Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

Prashanth L. A.† Prasad H. L.† Nirmi Desai$ S. Bhatnagar†

Gargi Dasgupta$

IISc-CSA-SSL-TR-2011-4


Stochastic Systems Lab
Computer Science and Automation
Indian Institute of Science, India
November 2011
Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

Prashanth L. A.† Prasad H. L.† Nirmit Desai§
S. Bhatnagar† Gargi Dasgupta§

Revision 2
April, 2012

Abstract

Service systems are labor intensive due to the large variation in the tasks required to address service requests from multiple customers. Aligning the staffing levels to the forecasted workloads adaptively in such systems is crucial for optimizing labor costs. However, it is a nontrivial task because of a large number of parameters and operational variations leading to a huge search space. One of the main challenges here is to optimize the staffing while maintaining the system in steady-state and compliant to aggregate SLA constraints. Further, because these parameters change on a weekly basis, the optimization should not take longer than a few minutes. We formulate this problem as a constrained hidden Markov cost process parameterized by the (discrete) staffing levels. We propose three novel discrete parameter simulation optimization methods SASOC-G, SASOC-H, and SASOC-W for solving this problem. The core of each of the algorithms is a multimescale scheme that incorporates a SPSA based algorithm for primal descent and couples it with a dual ascent scheme for the Lagrange multipliers. All SASOC algorithms incorporate a generalized projection operator that helps imitate a continuous parameter systems, thus facilitating the use

†Department of Computer Science and Automation, Indian Institute of Science {prashanth, hlprasu, shalabh}@csa.iisc.ernet.in. §IBM Research, Bangalore, INDIA {nirmit.desai, GaargiDasgupt}@in.ibm.com
of the simultaneous perturbation technique. The SASOC algorithms differ from each other in the gradient estimation and optimization methods used. Whereas SASOC-G is a first-order gradient estimation method, SASOC-H and SASOC-W are second-order Newton methods. We evaluate these optimization schemes on data from five real-life service systems and compare their performances with a state-of-the-art optimization tool-kit OptQuest. We observe that our schemes are more than an order of magnitude faster than OptQuest and thus are particularly suitable for adaptive labor staffing. Also, we observe that our schemes guarantee convergence and find better solutions than OptQuest in many cases. Amongst our schemes, SASOC-H and SASOC-W find better solutions than SASOC-G in many cases, with SASOC-W being marginally better than SASOC-H.

**Keywords:** Simulation based constrained optimization, discrete parameter simulation optimization, Adaptive labor staffing, Service systems, inequality constraints, Lagrange multiplier, Simultaneous perturbation stochastic approximation.

1 Introduction

A Service System (SS) is an organization composed of (i) the resources that support, and (ii) the processes that drive service interactions so that the outcomes meet customer expectations [1, 2, 3]. This paper focuses on the SS in the data-center management domain, where customers own data centers and other IT infrastructures supporting their businesses. Owing to size, complexity, and uniqueness of these technology installations, the management responsibilities of the same are outsourced to specialized service providers. A delivery center is a remotely-located workplace from where the service providers manage the data-centers. Each service request (SR) that arrives at a delivery center requires a specific skill and is supported by a service worker (SW) with the corresponding skill set. The SWs work in shifts which are typically aligned to the business hours of the supported customers. Hence, a group of customers supported by a group of SWs, along with the operational model of how SRs are routed constitutes an SS in this paper. A delivery center may consist of many SS.

We consider the problem of adaptive labor staffing in the context of service systems. The objective is to find the optimal staffing levels in a SS for a given dispatching policy (i.e., a map from service requests to service workers) while maintaining system steady-state and compliance to aggregate service level agree-
ment (SLA) constraints. The staffing levels constitute the worker parameter that we optimize and specify the number of workers in each shift and of each skill level. The SLA constraints specify the target resolution time and the aggregate percentage for an SR originating from a particular customer and with a specified priority level. For instance, a sample SLA constraint could specify that 95% of all SRs from customer 1 with ‘urgent’ priority must be resolved within 4 hours.

While the need for SLA constraints to be met is obvious, the requirement for having queues holding unresolved SRs bounded is also necessary because SLA attainments are calculated only for the work completed. The problem is challenging because analytical modeling of SS operations is difficult due to aggregate SLA constraints and also because the SS characteristics such as work patterns, technologies, and customers supported change frequently. An important aspect to consider in the design of the adaptive labor staffing algorithm is its computational efficiency. A truly adaptive labor staffing algorithm is required to have a low computational time requirement, which in turn helps in making staffing changes on a shorter timescale, for instance, every week.

We formulate this problem as a constrained hidden Markov cost process ([4] define an analogous Markov reward framework, i.e., in the unconstrained setting for a continuous parameter) that depends on the worker parameter. To have a sense of the search space size, an SS consisting of 30 SWs who work in 6 shifts and 3 distinct skill levels corresponds to more than 2 trillion configurations. Apart from the high cardinality of the discrete parameter set, the constrained Markov cost process involves a hidden or un-observed state component. We design a novel single stage cost function for the constrained Markov cost process in a way that balances the conflicting objectives of worker under-utilization and SLA under/over-achievement. The performance objective is a long-run average of this single stage cost function and the goal is to find the optimum steady state worker parameter (i.e., the one that minimizes this objective) from a discrete high-dimensional parameter set. However, our problem setting also involves constraints relating to queue stability and SLA compliance. Thus, the optimum worker parameter is in fact a constrained minimum. Another difficulty in finding the optimum (constrained) worker parameter is that the single stage cost function can be estimated only via simulation. Hence, the need is for a search algorithm that incrementally updates the worker parameter along a descent direction, while adhering to a set of queue stability and SLA constraints.

In this paper, we develop three novel discrete parameter simulation-based optimization algorithms for solving the above problem. Henceforth, we shall refer to these algorithms as SASOC (Staff Allocation using Stochastic Optimization with

---

4 Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems
Constraints) algorithms. The overall optimization scheme in all algorithms has two main stages: (a) evaluation of a candidate solution (worker parameter) to the long-run constrained optimization problem, and (b) updation of the worker parameter in the negative descent direction of the long-run performance objective. For the evaluation step (a) above, we leverage the simulation-based operational models developed in [5]. All the SASOC algorithms that we propose fall into the category of simulation-based optimization methods. These methods do not require a priori knowledge of the service system dynamics in general and the single stage cost function in particular. Instead, they use the output of service system simulation to find the optimum of a certain long term performance objective. Further, all SASOC algorithms that we propose are based on the crucial idea of perturbing the parameter vector using independent, identically distributed symmetric Bernoulli random variables. This idea constitutes the simultaneous perturbation stochastic approximation (SPSA) that is an integral part of all the SASOC algorithms proposed here. More general perturbation random variables may however be used, see for instance [6, 7]. All SASOC algorithms involve a certain generalized smooth projection operator, which is essential to project the continuous-valued worker parameter tuned by SASOC algorithms onto the discrete set. The smoothness is necessary to ensure that the underlying transition dynamics of the constrained Markov cost process is itself smooth (as a function of the continuous-valued parameter) - a critical requirement to prove the convergence of all SASOC algorithms.

2 Related Work

We now review literature in two different areas of related work: (1) techniques pertaining to service systems analysis and (2) developments in stochastic optimization approaches.

2.1 Service Systems

In [8], a two step mixed-integer program is formulated for the problem of dispatching SRs within service systems. While their goal is similar to ours, their formulation does not model the stochastic variations in arrivals and service times. Further, unlike our framework, the SLA constraints in their formulation cannot be aggregates. In [9], the authors propose a scheme for shift-scheduling in the context of third-level IT support systems. Unlike this paper, they do not validate their
method against data from real-life third-level IT support. In [10, 11], simulation-optimization methods are proposed for finding the optimal staffing in a multiskill call center. While [10] proposes a cutting plane algorithm for solving an integer program, [11] relies on obtaining a linear programming solution. However, unlike SASOC algorithms, steady-state system analysis is not performed there. Instead, they consider only a single iteration of the system. Further, the SLA constraints considered there are not aggregate in nature. Also, the algorithms proposed in [10, 11] are heuristic in nature, whereas the SASOC algorithms are provably convergent. In [12], the emergent behavior of a service system consisting of a large number of cells is studied by applying an agent based simulation method. Each cell contains an analytical M/M/1 queue model. The simulation helps observe how cells die and neighborhood patterns emerge among cells. While [12] exemplifies the human aspects of service systems which would be an important future direction for our work, it does not aim to propose a labor-optimization technique. In [13], usage of a simulation based search method is proposed for finding the optimal staffing levels in the context of a call-center domain. They evaluate the system given a staffing level with an analytical model, which is possible in their simplified domain but would not be feasible for service systems due to aggregate SLA constraints and dynamic queues. They apply simulation to search for the optimal staffing level based on a heuristic. An analysis of service systems using the ARENA simulation tool is presented in [14]. Unlike our model, the system there is not subjected to aggregate SLA constraints. Also, in their work, they do not consider preemption of low priority SRs by higher priority SRs and assignment of higher skilled SWs to growing queues of SRs requiring lower skill levels. Their model assumes a pool of dedicated SWs for each customer as well as a separate pool of shared SWs. In [5], a simulation framework for evaluating dispatching policies is proposed. A scatter search technique is used to search over the space of SS configurations and optimize the staff allocation. While we share their simulation model, the goal in this paper is to propose fundamentally new search algorithms that are based on stochastic optimization techniques for discrete parameter search. In general, none of the above papers propose optimization algorithms that are geared for SS and that apply simulation to evaluate and adapt the optimization search parameters.

### 2.2 Stochastic Optimization

A popular and highly efficient simulation based local optimization scheme for gradient estimation is Simultaneous Perturbation Stochastic Approximation (SPSA)
propose by [6]. SPSA is based on the idea of randomly perturbing the parameter vector using i.i.d., symmetric, zero-mean random variables. This scheme has the critical advantage that it needs only two samples of the objective function to estimate its gradient for any $N$-dimensional parameter. In [15], a one-simulation variant of SPSA was proposed. However, the algorithm in [15] was not found to work as well in practice as its two-simulation counterpart. Usage of deterministic perturbations instead of randomized ones was proposed in [16]. The deterministic perturbations there were based either on lexicographic or Hadamard matrix based sequences and were found to perform better than their randomized perturbation counterparts. In particular, the one-simulation variant of the SPSA algorithm that is based on Hadamard matrices was found to significantly perform better than the one-simulation random-perturbation algorithm presented in [15]. A Newton-based SPSA algorithm that needs four system simulations with Bernoulli random perturbations was proposed in [7]. In [17], three new SPSA based estimates of the Hessian that require three, two and one system simulations, respectively, were proposed. In [18], certain smoothed functional (SF) Newton algorithms that incorporate Gaussian-based perturbations were proposed. The algorithms of [7], [17] and [18] were for unconstrained optimization. In [19], continuous optimization techniques such as SPSA and SF, have been adapted to a setting of discrete parameter optimization. Two simulation based optimization algorithms that involve randomized projections have been proposed there for an unconstrained setting. In [20], several simulation based algorithms for constrained optimization have been proposed. Two of the algorithms proposed there use SPSA for estimating the gradient, after applying the Lagrange relaxation procedure to the constrained optimization problem, while the other two incorporate smoothed functional approximation. Constrained optimization in the context of Markov decision processes has been considered, for instance, in [21, 22]. The algorithm proposed in [21] is a three time-scale stochastic approximation scheme that incorporates an actor-critic algorithm for primal descent and performs dual ascent on Lagrange multipliers. However, it assumes full state representation for the underlying MDP. The algorithm proposed in [22] combines the ideas of multi-timescale stochastic approximation and reinforcement learning with function approximation to develop a simulation based online algorithm for a constrained control problem, involving function approximation.

Our SASOC algorithms differ from the stochastic optimization approaches outlined above in various ways. Many algorithms, for instance those proposed in [7, 17, 18], are for unconstrained optimization and in a continuous optimization setting. However, our staff optimization problem is for a discrete worker
Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

parameter and requires SLA and queue stability constraints to be satisfied. While the algorithms of [19] have been proposed for a discrete optimization setting, our SASOC algorithms differ in the projection technique used and more importantly solve an optimization problem under functional (inequality) constraints. While the algorithms of [20] have been developed for constrained optimization, our SASOC algorithms optimize a discrete parameter and also differ in the dual ascent procedure, as we do not accumulate the long-run averages of the constraint functions. Further, unlike the Newton-based methods of [20] which estimate the Hessian and perform an explicit matrix inverse, our SASOC-W algorithm directly tunes the inverse of the Hessian matrix by employing a procedure based on the Woodbury’s identity. Thus, the computational complexity of SASOC-W is significantly lower than its Hessian based counterparts in [20]. To the best of our knowledge, we are the first to present adaptations of Newton-based search approaches for constrained discrete optimization problem. Simulation-based optimization methods in a Markov reward framework have been proposed in [4]. However, these have been proposed for a continuous-valued parameter in a completely observed state setting and further, do not involve constraints. Our problem setting, on the other hand, involves un-observed or hidden state components and falls within the purview of constrained hidden Markov models.

We now outline the contributions of this paper.

2.3 Our contributions

1. We present three novel discrete parameter simulation optimization methods based on simultaneous perturbation stochastic approximation techniques for adaptive labor staffing in service systems. The first algorithm, referred to as SASOC-G, is a first order method that uses SPSA based gradient estimate in the primal. The second and third algorithms, referred to as SASOC-H and SASOC-W respectively, are second-order Newton methods. To the best of our knowledge, this is the first work that develops Newton-based algorithms for constrained discrete parameter optimization.

2. All the three SASOC algorithms that we propose are online, incremental and easy to implement. Further, they possess the necessary convergence properties.

(a) We combine the approaches discussed in [20] for continuous parameter constrained optimization as well as in [19] for discrete parameter
unconstrained optimization to obtain the aforementioned three new algorithms for constrained discrete optimization. Unlike [20] where an explicit inversion of the Hessian at each update step was advocated, we incorporate the Woodbury’s identity to obtain a novel update step for the inverse of the Hessian in our algorithm SASOC-W.

(b) Further, unlike [19] where fully randomized projections were used, we incorporate a generalized projection operator that is continuously differentiable in the parameter and works as a deterministic operator over a large portion of the search space and incorporates randomization over a small portion. This helps in bringing down the computational requirement as a deterministic projection scheme requires less computation than a fully randomized one.

3. We evaluate our algorithms on five real-life SS in the data-center management domain. For each of the SS, we collect operational data on work arrival patterns, service times, and contractual SLAs and feed this data into the simulation model of [5].

4. From the simulation experiments, we observe that our algorithms show overall better performance in comparison with the state-of-the-art OptQuest optimization toolkit [23]. Further, our algorithms are 25 times faster than OptQuest and have a significantly lower execution runtime.

The rest of the paper is organized as follows: In Section 3, we introduce the framework of service systems. In Section 4, we present the detailed problem formulation. In Section 5, we introduce our solution methodology and in Section 6, we present our stochastic optimization techniques, specifically the first order method SASOC-G as well as two second order methods - SASOC-H and SASOC-W, respectively. In Section 7, we discuss the implementation of our algorithms as well as the OptQuest algorithm and present the performance simulation results. In Section 8 we provide the concluding remarks. Finally, the proofs of convergence are provided in an appendix at the end of this paper.

3 Service System Framework

Formally, a service system is characterized by the following entities.

A set of customers, denoted by $C$, supported by the service system.
Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

A set of shifts, denoted by \( A \), across which the service workers are distributed.

A set of skill or complexity levels, denoted by \( B \).

A set of priority levels, denoted by the set \( P \).

A set of time intervals, denoted by \( I \), where during each interval the arrivals stay stationary, with the number of arrivals following a Poisson distribution whose rate parameter is given by the function \( \alpha \) described next.

Arrival rates specified by the mapping \( \alpha : C \times I \rightarrow \mathbb{R} \). We assume that each of the SR arrival processes from the various customers \( C_i \) are independent and Poisson distributed with \( \alpha(C_i, I_j) \) specifying the rate parameter.

Service time distributions characterized by the mapping \( \tau : P \times B \rightarrow (r_1, r_2), r_i \in \mathbb{R}, i = 1, 2 \). Here \( r_1 \) represents the mean and \( r_2 \) the standard deviation of a truncated lognormal distributed random variable corresponding to a particular priority-complexity pair. In other words, if \( M \) is a random variable following a normal distribution with mean \( r_1 \) and standard deviation \( r_2 \), then the truncated lognormal random variable is \( e^M \land \top \), where \( \top \) is a truncation constant that is chosen to be large in practice.

SLA constraints, given by the mapping \( \gamma : C \times P \rightarrow (r_1, r_2), r_i \in \mathbb{R}, i = 1, 2 \). Here \( \gamma(C_i, P_j) = (r_1, r_2) \) means that the SLA target for SRs from customer \( C_i \) and with priority \( P_j \) is \( (r_1, r_2) \), with \( r_1 \) specifying the SLA percentage target and \( r_2 \) the resolution time target (in hours). For instance, \( \gamma(C_1, P_1) = (95, 4) \) translates to the requirement that at least 95% of the SRs from customer \( C_1 \) with priority level \( P_1 \) should be closed within 4 hours. Note that the SLAs are computed at the end of each month and hence the aggregate SLA targets are applicable to all SRs that are closed within the month under consideration.

Note that each arriving SR has a customer identifier \( (\in C) \) and a priority identifier \( (\in P) \) and any SW works in a particular shift \( (\in A) \) and possesses a skill level \( (\in B) \). The set \( I \) and the mapping \( \alpha \) allow us to model the variations in arrival rates better than in a setting where the arrivals are assumed to be Poisson-distributed for the entire period. Further, the time taken by a SW to complete an SR is stochastic and follows a lognormal distribution, where the parameters of the distribution are learned by conducting time and motion exercises described in [5].

In the following we provide the specifics of the simulation framework.
• $\mathcal{I}$ contains one element for each hour of the week. Hence, $|\mathcal{I}| = 168$. Each time interval is one hour long.

• $P = \{P_1, P_2, P_3, P_4\}$, where, $P_1 > P_2 > P_3 > P_4$.

• $\mathcal{B} = \{\text{High, Medium, Low}\}$, where, High $>\text{Medium} > \text{Low}$.

• **Swing**: Swing is invoked when Low queue length $> 10$. When this happens, one SW is randomly chosen with skill level Medium and the SW is assigned SRs from Low queue until Low queue length $< 10$.

• **Preemption**: Preemption relation $\Rightarrow$ is the transitive closure of the tuples $P_1 \Rightarrow P_2$, $P_2 \Rightarrow P_3$, and $P_2 \Rightarrow P_4$.

• **Breaks**: Each SW takes 3 breaks in a shift independently of other SWs with a total duration of 75 minutes. The maximum priority SRs preemptible by breaks is $P_3$. In other words, breaks are modeled as internal SRs with priority $P_2$.

• **On-call**: Minimum priority above which SRs arriving out of shift hours are routed to on-call SW is $P_2$. Hence, SRs with priorities $P_3$ or $P_4$ would wait till the next shift if they arrive in non-shift hours.

• **Infrastructure down**: The mean-time-between-failures of work infrastructure is 178 hours and the mean down time is 104 minutes (based on historical data). Further, the time between failures and down time are distributed exponentially.

Table 1(a) illustrates a simple SS configuration, specifying the staffing levels across shifts and skill levels. This essentially constitutes the worker parameter that we optimize. In this example, $\mathcal{A} = \{S_1, S_2, S_3\}$ and $\mathcal{B} = \{\text{high, medium, low}\}$. Tables 1(b) and 2(b) provide sample utilizations and SLA attainments on a SS with three shifts, two customers and four priority levels. Table 2(a) illustrates the target SLA requirements for the same SS. Figure 1 shows the main components of the SS. The SRs arrive from multiple customers and the arrival rate is specific to the hour of week, i.e., within each hour of week, and for each customer-priority pair, the arrivals follow a Poisson distribution. The parameters of this distribution are learned from historical data over a period of at least 6 months. Once the SR arrives, it is queued up in a matching complexity queue by the queue manager and the dispatcher would then assign it to a SW based on the dispatching policy. For instance, in the PRIOPULL policy, SRs are queued in the complexity queues based directly on the priority assigned to them by the customers. On the other hand, in the EDF policy,
Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

Table 1: Sample workers and utilizations

<table>
<thead>
<tr>
<th></th>
<th>(a) Workers θ_i</th>
<th>(b) Utilizations u_{i,j}</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Skill levels</strong></td>
<td>Shift</td>
<td>High</td>
</tr>
<tr>
<td>S1</td>
<td>S2</td>
<td>S3</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>0</td>
</tr>
<tr>
<td></td>
<td></td>
<td>3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Shift</th>
<th>High</th>
<th>Med</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td>67%</td>
<td>34%</td>
<td>26%</td>
</tr>
<tr>
<td>S2</td>
<td>45%</td>
<td>55%</td>
<td>39%</td>
</tr>
<tr>
<td>S3</td>
<td>23%</td>
<td>77%</td>
<td>62%</td>
</tr>
</tbody>
</table>

Table 2: Sample SLA constraints

<table>
<thead>
<tr>
<th></th>
<th>(a) SLA targets γ_{i,j}</th>
<th>(b) SLA attainments γ’_{i,j}</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Customers</strong></td>
<td>Priority</td>
<td>Bossy Corp</td>
</tr>
<tr>
<td></td>
<td>P_1</td>
<td>95%4h</td>
</tr>
<tr>
<td></td>
<td>P_2</td>
<td>95%8h</td>
</tr>
<tr>
<td></td>
<td>P_3</td>
<td>100%24h</td>
</tr>
<tr>
<td></td>
<td>P_4</td>
<td>100%18h</td>
</tr>
</tbody>
</table>

the time left to SLA target deadline is used to assign the SRs to the SWs i.e., the SW works on the SR that has the earliest deadline.

A SW works in exactly one shift (working days and times) and a SS may operate in multiple shifts. We say that a particular configuration of workers across shifts and skill levels is feasible if (a) the SLA constraints are met and (b) the complexity queues do not become unbounded when using this configuration. While the need for (a) is obvious, the requirement for having bounded complexity queues is also necessary. This is because SLA attainments are calculated only for work completed and not for that waiting for completion in the complexity queues. For instance, say in a given month, 100 SRs arrive at various times from a customer to a SS and only 50 of them are completed within the target completion time stipulated by the SLA constraints. The remaining 50 SRs are still in progress without a known completion time and hence do not have an impact on the SLA attainment measures. Hence, a healthy SLA attainment alone is insufficient and the bound on the growth of complexity queues fills the gap.
In this section, we describe our problem framework. A similar framework is considered for instance in [4], except that the setting considered there is unconstrained and the parameter is continuous-valued. Moreover, in [4] the state is fully observable and rewards (and not costs) are considered. Our setting, however, involves a discrete-time, continuous-space hidden Markov chain represented by \(((X_n, Y_n), n \geq 0)\). The idea here is that whereas \((X_n, Y_n)\) is Markov for a given parameter \(\theta\) that we describe below, the portion \(X_n\) of this process is observed while \(Y_n\) is hidden. We describe \(X_n\) and \(Y_n\) more clearly in Section 4.2. The transition probabilities of this process depend on the worker parameter \(\theta = (\theta_1, \ldots, \theta_N)^T \in \mathcal{D}\), where \(N = |A| \times |B|\). In the above, \(\theta_i\) indicates the number of service workers whose skill level is \((i - 1)|B|\) and whose shift index is \((i - 1)/|B|\), with the indices into the set \(B\) starting from 0, i.e., \(i \in \{0, 1, \ldots, |B| - 1\}\). As an example, the worker parameter for the setting in Table 1(a) is \(\theta = (\theta_1, \ldots, \theta_9)^T = (1, 3, 7, 0, 5, 2, 3, 1, 2)^T\). The parameter vector \(\theta\) takes values in the set \(\mathcal{D}\), where \(\mathcal{D} = \{0, 1, \ldots, W_{\text{max}}\}^N\). \(W_{\text{max}}\) serves as an
Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

\[ \text{Simulate}(\theta_n, T) \]

Instant \( nT \) \( (n + 1)T \)
State \( X_n \) \( X_{n+1} \)

Figure 2: A portion of the time-line illustrating the process

The upper bound for the number of workers in any shift and of any skill level. Note that one can enumerate all the points in \( \mathcal{D} \) as \( \mathcal{D} = \{D^1, D^2, \ldots, D^p\} \) for some \( p > 1 \).

As illustrated in Fig 2, the system stochastically transitions from one state to another, while incurring a state-dependent cost. In addition, there are state-dependent single-stage (constraint) functions described via \( g_{i,j}(X_n), h(X_n), i = 1, \ldots, |C|, j = 1, \ldots, |P| \). These shall correspond to the SLA and queue stability constraints. Note that these functions depend explicitly on only the observed part \( X_n \) of the state process \( (X_n, Y_n), n \geq 0 \). The state together with the cost and constraint functions constitutes the constrained hidden Markov cost process. The \( n \)th system transition of this underlying process involves a simulation of the service system for a fixed period \( T \) with the current worker parameter \( \theta_n \). For instance, in our experiments, \( T = 10 \), i.e., we simulate the service system for a period of ten months with the staffing levels specified by \( \theta_n \). Also, note that this is a continuously running simulation, where at discrete time instants \( nT \) we update the worker parameter \( \theta_n \) and the simulation output causes a probabilistic transition from the current state \( (X_n, Y_n) \) to the next state \( (X_{n+1}, Y_{n+1}) \), while incurring a single stage cost \( c(X_n) \). The precise definitions of the state, the cost and the constraints functions are given in Section 4.2. By an abuse of notation, we refer to the state at instant \( nT \) as \( (X_n, Y_n) \).

4.1 The Objective

We use the long-run average cost as the performance objective in our setting. Thus, we are interested in optimizing the steady-state system performance. The
optimization problem is the following:

\[
\text{Find } \min_{\theta} J(\theta) \triangleq \lim_{n \to \infty} \frac{1}{n} \sum_{m=0}^{n-1} c(X_m)
\]
subject to
\[
G_{i,j}(\theta) \triangleq \lim_{n \to \infty} \frac{1}{n} \sum_{m=0}^{n-1} g_{i,j}(X_m) \leq 0, \\
\forall i = 1, \ldots, |C|, j = 1, \ldots, |P|,
\]
\[
H(\theta) \triangleq \lim_{n \to \infty} \frac{1}{n} \sum_{m=0}^{n-1} h(X_m) \leq 0.
\]

(1)

We assume below that the Markov process \{\(X_n, Y_n\)\} under any parameter \(\theta\) is ergodic. In such a case, the limits in (1) are well-defined. If this is not the case, one may replace the “lim” with “limsup” in the definitions of \(J(\theta), G_{i,j}(\theta)\) and \(H(\theta)\) in (1). Given the above constrained Markov cost process formulation, the optimization problem (1) essentially stipulates that the optimal worker parameter \(\theta^*\) should minimize the long-run average cost objective \(J(\cdot)\) while maintaining queue stability in steady-state (i.e., the long-run average of \(h(X_n)\) should not be above zero) and adhering to contractual SLAs, i.e., that the long-run average of \(g_{i,j}(X_n)\) should not be above zero, for any feasible \((i,j)\)-tuple.

The SASOC algorithms that we design subsequently (see Section 6) use the cost \(c(X_n)\) and constraint functions \(g_{i,j}(X_n), h(X_n)\) to tune the worker parameter \(\theta_n\) at instant \(nT\) and the system simulation would now continue with the updated worker parameter. While it is desirable to find the optimum \(\theta^* \in S\), i.e.,

\[
\theta^* = \arg\min \left\{ J(\theta) \text{ s.t. } \theta \in D, G_{i,j}(\theta) \leq 0, i = 1, \ldots, |C|, j = 1, \ldots, |P|, H(\theta) \leq 0 \right\},
\]

it is in general very difficult to achieve a global minimum. We apply the Lagrange relaxation procedure to the above problem and then provide SPSA based algorithms - both first as well as second order, for finding a locally optimum parameter \(\theta^*\). We now describe in detail the state, single stage cost and constraint functions that we adopt for the constrained hidden Markov cost process formulated for optimizing the staffing in the context of service systems.

4.2 State, Cost and Constraints

The observed part \(X_n\) of the state at instant \(n\) is the vector of the length of waiting SR queues corresponding to each skill level, the current utilization of workers for
Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

each shift and skill level, and the current SLA attainments for each customer and SR priority. The un-observed or hidden part $Y_n$ is the residual service times, i.e., the pending time to service completion of SRs which are being worked upon by individual SWs. Thus,

$$X_n = (\mathcal{N}(n), u(n), \gamma'(n), q(n)), \quad (2)$$

$$Y_n = (\mathcal{R}(n)), \quad (3)$$

where,

- $\mathcal{N}(n) = (\mathcal{N}_1(n), \ldots, \mathcal{N}_{|\mathcal{B}|}(n))^T$, where $\mathcal{N}_i(n)$ denotes the number of SRs in the system queue corresponding to skill level $i \in \mathcal{B}$.

- $\mathcal{R}(n) = (\mathcal{R}_{1,1}(n), \ldots, \mathcal{R}_{1,|\mathcal{B}|,\max}(n), \ldots, \mathcal{R}_{|\mathcal{A}|,|\mathcal{B}|,\max}(n))$ is the vector of residual service times. Here, $\mathcal{R}_{i,j,k}(n)$ denotes the residual service time of the SR currently being processed by the $k$th worker in shift $i$ and of skill level $j$. Note that if there is no $k$th worker corresponding to the shift $i$ and skill level $j$, then $\mathcal{R}_{i,j,k} = \kappa$, where $\kappa$ is a special value used to signify the non-existence of a worker. Considering that the service times follow a truncated lognormal distribution in our setting, the residual service time at any point cannot be precisely estimated and hence, is part of the unobserved or hidden state component $Y_n$.

- The utilization vector $u(n) = (u_{1,1}(n), \ldots, u_{|\mathcal{A}|,|\mathcal{B}|}(n))$, where each $u_{i,j}(n) \in [0, 1]$ is the average utilization of the workers in shift $i$ and skill level $j$, at instant $n$.

- The SLA attainment vector $\gamma'(n) = (\gamma'_{1,1}(n), \ldots, \gamma'_{|\mathcal{C}|,|\mathcal{P}|}(n))$, where $\gamma'_{i,j}(n) \in [0, 1]$ denotes the SLA attainment for customer $i$ and priority $j$, at instant $n$.

- $q(n)$ is a single scalar (Boolean) variable that denotes the queue feasibility status of the system at instant $n$. In other words, $q(n)$ is false if the growth rate of the SR queues (for each complexity) is beyond a threshold and is true otherwise. We need $q(n)$ to ensure system steady-state which is independent of SLA attainments because the latter are computed only on the SRs that were completed and not on those queued up in the system.

Let $S$ denote the state space. We observe that $S$ is a compact set. This is because each of the state components in $X_n$ and $Y_n$ are closed and bounded. In particular, each element of $u(n), \gamma'(n)$ takes values in $[0, 1]$ and $0 \leq q(n) \leq 1$. The system SR queues $\mathcal{N}$ are also of finite length and hence, $X_n$ is bounded. The residual
time vector in \( Y_n \) also takes values in a compact set in lieu of the fact that each element of \( \mathcal{R} \) is upper bounded by the total service times at the SR queues and that in turn takes values in \([0, \top]\).

Considering that the queue lengths, utilizations and SLA attainments at instant \( n + 1 \) depend only on the state at instant \( n \), i.e., \( \{(X_n, Y_n)\} \), we observe that \( \{(X_n(\theta), Y_n(\theta)), n \geq 0\} \) is a constrained hidden Markov cost process for any given (fixed) parameter \( \theta \). We now describe in detail the single stage cost function, whose long-run average sum we try to optimize in (1). We let the cost function \( c(X_n) \) have the form:

\[
c(X_n) = r \times \left( 1 - \sum_{i=1}^{\vert A \vert} \sum_{j=1}^{\vert B \vert} \alpha_{i,j} \times u_{i,j}(n) \right) + s \times \left( \frac{\sum_{i=1}^{\vert C \vert} \sum_{j=1}^{\vert P \vert} |\gamma'_{i,j}(n) - \gamma_{i,j}|}{\vert C \vert \times \vert P \vert} \right),
\]

where \( r, s \geq 0 \) and \( r + s = 1 \). Further, \( 0 \leq \gamma_{i,j} \leq 1 \) denotes the contractual SLA for customer \( i \) and priority \( j \). The single stage cost function here is a linear function of the state and remains bounded. In fact, from (4), we observe that \( 0 \leq c(X_n) \leq 1 \). This is because \( u_{i,j}(n), \gamma_{i,j}, \gamma'_{i,j}(n) \in [0, 1] \) and each component in (4) is upperbounded by 1.

The cost function is designed to balance between two conflicting objectives of maximizing the utilization of workers and meeting the SLA requirements simultaneously. By the first component in (4), we seek to minimize the under-utilization of workers as it is more fine-grained and hence, allows tighter minimization in comparison to minimizing just the sum of workers across shifts and skill levels. The second component in (4) represents the over/under-achievement of SLAs, which is the distance between attained and the contractual SLAs. While the need for meeting the target SLAs motivates the under-achievement part in the second component, it is also necessary to minimize over-achievement of SLAs. This is because an over-achieved SLA, for instance meeting 100% instead of the target of 95% for a particular customer, requires more time and effort than necessary from some workers and no additional reward is obtained in this case.

Note that the first term in (4) uses a weighted sum of utilizations over workers from each shift and across each skill level. Further, the weights \( \alpha_{i,j} \) are fixed and not time-varying. Using historical data on SR arrivals, the percentage of workload arriving in each shift and for each skill level is obtained. These percentages decide the weights \( \alpha_{i,j} \) used in (4), that in turn satisfy

\[
0 \leq \alpha_{i,j} \leq 1, \text{ and } \sum_{i=1}^{\vert A \vert} \sum_{j=1}^{\vert B \vert} \alpha_{i,j} = 1,
\]
Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

for \( i = 1, 2, \ldots, |A| \), and \( j = 1, 2, \ldots, |B| \). This prioritization of workers helps in optimizing the worker set based on the given workload. For instance, if 70% of the SRs requiring low skill worker attention arrive in shift 1, then one may set \( \alpha_{1,0} = 0.7 \), in the cost function (4), where 0 denotes the low skill level index.

The single-stage constraint functions \( g_{i,j}(\cdot), h(\cdot), i = 1, \ldots, |C|, j = 1, \ldots, |P| \), are given by:

\[
\begin{align*}
g_{i,j}(X_n) &= \gamma_{i,j} - \gamma'_{i,j}(n), \quad \forall i = 1, \ldots, |C|, j = 1, \ldots, |P|, \\
h(X_n) &= 1 - q(n).
\end{align*}
\] (5)

Here (5) specifies that the attained SLA levels should be equal to or above the contractual SLA levels for each customer-priority tuple. Further, (6) ensures that the SR queues for each complexity in the system stay bounded. In the constrained optimization problem formulated below, we attempt to satisfy these constraints in the long-run average sense (see (1)).

The SASOC algorithms treat the parameter as continuous-valued and tune it accordingly. Let us denote this continuous version of the worker parameter by \( \bar{\theta} = (\bar{\theta}_1, \ldots, \bar{\theta}_N) \). Note that \( \bar{\theta}_i \in [0, W_{\text{max}}], i = 1, 2, \ldots, N \). We now design a smooth projection operator \( \Gamma \) that projects \( \bar{\theta} \) on to the discrete space \( D \) so that the same can be used performing the simulation of the service system. We call the \( \Gamma \)-operator as a generalized projection scheme as it lies in between a fully deterministic projection scheme based on mere rounding off and a completely randomized scheme, whereby depending on the value of \( \bar{\theta}_j \) (for any \( j = 1, \ldots, N \)) one can find a \( \theta^k_j \) and \( \theta^{k+1}_j \) with \( \theta^k_j < \theta^{k+1}_j, \theta^k_j, \theta^{k+1}_j \in \{0, 1, \ldots, W_{\text{max}}\} \) such that \( \theta^k_j \leq \bar{\theta}_j \leq \theta^{k+1}_j \). Then, one sets the corresponding discrete parameter as

\[
\theta_j = \begin{cases} 
\theta^{k+1}_j & \text{w.p. } \frac{\bar{\theta}_j - \theta^k_j}{\theta^{k+1}_j - \theta^k_j} \\
\theta^k_j & \text{w.p. } \frac{\theta^{k+1}_j - \bar{\theta}_j}{\theta^{k+1}_j - \theta^k_j}
\end{cases}
\] (7)

4.3 A Generalized Projection Operator

For any \( \bar{\theta} = (\bar{\theta}_1, \ldots, \bar{\theta}_N) \) with \( \bar{\theta}_j \in [0, W_{\text{max}}], j = 1, 2, \ldots, N \), we define a projection operator \( \Gamma(\bar{\theta}) = \Gamma_1(\bar{\theta}_1), \ldots, \Gamma_N(\bar{\theta}_N) \) as follows: Let \( \zeta > 0 \) be a fixed real number and \( \bar{\theta}_j \) be such that \( D^j \leq \bar{\theta}_j \leq D^{j+1} \), \( D^j < D^{j+1} \) for some \( D^j, D^{j+1} \in D \). Let us consider an interval of length 2\( \zeta \) around the midpoint of \( [D^j, D^{j+1}] \) and denote it as \( [\hat{D}_1, \hat{D}_2] \), where \( \hat{D}_1 = \frac{D^j + D^{j+1}}{2} - \zeta \) and \( \hat{D}_2 = \frac{D^j + D^{j+1}}{2} + \zeta \). Then,
\[ \Gamma_i(\bar{\theta}_i) \text{ for } \theta_i \in [\mathcal{D}^j, \bar{\mathcal{D}}_1] \cup [\bar{\mathcal{D}}_2, \mathcal{D}^{j+1}] \text{ is defined by} \]

\[
\Gamma_i(\bar{\theta}_i) = \begin{cases} 
0 & \text{if } \bar{\theta}_i < 0 \\
\mathcal{D}^j & \text{if } \bar{\theta}_i \leq \frac{\mathcal{D}^j + \mathcal{D}^{j+1}}{2} - \zeta \\
\mathcal{D}^{j+1} & \text{if } \bar{\theta}_i \geq \frac{\mathcal{D}^j + \mathcal{D}^{j+1}}{2} + \zeta \\
W_{\max} & \text{if } \bar{\theta}_i \geq W_{\max}.
\end{cases}
\]  

Further, \( \Gamma_i(\bar{\theta}_i) \) for \( \theta_i \in [\bar{\mathcal{D}}_1, \bar{\mathcal{D}}_2] \) is given by

\[
\Gamma_i(\bar{\theta}_i) = \begin{cases} 
\mathcal{D}^j & \text{w.p. } f(\frac{\bar{\mathcal{D}}_2 - \bar{\theta}_i}{2\zeta}) \\
\mathcal{D}^{j+1} & \text{w.p. } 1 - f(\frac{\bar{\mathcal{D}}_2 - \bar{\theta}_i}{2\zeta})
\end{cases}
\]

In the above, w.p. stands for with probability and \( f \) is any continuously differentiable function defined on \([0, 1]\) such that \( f(0) = 0 \) and \( f(1) = 1 \). Note that we deterministically project onto either \( \mathcal{D}^j \) or \( \mathcal{D}^{j+1} \) if \( \bar{\theta}_i \) is outside of the interval \([\bar{\mathcal{D}}_1, \bar{\mathcal{D}}_2]\). Further, for \( \theta_i \in [\bar{\mathcal{D}}_1, \bar{\mathcal{D}}_2] \), we project randomly using a smooth function \( f \). It is necessary to have a smooth projection operator to ensure convergence of our SASOC algorithms as opposed to a deterministic projection operator that would project \( \bar{\theta}_i \in [\mathcal{D}^j, \mathcal{D}^j + \mathcal{D}^{j+1}] \) to \( \mathcal{D}^j \) and \( \bar{\theta}_i \in [\mathcal{D}^j + \mathcal{D}^{j+1}, \mathcal{D}^{j+1}] \) to \( \mathcal{D}^{j+1} \). The problem with such a deterministic operator is that there is a jump at the midpoint and hence, when extended for any \( \bar{\theta} \) in the convex hull \( \bar{\mathcal{D}} \), the transition dynamics of the process \( \{(X_n, Y_n), n \geq 0\} \) is not necessarily continuously differentiable. In other words, a non-smooth projection operator does not allow us to mimic a continuous parameter system.

The SASOC algorithms that we present subsequently tune the worker parameter in the convex hull of \( \mathcal{D} \), denoted by \( \bar{\mathcal{D}} \), a set that can be defined as \( \bar{\mathcal{D}} = [0, W_{\max}]^N \). This idea has been used in [24] for an unconstrained discrete optimization problem. However, the projection operator used there was a fully randomized operator. The generalized projection scheme that we incorporate has the advantage that while it ensures that the transition dynamics of the parameter extended Markov process is smooth (as desired), it requires a lower computational effort because in a large portion of the parameter space (assuming \( \zeta \) is small), the projection operator is essentially deterministic.

We also require another projection operator \( \bar{\Gamma} \) that projects any \( \theta \in \mathbb{R}^N \) onto the set \( \bar{\mathcal{D}} \) and is defined as \( \bar{\Gamma}(\theta) = (\bar{\Gamma}_1(\theta_1), \ldots, \bar{\Gamma}_N(\theta_N)) \), where \( \bar{\Gamma}_i(\theta_i) = \min(0, \max(\theta_i, W_{\max})) \), \( i = 1, \ldots, N \). Thus, \( \bar{\Gamma}(\cdot) \) keeps the parameter updates within the set \( \bar{\mathcal{D}} \) and \( \Gamma(\cdot) \) projects them to the discrete set \( \mathcal{D} \). The projected updated are then used as the parameter values for conducting the simulation of the service system.
4.4 Assumptions

We now make the following standard assumptions:

(A1) The Markov chain \( \{(X_n(\theta), Y_n(\theta)), n \geq 0\} \) under a given dispatching policy and parameter \( \theta \) is ergodic.

(A2) The single-stage cost functions \( c(\cdot), g_{i,j}(\cdot) \) and \( h(\cdot) \) are all continuous. The long-run average cost \( J(\cdot) \) and constraint functions \( G_{i,j}(\cdot), H(\cdot) \) are twice continuously differentiable with bounded third derivative.

(A2') The single-stage cost functions \( c(\cdot), g_{i,j}(\cdot) \) and \( h(\cdot) \) are all continuous. The long-run average cost \( J(\cdot) \) and constraint functions \( G_{i,j}(\cdot), H(\cdot) \) are continuously differentiable with bounded second derivative.

(A3) The step-sizes \( \{a(n)\}, \{b(n)\} \) and \( \{d(n)\} \) satisfy

\[
\sum_n a(n) = \sum_n b(n) = \sum_n d(n) = \infty; \sum_n (a^2(n) + b^2(n) + d^2(n)) < \infty,
\]

\[
\frac{b(n)}{d(n)} \cdot \frac{a(n)}{b(n)} \to 0 \text{ as } n \to \infty.
\]

Assumption (A1) ensures that the process \( \{X_n\} \) is stable for any given \( \theta \) and ensures that the long-run averages of the single stage cost and constraint functions in (1) are well-defined. For the algorithms that follow, one amongst (A2) and (A2’) will be assumed. More specifically, (A2) will be assumed for Hessian based schemes, while (A2’) will be assumed for gradient approaches. (A2) and (A2’) are technical requirements needed to push through suitable Taylor’s arguments in order to prove the convergence of the algorithms. The first two conditions in (A3) are standard requirements for step-size sequences and the last part ensures the separation of time scales between the different recursions in SASOC algorithms discussed in detail in Section 5.

5 Solution Methodology

The constrained long-run average cost optimization problem (1) can be expressed using the standard Lagrange multiplier theory as an unconstrained optimization
Problem given below.

\[
\max_{\lambda} \min_{\theta} L(\theta, \lambda) \triangleq \lim_{n \to \infty} \frac{1}{n} \sum_{m=0}^{n-1} E \left\{ c(X_m) + \sum_{i=1}^{\vert C \vert} \sum_{j=1}^{\vert P \vert} \lambda_{i,j} g_{i,j}(X_m) + \lambda_f h(X_m) \right\}
\]

(10)

where \( \lambda_{i,j} \geq 0, \forall i = 1, \ldots, \vert C \vert, j = 1, \ldots, \vert P \vert \) represent the Lagrange multipliers corresponding to constraints \( g_{i,j}(\cdot) \) and \( \lambda_f \) represents the Lagrange multiplier for the constraint \( h(\cdot) \), in the optimization problem (1). Also, \( \lambda = (\lambda_{i,j}, \lambda_f, i = 1, \ldots, \vert C \vert, j = 1, \ldots, \vert P \vert)^T \). The function \( L(\theta, \lambda) \) is commonly referred to as the Lagrangian. An optimal \((\theta^*, \lambda^*)\) is a saddle point for the Lagrangian, i.e., \( L(\theta, \lambda^*) \geq L(\theta^*, \lambda^*) \geq L(\theta^*, \lambda) \). Thus, it is necessary to design an algorithm which descends in \( \theta \) and ascends in \( \lambda \) in order to find the optimum point. The simplest iterative procedure for this purpose would use the gradients of the Lagrangian with respect to \( \theta \) and \( \lambda \) to descend and ascend respectively. However, for the given system, the computation of gradient with respect to \( \theta \) would be intractable due to lack of a closed form expression of the Lagrangian. Thus, a simulation based algorithm is required. The above explanation suggests that an algorithm for computing an optimal \((\theta^*, \lambda^*)\) would need three stages in each of its iterations.

1. The inner-most stage which performs one or more simulations over several time steps and aggregates data, i.e., does the averaging of the single-stage cost and constraint functions \( c(\cdot), g_{i,j}(\cdot) \) and \( h(\cdot) \) for any given \( \theta \) and \( \lambda \) updates.

2. The next outer stage which estimates the gradient of the Lagrangian along \( \theta \) and updates \( \theta \) along a descent direction. This stage would perform several iterations for a given \( \lambda \) and find a good estimate of \( \theta \); and

3. The outer-most stage which updates the Lagrange multipliers \( \lambda \) along an ascent direction, using the converged values of the inner two loops.

The above three steps will have to be performed iteratively till the solution converges to a saddle point described previously. Note that the loops are nested in the sense that the loop in iteration (1) would be a subloop for iteration (2). Likewise, iteration (2) would be a subloop for iteration (3). Thus, in between two successive updates of an outer loop (iterations (2) or (3)), one would potentially have to wait for a long time for convergence of the inner loop procedure (iteration...
Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

(1) or iterations (1) and (2), respectively). This problem gets addressed by using simultaneous updates to all three stages in a stochastic recursive scheme but with different step-size schedules, the outer-most having the smallest while the inner-most having the largest of step-sizes. The resulting scheme is a multiple time-scale stochastic approximation algorithm [25, Chapter 6].

6 Our Algorithms

In a deterministic optimization setting (i.e., without stochastic noise), any negative descent algorithm for obtaining the minimum (in \( \theta \)) of the Lagrangian (10) (for a given \( \lambda \)) would have the form

\[
\theta(n + 1) = \Gamma(\theta(n) - \gamma_k H(\theta(n))^{-1} \nabla_{\theta} L(\theta, \lambda)),
\]

(11)

where \( H(\theta(n)) \) is a positive definite matrix and \( \gamma_k > 0 \) is a given step-size parameter. It is easy to see that \(-H(\theta(n))^{-1} \nabla L(\theta, \lambda)\) is a descent direction, since \( H(\theta(n))^{-1} \) is a positive definite matrix. Using the fact that \( L(\theta, \lambda) \) has bounded second derivatives (see (A2)) and with suitable assumptions on the step-sizes \( \gamma_k \), it would follow that (11) converges.

However, if the single stage cost function \( c(\cdot) \) and the constraint functions \( g_{i,j}(\cdot) \) and \( h(\cdot) \) are observable only via simulation or in other words, that \( \nabla_{\theta} L(\theta, \lambda) \) is not computable, then a stochastic approximation based algorithm for obtaining a saddle point of the Lagrangian (10) would update the worker parameter along a descent direction as follows:

\[
\theta(n + 1) = \Gamma(\theta(n) - \gamma_k H_n^{-1} h_n).
\]

(12)

In the above, \( h_n \) represents the estimate of the gradient and \( H_n \) represent the positive definite matrix used at update instant \( n \). The convergence of our algorithms, which have the form (12) is established by using a diminishing stepsize sequence (see (A3) below) and employing a biased estimate using SPSA of the gradient, with the bias vanishing asymptotically.

All the SASOC algorithms that we propose are noisy variants of (12) and use SPSA techniques to estimate the gradient of the Lagrangian w.r.t. \( \theta \). In addition, two of our algorithms, SASOC-H and SASOC-W, use SPSA techniques to generate a sequence of positive definite matrices. The three algorithms mainly differ in the choice of \( H_n \) and hence the descent direction:
1. **SASOC-G**: Here $\mathcal{H}_n = I$ (identity matrix) and hence, this algorithm tunes the worker parameter in the direction of the negative gradient. The gradient is estimated using a two-measurement version of SPSA.

2. **SASOC-H**: Here $\mathcal{H}_n = \nabla^2 L(\theta, \lambda)(n)$, the Hessian of $L$ w.r.t. $\theta(n)$ and hence, this uses a Newton update for optimizing the worker parameter. We use a two perturbation sequence version of SPSA to simultaneously estimate the gradient and the Hessian. For simplicity in the numerical experiments, as with [17], we implement the Jacobi variant of SASOC-H wherein $\mathcal{H}_n$ is a diagonal matrix with diagonal entries corresponding to the second-order partial derivatives (same as corresponding entries in the Hessian) but with all other elements set to zero.

3. **SASOC-W**: Here, as with SASOC-H, $\mathcal{H}_n$ is the Hessian of $L$. However, in this algorithm, the inverse of the Hessian matrix is tuned directly by using a procedure based on the Woodbury’s identity. In contrast, SASOC-H estimates the Hessian and performs a matrix inverse. We use the full Hessian in our experiments because of ease of computation here over the case of direct Hessian inversion.

Thus, SASOC-G is a first-order method performing only gradient estimation while the other two algorithms SASOC-H and SASOC-W, are second-order methods that estimate both the gradient and the Hessian.

The overall flow of all SASOC algorithms can be diagrammatically represented as in Fig. 3. Each iteration of the algorithm involves two simulations (each for a period $T$) - one with $\Gamma(\theta(n))$, i.e., the current estimate of the parameter projected using the generalized projection operator so that it takes values in the discrete set $\mathcal{D}$ and the other with the (projected) perturbed parameter, $\Gamma(\theta(n) + p(n))$. Note that the perturbation $p(n)$ is algorithm-specific. For SASOC-G, $p(n) = \delta\Delta(n)$ and for SASOC-H/W, $p(n) = \delta_1\Delta(n) + \delta_2\hat{\Delta}(n)$. The rationale behind the choice of $p(n)$ will be subsequently clarified when the individual SASOC algorithms are presented. In every stage of SASOC algorithms, the two simulations are carried out as shown in Fig. 3. Using the state values of the two simulations, $X(n)$ and $\hat{X}(n)$, the worker parameter $\theta$ is updated in an algorithm-specific manner. Algorithm 1 gives the structure of all three of our incremental update SASOC algorithms.

### 6.0.1 Algorithm structure
Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

Figure 3: Overall flow of the algorithm 1.

Algorithm 1  Skeleton of SASOC algorithms

Input:

- $R$, a large positive integer;
- $\theta_0$, initial parameter vector; $p(\cdot); \Delta; K \geq 1$
- UpdateRule(), the algorithm-specific update rule for the worker parameter $\theta$ and Lagrange multiplier $\lambda$.
- Simulate($\theta, T) \rightarrow X$, the simulator of the SS

Output: $\theta^* \overset{\Delta}{=} \Gamma(\theta(R))$.

\[
\theta \leftarrow \theta_0, \ n \leftarrow 1
\]

loop

\[
X \leftarrow \text{Simulate}(\Gamma(\theta(n)), T).
\]

\[
\hat{X} \leftarrow \text{Simulate}(\Gamma(\theta(n) + p(n)), T).
\]

UpdateRule().

\[
n \leftarrow n + 1
\]

if $n = R$ then

Terminate and output $\Gamma(\theta(R))$.

end if

end loop
6.1 SASOC-G Algorithm

SASOC-G is a three time-scale stochastic approximation algorithm that does primal descent using a two-measurement SPSA while performing dual ascent on the Lagrange multipliers.

6.1.1 SPSA based gradient estimate

Here, the gradient of the Lagrangian w.r.t. $\theta$ is obtained according to

$$\nabla_\theta L(\theta, \lambda) = \lim_{\delta \to 0} \mathbb{E} \left[ \left( \frac{L(\theta + \delta \Delta, \lambda) - L(\theta, \lambda)}{\delta} \right) \Delta^{-1} \right], \tag{13}$$

where $\Delta$ is a vector of perturbation random variables that are independent, zero-mean, $\pm$-valued and have the symmetric Bernoulli distribution. More general distributions on these random variables can be chosen as described in [6, 7]. In (13), $\Delta^{-1}$ represents element-wise inverse of the $\Delta$ vector. This is a one-sided estimate whose convergence is shown in [26, Lemma 1]. For a sufficiently large $K > 1$ and small $\delta > 0$, the estimate of the gradient can be approximated as follows:

$$\nabla_\theta L(\theta, \lambda) \approx \frac{1}{K} \sum_{i=1}^{K} \left[ \left( \frac{L(\theta + \delta \Delta(i), \lambda) - L(\theta, \lambda)}{\delta} \right) \Delta(i)^{-1} \right].$$

6.1.2 Update rule of SASOC-G

From the form of the gradient estimator, it is clear that the Lagrangian function would be needed to compute the gradient estimate. However, for our problem, obtaining a closed-form expression for the Lagrangian itself is an intractable task. We overcome this by running two simulations with parameters $\Gamma(\theta(n))$ and $\Gamma(\theta(n) + p(n))$. Here, $p(n) = \delta \Delta(n)$, choice motivated by the form of the gradient estimate in (13). Using the output of the two simulations, we estimate the quantities $L(\theta + \delta \Delta, \lambda)$ and $L(\theta, \lambda)$, respectively on the faster timescale. These estimates are in turn used to tune the worker parameter $\theta$ in the negative gradient descent direction. For $\lambda_{i,j}$ and $\lambda_f$, values of $g_{i,j}(\cdot)$ and $h(\cdot)$ respectively provide a stochastic ascent direction, proof of which will be given later in Theorem 7. Since maximization of the Lagrangian w.r.t. $\lambda_{i,j}$ and $\lambda_f$ represents the outer most step, these parameters are updated on the slowest time-scale. The overall update rule...
Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

for this scheme, SASOC-G, is as follows: For all \( n \geq 0 \),

\[
\theta_i(n+1) = \tilde{\Gamma}_i\left( \theta_i(n) + b(n) \left( \frac{L(nK)-L'(nK)}{\delta \Delta_i(n)} \right) \right),
\]

\( \forall i = 1, 2, \ldots, N \),

where for \( m = 0, 1, \ldots, K - 1 \),

\[
\bar{L}(nK + m + 1) = \bar{L}(nK + m) + d(n)(c(X_{nK+m}) + \sum_{i=1}^{[C]} \sum_{j=1}^{[P]} \lambda_{i,j}(nK)g_{i,j}(X_{nK+m}) + \lambda_f h(X_{nK+m}) - \bar{L}(nK + m)),
\]

\[
\bar{L}'(nK + m + 1) = \bar{L}'(nK + m) + d(n)(c(\hat{X}_{nK+m}) + \sum_{i=1}^{[C]} \sum_{j=1}^{[P]} \lambda_{i,j}(nK)g_{i,j}(\hat{X}_{nK+m}) + \lambda_f h(\hat{X}_{nK+m}) - \bar{L}'(nK + m)),
\]

\[
\lambda_{i,j}(n+1) = (\lambda_{i,j}(n) + a(n)g_{i,j}(X_n))^+, \forall i = 1, 2, \ldots, [C], j = 1, 2, \ldots, [P],
\]

\[
\lambda_f(n+1) = (\lambda_f(n) + a(n)h(X_n))^+,
\]

where

- \( K \geq 1 \) is a fixed parameter which controls the rate of update of \( \theta \) in relation to that of \( \bar{L} \) and \( \bar{L}' \). This parameter allows for accumulation of updates to \( \bar{L} \) and \( \bar{L}' \) for \( K \) iterations in between two successive \( \theta \) updates;

- \( X_m \) represents the state at iteration \( m \) from the simulation run with nominal parameter \( \theta_{[nK]} \) while \( \hat{X}_m \) represents the state at iteration \( m \) from the simulation run with perturbed parameter \( \theta_{[nK]} + \delta \Delta_{[nK]} \). Here \([n/K]\) denotes the integer portion of \( n/K \). For simplicity, hereafter we use \( \bar{\theta} \) to denote \( \theta_{[nK]} \) and \( \bar{\theta} + \delta \Delta \) to denote \( \theta_{[nK]} + \delta \Delta_{[nK]} \);

- \( \delta > 0 \) is a fixed perturbation control parameter while \( \Delta \) is a vector of perturbation random variables that are independent, zero-mean and have the symmetric Bernoulli distribution;

- The operator \( \tilde{\Gamma}(\cdot) \) ensures that the updated value for \( \theta \) stays within the convex hull \( \bar{D} \) and is defined in Section 4.3; and

- \( \bar{L} \) and \( \bar{L}' \) represent Lagrange estimates corresponding to \( \theta \) and \( \theta + \delta \Delta \) respectively. Thus, for each iteration, two simulations are carried out, one with the
normal parameter $\theta$ and the other with the perturbed parameter $\theta + \delta \Delta$, the results of which are used to update $\bar{L}$ and $\bar{L}'$.

We achieve separation of time-scales between the recursions of $\theta_i$, $\bar{L}$, $\bar{L}'$ and $\lambda$ via the difference in the step-sizes $a(n)$, $b(n)$ and $d(n)$ (see (A3)). The chosen step-sizes ensure that the recursions of Lagrange multipliers $\lambda_{i,j}$ proceed ‘slower’ in comparison to those of the worker parameter $\theta$, while the updates of the average cost - $\bar{L}$ and $\bar{L}'$ proceed the fastest.

6.2 SASOC-H Algorithm

This is a second-order algorithm for adaptive labour staffing which uses SPSA based techniques to estimate both the gradient and the Hessian. As discussed before, the overall algorithm structure is represented by Fig. 3 with $p(n) = \theta(n) + \delta_1 \Delta(n) + \delta_2 \hat{\Delta}(n)$ in this case. Thus, each iteration of the algorithm involves two simulations (each for a period $T$) - one with $\Gamma(\theta(n))$ and the other with $\Gamma(\theta(n) + \delta_1 \Delta(n) + \delta_2 \hat{\Delta}(n))$. As explained in the next section, the two perturbation sequences $\Delta$ and $\hat{\Delta}$ are used to simultaneously estimate the gradient and the Hessian of the Lagrangian w.r.t. $\theta$.

6.2.1 SPSA based simultaneous estimates for the gradient and the Hessian

Suppose the Lagrangian in (10) is twice differentiable w.r.t. $\theta$, then we can look at possible second order schemes for computing updates to $\theta$. If the Lagrangian (10) were a second-degree equation, then the exact solution for $\theta$ update to reach the minimum point would have been

$$\theta^* = \theta_0 - [\nabla^2_\theta L(\theta_0)]^{-1} \nabla_\theta L(\theta_0),$$

with $\theta_0$ the starting point, i.e.,

$$\theta^* = \theta_0 - [\nabla^2_\theta L(\theta_0)]^{-1} \nabla_\theta L(\theta_0),$$

would be the optimal parameter. For a higher-degree Lagrangian, the above solution can be used with a step-size parameter iteratively till convergence to an optimal $\theta^*$. Let $\Delta$ and $\hat{\Delta}$ be two independent vectors of perturbation random variables that are independent, zero-mean, $\pm$-valued and have the symmetric Bernoulli distribution. More general distributions for $\Delta$ and $\hat{\Delta}$ may however be used, see [6, 7]. We use the following estimates for the gradient and Hessian, respectively, as described in [20, Section 3.2.1]:

$$\nabla_\theta L(\theta, \lambda) = \lim_{\delta_1, \delta_2 \to 0} E \left[ \left( \frac{L(\theta + \delta_1 \Delta + \delta_2 \hat{\Delta}, \lambda) - L(\theta, \lambda)}{\delta_2} \right) \hat{\Delta}^{-1} \right].$$
Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

$$\nabla^2_\theta L(\theta, \lambda) = \lim_{\delta_1, \delta_2 \downarrow 0} E \left[ \Delta^{-1} \left( \frac{L(\theta + \delta_1 \Delta + \delta_2 \hat{\Delta}, \lambda) - L(\theta, \lambda)}{\delta_1 \delta_2} \right)(\hat{\Delta}^{-1})^T \right],$$

where $\Delta^{-1}$ and $\hat{\Delta}^{-1}$ represent element-wise inverses of the $\Delta$ and $\hat{\Delta}$ vectors respectively. Thus, the inner terms of the above two expectations can be used for estimating the Hessian and also updating $\theta$.

### 6.2.2 Update rule of SASOC-H

Following the rest as in the previous algorithm, we propose the update rule of SASOC-H, using two perturbation sequences. For $n \geq 0$, we have

$$\theta_i(n+1) = \hat{\Gamma}_i \left( \theta_i(n) + b(n) \sum_{j=1}^N M_{i,j}(n) \left( \frac{\bar{L}(nK) - \bar{L}'(nK)}{\delta_2 \hat{\Delta}(n)} \right) \right),$$

$$H_{i,j}(n+1) = H_{i,j}(n) + b(n) \left( \frac{\bar{L}'(nK) - \bar{L}(nK)}{\delta_1 \Delta(n) \delta_2 \hat{\Delta}(n)} - H_{i,j}(n) \right),$$

for $i, j = 1, \ldots, N$.

- The update equations corresponding to $\bar{L}, \bar{L}', \lambda_{i,j}, i = 1, \ldots, |C|, j = 1, \ldots, |P|$ and $\lambda_f$ are the same as in SASOC-G (14). However, note that the perturbed parameter in this case is $(\theta(n) + \delta_1 \Delta + \delta_2 \hat{\Delta})$. Thus, unlike SASOC-G, $\hat{X}_m$ represents the state at iteration $m$ from the simulation run with perturbed parameter $\Gamma(\theta(n) + \delta_1 \Delta + \delta_2 \hat{\Delta})$, while $X_m$ continues to have the same interpretation as in the case of SASOC-G.

- $\delta_1, \delta_2 > 0$ are fixed perturbation control parameters while $\Delta$ and $\hat{\Delta}$ are two independent vectors of perturbation random variables that are independent, zero-mean, $\pm$-valued, and have the symmetric Bernoulli distribution;

- $H = [H_{i,j}]_{i=1,j=1}^{|A| \times |B|, |A| \times |B|}$ represents the Hessian (second-derivative w.r.t. $\theta$) estimate of the Lagrangian. $H(0)$ is a positive definite and symmetric matrix. We let $H(0) = cI$, with $c > 0$ and $I$ being the identity matrix; and

- $M(n) = \Upsilon(H(n))^{-1} = [M(n)_{i,j}]_{i=1,j=1}^{|A| \times |B|, |A| \times |B|}$ represents the inverse of the Hessian estimate $H$ of the Lagrangian, where $\Upsilon(\cdot)$ is a projection operator ensuring that the Hessian estimates remain symmetric and positive definite. The $\Upsilon$ operation is assumed to satisfy assumption (A4).
Assumption (A4)
The projection operator $\Upsilon(\cdot)$ projects a square matrix to a symmetric positive
definite matrix. If $\{A_n\}$ and $\{B_n\}$ are sequences of matrices in $\mathcal{R}^{N \times N}$ such
that $\lim_{n \to \infty} \|A_n - B_n\| = 0$, then $\lim_{n \to \infty} \|\Upsilon(A_n) - \Upsilon(B_n)\| = 0$ as well. Fur-
ther, for any sequence $\{C_n\}$ of matrices in $\mathcal{R}^{N \times N}$, if $\sup_n \|C_n\| < \infty$, then
$\sup_n \|\Upsilon(C_n)\| < \infty$ and $\sup_n \|\{\Upsilon(C_n)\}^{-1}\| < \infty$, as well.

6.3 SASOC-W Algorithm

The SASOC-H algorithm is more robust than SASOC-G. However, it requires
computation of the inverse of the Hessian $H$ at each stage which is a computa-
tionally intensive operation. We propose a modification using Woodbury’s identity to
the previous algorithm which makes updates less computationally intensive.

6.3.1 Woodbury’s Identity

In particular, an application of Woodbury’s identity brings down the computa-
tional complexity from $O(n^3)^2$ to $O(n^2)$ where $n = |A| \times |B|$. Woodbury’s
identity states that

$$(A + BCD)^{-1} = A^{-1} - A^{-1}B\left(C^{-1} + DA^{-1}B^{-1}\right)^{-1}DA^{-1}$$

where $A$ and $C$ are invertible square matrices and $B$ and $D$ are rectangular ma-
trices of appropriate sizes. The Hessian update in (15) without projection can be
rewritten as

$$H(n + 1) = (1 - b(n))H(n) + P(n)Z(nK)Q(n)$$

where

$$P(n) = \frac{1}{\delta_1} \left[ \frac{1}{\Delta_1(n)}, \frac{1}{\Delta_2(n)}, \ldots, \frac{1}{\Delta_{|A| \times |B|}(n)} \right]^T, Q(n) = \frac{1}{\delta_2} \left[ \frac{1}{\Omega_1(n)}, \frac{1}{\Omega_2(n)}, \ldots, \frac{1}{\Omega_{|A| \times |B|}(n)} \right]$$

and

$$Z(nK) = b(n) \left( \hat{L}'(nK) - \bar{L}(nK) \right).$$

---

2The popular Gauss-Jordan procedure for matrix inverse of a matrix is $O(n^3)$. 
Now, applying Woodbury’s identity to $H(n + 1)^{-1} = M(n + 1)$, gives us
\[
M(n+1) = \left( \frac{M(n)}{1 - b(n)} - \frac{b(n) \left( \bar{L}'(nK) - \bar{L}(nK) \right) P(n)Q(n)M(n)}{1 - b(n) + b(n) \left( \bar{L}'(nK) - \bar{L}(nK) \right) Q(n)M(n)P(n)} \right),
\]
which is a recursive update rule for directly updating the matrix $M(n)$ which is the inverse of $H(n), n \geq 0$.

### 6.3.2 Update rule of SASOC-W

The modified update scheme, named SASOC-W, is given below. For $n \geq 0$,
\[
\theta_i(n + 1) = \bar{\Gamma}_i \left( \theta_i(n) + b(n) \sum_{j=1}^{N} M_{i,j}(n) \left( \frac{\bar{L}(nK) - \bar{L}'(nK)}{\delta_1 \Delta_j(n)} \right) \right),
\]
(17)
\[
M(n + 1) = \Upsilon \left( \frac{M(n)}{1 - b(n)} - \frac{b(n) \left( \bar{L}'(nK) - \bar{L}(nK) \right) P(n)Q(n)M(n)}{1 - b(n) + b(n) \left( \bar{L}'(nK) - \bar{L}(nK) \right) Q(n)M(n)P(n)} \right).
\]

The rest of the update rule corresponding to $\bar{L}, \bar{L}', \lambda_{i,j}, i = 1, \ldots, |C|, j = 1, \ldots, |P|$ and $\lambda_f$ are the same as in SASOC-G/H. In the above, $M(0)$ is initialized to $cI$, $I$ being an identity matrix and $c > 0$.

### 6.3.3 Advantages of SASOC-W

The SASOC-W algorithm is computationally better than SASOC-H though it retains its robustness. Our SASOC algorithms differ from the algorithms of [20] in the following ways: (i) We allow the average cost estimates $\bar{L}$ and $\bar{L}'$ to accumulate and update the worker parameters $W_i$ and Lagrange multipliers $\lambda_{i,j}$ every $K$ instants. (ii) Unlike [20], we do not need to estimate the average cost for the constraint functions $G_{i,j}(\theta)$ and $H(\theta)$. Instead, we directly use the sample $g_{i,j}(\cdot)$ for performing gradient ascent in Lagrange multipliers. (iii) Since the SASOC-W algorithm does not involve explicit computation of the Hessian inverse, it is computationally more efficient than the second-order algorithms of [20]. The experiments considered in [20] incorporate the Jacobi version of the algorithms, i.e., estimates of only the second-order partials along the diagonal of the Hessian while setting all other entries to zero. By incorporating full Hessian information, we are able to improve the efficiency of the algorithm in finding an optimal parameter while at the same time reduce computational requirements through the use of Woodbury’s identity based procedure.
7 Simulation Experiments

We use the simulation framework developed in [5] for implementing all our algorithms. A number of dispatching policies have been developed in [5]. In particular, we study the PRIO-PULL and EDF policies for performance comparisons of the various algorithms. We implemented the following labor staffing algorithms:

- **SASOC-G**: This is a first order method that estimates $\nabla L(\theta, \lambda)$ using SPSA and is described in Section 6.1.

- **SASOC-H**: This is a second order Newton method that involves an explicit inverse of the Hessian matrix and is described in Section 6.2.

- **SASOC-W**: This is a second order Newton method, which unlike SASOC-H does away with the inversion of the Hessian matrix and leverages the Woodbury identity in order to estimate the inverse of the Hessian directly. This algorithm is described in Section 6.3.

- **OptQuest**: This is an algorithm that uses the state-of-the-art optimization toolkit OptQuest. In particular, we used the scatter search based variant of OptQuest for our experiments.

OptQuest employs an array of techniques including scatter and tabu search, genetic algorithms, and other meta-heuristics for the purpose of optimization and
Figure 5: Work arrival patterns over a week for each SS
is quite well-known as a tool for solving simulation optimization problems [23].
OptQuest along with several other engines from Frontline Systems won the INFORMS impact award\(^3\) in the year 2010.

We choose five real-life SS from two different countries providing server support to IBM’s customers. The five SS cover a variety of characteristics such as high vs. low workload, small vs. large number of customers to be supported, small vs. big staffing levels, and stringent vs. lenient SLA constraints. Collectively, these five SS staff more than 200 SWs with 40%, 30%, and 30% of them having low, medium, and high skill level, respectively. Also, these SS support more than 30 customers each, who make more than 6500 SRs every week with each customer having a distinct pattern of arrival depending on its business hours and seasonality of business domain. Fig 4(a) shows the total work hours per SW per day for each of the SS. The bottom part of the bars denotes customer SR work, i.e., the SRs raised by the customers whereas the top part of the bars denotes internal SR work, i.e., the SRs raised internally for overhead work such as meetings, report generation, and HR activities. This segregation is important because the SLAs apply only to customer SRs. Internal SRs do not have deadlines but they may contribute to queue growth. Note that while average work volumes are significant, they may not directly correlate to SLA attainment. Fig 4(b) shows the effort data, i.e., the mean time taken to resolve an SR (a lognormal distributed random variable in our setting) across priority and complexity classes. As shown in Figs 5, the arrival rates for SS4 and SS5 show much higher peaks than SS1, SS2, and SS3, although their average work volumes are comparable. The variations are significant because during the peak periods, many SRs may miss their SLA deadlines and influence the optimal staffing result.

We implemented our SASOC algorithms by invoking the simulation framework from [5] for both the perturbed and the unperturbed simulations (See \(X\) and \(\bar{X}\) computations in Algorithms 1 and ??). For our SASOC algorithms, the simulations were conducted for 1000 iterations, with each iteration having 20 simulation replications - ten each with unperturbed parameter \(\theta\) and perturbed parameter \(\theta + \delta \Delta\), respectively. Each replication simulated the operations of the respective SS for a 30 day period. Thus, we set \(R = 1000\) and \(K = 10\) for SASOC algorithms. On the other hand, for the OptQuest algorithm, simulations were conducted for 5000 iterations, with each iteration of 100 replications of the SS.

For all the SASOC algorithms, we set the weights in the single-stage cost function \(c(X_m)\), see (4), as \(r = s = 0.5\). We thus give equal weightage to both

\(^3\)http://www.solver.com/press201008.htm
the worker utilization and the SLA over-achievement components. The Boolean variable $q$ used in the constraint (6) was set to false (i.e., infeasible) if the queues were found to grow by 1000% over a two-week period during simulation. Each of the experiments are run on a machine with dual core Intel 2.1 GHz processor and 3 GB RAM.

The $\Upsilon$ operator implemented for SASOC-W can be described as follows. Let $\hat{H}$ be the Hessian update which needs to be projected. The following sequence of operations represent this projection. (i) $\hat{H} \leftarrow \frac{1}{2}(\hat{H} + \hat{H}^T)$; (ii) Perform eigen decomposition on $\hat{H}$ to get all eigen-values and corresponding eigen-vectors; (iii) Project each eigen-value to $[\epsilon, \frac{1}{\epsilon}]$ where $1 > \epsilon > 0$. $\epsilon$ is chosen to be a small number so as to allow for larger range of values, but not too small to avoid singularity. The upper limit in the projection range is to avoid singularity of the inverse of the Hessian estimate; and (iv) Reconstruct $\hat{H}$ using the projected eigen-values but with same eigen vectors. The $\Upsilon$ operator in the case of SASOC-H with diagonal Hessian is one that simply projects each diagonal entry to $[\epsilon, \frac{1}{\epsilon}]$. It is easy to see that the $\Upsilon$ operator satisfies assumption (A4).

On each SS, we compare our SASOC algorithms with the OptQuest algorithm using $W_{\text{sum}}$ and mean utilization as the performance metrics. Here $W_{\text{sum}} \triangleq \sum_{i=1}^{|A|} \sum_{j=1}^{|B|} W_{i,j}$ is the sum of workers across shifts and skill levels. The mean utilization here refers to a weighted average of the utilization percentage achieved for each skill level, with the weights being the fraction of the workload corresponding to each skill level.

As evident in Figs. 5(a) and 5(b), the SS pools SS1, SS2 and SS3 are characterized by a flat SR arrival pattern, whereas SS4 and SS5 are characterized by a bursty SR arrival pattern. We present and analyze the results on these pools separately, starting with the flat arrival pools in the next section.

7.1 Flat-Arrival SS pools

Figs. 6(a) and 7(a) compare the $W^*_{\text{sum}}$ achieved for OptQuest and SASOC algorithms using PRIO-PULL and EDF on three real life SS with a flat SR arrival pattern (see Fig. 5(a)). Here $W^*_{\text{sum}}$ denotes the value obtained upon convergence of $W_{\text{sum}}$. On these SS pools, namely SS1, SS2 and SS3, respectively, we observe that our SASOC algorithms find a better value of $W^*_{\text{sum}}$ as compared to OptQuest. Note in particular that on SS1, SASOC algorithms perform significantly better than OptQuest with an improvement of nearly 100%. Further, on SS2, OptQuest is seen to be infeasible whereas all the SASOC algorithms obtain a feasible and
It is evident that SASOC algorithms consistently outperform the OptQuest algorithm on these SS pools. Further, among the SASOC algorithms, we observe that SASOC-W finds better solutions in general as compared to the other two SASOC algorithms.

Fig. 7(a) presents similar results for the case of the EDF dispatching policy. The behavior of OptQuest and SASOC algorithms was found to be similar to that of PRIO-PULL with SASOC showing performance improvements over OptQuest here as well. We present the utilization percentages across different skill levels (low, medium and high) in Figs. 6(b) and 7(b). We observe mean utilization of workers is a crucial factor for a labor staffing algorithm and it is evident from Figs. 6(b) and 7(b) that SASOC algorithms exhibit a higher mean utilization of workers and hence, better overall performance in comparison to the OptQuest algorithm.
Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

Figure 7: Performance of OptQuest and SASOC for EDF dispatching policy on SS1, SS2 and SS3 (Note: OptQuest is infeasible over SS2). The mean utilization values have been rounded to nearest integer.

7.2 Bursty-Arrival SS pools

Figs. 8(a) and 9(a) compare the $W^*_\text{sum}$ achieved for OptQuest and SASOC algorithms using PRIO-PULL and EDF on two real life SS with a bursty SR arrival pattern (see Fig. 5(b)). The utilization percentages for these pools are presented in Figs. 8(b) and 9(b). From these performance plots, we observe that OptQuest is seen to be slightly better than SASOC-G and SASOC-W when the underlying dispatching policy is PRIO-PULL, whereas in the case of EDF dispatching policy, the SASOC algorithms clearly outperform OptQuest. The execution time advantage of SASOC algorithms over OptQuest hold in the case of these pools as well.

Observation 1 SASOC algorithms are computationally efficient and hence well-suited for adaptive labor staffing as compared to OptQuest.

Computational efficiency is a significant factor for any adaptive labor staffing algorithm. For instance, if a candidate labor staffing algorithm takes too long to
find the optimal staffing levels, it is not amenable for making staffing changes in a real SS. Both from the number of simulations required as well as the wall clock run time standpoints, SASOC algorithms are better than OptQuest. This is because OptQuest requires 5000 iterations with each iteration of 100 replications, whereas the SASOC algorithms require 1000 iterations of 20 replications each in order to find $W^*_{\text{sum}}$. This results in a $25X$ speedup for SASOC algorithms and also manifests in the wall clock runtimes of SASOC algorithms because simulation run-times are proportional to the number of SS simulations. We observe that the SASOC algorithms result in at least 10 to 15 times improvement as compared to OptQuest from the wall clock runtimes perspective. For instance, on SS1 the typical run-time of OptQuest was found to be 24 hours, whereas SASOC algorithms took less than 2.5 hours each to converge.

In fact, we observed in the case of SS2, OptQuest does not find a feasible solution even after repeated runs for 5000 search iterations. Also, because OptQuest depends heavily on SLA attainments and respective confidence intervals of previous iterations, it requires higher number of replications than SASOC. Further, we observed that SASOC algorithms converge within 500 iterations in all our ex-

Figure 8: Performance of OptQuest and SASOC for PRIO-PULL dispatching policy on SS4 and SS5

(a) $W^*_{\text{sum}}$ achieved

(b) Utilization of the workers across skill levels
Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

Figure 9: Performance of OptQuest and SASOC for EDF dispatching policy on SS4 and SS5. The mean utilization values have been rounded to nearest integer.

Experiments. Thus, SASOC algorithms require 50 times less number of simulations as compared to OptQuest, while searching for the optimal SS configuration. This runtime advantage ensures that an SS manager can make staffing changes even at the granularity of every week by making use of SASOC algorithms and the same may not be possible with OptQuest due to its longer runtimes.

Observation 2 The parameter vector $\theta$ converges to the optimum value in all SASOC algorithms.

We observe that the parameter $\theta$ (and hence $W_{sum}^*$) converges to the optimum value for each of the SS pools considered. This is illustrated by the convergence plots in Figs. 10(a) and 10(b). This is a significant feature of SASOC as we established convergence of our algorithm in section A and the plots confirm the same. In contrast, the OptQuest algorithm is not proven to converge to the optimum even after repeated runs, as illustrated in the case of SS2 in Fig 6(a).

From the above performance comparisons over SS pools with flat as well as bursty SR arrival patterns, it is evident that our SASOC algorithms show overall better performance in comparison with OptQuest. Among the SASOC algorithms,
we observe that the second order algorithms (SASOC-H and SASOC-W) perform better than the first order algorithm (SASOC-G) in many cases, with SASOC-W being marginally better than SASOC-H.

8 Conclusions

We motivated the discrete optimization problem of adaptively determining optimal staffing levels in SS. We presented three novel SASOC algorithms for optimizing staff allocation in the context of SS. We formulated the problem as a constrained hidden Markov cost process, with the aim of finding an optimum worker parameter that minimizes a certain long run cost objective, while adhering to a set of constraint functions, which are also long run averages. All SASOC algorithms are simulation based optimization methods as the single-stage cost and constraint functions are observable only via simulation and no closed form expressions are available. The single stage cost that we designed balanced the conflicting objectives of maximizing worker utilizations and minimizing the over-achievement of SLAs. For solving the constrained optimization problem, we applied the Lagrange relaxation procedure and used an SPSA based scheme for performing gradient descent in the primal and at the same time, ascent in the dual, for the Lagrange multipliers. All SASOC algorithms also incorporated a smooth projection operator that helped imitate a continuous parameter system with suit-
ably defined transition dynamics. Using the theory of multi-timescale stochastic approximation, we presented the convergence proof of our algorithms. Numerical experiments were performed to evaluate each of the algorithms based on real-life SS data against the state-of-the-art simulation optimization toolkit OptQuest in the context. SASOC algorithms in general showed overall superior performance compared to OptQuest, as they (a) exhibited more than an order of magnitude faster convergence than OptQuest, (b) consistently found solutions of good quality and in most cases better than those found by OptQuest, and (c) showed guaranteed convergence even in scenarios where OptQuest did not find feasibility even after repeated runs for 5000 iterations. Given the quick convergence of SASOC algorithms (in minutes), they are particularly suitable for adaptive labor staffing where a few days of optimization run like in OptQuest would fail to keep up with the changes. By comparing the results of the SASOC algorithms on two independent dispatching policies, we showed that SASOC’s performance is independent of the operational model of SS.

References


Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems


A Convergence analysis

Here we provide a sketch of the convergence of SASOC-G, SASOC-H and SASOC-W algorithms. The first step of the convergence analysis is common to all the SASOC algorithms and involves the extension of the transition dynamics $p_{\theta}(i,j)$ of the constrained parameterized hidden Markov process to the convex hull $\bar{D}$.

A.1 Extension of the transition dynamics $p_{i,j}(\theta)$

Recall that the discrete parameter $\theta$ of the Markov process $\{X_n(\theta), Y_n(\theta)\}$ takes values in the set $D$ defined earlier. Using the members of $D$, one can extend the transition dynamics $p_{\theta}(i,j)$ of the underlying Markov process to any $\theta$ in the convex hull $\bar{D}$ as follows:

$$
p_{\theta}(i,j) = \sum_{k=1}^{N} \beta_k(\theta)p_{D_k}(i,j), \quad \forall \theta \in \bar{D}, i,j \in S,
$$

where the weights $\beta_k(\theta)$ satisfy $0 \leq \beta_k(\theta) \leq 1, k = 1, \ldots, N$ and $\sum_{k=1}^{N} \beta_k(\theta) = 1$. $p_{\theta}(i,j), i,j \in S$ can be seen to satisfy the properties of transition probabilities. It is worth noting here that the mere existence of the weights $\beta_k(\theta)$ are necessary to ensure that the extended transition dynamics are smooth and our SASOC algorithms converge. In other words, in the SASOC algorithms we do not compute these weights while trying to solve the constrained optimization problem.
Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

(1). The weights \( \beta_k(\theta) \) can be obtained using the \( \Gamma \)-projection operator as follows: Consider the case when \( \theta = (\theta_1)^T \) and suppose \( \theta_1 \) lies between \( D_j \) and \( D_{j+1} \) (both members of \( \mathcal{D} \)). Let us consider an interval of length \( 2\zeta \) around the midpoint of \( [D_j, D_{j+1}] \) and denote it as \( [\tilde{D}_1, \tilde{D}_2] \), where \( \tilde{D}_1 = D_j + D_{j+1} - \zeta \) and \( \tilde{D}_2 = D_j + D_{j+1} + \zeta \). Then \( \beta_k(\theta) \) can be obtained as follows:

\[
(\beta_j(\theta_1), \beta_{j+1}(\theta_1)) = \begin{cases} 
(1, 0) & \text{if } \theta \in [D_j, \tilde{D}_1] \\
(f(\frac{D_j - \theta_1}{2\zeta}), 1 - f(\frac{D_j - \theta_1}{2\zeta})) & \text{if } \theta_1 \in [\tilde{D}_1, \tilde{D}_2] \\
(0, 1) & \text{if } \theta \in [\tilde{D}_2, D_{j+1}] 
\end{cases}
\]

In the above, \( f \) is obtained from the definition of \( \Gamma \)-projection and hence, is a continuously differentiable function defined on \( [0, 1] \) such that \( f(0) = 0 \) and \( f(1) = 1 \). The above can be similarly extended when the parameter \( \theta \) has \( N \) components.

We now claim the following:

**Lemma 1** For any \( \theta \in \mathcal{D}, i, j \in S \) and for any \( n \geq 1 \)

\[
p^n_\theta(i, j) \geq \sum_{k=1}^{N} \beta^\theta_k \beta^\theta_{Dk}(i, j).
\]

**Proof 1** Follows in a similar manner as Lemma 1 of [24].

**Lemma 2** For any \( \theta \in \mathcal{D}, \{X_n(\theta), Y_n(\theta), n \geq 1\} \) is ergodic Markov.

**Proof 2** Follows in a similar manner as Lemma 2 of [24].

Now, define analogues of the long-run average cost and constraint functions for any \( \theta \in \mathcal{D} \) as follows:

\[
\bar{J}(\theta) \triangleq \lim_{n \to \infty} \frac{1}{n} \sum_{m=0}^{n-1} c(X_m(\theta)), \theta \in \mathcal{D}
\]

\[
\bar{G}_{i,j}(\theta) \triangleq \lim_{n \to \infty} \frac{1}{n} \sum_{m=0}^{n-1} g_{i,j}(X_m(\theta)) \leq 0, \\
\forall i = 1, \ldots, |C|, j = 1, \ldots, |P|, \theta \in \mathcal{D} \tag{20}
\]

\[
\bar{H}(\theta) \triangleq \lim_{n \to \infty} \frac{1}{n} \sum_{m=0}^{n-1} h(X_m(\theta)) \leq 0, \theta \in \mathcal{D}.
\]
The difference between the above and the corresponding entities defined in (1) is that \( \theta \) can take values in \( \mathcal{D} \) in the above. In lieu of Lemma 2, the above limits are well-defined for all \( \theta \in \mathcal{D} \).

**Lemma 3** \( \bar{J}(\theta), \bar{G}_{i,j}(\theta), i = 1, \ldots, |C|, j = 1, \ldots, |P|, \) and \( \bar{H}(\theta) \) are continuously differentiable in \( \theta \in \mathcal{D} \).

**Proof 3** Follows in a similar manner as Lemma 3 of [24].

We now prove the SASOC algorithms described previously are equivalent to their analogous continuous parameter \( \bar{\theta} \) counterparts under the extended Markov process dynamics.

**Lemma 4** Under the extended dynamics \( p_\theta(i,j) \) for any \( \theta \in \mathcal{D} \) and \( i, j \in S \), we have

(i) SASOC-G algorithm is analogous to its continuous counterpart where \( \Gamma(\theta) \) and \( \Gamma(\theta + \delta \Delta) \) are replaced by \( \bar{\Gamma}(\theta) \) and \( \bar{\Gamma}(\theta + \delta \Delta) \) respectively.

(ii) SASOC-H and SASOC-W algorithms are analogous to their continuous counterpart where with \( \theta_i \) replaced by \( \bar{\theta}_i \).

**Proof 4** (i): Suppose the worker parameter \( \theta(n) \) at instant \( n \) lie in the interior of the convex hull \( \mathcal{D} \), which is denoted by \( \mathcal{D}^o \). Let \( \delta \) be such that \( \Gamma(\theta(n) + \delta \Delta(n)) = (\theta(n) + \delta \Delta(n)) \). Thus, \( \theta(n), (\theta(n) + \delta \Delta(n)) \in \mathcal{D}^o \). These are equal to \( \mathcal{D}^k \) with probabilities \( \beta_k(\theta) \) and \( \beta_k(\theta + \delta \Delta) \). Hence, with probabilities \( \beta_k(\theta) \) and \( \beta_k(\theta + \delta \Delta) \), the transition probabilities of the underlying Markov chain is equal \( p_{D^k}(i,j) \), \( i, j \in S \).

On the other hand, in the extended system with parameters \( \bar{\Gamma}(\theta) \) and \( \bar{\Gamma}(\theta + \delta \Delta) \), the transition probability is given by

\[
p_{\bar{\theta}}(i,j) = \sum_{k=1}^{N} \beta_k(\bar{\theta})p_{D^k}(i,j), \quad \forall \bar{\theta} \in \mathcal{D}, i, j \in S.
\]

Thus, with probability \( \beta_k(\bar{\theta}) \), a transition probability \( p_{D^k}(i,j) \) is used above. Hence, it can be seen that the two systems - the original in SASOC-G and the other with the extended dynamics - can be seen to be equivalent. A similar argument holds when \( \theta(n) \) is a boundary point i.e., \( \theta(n) \notin \mathcal{D}^o \).

(ii): Follows in a similar manner as part (i) above.
As a consequence of Lemma 4, we can analyze SASOC algorithms with the continuous parameter \( \bar{\theta} \) used in place of \( \theta \) and under the extended transition dynamics (18). By an abuse of notation, we shall henceforth use \( \theta \) to refer to the latter.

### A.2 SASOC-G

The convergence analysis of SASOC-G can be split into four stages:

1. The fastest time-scale in SASOC-G is \( \{d(n)\} \) which is used to update the Lagrangian estimates \( \bar{L} \) and \( \bar{L}' \) corresponding to simulations with \( \theta \) and \( \theta + \delta \Delta \) respectively. Firstly, we show that these estimates indeed converge to the Lagrangian values \( L(\theta, \lambda) \) and \( L(\theta + \delta \Delta, \lambda) \) defined in equation (10). Note that the \( \theta \) and \( \lambda \) which are updated on slower time-scales, can be assumed to be time invariant quantities for the purpose of analysis of these Lagrangian estimates.

2. Next, we show that the parameter updates \( \theta(n) \) using SASOC-G converge to a limit point of the ODE

\[
\dot{\theta}(t) = \Pi (-\nabla_{\theta} L(\theta(t), \lambda)),
\]

where \( \Pi \) is defined as follows: For any bounded continuous function \( \epsilon(\cdot) \),

\[
\Pi(\epsilon(\theta(t))) = \lim_{\eta \to 0} \frac{\Pi(\theta(t) + \eta \epsilon(\theta(t))) - \theta(t)}{\eta}.
\]

The projection operator \( \Pi(\cdot) \) ensures that the evolution of \( \theta \) stays within the bounded set \( M \). Again for the analysis of the \( \theta \)-update, the value of \( \lambda \) which is updated on the slowest time-scale is assumed constant.

3. We then show that \( \lambda_{i,j} \)'s and \( \lambda_f \) converge respectively to limit points of the ODEs

\[
\begin{align*}
\dot{\lambda}_{i,j}(t) &= \Gamma(G_{i,j}(\theta^*)) , \forall i = 1, 2, \ldots, |C|, j = 1, 2, \ldots, |P|, \\
\dot{\lambda}_f(t) &= \Gamma(H(\theta^*)),
\end{align*}
\]

where \( \theta^* \) is the converged parameter value of SASOC-G corresponding to Lagrange parameter \( \lambda(t) \overset{\Delta}{=} (\lambda_{i,j}(t), \lambda_f(t), i = 1, \ldots, |C|, j = 1, \ldots, |P|)^T \), and for any bounded continuous functions \( \tilde{\epsilon}(\cdot) \),

\[
\Gamma(\tilde{\epsilon}(\lambda(t))) = \lim_{\eta \to 0} \frac{\lambda(t) + \eta \tilde{\epsilon}(\lambda(t)))^+ - \lambda(t)}{\eta}.
\]
Here again, the projection operator $\Gamma$ ensures that the evolution of $\lambda$ stays non-negative. From the definition of the Lagrangian given in equation (10), the gradient of the Lagrangian w.r.t. $\lambda_{i,j}$ can be seen to be $G_{i,j}(\theta^*)$ and that w.r.t. $\lambda_f$ to be $H(\theta^*)$. Thus, the above ODEs suggest that in SASOC-G $\lambda_{i,j}$s and $\lambda_f$ are ascending in the Lagrangian value and converge to a local maximum point.

4. Finally, we show that the algorithm indeed converges to a saddle point of the Lagrangian with local maximum in $\lambda_{i,j}$s and $\lambda_f$, and local minimum in $\theta$.

**Lemma 5** $\|\bar{L}(n) - L(\theta(n), \lambda(n))\| \to 0$ w.p. 1, as $n \to \infty$.

**Proof 5** $\theta$ and $\lambda$ values are being updated on slower time-scales, thus assumed to be constant in this proof. Let

$$l(X_m) = c(X_{nK+m}) + \sum_{i=1}^{|C|} \sum_{j=1}^{|P|} \lambda_{i,j}(nK)g_{i,j}(X_{nK+m}) + \lambda_f h(X_{nK+m}).$$

The $\bar{L}$ update can be re-written as

$$\bar{L}(m+1) = \bar{L}(m) + d(m) \left( E[l(X_m)|\mathcal{F}_{m-1}] - \bar{L}(m) + M_{m+1} \right),$$

where $\mathcal{F}_m = \sigma(X_n, \lambda(n), \theta(n), n \leq m), m \geq 0$ are the associated $\sigma$-fields and $M_{m+1} = l(X_m) - E[l(X_m)|\mathcal{F}_{m-1}]$. Let $N_m = \sum_{n=0}^{m} d(n) M_{n+1}$. It can be easily verified that $(N_m, \mathcal{F}_m), m \geq 0$ is a square-integrable martingale obtained from the corresponding martingale difference $\{M_m\}$. Further, from the square summability of $d(n), n \geq 0$, and by martingale convergence theorem, $N_m, m \geq 0$, converges. The rest of the proof follows from the Hirsch lemma [27, Theorem 1, pp. 339].

On similar lines, $\|\bar{L}(n) - L(\theta(n) + \delta \Delta(n), \lambda(n))\| \to 0$ w.p. 1, as $n \to \infty$. Thus, $\theta$ updates which are on the slower time scale $\{b(n)\}$, can be re-written as

$$W_i(n+1) = W_i(n) - b(n) \left( \frac{L(\theta(n) + \delta \Delta_i(n), \lambda) - L(\theta(n), \lambda)}{\delta \Delta_i(n)} \right) + b(n) \chi_{n+1},$$

$$\forall i = 1, 2, \ldots, |A| \times |B|$$

where $\chi_n = o(1)$ in view of Lemma 5. Note here that $\lambda(n) \equiv \lambda \ \forall n$.

Now for the ODE (21), $V^\lambda(\cdot) = L(\cdot, \lambda)$ serves as an associated Lyapunov function and the stable fixed points of this ODE lie within the set $K^\lambda = \{\theta \in S : \bar{\Pi}(-\nabla L(\theta(t), \lambda)) = 0\}$.
**Theorem 6** Under (A1)-(A3), in the limit as $\delta \to 0$, $\theta(R) \to \theta^* \in K^\lambda$ almost surely as $R \to \infty$.

**Proof 6** From assumption (A2), $L(\theta, \lambda)$ is assumed to be continuous. Hence over the compact set $M$, $L(\theta, \lambda)$ is uniformly bounded. Thus, from Lasalle’s invariance theorem [28] [29, Theorem 2.3, pp. 76], $\theta(R) \to \theta^* \in K^\lambda$ a.s. as $R \to \infty$.

Thus (23) can be seen to be an Euler discretization of (21) and converges a.s. to $K^\lambda$ in the limit as $\delta \to 0$.

For $\{\lambda(n)\}$ updates on the slowest time-scale $\{a(n)\}$, we can assume that $\theta$ has converged to $\theta^* \in K^\lambda$. Let

$$F^{\theta^*} = \{\lambda \geq 0 : \Gamma (G_{i,j}(\theta^*)) = 0, \forall i = 1, 2, \ldots, |C|, j = 1, 2, \ldots, |P|; \Gamma (H(\theta^*)) = 0\}.$$ 

**Theorem 7** $\lambda(R) \to \lambda^* \in F$ w.p. 1 as $R \to \infty$.

**Proof 7** The $\lambda$ update in (14) can be re-written as

$$\lambda_{i,j}(n+1) = \lambda_{i,j}(n) + a(n) [G_{i,j}(\theta^*) + N_{n+1} + M_{n+1}],$\

where $N_{n+1} = E[g_{i,j}(X_n)|\mathcal{F}_{n-1}] - G_{i,j}(\theta^*), M_{n+1} = g_{i,j}(X_n) - E[g_{i,j}(X_n)|\mathcal{F}_{n-1}]$. It is easy to see that from assumption (A1), $N_n \to 0$ as $n \to \infty$. Further, $\{M_n\}$ is a martingale difference sequence with $\sum_{i=0}^n a(i) M_{i+1}, n \geq 0$, being the associated martingale that can be seen to be a.s. convergent (See Prop. 4.4 of [20]). Thus from [25, Extension 3 of Section 2.2], the result follows for $\lambda_{i,j}s$. Similarly, one can show convergence for $\lambda_f$.

Now, we need to show that the convergence of the algorithm is indeed to a saddle point i.e. $\theta^* \in K^\lambda^*$ and $\lambda^* \in F^{\theta^*}$. This can be shown by invoking the envelope theorem of mathematical economics [30, pp. 964-966]; see remark (2) in [20, pp 15].

### A.3 SASOC-H

Convergence analysis of SASOC-H follows along similar lines as that of the SASOC-G algorithm as given below.

1. Following the result of Lemma 5, one can see that $\bar{L}$ and $\bar{L}'$ iterations converge as follows:

$$\|\bar{L}(n) - L(\theta(n), \lambda(n))\| \to 0 \text{ w.p. 1},$$

$$\|\bar{L}'(n) - L(\theta(n) + \delta_1 \Delta(n) + \delta_2 \hat{\Delta}(n), \lambda(n))\| \to 0 \text{ w.p. 1},$$

as $n \to \infty$. 

2. Next, we show that the parameter updates $\theta(n)$ of SASOC-H converge to a limit point of the ODE

\[ \dot{\theta}(t) = \Pi \left( -\Upsilon(\nabla^2_{\theta}L(\theta(t), \lambda))^{-1} \nabla_{\theta}L(\theta(t), \lambda) \right), \]  

(24)

where $\Pi$ is as defined in equation (22).

3. The rest of the analysis of slower time-scale updates of $\lambda_{i,j}$s and $\lambda_f$, and saddle point behaviour follows from that of SASOC-G.

**Lemma 8**

\[ \left\| \frac{L(\theta(n) + \delta_1 \Delta(n) + \delta_2 \hat{\Delta}(n), \lambda(n)) - L(\theta(n))}{\delta_2 \hat{\Delta}_i(n)} - \nabla_{\theta_i}L(\theta(n), \lambda(n)) \right\| \rightarrow 0 \text{ w.p. } 1, \]

with $\delta_1, \delta_2 \rightarrow 0$ as $n \rightarrow \infty \; \forall i \in \{1, 2, \ldots, |A| \times |B|\}$.

**Proof 8** Follows from [20, Proposition 4.10].

**Lemma 9**

\[ \left\| \frac{L(\theta(n) + \delta_1 \Delta(n) + \delta_2 \hat{\Delta}(n), \lambda(n)) - L(\theta(n))}{\delta_1 \Delta_i(n)\delta_2 \hat{\Delta}_j(n)} - \nabla^2_{\theta_{i,j}}L(\theta(n), \lambda(n)) \right\| \rightarrow 0 \text{ w.p. } 1, \]

with $\delta_1, \delta_2 \rightarrow 0$ as $n \rightarrow \infty \; \forall i, j \in \{1, 2, \ldots, |A| \times |B|\}$.

**Proof 9** Follows from [20, Proposition 4.9].

**Lemma 10**

\[ \left\| H_{i,j}(n) - \nabla^2_{\theta_{i,j}}L(\theta(n), \lambda(n)) \right\| \rightarrow 0 \text{ w.p. } 1, \]

with $\delta_1, \delta_2 \rightarrow 0$ as $n \rightarrow \infty \; \forall i, j \in \{1, 2, \ldots, |A| \times |B|\}$.

**Proof 10** Follows from Lemma 9 applied to the Hessian update of SASOC-H.

**Lemma 11**

\[ \left\| M(n) - \Upsilon(\nabla^2_{\theta}L(\theta(n), \lambda(n)))^{-1} \right\| \rightarrow 0 \text{ w.p. } 1, \]

with $\delta_1, \delta_2 \rightarrow 0$ as $n \rightarrow \infty \; \forall i, j \in \{1, 2, \ldots, |A| \times |B|\}$.

**Proof 11** Follows from Lemma 10 and [18, Lemma A.9].
Simultaneous Perturbation Methods for Adaptive Labor Staffing in Service Systems

Let
\[ \bar{K}^\lambda = \left\{ \theta \in S : \frac{dL(\theta(t), \lambda)}{dt} = -\nabla_\theta L(\theta(t), \lambda)^T \Upsilon(\nabla^2_\theta L(\theta(t), \lambda))^{-1} \nabla_\theta L(\theta(t), \lambda) = 0 \right\}. \]

**Theorem 12** Under assumptions (A1)-(A4), in the limit as \( \delta_1, \delta_2 \to 0 \), \( \theta(R) \to \theta^* \in \bar{K}^\lambda \) almost surely as \( R \to \infty \).

**Proof 12** Following Lemmas 5, 8 and 11, with \( \delta_1, \delta_2 \to 0 \), the update of parameter \( \theta \) can be re-written in vector form as
\[ \theta_{n+1} = \Pi \left( \theta_n - b(n)\Upsilon(\nabla^2_\theta L(\theta(t), \lambda))^{-1} \nabla_\theta L(\theta(t), \lambda) + b(n)\chi_n \right) \]
with \( \chi_n = o(1) \). Thus, the update of parameter \( \theta \) can be viewed as a noisy Euler discretization of the ODE (24) using a standard approximation argument as in [31, pp. 191-196]. Note that \( V^\lambda(\cdot) = L(\cdot, \lambda) \) itself serves as the associated Lyapunov function [29, pp. 75] for the ODE (24) with stable limit points of the ODE lying within the set \( \bar{K}^\lambda \). From assumption (A2), \( L(\theta, \lambda) \) is assumed to be continuous. Hence over the compact set \( M, L(\theta, \lambda) \) is uniformly bounded. Thus, from Lasalle’s invariance theorem [28], \( \theta(n) \to \theta^* \in K^\lambda \) a.s. as \( n \to \infty \).

Rest of the analysis follows along similar lines as that of SASOC-G.

**A.4 SASOC-W**

Convergence analysis of SASOC-W follows from that of SASOC-H given the following lemma instead of Lemma 11.

**Lemma 13**
\[ \|M(n) - \Upsilon(\nabla^2_\theta L(\theta(n), \lambda(n)))^{-1}\| \to 0 \text{ w.p. } 1, \]
with \( \delta_1, \delta_2 \to 0 \) as \( n \to \infty \), \( \forall i, j \in \{1, 2, \ldots, |A| \times |B| \} \).

**Proof 13** From Woodbury’s identity, since \( M(n), n \geq 1 \) sequence of SASOC-W is identical to the \( \Upsilon(H(n))^{-1}, n \geq 1 \) sequence of SASOC-H, the result follows from Lemma 11.