Adaptive Smoothed Functional Algorithms for Optimal Staffing Levels in Service Systems

H.L. Prasad†, Prashanth L.A.†, Shalabh Bhatnagar†, Nirmit Desai♯,

October 26, 2012

Abstract

Service systems are people-centric. The service providers employ a large work force to service many clients, aiming to meet the SLAs and deliver a satisfactory client experience. A challenge is that the volumes of service requests change dynamically and the types of such requests are unique to each client. The task of adapting the staffing levels to the workloads in such systems while complying with aggregate SLA (Service-Level Agreement) constraints, is non-trivial. We formulate this problem as a constrained parametrized Markov process with a discrete parameter 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 the Cauchy smoothed-functional algorithm for primal descent. Further, all SASOC algorithms incorporate a generalized projection operator (L.A. et al. [2012],[Bhatnagar et al., 2012, Chapter 9]) that extends the system to a continuous setting with suitably defined transition probabilities. We validate these optimization schemes on five real-life service systems and compare their performance with a recent algorithm, SASOC-SPSA, from Prashanth et al. [2011], and a commercial optimization software – OptQuest. Our algorithms are observed to be 25 times faster than OptQuest and have proven convergence guarantees to the optimal staffing levels, whereas OptQuest fails to find feasible solutions in some cases even under reasonably high threshold on the number of search iterations. From the optimization experiments, 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, service systems, smoothed functional (SF) algorithm, Gaussian and Cauchy perturbations

1 Introduction

World economies have steadily transformed from being product-based to services-based. In service-based economies, the clients and service providers engage and co-create value to achieve outcomes that satisfy the client. Due to the uniqueness of their respective business contexts, there is a vast variety in the types of service requests that the providers must address. Specifically, the service requests vary in terms of the expected turn-around times, the content of the business need they represent, and hence the skill required to address them. As a result, automation of service delivery is difficult and it remains a

†Dept. of Computer Science and Automation, Indian Institute of Science, Bangalore
♯IBM Research, Bangalore

labour-intensive business. Naturally, for the service providers, optimization of their workforce cost is crucial for competitiveness.

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 client expectations [Alter, 2008, Spohrer et al., 2007, Ramaswamy and Banavar, 2008]. This paper chooses to focus on the SS in the domain of IT services. However, the approach and the result are applicable to all domains of services. In IT services, the clients heavily rely on the IT infrastructures supporting their businesses. The sheer size, complexity, and uniqueness of such IT infrastructures drive outsourcing of the IT services to specialized IT service providers. The service providers manage the infrastructures 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 clients that drive the operations of a delivery center constitute a SS. A delivery center typically constitutes many such SS which also interact to ensure consistent service outcomes.

Depending on the technology areas supported by SS, their operational models may differ significantly, e.g., managing storage infrastructures versus managing mainframes. However, a critical entity in these operational models is the policy used for assigning SRs to SWs, called the *dispatching policy*. There are two fundamental challenges.

- The minimum staffing level in each shift and skill combination of an SS that allows the service provider to maintain compliance to the aggregate SLA constraints needs to be determined. Also, such a determination is conditional upon the unique operational characteristics of an SS. For instance, an SLA constraint could specify that 95% of all urgent SRs in a month from a client must be resolved within 4 hours. Note that the aforementioned 4 hour deadline does not apply to all individual SRs, but only to 95% of those which arrive in a month from that particular client.

- Because the SS characteristics such as work arrival and service time patterns, technologies, and clients supported change frequently, the optimization of staffing needs to keep up with these changes. Also, there often are "soft" requirements that the clients and the providers may want factored into the staffing determinations. As a result, an approach for the staffing optimization needs to be responsive to what-if analyses, necessitating quick turn-around times of the optimization algorithms.

We formulate the above problem as a constrained parametrized Markov process, with the aim of finding an optimum (discrete) worker parameter that minimizes a certain long-run cost objective, while complying to a set of constraint functions, which are also long run averages. We present two stochastic optimization algorithms — *SASOC (Staff Allocation using Stochastic Optimization with Constraints)* — to address both of the aforementioned challenges. Both SASOC algorithms that we propose 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. Further, the core of each SASOC algorithm is a three-timescale stochastic approximation scheme, that perfoms gradient descent in the primal and an ascent in the dual 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 smoothen the objective

(i.e., the Lagrangian). While the service system framework (see Figure 1) is similar to that in Banerjee et al. [2011], the constrained Markov process formulation is entirely new here. Further, unlike the OptQuest algorithm [Laguna, 1998] studied there, our SASOC algorithms are provably convergent and computationally efficient as well.

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 is a novel stochastic optimization scheme that uses Cauchy distribution for smoothing and requires only two simulations. We shall refer to the latter algorithm as *SASOC-SF-C*. The smoothed functional approach was originally proposed by Katkovnik and Kulchitsky [1972] where convolution with the Gaussian density was explored. Styblinski and Tang [1990] proposed an algorithm where convolution with the Cauchy distribution was used; 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 dimension, unlike the Cauchy based SF algorithm proposed by Styblinski and Tang [1990]. The overall optimization procedure in each of our algorithms involves two main stages: search of a candidate solution followed by an evaluation of the same. Evaluation of a solution is carried out by simulating the SS operations. We use the simulation framework developed by Banerjee *et al.* Banerjee et al. [2011] under two different dispatching policies, i.e., PRIO-PULL (basic priority scheme) and EDF (earliest deadline first). For the sake of comparison, we also implement the SASOC-SPSA algorithm from [Prashanth et al., 2011]. 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 from a large IT services provider. 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 estimated based on this data and supplied as inputs to the simulation framework [Banerjee et al., 2011]. In comparison with the commercial optimization software OptQuest [Laguna, 1998], we find that both the SASOC algorithms that we propose (a) are 25 times faster than OptQuest, (b) find better solutions, resulting in *lower labour costs* than those found by OptQuest in majority of the SS, and (c) guarantee convergence even in cases where OptQuest does not find feasible solutions (at least until five thousand search iterations). We argue that due to guaranteed convergence and a much lower computational time requirement than OptQuest, SASOC algorithms are better suited to address the two challenges highlighted above. We study the performance of our SASOC algorithms on PRIO-PULL and EDF dispatching policies — signifying two unique operational models — and observe that the SASOC algorithms exhibit consistent performance benefits. Further, from the simulation experiments we find that SASOC-SF-C performs slightly better as compared to SASOC-SPSA and SASOC-SF-N.

2 Literature Survey

We survey relevant literature in the context of analysis of service systems and development of stochastic optimization algorithms. Further, we also outline the differences of our model and technique from the other approaches.

2.1 Service Systems

Verma et al. [2011] proposed a novel dispatching policy by solving a certain integer programming problem. The integer program formulated there did not involve aggregate SLA constraints. Further, unlike our work, they did not consider the problem of optimizing the staffing levels for a given dispatching policy. An algorithm for scheduling workers across shifts in the context of a third-level IT support system is proposed in Wasserkrug et al. [2008]. However, unlike us, the algorithm proposed by Wasserkrug et al. [2008] is not validated with data from real-life service systems. Chan [2008] considered a setting where a service system is seen to be composed of several M/M/1 queues and applied an agent based simulation procedure to understand the behavior of such a system. Robbins and Harrison [2008] proposed the usage of a simulation based search method for finding the optimal staffing levels in the context of a call-center domain. However, unlike us, an analytical model of the system is assumed and the model there does not include aggregate SLA constraints as well as some of the service systems dynamics such as preemption and swing policies. Simulation based methods for finding the optimal staffing in the context of a multiskill call center are proposed, for instance, in [Cezik and L'Ecuyer, 2008, Bhulai et al., 2008]. While Cezik and L'Ecuyer [2008] proposed a cutting plane algorithm for solving an integer program, Bhulai et al. [2008] relied on obtaining a linear programming solution. However, the problem formulations by Cezik and L'Ecuyer [2008], Bhulai et al. [2008] are for a single iteration of the system and they do not consider finding the optimal staffing level that minimizes a certain long term objective. Also, the solutions proposed by Cezik and L'Ecuyer [2008] as well as Bhulai et al. [2008] are heuristic (i.e., without proven convergence results) and further, they do not consider aggregate SLA and queue stability constraints. Banerjee et al. [2011] proposed a framework for service system simulation and we leverage the same to evaluate the various staff optimization schemes proposed here. In general, none of the above papers propose an adaptive scheme that optimizes staffing in the long run while adhering to certain aggregate SLA and queue stability constraints.

The closest to our work is the SPSA based scheme for staffing optimization in SS, proposed by us (see Prashanth et al. [2011]). We propose two new smoothed functional algorithms and prove their convergence to the optimal staffing levels. Further, we provide detailed experimental results in comparison with the SASOC-SPSA algorithm of Prashanth et al. [2011].

2.2 Stochastic Optimization

Gradient descent techniques are commonly used 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 model information and can work with both real or simulated data. An important subclass of these methods that are commonly referred to as perturbation analysis (PA) techniques, see for instance, Chong and Ramadge [1993], Chong and Ramadge [1994], Ho and Cao [1991], Fu [1990], are based on sample path gradients and largely use only one simulation. Likelihood ratio (LR) approaches (Andradóttir [1996], L'Ecuyer and Glynn [1994]) are also based on one-simulation estimates and require knowledge of pathwise gradients. Where applicable, both PA and LR approaches are known to perform well. However, many times, PA and LR approaches are not applicable as they require certain additional regularity conditions on the system model and performance functions making them applicable over a restricted class of systems.

Another popular subclass of simulation optimization methods use SF and SPSA schemes to estimate the gradient. Both of these schemes estimate the gradient of the objective by generating simulations with randomly perturbed parameters. However, these methods are more generally applicable as they do not require regularity conditions similar to PA and LR approaches.

The SF schemes estimate the gradient by convolving the gradient function with a probability density function that satisfies certain properties. Normal, Cauchy and uniform densities are known to be suitable candidates for the perturbation random variables in SF schemes. SF schemes were first proposed by Katkovnik and Kulchitsky [1972] and they required only one simulation. Styblinski and Tang [1990] proposed a two-simulation variant of SF and it was found to have lower variance when compared to the original one-simulation SF, see Katkovnik and Kulchitsky [1972]. Bhatnagar [2007] derived SF estimates of the Hessian using Gaussian distributed perturbations and proposed Newton based simulation optimization procedures using these estimates. While second-order Newton based algorithms are more accurate in comparison to the gradient based scheme, the accuracy comes at a significantly higher computational cost. This is because Newton based algorithms involve projection operators that keep the iterates within the set of positive definite matrices and moreover, inverting the Hessian is computationally expensive. Hence, 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 by Spall [1992] 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 requires only two samples of the objective function regardless of the parameter dimension N. Spall [1997] proposed a one-simulation variant of SPSA. However, unlike its two-simulation counterpart [Spall, 1992], the algorithm by Spall [1997] was not found to work as well in practice. Usage of Hadamard matrix based deterministic perturbations instead of randomized perturbations was proposed by Bhatnagar et al. [2003] with the resulting one-simulation algorithm exhibiting considerably superior performance over its one-simulation random perturbation counterpart.

Bhatnagar et al. [2011a] proposed four simulation-based algorithms for general constrained optimization problems. The Lagrange relaxation procedure has been applied by Bhatnagar et al. [2011a] to the constrained optimization problem. Two of the algorithms proposed by Bhatnagar et al. [2011a] 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 also been considered, for instance, in [Borkar, 2005, Bhatnagar, 2010]. The algorithm proposed by Borkar [2005] 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 by Bhatnagar [2010] combines the ideas of multi-timescale stochastic approximation, reinforcement learning and function approximation and obtains an optimal policy for a constrained approximate MDP. We refer the reader to Bhatnagar et al. [2012] for a textbook treatment of simultaneous perturbation approaches for stochastic optimization including engineering applications.

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

1. Smoothed functional ideas were originally discussed by Katkovnik and Kulchitsky [1972]. However, the original application by Katkovnik and Kulchitsky [1972], of smoothing techniques to gradient estimation was to find local minima in (i) non-differentiable functions, and (ii) rapidly fluctuating multi-extremal functions. Only later [Kreimer and Rubinstein, 1988, Styblinski and

Tang, 1990], were smoothing techniques used for simulation based gradient estimation. Our approach is also for simulation-based optimization.

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 by Styblinski and Tang [1990], their procedure requires several simulations to estimate the gradient. Styblinski and Tang [1990] provided only an experimental validation for the procedure proposed there whereas we provide a proof of convergence for our algorithm SASOC-SF-C.

3. Many existing algorithms in the literature [Spall, 2000, Bhatnagar, 2005, 2007] are for unconstrained optimization whereas our labour optimization algorithms work with SLA and queue feasibility constraints, that are in fact certain long-run average cost functions.

4. Our constrained optimization algorithms differ from those by Bhatnagar et al. [2011a] in the way Lagrange multipliers are estimated and also the accumulation procedure for computing gradients. Moreover, Cauchy-based perturbations for SF algorithms have not been considered in Bhatnagar et al. [2011a].

5. It will be described later in Section 3, that the problem being formulated is parametrized by worker parameter, that specifies the number of workers across shifts and skill levels. Since, the number of workers is strictly an integer, the stochastic optimization techniques that are devised here are of the discrete optimization type, unlike the schemes given by Bhatnagar et al. [2011a] which are for continuous parameter constrained optimization. Another paper by Bhatnagar et al. [2011b] incorporates a discrete parameter setup for an unconstrained optimization problem. They use a discrete projection operator which is fully randomized for smoothing purposes. We apply a generalized projection scheme, similar to the one described in L.A. et al. [2012], which is partly deterministic and partly randomized as discussed in Section 5.2.

3 The Setting

The setting is mostly borrowed from that given in Banerjee et al. [2011]. We provide here a summary of the setting relevant to the labour staffing problem which will be formulated in the next section (Section 4). Figure 1 shows the main components in the operational model of SS. SRs arrive from multiple clients 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 assigns 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 high-skilled workers to low complexity queues if such queues are growing. Finally, a *preemption* policy specifies the preemption action as well as the 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.

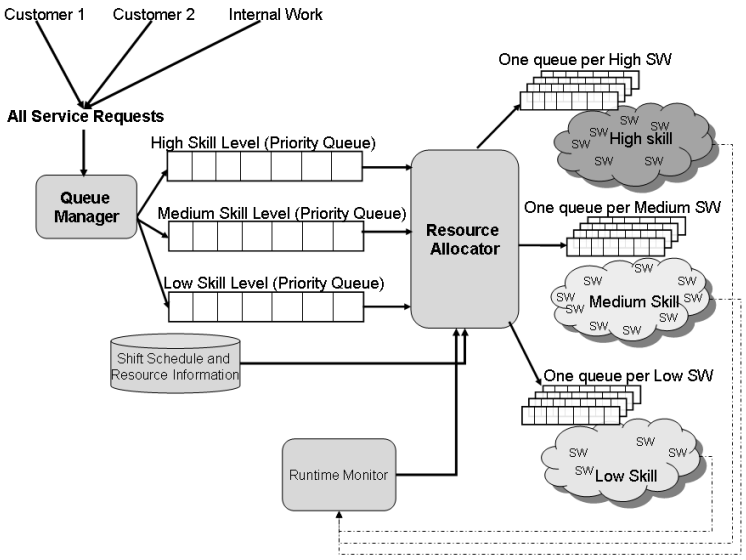


Figure 1: Components of the operational models of SS

For example, from midnight to 1 am on monday, the rate may be 1 SR per hour, from 1am to 2am on the same day, 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 parameters specified for each priority and complexity combination. Thus, the service times for all SRs having identical priority and complexity follow a unique log-normal 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, see Banerjee et al. [2011] for a detailed description.

A SW works in exactly one shift (working days and times) and a SS may operate in 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 do not grow 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 the work completed. For example, say in a given month, 100 SRs arrived at various times from a client to a SS and 20 of them were completed. If 15 of these SRs were completed within the target time, the SLA attainment would be $\frac{15}{20}$, i.e., 75%. The remaining 80 SRs would be in progress without a known completion time and do not impact the SLA attainment measures. Hence, a healthy SLA attainment of 75% alone does not provide a complete view of the health of the SS, as the SLA attainment only captures the completed tasks and not necessarily the pending tasks. In order to handle this, a bound on the growth of the queue of pending tasks would be required.

4 Problem Formulation

Our aim here is to optimize the worker parameter, i.e., the number of workers across shifts and skill levels while satisfying a set of SLA and queue stability constraints. We formulate this as a constrained parametrized Markov process with a discrete worker parameter. Both the objective and the constraints

are long-run averages of single-stage cost and constraint functions (see below). We employ the long-run average cost framework in order to understand the steady-state system behavior. Note that the optimization of the number of workers is carried out for a given dispatching policy.

We denote the worker parameter vector by θ, which is defined as follows:

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

In the above,

- A denotes the set of shifts of the workers and B the set of worker skill levels.
- W_i denotes the number of service workers with skill level $(i-1)\%|B|$ and shift index $(i-1)/|B|$.
- $\mathcal{D} = \{0, 1, \ldots, W_{max}\}^{|A|\times|B|}$ is the (finite) set of possible discrete values that the worker parameter can take. We shall henceforth refer to the members of the discrete set $\mathcal{D}$ as D^j, where $j = 1, 2, \ldots, q$, for some $q > 1$.

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 to the distribution of workers as in Table 1, $A = \{$S1, S2, S3$\}$ and $B = \{$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 $\theta = (W_1, \ldots, W_9)^T = (1, 3, 7, 0, 5, 2, 3, 1, 2)^T$.

Table 1: Workers W_i

	Skill levels		
Shift	High	Med	Low
S1	1	3	7
S2	0	5	2
S3	3	1	2

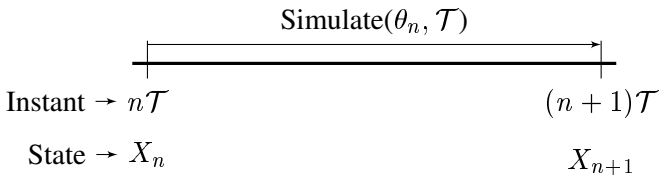


Figure 2: A portion of the time-line illustrating the process

The system evolves probabilistically over states as a constrained parametrized Markov process, with each system transition from one state to another illustrated in Figure 2. In essence, we continue the simulation of the service system for a fixed period $\mathcal{T}$ with the current worker parameter θ_n. $\mathcal{T}$ is chosen based on the period over which the contractual SLAs are evaluated by the clients. In our case, contractual SLAs are evaluated every month and $\mathcal{T}$ is set to 10 months. The simulation output causes a probabilistic transition from the current state X_n at instant $n\mathcal{T}$ to the next state X_{n+1} (at instant $(n+1)\mathcal{T}$), while incurring a single stage cost $c(X_n)$. The precise definitions of the state X_n and the cost function $c(X_n)$ are given as part of the constrained parametrized Markov process formulation in Section 4.1.

The SASOC algorithms that we design subsequently (see Section 5) use the cost $c(X_n)$ to tune the worker parameter θ and the system simulation would in this case continue with a new worker parameter θ_{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 θ and use the subsequent cost output $c(X_n)$ to tune θ. We now present a description of the underlying constrained parametrized Markov process below.

4.1 Constrained Parametrized Markov Process

The state X_n at instant n is given by

$$X_n = (\mathcal{Q}^T(n), u_{1,1}(n), \ldots, u_{|A|,|B|}(n), \gamma'_{1,1}(n), \ldots, \gamma'_{|C|,|P|}(n), q(n), \mathbb{R}(n))^T. \tag{1}$$

In the above,

- $\mathcal{Q}(n)$ is a vector of complexity queue lengths and is defined as $\mathcal{Q}(n) = (\mathcal{Q}_1(n), \ldots, \mathcal{Q}_{|B|}(n))^T$, with $\mathcal{Q}_i(n)$ denoting the complexity queue length corresponding to the skill level i, where $i = 1, \ldots, |B|$. For instance, $\mathcal{Q}_0(n)$ would give the length of the complexity queue housing 'low' complexity SRs and $\mathcal{Q}_1(n)$, $\mathcal{Q}_2(n)$ would give the corresponding numbers for 'medium' and 'high' complexity SRs. Note that since all the complexity queues are of finite size, we have $\mathcal{Q}_i(n) \leq \varsigma, i = 1, \ldots, |B|$, where $\varsigma > 0$ is a sufficiently large constant.
- C denotes the set of all clients and P, the set of all possible priorities in the SS under consideration. Note that any arriving SR has a client 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 client i and priority j, at instant n.
- $q(n)$, denotes the queue stability at instant n and takes values in $\{0, 1\}$. Note that $q(n)$ is 0 if the SR complexity queues growth rate crosses a threshold and is 1 otherwise. We require queue stability variable $q(n)$ to keep the SR complexity queues bounded. Note that SLA attainments are calculated only on completed SRs and not on those being queued up in the system.
- $\mathbb{R}(n)$ is a vector of residual service times and is defined as $\mathbb{R}(n) = (\mathbb{R}_{1,1,1}(n), \ldots, \mathbb{R}_{1,1,W_{\max}}(n), \ldots, \mathbb{R}_{|A|,|B|,W_{\max}}(n))$, where $\mathbb{R}_{i,j,k}(n)$ denotes the residual service time of the SR currently being processed at instant $n\mathcal{T}$ by the kth worker in shift i and skill level j. By convention, we set $\mathbb{R}_{i,j,k} = \kappa$ if there is no worker corresponding to the shift i, skill level j and index k with this shift-skill combination. Here κ is a special value used to signify the absence of a worker in a particular shift-skill level combination.

Remark 1 *In our setting, the service times follow a truncated log-normal distribution. The residual service time at any instant cannot be precisely estimated in a real system. Hence, $\mathbb{R}(n)$ is the unobserved or hidden state component for the Markov process. Strictly speaking, the setting falls under the realm of hidden Markov processes. However, the cost function $c(\cdot)$ and other constraint functions of X_n formulated later in this section do not depend on the value of $\mathbb{R}(n)$ and instead work with the observable components of the state. Further, the algorithms we propose aim to minimize a long-run average of the cost function, while satisfying the constraints (also long-run averages) and hence, do not require $\mathbb{R}(n)$ in their update rules.*

Remark 2 *We observe that each of the state components in* (1) *is closed and bounded. For instance,* $u(n)$, $\gamma'(n) \in [0, 1]$, *the complexity queue lengths* $\mathcal{Q}_i(n) \leq \varsigma$ *and* $\mathbb{R}(n)$ *is also closed and bounded as it follows a truncated log-normal distribution. Thus, the state space of the Markov process* $\{X_n, n \geq 0\}$ *is a compact set.*

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

	Skill levels		
Shift	High	Med	Low
S1	67%	34%	26%
S2	45%	55%	39%
S3	23%	77%	62%

Table 3: Sample SLA constraints

(a) SLA targets $\gamma_{i,j}$

	Customers	
Priority	Bossy Corp	Cool Inc
P_1	95%4h	89%5h
P_2	95%8h	98%12h
P_3	100%24h	95%48h
P_4	100%18h	95%144h

(b) SLA attainments $\gamma'_{i,j}$

	Customers	
Priority	Bossy Corp	Cool Inc
P_1	98%4h	95%5h
P_2	98%8h	99%12h
P_3	89%24h	90%48h
P_4	92%18h	95%144h

Tables 2 and 3(b) provide sample utilizations and SLA attainments on a SS with three shifts, two clients and four priority levels. Table 3(a) illustrates the target SLA requirements for the same SS.

By a constrained parametrized Markov process, we mean an $\mathbb{R}^d$-valued process $\{X_n\}$ whose evolution depends on a parameter $\theta \in \mathbb{R}^N$ where $N \stackrel{\triangle}{=} |A| \times |B|$. Also, given θ, the process $\{X_n\}$ is Markov. Let $p_\theta(x, dy)$, $x, y \in \mathcal{S} \subset \mathbb{R}^d$, denote the transition kernel of $\{X_n\}$ where θ is a parameter. Here, the set $\mathcal{S} \subset \mathbb{R}^d$, denotes the state space of the process, that is, the set of all possible values the Markov process could possibly take. When $X_n = x$, an immediate, i.e., single-stage, cost $c(x)$ is incurred. Further, 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 θ that minimizes a long-term cost function that in turn depends on the single-stage cost $c(\cdot)$ and that is subject to certain long-term constraints getting satisfied. 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 $c(X_n)$ in a way as to minimize (i) the under-utilization of workers across all shifts and various skill levels, and (ii) the 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 also possibly consider 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 can be taken into account using the distance between the attained and the contractual SLAs. It is necessary that the SLAs attained meet the target or contractual SLAs and hence, the need for penalizing

under-achievement in the cost function is obvious. However, if the SLAs are over-achieved, for instance, if a client requests that 95% of his high-priority SRs be closed within 4 hours and if this deadline is met 100% of the time, then it essentially translates into the time and effort of some worker(s) in meeting 100% requirement for this particular client. And, in our constrained setting, this is unnecessary as the client would in any case be happy if his SLA requirements are met at 95%. 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:

$$\begin{aligned} c(X_n) = r \times \Big(1 - \textstyle\sum_{i=1}^{|A|} \sum_{j=1}^{|B|} \alpha_{i,j} \times u_{i,j}(n)\Big) \\ +s \times \frac{\sum_{i=1}^{|C|} \sum_{j=1}^{|P|} \left|\gamma'_{i,j}(n) - \gamma_{i,j}\right|}{|C| \times |P|}, \end{aligned} \tag{2}$$

where $r, s \geq 0$ and $r + s = 1$. Further, $0 \leq \gamma_{i,j} \leq 1$ denotes the contractual SLA for client i and priority j. Note that the first term in (2) 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 Section 4.1.1. The definition of the single-stage cost $c(\cdot)$ is such that $0 \leq c(X_n) \leq 1, \forall X_n$.

4.1.1 The 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 (2), and 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 (2), 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:

$$g_{i,j}(X_n) = \gamma_{i,j} - \gamma'_{i,j}(n) \leq 0, \forall i = 1, \ldots, |C|, j = 1, \ldots, |P|, \tag{3}$$

$$h(X_n) = 1 - q(n) \leq 0. \tag{4}$$

Here (3) specifies that the attained SLA levels should be equal to or above the contractual SLA levels for each client-priority tuple. Further, (4) 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 only in the long-run average sense (see (5)).

4.2 The Constrained Optimization Problem

We want to find an optimal worker parameter θ that minimizes a certain long-run average cost objective while adhering to a set of SLA and queue stability constraints, which are long-run averages as well. The single stage cost $c(X_n)$ is used for the long run average sum in the objective, while the functions $h(X_n)$ and $g_{i,j}(X_n)$ are used for the corresponding sums in the queue-stability and SLA constraints.

The constrained optimization problem is given by

$$\begin{aligned} & J(\theta) \stackrel{\triangle}{=} \lim_{n\to\infty} \tfrac{1}{n} \sum_{m=0}^{n-1} E[c(X_m)] \\ & \text{subject to} \\ & G_{i,j}(\theta) \stackrel{\triangle}{=} \lim_{n\to\infty} \tfrac{1}{n} \sum_{m=0}^{n-1} E[g_{i,j}(X_m)] \leq 0, \\ & \qquad\qquad \forall i = 1,\ldots,|C|, j = 1,\ldots,|P|, \\ & H(\theta) \stackrel{\triangle}{=} \lim_{n\to\infty} \tfrac{1}{n} \sum_{m=0}^{n-1} E[h(X_m)] \leq 0. \end{aligned} \tag{5}$$

As illustrated in Figure 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 θ 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 θ is ergodic.

This ensures that the long-run average cost and constraint functions in (5) are well-defined for any parameter θ under the given dispatching policy. Considering the fact that the state space of the Markov process $\{X_n, n \geq 0\}$ is compact, ergodicity will follow if one ensures that there is at least one worker for each complexity class. 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 real-valued, continuous functions. The long-run average cost $J(\cdot)$ is continuously differentiable with bounded second derivative.

This is a technical requirement used for convergence. By the first part of (A2) and the fact that S is a compact space, the functions $c(\cdot)$, $g_{i,j}(\cdot)$ and $h(\cdot)$ are each uniformly bounded. The latter part of **(A2)** is needed to push through a Taylor's argument (See Appendix) to show the convergence of the scheme. Also, the long-run average cost $J(\cdot)$ itself would be shown to serve the role of a Lyapunov function for the ODE corresponding to the parameter updates, for which continuous differentiability of $J(\cdot)$ would be necessary.

In general, it is very difficult to find a globally optimal $\theta^* \in S$, i.e.,

$$\theta^* = \text{argmin} \left\{ J(\theta) \text{ s.t. } \theta \in S, G_{i,j}(\theta) \leq 0, i = 1,\ldots,|C|, j = 1,\ldots,|P|, H(\theta) \leq 0 \right\}, \tag{6}$$

Note that the parameter set here is discrete, and one requires discrete optimization schemes to find an optimum parameter. We apply the Lagrange relaxation procedure to the above problem and then provide smoothed-functional stochastic optimization algorithms with appropriate discretization procedures, for finding a locally optimum parameter θ^*.

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 θ^* for the optimization problem (5). First, we formulate the Lagrangian of (5) 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 (5) can be relaxed using the standard Lagrange multiplier theory as an unconstrained optimization problem given below.

$$\max_{\lambda} \min_{\theta} L(\theta, \lambda) \stackrel{\triangle}{=} \lim_{n\to\infty} \frac{1}{n} \sum_{m=0}^{n-1} E\left\{ c(X_m) + \lambda_f h(X_m) + \sum_{i=1}^{|C|} \sum_{j=1}^{|P|} \lambda_{i,j} g_{i,j}(X_m) \right\}. \tag{7}$$

In the above,

- $\lambda_{i,j} \geq 0, \quad \forall i = 1, \ldots, |C|, j = 1, \ldots, |P|$ represent the Lagrange multipliers corresponding to constraints $g_{i,j}(\cdot)$ and
- $\lambda_f \geq 0$, represents the Lagrange multiplier for the constraint $h(\cdot)$, in the optimization problem (5).

Let $\lambda \stackrel{\triangle}{=} (\lambda_f, \lambda_{i,j}, i = 1, \ldots, |C|, j = 1, \ldots, |P|)^T$. The function $L(\theta, \lambda)$ is commonly referred to as the Lagrangian. An optimal (θ^*, λ^*) is a saddle point for the Lagrangian, i.e.,

$$L(\theta, \lambda^*) \geq L(\theta^*, \lambda^*) \geq L(\theta^*, \lambda), \forall \theta, \forall \lambda.$$

Thus, it is necessary to design an algorithm which descends in θ and ascends in λ to find the optimum point. The simplest iterative procedure for this purpose would use the gradient of the Lagrangian with respect to θ and λ to descend and ascend respectively. However, for the given system, computation of the gradient with respect to θ would be intractable due to lack of a closed form expression of the Lagrangian. This is because, for a given staffing level specified by θ, the values of the cost function $c(X_n)$ and the constraint functions $g_{i,j}(X_n)$ and $h(X_n)$, respectively, 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 (θ^*, λ^*) would require three stages in each of its iterations.

1. The first stage which is the inner-most one, performs the service system simulation for a period $\mathcal{T}$;
2. The next outer stage updates θ along a descent direction using the simulation results from the inner most stage. In this stage, the best value of θ is computed iteratively for a given λ; and
3. The last stage which is the outer-most one, the long-run average value of each constraint is computed using the results of the inner stages and then the Lagrange multipliers λ are updated 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 increments in the outer-loop update are uniformly smaller than those of the inner-loop procedure. 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 [Borkar, 2008, Chapter 6].

5.2 Structure of the SASOC Algorithms

In a deterministic optimization setting (i.e., without stochastic noise), any negative descent algorithm for obtaining the minimum (in θ) of the Lagrangian (7) (for a given λ) would have the form

$$\theta(n+1) = \bar{\Pi}(\theta(n) - \gamma_n \mathcal{H}(\theta(n))^{-1} \nabla_\theta L(\theta(n), \lambda(n))), \tag{8}$$

where $\mathcal{H}(\theta(n))$ is a positive definite matrix and $\gamma_n > 0, \forall n \geq 0$ is a given sequence of step-size parameters. Also, $\bar{\Pi}$ is a projection operator to a compact set and will be explained in detail below. It is easy to see that $-\mathcal{H}(\theta(n), \lambda(n))^{-1} \nabla L(\theta, \lambda)$ is a descent direction, since $\mathcal{H}(\theta(n), \lambda(n))^{-1}$ 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 γ_n, it can be shown that (8) 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 for any (θ, λ) tuple, then a stochastic approximation algorithm for obtaining a saddle point of the Lagrangian (7) would update the worker parameter along a descent direction as follows:

$$\theta(n+1) = \bar{\Pi}(\theta(n) - \gamma_n \mathcal{H}_n^{-1} h_n). \tag{9}$$

In the above, h_n represents the estimate of the gradient and $\mathcal{H}_n$ the estimate of the positive definite matrix used at update instant n. The convergence of our algorithms, which have the form (9) is established by using a diminishing stepsize sequence (see (A3) below) and by employing an SF estimate of the gradient.

All the SASOC algorithms that we propose are noisy variants of (9) and use SF techniques to estimate the gradient of the Lagrangian w.r.t. θ. These algorithms have the form:

$$\left.\begin{aligned} \theta(n+1) &= \bar{\Pi}\left(\theta(n) - b(n) \nabla_\theta L(\Pi(\theta(n)), \lambda(n))\right), \\ \lambda(n+1) &= (\lambda(n) + d(n) \nabla_\lambda L(\Pi(\theta(n)), \lambda(n)))^+, \end{aligned}\right\} \tag{10}$$

where $\nabla_\theta L(\Pi(\theta(n)), \lambda(n))$ and $\nabla_\lambda L(\Pi(\theta(n)), \lambda(n))$ represent the gradients of the Lagrangian L with respect to θ and λ respectively. In the above, $\bar{\Pi}(\cdot)$ is a projection operator which ensures that updates to θ

remain bounded within $\bar{\mathcal{D}}$ which is the closed and convex hull of the discrete set $\mathcal{D}$. Thus, $\bar{\Pi}(\cdot)$ projects the updates to the nearest point in the set $\bar{\mathcal{D}} \stackrel{\triangle}{=} [0, W_{max}]^{|A|\times|B|}$. This helps in applying continuous optimization methods to the discrete setting here. The other projection operator $\Pi(\cdot)$ projects to the discrete set $\mathcal{D}$ itself. The operator $(\cdot)^+$ used for the updates of $\lambda(n)$, ensures that these updates remain non-negative. In other words, $(x)^+ = \max(x, 0)$, for any $x \in \mathbb{R}$. 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 (10). We use two simulations per iteration to estimate the gradient.

One can see by the generic update equations (10) of our algorithms, that updates to θ happen on a compact and convex set $\bar{\mathcal{D}}$ while the actual parameters that we consider for measurement of the gradient of the Lagrangian are the discretized versions of the updates of θ belonging to the discrete (finite) set $\mathcal{D}$. Upon convergence, we consider the value of θ projected to the set $\mathcal{D}$ as the final converged parameter. The two algorithms differ in the way we use the two simulations to estimate $\nabla_\theta L(\Pi(\theta), \lambda)$:

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 the Cauchy distribution results in a larger spread in value and a likely better exploration. We observe the same in our simulation results.

The overall algorithm flow can be diagrammatically represented as in Figure 3. Each iteration of the algorithm involves two simulations (each for a period $\mathcal{T}$) - 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 Figure 3. 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.

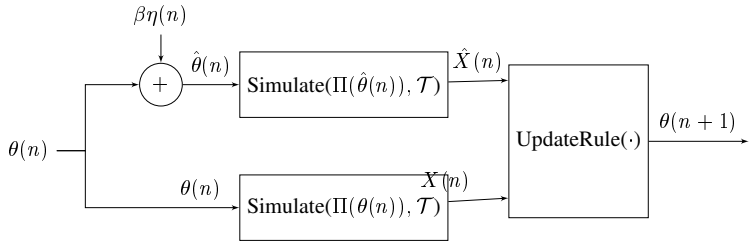


Figure 3: Overall flow of the algorithm 1.

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;

- $\beta > 0$, a fixed smoothing control parameter;
- $K \geq 1$, a fixed integer used to control the duration of the average cost accumulation (c.f. (16));
- $\{\eta(n), n \geq 1\}$, N-dimensional vector of i.i.d. Gaussian/Cauchy random variables for SASOC-SF-N/SASOC-SF-C respectively.
- UpdateRule(), the stochastic update rule of the particular algorithm.
- $\mathcal{T}$, the inter-update simulation duration described in Figure 2.
- Simulate(θ, $\mathcal{T}$) $\rightarrow X$, the simulator of the SS. Here, X represents the state of the underlying constrained Markov process at the end of the simulation.

Output: θ^*, the parameter vector after R iterations.

1. $\theta \leftarrow \theta(0)$, $\lambda \leftarrow 0$, $n \leftarrow 1$
loop
  2. $X \leftarrow$ Simulate($\Pi(\theta), \mathcal{T}$).
  3. $\hat{X} \leftarrow$ Simulate($\Pi(\theta + \beta\eta), \mathcal{T}$).
  4. $(\theta, \lambda) \leftarrow$UpdateRule($X, \hat{X}, \theta, \lambda; K$).
  if $n = R$ **then**
    5. Terminate with $\Pi(\theta)$.
  end if
  6. $n \leftarrow n + 1$.
end loop

5.2.1 The discrete projection operator, $\Pi(\cdot)$

The discrete projection operator $\Pi(\cdot)$ used in the generic update equations (10) can be (i) *deterministic* where it projects the value to the nearest point in $\mathcal{D}$, or, (ii) *randomized* in which case the value is projected to a point in $\mathcal{D}$ based on an appropriate probability measure, or else, (iii) *generalized* or *mixed* in which case the value is projected for some ranges of input, deterministically, and for the remaining probabilistically. The deterministic projection is the simplest one but it is discontinuous, which is a disadvantage for good convergence behaviour as discussed later in this section. Randomized projection solves this problem, making the projection operation continuous as well as smooth, by choosing suitable probabilities for picking a discrete parameter from $\mathcal{D}$. A typical randomized projection scheme is as follows. Let $\bar{\theta} = \left(\bar{\theta}_1, \bar{\theta}_2, \ldots, \bar{\theta}_N\right) \in \bar{\mathcal{D}}$, represent the continuous parameter which needs to be projected. Here $N = |A| \times |B|$. Depending on the value of $\bar{\theta}_j$ (for any $j = 1, \ldots, N$), one can find $\mathcal{D}^k$ and $\mathcal{D}^{k+1}$ with $\mathcal{D}^k < \mathcal{D}^{k+1}$, $\mathcal{D}^k, \mathcal{D}^{k+1} \in \mathcal{D}$ such that $\mathcal{D}^k$ and $\mathcal{D}^{k+1}$ are the immediate neighbours of $\bar{\theta}_j$ in the set $\mathcal{D}$. Then, the randomized projection is given by,

$$\theta_j = \begin{cases} \mathcal{D}^{k+1} & \text{w.p. } \dfrac{\bar{\theta}_j - \mathcal{D}^k}{\mathcal{D}^{k+1} - \mathcal{D}^k}, \\ \mathcal{D}^k & \text{w.p. } \dfrac{\mathcal{D}^{k+1} - \bar{\theta}_j}{\mathcal{D}^{k+1} - \mathcal{D}^k}. \end{cases} \tag{11}$$

The randomized scheme can be computationally expensive if the number of parameters (that is, the cardinality of the set $\mathcal{D}$) is large. We consider therefore the third scheme, the generalized projection

scheme, which brings in the simplicity of deterministic projections as well as the smoothing behaviour of randomized projections. It requires lower computational effort as compared to a randomized projection scheme while at the same time it provides smoothing and hence also a good convergence behaviour, unlike the deterministic projection scheme. The specifics of the generalized projection scheme are given below: For any $\bar{\theta} = (\bar{\theta}_1, \ldots, \bar{\theta}_N)^T$ with $\bar{\theta}_j \in [0, W_{\max}], j = 1, 2, \ldots, N$, we define a projection operator $\Pi(\bar{\theta}) = (\Pi_1(\bar{\theta}_1), \ldots, \Pi_N(\bar{\theta}_N)) \in \mathcal{D}$ which projects $\bar{\theta}$ onto the discrete set $\mathcal{D}$ as follows:

Let $\zeta > 0$ be a small fixed real number and $\bar{\theta}_i$ with $\mathcal{D}^j, \mathcal{D}^{j+1} \in \mathcal{D}$ such that, $\mathcal{D}^j \leq \bar{\theta}_i \leq \mathcal{D}^{j+1}, \mathcal{D}^j < \mathcal{D}^{j+1}$ and $\mathcal{D}^j, \mathcal{D}^{j+1}$ represent the nearest two discrete neighbours in $\mathcal{D}$ of the (continuously valued) parameter component $\bar{\theta}_i$. Let us consider an interval of length 2ζ around the midpoint of $[\mathcal{D}^j, \mathcal{D}^{j+1}]$ and denote it as $[\tilde{\mathcal{D}}_1, \tilde{\mathcal{D}}_2]$, where $\tilde{\mathcal{D}}_1 = \frac{\mathcal{D}^j + \mathcal{D}^{j+1}}{2} - \zeta$ and $\tilde{\mathcal{D}}_2 = \frac{\mathcal{D}^j + \mathcal{D}^{j+1}}{2} + \zeta$. Then, $\Pi_i(\bar{\theta}_i)$ for $\theta_i \in [\mathcal{D}^j, \tilde{\mathcal{D}}_1] \cup [\tilde{\mathcal{D}}_2, \mathcal{D}^{j+1}]$ is defined by

$$\Pi_i(\bar{\theta}_i) = \begin{cases} 0, & \text{if } \bar{\theta}_i < 0, \\ \mathcal{D}^j, & \text{if } \bar{\theta}_i \leq \frac{\mathcal{D}^j + \mathcal{D}^{j+1}}{2} - \zeta, \\ \mathcal{D}^{j+1}, & \text{if } \bar{\theta}_i \geq \frac{\mathcal{D}^j + \mathcal{D}^{j+1}}{2} + \zeta, \\ W_{\max}, & \text{if } \bar{\theta}_i \geq W_{\max}. \end{cases} \tag{12}$$

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

$$\Pi_i(\bar{\theta}_i) = \begin{cases} \mathcal{D}^j, & \text{w.p. } f(\frac{\tilde{\mathcal{D}}_2 - \bar{\theta}_i}{2\zeta}), \\ \mathcal{D}^{j+1}, & \text{w.p. } 1 - f(\frac{\tilde{\mathcal{D}}_2 - \bar{\theta}_i}{2\zeta}). \end{cases} \tag{13}$$

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 $[\tilde{\mathcal{D}}_1, \tilde{\mathcal{D}}_2]$. Further, for $\bar{\theta}_i \in [\tilde{\mathcal{D}}_1, \tilde{\mathcal{D}}_2]$, we project randomly using a continuously differentiable function f. A similar operator has been considered for projection in L.A. et al. [2012].

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 $\bar{\theta}_i \in [\mathcal{D}^j, \frac{\mathcal{D}^j + \mathcal{D}^{j+1}}{2})$ to $\mathcal{D}^j$ and $\bar{\theta}_i \in [\frac{\mathcal{D}^j + \mathcal{D}^{j+1}}{2}, \mathcal{D}^{j+1}]$ 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 θ in the convex hull $\bar{\mathcal{D}}$, the transition dynamics of the process $\{X_n, n \geq 0\}$ is not necessarily continuously differentiable, a key requirement for the analysis that follows. 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 Bhatnagar et al. [2011b] 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 ζ is small), the projection operator is essentially deterministic.

5.3 Gaussian Smoothed-Functional Algorithm

In what follows, we shall describe the gradient estimators assuming that the objective and the constraint functions are continuously differentiable as functions of a (continuously-valued) parameter. We shall later show in Section A.1 that the dynamics of the underlying Markov process can be extended to the convex hull of $\bar{\mathcal{D}}$ and that the extended objective and constraint functions under the extended dynamics are indeed continuously differentiable.

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 the SASOC-SF-N algorithm.

5.3.1 Conditions for a smoothing function

[Kreimer and Rubinstein, 1988, pp. 471] enumerates a set of conditions for a function, $h_\beta(\theta)$, to be a smoothing function. These conditions are (i) $h_\beta(\theta) = \frac{1}{\beta^{|A|\times|B|}} h_1(\theta)$, is a piece-wise differentiable function with respect to θ, (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 functionals. Let α be a $(|A| \times |B|)$-dimensional vector of $N(0, \beta^2)$ random variables with a given $\beta > 0$. Let $G_\beta(\cdot)$ denote the probability density function (p.d.f.) of α. The Gaussian smoothed functional estimate, obtained as a convolution of the gradient of the Lagrangian (with respect to θ) with the Gaussian density function $G_\beta(\cdot)$ is given by,

$$F_{\beta,1}(\theta) = \int_\alpha G_\beta(\theta-\alpha)\nabla_\alpha L(\alpha,\lambda)d\alpha. \tag{14}$$

Upon integration by parts and simplification, we get,

$$F_{\beta,1}(\theta) = \int_\alpha \nabla_\alpha G_\beta(\theta-\alpha)L(\alpha,\lambda)d\alpha.$$

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,

$$F_{\beta,1}(\theta) = \frac{1}{\beta}\int_\eta -\eta G_1(\eta)L(\theta-\beta\eta,\lambda)d\eta,$$

where $G_1(\eta)$ is the 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

$$\nabla_\theta L(\theta,\lambda) = \lim_{\beta\downarrow 0} E\left[\frac{\eta}{\beta}L(\theta+\beta\eta,\lambda)\middle|\theta,\lambda\right],$$

where the expectation is with respect to the p.d.f., $G_1(\cdot)$. Since one-simulation based estimates are known to have higher variability as opposed to two-simulation estimates (see [Styblinski and Tang,

1990, Figure 3]), we use a two simulation estimate, similar to that in [Bhatnagar et al., 2011a, CG-SF], as given below:

$$\nabla_\theta L(\theta,\lambda) = \lim_{\beta\downarrow 0} E\left[\frac{\eta}{\beta}\left(L(\theta+\beta\eta,\lambda) - L(\theta,\lambda)\right)\middle|\, \theta,\lambda\right]. \tag{15}$$

Here η is a $(|A|\times|B|)$-vector of independent $N(0,1)$ random variables.

5.3.3 SASOC-SF-N Algorithm

We estimate the quantities $L(\Pi(\theta+\beta\eta),\lambda)$ and $L(\Pi(\theta),\lambda)$ that are in turn required for estimating the gradient $\nabla_\theta L(\Pi(\theta),\lambda)$ as in (15) above, by using two iterations running with parameters $\Pi(\theta+\beta\eta)$ and $\Pi(\theta)$ on a faster time-scale. For $\lambda_{i,j}$ and λ_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 7. Since maximization of the Lagrangian with respect to $\lambda_{i,j}$ and λ_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 $n \geq 0$,

$$\left.\begin{aligned}
&W_i(n+1) = \bar{\Pi}_i\left[W_i(n) + b(n)\left(\frac{\eta_i(nK)}{\beta}(\bar{L}(nK) - \bar{L}'(nK))\right)\right], \forall i = 1,2,\ldots,|A|\times|B|,\\
&\text{where for } m = 0,1,\ldots,K-1,\\
&\bar{L}(nK+m+1) = \bar{L}(nK+m) + d(n)(l(X_{nK+m},\lambda(nK)) - \bar{L}(nK+m)),\\
&\bar{L}'(nK+m+1) = \bar{L}'(nK+m) + d(n)(l(\hat{X}_{nK+m},\lambda(nK)) - \bar{L}'(nK+m)),\\
&\lambda_{i,j}(n+1) = (\lambda_{i,j}(n) + a(n)g_{i,j}(X_n))^+, \forall i = 1,2,\ldots,|C|, j = 1,2,\ldots,|P|,\\
&\lambda_f(n+1) = (\lambda_f(n) + a(n)h(X_n))^+.
\end{aligned}\right\} \tag{16}$$

In the above,

- $l(X,\lambda) = c(X) + \sum_{i=1}^{|C|}\sum_{j=1}^{|P|}\lambda_{i,j}g_{i,j}(X) + \lambda_f h(X)$ is the single stage sample of the Lagrangian;
- X_m denotes the state at instant m, where the simulation was run with the nominal parameter $\Pi(\theta_{[\frac{n}{K}]})$, whereas $\hat{X}_m$ denotes the state at instant m where the simulation was run with the perturbed parameter $\Pi(\theta_{[\frac{n}{K}]} + \beta\eta_{[\frac{n}{K}]})$. Thus, two simulations are carried out for each iteration, one with $\Pi(\theta_{[\frac{n}{K}]})$, and the other with $\Pi(\theta_{[\frac{n}{K}]} + \beta\eta_{[\frac{n}{K}]})$, respectively. Here $[\frac{n}{K}]$ denotes the integer portion of $\frac{n}{K}$. For the sake of simplicity, we shall hereafter use $\Pi(\theta)$ to denote $\Pi(\theta_{[\frac{n}{K}]})$ and $\Pi(\theta+\beta\eta)$ to denote $\Pi(\theta_{[\frac{n}{K}]} + \beta\eta_{[\frac{n}{K}]})$ respectively;
- $\bar{L}$ and $\bar{L}'$ are initialized with zero. X and $\hat{X}$, which correspond to the two simulations with θ 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 the θ-update in relation to that of $\bar{L}$ and $\bar{L}'$. 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, however, holds for any $K \geq 1$;
- $\Pi(\cdot)$ is the generalized projection operator given in equation (12)–(13) (see Section 5.2.1). Also, $\bar{\Pi}(\cdot)$ is the regular projection operator to the set $\bar{\mathcal{D}} = [0, W_{\max}]^N$ and is defined as follows: For any $x = (x_1,\ldots,x_N)^T \in \mathbb{R}^N$, $\bar{\Pi}(x) = (\bar{\Pi}_1(x_1),\ldots,\bar{\Pi}_N(x_N))^T$, where $\bar{\Pi}_i(x_i) = \min(W_{\max}, \max(x_i, 0))$, for all $i = 1,\ldots,N$;

- $\beta > 0$ is a fixed smoothing control parameter;
- $\eta = \left(\eta_1, \eta_2, \ldots, \eta_{(|A|\times|B|)}\right)^T$ is a vector of $|A| \times |B|$ independent $N(0,1)$ random variables; and
- The step-sizes $a(n), b(n)$ and $d(n), n \geq 0$, satisfy the requirements in Assumption **(A3)** below.

To summarize, the update rule (16) 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 θ and $\theta + \beta\eta$, respectively.
2. The update to the worker parameter $W_i(\cdot)$ in (16) essentially uses the Lagrange estimates $\bar{L}$ and $\bar{L}'$ to tune W_i in roughly the negative gradient direction, i.e., $-\nabla_\theta L(\Gamma(\theta), \lambda)$. Note that the estimate used for the gradient in (16) is motivated by (15), with the primary difference being that the SASOC-SF-N algorithm uses a fixed $\beta > 0$, see Lemma 5 discussed later in the convergence analysis. The worker parameter $W_i(\cdot)$ in (16) is updated in $\bar{\mathcal{D}}$, i.e., the updates are continuously valued. The projection of $W_i(\cdot)$ to the discrete set $\mathcal{D}$, using the $\Pi(\cdot)$ operator, is done only for simulation purposes and upon convergence, for declaring the optimized solution.
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 (16) converges to a saddle point $(\Gamma(\theta^*), \lambda^*)$ of the Lagrangian.

Assumption (A3)

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

$$\sum_n a(n) = \sum_n b(n) = \sum_n d(n) = \infty;$$

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

$$\frac{b(n)}{d(n)}, \frac{a(n)}{b(n)} \to 0 \text{ as } n \to \infty.$$

The first two requirements above are standard stochastic approximation conditions, while the third requirement ensures the desired separation of time-scales between recursions governed by the various step-sizes in our algorithms. Step-sizes chosen according to **(A3)** ensure that the recursions of Lagrange multipliers $\lambda_{i,j}$ and λ_f, proceed 'slower' in comparison to those of the worker parameter θ, while the updates of the average cost - $\bar{L}$ and $\bar{L}'$ proceed the fastest. In our experiments, for all algorithms, we select the step-sizes $a(n), b(n)$ and $d(n), n \geq 0$ as follows:

$$a(0) = \hat{a},\ b(0) = \hat{b},\ d(0) = \hat{d},$$
$$a(n) = \hat{a}/n,\ b(n) = \hat{b}/n^\beta,\ d(n) = \hat{d}/n^\alpha,\ n \geq 1,$$

with $1/2 < \alpha < \beta < 1, 0 < \hat{a}, \hat{b}, \hat{d} < \infty$. The above choice of step-sizes can be seen to satisfy **(A3)**.

5.4 Cauchy Smoothed-Functional Algorithm

We now describe SASOC-SF-C, which uses the Cauchy density instead of Gaussian 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 $\Lambda = [-W_{max}, W_{max}]^N$. For some scalar constant $\beta > 0$, let

$$\hat{F}_{\beta,1} L(\theta, \lambda) = \int_{\eta} C_\beta(\theta - \eta) \nabla_\theta L(\theta, \lambda) d\eta \tag{17}$$

represent the convolution of the gradient (with respect to θ) of the Lagrangian with an N-dimensional multi-variate truncated Cauchy p.d.f.,

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

where $\theta, \eta \in \Re^N$, $\Gamma(\cdot)$ is the Gamma function and Ω is a scalar factor, see [Nadarajah and Kotz, 2007], that results from the truncation of the original Cauchy density. Truncation is necessary to ensure that the integral in (17) 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 Λ^0 is the interior of the set Λ, and following the procedure similar to that in Section 5.3.2, we obtain

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

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

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

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

$$\nabla_\theta L(\theta, \lambda) = \lim_{\beta \downarrow 0} E\left[\frac{\eta(N+1)}{\beta(1+\eta^T\eta)} L(\theta + \beta\eta, \lambda) \middle| \theta, \lambda \right]. \tag{18}$$

The estimate of $\nabla_\theta L(\theta, \lambda)$ as in (18) requires only one simulation that however exhibits more variability than two-simulation estimates (see [Styblinski and Tang, 1990, Figure 3]). In order to reduce the variability in our estimates of the gradient, we extend the estimate (18) to use two simulations as follows:

$$\nabla_\theta L(\theta, \lambda) = \lim_{\beta \downarrow 0} E\left[\frac{\eta(N+1)}{\beta(1+\eta^T\eta)} \left(L(\theta + \beta\eta, \lambda) - L(\theta, \lambda)\right) \middle| \theta, \lambda \right]. \tag{19}$$

This form of the gradient estimate is motivated from Chin [1997], Bhatnagar [2007], Bhatnagar et al. [2011a], 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 gradient estimate in (19) is indeed a valid estimate of the gradient.

5.4.2 SASOC-SF-C Algorithm

Using (19) as the gradient estimate for the Lagrangian, the overall update rule for SASOC-SF-C is as follows:

$$\left.\begin{aligned}
&W_i(n+1) = \bar{\Pi}_i\left[W_i(n) + b(n)\left(\frac{\eta_i(nK)(N+1)}{\beta(1+\eta(nK)^T\eta(nK))}(\bar{L}(nK) - \bar{L}'(nK))\right)\right], \\
&\qquad\qquad \forall i = 1, 2, \ldots, N, \\
&\text{where for } m = 0, 1, \ldots, K-1, \\
&\bar{L}(nK+m+1) = \bar{L}(nK+m) + d(n)(l(X_{nK+m}, \lambda(nK)) - \bar{L}(nK+m)), \\
&\bar{L}'(nK+m+1) = \bar{L}'(nK+m) + d(n)(l(\hat{X}_{nK+m}, \lambda(nK)) - \bar{L}'(nK+m)), \\
&\lambda_{i,j}(n+1) = (\lambda_{i,j}(n) + a(n)g_{i,j}(X_n))^+, \forall i = 1, 2, \ldots, |C|, j = 1, 2, \ldots, |P|, \\
&\lambda_f(n+1) = (\lambda_f(n) + a(n)h(X_n))^+ .
\end{aligned}\right\} \tag{20}$$

In the above, η is an N-dimensional multi-variate Cauchy random vector truncated to Λ 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 (20) of SASOC-SF-C and η being a truncated N-dimensional multi-variate Cauchy random vector, (as described previously).

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 Banerjee et al. [2011], which in turn is based on the 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.
- **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 Prashanth et al. [2011] 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 [Laguna, 1998]. OptQuest along with several other engines from Frontline Systems won the INFORMS impact award[1] in the year 2010.

The algorithms were compared using the performance metrics of W_{sum} and mean utilization. Here $W_{sum} \stackrel{\triangle}{=} \sum_{i=1}^{|A|}\sum_{j=1}^{|B|} W_{i,j}$ is the total number of workers across shifts and skill levels. The mean

[1] http://www.solver.com/press201008.htm

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 by Prashanth et al. [2011].

6.1 SASOC-SPSA

As with the SF based schemes that we proposed, SASOC-SPSA [Prashanth et al., 2011] 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. Using the notation as before, the update rule of SASOC-SPSA is as follows:

$$\left.\begin{aligned}
&W_i(n+1) = \bar{\Pi}_i\left[W_i(n) + b(n)\left(\tfrac{\bar{L}(nK)-\bar{L}'(nK)}{\beta\eta_i(n)}\right)\right], \forall i = 1,2,\ldots,N,\\
&\text{where for } m = 0,1,\ldots,K-1,\\
&\bar{L}(nK+m+1) = \bar{L}(nK+m) + d(n)(l(X_{nK+m},\lambda(nK)) - \bar{L}(nK+m)),\\
&\bar{L}'(nK+m+1) = \bar{L}'(nK+m) + d(n)(l(\hat{X}_{nK+m},\lambda(nK)) - \bar{L}'(nK+m)),\\
&\lambda_{i,j}(n+1) = (\lambda_{i,j}(n) + a(n)g_{i,j}(X_n))^+, \forall i = 1,2,\ldots,|C|, j = 1,2,\ldots,|P|,\\
&\lambda_f(n+1) = (\lambda_f(n) + a(n)h(X_n))^+.
\end{aligned}\right\} \quad (21)$$

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 (21) 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 clients. To allow a fair comparison, we choose SS having a variety of characteristics along the dimensions of geographical location, workload, number of clients supported, number of SWs, and SLA constraints. Figure 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 clients 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 seggregation is important because the SLAs apply only to C SRs. I SRs do 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 Figure 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 Banerjee et al. [2011] for the evaluation step. While a number of dispatching policies were developed in Banerjee et al. [2011], we

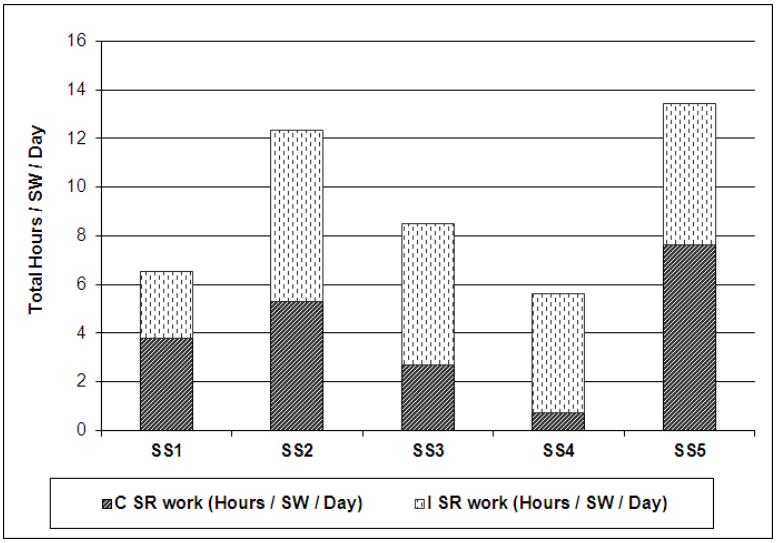
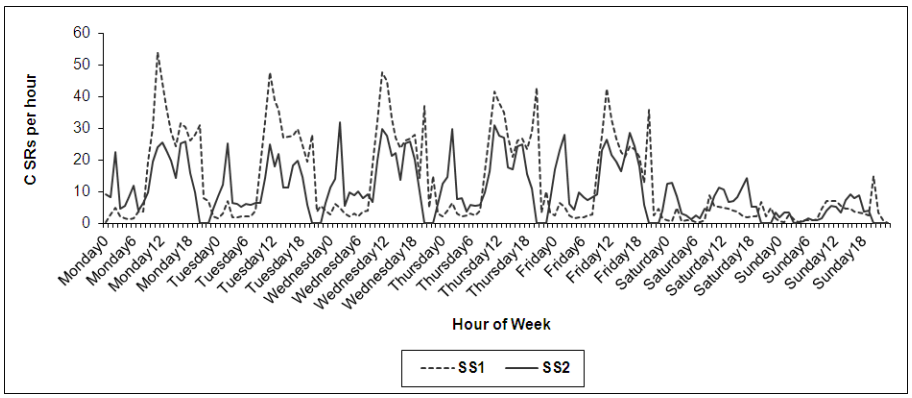
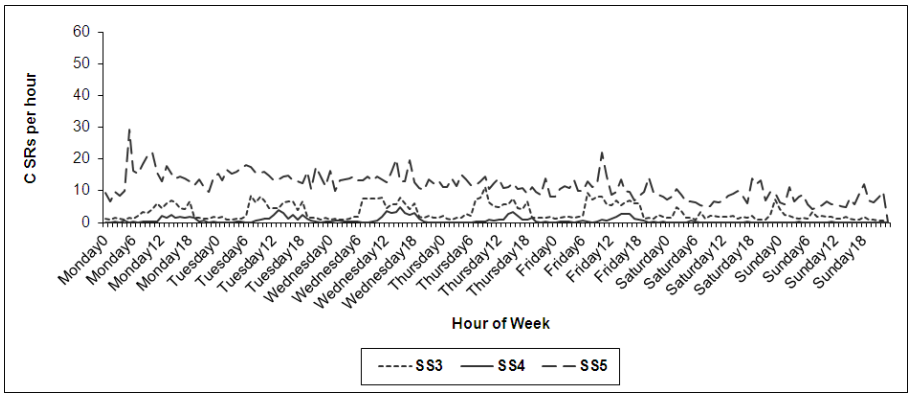


Figure 4: Total work volume statistics for each SS



(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

focus on the PRIO-PULL and EDF policies here. In the PRIO-PULL policy, the client priority field of the SRs is used to assign them to SWs, whereas in the EDF policy, the time left to SLA target is used. In the experiments carried out by Banerjee et al. [2011], EDF performed the best and PRIO-PULL performed the worst among all dispatching policies. Hence, selecting these policies exposes the SASOC algorithms to two very different operational models of SS.

We implemented all the three SASOC algorithms (including SASOC-SPSA) by invoking the simulation framework from Banerjee et al. [2011] to compute X and $\hat{X}$, corresponding to perturbed and unperturbed simulations, respectively, as shown in Algorithm 1. We also implemented 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 θ and perturbed parameter $\theta+\beta\eta$, respectively. In other words, we set 10 months as the value of the parameter $\mathcal{T}$. 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 (2), as $r = s = 0.5$, thus giving equal weight to both the components in (2). The values of β and K were set to 0.5 and 10 respectively in all our experiments. The queue stability variable q used in the constraint (4) is set to 0 (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 RAM.

6.3 Results

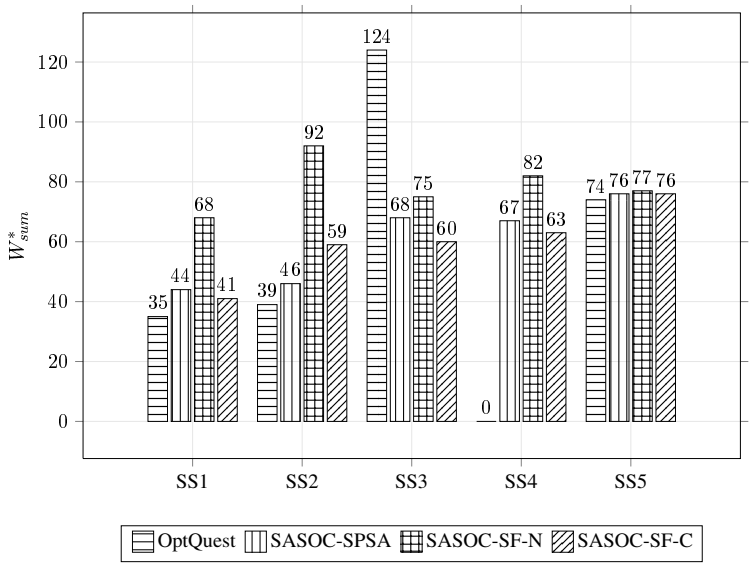
Figure 6(a) presents the W^*_{sum} achieved for OptQuest and SASOC algorithms on five real life SS, with PRIO-PULL as the underlying dispatching policy. Here W^*_{sum} denotes the value obtained upon convergence of W_{sum}. Figure 7(a) presents similar results for the case of the EDF dispatching policy.

We observe that our SASOC algorithms find a significantly better value of W^*_{sum} as compared to OptQuest on two SS pools, namely SS3 and SS4, while on SS5, SASOC algorithms perfom on par with OptQuest. Note in particular that the performance difference between the SASOC algorithms, and OptQuest on SS3 is nearly 100%. Further, on SS4, OptQuest repeatedly does not find a feasible solution even 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.

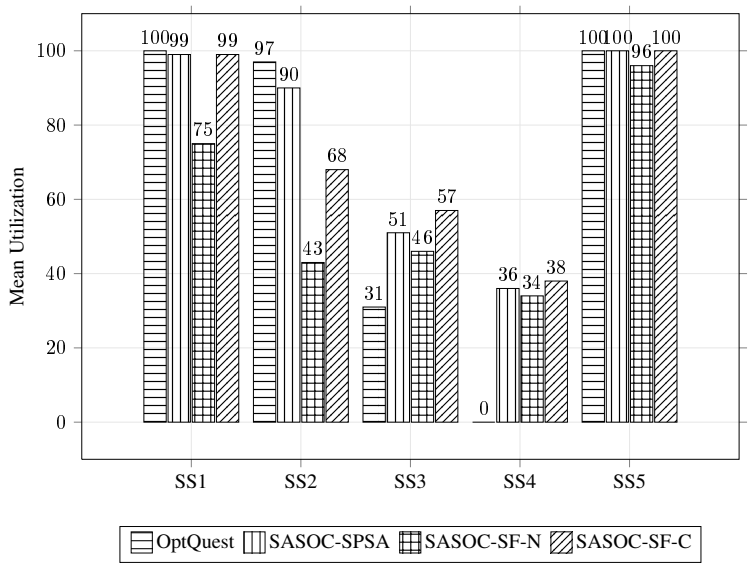
Among SASOC algorithms, the Cauchy variant of the smoothed functional algorithm SASOC-SF-C exhibits the best overall performance, under both dispatching policies considered. SASOC-SF-C outperforms the Gaussian variant of the smoothed functional algorithm SASOC-SF-N on most of the SS pools. Further, in comparison to SASOC-SPSA, SASOC-SF-C finds a better value of W^*_{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 the Cauchy distribution which results in occasionally high perturbation values about the current best estimate of the parameter and an 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^*_{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 about 3 hours to converge.

We also observe that the parameter θ (and hence W_{sum}) converges to the optimum value for each

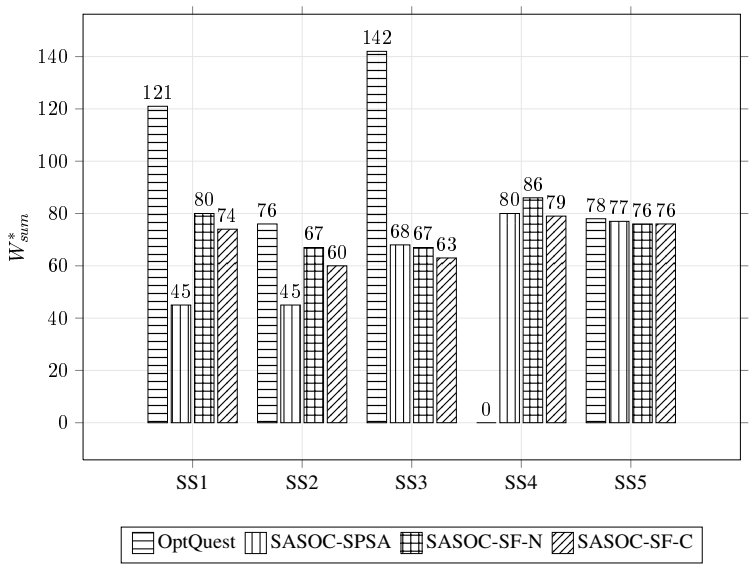


(a) W^*_{sum} achieved

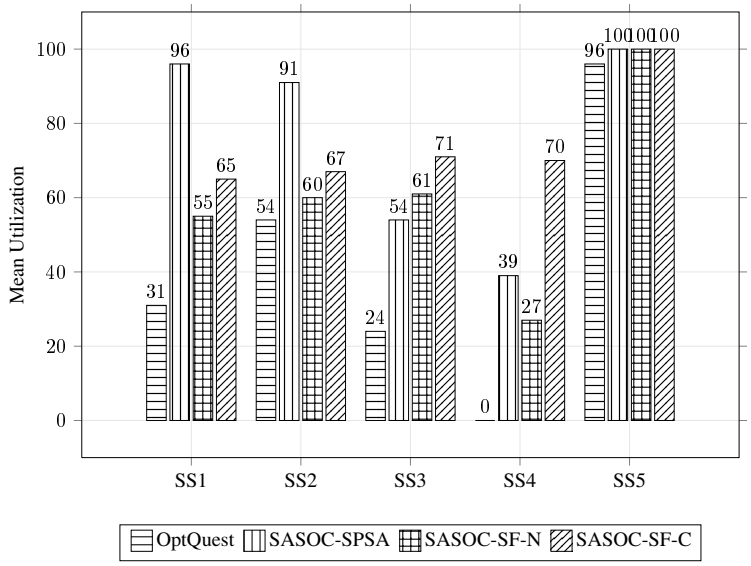


(b) Utilization of the workers across skill levels

Figure 6: Performance of OptQuest and SASOC for PRIO-PULL dispatching policy on five real SS (*Note:* OptQuest is infeasible over SS4)

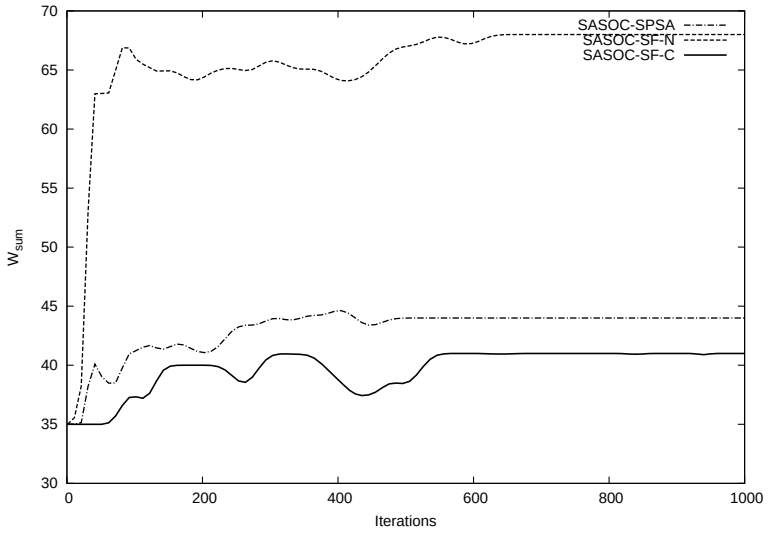


(a) W^*_{sum} achieved

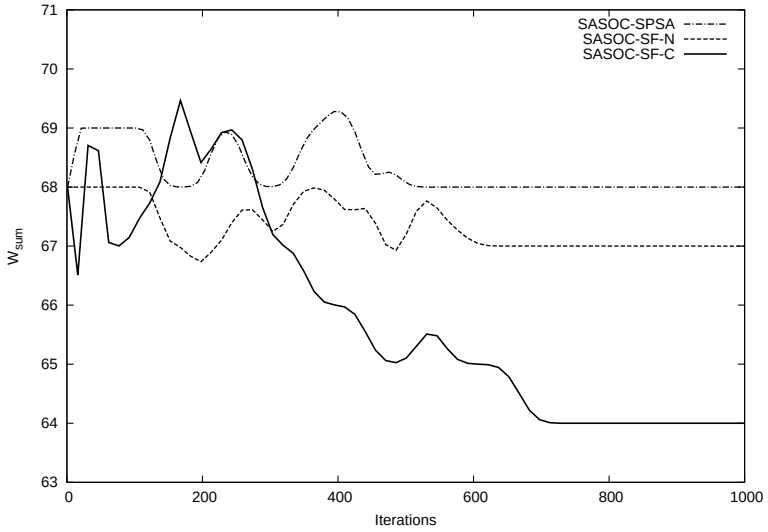


(b) Utilization of the workers across skill levels

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 integers.



(a) Using PRIO-PULL



(b) Using EDF

Figure 8: Convergence of W_{sum} - Illustration on SS1 and SS3 for two dispatching policies

SASOC algorithm - SASOC-SF-C, SASOC-SF-N and SASOC-SPSA - on each of the SS pools considered. This can be seen in the convergence plots in Figure 8(a) and Figure 8(b), where the evolution of W_{sum} is shown for all SASOC algorithms for both the dispatching policies considered. This is a significant feature of SASOC algorithms as we have established convergence of our algorithm in Appendix 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 and as mentioned above, does not converge for SS4 (see Figure 6(a)).

Figures 6(b) and 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 Figures 6(a) and 7(a) are valid here, with the SASOC algorithms showing better overall mean utilization performance as compared to 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 achieve a higher mean utilization by incorporating the utilization of workers into the single stage cost function (2).

7 Conclusions

The aim here was to optimize the 'workforce', the critical resource of any service system. However, adapting the staffing levels to the workloads in such systems is challenging due to the queue stability and aggregate SLA constraints. We formulated this problem as a parametrized Markov process with (a) a discrete worker parameter that specifies the number of workers across shifts and skill levels, (b) a novel single stage cost function that balances the conflicting objectives of utilizing workers better and attaining the target SLAs, and (c) the aggregate SLA and queue stability constraints. The aim was to find the optimum worker parameter from a discrete high-dimensional parameter set, that minimizes the long run average of the single-stage cost function, while adhering to the constraints relating to queue stability and SLA compliance. Another difficulty in finding the optimum (constrained) worker parameter is that the single stage cost and constraint functions can be estimated only via simulation.

For solving the above problem, we proposed two novel smoothed functional based algorithms. Amongst our algorithms, SASOC-SF-C is a novel discrete parameter simulation optimization scheme that uses Cauchy perturbations to estimate the gradient in the primal and unlike the algorithms proposed previously in the literature that require several simulations, SASOC-SF-C works with only two simulations. All the SASOC algorithms that we propose are online, incremental, easy to implement and with provable convergence guarantees. Further, they 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. Numerical experiments were performed to evaluate each of the algorithms using data from several 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) were 25 times faster than OptQuest, (b) consistently found solutions of good quality and in most cases better than those found by OptQuest, and (c) had convergence guarantees unlike OptQuest. 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 model of SS. Finally, we provided in an appendix, the proof of convergence of our algorithms, using the theory of multi-timescale stochastic approximation.

As future work, one could incorporate enhancements into the single-stage cost function where even monetary costs are considered apart from staff utilization and SLA attainment factors. Note that our SASOC algorithms are generic, i.e., not designed for a particular single-stage cost function and hence, can be used with other cost structures that include worker salaries. An orthogonal direction of future work in this context is to develop skill updation algorithms, i.e., derive novel work dispatch policies that improve the skills of the workers beyond their current levels by way of assigning work of a higher complexity. However, the setting is still constrained and the SLAs would need to be met while improving the skill levels of the workers. The skill updation scheme could then be combined with the SASOC algorithms presented in this paper, to optimize the staffing levels on a slower timescale.

Acknowledgements

The work of S.Bhatnagar was partially supported by the Department of Science and Technology through project no. SR/S3/EECE/0104/2012.

References

S. Alter. Service system fundamentals: Work system, value chain, and life cycle. *IBM Systems Journal*, 47(1):71–85, 2008.

S. Andradóttir. Optimization of the transient and steady-state behavior of discrete event systems. *Management Science*, 42(5):717–737, 1996.

D. Banerjee, N. Desai, and G. Dasgupta. Simulation-based evaluation of dispatching policies in service systems. In *Winter simulation conference*, 2011.

S. Bhatnagar. Adaptive multivariate three-timescale stochastic approximation algorithms for simulation based optimization. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 15(1): 74–107, 2005.

S. Bhatnagar. Adaptive Newton-based multivariate smoothed functional algorithms for simulation optimization. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 18(2):1–35, 2007.

S. Bhatnagar. An actor-critic algorithm with function approximation for discounted cost constrained Markov decision processes. *Systems & Control Letters*, 2010.

S. Bhatnagar, M.C. Fu, S.I. Marcus, and I. Wang. Two-timescale simultaneous perturbation stochastic approximation using deterministic perturbation sequences. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 13(2):180–209, 2003. ISSN 1049-3301.

S. Bhatnagar, N. Hemachandra, and V. K. Mishra. Stochastic approximation algorithms for constrained optimization via simulation. *ACM Transactions on Modeling and Computer Simulation (TOMACS)*, 21(3):15, 2011a.

S. Bhatnagar, V. Mishra, and N. Hemachandra. Stochastic algorithms for discrete parameter simulation optimization. *IEEE Transactions on Automation Science and Engineering*, 8(4):780 –793, 2011b.

S. Bhatnagar, H.L. Prasad, and L.A. Prashanth. *Stochastic Recursive Algorithms for Optimization*. Springer, 2012.

S. Bhulai, G. Koole, and A. Pot. Simple methods for shift scheduling in multi-skill call centers. *Manufacturing and Service Operations Management*, 10(3), 2008.

V. S. Borkar. An actor-critic algorithm for constrained Markov decision processes. *Systems & control letters*, 54(3):207–213, 2005.

V. S. Borkar. *Stochastic Approximation: A Dynamical Systems Viewpoint*. Cambridge Univ Press, 2008.

M. T. Cezik and P. L'Ecuyer. Staffing multiskill call centers via linear programming and simulation. *Management Science*, 54(2):310–323, 2008.

W. K. V. Chan. An analysis of emerging behaviors in large-scale queueing-based service systems using agent-based simulation. In *Winter Simulation Conference*, pages 872–878, 2008.

D. C. Chin. Comparative study of stochastic algorithms for system optimization based on gradient approximation. *IEEE Trans. Sys. Man. Cyber. – Part B*, 2(27):244–249, 1997.

E. K. P. Chong and P. J. Ramadge. Optimization of queues using an infinitesimal perturbation analysis-based stochastic algorithm with general update times. *SIAM J. Cont. and Optim.*, 31(3):698–732, 1993.

E. K. P. Chong and P. J. Ramadge. Stochastic optimization of regenerative systems using infinitesimal perturbation analysis. *IEEE Trans. Auto. Cont.*, 39(7):1400–1410, 1994.

M. C. Fu. Convergence of a stochastic approximation algorithm for the $GI/G/1$ queue using infinitesimal perturbation analysis. *J. Optim. Theo. Appl.*, 65:149–160, 1990.

M. W. Hirsch. Convergent activation dynamics in continuous time networks. *Neural Networks*, 2(5): 331–349, 1989.

Y. C. Ho and X. R. Cao. *Perturbation Analysis of Discrete Event Dynamical Systems*. Kluwer, Boston, 1991.

V. Y. Katkovnik and Y. U. Kulchitsky. Convergence of a class of random search algorithms. *Automat. Remote Control*, 8:81–87, 1972.

J. Kreimer and R. Y. Rubinstein. Smoothed functionals and constrained stochastic approximation. *SIAM Journal on Numerical Analysis*, 25(2):470–487, 1988.

H. J. Kushner and D. S. Clark. *Stochastic approximation methods for constrained and unconstrained systems*. Springer-Verlag, 1978. ISBN 0-387-90341-0.

Prashanth L.A., H.L. Prasad, N. Desai, S. Bhatnagar, and G. Dasgupta. Simultaneous perturbation methods for adaptive labor staffing in service systems. Technical report, Stochastic Systems Lab, IISc, 2012. URL `http://stochastic.csa.iisc.ernet.in/www/research/files/IISc-CSA-SSL-TR-2011-4-rev2.pdf`.

M. Laguna. Optimization of complex systems with OptQuest. *OptQuest for Crystal Ball User Manual, Decisioneering*, 1998.

P. L'Ecuyer and P. W. Glynn. Stochastic optimization by simulation: convergence proofs for the $GI/G/1$ queue in steady-state. *Management Science*, 40(11):1562–1578, 1994.

S. Nadarajah and S. Kotz. A truncated bivariate Cauchy distribution. *Bulletin of the Malaysian Mathematical Sciences Society. Second Series*, 30(2):185–193, 2007. ISSN 0126-6705.

L.A. Prashanth, H.L. Prasad, N. Desai, S. Bhatnagar, and G. Dasgupta. Stochastic Optimization for Adaptive Labor Staffing in Service Systems. *Service-Oriented Computing*, pages 487–494, 2011.

L. Ramaswamy and G. Banavar. A formal model of service delivery. In *IEEE International Conference on Services Computing*, pages 517–520, 2008.

T. R. Robbins and T. P. Harrison. A simulation based scheduling model for call centers with uncertain arrival rates. In *Winter Simulation Conference*, pages 2884–2890, 2008.

J. C. Spall. Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. *IEEE Transactions on Automatic Control*, 37(3):332–341, 1992. ISSN 0018-9286.

J. C. Spall. A one-measurement form of simultaneous perturbation stochastic approximation. *Automatica*, 33(1):109–112, 1997. ISSN 0005-1098.

J. C. Spall. Adaptive stochastic approximation by the simultaneous perturbation method. *IEEE Transactions on Automatic Control*, 45(10):1839–1853, 2000.

J. Spohrer, P. P. Maglio, J. Bailey, and D. Gruhl. Steps toward a science of service systems. *Computer*, 40(1):71–77, 2007.

M. A. Styblinski and T.S. Tang. Experiments in nonconvex optimization: stochastic approximation with function smoothing and simulated annealing. *Neural Networks*, 3(4):467–483, 1990.

A. Verma, N. V. Desai, A. Bhamidipaty, A. N. Jain, J. Nallacherry, S. Roy, and S. Barnes. Automated optimal dispatching of service requests. *SRII Global Conference*, 2011.

S. Wasserkrug, S. Taub, S. Zeltyn, D. Gilat, V. Lipets, Z. Feldman, and A. Mandelbaum. Creating operational shift schedules for third-level IT support: challenges, models and case study. *International Journal of Services Operations and Informatics*, 3(3):242–257, 2008.

A Appendix: Convergence Analysis

We collectively refer to the two algorithms as SASOC algorithms. Here we provide a sketch of the convergence of SASOC in two stages. First in Section A.1, we extend the dynamics of the underlying Markov process from a discrete-valued parameter dependent process with the parameter taking values in the set $\mathcal{D}$ to a continuous-valued parameter dependent process with the parameter taking values in $\bar{\mathcal{D}}$. We then show that for the process with extended dynamics, analyzing the algorithm for convergence in $\bar{\mathcal{D}}$ is analogous to analyzing the convergence of the algorithm for the original process in $\mathcal{D}$. Next in Section A.2, we show convergence of the two algorithms in continuous parameter.

A.1 Extension to the continuous parameter system

Here we describe the extension of the transition dynamics, i.e., $p_{i,j}(\theta)$, for any $\theta \in \bar{\mathcal{D}}$, where $\bar{\mathcal{D}}$ is the closed convex hull of the discrete set $\mathcal{D}$. Note that the discrete parameter θ of the Markov process $\{X_n(\theta)\}$ takes values in the set $\mathcal{D}$. The $p_\theta(i,j)$ in the extended setting is defined as follows:

$$p_\theta(i,j) = \sum_{k=1}^{q} \beta_k(\theta) p_{D^k}(i,j), \forall \theta \in \bar{\mathcal{D}}, i,j \in \mathcal{S}. \tag{22}$$

In the above, the weights $\beta_k(\theta)$ satisfy

$$0 \leq \beta_k(\theta) \leq 1, k = 1,2,\ldots,q \text{ and } \sum_{k=1}^{q} \beta_k(\theta) = 1.$$

Here, $D^k \in \mathcal{D}, k = 1,2,\ldots,q$, represent all the elements of the discrete parameter set $\mathcal{D}$. $p_\theta(i,j), i,j \in \mathcal{S}, \theta \in \bar{\mathcal{D}}$ can be seen to satisfy the properties of transition probabilities. Note that the weights $\beta_k(\theta)$ are required to be continuously differentiable in order to ensure that the extension (22) above is smooth. However, we do not require an explicit computation of these weights in any of the SASOC algorithms, which attempt to find the saddle point of (7). Instead, it will suffice to ensure the existence of the weights $\beta_k(\theta)$ for establishing the convergence of SASOC algorithms.

We now give a procedure for computing the weights $\beta_k(\theta)$. Consider the case when θ is one-dimensional, that is, $\theta = \theta_1$ and suppose θ_1 lies between D^j and D^{j+1} (both members of $\mathcal{D}$). One may view $\beta_k(\theta)$ as the probability with which projection is done on $[D^j, D^{j+1}]$ and is obtained using the Π-projection operator as follows: Let us consider an interval of length 2ζ around the midpoint of $[D^j, D^{j+1}]$ and denote it as $[\tilde{D}_1, \tilde{D}_2]$, where $\tilde{D}_1 = \frac{D^j + D^{j+1}}{2} - \zeta$ and $\tilde{D}_2 = \frac{D^j + D^{j+1}}{2} + \zeta$. Then, the weights $\beta_k(\theta)$ are set in the following manner: $\beta_k(\theta) = 0, \forall k \notin \{j, j+1\}$ and $\beta_j(\theta), \beta_{j+1}(\theta)$ are given by:

$$(\beta_j(\theta_1), \beta_{j+1}(\theta_1)) = \begin{cases} (1,0) & \text{if } \theta_1 \in \left[D^j, \tilde{D}_1\right], \\ (f(\frac{\tilde{D}_2 - \theta_1}{2\zeta}), 1 - f(\frac{\tilde{D}_2 - \theta_1}{2\zeta})) & \text{if } \theta_1 \in \left[\tilde{D}_1, \tilde{D}_2\right], \\ (0,1) & \text{if } \theta_1 \in \left[\tilde{D}_2, D^{j+1}\right]. \end{cases} \tag{23}$$

In the above, f is the function that is used in the Π-projection and is continuously differentiable and defined on $[0,1]$ such that $f(0) = 0$ and $f(1) = 1$. The above can be similarly extended to the case when the parameter θ has $N = |A| \times |B|$ components.

It is clear that $\beta_k(\theta), k = 1,\ldots,p$ constructed as outlined above are continuously differentiable functions of θ and hence, the extended transition probabilities $p_\theta(i,j), \forall \theta \in \bar{\mathcal{D}}, i,j \in \mathcal{S}$ are continuously differentiable as well.

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

Proof: Follows in a similar manner as [Bhatnagar et al., 2011b, Lemma 2]. ■

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

follows:

$$\begin{aligned}
\bar{J}(\theta) &\stackrel{\triangle}{=} \lim_{n\to\infty} \tfrac{1}{n} \sum_{m=0}^{n-1} c(X_m(\theta)), \theta \in \bar{\mathcal{D}} \\
\bar{G}_{i,j}(\theta) &\stackrel{\triangle}{=} \lim_{n\to\infty} \tfrac{1}{n} \sum_{m=0}^{n-1} g_{i,j}(X_m(\theta)) \leq 0, \\
&\qquad\qquad \forall i = 1, \ldots, |C|, j = 1, \ldots, |P|, \theta \in \bar{\mathcal{D}} \\
\bar{H}(\theta) &\stackrel{\triangle}{=} \lim_{n\to\infty} \tfrac{1}{n} \sum_{m=0}^{n-1} h(X_m(\theta)) \leq 0, \theta \in \bar{\mathcal{D}}.
\end{aligned} \tag{24}$$

The difference between the above and the corresponding entities defined in (5) is that θ can take values in $\bar{\mathcal{D}}$ in the above. In lieu of Lemma 1, the above limits are well-defined for all $\theta \in \bar{\mathcal{D}}$.

Lemma 2 *$\bar{J}(\theta), \bar{G}_{i,j}(\theta), i = 1, \ldots, |C|, j = 1, \ldots, |P|$, and $\bar{H}(\theta)$ are continuously differentiable in $\theta \in \bar{\mathcal{D}}$.*

Proof: Follows in a similar manner as [Bhatnagar et al., 2011b, Lemma 3]. ■

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

Lemma 3 *Under the extended dynamics, $p_\theta(i,j), i,j \in \mathcal{S}$ of the Markov process $\{X_n(\theta)\}$ defined over all $\theta \in \bar{\mathcal{D}}$, we have that the SASOC algorithms are analogous to their respective continuous counterparts where $\Pi(\theta)$ and $\Pi(\theta + \beta\eta)$ in steps 2 and 3 of Algorithm 1 are replaced by $\bar{\Pi}(\theta)$ and $\bar{\Pi}(\theta + \beta\eta)$ respectively.*

Proof: Consider the SASOC-SF-N algorithm (16). Let $\theta(m)$ be a given parameter update that lies in $\bar{\mathcal{D}}^o$ (where $\bar{\mathcal{D}}^o$ denotes the interior of the set $\bar{\mathcal{D}}$). Let $\beta > 0$ be sufficiently small so that $\bar{\theta}^1(m) = (\bar{\Pi}_j(\theta_j(m) + \beta\eta_j(m)), j = 1, \ldots, N)^T = (\theta_j(m) + \beta\eta_j(m)), j = 1, \ldots, N)^T$.

Consider now the Π-projected parameters $\theta^1(m) = (\Pi_j(\theta_j(m) + \beta\eta_j(m)), j = 1, \ldots, N)^T$ and $\theta^2(m) = (\Pi_j(\theta_j(m)), j = 1, \ldots, N)^T$, respectively. By the construction of the generalized projection operator, these parameters are equal to $D^k \in \mathcal{D}$ with probabilities $\beta_k((\theta_j(m) + \beta\eta_j(m), j = 1, \ldots, N)^T)$ and $\beta_k(\theta_j(m), j = 1, \ldots, N)^T)$, respectively. Here we view the quantities $\beta_k(\cdot)$ as probabilities, for the mapping. When the operative parameter is D^k, the transition probabilities are $p_{D^k}(i,l), i,l \in \mathcal{S}$. Thus with probabilities $\beta_k((\theta_j(m) + \beta\eta_j(m), j = 1, \ldots, N)^T)$ and $\beta_k((\theta_j(m), j = 1, \ldots, N)^T)$, respectively, the transition probabilities in the two simulations equal $p_{D^k}(i,l), i,l \in \mathcal{S}$.

Next, consider the alternative (extended) system with parameters $\bar{\theta}^1(m) = (\bar{\Pi}_j(\theta_j(m) + \beta\eta_j(m))$ and $\bar{\theta}^2(m) = \bar{\Pi}_j(\theta_j(m))$, respectively, in the two simulations. The transition probabilities are now given by

$$p_{\bar{\theta}^i(m)}(j,l) = \sum_{k=1}^{p} \beta_k(\bar{\theta}^i(m)) p_{D^k}(j,l),$$

$i = 1, 2, j, l \in \mathcal{S}$. Thus with probability $\beta_k(\bar{\theta}^i(m))$, a transition probability of $p_{D^k}(j,l)$ is obtained in the ith system. Thus the two systems (original and the one with extended dynamics) are analogous.

Now consider the case when $\theta(m) \in \partial\bar{\mathcal{D}}$, i.e., is a point on the boundary of $\bar{\mathcal{D}}$. Then, one or more components of $\theta(m)$ are extreme points. For simplicity, assume that only one component (say the ith component) is an extreme point as the same argument carries over if there are more parameter components that are extreme points. By the ith component of $\theta(m)$ being an extreme point, we mean that $\theta_i(m)$ is either 0 or $W_{\max}$. The other components $j = 1, \ldots, N, j \neq i$ are not extreme. Thus,

$\theta_i(m) + \beta\eta_i(m)$ can lie outside of the interval $[0, W_{\max}]$. For instance, suppose that $\theta_i(m) = W_{\max}$ and that $\theta_i(m) + \beta\eta_i(m) > W_{\max}$ (which will happen if $\eta_i(m) > 0$). In such a case, $\theta_i^1(m) = \Pi_i(\theta_i(m) + \beta\eta_i(m)) = W_{\max}$ with probability one. Then, as before, $\theta^1(m)$ can be written as the convex combination $\theta^1(m) = \sum_{k=1}^{p} \beta_k(\theta^1(m))D^k$ and the rest follows as before.

The same procedure can be applied to SASOC-SF-C as well. ■

As a consequence of Lemma 3, we can analyze the SASOC algorithms with the continuous parameter $\bar{\theta}$ used in place of θ and under the extended transition dynamics (22). By an abuse of notation, we shall henceforth use θ to refer to the latter.

A.2 Convergence in continuous parameter

The fastest time-scale in SASOC corresponds to the step-size sequence $\{d(n)\}$ that is used to update the Lagrangian estimates $\bar{L}$ and $\bar{L}'$ corresponding to simulations with θ 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 (7). Note that the quantities $\theta(n) \equiv \theta$, and $\lambda(n) \equiv \lambda$, as they are updated along slower time-scales can be treated to be quasi-constants for the purpose of analysis of the Lagrangian estimates. The ODE associated with the Lagrangian estimates can be written as

$$\dot{\bar{L}}(t) = L(\theta, \lambda) - \bar{L}(t), t \geq 0. \tag{25}$$

Second, we show that the parameter updates $\theta(n)$ in SASOC, converge to a limit point of the ODE,

$$\dot{\theta}(t) = \hat{\Pi}\left(-\nabla_\theta L(\theta(t), \lambda)\right), \tag{26}$$

assuming $\lambda(t) \equiv \lambda$, (the λ-update being on a slower time-scale as compared to the update of θ), and provided that the sensitivity parameter β tends to zero in the algorithm. In (26), for any bounded continuous function $\epsilon(\cdot)$,

$$\hat{\Pi}(\epsilon(\theta(t))) = \lim_{\beta \downarrow 0} \frac{\bar{\Pi}(\theta(t) + \beta\epsilon(\theta(t))) - \theta(t)}{\beta}. \tag{27}$$

Note that if $\theta(t) \in \bar{\mathcal{D}}^o$ (the interior of $\bar{\mathcal{D}}$), then $\hat{\Pi}(\epsilon(\theta(t))) = \epsilon(\theta(t))$, since $\bar{\Pi}(\theta(t) + \beta\epsilon(\theta(t))) = \theta(t) + \beta\epsilon(\theta(t))$ for $\beta > 0$ sufficiently small. Also, if $\theta(t) \in \partial\bar{\mathcal{D}}$ (boundary of set $\bar{\mathcal{D}}$), is such that $\theta(t) + \beta\epsilon(\theta(t)) \notin \bar{\mathcal{D}}$, then $\hat{\Pi}(\epsilon(\theta(t)))$ is the projection of $\epsilon(\theta(t)$ to $\bar{\mathcal{D}}$. The limit in (27) is well defined because $\bar{\mathcal{D}}$ is a compact and convex set. Further, we show that $\lambda_{i,j}$s and λ_f converge respectively to the limit points of the ODEs,

$$\dot{\lambda_{i,j}}(t) = \Gamma\left(\bar{G}_{i,j}(\theta^{\lambda(t)})\right), \forall i = 1, 2, \ldots, |C|, j = 1, 2, \ldots, |P|,$$
$$\dot{\lambda_f}(t) = \Gamma\left(\bar{H}(\bar{\theta}^{\lambda(t)})\right),$$

where θ^λ is the converged continuous parameter value of ODE (26) for a fixed value of λ. For any bounded continuous function $\bar{\epsilon}(\cdot)$,

$$\Gamma(\bar{\epsilon}(\lambda(t))) = \lim_{\beta \downarrow 0} \frac{(\lambda(t) + \beta\bar{\epsilon}(\lambda(t)))^+ - \lambda(t)}{\beta}.$$

The operator Γ is similar to $\bar{\Pi}$. From the definition of the Lagrangian given in equation (7), the gradient of the Lagrangian with respect to $\lambda_{i,j}$ can be seen to be $G_{i,j}(\theta)$ and that with respect to λ_f to be $H(\theta)$

for a given value of θ. Thus, the above ODEs suggest that in SASOC, $\lambda_{i,j}$s and λ_f are ascending in the Lagrangian value and converge to a local maximum point for a fixed value of θ. Lastly, we argue that the point to which SASOC converges to, is a (local) saddle point with it being a local minimum in θ and local maximum in the $\lambda_{i,j}$ and λ_f. We establish these convergences for the three SASOC algorithms via a sequence of Lemmas given below.

A.2.1 SASOC-SF-N

Lemma 4 *Recall that $\bar{L}(n)$ is the estimate of the Lagrangian at the n^{th} iteration in the update rule* (16) *of SASOC-SF-N. Then, $\|\bar{L}(n) - L(\theta(n), \lambda(n))\| \to 0$ w.p. 1, as $n \to \infty$.*

Proof: 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 σ-fields. Then, $(M_m, \mathcal{F}_m), m \geq 0$, forms a martingale difference sequence. Now since,

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

and $c(\cdot), g_{i,j}(\cdot), h(\cdot)$, are continuous functions, and the state space (S) of $\{X_n\}$ is compact, i.e., is a bounded and closed set, the aforementioned functions are uniformly bounded. Now since λ is fixed, we have

$$|l(X,\lambda)| \leq \bar{C}_\lambda$$

for some $\bar{C}_\lambda > 0, \forall X$ and given λ. It is now easy to see that $(N_m, \mathcal{F}_m), m \geq 0$, 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,

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

almost surely, since $\sum_n d^2(n) < \infty$, and moreover, $E[M_{n+1}^2|\mathcal{F}_n], n \geq 0$, is uniformly bounded almost surely by the foregoing. Now, note that since the process $\{X_n\}$ is ergodic Markov for any given θ, 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 [Borkar, 2008, Chapter 6.2] for a detailed discussion of natural time-scale convergence). The rest follows from the Hirsch lemma [Hirsch, 1989, Thm. 1, pp. 339], applied to the ODE (25). ■

On similar lines, $\|\bar{L}'(n) - L(\theta(n) + \beta\eta(n), \lambda(n))\| \to 0$ w.p. 1, as $n \to \infty$. Now, since the updates of λ are occurring on the slowest time-scale, we let $\lambda(n) \equiv \lambda, \forall n$ for the analysis of the θ-recursion. Define

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

Lemma 5 *As $\beta \to 0$,*

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

Proof: Follows from [Bhatnagar et al., 2011a, Proposition 4.2]. ■

Let $K^\lambda = \{\theta \in \bar{\mathcal{D}} | \bar{\Pi}(-\nabla L(\theta, \lambda)) = 0\}$ denote the set of fixed-points of the ODE (26). Let $\bar{K}^\lambda \subset K^\lambda$, denote the set of stable fixed points of the ODE (26). Note that K^λ may contain unstable attractors such as local maxima, saddle points etc., in addition to local minima. For a given $\epsilon > 0$, let

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

denote the set of all points that are in the ϵ-neighborhood of the set K^λ, and let

$$(P^\lambda)^\epsilon = \{\hat{\theta} \in \mathcal{D} | \hat{\theta} = \Pi(\theta), \theta \in (K^\lambda)^\epsilon\},$$

denote the set of all discrete parameter values in the discrete-neighborhood of the set $(K^\lambda)^\epsilon$.

Theorem 6 *Under (A1)-(A3), given $\epsilon > 0$, $\exists \beta_0 > 0$, s.t. $\forall \beta \in (0, \beta_0]$, $\theta(n) \to \theta^* \in (P^\lambda)^\epsilon$, with probability one as $n \to \infty$.*

Proof: As before, one can let $\lambda(n) \equiv \lambda, \forall n$, for the analysis of the update of the parameter θ. Let $\mathcal{G}_m = \sigma\left(X_n, \hat{X}_n, \theta(n), n \leq m; \eta(n), n < m\right), m \geq 1$, denote a sequence of associated σ-fields. In lieu of Lemma 3, we consider $\bar{\Pi}$ instead of Π in steps 2 and 3 of Algorithm 1. Rewrite the parameter update as: For $i = 1, 2, \ldots, |A| \times |B|$,

$$\begin{aligned} W_i(n+1) = \bar{\Pi}_i\Big(& W_i(n) - b(n)\nabla_{\theta,i} L(\theta(n), \lambda) \\ & + b(n)\left(F^i_{\beta,2} L(\theta(n), \lambda) - \nabla_{\theta,i} L(\theta(n), \lambda)\right) + b(n)\xi^i_{n+1}\Big), \end{aligned} \tag{28}$$

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

$$\sum_{n=0}^{\infty} E[(\bar{M}^i_{n+1} - \bar{M}^i_n)^2 | \mathcal{G}_n] = \sum_n b^2(n) E[(\xi^i_{n+1})^2 | \mathcal{G}_n] < \infty$$

almost surely, since $L(\cdot, \lambda)$ is uniformly bounded and so is $F^i_{\beta,2} L(\theta(n), \lambda)$. Also, $\sum_n b^2(n) < \infty$ and $E[\eta^2_i(n)] = 1 < \infty$. Thus, by the martingale convergence theorem, $(\bar{M}^i_n, \mathcal{G}_n), n \geq 0$, is an almost surely convergent martingale sequence. Now, $\theta(n) \to (K^\lambda)^\epsilon$ as $n \to \infty$, follows as a consequence of [Kushner and Clark, 1978, Theorem 5.3.3, pp. 191-196]. Thus, $\Pi(\theta(n)) \to \theta^* \in (P^\lambda)^\epsilon$. ■

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 $(P^{\lambda(n)})^\epsilon$, i.e., would have converged. Let $\theta^{\lambda(n)} \in (P^{\lambda(n)})^\epsilon$, denote the θ-parameter to which the θ-update converges (given $\lambda(n)$). Let

$$\begin{aligned} F = \{\lambda \geq 0 | & \Gamma(\bar{H}(\bar{\theta}^\lambda)) = 0, \\ & \Gamma(\bar{G}_{i,j}(\bar{\theta}^\lambda)) = 0, \forall i = 1, \ldots, |C|, j = 1, \ldots, |P|, \bar{\theta}^\lambda \in (K^\lambda)^\epsilon\}. \end{aligned}$$

Theorem 7 *$\lambda(n) \to F$ with probability one as $n \to \infty$.*

Proof: Rewrite the update of $\lambda_{i,j}$ in (16) as follows: For all $i = 1, \ldots, |C|, j = 1, \ldots, |P|$,

$$\lambda_{i,j}(n+1) = \Big(\lambda_{i,j}(n) + a(n)\bar{G}_{i,j}(\theta^{\lambda(n)}) + N^1_{i,j}(n) + M^1_{i,j}(n+1)\Big)^+,$$

$$\lambda_f(n+1) = \Big(\lambda_f(n) + a(n)\bar{H}(\theta^{\lambda(n)}) + N^2(n) + M^2(n+1)\Big)^+,$$

where $N^1_{i,j}(n) = E[g_{i,j}(X_n)|\mathcal{F}_{n-1}] - \bar{G}_{i,j}(\theta^{\lambda(n)})$, $M^1_{i,j}(n+1) = (g_{i,j}(X_n) - E[g_{i,j}(X_n)|\mathcal{F}_{n-1}])$, $n \geq 1$, respectively. Similarly, $N^2(n) = E[h(X_n)|\mathcal{F}_{n-1}] - \bar{H}(\theta^{\lambda(n)})$, $M^2(n+1) = (h(X_n) - E[h(X_n)|\mathcal{F}_{n-1}]), n \geq 1$. Note that $N^1_{i,j}(n) \to 0$ almost surely as $n \to \infty$ along the 'natural' time-scale which is clearly faster than the time-scale of the algorithm. See [Borkar, 2008, 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 $\sum_{m=0}^{n} a(m)M^1_{i,j}(m+1), n \geq 1$, is an almost surely convergence martingale sequence. The claim now follows from the [Kushner and Clark, 1978, Theorem 5.3.3, pp. 191-196]. ■

A.2.2 SASOC-SF-C

Let

$$\hat{F}_{\beta,2}L(\theta(n),\lambda) = E\Bigg(\frac{\eta(n)(N+1)}{\beta(1+\eta(n)^T\eta(n))}\Big(L(\theta(n)+\beta\eta(n),\lambda) - L(\theta(n),\lambda)\Big)\Big|\theta(n),\lambda\Bigg).$$

Lemma 8 *As $\beta \to 0$,*

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

Proof: Follows in a similar manner as [Bhatnagar et al., 2011a, Proposition 4.2]. ■

Also, note that $E[\eta_i^2(n)] < \infty$ as $\eta(n)$ has a truncated Cauchy density. The rest of the analysis follows as for SASOC-SF-N.

H. L. Prasad received his Bachelor's degree in Engineering in the field of Electrical and Electronics, from National Institute of Technology Karnataka, Surathkal, in the year 2005. He obtained his Master's degree in Engineering in the field of Systems Science and Automation (SSA), from Indian Institute of Science, Bangalore, in the year 2007. He worked as a Project Associate from 2007 to 2008 on a DST sponsored project. He has currently submitted his PhD thesis in the area of stochastic games and stochastic optimization techniques, under the guidance of Prof. Shalabh Bhatnagar of the department of Computer Science and Automation at the Indian Institute of Science, Bangalore.

He, along with Prashanth, collaborated with Dr. Nirmit Desai of IBM Research, India, for over an year, in the area of service systems. His research interests are in stochastic games, dynamic mechanism design, stochastic control and optimization, reinforcement learning and its applications.

Prashanth L A was born in Gauribidanur, India. He received his B.E. degree in Computer Science from National Institute of Technology, Surathkal, India, in the year 2002, and the M.Sc.(Engg.) degree from Indian Institute of Science, in the year 2008. He submitted his PhD thesis in the Department of Computer Science and Automation at the Indian Institute of Science, Bangalore in the year 2012 and is currently a Postdoctoral Researcher with Remi Munos at INRIA team SequeL, Lille, France.

He was with Texas Instruments (India) Pvt Ltd, Bangalore, India for over six years, as a Senior Software Systems Engineer. His research interests are in stochastic control and optimization, reinforcement learning and its application to road traffic control, service systems and wireless networks.

Shalabh Bhatnagar received a Bachelors in Physics (Hons) from the University of Delhi in 1988. He received his Masters and Ph.D degrees in Electrical Engineering from the Indian Institute of Science, Bangalore in 1992 and 1997, respectively.

He was a Research Associate at the Institute for Systems Research, University of Maryland, College Park, during 1997 to 2000 and a Divisional Postdoctoral Fellow at the Free University, Amsterdam, during 2000 to 2001. He is currently working with the Department of Computer Science and Automation at the Indian Institute of Science, Bangalore where he is a Professor. He has also held visiting positions at the Indian Institute of Technology, Delhi and the University of Alberta, Canada.

Dr.Bhatnagar's interests are in stochastic control, reinforcement learning, and simulation optimization. He is mainly interested in applications in communication and wireless networks, and more recently in vehicular traffic control. He has authored or co-authored more than 100 research articles in various journals and conferences. He has received a young scientist award from the Systems Society of India in 2007 and two outstanding young faculty awards from Microsoft Research India in 2007 and 2008. He was an Associate Editor of the IEEE Transactions on Automation Science and Engineering during 2007 to 2011. He is a Senior Associate of the Abdus Salam International Center for Theoretical Physics (ICTP), Trieste, Italy and a Professional Member of the ACM.

Nirmit Desai is a Senior Researcher at IBM Research, India. He is interested in the analytical and socio-technical aspects of service systems. Nirmit graduated with a Ph.D. in Computer Science from North Carolina State University (NCSU), USA in 2007. His dissertation work focused on the representation and reasoning problems in modeling cross-organizational interactions and was recognized by the NCSU College of Engineering Dissertation Award. Nirmit has published more than forty papers to leading journals and conferences in the area of AI and services.