an ED context:

- Firstly, an ED is commonly characterized by a rather *small system size*; arrival rates tend to be much lower than in the examples provided in [2]. As discussed in the original publication, the delay probabilities obtained by the original ISA tend to be less stable in periods with low demand, as even a small change in capacity has a substantial impact on performance. Moreover, the conventional stopping rule of the ISA might pose problems: the algorithm stops when the change in staffing requirements is at most 1 unit in all staffing intervals and thus staffing changes of +/- 1 server are disregarded. In small systems however, the addition or removal of one server can result in substantial differences in performance.

- Secondly, the ISA does not explicitly deal with the *length of the staffing intervals*, i.e. the time period over which capacity remains constant (all examples used in [2] assume small staffing intervals with a length of $0.1/\mu$). It can be expected that increasing the length of the staffing interval has a negative impact on the algorithm's performance, as the number of customers in system is measured only once every interval (so increasing interval length obviously leads to a reduction in accuracy). In the method we present in Section 4, this problem is addressed by making a distinction between staffing intervals and (smaller) calculation intervals.
- Finally, the results in [2] indicated that a staffing function that stabilizes delay probability does not automatically stabilize other performance measures (such as abandonment probabilities, average queue lengths and average waiting times). Moreover, the delay probability appears not to be the most appropriate performance measure in an ED environment, as it seems natural that most patients are able to tolerate a (small) waiting time. Consequently, it appears more practical to control the probability of excessive waiting. This allows more flexibility, as the decision maker can decide both on the maximum acceptable waiting time that should be met, and on the service level (i.e., the probability that this maximum waiting time is met). In the original ISA algorithm, using the delay probability is advantageous because of the clear link between the distribution of the number in system (which is an outcome of the simulation model) and the staffing levels needed to achieve the desired performance (cf. Expression 6). Controlling the probability of excessive waiting implies that the length of the waiting time has to be accounted for. Consequently, as discussed in Section 4.1, the distribution of the number in system no longer suffices to determine the appropriate staffing levels. Indeed, for any customer, the waiting time is influenced not only by the number in system (and queue length) upon arrival, but also by the rate at which patients leave the system (i.e. the speed at which the queue decreases).

This justifies the search for a method which (1) is suitable for very small system sizes such as an ED, (2) is capable of dealing with staffing intervals over which the capacity remains constant and (3) emphasizes low probabilities of excessive wait rather than delay probabilities. In the next sections, a more detailed description of the ISA(τ) method is given.

4 ISA(τ): procedure

4.1 Evaluation of the excess wait probability

Measuring the excess wait probabilities ($Pr(W(t) > \tau)$, for all t) in a computationally efficient way poses a challenge. One could simply try to obtain $Pr(W(t) > \tau)$ from the simulation, by splitting the time horizon in successive intervals, and evaluating the probability that a customer arriving in an interval experiences an excessive wait. A detailed observation of performance over

time requires these evaluation intervals to be small. This approach may, however, require a prohibitively large number of replications if the analyst wishes to obtain accurate estimates. Indeed, as the intervals over which performance is measured become smaller, the probability that no customers arrive during the interval increases, which implies that many replications are needed to obtain precise estimates. In this section, we propose a performance assessment method that is computationally efficient while still providing reasonably accurate performance estimates.

Intuitively, for a customer arriving at time t, the excess wait probability at that time corresponds to the probability that the number of previously arrived patients which actually leave the system over the interval $[t, t+\tau]$ (this will be referred to as the number of *departures*) is smaller than or equal to the queue length the customer encounters upon arrival. A departure can result from a service completion, or an abandonment.

Consequently, the excess wait probability can be expressed in terms of the number in system at time t and the number of departures over interval $[t, t+\tau]$. Let $\pi_n(t)$ represent the probability that a customer arriving at time t finds n customers in the system (i.e. in queue and in service). The queue length q is then determined as $max\{n - s(\tilde{t}|t \in \tilde{t}), 0\})$. When no capacity change occurs during interval $[t, t+\tau]$, the excess wait probability can be expressed as

$$P(t) = \sum_{n=s_i(\tilde{t}|t\in\tilde{t})}^{\infty} \pi_n(t)\Big(Pr(d_n \leq q)\Big) \qquad 0 \leq t \leq T \tag{8}$$

where d_n refers to the number of departures given that n customers are present at time t. In case of a capacity change during interval $[t, t+\tau]$, a distinction should be made between the intervals preceding and following the capacity shock[4]. Let t' denote the time at which a capacity increase ($\Delta C > 0$) or decrease ($\Delta C < 0$) occurs. Let $d_{1,n}$ and $d_{2,n}$ represent the number of departures over the time periods $[t, t']$ and $]t', t+\tau]$ respectively, given n customers in the system at time t. Then, given a queue length q, the excess wait probability $P(t)$ equals the probability that the patient is unable to start service both in interval $[t, t']$ *and* in interval $[t', t+\tau]$. Or, equivalently:

$$P(t) = \sum_{n=s_i(\tilde{t}|t\in\tilde{t})}^{\infty} \pi_n(t)\Big(Pr(d_{1,n} \leq q)Pr(d_{1,n} + d_{2,n} \leq q - \Delta C|d_{1,n} \leq q)\Big) \quad 0 \leq t \leq T \tag{9}$$

As results in Section 5.1 indicate, this performance evaluation method provides reasonably accurate results, even when only a small number of replications is performed.

Note that Expression 9 assumes that ΔC takes effect immediately (i.e., at time

[4]Here, we assume at most one capacity shock over the time interval $[t, t+\tau]$, but the method generalizes easily towards a higher number of capacity shocks.

t'). While this is realistic for a capacity increase, it is unlikely to hold in case of a capacity decrease: the doctor will presumably finish the treatment of the current patient before leaving. This is referred to as an exhaustive policy in the literature (see e.g. [13] and [12]). Given its practical relevance, the exhaustive service policy is used in the simulation model. This implies that Expression 9 is likely to overestimate the excess wait probability in the intervals preceding a capacity decrease, as the number of busy servers is likely to surpass the number of scheduled servers.

4.2 Update of the staffing level function

Initially, the capacity in each staffing interval is set equal to the offered load that would prevail if the system was stationary, namely the ratio of the average arrival rate over the time horizon to the service rate. A first stage in the algorithm (referred to as PHASE I, and summarized in Algorithm 1) aims at quickly finding a staffing vector for which performance is *close* to the target (but not necessarily below the target value at all times). To this purpose, the current staffing level function $s_i(\tilde{t})$ is altered iteratively, based on the simulation output. In general, capacity is increased if performance is unsatisfactory and decreased otherwise. An important remark should be made concerning the time period over which the capacity level in a given staffing interval impacts performance. Due to the focus on excessive waiting times, this interval does not coincide with the staffing interval. If a capacity shock occurs at t_s, then the excess wait probabilities are affected for all arrivals after $(t_s - \tau)$, as τ represents the time window within which these patients should start service in order to have a waiting time below the maximum wait (as illustrated in Figure 2).

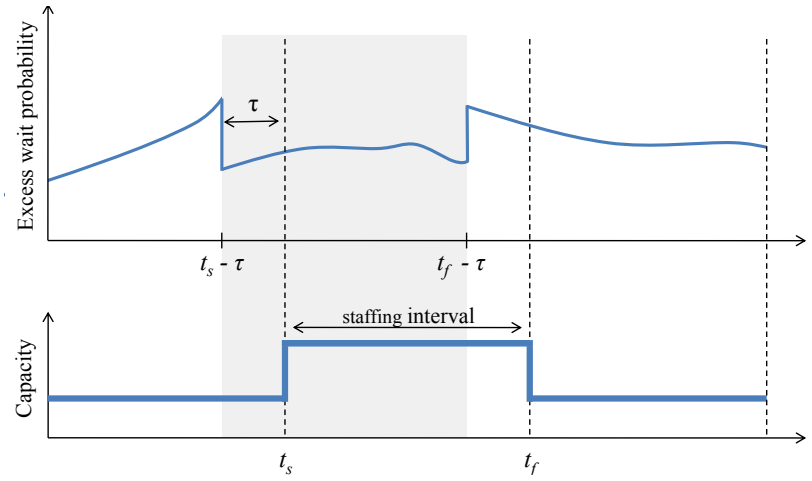


Figure 2: Interval over which capacity impacts performance

Algorithm 1 ISA(τ): PHASE I

Initial staffing vector: $s_0(\tilde{t}) = \frac{\bar{\lambda}}{\mu} \quad \forall \tilde{t}$
Define SCV_i the squared coefficient of variation of $\mathbf{P}_{max,i}$

for all Iterations i **do**
  Simulate staffing vector $\mathbf{s}_i$ (result: performance vector $\mathbf{P}_{max,i}$)
  Update $\mathbb{I}\Big(SCV_i\Big) = \begin{cases} 1 & \text{if } SCV_i \leq SCV_{i-1} \\ 0 & \text{otherwise} \end{cases}$
  Update capacity in all $\tilde{t}$:
    $A_i(\tilde{t}) = 1 + \frac{P_{max,i}(\tilde{t}) - \alpha}{\alpha i}$
    $s_{i+1}(\tilde{t}) = \begin{cases} \lceil s_i(\tilde{t}) A_i(\tilde{t}) \rceil & \text{if } A_i(\tilde{t}) \geq 1, \quad \forall \tilde{t} \\ \lfloor s_i(\tilde{t}) A_i(\tilde{t}) \rfloor & \text{if } A_i(\tilde{t}) < 1, \quad \forall \tilde{t} \end{cases}$
  if $\exists j < i | \forall t : s_j(\tilde{t}) = s_{i+1}(\tilde{t})$ **then**
    Repetition in staffing levels: proceed to PHASE II
  else if Alternating behavior in $\mathbb{I}\Big(SCV_i\Big)$ and $SCV_i \leq 1$ **then**
    SCV performance is stabilizing: proceed to PHASE II
  end if
end for

Thus, the capacity in a staffing interval $\tilde{t}$ starting at t_s and ending at t_f has a direct[5] effect on the performance of patients arriving during interval $[t_s - \tau, t_f - \tau[$. In this respect, in every iteration i, the staffing level in interval $\tilde{t}$ is altered based on vector $P_{max,i}(\tilde{t}) = \max\big\{P_i(t) : t \in \quad [t_s - \tau, t_f - \tau[\big\}$, which represents the maximum excess wait probability over the interval in which performance is impacted by the capacity in staffing interval $\tilde{t}$. More specifically, during each iteration i of the algorithm, $s_i(\tilde{t})$ is updated as follows:

$$s_{i+1}(\tilde{t}) = \begin{cases} \lceil s_i(\tilde{t}) A_i(\tilde{t}) \rceil & \text{if } A_i(\tilde{t}) \geq 1, \quad \forall \tilde{t} \\ \lfloor s_i(\tilde{t}) A_i(\tilde{t}) \rfloor & \text{if } A_i(\tilde{t}) < 1, \quad \forall \tilde{t} \end{cases} \tag{10}$$

where $A_i(\tilde{t})$ refers to an amplification factor, which is determined based on the deviation from the target (in percent) and the number of iterations performed so far:

$$A_i(\tilde{t}) = 1 + \frac{P_{max,i}(\tilde{t}) - \alpha}{\alpha i} \qquad \forall \tilde{t} \tag{11}$$

where $P_{max,i}(\tilde{t})$ is derived from the simulation results, and α denotes the target w.r.t. the excess wait probability. Excess wait probabilities below (above) target will result in an $A_i(\tilde{t})$ below (above) 1 and thus an decrease (increase) in capacity in the corresponding interval (note that due to the rounding in Expression 10, capacity is always increased or decreased with at least one unit).

Expression 11 ensures that $A_i(\tilde{t})$ approaches 1 ($\forall \tilde{t}$) as the number of iterations increases, forcing the algorithm to decrease the size of the staffing changes

[5]It is clear that an indirect effect is present as well; i.e. the capacity level in any staffing interval also has an impact on the performance in all later time instants, through the number of customers in system.

as it progresses, eventually switching to unit-size changes (and, in the limit, converging to a final staffing vector despite the fact that possible deviations from the target may still remain). The aim of PHASE I is to quickly zoom in on a number of staffing vectors which are close to the target (hence the term "exploration phase"); these are then further fine-tuned for feasibility in PHASE II (exploitation phase). Note that, in fact, the choice of the factor i in the denominator of Expression 11 is rather arbitrary; other factors may be considered, such as $i/2$ or i^2. Especially in large systems, high values for the factor should be used with caution: they may lead the algorithm to switch to unit-size capacity changes too soon, causing the number of iterations in the exploration phase to increase substantially. In Appendix A, the impact of a change in the factor, for the small and large system studied in this paper is shown. As evident from the Appendix, the number of iterations in the exploration phase increases drastically for the large system when a factor i^2 is used. Moreover, there is no evidence that this decrease in efficiency in the exploration phase pays off in the exploitation phase, nor does it seem to impact the quality of the final solution (both in terms of excess wait probability and staffing cost).

We allow the algorithm to stop exploring when either (1) cycling occurs (meaning that the staffing level vector put forward in the current iteration has already been assessed during a previous iteration), or (2) when the excess wait probability is stabilizing[6]. To check condition (2), we keep track of the squared coefficient of variation of the excess wait probability over the time horizon in each iteration i (denoted as SCV_i), and record whether it increases or decreases with respect to the previous iteration (denoted as $\mathbb{I}(SCV_i) = 1$ if $SCV_i <= SCV_{i-1}$, 0 otherwise). The algorithm stops when SCV_i is sufficiently low (we obtained good results with the criterion $SCV_i \leq 1$ in our experiments) and a cycle occurs in $\mathbb{I}(SCV_i)$, signaling that SCV_i starts alternating between increase and decrease for subsequent iterations without any significant improvement. Though condition (2) is more ad hoc, it is particularly useful in large systems: we observed that although many iterations are needed before cycling in the staffing levels occurs (i.e. before condition (1) is met), SCV_i usually stabilizes far more quickly. An illustration of the typical evolution of SCV_i throughout the algorithm in a large problem setting is given in Figure 3. As such, criterion (2) can substantially lower the computational time in PHASE I for large systems.
Note that the exploration phase does not necessarily result in a feasible staffing vector. Consequently, an additional fine-tuning procedure (PHASE II or "exploitation phase", detailed in Algorithm 2) is needed. The aim of PHASE II is to deduce feasible solutions from the infeasible staffing levels obtained in PHASE I, in hope of finding a solution which outperforms the best feasible solution found so far (if any) in terms of labor cost. To that end, all infeasible solutions are first sorted based on increasing maximum excess wait probability over the time

[6]Recall that the original ISA utilizes a different terminating condition: it stops when the staffing level changes at most with one unit in each interval, compared to the previous iteration. We opt not to use this stop criterion due to the focus on small system sizes, where one unit capacity changes may occur more frequently (causing the algorithm to stop prematurely).

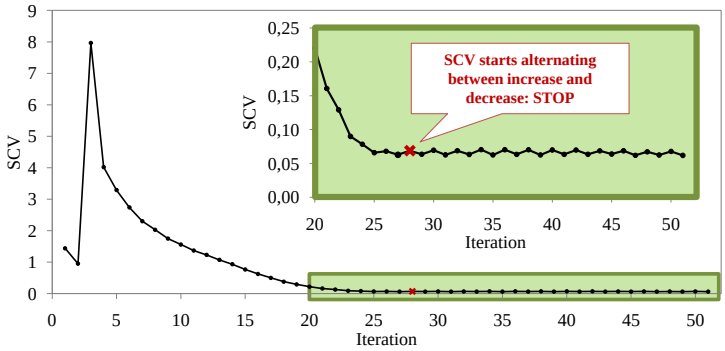


Figure 3: Evolution SCV_i

horizon. Indeed, a lower value indicates smaller deviations from the target, which makes the corresponding solution more promising to explore. In case of a tie, the projected staffing cost (expressed in man-hours) is considered, i.e. the staffing cost that would result when adding 1 unit of capacity to each staffing interval that causes the excess wait probability to surpass the target. Indeed, if this target is exceeded in just a limited number of intervals, a small number of capacity increases may be sufficient to obtain a feasible staffing level vector, which is appealing from a labor cost perspective. All infeasible solutions are improved one by one (in sorted order), each time adding capacity in all intervals where performance is unsatisfactory and assessing the new staffing vector's performance through simulation. This is repeated until either:

- The performance constraint is satisfied at all times; in this case a new feasible solution is found, which is stored if it is less costly than the current best feasible solution in terms of labor cost.
- The performance constraint is not yet met, but further increasing capacity would result in a staffing cost that surpasses the cost of the best feasible solution found so far. In this case, further exploiting the current infeasible solution is futile.

Note that the procedures described in Phase I and Phase II are suitable for small system sizes, largely avoid cyclic behavior and moreover guarantee that the algorithm yields a staffing vector meeting the performance constraint.

Algorithm 2 ISA(τ): PHASE II

Define c_i the cost of a staffing vector $\mathbf{s}_i$
Define c^* the cost of the cheapest feasible solution found so far
Initialize c^* = cost of best feasible solution found during PHASE I (if any), ∞ otherwise

Sort all infeasible solutions considered in PHASE I, based on
1) increasing $\max_{\tilde{t}} \left\{P_{max}(\tilde{t})\right\}$
2) cost + number of staffing intervals with $P_{max}(\tilde{t}) > \alpha$

for all infeasible solutions j (in sorted order) **do**
$s'_j(\tilde{t}) = \begin{cases} s'_j(\tilde{t}) + 1 & \text{if } P_{max,j}(\tilde{t}) > \alpha \quad \forall \tilde{t} \\ s'_j(\tilde{t}) & \text{if } P_{max,j}(\tilde{t}) \leq \alpha \quad \forall \tilde{t} \end{cases}$
Determine c_j = cost of $s'_j(\tilde{t})$

while $c_j < c^*$ **do**
Simulate $s'_j(\tilde{t})$
if $\max_{\tilde{t}} \left\{P_{max,j}(\tilde{t})\right\} > \alpha$ **then**
$s'_j(\tilde{t}) = \begin{cases} s'_j(\tilde{t}) + 1 & \text{if } P_{max,j}(\tilde{t}) > \alpha \quad \forall \tilde{t} \\ s'_j(\tilde{t}) & \text{if } P_{max,j}(\tilde{t}) \leq \alpha \quad \forall \tilde{t} \end{cases}$
Determine c_j = cost of $s'_j(\tilde{t})$
else
if $c_j < c^*$ **then**
Better feasible solution found: store $s'_j(\tilde{t})$
Store $c^* = c_j$
end if
end if
end while
end for

5 Computational results

5.1 Evaluation of the performance measurement method

First, the performance calculation method presented in Section 4.1 is evaluated. For a given staffing level function (shown in the top pane of Figure 4), the bottom pane of Figure 4 represents on the one hand the excess wait probability (and delay probability) calculated by means of Expression 9, using 2500 replications[7]. On the other hand, it shows the average excess wait probability (and delay probability) determined by means of a simple simulation, i.e. based on the waiting times *observed* in each calculation interval, during a simulation run

[7]The results shown pertain to a simulation model using the arrival pattern, abandonment rate and service rates of the small sized example shown in Table 1. The maximum acceptable wait was set to $\tau = 10$ minutes.

consisting of 50 000 replications. Figure 4 clearly indicates that the excess wait probability calculation method suggested in Expression 9 corresponds closely to the simulated values. Also, the method suggested in Expression 9 requires far less replications to obtain a "smooth" performance curve, which is due to the absence of missing observations that would prevail if performance would be measured based on the waiting times observed during each replication[8]. Moreover, Figure 4 indicates that the delay probability and the excess wait probability move in sync. Obviously, the delay probability always exceeds the excess wait probability.

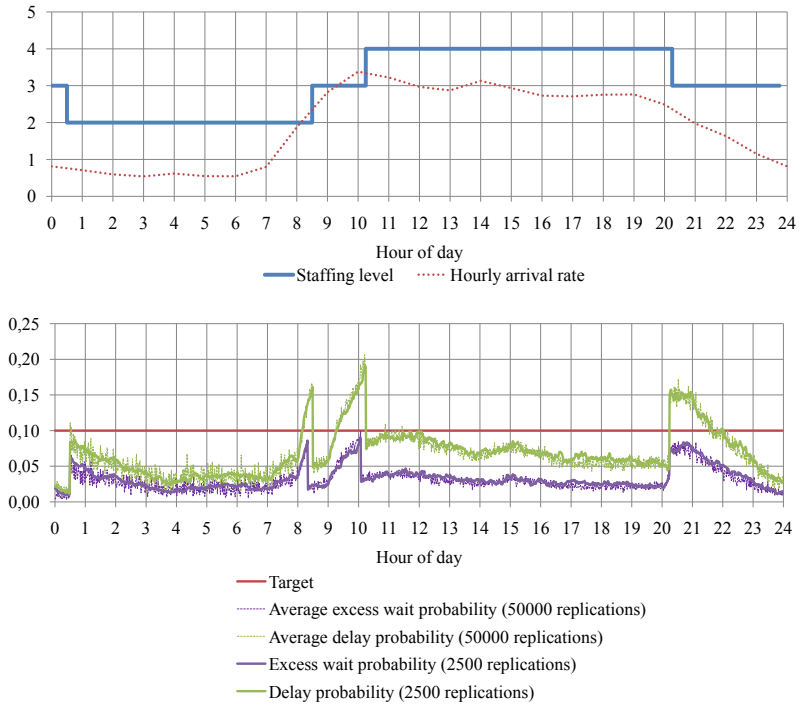


Figure 4: Evaluation performance calculation method

Similar conclusions hold for the large system (a plot is given in Appendix B). There however, both performance methods almost coincide, as the calculation which is based on 10 000 instead of 50 000 replications, is far more accurate compared to the small system ("unobserved" waiting times occur less frequently, due to the large system size).

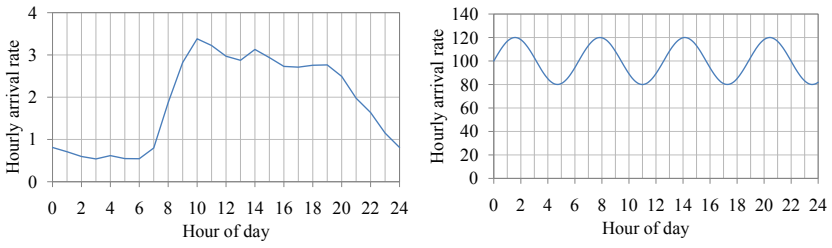
[8]Important to note here is that the excess wait probability is *only* observed if an arrival occurs during the considered interval, so in certain intervals the number of observations used in the calculation of this average excess wait probability might be substantially less than 50 000.

5.2 Evaluation of ISA(τ)

The method proposed in Section 4 was tested for two settings: the first is similar to the example used in [2] (we added the assumption of 15 minute staffing intervals), the second setting was determined based on real-life arrival data obtained from a Belgian hospital. Table 1 summarizes the main characteristics for both examples; in Figure 5 the corresponding arrival rates are plotted (note that the example derived from [2] represents a large scale system, with arrival rates varying between 80 and 120 customers per hour). For both examples, a 24-hour time horizon is considered and performance was calculated quasi-continuously, i.e. once every minute (this will be referred to as the length of the *calculation interval*). The length of the staffing interval equals 15 minutes and the maximum acceptable waiting time τ is set to 10 minutes. Per iteration of the algorithm, 2500 replications are performed. The algorithm's performance for the 2 examples is first evaluated in a $M(t)/M/s(t)+M$ context in Section 5.2.1. However, the algorithm remains applicable in a $M(t)/G/s(t)+G$ context, as the experiments in Section 5.2.2 indicate. In Section 5.2.3, we compare the proposed ISA(τ) method with some heuristics available in the literature.

	Belgian hospital	**Example based on [2]**
Service rate (customers/hour)	2	1
Abandonment rate (customers/hour)	0.25	1
Time horizon	24 hours	
Calculation interval	1 minute	
Staffing interval	15 minutes	
Maximum acceptable wait τ	10 minutes	
Target excess wait probability α	0.1	
Performance constraint	$Pr(W(t) > \tau) \leq \alpha \quad \forall t$	
Number of replications (per iteration)	2500	

Table 1: System parameters



(a) (b)

Figure 5: Arrival rate (a) Belgian hospital (b) example based on [2]

5.2.1 ISA(τ): Exponential service and abandonment times

A comparison of the results for both small and large settings in an $M(t)/M/s(t)+M$ setting (see Table 2), leads to the conclusion that our algorithm results in staffing levels that indeed meet the desired performance targets in relatively few iterations. That is, a staffing vector is found for which the probability of excessive wait lies below the chosen acceptable waiting time threshold and does not depend on the customer's arrival time. However, the number of iterations needed is proportional to the system size: small systems (which we focus on) require less iterations than large systems. Moreover, as the capacity shocks are more frequent and larger in size in the large scale system (see Figure 7), so are the performance shocks (for both excess wait probability and delay probability). Finally, the performance graphs in Figures 6 and 7 clearly indicate that the delay probability is indeed a more restrictive performance measure and, therefore, overstaffing might occur if the delay probability is used as a performance constraint (assuming that customers tolerate a waiting time of $\tau > 0$).

	Belgian hospital	**Example based on [2]**
Number iterations PHASE I	6 (no feasible)	17 (no feasible)
Number iterations PHASE II	5 (3 feasible)	6 (2 feasible)
$\max_t \left\{P(t)\right\}$	0.090	0.100
Staffing cost	74	2257.25

Table 2: Results ISA(τ): exponential service and abandonment times

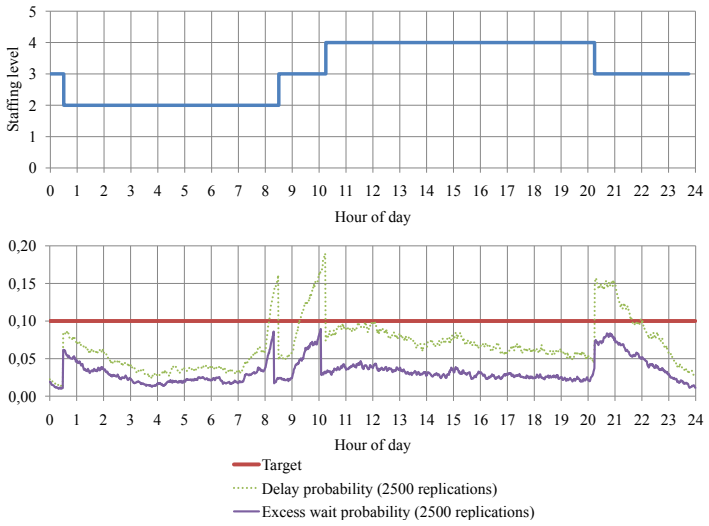


Figure 6: Small system: staffing vector and resulting performance

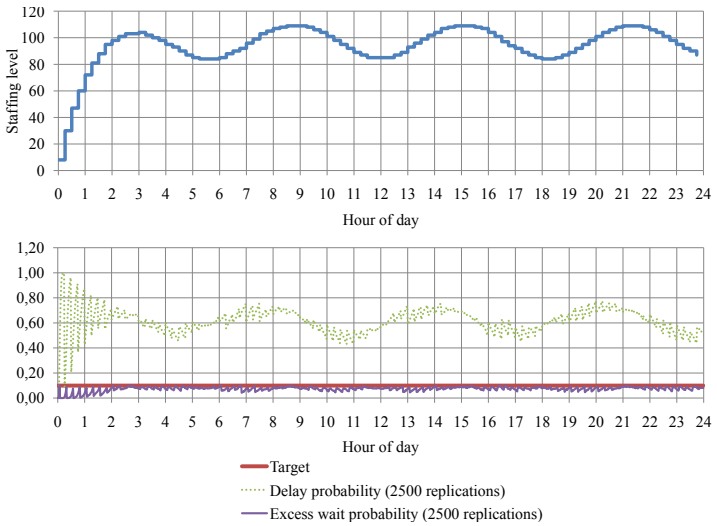


Figure 7: Large system: staffing vector and resulting performance

To gain insight in the part of the solution space unexplored by the ISA(τ) algorithm and the quality of the solution obtained, better solutions were sought by means of implicit enumeration, for the test setting of the Belgian hospital. To keep the solution space fairly small, we assume only 6 staffing intervals (which can be interpreted as 4-hour shifts). The capacity in each interval is bounded below by 1, as at least one server should be present at each moment in time. A reasonable estimate of the upper bound on capacity in each staffing interval is derived based on three overly restrictive assumptions: the upper bound represents the capacity needed to guarantee a *delay probability* below the target in a stationary $M/M/s$ model (i.e. *without abandonments*) where the arrival rate equals the *maximum arrival rate* over the time horizon. Following this reasoning, each staffing interval requires at most 5 units of capacity, resulting in 15625 possible staffing vectors. For this simplified setting, ISA(τ) yields the optimal solution (i.e. the lowest-cost solution which is feasible with respect to the performance constraint). In addition, for a setting with only 3 staffing intervals (or, 8-hour shifts), complete enumeration was used in view of visualizing the solution space. A graphical representation of the solution space can be found in Figure 8: for each staffing vector, the cost and the maximal excess wait probability are plotted (the performance constraint is met if the maximal excess wait probability of a staffing vector lies below the target). The staffing vectors explored by ISA(τ) are indicated as well. Again, the optimal solution can be obtained by means of ISA(τ). Though these results cannot be generalized (ad hoc experimentations on the problems with 96 staffing intervals described in Table 1 showed that better solutions can be found there), they give an indication of the solution quality of ISA(τ).

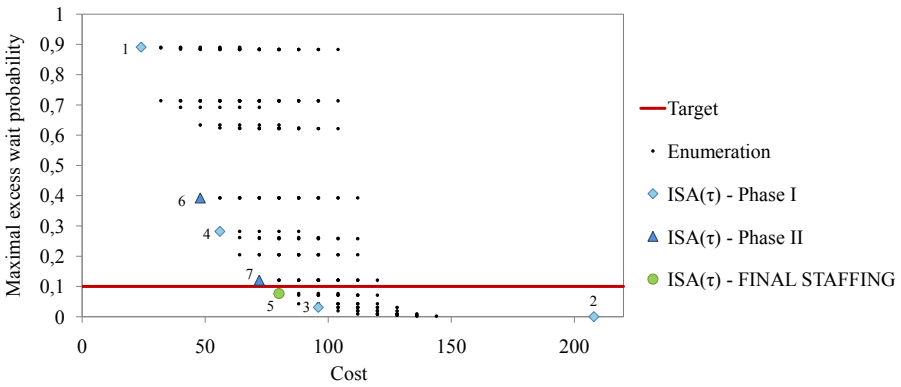


Figure 8: Comparison ISA(τ) staffing vs. complete enumeration (small system size, 3 staffing intervals)

5.2.2 Lognormal service and abandonment times

One of the major advantages of the proposed method consists in its general applicability (due to the use of a simulation model to assess performance). To this end, experiments were repeated assuming an $M(t)/G/s(t)+G$ setting, assuming a lognormal distribution for the service and abandonment times. Two cases were examined, in which service and abandonment times are lognormally distributed with squared coefficient of variation (SCV) equal to 1.5 and 0.5. As the results in Table 3 indicate, the algorithm's effectiveness does not change when departing from the exponential service time and abandonment time distributions. Also, the number of iterations needed by the algorithm does not change substantially.

	Small system ($\mu = 2$)		Large system ($\mu = 1$)	
	SCV = 0.5	SCV = 1.5	SCV = 0.5	SCV = 1.5
Number iterations PHASE I	6	5	13	22
Number iterations PHASE II	5	4	16	6
$\max_t \left\{P(t)\right\}$	0.090	0.083	0.099	0.099
Staffing cost	73.25	74.5	2492.25	2363.4

Table 3: Results ISA(τ): Lognormal service and abandonment times

5.2.3 Comparison to other staffing heuristics

In this section, the staffing vector obtained by the algorithm proposed in Section 4 is compared to some readily implementable staffing heuristics available in the literature. These are all so-called *stationary approximations*, which implies that

a stationary model is used in each separate staffing interval to determine the desired capacity level. The key features of the selected heuristics are summarized in Table 4; for a more elaborate description we refer to Section 3.1.
Each of the four heuristics was applied to both the small and large system discussed in Section 5.2. Figures 9 and 10 show the corresponding staffing level vectors and the resulting excess wait probabilities for the large system; Figures 11 and 13 show the results for the small system. In each case, the results are compared to those of ISA(τ).

Abbreviation	Staffing by means of	Applied to	Used arrival rate or offered load
CF_MOL	Closed form formula $M/M/s$	each time instant t	$\lambda_{MOL}(t) \equiv m_\infty(t)\mu$
SRS_SIPPmax	SRS formula $M/M/s+M$ (using Garnett delay function)	each staffing interval $\tilde{t}$	$\lambda(\tilde{t}) \equiv$ maximum $\lambda(t)$ over staffing interval $\tilde{t}$
SRS_lagSIPPmax	SRS formula $M/M/s+M$ (using Garnett delay function)	each staffing interval $\tilde{t}$	$\lambda(\tilde{t}) \equiv$ maximum $\lambda(t-1/\mu)$ over staffing interval $\tilde{t}$
SRS_MOL	SRS formula $M/M/s+M$ (using Garnett delay function)	each time instant t	$m(t) = m_\infty(t)$

Table 4: Other heuristics available in the literature

The obtained staffing levels for the large system (represented in Figure 9) clearly illustrate the importance of the time lag between the arrival rate and the offered load. For SRS_SIPPmax, which utilizes a simple stationary approximation of the arrival rate and does not account for this time lag, the peak staffing levels clearly occur earlier than the peaks in offered load, leading to extremely poor performance. The time-varying nature of the offered load is captured better if a time lag is added (cf. SRS_lagSIPPmax) or if the MOL-approximation is used (cf. CF_MOL and SRS_MOL). The observed performance for CF_MOL in the large system (represented in Figure 10) indicates that it leads to overstaffing, a result that can be explained by the fact that the underlying closed form $M/M/s$ formula ignores the presence of abandonments.

For the large system, the SRS_MOL heuristic performs pretty well. This is

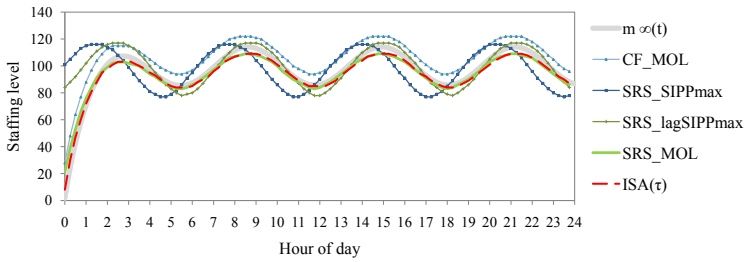


Figure 9: Comparison ISA(τ) staffing vs. other heuristics: Staffing (large system)

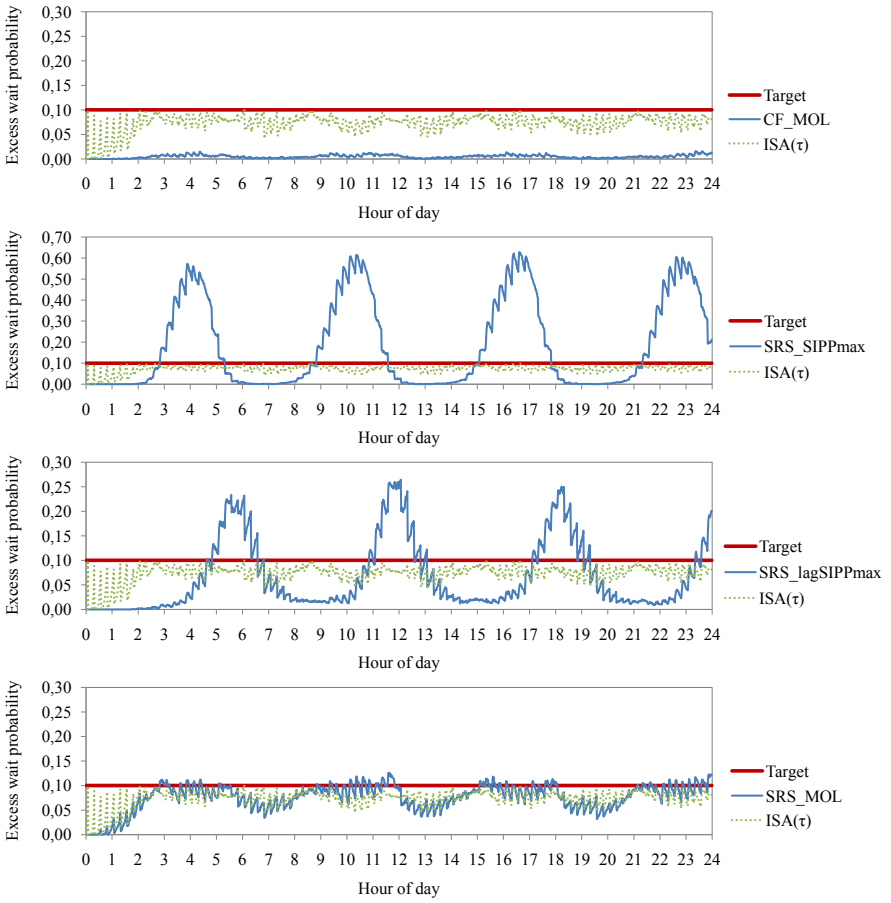


Figure 10: Comparison ISA(τ) staffing vs. other heuristics: Performance (large system)

not surprising, as the offered load used in the SRS_MOL heuristic is exact[9] and in addition the conditions for the Garnett delay function are met (i.e., exponential service and abandonment times and a sufficiently large number of servers). For the large system, the SRS_MOL staffing vector yields the best results among all approximations. The excess wait probability occasionally exceeds the target though, which might be explained in part by the presence of staffing intervals, and by the fact that the SRS rule is a rule of thumb and therefore provides no guarantee for the performance constraint being met. The excess wait probabilities obtained through ISA(τ) closely resemble those of SRS_MOL, yet for

[9]The distribution of number in system in any $M(t)/G/s(t)+G$ system is identical to that of the infinite server model, if the specific condition holds that the abandonment rate is equal to the service rate [22], as is the case in this setting.

ISA(τ) the performance constraint is met at all times.

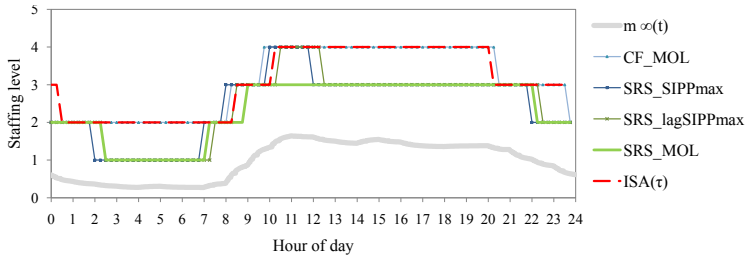


Figure 11: Comparison ISA(τ) staffing vs. other heuristics: Staffing (small system)

The results for the small example setting (given in Figures 11 and 13) show that none of the SRS-based heuristics result in adequate staffing. This might be addressed to the SRS rule performing best for moderate to large offered loads [4], whereas we applied it to a very small system. The closed-form formula results in slightly better staffing vectors, although again, it has the tendency to overstaff as the presence of abandonments is ignored (for this example, the overstaffing is limited due to the low abandonment rate).

Figures 12 and 14 reveal a key advantage of ISA(τ). Whereas in a Markovian setting, the SRS_MOL heuristic yielded performance comparable to ISA(τ) for the large system, this no longer holds when general service and abandonment times prevail: the excess wait probability is considerably above target. For the remaining heuristics, the insights from the Markovian setting largely remain valid: i.e., performance is poor. Consequently, we may conclude that ISA(τ) is the only heuristic that yields consistent and satisfactory performance, both for small and large systems, and in particular in settings where the Markovian assumptions do not hold.

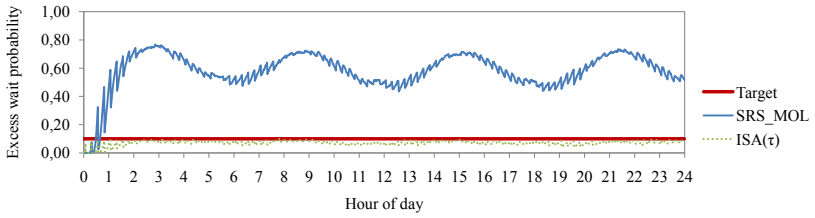


Figure 12: Comparison ISA(τ) staffing vs. other heuristics: Performance (large system, SCV = 0.5)

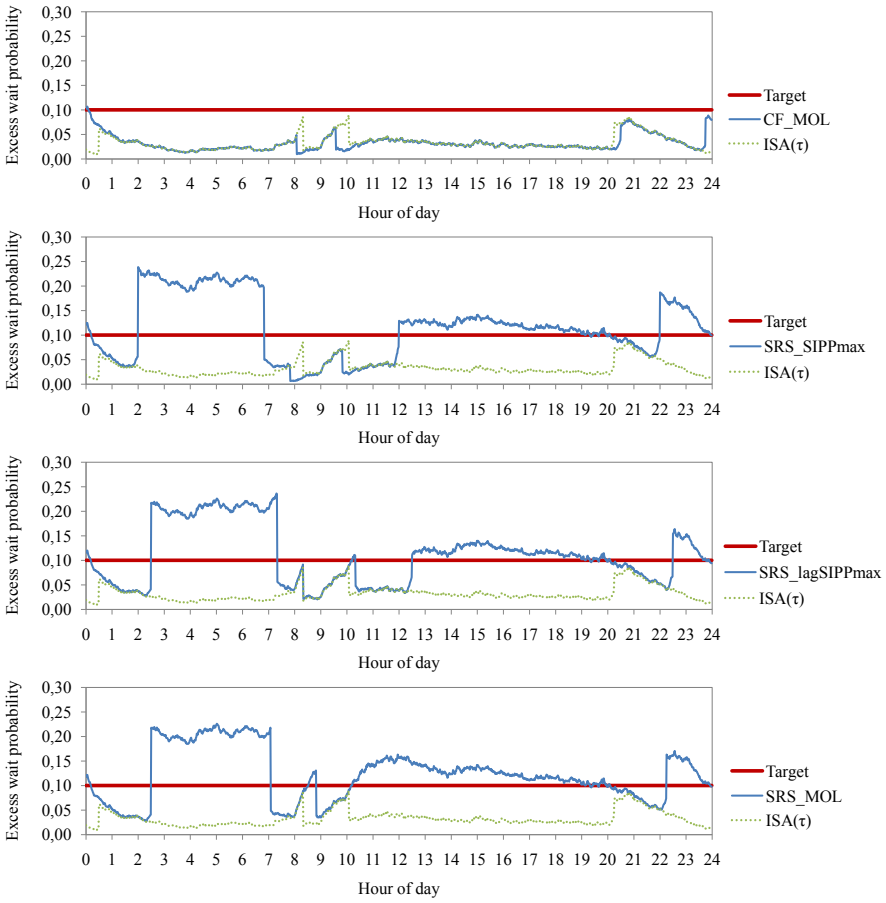


Figure 13: Comparison ISA(τ) staffing vs. other heuristics: Performance (small system)

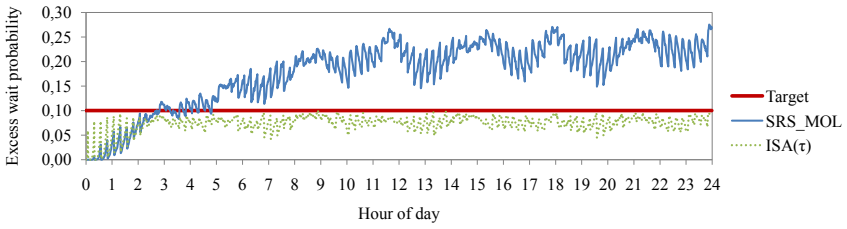


Figure 14: Comparison ISA(τ) staffing vs. other heuristics: Performance (large system, SCV = 1.5)

6 Conclusions and future research

This paper suggests an extension of the simulation-based Iterative Staffing Algorithm (ISA) proposed by Feldman and others [2] as a method to set staffing levels throughout the day. This extension (called ISA(τ)) enables the use of performance measures based on the probability of an excessive wait, instead of the common focus on delay probability as a performance metric. Moreover, it takes into account the sensitivity of small scale systems to changes in the staffing levels, and the presence of staffing intervals. Meanwhile, the advantages of the traditional ISA (namely general applicability, automatic validation) remain valid.

Experiments on two settings (a large system with sinusoidal arrival pattern on the one hand, and a more realistic small ED system on the other hand) illustrate that ISA(τ) is effective in numerous contexts, and consistently outperforms heuristics based on stationary approximations in particular for settings in which the service and abandonment processes are not Markovian. In every test setting, ISA(τ) succeeds in detecting a staffing vector that meets target performance, within a limited number of iterations. In general, the efficiency of the algorithm tends to depend on its parameters (the amplification factor in Expression 11 and the SCV-based stop criterion can be tuned), and the size of the system (larger systems require more computation time).

Future research will focus on including shift constraints in the algorithm and improving the capacity updating function. In addition, further improvements to the algorithm, in view of reducing the computation time and number of iterations needed to obtain a solution, will be explored. Moreover, testing the algorithm on a more elaborate set of problem instances is advisable.

Acknowledgments

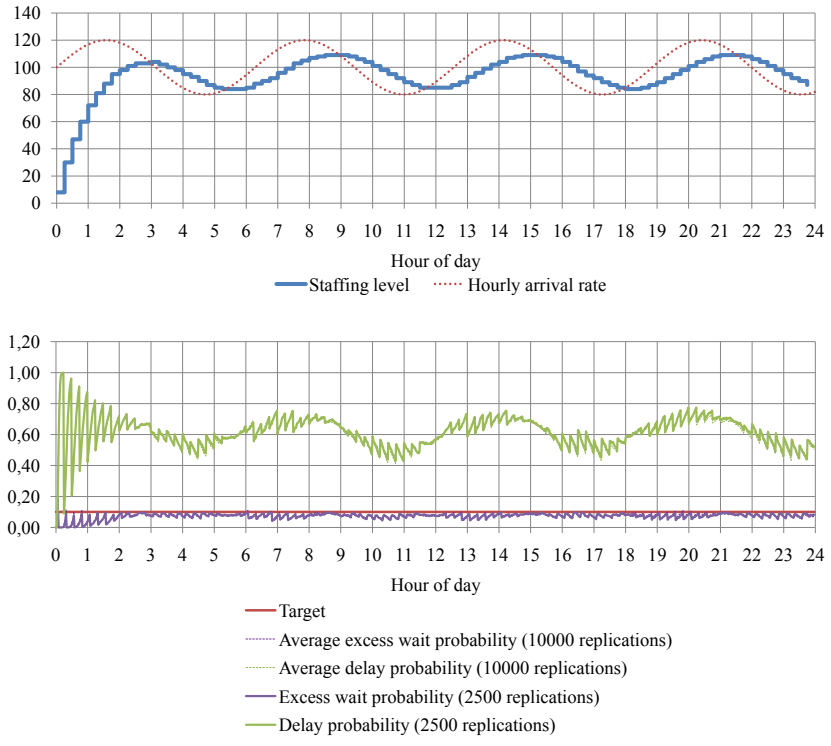
This research was supported by the Research Foundation-Flanders (FWO) (grant no G.0547.09).

Appendix A: Impact of i in $A(\tilde{t})$

	Small system				Large system			
	$i/2$	i	$2i$	i^2	$i/2$	i	$2i$	i^2
Number iterations PHASE I	2	6	5	7	17	17	18	79
Number iterations PHASE II	3	5	5	5	10	6	7	10
$\max_t \left\{P(t)\right\}$	0.090	0.090	0.090	0.086	0.100	0.100	0.099	0.100
Staffing cost	74	74	74	74	2999.75	2257.25	2325.25	2258

Table 5: Impact of i in $A(\tilde{t})$

Appendix B: Evaluation performance calculation method for the large system



References

[1] S.G. Eick,W.A. Massey, W. Whitt, 1993, The Physics of the $M_t/G/\infty$ Queue, Operations Research 41(4) (1993) 731–742.

[2] Z. Feldman, A. Mandelbaum, W.A. Massey, W. Whitt, Staffing of Time-Varying Queues to Achieve Time-Stable Performance, Management Science 54(2) (2008) 324–338.

[3] N. Gans, G. Koole, A. Mandelbaum, Telephone Call Centers: Tutorial, Review, and Research Prospects, Manufacturing & Service Operations Management 5(2) (2003) 79–141.

[4] O. Garnett, A. Mandelbaum, M. Reiman, Designing a Call Center with Impatient Customers, Manufacturing & Service Operations Management 4(3) (2002) 208–227.

[5] L.V. Green, P.J. Kolesar, On the Accuracy of the Simple Peak Hour Approximation for Markovian Queues, Management Science 41(8) (1995) 1353–1370.

[6] L.V. Green, P.J. Kolesar, J. Soares, Improving the SIPP Approach for Staffing Service Systems That Have Cyclic Demands, Operations Research 49(4) (2001) 549–564.

[7] L. V. Green, P.J. Kolesar, J. Soares, An Improved Heuristic for Staffing Telephone Call Centers with Limited Operating Hours, Production and Operations Management 12(1) (2003) 46–61.

[8] L. V. Green, P.J. Kolesar, W. Whitt, Coping with Time-Varying Demand When Setting Staffing Requirements for a Service System, Production and Operations Management 16(1) (2007) 13–39.

[9] D. Gross, J.F. Shortle, J.M. Thompson, C.M. Harris, Fundamentals of Queueing Theory (4th Edition), Wiley Series in Probability and Statistics, Wiley-Blackwell, 2008.

[10] S. Halfin, W. Whitt, Heavy-Traffic Limits for Queues with Many Exponential Servers, Operations Research 29(3) (1981) 567–588.

[11] R. Hall, D. Belson, P. Murali, M. Dessouky, Modeling Patient Flows Through the Healthcare System, in: R. W. Hall, (Ed.), Patient Flow: Reducing Delay in Healthcare Delivery, Springer US, 2006, 91, pp. 1–44.

[12] A. Ingolfsson, Modeling the $M(t)/M/s(t)$ Queue with an Exhaustive Discipline, University of Alberta, Alberta, available at: http://www.business.ualberta.ca/aingolfsson/publications.htm, 2005.

[13] A. Ingolfsson, E. Akhmetshina, S. Budge, L. Yongyue, W. Xudong, A Survey 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 (2007) 201–214.

[14] F. Iravani, B. Balcioglu, Approximations for the $M/GI/N+GI$ Type Call Center, Queueing Systems 58(2) (2008) 137–153.

[15] D.L. Jagerman, Nonstationary Blocking in Telephone Traffic, Bell System Technical Journal 54(3) (1975) 625–661.

[16] G. Koole, A. Mandelbaum, Queueing Models of Call Centers: An Introduction, Annals of Operations Research 113(1-4) (2002) 41–59.

[17] A. Mandelbaum, S. Zeltyn, Service Engineering in Action: The Palm/Erlang-A Queue, with Applications to Call Centers, in: D. Spath, K.-P. Fähnrich, (Eds.), Advances in Services Innovations, Springer Berlin Heidelberg, 2007, pp. 17–45.

[18] W.A. Massey, W. Whitt, An Analysis of the Modified Offered-Load Approximation for the Nonstationary Erlang Loss Model, The Annals of applied probability 4(4) (1994) 1145–1160.

[19] W.A. Massey, W. Whitt, Peak Congestion in Multi-Server Service Systems with Slowly Varying Arrival Rates, Queueing Systems 25(1-4) (1997) 157–172.

[20] W. Whitt, Understanding the Efficiency of Multiserver Service Systems, Management Science 38(5) (1992) 708–723.

[21] W. Whitt, Engineering Solution of a Basic Call-Center Model, Management Science 51(2) (2005) 221–235.

[22] W. Whitt, What You Should Know About Queueing Models to Set Staffing Requirements in Service Systems, Naval Research Logistics 54(5) (2007) 476–484.

[23] S. Zeltyn, A. Mandelbaum, Call Centers with Impatient Customers: Many-Server Asymptotics of the $M/M/n+G$ Queue, Queueing Systems 51(3-4) (2005) 361–402.