Adaptive Smoothed Functional based Algorithms for Labor Cost Optimization in Service Systems

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

IISc-CSA-SSL-TR-2012-2


Stochastic Systems Lab
Computer Science and Automation
Indian Institute of Science, India

April 2012
Adaptive Smoothed Functional based Algorithms for Labor Cost Optimization in Service Systems

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

April 25, 2012

Abstract

Service systems are labor intensive with time varying workloads. The task of adapting the staffing levels to the workloads in such systems challenges while maintaining system steady-state and to comply with aggregate SLA (Service-Level Agreement) constraints, is non-trivial. We formulate this problem as a constrained parameterized Markov cost process and propose two multi-timescale smoothed functional (SF) based stochastic optimization algorithms: SASOC-SF-N and SASOC-SF-C, respectively for its solution. While SASOC-SF-N uses Gaussian based smoothed-functional, SASOC-SF-C uses Cauchy smoothed-functional for gradient estimation for primal descent. We validate these optimization schemes on five real-life service systems and compare them with a recent algorithm, SASOC-SPSA, from [1], and a state-of-the-art optimization tool-kit OptQuest. The performance of SASOC-SF-N is found to be comparable to that of SASOC-SPSA, while that of SASOC-SF-C is marginally better than SASOC-SPSA. Being an order of magnitude faster than OptQuest, our algorithms are particularly suitable for adaptive labor staffing. Also, we show that our algorithms guarantee convergence whereas OptQuest fails to find feasible solutions in

†Department of Computer Science and Automation, Indian Institute of Science {hlprasu,prashanth, shalabh}@csa.iisc.ernet.in,§IBM Research, Bangalore, INDIA {nirmit.desai}@in.ibm.com
some cases under reasonably high threshold on number of search iterations. We observe that our algorithms find better solutions than OptQuest in many cases and among our algorithms, SASOC-SF-C performs marginally better than SASOC-SF-N.

**Keywords:** Constrained optimization, Stochastic approximation, labor optimization

1 Introduction

Service-based economies and business models have gained significant importance. The clients and service providers exchange value through service interactions and reach service outcomes. Service requests of clients can vary greatly in terms of skills required to fulfill the requests, expected turn-around times, and the contexts of the clients’ business needs. As a result, service delivery is a labor-intensive business. Hence, for the service providers, optimization of labor costs is crucial.

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 [2, 3, 4]. This paper focuses on the SS in the data-center management domain. However, the contributions of this paper extend to all domains of services. In the domain of data-center management, the customers own data centers and other IT infrastructures supporting their businesses. The size, complexity, and uniqueness of the technology installations drive outsourcing of the management responsibilities to specialized service providers. The service providers manage the data-centers from remote locations called delivery centers where groups of service workers (SW) skilled in specific technology areas support corresponding service requests (SR). In each group, the processes, the people and the customers that drive the operations of a delivery center constitute a SS. A delivery center is a system of multiple SS.

Depending on the technology areas supported by SS, their operational models may differ significantly, e.g., managing storage infrastructures versus managing mainframes. However, a central component in these operational models is the policy for assigning SRs to SWs, called the dispatching policy. There are two fundamental challenges in this domain. First, given an SS with its operational characteristics, the staffing across skill levels and shifts needs to be optimized while maintaining steady-state and compliance to aggregate Service Level Agreement (SLA) constraints, for instance, an SLA constraint could specify that 95% of all urgent SRs in a month from a customer must be resolved in 4 hours. Note
that the aforementioned 4 hour deadline does not apply to all individual SRs, but to 95\% of those which arrive in a month from that customer. Second, because the SS characteristics such as work patterns, technologies and customers supported change frequently, the optimization of staffing needs to keep up with these changes.

This paper presents two stochastic optimization SASOC (Staff Allocation using Stochastic Optimization with Constraints) algorithms to address both of these challenges. Each of our algorithms applies a three-timescale stochastic approximation scheme that performs gradient descent in the primal and couples it with dual ascent for the Lagrange multipliers. Both the algorithms that we propose use a smoothed functional (SF) technique to estimate the gradient in the primal. This technique, in essence, involves convolving the gradient of the Lagrangian with a suitable distribution function that satisfies certain properties. The convolution with the distribution function is seen to smooth the objective (i.e., the Lagrangian). In our first algorithm that we propose, we use Gaussian or Normal as the smoothing distribution. Henceforth, we shall refer to this algorithm as SASOC-SF-N. On the other hand, the second algorithm uses a Cauchy distribution for smoothing and we shall refer to this algorithm as SASOC-SF-C. The smoothed functional approach was originally proposed by Katkovnik and Kulchitsky [5] where convolution with Gaussian density was explored. In [6], convolution with Cauchy distribution was proposed; however, the algorithm there required many simulations. A significant advantage with our SASOC-SF-C algorithm is that it requires only two simulations regardless of the parameter dimensions, unlike the Cauchy based SF algorithm proposed in [6]. The overall optimization scheme in all the algorithms has two main stages: search of a candidate solution followed by an evaluation of the same. In the evaluation step, we leverage the simulation-based operational models developed in [7] for two of the dispatching policies, namely, PRIO-PULL (basic priority scheme) and EDF (earliest deadline first). For the sake of comparison, we also implement the SASOC-SPSA algorithm from [1]. This algorithm uses a Simultaneous Perturbation Stochastic Approximation (SPSA) based gradient estimation for the primal problem.

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 historical work arrivals, time spent on the various types of activities, and contractual SLAs. Models of parameters such as arrival patterns and service times are computed based on these data and are supplied as inputs to the simulation model [7]. In comparison with the state-of-the-art OptQuest optimization toolkit [8], we find that both the SASOC algorithms that we propose (a) are 25 times faster than OptQuest,
and (b) find better solutions, resulting in lower labor costs than those found by OptQuest in majority of the SS, and (c) guarantee convergence even in scenarios where OptQuest does not find feasible solutions (at least until five thousand search iterations). Precisely due to the reasons of guaranteed convergence and a much lower computational time requirement than OptQuest, SASOC algorithms are better suited to address the two challenges highlighted above, especially that of adapting to the frequent changes in SS operational characteristics. We study the performance of our SASOC algorithms on two independent operational models corresponding to PRIO-PULL and EDF dispatching policies, respectively, and observe that the SASOC algorithms exhibit uniformly good performance even when the underlying dispatching policies change. Further, from the simulation experiments we find that SASOC-SF-C performs marginally better as compared to SASOC-SPSA and SASOC-SF-N.

1.1 Organization

The rest of the paper is organized as follows: Section 2 surveys the relevant literature. Section 3 describes the overall model of the service system, its components, policies and its operational flow. Building on the setting of service systems described in Section 3, a detailed mathematical formulation of the labor cost optimization problem is discussed in Section 4. Our smoothed functional algorithms (SASOC-SF-N and SASOC-SF-C) are proposed in Section 5 and their convergence analyses are given briefly in the Appendix. Section 6 provides the simulations results for five different service systems. The simulation results contain a comparison of the performance of our smoothed functional algorithms with SASOC-SPSA as well as OptQuest. For the sake of comparison, the SASOC-SPSA algorithm [1] is presented briefly in the beginning of Section 6. Section 7 provides the concluding remarks.

2 Literature Survey

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

In [9], a two step mixed-integer program is formulated for the problem of dispatching SRs within service systems. While their goal is to improve the dispatching policy with a given set of personnel, our objective is to reduce the number of personnel for a given dispatching policy. Further, unlike our framework, the SLA constraints, in the formulation considered in [9] cannot be aggregates. In [10], the authors propose a scheme for shift-scheduling in the context of third-level IT support system. Unlike us, they do not validate their method against data from real-life third-level IT support. In [11], 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 is modelled as an analytical M/M/1 queue. The simulation helps observe how cells die and neighborhood patterns emerge among cells. While [11] 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 [12], the 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 using an analytical model, which is possible in their simplified domain but does not work in the case of service systems that incorporate aggregate SLA constraints and dynamic policies such as preemption and swing impacting queues. They apply simulation to search for the optimal staffing level based on a heuristic. Simulation based methods for finding the optimal staffing in the context of a multiskill call center are proposed, for instance, in [13, 14]. While [13] proposes a cutting plane algorithm for solving an integer program, in [14] relies on solving a linear programming solution. However, the problem formulation in [13, 14] is for a single iteration of the system and they do not consider finding the optimal staffing level that minimizes a certain long run cost objective. Also, the solutions proposed in [13, 14] are heuristic (i.e., without proven convergence results) and further, they do not consider aggregate SLA and queue stability constraints. An analysis of service systems using the ARENA simulation tool is presented in [15]. Unlike our model, the system is not subjected to aggregate SLA constraints. Also, neither preemption nor swing policies are considered in their model. The model of [15] is slightly different from ours because they assume a pool of dedicated SWs for each customer as well as a separate pool of shared SWs. Nonetheless, they do not propose a new optimization technique either. In [7], a simulation framework for evaluating dispatching policies is proposed. While we share their simulation model, the goal in this paper is to propose a stochastic iterative scheme in the framework of constrained
Markov cost processes and prove their convergence to the optimal staffing levels in a SS. In general, none of the above papers propose an optimization algorithm that is geared for SS and that leverages simulation to adapt optimization search parameters, when both the objective and the constrained functions are suitable long-run averages.

In [1], an algorithm based on SPSA (Simultaneous Perturbation Stochastic Approximation) scheme [16], was proposed for the problem of staffing optimization in service systems. While our setting is similar to that in [1], we propose two new smoothed functional algorithms with proven convergence to the optimal staffing levels. Further, we provide detailed experimental results in comparison with the SASOC-SPSA algorithm in [1]. We observe in our experimental results our smoothed functional algorithm with Cauchy distributed perturbations (SASOC-SF-C) exhibits a superior performance overall in comparison to the algorithm in [1].

2.2 Stochastic Optimization

Gradient descent is a commonly used technique for finding an optimal parameter that locally minimizes a cost objective. In the absence of an analytical form for the cost objective, simulation based approaches become necessary. Simulation optimization methods usually work under the assumption of non-availability of any model information and can work with both real or simulated data. Popular approaches in this class of methods that estimate the gradient are SF (Smoothed Functional) and SPSA (Simultaneous Perturbation Stochastic Approximation) schemes. Both of these schemes estimate the gradient of the objective by generating simulations with randomly perturbed parameters.

SF schemes estimate the gradient by convolving the gradient function with a density that satisfies certain properties. Normal, Cauchy and uniform densities are known to be suitable for SF schemes. SF schemes were first proposed in [5] and they required one simulation. In [6], a two-simulation variant of SF was proposed and it was found to have lower variance when compared to the original one-simulation SF [5]. In [17], SF estimates of the Hessian were obtained and Newton based simulation optimization procedures using these estimates were derived. Newton based algorithms, even though more accurate than gradient based schemes, require significantly higher computational effort since they involve 1. projection of the Hessian update at each iteration to the set of positive definite and symmetric matrices and 2. inversion of the projected Hessian which is a costly operation at higher dimensions, after each update. Hence, in
Adaptive Smoothed Functional based Algorithms for Labor Cost Optimization in Service Systems

In this paper, we focus only on first-order SF based methods for finding the optimal staffing levels in a service system.

SPSA was first proposed in [16] and is based on the idea of randomly perturbing the parameter vector using i.i.d., symmetric, zero-mean random variables that satisfy a finite inverse moment bound. This algorithm has the critical advantage that it needs only two samples of the objective function for any finite $N$-dimensional parameter. In [18], a one-simulation variant of SPSA was proposed. However, unlike its two-simulation counterpart [16], the algorithm in [18] was not found to work as well in practice. Usage of Hadamard matrix based deterministic perturbations instead of randomized perturbations was proposed in [19] with the resulting one-simulation algorithm exhibiting considerably better performance over its random perturbation counterpart.

In [20], four simulation based algorithms for constrained optimization have been proposed. The Lagrange relaxation procedure is applied in [20] to the constrained optimization problem. Two of the algorithms proposed in [20] use SPSA for estimating the gradient of the Lagrangian while the other two use SF. 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-timescale 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. On the other hand, the algorithm proposed in [22] combines the ideas of multi-timescale stochastic approximation, reinforcement learning and function approximation to develop a simulation based algorithm that obtains an optimal policy for a constrained approximate MDP.

Our algorithms differ from the stochastic optimization approaches outlined above in various ways.

1. Smoothed functional concepts are originally from [5]. However, the original application in [5], of smoothing techniques to gradients were to find local minima (i) in non-differentiable functions, and (ii) in rapidly fluctuating multi-extremal functions. Only later in [23, 6], smoothing techniques were used for simulation based gradient estimation.

2. SASOC-SF-C is the first algorithm to use Cauchy based smoothing with only two simulations. While Cauchy based smoothing has been proposed previously in [6], the procedure in [6] requires several simulations to estimate the gradient. Also, [6] provides only an experimental validation for
the procedure proposed there whereas we provide a proof of convergence for SASOC-SF-C.

3. Many existing algorithms in the literature [24, 25, 17] are for unconstrained optimization whereas our labor optimization algorithms work with SLA and queue feasibility constraints, that are in fact certain long-run average cost functions.

4. Our algorithms differ from those in [20] in the way Lagrange multipliers are estimated and also the accumulation procedure for computing gradients.

3 The Setting

Fig. 1 shows the main components in the operational model of SS. SRs arrive from multiple customers supported by the SS and get classified and queued into high, medium, or low complexity queues by a queue manager (automatic or human). Also, depending on the dispatching policy in place, the SRs are assigned a priority in each of the complexity queues. SWs are grouped according to their skill level of high, medium, or low and work in shifts. Depending on the dispatching policy in place, the resource allocator (automatic or human) either pushes the SRs to SWs pro-actively or else the SWs pull the highest priority SR from the complexity queue when it becomes available. In the former case, each of the SWs
have an associated priority queue. Generally, SWs work on SRs with complexity matching to their skill levels. However, a swing policy may kick in dynamically and assign higher-skilled workers to lower complexity queues if the latter queues are growing. Finally, a preemption policy specifies the preemption action as well as preempted priority levels.

To capture the peak and off-peak work arrivals each week, the SR arrivals are assumed to follow a Poisson process with a rate of arrival that varies with the hour of the day as well as the day of the week. For example, from midnight to 1 am on Monday the rate may be 1 SR per hour, from 1am to 2am on Monday, the rate may change to 2.5 arrivals per hour, and so on. The arrival rate for each of the 168 hours of the week is computed based on the historical data on SR arrivals.

The stochastic variation in the time it takes to resolve an SR is modeled as a log-normal distribution with the mean and standard deviation parameter specified for each priority and complexity combination. Thus, the service time for all SRs having identical priority and complexity follows a unique lognormal distribution. The mean and standard deviation parameters are computed based on the data on actual effort spent on each of the SRs by each SW of an SS.

A SW works in exactly one shift (working days and times) and a SS may operate multiple shifts. We say that a SS configuration, i.e., a specification of the number of workers across shifts and skill levels, is feasible if it ensures that the SLA constraints are met and the complexity queues are not growing unbounded. While the need for SLA constraints to be met is obvious, the requirement for having bounded complexity queues is also necessary because SLA attainments are calculated only for work completed. For example, say in a given month, 100 SRs arrived at various times from a customer to a SS and 20 of them were completed. If 15 of these were completed in target time, the SLA attainment would be \( \frac{15}{20} \), i.e., 75%. The remaining 80 SRs are in progress without a known a completion time and hence do not impact the SLA attainment measures. Hence, a healthy SLA attainment of 75% alone does not provide a complete view into the health of the SS and queue growth fills in this gap.

4 Problem Formulation

We consider the problem of finding the optimal number of workers for each shift and of each skill level in a SS (see Fig. 1) for any given dispatching policy, while adhering to a set of SLA constraints. We formulate this as a constrained opti-
mization problem with the objective of minimizing the labor cost in the long run average sense, with the constraints being on SLA attainments and queue stability. The motivation for using the long-run average cost framework is to understand the steady-state system behavior. The underlying dispatching policy (that maps the service requests to the workers) is fixed and is in fact, parameterized by the set of workers. In essence, the problem is to find the ‘best’ parameter (set of workers) for a given dispatching policy.

We denote the worker parameter vector by \( \theta \), which is defined as follows:

\[
\theta = (W_1, \ldots, W_{|A| \times |B|})^T \in \mathcal{R}^{|A| \times |B|}.
\]

In the above,

- \( A \) denotes the set of shifts of the workers and \( B \) the set of worker skill levels.
- \( W_i \) indicates the number of service workers whose skill level is \((i - 1)\% |B|\) and whose shift index is \((i - 1)/|B|\).

We let the parameter vector \( \theta \in \mathcal{R}^{|A| \times |B|} \) take values in a compact set \( S \triangleq [0, W_{\text{max}}]^{|A| \times |B|} \).

Table 1 illustrates a simple SS configuration, specifying the staffing levels across shifts and skill levels. This essentially constitutes the worker parameter that we are trying to optimize. For instance, for the SS corresponding with the worker distributed as in Table 1, \( A = \{\text{S1, S2, S3}\} \) and \( B = \{\text{high, medium, low}\} \) with S1 corresponding to the 0th index into \( A \) and ‘high’ the same for \( B \). The worker parameter for this setting is then given by \( \theta = (W_1, \ldots, W_0)^T = (1, 3, 7, 0, 5, 2, 3, 1, 2)^T \).

The system evolves probabilistically over states as a constrained parameterized Markov cost process, with each system transition from one state to another illustrated in Fig 2. In essence, we continue the simulation of the service system for a fixed period \( D \) with the current worker parameter \( \theta_n \). \( D \) is chosen based on the period over which the contractual SLAs are evaluated by the customers. In
Table 1: Workers $W_i$

<table>
<thead>
<tr>
<th>Skill levels</th>
<th>Shift</th>
<th>High</th>
<th>Med</th>
<th>Low</th>
</tr>
</thead>
<tbody>
<tr>
<td>S1</td>
<td></td>
<td>1</td>
<td>3</td>
<td>7</td>
</tr>
<tr>
<td>S2</td>
<td></td>
<td>0</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>S3</td>
<td></td>
<td>3</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

The SASOC algorithms that we design subsequently (see Section 5) use the cost $c(X_n)$ to tune the worker parameter $\theta$ and the system simulation would now continue with a new worker parameter $\theta_{n+1}$. Note that the service system simulation is run continuously, but at discrete time instants $n, n + 1, \ldots$, we modify the worker parameter $\theta$ and use the subsequent cost output $c(X_n)$ to tune $\theta$. We now present a description of the states, costs and constraints of the underlying constrained Markov process below.

4.1 Constrained Markov Cost Process

The state $X_n$ at instant $n$ is the vector of the current utilization of workers for each shift and skill level, and the current SLA attainments for each customer and SR priority, and is given by

$$X_n = (u_{1,1}(n), \ldots, u_{|A||B|}(n), \gamma'_1,1(n), \ldots, \gamma'_{|C||P|}(n), q(n)).$$

In the above,

- $C$ denotes the set of all customers and $P$, the set of all possible priorities in the SS under consideration. Note that any arriving SR has a customer identifier and a priority identifier.

- $0 \leq u_{i,j}(n) \leq 1$ is the average utilization of the workers in shift $i$ and skill level $j$, at instant $n$.

- $0 \leq \gamma'_{i,j}(n) \leq 1$ denotes the SLA attainment for customer $i$ and priority $j$, at instant $n$. 
• $q(n)$ is a 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 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.

Table 2: Utilizations $u_{i,j}$

<table>
<thead>
<tr>
<th>Skill levels</th>
<th>Shift</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>High</td>
</tr>
<tr>
<td>S1</td>
<td>67%</td>
</tr>
<tr>
<td>S2</td>
<td>45%</td>
</tr>
<tr>
<td>S3</td>
<td>23%</td>
</tr>
</tbody>
</table>

Table 3: SLA targets $\gamma_{i,j}$

<table>
<thead>
<tr>
<th>Priority</th>
<th>Customers</th>
<th>Bossy Corp</th>
<th>Cool Inc</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td></td>
<td>95%4h</td>
<td>89%5h</td>
</tr>
<tr>
<td>$P_2$</td>
<td></td>
<td>95%8h</td>
<td>98%12h</td>
</tr>
<tr>
<td>$P_3$</td>
<td></td>
<td>100%24h</td>
<td>95%48h</td>
</tr>
<tr>
<td>$P_4$</td>
<td></td>
<td>100%18h</td>
<td>95%144h</td>
</tr>
</tbody>
</table>

Table 4: SLA attainments $\gamma'_{i,j}$

<table>
<thead>
<tr>
<th>Priority</th>
<th>Customers</th>
<th>Bossy Corp</th>
<th>Cool Inc</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_1$</td>
<td></td>
<td>98%4h</td>
<td>95%5h</td>
</tr>
<tr>
<td>$P_2$</td>
<td></td>
<td>98%8h</td>
<td>99%12h</td>
</tr>
<tr>
<td>$P_3$</td>
<td></td>
<td>89%24h</td>
<td>90%48h</td>
</tr>
<tr>
<td>$P_4$</td>
<td></td>
<td>92%18h</td>
<td>95%144h</td>
</tr>
</tbody>
</table>

Tables 2 and 4 provide sample utilizations and SLA attainments on a SS with three shifts, two customers and four priority levels. Table 3 illustrates the target SLA requirements for the same SS.
By a constrained Markov cost process, we mean an $\mathbb{R}^d$-valued process $\{X_n\}$ whose evolution depends on a parameter $\theta \in \mathbb{R}^N$. Also, given $\theta$, the process $\{X_n\}$ is Markov. Let $p_\theta(x,dy), x, y \in \mathbb{R}^d$, denote the transition kernel of $\{X_n\}$ where $\theta$ is a parameter. When $X_n = x$, an immediate, i.e., single-stage, cost $c(x)$ is incurred. In addition, there are additional single-stage cost functions, say, $g_1(x), \ldots, g_N(x)$, that determine whether or not certain functional constraints are met. The overall objective is to find a parameter $\theta$ that minimizes a long-term cost function that depends on the single-stage cost $c(\cdot)$ and that is subject to certain long-term constraints being satisfied. In this paper, we consider the long-term cost and constraint functions to be certain long-run averages whose precise form is described in Section 4.2.

We design the single stage cost function $c(X_n)$ to minimize (i) the under-utilization of workers across all shifts and various skill levels, and (ii) over/under-achievement of SLAs. Here, minimization of under-utilization of workers is equivalent to maximizing the utilization of workers in each shift. Instead of minimizing the under-utilization of workers, one could have had possibly considered minimizing just the sum of workers across shifts and skill levels. However, the quantity representing utilization of workers is more fine-grained allowing for tighter minimization. The over/under-achievement of SLAs is the distance between attained and the contractual SLAs. It is necessary that the SLAs attained need to meet the target or contractual SLAs and hence, the need for under-achievement in the cost function is obvious. However, if the SLAs are over-achieved, for instance, if a customer request that 95% of his high-priority SRs be closed with 4 hours and if this deadline is met 100% of the time, then it essentially translates to some worker(s) time and effort in meeting 100% for this particular customer. And, in our constrained setting, this is unnecessary as the customer would be happy if his SLAs are met and does not expect over-achievement. Hence, the cost function is designed to balance between two conflicting objectives as increasing the workers would lower their utilizations (first component) while meeting the SLAs (second component) and vice-versa.

The cost function $c(\cdot)$ has the form:

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

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 $j$. Note that the first term in (1) 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. The choice of these weights is
described in the Section 4.1.1.

### 4.1.1 Choice of weights $\alpha_{i,j}$:

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 (1), which satisfy

$$0 \leq \alpha_{i,j} \leq 1, \text{ and } \sum_{i=1}^{|A|} \sum_{j=1}^{|B|} \alpha_{i,j} = 1,$$

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 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 (1), where 0 denotes the low skill level index.

### 4.1.2 Single-stage constraints

We let $g_{i,j} (\cdot), h (\cdot), i = 1, \ldots, |C|, j = 1, \ldots, |P|$, denote the single-stage constraint functions. The constraints are on the SLA attainments and are given by:

$$g_{i,j} (X_n) = \gamma_{i,j} - \gamma'_{i,j} (n) \leq 0, \forall i = 1, \ldots, |C|, j = 1, \ldots, |P|,$$

$$h (X_n) = 1 - q(n) \leq 0.$$

Here (2) specifies that the attained SLA levels should be equal to or above the contractual SLA levels for each customer-priority tuple. Further, (3) 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 (4)).

### 4.2 Constrained Optimization

We want to find an optimal value for the parameter vector $\theta$ that minimizes the long-run average sum of single-stage costs $c(X_n)$ while maintaining queue stability in steady-state, i.e., the long-run average of $h(X_n)$ should not be above zero. Further, one requires compliance to contractual SLAs, i.e., that the long-run average of $g_{i,j} (X_n)$ should not be above zero as well, for any feasible $(i, j)$-tuple.
Our aim is to find a parameter $\theta$ that minimizes the long run average cost,

$$J(\theta) \triangleq \lim_{n \to \infty} \frac{1}{n} \sum_{m=0}^{n-1} E[c(X_m)]$$

subject to

$$G_{i,j}(\theta) \triangleq \lim_{n \to \infty} \frac{1}{n} \sum_{m=0}^{n-1} E[g_{i,j}(X_m)] \leq 0,$$

$$H(\theta) \triangleq \lim_{n \to \infty} \frac{1}{n} \sum_{m=0}^{n-1} E[h(X_m)] \leq 0. \tag{4}$$

As illustrated in Fig. 2, here each step from $n$ to $n+1$ indicates a state transition from $X_n$ to $X_{n+1}$, and incurs a cost of $c(X_n)$. Further, there are additional costs that constitute the functional constraints and correspond to $g_{i,j}(X_n), i = 1, \ldots, |C|, j = 1, \ldots, |P|$ and $h(X_n)$, respectively. The parameter $\theta$ determines the long-run average cost incurred and whether the constraints are met.

We now make the following standard assumptions:

**Assumption (A1)**

The Markov chain $\{X_n, n \geq 1\}$ under a given dispatching policy and parameter $\theta$ is ergodic.

This ensures that the long-run average cost and constraint functions in (4) are well-defined for any parameter $\theta$ under the given dispatching policy. It is also required to ensure that the process $\{X_n\}$ remains stable under the sequence $\{\theta_n\}$ of parameter updates, obtained in our algorithms. We make the following assumption on functions $c(\cdot), g_{i,j}(\cdot), h(\cdot)$ and $J(\cdot)$:

**Assumption (A2)**

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

This is a technical requirement for convergence. The latter part of (A2) is needed to push through a Taylor’s argument (See Appendix). Also, the long-run average cost $J(\cdot)$ itself would be shown to serve the role of Lyapunov function for ODE corresponding to the parameter updates, for which continuous differentiability of $J(\cdot)$ would be necessary.
While it is desirable to find the optimum $\theta^* \in S$, i.e.,

$$
\theta^* = \arg\min \left\{ J(\theta) \text{ s.t. } \theta \in S, \right\}
$$

it is in general very difficult to achieve a global minimum. We apply the Lagrange relaxation procedure to the above problem and then provide smoothed-functional stochastic optimization based algorithms for finding a locally optimum parameter $\theta^*$.

## 5 Our Algorithms

We derive in this section two smoothed-functional gradient based algorithms (SASOC-SF-N and SASOC-SF-C) for computing a locally optimal $\theta^*$ for the optimization problem (4). First, we formulate the Lagrangian of (4) in Section 5.1, that is then followed by a description of the general structure of our algorithms in Section 5.2. Next, in Section 5.3, we provide the Gaussian smoothed-functional gradient (SASOC-SF-N) algorithm and in Section 5.4, we provide the Cauchy smoothed-functional gradient (SASOC-SF-C) algorithm, respectively.

### 5.1 Formulation of the Lagrangian

The constrained long-run average cost optimization problem (4) can be expressed using the standard Lagrange multiplier theory as an unconstrained optimization problem given below.

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

In the above,

- $\lambda_{i,j} \geq 0$, $\forall i = 1, \ldots, \left|C\right|$, $j = 1, \ldots, \left|P\right|$ represent the Lagrange multipliers corresponding to constraints $g_{i,j}(\cdot)$ and
Adaptive Smoothed Functional based Algorithms for Labor Cost Optimization in Service Systems

- $\lambda_f \geq 0$, represents the Lagrange multiplier for the constraint $h(\cdot)$, in the optimization problem (4).

Let $\lambda \triangleq (\lambda_f, (\lambda_{i,j}, i = 1, \ldots, |C|, j = 1, \ldots, |P|))^T$. The function $L(\theta, \lambda)$ is commonly referred to as the Lagrangian. An optimal $(\theta^*, \lambda^*)$ is a saddle point in 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 $\lambda$ to find the optimum point. The simplest iterative procedure for this purpose would use the gradient of the Lagrangian with respect to $\theta$ and $\lambda$ to descend and ascend respectively. However, for the given system, computation of the gradient with respect to $\theta$ would be intractable due to lack of a closed form expression of the Lagrangian. This is because, for a given staffing level specified by $\theta$, the values of the cost function $c(x_n)$ and constraint functions $g_{i,j}(x_n)$ and $h(x_n)$ can only be observed via simulation. Thus, a simulation based stochastic optimization 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 continues the service system simulation for a period $D$;

2. The next outer stage which updates $\theta$ along a descent direction using simulation results of the inner most stage. This stage would perform several iterations for a given $\lambda$ in order to find the best value of $\theta$; and

3. The outer-most stage which computes the long-run average value of each constraint using the iterations in the inner two stages and updates the Lagrange multipliers $\lambda$ along an ascent direction.

The above three steps need to be performed iteratively till the solution converges to a saddle point described previously. The problem however is the computational difficulty in executing the whole procedure as one outer stage update would happen only after one full run of inner stages at both levels. Further, each run of an inner stage would typically proceed until convergence of the corresponding (inner-loop) procedure. This problem gets addressed by using simultaneous updates to all three stages in a stochastic recursive scheme but with different step-size schedules, with the recursion corresponding to the outer-most stage being driven by a step-size schedule that converges to zero the fastest, while the recursion in
the inner-most stage being driven by a step-size sequence that converges to zero the slowest. We show that in the asymptotic limit one obtains the desired convergence behaviour. This happens because there exists \( N_0 > 0 \), such that for all \( n \geq N_0 \), the step-sizes governing the outer-loop are uniformly smaller than those governing the inner-loop update. Thus, when viewed from the time-scale of the outer-loop recursions, the inner-loop procedure would appear to have converged. On the other hand, when viewed from the time-scale of the inner-loop recursion, the outer-loop procedure would appear to be quasi-static. The resulting scheme that we incorporate is a multiple time-scale stochastic approximation algorithm [26, Chapter 6].

5.2 Structure of SASOC Algorithms

We provide two smoothed-functional stochastic gradient algorithms for obtaining a saddle point of the Lagrangian (6). Both algorithms update the worker parameter \( \theta \) along the steepest descent direction and they update the Lagrange multipliers \( \lambda \) along steepest ascent direction. These algorithms have the form:

\[
\begin{align*}
\theta(n+1) &= \theta(n) - b(n) \nabla_\theta L(\theta(n), \lambda(n)), \\
\lambda(n+1) &= \lambda(n) + d(n) \nabla_\lambda L(\theta(n), \lambda(n)),
\end{align*}
\]

(7)

where \( \nabla_\theta L(\theta(n), \lambda(n)) \) and \( \nabla_\lambda L(\theta(n), \lambda(n)) \) represent the gradient of the Lagrangian \( L(\theta, \lambda) \) w.r.t. \( \theta \) and \( \lambda \) respectively. Further, \( b(n), d(n) > 0 \) are step-size sequences that satisfy Assumption (A3) (defined later). Strictly speaking, an additive noise term needs to be considered for both of the above iterations indicating that these algorithms perform a noisy estimate of the gradient based on simulation measurements. However, for simplicity, we have not shown the noise terms in (7). We use two simulations per iteration to estimate the gradient. The two algorithms differ in the way we use the two simulations to estimate the gradient. The two algorithms differ in the way we use the two simulations to estimate \( \nabla_\theta L(\theta, \lambda) \):

1. **SASOC-SF-N**: Here we use the smoothed functional approach to estimate the gradient for parameter tuning with Gaussian distributed perturbation random variables.

2. **SASOC-SF-C**: Here also we use the smoothed functional approach but with a multi-variate Cauchy random vector for smoothing. Note that Cauchy distribution is heavy tailed in comparison to Gaussian. Thus, smoothing with Cauchy results in a larger spread in value and likely better exploration. We observe the same in our simulation results.
Adaptive Smoothed Functional based Algorithms for Labor Cost Optimization in Service Systems

The overall algorithm flow can be diagrammatically represented as in Fig. 3. Each iteration of the algorithm involves two simulations (each for a period $D$) - one with the current best estimate of the parameter, $\theta(n)$ and the other with the perturbed parameter, $\theta(n) + \beta \eta(n)$. In every stage of the algorithm, the two simulations are carried out as represented in Fig. 2. Using the state values of the two simulations, $X(n)$ and $\hat{X}(n)$, the update rule is carried out specific to SASOC-SF-N or SASOC-SF-C.

![Diagram of Algorithm Flow]

Figure 3: Overall flow of the algorithm 1.

The complete algorithm structure in both cases can be expressed as below.

**Algorithm 1**  The Complete Algorithm Structure

**Input:**

- $R$, a large positive integer representing the number of iterations;
- $\theta(0)$, initial parameter vector;
- $\beta > 0$ is a fixed smoothing control parameter;
- $K \geq 1$ is fixed integer used to control the duration of the average cost accumulation (c.f. (10));
- $\{\eta(n), n \geq 1\}$, N-dimensional i.i.d. Gaussian/Cauchy random variables for SASOC-SF-N/SASOC-SF-C respectively.
- $\text{UpdateRule}()$, the stochastic update rule of the particular algorithm.
- $D$, the inter-update simulation duration described in Fig. 2.
- $\text{Simulate}(\theta, D) \rightarrow X$, the simulator of the SS. $X$ represents the state of the underlying constrained Markov process at the end of the simulation.
Output: $\theta^*$, the parameter vector after $R$ iterations.

\[
\theta \leftarrow \theta(0), \quad \lambda \leftarrow 0, \quad n \leftarrow 1
\]

\textbf{loop}

\[ X \leftarrow \text{Simulate}(\theta, D). \]
\[ \hat{X} \leftarrow \text{Simulate}(\theta + \beta \eta(n), D). \]
\[ (\theta, \lambda) \leftarrow \text{UpdateRule}(X, \hat{X}, \theta, \lambda; K). \]

\textbf{if} $n = R$ \textbf{then}

\[ \text{Terminate with } \theta. \]

\textbf{end if}

\[ n \leftarrow n + 1. \]

\textbf{end loop}

5.3 Gaussian Smoothed-Functional Algorithm

We describe SASOC-SF-N, a three time-scale stochastic approximation algorithm that does primal descent using a Gaussian/normal smoothed functional gradient estimate while performing dual ascent on the Lagrange multipliers. We first enumerate the conditions for a function to be a smoothing function, followed by an outline of the Gaussian SF gradient estimation technique in section 5.3.2. In section 5.3.3, we provide the update rule of SASOC-SF-N algorithm.

5.3.1 Conditions for a smoothing function

[23, pp. 471] enumerates a set of conditions for a function, $h_\beta(\theta)$, to be a smoothing function. They are (i) $h_\beta(\theta) = \frac{1}{|A| \times |B|} h_1(\theta)$, is a piece-wise differentiable function with respect to $\theta$, (ii) $\lim_{\beta \to 0} h_\beta(\theta) = \delta(\theta)$, where $\delta(\cdot)$ is the Dirac-Delta function, (iii) $\lim_{\beta \to 0} \int_\alpha h_\beta(\theta - \alpha) g(\alpha) = g(\theta)$, and (iv) $h_\beta(\cdot)$ is a probability density function.

5.3.2 Gaussian SF Gradient Estimate

We provide here a brief idea about the key concept of gradient estimation using smoothed functional. Let $\alpha$ be a $|A| \times |B|$-dimensional vector of $N(0, \beta^2)$ random variables with $\beta > 0$. Let $G_\beta(\cdot)$ denote the probability density function (p.d.f.) of $\alpha$. The Gaussian smoothed functional estimate, obtained as a convolution of the gradient of the Lagrangian (w.r.t. $\theta$) with the Gaussian density function $G_\beta(\cdot)$ is
given by,
\[ D_{\beta,1}(\theta) = \int_{\alpha} G_\beta(\theta - \alpha) \nabla_\alpha L(\alpha, \lambda) d\alpha. \] (8)

Upon integration by parts and simplification, we get,
\[ D_{\beta,1}(\theta) = \int_{\alpha} \nabla_\alpha G_\beta(\theta - \alpha) L(\alpha, \lambda) d\alpha. \]

Observing that \( \nabla_\alpha G_\beta(\alpha) = -\frac{\alpha}{\beta^2}G_\beta(\alpha) \) and using \( \eta = \frac{\alpha}{\beta} \) in the above, we get,
\[ D_{\beta,1}(\theta) = \frac{1}{\beta} \int_{\eta} -\eta G_1(\eta)L(\theta - \beta \eta, \lambda) d\eta, \]
where \( G_1(\eta) \) is standard normal \((N(0,1))\) density function, i.e., \( \beta = 1 \). Since Gaussian density \( G_\beta(\cdot) \) satisfies conditions for a smoothing function given in section 5.3.1, we have
\[ \nabla_\theta L(\theta, \lambda) = \lim_{\beta \downarrow 0} E \left[ \left. \frac{\eta}{\beta} (L(\theta + \beta \eta, \lambda)) \right| \theta, \lambda \right], \]
where the expectation is with respect to the p.d.f., \( G_1(\cdot) \). Since single simulation based estimates are known to have higher variability (see [6, Fig. 3]), we use a two simulation estimate, similar to that in [27, CG-SF], as given below:
\[ \nabla_\theta L(\theta, \lambda) = \lim_{\beta \downarrow 0} E \left[ \left. \frac{\eta}{\beta} (L(\theta + \beta \eta, \lambda) - L(\theta, \lambda)) \right| \theta, \lambda \right]. \] (9)
Here \( \eta \) is a \(|A| \times |B|\)-vector of independent \( N(0,1) \) random variables.

### 5.3.3 SASOC-SF-N Algorithm

We estimate the quantities \( L(\theta + \beta \eta, \lambda) \) and \( L(\theta, \lambda) \) that are in turn required for estimating the gradient \( \nabla_\theta L(\theta, \lambda) \) as in (9) above, by using two iterations running with parameters \( \theta + \beta \eta \) and \( \theta \) on a faster time-scale. For \( \lambda_{i,j} \) and \( \lambda_f \), the values of \( g_{i,j}(\cdot) \) and \( h(\cdot) \) respectively provide a stochastic ascent direction, proof of which will be given later in Theorem 4. 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 for this scheme, SASOC-SF-N, is as follows:
For all \( n \geq 0 \),

\[
W_i(n + 1) = \Pi \left[ W_i(n) + b(n) \left( \frac{\eta(nK)}{\beta} (\bar{L}(nK) - \bar{L}'(nK)) \right) \right],
\]

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

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

\[
\bar{L}(nK + m + 1) = \bar{L}(nK + m) + d(n)(l(X_{nK+m}, \lambda(nK)) - \bar{L}(nK + m)),
\]

\[
\bar{L}'(nK + m + 1) = \bar{L}'(nK + m) + d(n)(l(X_{nK+m}, \lambda(nK)) - \bar{L}'(nK + m)),
\]

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

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

In the above,

- \( l(X, \lambda) = c(X) + \sum_{i=1}^{\lfloor \frac{n}{K} \rfloor} \sum_{j=1}^{\lfloor \frac{|P|}{|C|} \rfloor} \lambda_{i,j}g_{i,j}(X) + \lambda_f h(X) \) is the single stage sample of the Lagrangian;

- \( X_m \) represents the state at iteration \( m \) from the simulation run with the nominal parameter \( \theta_{\lfloor \frac{n}{K} \rfloor} \) while \( \hat{X}_m \) represents the state at iteration \( m \) from the simulation run with the perturbed parameter \( \theta_{\lfloor \frac{n}{K} \rfloor} + \beta \eta_{\lfloor \frac{n}{K} \rfloor} \). Thus, two simulations are carried out for each iteration, one with \( \theta_{\lfloor \frac{n}{K} \rfloor} \), and the other with perturbed parameter \( \theta_{\lfloor \frac{n}{K} \rfloor} + \beta \eta_{\lfloor \frac{n}{K} \rfloor} \). Here \( \lfloor \frac{n}{K} \rfloor \) denotes the integer portion of \( \frac{n}{K} \). For simplicity, hereafter, we use \( \theta \) to denote \( \theta_{\lfloor \frac{n}{K} \rfloor} \) and \( \theta + \beta \eta \) to denote \( \theta_{\lfloor \frac{n}{K} \rfloor} + \beta \eta_{\lfloor \frac{n}{K} \rfloor} \);

- \( \bar{L} \) and \( \bar{L}' \) are initialized with zero. \( X \) and \( \hat{X} \), which correspond to the two simulations with \( \theta \) and \( \theta + \beta \eta \), are used to update \( \bar{L} \) and \( \bar{L}' \), respectively;

- \( K \geq 1 \) is a fixed parameter which controls the frequency 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. It is generally observed that a value of \( K > 1 \) (say in the range of 10 to 500) shows good numerical performance. The convergence analysis holds for any \( K \geq 1 \);
Adaptive Smoothed Functional based Algorithms for Labor Cost Optimization in Service Systems

- $\Pi(\cdot)$ is a projection operator that ensures that the updated value for $\theta$ stays within the compact set $S$ and is defined by
  \[ \Pi(\theta) \triangleq (\pi(W_{1,1}), \ldots, \pi(W_{|A||B|}))^T, \theta \in \mathcal{R}^{|A|\times|B|}. \]
  Here $\pi(x) \triangleq \min(\max(0, x), W_{\max})$.
  Hence, the projection operator $\Pi$ keeps each $W_{i,j}$ bounded between 0 and $W_{\max}$ and this is necessary for ensuring the convergence of $\theta$;

- $\beta > 0$ is a fixed smoothing control parameter; and

- $\eta = (\eta_1, \eta_2, \ldots, \eta_{|A|\times|B|})^T$ is a vector of $|A| \times |B|$ independent $N(0, 1)$ random variables.

To summarize, the update rule (10) essentially performs a gradient descent in the worker parameter $W_{i}(\cdot)$ and couples it with dual ascent for Lagrange multipliers $\lambda_{i,j}(\cdot)$. This is evident from the following:

1. Updates of $\bar{L}$ and $\bar{L}'$ accumulate the long-run average cost for the relaxed problem (i.e., the Lagrangian) with parameters $\theta$ and $\theta + \beta\eta$ respectively.

2. The update to the worker parameter $W_{i}(\cdot)$ in (10) in essence uses the Lagrange estimates $\bar{L}$ and $\bar{L}'$ to tune $W_{i}$ in the negative gradient descent direction, i.e., $-\nabla_{\theta}L(\theta, \lambda)$. Note that the estimate used for the gradient in (10) is motivated by (9), with the primary difference being that the SASOC-SF-N algorithm uses a fixed $\beta > 0$.

3. The Lagrange multipliers $\lambda_{i,j}(\cdot)$ are tuned in the ascent direction.

4. It will be shown later in Appendix A that the SASOC-SF-N algorithm (10) converges to a saddle point $(\theta^*, \lambda^*)$ of the Lagrangian.

Assumption (A3)

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

\[
\sum_n a(n) = \sum_n b(n) = \sum_n d(n) = \infty;
\]

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

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

The first two requirements in (A3) are standard in stochastic approximation algorithms. In particular, the first requirement ensures that the recursions do not
converge prematurely while the second requirement aids in cancelling the effect of stochastic noise as well as errors in gradient estimation. As described previously, the third requirement in (A3) essentially gives rise to the desired separation of time-scales between recursions in our algorithms. Step-sizes chosen according to (A3) ensure that the recursions of Lagrange multipliers $\lambda_{i,j}$ and $\lambda_f$, 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. As we do in our experiments, for all algorithms, one can select for instance the following step-sizes $a(n), b(n)$ and $d(n), n \geq 0$ that can be seen to satisfy the requirements in (A3).

$$a(0) = \hat{a}, \quad b(0) = \hat{b}, \quad d(0) = \hat{d},$$

$$a(n) = \frac{\hat{a}}{n}, \quad b(n) = \frac{\hat{b}}{n^{\beta}}, \quad d(n) = \frac{\hat{d}}{n^{\alpha}}, \quad n \geq 1,$$

with $1/2 < \alpha < \beta < 1, 0 < \hat{a}, \hat{b}, \hat{d} < \infty$.

### 5.4 Cauchy Smoothed-Functional Algorithm

We now describe SASOC-SF-C, which uses the Cauchy density instead of Gaussian density for the smoothed functional estimate. We first briefly describe the Cauchy SF gradient estimation technique in section 5.4.1 and then in section 5.4.2 we provide the update rule of the SASOC-SF-C algorithm.

#### 5.4.1 Cauchy SF Gradient Estimate

We first explain the key concept of gradient smoothing with the Cauchy density. Let $N = |A| \times |B|$ and $\Lambda = [-W_{\text{max}}, W_{\text{max}}]^N$. For some scalar constant $\beta > 0$, let

$$\hat{D}_{\beta,1}L(\theta, \lambda) = \int_{\eta} C_{\beta}(\theta - \eta) \nabla_{\theta} L(\theta, \lambda) d\eta$$

represent the convolution of the gradient (w.r.t. $\theta$) of the Lagrangian with an $N$-dimensional multi-variate truncated Cauchy p.d.f.,

$$C_{\beta}(\theta - \eta) = \begin{cases} \frac{\Gamma\left(\frac{N+1}{2}\right)}{\pi^{\frac{N+1}{2}} \beta^{N} \Omega^{\frac{N+1}{2}}} \left(1 + \frac{(\theta - \eta)^T(\theta - \eta)}{\beta^2}\right)^{-\frac{N+1}{2}} & \text{for } \eta \in \Lambda, \\ 0 & \text{otherwise,} \end{cases}$$

with $\Omega = \sum_{i,j} \lambda_{i,j} \lambda_{f}$. 

---

*H.L. Prasad et al.*

25
where $\theta, \eta \in \mathbb{R}^N$, $\Gamma(\cdot)$ is the Gamma function and $\Omega$ is a scalar factor [28] that results from the truncation of the original Cauchy density. Truncation is necessary to ensure that the integral in (11) is well defined. Now observing that

$$\nabla_{\eta} C_{\beta}(\eta) = -\frac{\eta(N + 1)}{(\beta^2 + \eta^T \eta)} C_{\beta}(\eta) \text{ for } \eta \in \Lambda^0,$$

where $\Lambda^0$ is the interior of the set $\Lambda$, and following the procedure similar to that in section 5.3.2, we obtain

$$\hat{D}_{\beta,1} L(\theta, \lambda) = E \left[ \frac{\eta(N + 1)}{\beta(1 + \eta^T \eta)} L(\theta + \beta \eta, \lambda) \right],$$

where the expectation is over truncated standard multi-variate Cauchy density with p.d.f.,

$$C(\eta) = \begin{cases} 
\frac{\Gamma\left(\frac{N+1}{2}\right)}{\pi^{\frac{N}{2}} \Omega} \frac{1}{(1 + \eta^T \eta)^{\frac{N+1}{2}}} & \text{for } \eta \in \Lambda, \\
0 & \text{otherwise.}
\end{cases}$$

It is easy to see that the four conditions for a smoothing function, given in section 5.3.1, are satisfied by truncated Cauchy density function $C_{\beta}(\cdot)$. Thus,

$$\nabla_{\theta} L(\theta, \lambda) = \lim_{\beta \to 0} E \left[ \frac{\eta(N + 1)}{\beta(1 + \eta^T \eta)} L(\theta + \beta \eta, \lambda) \right] \bigg| \theta, \lambda \right). \quad (12)$$

The estimate of $\nabla_{\theta} L(\theta, \lambda)$ as in (12) requires only one simulation that however exhibits more variability than two simulation estimates (see [6, Fig. 3]). In order to reduce the variability in our estimates of the gradient, we extend the estimate (12) to use two simulations as follows:

$$\nabla_{\theta} L(\theta, \lambda) = \lim_{\beta \to 0} E \left[ \frac{\eta(N + 1)}{\beta(1 + \eta^T \eta)} (L(\theta + \beta \eta, \lambda) - L(\theta, \lambda)) \right] \bigg| \theta, \lambda \right). \quad (13)$$

This form of the gradient estimate is motivated from [29, 17], where two-sided estimates have been seen to perform better than their one-simulation counterpart in the case of Gaussian perturbations. We shall show later in appendix A that the above gradient estimate is indeed valid.
5.4.2 SASOC-SF-C Algorithm

Using (13) as the gradient estimate for Lagrangian, the overall update rule for SASOC-SF-C is as follows:

\[
W_i(n + 1) = \Pi \left[ W_i(n) + b(n) \left( \frac{\eta_i(nK)(N + 1)}{\beta(1 + \eta_i(nK)^T \eta(nK))} (L(nK) - \bar{L}'(nK)) \right) \right],
\]

\[\forall i = 1, 2, \ldots, N, \]

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

\[
\bar{L}(nK + m + 1) = \bar{L}(nK + m) + d(n) (l(X_{nK+m}, \lambda(nK)) - \bar{L}(nK + m)),
\]

\[
\bar{L}'(nK + m + 1) = \bar{L}'(nK + m) + d(n) (l(\hat{X}_{nK+m}, \lambda(nK)) - \bar{L}'(nK + m)),
\]

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

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

In the above, \( \eta \) is an \( N \)-dimensional multi-variate Cauchy random vector truncated to \( \Lambda \) while the rest of the terms have the same interpretation as in the SASOC-SF-N algorithm. The complete SASOC-SF-C algorithm follows as in Algorithm 1 with the UpdateRule() as in (14) of SASOC-SF-C and \( \eta \) being a truncated \( N \)-dimensional multi-variate Cauchy random vector.

6 Simulation Results

We now provide numerical results to illustrate the performance of the various SASOC algorithms. For the purpose of service system simulation, we used the simulation framework developed in [7], which in turn is based on AnyLogic simulation toolkit. We implemented the following staff optimization algorithms:

- **SASOC-SF-N**: This is the smoothed functional algorithm with Gaussian perturbations described in Section 5.3.
Adaptive Smoothed Functional based Algorithms for Labor Cost Optimization in Service Systems

- **SASOC-SF-C**: This is the smoothed functional algorithm with Cauchy perturbations described in Section 5.4.

- **SASOC-SPSA**: This is the SPSA based algorithm proposed in [1] and described in Section 6.1.

- **OptQuest**: This is a staff optimization algorithm that uses the state-of-the-art optimization tool-kit 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 is quite well-known as a tool for solving simulation optimization problems [8]. OptQuest along with several other engines from Frontline Systems won the INFORMS impact award\(^2\) in the year 2010.

The algorithms were compared using the performance metrics of \(W_{sum}\) and mean utilization. Here \(W_{sum} \triangleq \sum_{i=1}^{|A|} \sum_{j=1}^{|B|} W_{i,j}\) is the total number 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.

We now recall the SASOC-SPSA algorithm proposed in [1].

### 6.1 SASOC-SPSA

As with the SF based schemes we proposed, SASOC-SPSA [1] is a multi-timescale stochastic approximation algorithm that performs gradient descent in the primal and couples it with dual ascent for the Lagrange multipliers. However, for primal descent, SASOC-SPSA uses an SPSA-based gradient estimation technique [1].

\(^2\)http://www.solver.com/press201008.htm
Using the notation as before, the update rule of SASOC-SPSA is as follows:

\[
W_i(n+1) = \Pi \left( W_i(n) + b(n) \left( \frac{L(nK) - L'(nK)}{\beta \eta_i(n)} \right) \right),
\]
\[\forall i = 1, 2, \ldots, N,
\]

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

\[
\bar{L}(nK + m + 1) = \bar{L}(nK + m) + d(n)(l(X_{nK+m}, \lambda(nK)) - \bar{L}(nK + m)),
\]
\[
\bar{L}'(nK + m + 1) = \bar{L}'(nK + m) + d(n)(l(\hat{X}_{nK+m}, \lambda(nK)) - \bar{L}'(nK + m)),
\]
\[
\lambda_{i,j}(n+1) = (\lambda_{i,j}(n) + a(n)g_{i,j}(X_n))^+,
\]
\[\forall i = 1, 2, \ldots, |C|, j = 1, 2, \ldots, |P|,
\]
\[
\lambda_f(n+1) = (\lambda_f(n) + a(n)h(X_n))^+.
\]

(15)

In the above, for all update instances \( n \), \( \eta(n) = (\eta_1(n), \eta_2(n), \ldots, \eta_N(n))^T \) is a vector of \( N \) independent Bernoulli random variables, \( \eta_i(n), i = 1, 2, \ldots, N \). The rest of the terms have the same interpretation as in the SASOC-SF-N and SASOC-SF-C algorithms. In fact the last four recursions in (15) are identical to corresponding recursions in SASOC-SF-N and SASOC-SF-C. Again, the complete algorithm follows as in Algorithm 1 except that the UpdateRule() is of SASOC-SPSA with \( \{ \eta(n) : n \geq 1 \} \) as described above. We implemented this algorithm and compared the performance of our SF based schemes (SASOC-SF-N and SASOC-SF-C) with the same.

### 6.2 Implementation

To evaluate our algorithms, we leverage real-life service system data from five SS that provide server support to IBM’s customers. To allow a fair comparison, we choose SS having a variety of characteristics along the dimensions of geographical location, workload, number of customers supported, number of SWs, and SLA constraints. Fig 4 shows the total work hours per SW per day for each of the SS. The bottom part of the bars denotes C SR work, i.e., the SRs raised by the customers whereas the top part of the bars denotes I 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 C SRs. I SRs do
Figure 4: Total work volume statistics for each SS

(a) SS1 and SS2 work arrival pattern

(b) SS3, SS4 and SS5 work arrival pattern

Figure 5: Work arrival patterns over a week for each SS
not have deadlines but they may contribute to queue growth. While average work volumes is a useful metric, it does not directly correlate with performance against SLA constraints. As shown in Fig. 5, the arrival rates for SS1 and SS2 show much higher peaks than SS3, SS4, and SS5, although their average work volumes are comparable. Such fluctuations are significant because during the peak periods, many SRs may miss their SLA deadlines and influence the optimal staffing result.

All of our algorithms leverage the simulation framework developed in [7] for the evaluation step. While a number of dispatching policies were developed in [7], we focus on the PRIO-PULL and EDF policies here. In the PRIO-PULL 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, 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. In the experiments carried out by Banerjee et al [7], EDF performed the best and PRIO-PULL performed the worst among all dispatching policies. Hence, choosing these policies exposes SASOC algorithms to two very different operational models of SS.

We implemented our SASOC algorithms by invoking the simulation framework from [7] to compute $X$ and $\hat{X}$, corresponding to perturbed and unperturbed simulations respectively, as shown in Algorithm 1. As mentioned before, for the sake of comparison, we also implemented SASOC-SPSA [1] and a staff allocation algorithm using OptQuest.

For all the SASOC algorithms, the simulations were conducted for 1000 iterations, with each iteration having 20 replications of the SS - ten each with unperturbed parameter $\theta$ and perturbed parameter $\theta + \beta \eta$, respectively. In other words, we set 10 months for the parameter $D$. For the OptQuest algorithm, simulations were conducted for 5000 iterations, with each iteration involving 100 replications of the SS. For the SASOC algorithm, we set the weights in the single-stage cost function $c(X_m)$, see (1), as $r = s = 0.5$. We thus give equal weight-age to both the worker utilization and the SLA over-achievement components. The values of $\beta$ and $K$ were set to 0.5 and 10 respectively in all our experiments. The feasibility Boolean variable $q$ used in the constraint (3) is set to false (i.e., infeasible) if the queues are found to grow by 1000% or more over a two-week period. Each of the experiments are run on a machine with dual core Intel 2.1 GHz processor and 3 GB of RAM.
6.3 Results

Fig. 6(a) presents the $W^*_\text{sum}$ achieved for OptQuest and SASOC algorithms on five real life SS, with PRIO-PULL as the underlying dispatching policy. Here $W^*_\text{sum}$ denotes the value obtained upon convergence of $W_{\text{sum}}$. Fig. 7(a) presents similar results for the case of the EDF dispatching policy.

We observe that our SASOC algorithms find a significantly better value of $W^*_\text{sum}$ as compared to OptQuest on two SS pools, namely SS3 and SS4, while on SS5, SASOC algorithms perform on par with OptQuest. Note in particular that the performance difference between SASOC algorithms, and OptQuest on SS3 is nearly 100%. Further, on SS4, OptQuest repeatedly does not find a feasible solution in 5000 search iterations whereas the SASOC algorithms obtain a feasible good allocation. On the other two pools, SS1 and SS2, OptQuest is seen to be slightly better than SASOC.

Among SASOC algorithms, the Cauchy variant of the smoothed functional algorithm SASOC-SF-C exhibits the best overall performance, under both dispatching policies considered. SASOC-SF-C outperforms the Gaussian variant of the smoothed functional algorithm SASOC-SF-N on most of the SS pools. Further, in comparison to SASOC-SPSA, SASOC-SF-C finds a better value of $W^*_\text{sum}$ on SS3, SS4 and SS5, while showing comparable results on the other two pools. This behaviour of SASOC-SF-C could be attributed to the heavy tailed nature of Cauchy distribution which results in occasionally high perturbation values about the current best estimate of parameter and overall better search for an optimal point.

A significant advantage of our SASOC algorithms over OptQuest is the reduced execution time. OptQuest requires 5000 iterations with each iteration of 100 replications, whereas all our SASOC algorithms need 1000 iterations of 20 replications each to find $W^*_\text{sum}$. This implies an order of magnitude improvement in the number of replications needed, while searching for the optimal SS configuration in SASOC as compared to OptQuest. In fact, as we observed in some cases, OptQuest is unable to find a feasible solution even after 5000 iterations. Further, on comparing the simulation run-times, we observe that all SASOC algorithms result in atleast 10 to 12 times improvement as compared to OptQuest. For instance, on SS5 the typical run-time of OptQuest was found to be 29 hours, whereas SASOC algorithms took 3 hours to converge.

We also observe that the parameter $\theta$ (and hence $W_{\text{sum}}$) converges to the optimum value for each SASOC algorithm - SASOC-SF-C, SASOC-SF-N and SASOC-SPSA - on each of the SS pools considered. This is illustrated by the
convergence plots in Fig. 8(a) and Fig. 8(b), where the evolution of $W_{\text{sum}}$ is shown for all SASOC algorithms for both the dispatching policies considered. This is a significant feature of SASOC algorithms as we established convergence of our algorithm in section A and the plots confirm the same. In fact, convergence of all the SASOC algorithms is achieved within 500 to 700 iterations. In contrast, the OptQuest algorithm is not proven to converge to the optimum, which is observed in the case of SS4 in Fig. 6(a).

Fig. 6(b) and Fig. 7(b) present the mean utilization percentages achieved for SASOC algorithms and OptQuest for PRIO-PULL and EDF dispatching policies respectively, on five real life SS. Similar observations as for Fig. 6(a) and Fig. 7(a) are valid here, with SASOC algorithms showing better overall mean utilization performance over OptQuest. It is clear that in order to obtain the optimal staffing levels, any algorithm has to utilize the workers better and our SASOC algorithms achieved a higher mean utilization by incorporating the utilization of workers into the single stage cost function (1).

7 Conclusions

We presented two smoothed-functional based SASOC algorithms for optimizing staff allocation in the context of SS. We formulated the problem as a constrained optimization problem where both the objective and the constraint functions were long run averages of certain state dependent single-stage cost functions. A single stage cost that balanced the conflicting objectives of maximizing worker utilizations and minimizing the over-achievement of SLAs was employed. For solving the constrained optimization problem, we proposed two novel smoothed functional algorithms for estimating the gradient in the primal. Using the theory of multi-timescale stochastic approximation, we presented a convergence proof of our algorithms. Numerical experiments were performed to evaluate each of the algorithms against prior work in the context of real-life service systems. Our SASOC algorithms in general showed much superior performance when compared with the state-of-the-art simulation optimization tool-kit OptQuest, as they (a) exhibited 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 five thousand search iterations. By comparing the results of the SASOC algorithms on two independent dispatching policies, we showed that the performance of SASOC is agnostic of the operational
Adaptive Smoothed Functional based Algorithms for Labor Cost Optimization in Service Systems

References


Adaptive Smoothed Functional based Algorithms for Labor Cost Optimization in Service Systems


A Convergence Analysis

We collectively refer to the three algorithms as SASOC. Here we provide a sketch of the convergence of SASOC. The fastest time-scale in SASOC is \{d(n)\} which is used to update the Lagrangian estimates \(\bar{L}\) and \(\bar{L}'\) corresponding to simulations with \(\theta\) and \(\theta + \beta \eta\) respectively. First, we show that these estimates indeed converge to the Lagrangian values \(L(\theta, \lambda)\) and \(L(\theta + \beta \eta, \lambda)\) defined in equation (6). Note that the quantities \(\theta\) and \(\lambda\) which are updated along slower time-scales, can be assumed to be quasi-static for the purpose of analysis of the Lagrangian estimates. Second, we show that the parameter updates \(\theta(n)\) in SASOC, converge to a limit point of the ODE,

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

provided the sensitivity parameter \(\beta\) tends to zero in the algorithm. In (16), for any bounded continuous function \(\epsilon(\cdot)\),

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

Note that if \(\theta(t) \in S^o\) (the interior of \(S\)), then \(\Pi(\epsilon(\theta(t))) = \epsilon(\theta(t))\), since \(\Pi(\theta(t) + \delta \epsilon(\theta(t))) = \theta(t) + \delta \epsilon(\theta(t))\) for \(\delta > 0\) sufficiently small. Also, if \(\theta(t) \in \partial S\) (boundary of set \(S\)), is such that \(\theta(t) + \delta \epsilon(\theta(t)) \notin S\), then \(\Pi(\epsilon(\theta(t)))\) is the projection of \(\epsilon(\theta(t))\) to \(S\). The limit in (17) is well defined because \(S\) is a compact and convex set. For analysis of the \(\theta\)-recursion, the value of \(\lambda\) which is updated on the slowest time-scale is assumed constant. Further, we show that \(\lambda_{i,j}\)'s and \(\lambda_f\) converge respectively to the limit points of the ODEs,

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

where \(\theta^*\) is the converged parameter value of SASOC and for any bounded continuous function \(\bar{\epsilon}(\cdot)\),

\[
\Gamma(\bar{\epsilon}(\lambda(t))) = \lim_{\delta \downarrow 0} \frac{(\lambda(t) + \delta \bar{\epsilon}(\lambda(t)))^+ - \lambda(t)}{\delta}.
\]
The operator $\Gamma$ is similar to $\overline{\Pi}$. From the definition of the Lagrangian given in equation (6), 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, $\lambda_{i,j}$s and $\lambda_f$ are ascending in the Lagrangian value and converge to a local maximum point. Lastly, we argue that the point to which SASOC converges to, is a saddle point with it being a local minimum in $\theta$ and local maximum in the $\lambda_{i,j}$ and $\lambda_f$. We establish these convergences for the three SASOC algorithms via a sequence of Lemmas given below.

### A.1 SASOC-SF-N

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

**Proof 1** We let $\theta(n) \equiv \theta$ and $\lambda(n) \equiv \lambda, \forall n$, i.e., constants as these quantities are updated on the slower time-scale. Rewrite the $\bar{L}$ update as

$$\bar{L}(m + 1) = \bar{L}(m) + d(m) \left( L(\theta, \lambda) + \xi(m) + M_{m+1} - \bar{L}(m) \right),$$

where $\xi(m) = E[l(X_m, \lambda)|\mathcal{F}_{m-1}] - L(\theta, \lambda)$, and $M_{m+1} = l(X_m, \lambda) - E[l(X_m, \lambda)|\mathcal{F}_{m-1}], m \geq 1$, respectively. Also, $\mathcal{F}_m = \sigma\left(X_n, \hat{X}_n, \lambda(n), \theta(n), n \leq m\right), m \geq 0$, are the associated $\sigma$-fields. Then, $(M_m, \mathcal{F}_m)$ forms a martingale difference sequence. Now since,

$$l(X_m, \lambda) = c(X_m) + \sum_{i=1}^{\mid C \mid} \sum_{j=1}^{\mid P \mid} \lambda_{i,j}g_{i,j}(X_m) + \lambda_f h(X_m),$$

and $c(\cdot), g_{i,j}(\cdot), h(\cdot)$, are Lipschitz continuous functions, it is easy to see that $l(\cdot, \lambda)$ is Lipschitz continuous in $\cdot \cdot \cdot$. Thus,

$$|l(X, \lambda) - l(0, \lambda)| \leq |l(X, \lambda) - l(0, \lambda)| \leq \bar{K} \|X\|,$$

where $\bar{K} > 0$, is the Lipschitz constant of the function $l$. Thus,

$$|l(X, \lambda)| \leq \bar{K}(1 + \|X\|),$$

for $\hat{K} = \max(|l(0, \lambda)|, \bar{K})$. Now, observe that the state space of the process $\{X_n\}$ by definition is compact, i.e., closed and bounded. Thus,

$$|l(x, \lambda)| \leq \bar{C}$$
for some $C > 0$, $\forall x$ and $\lambda$. It is now easy to see that $(N_m, F_m), m \geq 0$, where $N_m, m \geq 0$, is defined as $N_m = \sum_{n=0}^{m} d(n) M_{n+1}$, is a square-integrable martingale. Further, 

$$\sum_m E[(N_{m+1} - N_m)^2 | F_m] = \sum_n d^2(n) E[M_{n+1}^2 | F_n] < \infty$$

almost surely, since $\sum_n d^2(n) < \infty$, and moreover $E[M_{n+1}^2 | F_n], n \geq 0$, is uniformly bounded almost surely by the foregoing. Now, note that since the process $\{X_n\}$ is ergodic Markov for any given $\theta$, we have that $\xi(m) \to 0$, almost surely as $m \to \infty$ on the ‘natural time-scale’ which is faster than the time-scale of the algorithm (see [26, Chapter 6.2] for a detailed discussion of natural time-scale convergence). The rest follows from the Hirsch lemma [30, Thm. 1, pp. 339].

On similar lines, $\|\bar{L}'(n) - L(\theta(n) + \beta \eta(n), \lambda(n))\| \to 0$ w.p. 1, as $n \to \infty$. Note that the update of parameter $\theta$ is on a slower time-scale than that of updates of $L$ and $L'$. Thus, from the point-of-view of updates to parameter $\theta$, $L$ and $L'$ would appear to have converged to $L(\theta(n), \lambda(n))$ and $L(\theta(n) + \beta \eta(n), \lambda(n))$. Since the updates of $\lambda$ are occurring on the slowest time-scale, we let $\lambda(n) \equiv \lambda, \forall n$. Define

$$D_{\beta, 2} L(\theta(n), \lambda) =
E \left[ \frac{\eta(n)}{\beta} (L(\theta(n) + \beta \eta(n), \lambda) - L(\theta(n), \lambda)) \bigg| \theta(n), \lambda \right].$$

**Lemma 2** As $\beta \to 0$, 

$$\|D_{\beta, 2} L(\theta(n), \lambda) - \nabla_\theta L(\theta(n), \lambda)\| \to 0 \text{ a.s.}$$

**Proof 2** Follows from [20, Proposition 4.2].

Let $K^\lambda = \{ \theta \in S| \Pi (-\nabla L(\theta, \lambda)) = 0 \}$ denote the set of fixed-points of the ODE (16). Let $\bar{K}^\lambda \subset K^\lambda$, denote the set of stable fixed points of the ODE (16). Note that $K^\lambda$ may contain unstable attractors such a local maxima, saddle points etc. in addition to local minima. For a given $\epsilon > 0$, let 

$$(\bar{K}^\lambda)^\epsilon = \{ \theta \in S\|\theta - \theta_0\| < \epsilon, \theta_0 \in \bar{K}^\lambda \},$$

denote the set of all points that that are in the $\epsilon$-neighborhood of the set $\bar{K}^\lambda$. 
Theorem 3 Under (A1)-(A3), given \( \epsilon > 0, \exists \beta_0 > 0 \), s.t. \( \forall \beta \in (0, \beta_0], \theta(n) \rightarrow (K^\lambda)^\epsilon \), with probability one as \( n \rightarrow \infty \).

Proof 3 As before, one can let \( \lambda(n) \equiv \lambda, \forall n \), for the analysis of the update of parameter \( \theta \). Let \( \mathcal{G}_m = \sigma \left( X_n, \hat{X}_n, \lambda, \theta(n), n \leq m; \eta(n), n < m \right), m \geq 1 \), denote a sequence of associated \( \sigma \)-fields. Rewrite the parameter update as: For \( i = 1, 2, \ldots, |A| \times |B| \),

\[
W_i(n + 1) = \Pi \left( W_i(n) - b(n)\nabla_{\theta,i} L(\theta(n), \lambda) + b(n)\xi_{i,n+1}^i \right), \tag{18}
\]

where \( \xi_{i,n+1}^i = \left( \frac{\eta(n)}{\beta} (L(\theta(n) + \beta \eta(n), \lambda) - L(\theta(n), \lambda)) - D_{\beta,i}^2 L(\theta(n), \lambda) \right). \) Let \( \bar{M}_n^i = \sum_{m=0}^n b(m)\xi_{m+1}^i, i = 1, \ldots, |A| \times |B| \). It is easy to see that \( (\bar{M}_n^i, \mathcal{G}_n), n \geq 0 \), are martingale sequences. Now observe that \( L(\cdot, \lambda) \) is a continuous function (given \( \lambda \)) by Assumption (A2) and is uniformly bounded since \( \cdot \in S \) (a compact set). Now note that

\[
\sum_{n=0}^\infty E[|\bar{M}_{n+1}^i - \bar{M}_n^i|^2 |\mathcal{G}_n] = \sum_n b^2(n)E[|\xi_{n+1}^i|^2 |\mathcal{G}_n] < \infty
\]

almost surely, since \( L(\cdot, \lambda) \) is uniformly bounded, \( \sum_n b^2(n) < \infty \) and \( E[\eta_i^2(n)] = \frac{1}{\epsilon} < \infty \). Thus, by the martingale convergence theorem, \( (\bar{M}_n^i, \mathcal{G}_n), n \geq 0 \), is an almost surely convergent martingale sequence. The rest follows as a consequence of [31, Theorem 5.3.3, pp. 191-196].

Finally, we consider the update of the Lagrange multiplier \( \lambda(n) \) along the slowest time-scale. For any \( \lambda(n) \), the corresponding \( \theta(n) \) can be considered to be an element of \( (K^\lambda)^\epsilon \), i.e., would have converged. Let \( \tilde{\theta}(n) \in (K^\lambda)^\epsilon \), denote the \( \theta \)-parameter to which the \( \theta \)-update converges (given \( \lambda(n) \)). Let

\[
F_{\tilde{\theta}\lambda(n)} = \{ \lambda \geq 0 | \Gamma(H(\tilde{\theta}(\lambda(n))) = 0, \Gamma(G_{i,j}(\tilde{\theta}(\lambda(n))) = 0, \forall i = 1, \ldots, |C|, j = 1, \ldots, |P| \}.
\]

Theorem 4 \( (\theta(n), \lambda(n)) \rightarrow (K^\lambda)^\epsilon \times F^\theta \), with probability one as \( n \rightarrow \infty \).
Proof 4 Rewrite the update of $\lambda_{i,j}$ in (10) as follows: For all $i = 1, \ldots, |C|$, $j = 1, \ldots, |P|$, 

$$\lambda_{i,j}(n+1) = \left( \lambda_{i,j}(n) + a(n)G_{i,j}(\theta^{(n)}) + N_n + M_{n+1} \right)^{+},$$

where $N_n = \mathbb{E}[g_{i,j}(X_n)|\mathcal{F}_{n-1}]-G_{i,j}(\theta^{(n)})$, $M_{n+1} = (g_{i,j}(X_n)-\mathbb{E}[g_{i,j}(X_n)|\mathcal{F}_{n-1}])$, $n \geq 1$, respectively. The process of $N_n, n \geq 0$, constitutes the Markov noise. Further, $N_n \to 0$ as $n \to \infty$ along the ‘natural’ time-scale which is clearly faster than the time-scale of the algorithm. See [26, Chapter 6.2] for a detailed discussion on the convergence of natural time-scale recursions. Again since $S$ is a compact set, we can show using the martingale convergence theorem that $\sum_{m=0}^{n} a(m) M_{m+1}, n \geq 1$, is an almost surely convergence martingale sequence. It now follows from [31, Theorem 5.3.3, pp. 191-196] that $\|\lambda(n) \to F_{\theta^{(n)}}\| \to 0$ as $n \to \infty$, almost surely. Now, from Theorem 3, it follows that $\|\theta(n) - (\bar{K}^{(n)})^\epsilon\| \to 0$ almost surely as $n \to \infty$. Now, the map $\theta^{(\cdot)} : \mathbb{R}^+ \to S$ is Lipschitz continuous by the implicit function theorem [32, Theorem 1]. The claim now follows from [26, Chapter 6, Theorem 2].

A.2 SASOC-SF-C

Let

$$\hat{D}_{\beta,2}L(\theta(n), \lambda) = \mathbb{E} \left[ \frac{\eta(n)(N+1)}{\beta(1+\eta(n)^2\eta(n))} \left( L(\theta(n) + \beta \eta(n), \lambda) - L(\theta(n), \lambda) \right) | \theta(n), \lambda \right].$$

Lemma 5 As $\beta \to 0$, 

$$\|\hat{D}_{\beta,2}L(\theta(n), \lambda) - \nabla_\theta L(\theta(n), \lambda)\| \to 0 \text{ a.s.}$$

Proof 5 Follows in a similar manner as [20, Proposition 4.2].

Also, note that $\mathbb{E}[^{2}\eta(n)] < \infty$ as $\eta(n)$ has a truncated Cauchy density. The rest of the analysis follows as for SASOC-SF-N.
42 Adaptive Smoothed Functional based Algorithms for Labor Cost Optimization in Service Systems

Figure 6: Performance of OptQuest and SASOC for PRIO-PULL dispatching policy on five real SS (Note: OptQuest is infeasible over SS4)
Figure 7: Performance of OptQuest and SASOC for EDF dispatching policy on five real SS (Note: OptQuest is infeasible over SS4). The utilization values in (6(b)) and (7(b)) have been rounded to nearest integer.
Figure 8: Convergence of $W_{sum}$ - Illustration on SS1 and SS3 for two dispatching policies