Switching Constrained Max-Weight Scheduling for Wireless Networks

Soumya Basu*, Sanjay Shakkottai†
Department of ECE, The University of Texas at Austin
Email: *basusoumya@utexas.edu, †shakkott@mail.utexas.edu

***Abstract*—We consider the wireless scheduling problem of jointly activating/de-activating base-stations and (opportunistically) scheduling from among the active base stations. Such systems are of increasing relevance in emerging wireless networks with dense overlapping coverage, where it suffices for only a (time-varying) subset of the base-stations to be active at any given time to satisfy traffic demands. In addition to queue stability (to ensure that traffic demands are met), we focus on optimizing for costs arising due to activating base-stations (switching base-station state between active/inactive), and maintaining activation (these costs arising due to energy consumption).**

We propose two algorithms—LASS-Static and LASS-Dynamic (LASS: Learning Aided Switching and Scheduling), both of which are explore-exploit policies for base-station switching and channel scheduling. In our setting, the switching action consists of two key decisions: when to switch, and what base-station activation state to switch to. Both LASS-Static and LASS-Dynamic determine the resulting switching state (i.e. 'what to switch to'), as well as the schedule using current queue-lengths and (estimated) channel states. The crucial difference is in 'when to switch'—LASS-Static determines these statically (motivated by an epsilon-greedy bandit approach), whereas LASS-Dynamic does so using current queue-lengths (thus correlating switching times, switching states and schedules). For either algorithm, existing Lyapunov-based techniques fail to establish stability, as the switching state dynamics correlate the base-station activation decisions with the channel evolution over time. Using novel drift based techniques, in this paper we derive stability, and provide explicit bounds on the expected cost and queue lengths for both algorithms. Furthermore, we show that adaptively selecting switching times in LASS-Dynamic results in an improved upper-tail of queue lengths compared to LASS-Static.

***Index Terms*—Switching cost, Constrained Max-weight**

I. Introduction

Ultra-densification, which is essential for supporting ever-increasing data traffic [1], [2], has become a core feature in modern cellular networks [3], [4]. Through overlapping coverage, such ultra-densification has led to high spectral efficiency and can support large traffic rates [4]. However, this has led to increased energy consumption of base-stations (BSs) [5], [6], raising questions about the environmental impact and economic feasibility [7], [8]. One approach to address this is to extend the framework of opportunistic scheduling, where in addition to the task of scheduling users to channels, we could also schedule the activation state (switch on or off) of base-stations based on instantaneous traffic demands. A series of papers on dynamic resource optimization and utility maximization [9]–[13], can be brought to bear on this problem. These studies have culminated in general techniques for designing greedy resource allocation algorithms through Lyapunov optimization, where bounded delay and energy efficiency is maintained by greedily activating/ deactivating BSs depending on the queue length fluctuations of the system.

However, under the fast dynamics of today's networks, these greedy strategies often lead to frequent switching between *active* and *sleep* modes of BSs, bringing switching costs to the forefront [12], [14]. Indeed, active to sleep mode transitions incur costs for hand-off and state exchange between BSs, and inactive to active transitions may incur start-up costs [12]. Moreover, from hardware standpoint feasibility of such high frequency switching remains questionable [15].

The current paper addresses the question of jointly optimizing over the operational and switching costs arising in wireless networks while providing delay guarantees [12], [14]. We propose two algorithms LASS-Static and LASS-Dynamic (LASS: Learning Aided Switching and Scheduling) that perform, without any prior knowledge of channel statistics, BS switching, activation and channel scheduling in an explore-exploit fashion. Using a novel *switching constrained max-weight* based activation and scheduling strategy we provide QoS guarantees for both the algorithms (LASS-Dynamic having guarantees superior to LASS-Static) while maintaining a near optimal aggregate cost. We next provide the details of the proposed algorithms, their corresponding guarantees, and technical contributions.

A. Main Contributions

The operation of our time varying wireless network consists of three decisions at each time slot. The first decision is 'when to switch': choosing a time slot where a new subset is activated, followed by 'what to switch to': selecting a subset of BS, a.k.a. *switching state*, (using channel estimates). Finally, 'what to schedule': observing channels and scheduling channels to users from active BSs. Furthermore, the channel estimation is dependent on BS activation as the state of the channel can only be observed with all BSs active, thus making our algorithms explore-exploit in nature. The challenge is to orchestrate all the above processes in such a manner that joint *stability* and *optimality* (w.r.t. operational and switching cost) of the system is guaranteed.

In the algorithmic side, our contributions include improvements from the state-of-the art in two stages.

1) LASS-Static: Statically constrained Max-weight: The algorithm LASS-Static (Algo. 1 along with Algo. 2) decides the switching state ('what to switch') and the channel schedules using current queue lengths, in two stages.

In the first stage, at each time slot, the algorithm allows switching ('when to switch') with a constant probability, ϵ_s, independently (of all other events), thus maintaining an explicit (slow) time-scale for switching, and limiting the switching cost. If switching is allowed, it chooses a BS subset that maximizes a drift-plus-penalty function (parameterized with V) [10] computed using channel estimate (*Max-weight activation*). When switching is not allowed it sticks to the switching state from previous slot, thus correlating available channels with past decisions. These past decisions are again correlated with queue lengths due to max-weight activation. Finally, it decides whether to exploit (choose the current switching state) or explore (activate all the BSs), where the latter happens independently with probability $O(\log(t)/t)$ in time slot t.

In the second stage, after channel observation, it opportunistically choses a schedule, feasible w.r.t. the active BS, that maximizes drift (*Max-weight scheduling*).

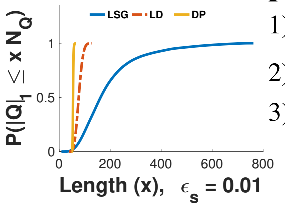
2) LASS-Dynamic: Switch queue constrained Max-weight: The second algorithm LASS-Dynamic differs from the first in 'when to switch', as it switches based on dynamics of two states, a *switch counter* and a *switch queue*, which evolve jointly with the queues of the system. The *switch counter* encodes the time since an optimal BS subset was activated last, and it is reset to zero whenever this happens. The *switch queue*, admits one packet iff a switching event occurs, and, releases one packet (if non-empty) with probability ϵ_s independently in each time slot. Finally, a *switching event* occurs when the switch counter becomes larger or equal to the current switch-queue length. Thus, all the variables become correlated. More importantly, switching instances are decided based on queue length dynamics.

3) Performance guarantees for LASS-Static and Dynamic: We prove that for both LASS-Static and LASS-Dynamic algorithms the joint dynamics of rate allocation, activation, and learning—stabilizes the system for all $\epsilon_s \in (0, 1)$ (switching parameter), and finite V (drift-plus-penalty parameter), while the average cost can be steered arbitrarily close to the optimal cost by choosing sufficiently small ϵ_s and large V. Furthermore, we characterize how the upper tail of sum-of-queue lengths decay with time for all $t \geq 1$. Specifically, both LASS—Static and Dynamic, enjoy exponential decay. However, the latter has an order-wise larger decay-rate than the former.

Let ϵ_g be the capacity gap of the system (defined in Sec. III), ϵ_s and V be two tunable parameters of the algorithm. Our theoretical guarantees are summarized in Fig. 1. See Theorem V.1 and Theorem V.3 in Sec.V for formal results.

4) Proof technique: Difficulties and Novelties: The proposed algorithms have two key features: constrained switching and vanishing exploration, which make the analysis challenging.

a) Key challenges: *Constrained switching* correlates the available control decisions over time: for LASS-Static the *switching states* become correlated, and, for LASS-Dynamic both



Performance Bounds

1) Average Queue Size: $O\left(\frac{V}{\epsilon_g} + \frac{1}{\epsilon_g \epsilon_s}\right)$

2) Average Cost Gap: $O\left(\epsilon_s + \frac{1}{V\epsilon_s}\right)$

3) Decay-rate[1] of Queue Size:

- LASS-Static: $\Theta(\epsilon_s \epsilon_g)$
- LASS-Dynamic: $\Theta(\epsilon_g)$

Fig. 1: **Left:** Cdf of sum-of-queue lengths (ϵ_s=0.01, V=250, ϵ_g=0.1). Where, LSG: LASS-Static, LD: LASS-Dynamic, and DP: Max-weight without switching constraint (baseline). **Right:** Theoretical bounds for LASS-Static/Dynamic.

switching instances and *switching states* become correlated over time. Due to the above correlation, in both algorithms, expected Lyapunov drift, may not become negative in any *deterministic* number of steps. Furthermore, we consider a system where unknown channel statistics is learned through *vanishing exploration*. This makes the drift *non-stationary* and *correlated* in nature. Therefore, the existing Lyapunov drift based techniques for proving joint stability and optimality of greedy resource allocation fail to handle such correlations and learning dynamics. Further, for similar reasons the existing non-asymptotic analysis for queue lengths ([16],[17]) is insufficient for our algorithms.

b) Stability and Optimality: To study constrained switching we introduce the notion of *drift regret*. Specifically, the drift is divided into two parts, 1) drift from the unconstrained drift-plus-penalty algorithm [18] with known channel statistics (*exact optimal*), and 2) difference of drift between the *exact optimal* and the proposed algorithm—the *drift regret*. Let the *time since last switching instance* at time-slot t be denoted as $T(t)$. We show that, for all $t \geq 1$, the *drift regret* of the proposed algorithms remain bounded as $O(T(t) + \log(t)/t)$ w.p. $(1-\Omega(1/t^{10}))$. Finally, we devise techniques to bound $T(t)$ in LASS-Dynamic.[2] In LASS-Dynamic the joint dynamics of the switch queue and the switch counter is not additive, as the switch counter resets on each switching event. To address this, we bound $T(t)$ using a *novel Lyapunov function* which is *quadratic in switch queue* length and *linear in switch counter*.

c) Non-asymptotic bounds: The novel component in our non-asymptotic bound is bounding the moment generating function (mgf) of the *drift regret*. Firstly, the learning rate $O(\log(t)/t)$ in itself is insufficient: at time t the probability of error events decay as $t^{-log(t)}$, whereas the mgf of drift regret may grow as $exp(t)$ in the worst case. We introduce the idea of *fallback*: detect extreme events and take *immediate* actions to circumvent such events. This ensures such extreme events do not propagate in time. In our system *fallback* amounts to activating *all* BSs if queues are $\Omega(\log(t))$. Secondly, bounding the mgf of the switch counter and the switch queue requires a new drift analysis technique due to their non-linear co-evolution and their interaction with the physical queues. By creating a separate coupling for each switching interval and finally combining them we arrive at the mgf bound.

[1]For a random variable X, Decay-rate$(X) \equiv -\lim_{x\to\infty} \frac{1}{x} \log \mathbb{P}[X > x]$.

[2]In LASS-Static bounding $T(t)$ is standard.

II. System Model

Our system model and notation follows from [14]. We consider a discrete time wireless network where downlink traffic for N_u users are served by N_m base stations (BS).

A. Network, BS Configurations and Costs

We assume that the users and BSs are connected (partially) forming a *bipartite graph* (possibly incomplete) $\mathcal{G}$, where an edge $(m, u) \in \mathcal{G}$ denotes connectivity between user u and BS m. Our system allows for two distinct mode of operation for each BS, *active* mode and *sleep* mode. Service is available only from the BSs in active mode.

A *BS configuration* denotes a subset of BSs. We assume that in each time slot only configurations belonging to a collection of subsets $\mathcal{J} \subseteq 2^{[N_m]}$ can be activated simultaneously. Let us denote the BS configuration active in time slot t as $\mathbf{J}(t)$, where $J_m(t) = 1$ implies that BS m is active in time slot t. We assume that it is allowed to activate all the BSs at once, i.e. $\mathbf{1}_{N_m} \in \mathcal{J}$. When it is clear from context, we may denote 'all active' configuration ($\mathbf{1}_{N_m}$) as $\mathbf{1}$.[3]

In our model, each BS expends zero energy (see remark afterwards for generalizations) in sleep mode, while in active mode it expends c_1 units of energy per time slot. This constitutes the *operational cost* of the system and it is given as $c_1\|\mathbf{J}(t)\|_1$. Additionally, when in two subsequent time slots the BS configuration is switched the energy cost is given by c_0 units. The *switching cost* in time t is given as $c_0\mathbb{1}(\mathbf{J}(t) \neq \mathbf{J}(t-1))$. Therefore, the *aggregate cost* (or simply cost) equals: $\boxed{C(t) = c_0\mathbb{1}(\mathbf{J}(t) \neq \mathbf{J}(t-1)) + c_1\|\mathbf{J}(t)\|_1.}$

B. Queues and Arrival Model

In the above network, each BS maintains a separate queue for each user it is connected to. Packets bound for the users accumulate in these queues and are served in a first-in-first-out (FIFO) manner. Specifically, the queues in time t, for each $t \geq 1$, are collectively denoted as $\mathbf{Q}(t) = \{Q_{m,u}(t) | (m, u) \in \mathcal{G}\}$. The number of queues is denoted as N_Q.

The queues are fed by exogenous arrivals, where on each time t, $A_{m,u}(t)$ new packets (unit sized) arrive in the queue $Q_{m,u}(t)$. The arrival vector in time t is denoted as $\mathbf{A}(t) = \{A_{m,u}(t) | (m, u) \in \mathcal{G}\}$. We assume that the arrival process is i.i.d. (independent of all past arrivals, past channels and the system). However, the arrivals across different queues in a given time slot maybe correlated. The arrival process has mean $\mathbb{E}[\mathbf{A}(1)] = \boldsymbol{\lambda}$ with bounded support $[\mathbf{0}, A\mathbf{1}]$.

C. Channel States, Visibility and Schedules

In our model, the evolution of the channels is dictated by an underlying *channel state* process, $H(t)$, for $t \geq 1$, where each $H(t)$ takes value in a finite set $\mathcal{H}$. We assume that the *channel state* process is i.i.d. (independent of all past arrivals, channel and system state) with *channel state probability* $\boldsymbol{\mu}$.

The instantaneous channel states $H(t)$ are only visible after BS activation is completed. Furthermore, we assume that the channel state visibility is not uniform across all BS configurations. In particular, on time t from the active configuration $\mathbf{J}(t)$ the *visible channel state* is $H(t)|\mathbf{J}(t)$, where $h|j$ is defined for all $h \in \mathcal{H}$ and $j \in \mathcal{J}$. Activating all BSs provide the complete information about the channels, thus we have $h|\mathbf{1}_{N_m} = h$ for all channel state $h \in \mathcal{H}$.

The feasible service in each time slot depends on both the channel and the active BS configuration. We model this after [14] as a set of feasible *rate vectors* or *schedules*, $\mathcal{R}(j, h)$ for all $j \in \mathcal{J}$ and $h \in \mathcal{H}$. We assume that given the observed channel state $h|j$ the set of rate vectors for configuration j can be computed. Specifically, we assume $\mathcal{R}(j, h|j) = \mathcal{R}(j, h)$ for all $h \in \mathcal{H}$ and $j \in \mathcal{J}$. We impose a monotonicity structure on the sets $\mathcal{R}(j, h)$, in its first argument. For all $j_1, j_2 \in \mathcal{J}$ such that $j_1 \subseteq j_2$, we have $\mathcal{R}(j_1, h) \subseteq \mathcal{R}(j_2, h)$. This readily implies that maximum service is possible when all BSs are activated, i.e. $\mathcal{R}(j, h) \subseteq \mathcal{R}(\mathbf{1}, h)$ for all $j \in \mathcal{J}$ (see the remark afterwards for more discussion). The maximum possible service to one queue admits an upper bound R. The maximum feasible service or arrival to one queue is denoted as $B_{max} = \max\{A, R\}$.

D. BS Switching and Scheduling

In the beginning of time slot t, the system rests in state $\mathbf{J}(t-1)$ with queue lengths $\mathbf{Q}(t)$. In each time slot t, four sequential events occur. In the order of their occurrences, the events are: 1) BS Activation, 2) Channel Observation, 3) Packet Arrival, and 4) Scheduling.

BS Activation: Based on switching dynamics a new BS configuration $\mathbf{J}(t)$ is activated with a cost $C(t)$.

Channel Observation: Next, the channel state $H(t)$ is realized, while only the visible channel state $H(t)|\mathbf{J}(t)$ is revealed. Based on the visible channel state $\mathcal{R}(\mathbf{J}(t), H(t))$ is computed.

Packet Arrival: In the following step, exogenous packets $\mathbf{A}(t)$ arrive and are added to the queues.

Scheduling: Finally, a schedule $\mathbf{S}(t) \in \mathcal{R}(\mathbf{J}(t), H(t))$ is chosen for service, where $S_{m,u}(t)$ denotes allocated service to queue $Q_{m,u}(t)$ for all $(m, u) \in \mathcal{G}$. Based on the allocated schedule packets are served and the queue lengths get updated to

$$\mathbf{Q}(t+1) = (\mathbf{Q}(t) + \mathbf{A}(t) - \mathbf{S}(t))^+ . \quad (1)$$

Remarks on System Model. 1) **Network Cost:** The network cost can be generalized to scenarios, where different BS configurations require different operational powers. Furthermore, it is possible to allow for switching cost that varies with the specific switching involved: from j_1 to j_2; as long as the cost for switcihng is not less than being in the same state. In particular, the cost function can be generalized to $C(t) = C_1(\mathbf{J}(t)) + C_0(\mathbf{J}(t), \mathbf{J}(t-1))$, where $0 \leq C_1(j) \leq C_{max,1} < \infty$ for all $j \in \mathcal{J}$, and $0 \leq C_{min,0} \leq C_2(j_1, j_2) \leq C_{max,0} < \infty$ and $C_2(j_1, j_1) = C_{min,0}$, for all $j_1, j_2 \in \mathcal{J}$.

2) **Channel Rate Vectors:** The assumption that for all $j_1, j_2 \in \mathcal{J}$, $\mathcal{R}(j_1, h) \subseteq \mathcal{R}(j_2, h)$, is not always true. This is due to the fact that active BSs may cause interference

[3]By bold face symbols we denote vectors throughout the paper. The symbol $\mathbf{1}_n$ denotes the all 1 vector of dimension n.

causing loss of possible services. We can generalize the assumption to: there exists a *known* j^*, such that for all $j \in \mathcal{J}$, $\mathcal{R}(j,h) \subseteq \mathcal{R}(j^*,h)$. In that scenario during fall-back events (see Section IV) we activate the configuration j^*, instead of $\mathbf{1}_{N_m}$, to retain all the guarantees of this paper.

III. Capacity and Cost Optimality

We now present some necessary definitions and notations that will be used throughout the paper (some borrowed from [14]). We denote by $\mathbb{E}_\phi[\cdot]$ and $\mathbb{P}_\phi[\cdot]$, the expectation and probability, resp., under a policy ϕ (defined shortly). Further, let $\mathcal{P}(\mathfrak{S})$ denote the probability simplex on a set $\mathfrak{S}$.

A *network* is defined uniquely by the channel rate vector $\boldsymbol{\mu}$. A network and an arrival rate $\boldsymbol{\lambda}$ jointly characterize a *system* $(\boldsymbol{\lambda}, \boldsymbol{\mu})$. $\Psi_{sys}(t) = \{\mathbf{Q}(t), \mathbf{J}(t-1)\}$ denotes the system state at the beginning of time slot t, for all $t \geq 1$. A *policy* is defined as a sequence of switching and scheduling actions, $\{(\mathbf{J}(t), \mathbf{S}(t)) : t \geq 1\}$. A policy ϕ is *causal* if in time slot t the activation, $\mathbf{J}(t)$, and scheduling, $\mathbf{S}(t)$, are functions of the history $\{\Psi_{sys}(t') | 1 \leq t' \leq t\}$ and system statistics (full information setting). A causal policy ϕ is *admissible* if under the policy ϕ the system state $\boldsymbol{\Psi}_{sys} = \{\Psi_{sys}(t) | t \geq 1\}$ forms an *irreducible* and *aperiodic* discrete time *countable state* Markov chain.

We now define the stability of a system and the capacity region of a given network.

Definition III.1 (Average queue length and Stability)**.** *The long term average queue length of a system with arrival rate vector $\boldsymbol{\lambda}$, channel rate vector $\boldsymbol{\mu}$ under a policy ϕ, is*

$$\mathcal{Q}^\phi(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \limsup_{M \to \infty} \tfrac{1}{M} \sum_{t=1}^{M} \mathbb{E}_\phi \left[\|\mathbf{Q}(t)\|_1 | \, \Psi_{sys}(1) \right].$$

A policy ϕ stabilizes the system if $\mathcal{Q}^\phi(\boldsymbol{\lambda}, \boldsymbol{\mu}) < \infty$.

The above definition of system stability, further, ensures that under an admissible policy the queue length process $\mathbf{Q}(t)$ is positive recurrent, implying the queue lengths are bounded with probability 1. It also implies that the Markov chain $\boldsymbol{\Psi}_{sys}$ is positive recurrent as $\mathcal{J}$ is finite.

Definition III.2 (Capacity Region)**.** *For a network with channel rate vector $\boldsymbol{\mu}$ the capacity region $\Lambda(\boldsymbol{\mu})$ is the set of all arrival rates $\boldsymbol{\lambda}$, for which the system $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is stable under some admissible policy.*

The capacity region of a network with channel rate vectors $\boldsymbol{\mu}$, is characterized as: $\boldsymbol{\lambda} \in \boldsymbol{\Lambda}(\boldsymbol{\mu})$ if and only if there exists $\boldsymbol{\alpha}(\mathbf{1}, h) \in \mathcal{P}(\mathcal{R}(\mathbf{1}, h))$, $\forall h \in \mathcal{H}$, such that $\boldsymbol{\lambda} \prec \sum_h \mu_h \sum_{r \in \mathcal{R}(\mathbf{1}, h)} \alpha_r(\mathbf{1}, h) \mathbf{r}$ (see [19], also [14]). Further, for a network $\boldsymbol{\mu}$, and an arrival rate vector $\boldsymbol{\lambda}$ the *capacity gap* is $\epsilon_g = \max\{\epsilon : \boldsymbol{\lambda} {=} (\boldsymbol{\lambda}' {-} \epsilon \mathbf{1})^+, \boldsymbol{\lambda}' \in \Lambda(\boldsymbol{\mu})\}$. From hereon we focus on systems with strictly positive capacity gap.

Following [14], the cost of a policy for a given network and a given arrival rate vector is defined as follows.

Definition III.3 (Average Network Cost)**.** *The long term average cost of a system $(\boldsymbol{\lambda}, \boldsymbol{\mu})$, under a policy ϕ, is*

$$\mathcal{C}^\phi(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \limsup_{M \to \infty} \tfrac{1}{M} \sum_{t=1}^{M} \mathbb{E}_\phi \left[C(t) | \, \Psi_{sys}(1) \right].$$

Definition III.4 (Optimal cost and Optimal policy)**.** *The optimal cost of a system $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is denoted by $\mathcal{C}^*_\Phi(\boldsymbol{\lambda}, \boldsymbol{\mu})$, and is defined as $\mathcal{C}^*(\boldsymbol{\lambda}, \boldsymbol{\mu}) = \inf\{\mathcal{C}^\phi(\boldsymbol{\lambda}, \boldsymbol{\mu}) \mid \mathcal{Q}^\phi(\boldsymbol{\lambda}, \boldsymbol{\mu}) < \infty, \phi \in$ 'admissible'$\}$. An admissible policy ϕ with average cost $\mathcal{C}^*(\boldsymbol{\lambda}, \boldsymbol{\mu})$ is optimal.*

The characterization of an optimal policy among the class of Markov policies was given in [14]. However, the authors argued: due to computational complexity it is practical to optimize with respect to the *operational cost*, $\underline{\mathrm{C}}(t) = c_1 \|J(t)\|_1$ with an additional constraint on the BS switching rate.

We take a different approach towards proving optimality of the proposed policies over the class of *admissible policies*. For any system $(\boldsymbol{\lambda}, \boldsymbol{\mu})$, we show existence of policies that have *operational cost* arbitrarily close to $\mathcal{C}^*(\boldsymbol{\lambda}, \boldsymbol{\mu})$. Later we use properties of such policies to provide our joint optimality (with switching and operational cost) and queue length guarantees.

A static-split policy, on each time slot $t \geq 1$, (i) first activates BS configuration j w.p. $\sigma(j)$, and then, after observing channel state $H(t) = h$, (ii) schedules a rate vector $\boldsymbol{r} \in \mathcal{R}(j, h)$ w.p. $\alpha_r(j, h)$. The following theorem proves optimality of a static split policy (in the above sense).

Theorem III.5. *For a system $(\boldsymbol{\lambda}, \boldsymbol{\mu})$ with capacity gap $\epsilon_g {>} 0$, $\forall \epsilon'_g {\in} (0, \epsilon_g)$, there exists a static-split policy $\phi(\epsilon'_g)$ such that*
1) $\phi(\epsilon'_g)$ stabilizes arrival rate $\boldsymbol{\lambda}'$, i.e. $\mathcal{Q}^{\phi(\epsilon'_g)}(\boldsymbol{\lambda}', \boldsymbol{\mu}) < \infty$, where $\lambda'_{m,u} = \lambda_{m,u} + \epsilon'_g \mathbb{1}(\lambda_{m,u} > 0)$ for all $(m, u) \in \mathcal{G}$.
2) the average cost of $\phi(\epsilon'_g)$ w.r.t. cost function $\underline{\mathrm{C}}(t)$ satisfies,

$$\underline{\mathcal{C}}^{\phi(\epsilon'_g)}(\boldsymbol{\lambda}, \boldsymbol{\mu}) \leq \mathcal{C}^*(\boldsymbol{\lambda}, \boldsymbol{\mu}) + \kappa_{\boldsymbol{\mu}} \epsilon'_g,$$

where $\kappa_{\boldsymbol{\mu}}$ is independent of arrival rate vector $\boldsymbol{\lambda}$ and the capacity gap ϵ_g, and may depend on the network $\boldsymbol{\mu}$.

IV. Policies with Explicit Learning

In this section we design two algorithms for BS activation and rate allocation/scheduling with the purpose of stabilizing the system with near optimal cost (both operation and switching). In the proposed algorithms, we decouple the optimization of operational and switching costs. Specifically, to bound switching cost the BS switching rate is constrained, and to minimize the operational cost, on each switching instance, a BS configuration is activated using a greedy drift-plus-penalty maximization. The drift-plus-penalty based activation allows the algorithms to adapt to the queue lengths, resulting in drastically reduced queue lengths compared to stationary BS activation scheme in [14].

The algorithms differ in the way they allow BS switching while satisfying the switching constraint. Given an upper bound on switching rate, say $\epsilon_s > 0$, the first algorithm allows switching at each time slot with probability ϵ_s independent of any other events. This limits the speed at which the algorithm can react to the changes in queue lengths, and as a result the queues occasionally become very large. In our last algorithm, we address this issue by making the switching adaptive to the changes in the system. This reduces the occasional large queue backlogs, which we show theoretically and validate through simulations.

A. Learning Aided Policy

We next specify the proposed max weight based algorithm augmented with BS activation and learning of channel states. The algorithm has three parts, (1) BS Activation, (2) Channel Scheduling, and (3) Channel Estimation.

The two proposed algorithms have similarities in structure, channel scheduling, and channel estimation. In our presentation, we represent this common structure in Algorithm 1 which performs channel scheduling and channel estimation, and delegates the BS activation to another process called ACTIVATE. We now describe Algorithm 1.

Algorithm 1: In each time slot $t \geq 1$ Algorithm 1 first passes the current queue length, $\mathbf{Q}(t)$, and channel state estimate, $\hat{\boldsymbol{\mu}}(t)$, to ACTIVATE and waits for a response. The ACTIVATE routine returns a BS configuration $\mathbf{J}(t)$. Upon receiving $\mathbf{J}(t)$, Algorithm 1 activates all BS $j \in \mathbf{J}(t)$. In the next step, the current channel state is observed from the activated BSs, $H(t)|\mathbf{J}(t)$. Then a schedule, $\mathbf{S}(t)$, is chosen greedily following Max-weight algorithm (2). Next, after the new packets arrive, it serves the packets according to the chosen schedule and updates the queues. Finally, if all BSs are active, i.e. $\mathbf{J}(t)=\mathbf{1}_{N_m}$, the channel state estimate $\hat{\boldsymbol{\mu}}(t)$ is updated.

B. Base Station Activation

The main focus of the paper is designing algorithm for BS Activation to, simultaneously, stabilize the system and guarantee near optimality in terms of operational and switching cost. Additionally, the proposed algorithm should integrate smoothly with channel scheduling and estimation. We propose two different algorithms for the ACTIVATE routine and explain them separately. The BS activation mechanism on switching instances are the same, however, the decision of when to switch is different for the two proposed algorithms.

Algorithm 2: LASS with Static Switching

In each time slot $t \geq 1$ the Algorithm 2 first obtains the current queue lengths $\mathbf{Q}(t)$ and the channel state estimate $\hat{\boldsymbol{\mu}}(t)$ from Algorithm 1. The algorithm maintains an internal state $\mathbf{J}_A(t)$ to keep track of the past activated BSs. It first statically decides 'when to switch': with probability ϵ_s it allows switching at each time t, otherwise, it keeps the internal state unchanged, i.e. $\mathbf{J}_A(t) = \mathbf{J}_A(t-1)$. When switching is allowed, it computes a new optimal BS configuration $\mathbf{J}^*_A(t)$ following the drift-plus-penalty method (Equation (3)). Then it sets $\mathbf{J}_A(t) = \mathbf{J}^*_A(t)$. Finally, in time slot t, with probability $2\log(t)/t$, it decides to *explore*, otherwise it may *exploit* or *fall-back*. In an *explore* step it passes the configuration $\mathbf{1}_{N_m}$ (all ON) to Algorithm 1 (to improve estimate of $\boldsymbol{\mu}$). If not exploring, it decides to *fall back* when the sum of

Algorithm 1 Learning-Aided Switching & Scheduling (LASS)

```
1:  Input: Initial State $\mathbf{Q}(1)$,
2:         Arrivals $\mathbf{A}(t)$ and channel states $H(t)$, $\forall t \geq 1$,
3:         BS configurations $\mathbf{J}(t)$, $\forall t \geq 1$.
4:  Initialize: Channel estimate, $\hat{\boldsymbol{\mu}}(1) = \mathbf{0}$,
5:              Number of exploration events, $n_{ex}(1) = 0$.
6:  for all $t \geq 1$ do
      1) BS Activation:
7:    Pass $\mathbf{Q}(t)$, and $\hat{\boldsymbol{\mu}}(t)$ to ACTIVATE (Algo. 2 or 3)
8:    Wait and receive $\mathbf{J}(t)$ from ACTIVATE (Algo. 2 or 3)
9:    Activate BS configuration $\mathbf{J}(t)$
      2) Channel Scheduling:
10:   Observe, $H(t)|\ \mathbf{J}(t)$          ▷ Partially if $\mathbf{J}(t)\neq\mathbf{1}_{N_m}$
11:   Select schedule $\mathbf{S}(t)$ as follows,
          $$\mathbf{S}(t) = \underset{\mathbf{r}\in\mathcal{R}(\mathbf{J}(t),H(t)|\ \mathbf{J}(t))}{\arg\max} \langle \mathbf{Q}(t), \mathbf{r} \rangle. \quad (2)$$
12:   Wait and receive new arrivals $\mathbf{A}(t)$
13:   Update queues, $\mathbf{Q}(t+1) \leftarrow (\mathbf{Q}(t) + \mathbf{A}(t) - \mathbf{S}(t))^+$
      3) Channel Estimation:
14:   if $\mathbf{J}(t) = \mathbf{1}_{N_m}$ then ▷ Updated under full observation
15:       $n_{ex}(t+1) \leftarrow n_{ex}(t) + 1$
16:       $\hat{\boldsymbol{\mu}}(t+1) \leftarrow \hat{\boldsymbol{\mu}}(t)\left(1 - \frac{1}{n_{ex}(t+1)}\right) + \frac{\mathbf{e}_{H(t)}}{n_{ex}(t+1)}$
17:   else          ▷ Unchanged under partial observation
18:       $n_{ex}(t+1) \leftarrow n_{ex}(t)$
19:       $\hat{\boldsymbol{\mu}}(t+1) \leftarrow \hat{\boldsymbol{\mu}}(t)$
```

queue lengths is larger than $Q_{th}\log(t)$, where $Q_{th} > 0$ is a constant. In this case it passes the configuration $\mathbf{1}_{N_m}$ (to ensure maximum service). Otherwise, it *exploits* by the passing configuration $\mathbf{J}_A(t)$ to Algorithm 1 (to jointly optimize and stabilize the system under the switching constraint).

- **Fallback:** The fallback is an approach to tackle the extreme error event where aggregate queue length crosses threshold $Q_{th}\log(t)$. Such error events imply that despite positive capacity gap, the system is 'apparently' unstable. In that case, the algorithm uses the *knowledge* that activating all BSs maximizes service, and falls back to that choice.
- **Penalty parameter:** The parameter V in the Equation (3) determines the (sub) optimality of the proposed algorithm, the larger the value of V the smaller the sub optimality of the algorithm. It is one of the central ideas in drift-plus-penalty based network optimization [18].
- **Internal State:** The exploration or fall-back events do not affect the evolution of the internal state $\mathbf{J}_A(t)$, whereas it changes the *switching state* $\mathbf{J}(t)$. This ensures the exploration or fall-back decisions do not propagate in time.

Algorithm 3: LASS with Dynamic Switching

In each time slot $t\geq 1$, Algorithm 3 first receives the current queue length and channel state information from Algorithm 1.

It first decides 'when to switch'. To dynamically make the decision it maintains two additional states, switch queue $Q_{sw}(t)$ and switch counter $T(t)$. The switch queue $Q_{sw}(t)$ is a token queue. Packets are served from the switch queue at a constant rate of ϵ_s, whereas, a packet enters the queue if a

Algorithm 2 ACTIVATE with Static Switching

```
1:  Parameter: Penalty scale $V$, and switching rate $\epsilon_s$,
2:  fallback threshold $Q_{th}$
3:  Input: Initial State $\mathbf{J}_A(0)$,
4:  Queue lengths $\mathbf{Q}(t)$, and channel estimate $\hat{\boldsymbol{\mu}}(t)$, $\forall t \geq 1$.
5:  for all $t \geq 1$ do
6:      Wait and receive $\mathbf{Q}(t)$ and $\hat{\boldsymbol{\mu}}(t)$ from Algorithm 1
        1) Static Switching and Dynamic Activation
7:      Generate independent r.v. $E_s(t) \sim Ber(\epsilon_s)$
8:      if $E_s(t) = 1$ then                                  ▷ Switch
9:          $\mathbf{J}_A(t) \leftarrow \mathbf{J}^*_A(t) = \arg\max_{j \in \mathcal{J}} \mathrm{DP}(j,t)$, where
            $\mathrm{DP}(j,t) = \sum_{h \in \mathcal{H}} \hat{\mu}_h(t) \max_{\mathbf{r} \in \mathcal{R}(j,h)} (\langle \mathbf{Q}(t), \mathbf{r} \rangle - V c_1 \|j\|_1)$   (3)
10:     else
11:         $\mathbf{J}_A(t) \leftarrow \mathbf{J}_A(t-1)$
        2) Explore, Fallback, or Exploit
12:     Generate independent r.v. $E_{ex}(t) \sim Ber\left(\frac{2\log(t)}{t}\right)$
13:     if $(E_{ex}(t) = 1) \vee (\|\mathbf{Q}(t)\|_1 > Q_{th}\log(t))$ then
14:         Pass $\mathbf{J}(t) = \mathbf{1}_{N_m}$ to Algo. 1   ▷ Explore/ Fallback
15:     else
16:         Pass $\mathbf{J}(t) = \mathbf{J}_A(t)$ to Algo. 1          ▷ Exploit
```

Algorithm 3 ACTIVATE with Dynamic Switching

```
1:  Parameter: Same as Algo. 2        Input: Same as Algo. 2
2:  Initialize: Switch Queue, $Q_{sw}(1) = 0$, and $T(1) = 0$.
3:  for all $t \geq 1$ do
4:      Wait and receive $\mathbf{Q}(t)$ and $\hat{\boldsymbol{\mu}}(t)$ from Algorithm 1
        1) Dynamic Switching and Activation
        1.1) Switch Queue Update
5:      $E_{ext}(t) \leftarrow \mathbb{1}(T(t) \geq Q_{sw}(t))$ ▷ Arrival to $Q_{sw}$/Switch
6:      Generate $E_s(t) \sim Ber(\epsilon_s)$            ▷ Service to $Q_{sw}$
7:      $Q_{sw}(t+1) \leftarrow (Q_{sw}(t) + E_{ext}(t) - E_s(t))^+$
        1.2) Dynamic Activation
8:      Compute $\mathbf{J}^*_A(t) = \arg\max_{j \in \mathcal{J}} \mathrm{DP}(j,t)$ [see, Eq. (3)]
9:      if $E_{ext}(t) = 1$ then
10:         $\mathbf{J}_A(t) \leftarrow \mathbf{J}^*_A(t)$         ▷ BS configuration Switch
11:     else
12:         $\mathbf{J}_A(t) \leftarrow \mathbf{J}_A(t-1)$
        1.3) Switch Counter Update
13:     if $\mathbf{J}_A(t) = \mathbf{J}^*_A(t)$ then
14:         $T(t+1) \leftarrow 0$                    ▷ Reset switch counter
15:     else
16:         $T(t+1) \leftarrow T(t) + 1$  ▷ Increment switch counter
        2) Explore, Fallback, or Exploit: Same as Algo. 2
```

switching happens. Thus a finite size of switch queue ensures that the (asymptotic) switching rate is less or equal to ϵ_s.

The switching happens whenever the switch counter $T(t)$ (described shortly) is greater or equal to the switch queue $Q_{sw}(t)$, indicated by $E_{ext}(t) = \mathbb{1}(T(t) \geq Q_{sw}(t))$. The algorithm next computes an *estimated optimal* BS config $\mathbf{J}^*_A(t)$ following drift-plus-penalty maximization, to decide 'what to switch to' (if switching is allowed). For dynamic activation, it sets $\mathbf{J}_A(t){=}\mathbf{J}^*_A(t)$ if switching is allowed (i.e., $E_{ext}(t){=}1$), otherwise, it sets $\mathbf{J}_A(t){=}\mathbf{J}_A(t{-}1)$.

The switch counter $T(t)$ encodes the time since the internal state was equal to the estimated optimal ($\mathbf{J}_A(t') = \mathbf{J}^*_A(t')$). The algorithm does so by increasing the counter when $\mathbf{J}_A(t) \neq \mathbf{J}^*_A(t)$, and reseting it to 0, otherwise. In the final step, explore/fall-back/exploit happens similar to Algorithm 2.

• **Switch Queue and Switch Counter:** In Algorithm 3 we ensure that if the switch queue is large switching is less aggressive. This enables the switch queue to drain before a new arrival in switch queue occurs. As a result, the switch queue rarely becomes large. This in turn ensures that the switch counter does not grow large. Indeed, inside any two consecutive switching instances (excluding the later instance) the switch counter is always less than the switch queue.

• **Internal Reset and External Switching:** The switch counter reset occurs due to two possible events. Firstly, a switching can occur as $T(t)$ becomes large or equal to $Q_{sw}(t)$, which makes $J_A(t) = J^*_A(t)$ triggering a reset. Secondly, if $J_A(t-1) = J^*_A(t)$ then, even if $T(t) < Q_{sw}(t)$, we have $J_A(t) = J_A(t-1) = J^*_A(t)$ resulting in a switch counter reset without an arrival to the switch queue. We denote the latter as an *internal reset* event, whereas the former is called an *external switching* event.

• **Time-scale Separation in LASS:** Both the proposed algorithms operate in three timescales. In the slowest time scale, at a vanishing rate of $O(\log(t)/t)$ the algorithms learn the channel state distribution by activating all the BSs. The rate used here ensures that the costly exploration vanishes asymptotically, while the error in channel estimation does not jeopardize the BS activation.

In the intermediate time scale BS switching occurs at a rate of $\epsilon_s < 1$, an algorithmic parameter. In the switching instances, a BS configuration is chosen greedily depending upon a drift-plus-penalty value and it remains fixed until the next switching instance. In Algo. 2 this second time scale is explicit, whereas in Algo. 3 this second time scale is maintained implicitly through the stabilization of the switch queue $Q_{sw}(t)$.

In the third and the fastest time scale, conditioned on the activated BS configuration a new schedule is chosen in each time slot. Updating the schedule at *each time slot* is the key to opportunistically use the time-varying channels and is necessary for throughput optimality.

V. Performance Guarantees of LASS

In this section we provide performance guarantees of the two proposed policies. Due to lack of space, the proofs are deferred to [20].

The combined algorithm and system state is $\Psi(t) = (\mathbf{J}_A(t{-}1), \mathbf{Q}(t), \mathbf{Q}_{sw}(t), \hat{\boldsymbol{\mu}}(t), T(t))$. Here, $T(t)$ denotes the duration from the last instance the estimated optimal BS config was identical to the chosen BS config, i.e. $T(t) = \inf\{t{-}t' : t' \leq t, \mathbf{J}_A(t') = \mathbf{J}^*_A(t')\}$. Also, $Q_{sw}(t)$ denotes the switch queue. By convention, for LASS-Static the switch queue $Q_{sw}(t) = 0$ for all $t \geq 1$. The *drift-plus-penalty* function with operational cost as penalty and *penalty scale* as $V > 0$ is $(L(t+1) - L(t) + Vc_1\|\mathbf{J}(t)\|_1)$, where $L(t) = \frac{1}{2}\sum_{(m,u)\in\mathcal{G}} Q_{m,u}^2(t)$.

Our first key result provides bounds on the expected average of sum-of-queue lengths, $\mathcal{Q}(\boldsymbol{\lambda},\boldsymbol{\mu})$, and the expected average cost (operational and switching cost) $\mathcal{C}(\boldsymbol{\lambda},\boldsymbol{\mu})$. Formally,

Theorem V.1. *For system $(\boldsymbol{\lambda},\boldsymbol{\mu})$ with a capacity gap $\epsilon_g > 0$, under LASS-Static or LASS-Dynamic with parameters $\epsilon_s \in (0,1]$ and $V > 0$,*
1) the expected cost satisfies,
$$\mathcal{C}(\boldsymbol{\lambda},\boldsymbol{\mu}) \le \mathcal{C}^*(\boldsymbol{\lambda},\boldsymbol{\mu}) + c_0\epsilon_s + \frac{4B_{\max}^2 N_Q}{V\epsilon_s},$$
2) the expected queue length satisfies,
$$\mathcal{Q}(\boldsymbol{\lambda},\boldsymbol{\mu}) \le (1.25C^*(\boldsymbol{\lambda},\boldsymbol{\mu}) + \kappa_{\boldsymbol{\mu}}\epsilon_g)\frac{V}{\epsilon_g} + \frac{(4+\epsilon_g/2)B_{\max}^2 N_Q}{\epsilon_g\epsilon_s}.$$

The following bound on the one-step expectation of drift-plus-penalty in Lemma V.2 plays key role in the proof of the above theorem.

Lemma V.2 (Parameterized Drift plus Penalty)**.** *For system $(\boldsymbol{\lambda},\boldsymbol{\mu})$ with a capacity gap $\epsilon_g > 0$, for algorithm $\phi \in \{$LASS-Static, LASS-Dynamic$\}$ with parameters $\epsilon_s \in (0,1)$ and $V > 0$, the following drift plus penalty equation holds for all time slots $t \ge 1$, and for any $\epsilon_g' \in (0,\epsilon_g)$ and $\gamma \in (0,1)$,*
$$\mathbb{E}_\phi\left[L(t+1) - L(t) + (1-\gamma)\epsilon_g'\|\mathbf{Q}(t)\|_1 + Vc_1\|\mathbf{J}(t)\|_1 \,\middle|\, \Psi(t)\right] < VC_{opt}(\epsilon_g') + B_{\max}^2 N_Q(1 + (1+\gamma\epsilon_g')T(t)) + err(t) \quad (4)$$
where, i) $err(t)$ satisfies $\limsup_{M\to\infty} \frac{1}{M}\sum_{t=0}^{M} \mathbb{E}_\phi[err(t)|\Psi(1)] = 0$,
ii) $T(t)$ satisfies $\limsup_{M\to\infty} \frac{1}{M}\sum_{t=0}^{M-1} \mathbb{E}_\phi[T(t)|\Psi(1)] \le \frac{3}{2} + \frac{1}{\epsilon_s}$.

Remarks on Expected Average Bounds.
• Theorem V.1 implies that in presence of switching cost, the queue length scales as $1/(\text{optimality gap})^2$ (set $V{=}1/\epsilon_s^2$). Whereas, when switching cost is absent queue length scales as $1/(\text{optimality gap})$ under max-weight algorithm [21].
• The one-step bound in equation (4) depends on two time-varying processes, in addition to the sum of queue lengths $\|\mathbf{Q}(t)\|_1$. It depends on, i) the empirical estimate, $\hat{\boldsymbol{\mu}}(t)$, and ii) the time since the estimated optimal BS config. equals the constrained optimal, $T(t)$.
• The Lemma V.2 provides multiple bounds on expected drift-plus-penalty (4), each parameterized with $\epsilon_g'{\in}(0,\epsilon_g)$. We use different such bounds for bounding expected average of cost and sum-of-queue lengths in Theorem V.1.

Our second key result is the non-asymptotic upper-tail bounds on the sum-of-queue lengths, $\|\mathbf{Q}(t)\|_1$. Formally,

Theorem V.3. *For system $(\boldsymbol{\lambda},\boldsymbol{\mu})$ with capacity gap $\epsilon_g > 0$, under algorithm $\phi \in \{$LASS-Static, LASS-Dynamic$\}$ with parameters $\epsilon_s \in (0,1]$ and $V > 0$, for all $r \in (0, r_{\max,\phi})$, and for all $t \ge 1$, $x > 0$,*
$$\mathbb{P}_\phi[\|\mathbf{Q}(t)\|_1 {\ge} B_{max}N_Q {+} \theta_\phi {+} x|\Psi(1)] {\le} \frac{e^{-rx}}{r\epsilon_g}\left(1{+}O\left(\frac{\log(t)}{t}\right)\right)$$
In particular the following statements hold.
1) For both ϕ, $\theta_\phi{=}\Theta\left(\frac{2Vc_1N_m + B_{\max}^2 N_Q}{\epsilon_g} {+} \frac{B_{\max}^2 N_Q}{\epsilon_g\epsilon_s}\right)$.
2) For $\phi =$ LASS-Static, $r_{max,\phi} = \Theta\left(\frac{\epsilon_s\epsilon_g}{(B_{\max}N_Q)^2}\right)$.
3) For $\phi =$ LASS-Dynamic, $r_{max,\phi} = \Theta\left(\frac{\epsilon_g}{(B_{\max}N_Q)^2}\right)$.

The next lemma bounds the MGF of the sum of queue lengths as a function of time and leads Theorem V.3.

Lemma V.4. *For system $(\boldsymbol{\lambda},\boldsymbol{\mu})$ with capacity gap $\epsilon_g > 0$, for algorithm $\phi \in \{$LASS-Static, LASS-Dynamic$\}$, for any $r \in (0, r_{\max,\phi})$, and for all $t \ge 1$,*
$$\mathbb{E}_\phi\left[\exp(r\|\mathbf{Q}(t{+}1)\|_1)|\Psi(1)\right] {<} O\left(\frac{e^{(r(B_{max}N_Q + \theta_\phi))}}{r\epsilon_g} {+} \frac{\log(t)}{t}\right)$$
where, parameters θ_ϕ, and $r_{max,\phi}$ are as given in Theorem V.3.

Remarks on Non-asymptotic Bounds.
• Lemma V.4 provides drift-analysis results including learning and switching events. It extends the results in [16], [17].
• Theorem V.3 shows LASS-Dynamic has a larger decay rate, hence better performance, than its static counterpart. This provides evidence that dynamic utilization of the switching resources based on the drift regret is more efficient.

VI. Simulation Results

A. Simulation Setup

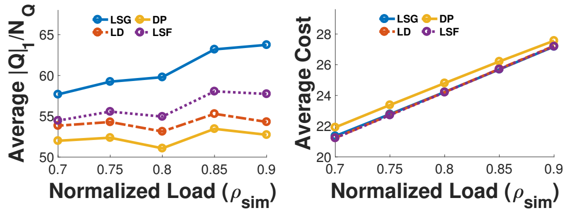
In this section we validate our results using simulations. We consider a system with $N_m = 3$ BSs and $N_u = 8$ users. The connectivity graph $\mathcal{G}$ is: (i) BS 1 is connected to all users except 2, (ii) BS 2 is connected to $\{2,5,6,7,8\}$, and (iii) BS 3 is connected to users $\{5,6,7,8\}$. All BS configurations are allowed $\mathcal{J} = 2^{[3]}$. There are three possible channel states with correlated links. Under each of the channel state any link (m,u) is either good (serves 10 packets) or bad (serves 0 packet), and is represented jointly by a $N_m \times N_u$ binary matrix. The channel process $H(t)$ is i.i.d. w.p. $\boldsymbol{\mu} = (0.4, 0.3, 0.3)$. In each time slot, a BS, if active, can serve at most one user, and an user can receive from only one BS. Let $\boldsymbol{\lambda}^*$ lie on the boundary of the capacity region. For a *normalized load* ρ_{sim} the arrival rate vector is $\boldsymbol{\lambda} = \rho_{sim}\boldsymbol{\lambda}^*$ and the associated capacity gap $\epsilon_g = (\min\{\lambda_{m,u}^* : \lambda_{m,u}^* > 0\})(1 - \rho_{sim})$. For each queue $Q_{m,u}$ the arrival process, $A_{m,u}(t)$, is a Poisson random variable with mean $\lambda_{m,u}$ distributed i.i.d., coordinate wise and for all $t \ge 1$. The network cost constants are $c_0 = 1$ (switching) and $c_1 = 10$ (operational). All the simulations are run for 5×10^5 iterations. We observed that at most 1×10^5 iterations, which includes at least 500 switching events, were sufficient for convergence in all the reported cases.

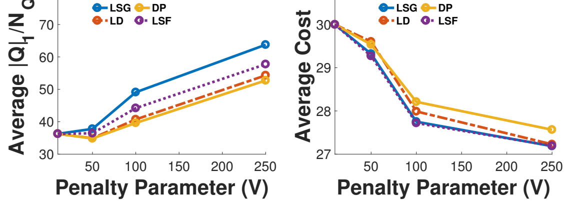
B. Performance of LASS-Static and Dynamic

We study the performance of the following algorithms:
1) **LSG**: LASS-Static, 2) **LD**: LASS-Dynamic,
3) **LSF**: LASS-Static with fixed period length $\lfloor 1/\epsilon_s \rfloor$,
4) **DP**: Drift-plus-Penalty, unconstrained and with known $\boldsymbol{\mu}$.

The effect of normalized load (ρ_{sim}), penalty parameter (V) and the switching rate (ϵ_s) on the following two metrics are studied, 1) queue length $\frac{\mathcal{Q}(\boldsymbol{\lambda},\boldsymbol{\mu})}{N_Q}$, and, 2) aggregate cost, $\mathcal{C}(\boldsymbol{\lambda},\boldsymbol{\mu})$. Smaller values of the metrics indicate better performance.

We first focus on the effect of the relevant parameters on the performance (Fig. 2). We reach the conclusion that at high


(a) Effect of varying ρ_{sim}; $V = 250$, $\epsilon_s = 0.1$


(b) Effect of varying V; $\rho_{sim} = 0.9$, $\epsilon_s = 0.1$

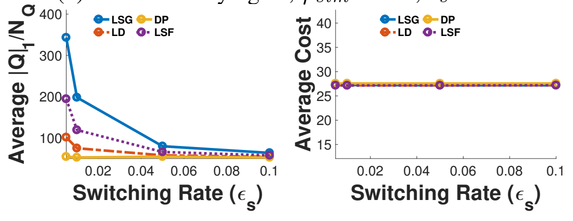

(c) Effect of varying ϵ_s; $\rho_{sim} = 0.9$, $V = 250$

Fig. 2: Influence of parameters on the average aggregate cost and the average queue length.

ρ_{sim} or large V the theoretical results provide qualitatively correct results. Further, the results for ϵ_s seem to follow the theoretically predicted trend, for a wide range of values. We now shift our focus on the performance of various algorithms. Throughout all parameters the performance, in term of average queue length, follows the trend LSG<LSF<LD<DP. At *high loads*, *large* V or *small* ϵ_s, we observe that the average queue length under LD is smaller compared to LSG and LSF. For large V, the average costs under LSF, LSG, and LD have negligible difference; however, due to unconstrained switching DP has higher cost.

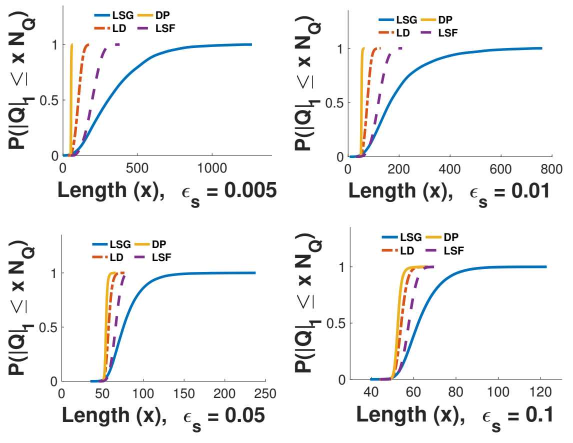


Fig. 3: Effect of switching rate on the cdf of queue length.

In Fig. 3 we plot the cumulative distribution function (cdf) of the queue lengths for different ϵ_s, where $V = 250$ and $\rho_{sim} = 0.9$. The figure clearly depicts that for a large range of ϵ_s the decay rate follow the order DP>LD>LSF>LSG. Specifically, LSG has slower rate of decay than LD, thus corroborating Theorem V.3. The difference in decay rate for various algorithms becomes more pronounced at smaller ϵ_s. LD remains close to DP in terms of decay rate throughout the range $[0.005, 0.1]$. We observe the same behavior for moderate to high ρ_{sim} ($\approx [0.75, 0.95]$) and large V (100 and above).

VII. Related Work

The energy efficient operation of communication networks (e.g. cellular networks) that meets user expectations is an important topic of research ([6], [8], [22], [23]), and various strategies to tackle this problem have been proposed [5], [12], [24]–[26]. One promising approach, is the application of opportunistic scheduling in BS activation. Under time varying channels, using this framework, joint stability and energy efficiency of the networks, has been established through greedy Lyapunov function optimization—drift-plus-penalty method [10], [18], [27], and greedy primal dual method [9], [28]. While these works require instantaneous information of channel states, in many situations (including ours) such information may only be revealed through an action. In these scenarios, two-stage decisions are necessary, where first stage depends on channel statistics [29], [30]. Authors in [29] assume knowledge of channel statistics to provide throughput optimality with two-stage decisions. Further, in [30] similar guarantees are shown when statistics is learned through exploration at a *constant rate*. Notably, these works use queue length based decisions in both the stages. In [14] a *joint* learning and resource allocation algorithm is developed using *decaying rate* of exploration. However, in [14] the first stage decision—BS activation/de-activation, is made queue independent. Therefore, our work introduces *joint* learning and two-staged resource allocation algorithms; where 1) exploration rate decays with time, and 2) in both stages, resource allocation is queue length dependent.

Furthermore, as the authors in [14] point out, these greedy strategies (including the two-stage algorithms) are inadequate for minimizing operational plus *switching cost*. When complexity is not an issue, this problem can be formulated as a constrained Markov decision process (CMDP) and solved optimally [31], [32]. The authors in [14] first proposed a low complexity near-optimal strategy using constrained switching. However, in that work the BS activation and switching does not adapt to the instantaneous load. To mitigate this shortcoming, in LASS-Static, we make the BS activation decisions queue dependent. Further, in LASS-Dynamic, we make both BS activation and switching queue dependent.

The concept of slowing down decisions appear in different contexts, e.g. queues with setup times [33], [34], which do not explicitly constrain the switching cost. Finally, our non-asymptotic analysis builds on the previous works [16], [35].

VIII. Conclusion and Discussions

In this paper we provided two BS switching and scheduling algorithms for joint stability, and cost-optimality guarantees, when BS switching cost is included in addition to BS operational cost. Proposed algorithm LASS-Static makes BS activation system load dependent using drift-plus-penalty based approach [18]. LASS-Dynamic improves further; it makes both 'when to switch' and 'what to switch to' system load dependent. Finally, in both the algorithms the switching decisions are integrated with max-weight based scheduling and explore-exploit based learning. We show, when the capacity gap is ϵ_g, both algorithms attain, an expected average queue length of $O(1/\epsilon_s^2\epsilon_g)$ and an additive $O(\epsilon_s)$ sub-optimality to the expected cost. Furthermore, we show that the sum-of-queue lengths under LASS-Static decays exponentially with rate $\Theta(\epsilon_g\epsilon_s)$, whereas under LASS-Dynamic it decays at a rate $\Theta(\epsilon_g)$.

Acknowledgements

This work was partially supported by NSF grant CNS-1731658, ARO grant W911NF-17-1-0359, the Wireless Networking and Communications Group Industrial Affiliates Program, and the the US DoT supported D-STOP Tier 1 University Transportation Center.

References

[1] B. Bangerter, S. Talwar, R. Arefi, and K. Stewart, "Networks and devices for the 5g era," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 90–96, 2014.

[2] A. Ali, W. Hamouda, and M. Uysal, "Next generation m2m cellular networks: challenges and practical considerations," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 18–24, 2015.

[3] N. Bhushan, J. Li, D. Malladi, R. Gilmore, D. Brenner, A. Damnjanovic, R. Sukhavasi, C. Patel, and S. Geirhofer, "Network densification: the dominant theme for wireless evolution into 5g," *IEEE Communications Magazine*, vol. 52, no. 2, pp. 82–89, 2014.

[4] X. Ge, S. Tu, G. Mao, C.-X. Wang, and T. Han, "5g ultra-dense cellular networks," *IEEE Wireless Communications*, vol. 23, no. 1, pp. 72–79, 2016.

[5] S. Buzzi, I. Chih-Lin, T. E. Klein, H. V. Poor, C. Yang, and A. Zappone, "A survey of energy-efficient techniques for 5g networks and challenges ahead," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 4, pp. 697–709, 2016.

[6] X. Ge, J. Yang, H. Gharavi, and Y. Sun, "Energy efficiency challenges of 5g small cell networks," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 184–191, 2017.

[7] M. A. Marsan, L. Chiaraviglio, D. Ciullo, and M. Meo, "Optimal energy savings in cellular access networks," in *Communications Workshops, 2009. ICC Workshops 2009. IEEE International Conference on*. IEEE, 2009, pp. 1–5.

[8] A. Fehske, G. Fettweis, J. Malmodin, and G. Biczok, "The global footprint of mobile communications: The ecological and economic perspective," *IEEE Communications Magazine*, vol. 49, no. 8, 2011.

[9] A. L. Stolyar, "Maximizing queueing network utility subject to stability: Greedy primal-dual algorithm," *Queueing Systems*, vol. 50, no. 4, pp. 401–457, 2005.

[10] M. J. Neely, "Optimal energy and delay tradeoffs for multiuser wireless downlinks," *IEEE Transactions on Information Theory*, vol. 53, no. 9, pp. 3095–3113, 2007.

[11] E. Oh, B. Krishnamachari, X. Liu, and Z. Niu, "Toward dynamic energy-efficient operation of cellular network infrastructure," *IEEE Communications Magazine*, vol. 49, no. 6, 2011.

[12] J. Gong, S. Zhou, and Z. Niu, "A dynamic programming approach for base station sleeping in cellular networks," *IEICE transactions on communications*, vol. 95, no. 2, pp. 551–562, 2012.

[13] N. Yu, Y. Miao, L. Mu, H. Du, H. Huang, and X. Jia, "Minimizing energy cost by dynamic switching on/off base stations in cellular networks," *IEEE Transactions on Wireless Communications*, vol. 15, no. 11, pp. 7457–7469, 2016.

[14] S. Krishnasamy, P. Akhil, A. Arapostathis, S. Shakkottai, and R. Sundaresan, "Augmenting max-weight with explicit learning for wireless scheduling with switching costs," in *INFOCOM 2017-IEEE Conference on Computer Communications, IEEE*. IEEE, 2017, pp. 1–9.

[15] A. Abbasi and M. Ghaderi, "Distributed base station activation for energy-efficient operation of cellular networks," in *Proceedings of the 16th ACM international conference on Modeling, analysis & simulation of wireless and mobile systems*. ACM, 2013, pp. 427–436.

[16] B. Hajek, "Hitting-time and occupation-time bounds implied by drift analysis with applications," *Advances in Applied probability*, vol. 14, no. 3, pp. 502–525, 1982.

[17] H. Yu, M. Neely, and X. Wei, "Online convex optimization with stochastic constraints," in *Advances in Neural Information Processing Systems*, 2017, pp. 1427–1437.

[18] M. J. Neely, E. Modiano, and C. E. Rohrs, "Dynamic power allocation and routing for time-varying wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 1, pp. 89–103, 2005.

[19] L. Tassiulas and A. Ephremides, "Stability properties of constrained queueing systems and scheduling policies for maximum throughput in multihop radio networks," *IEEE transactions on automatic control*, vol. 37, no. 12, pp. 1936–1948, 1992.

[20] S. Basu and S. Shakkottai, "Switching Constrained Max-Weight Scheduling for Wireless Networks," The University of Texas at Austin, ECE, Tech. Rep., 2018.

[21] L. Huang, S. Moeller, M. J. Neely, and B. Krishnamachari, "Lifo-backpressure achieves near-optimal utility-delay tradeoff," *IEEE/ACM Transactions On Networking*, vol. 21, no. 3, pp. 831–844, 2013.

[22] A. J. Fehske, F. Richter, and G. P. Fettweis, "Energy efficiency improvements through micro sites in cellular mobile radio networks," in *GLOBECOM Workshops, 2009 IEEE*. IEEE, 2009, pp. 1–5.

[23] J. Wu, Y. Zhang, M. Zukerman, and E. K.-N. Yung, "Energy-efficient base-stations sleep-mode techniques in green cellular networks: A survey," *IEEE communications surveys & tutorials*, vol. 17, no. 2, pp. 803–826, 2015.

[24] Z. Niu, Y. Wu, J. Gong, and Z. Yang, "Cell zooming for cost-efficient green cellular networks," *IEEE communications magazine*, vol. 48, no. 11, 2010.

[25] Z. Niu, "Tango: Traffic-aware network planning and green operation," *IEEE Wireless Communications*, vol. 18, no. 5, 2011.

[26] E. Oh, K. Son, and B. Krishnamachari, "Dynamic base station switching-on/off strategies for green cellular networks," *IEEE transactions on wireless communications*, vol. 12, no. 5, pp. 2126–2136, 2013.

[27] J. Liu, A. Eryilmaz, N. B. Shroff, and E. S. Bentley, "Heavy-ball: A new approach to tame delay and convergence in wireless network optimization," in *INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE*. IEEE, 2016, pp. 1–9.

[28] A. Eryilmaz and R. Srikant, "Fair resource allocation in wireless networks using queue-length-based scheduling and congestion control," *IEEE/ACM Transactions on Networking (TON)*, vol. 15, no. 6, pp. 1333–1344, 2007.

[29] A. Gopalan, C. Caramanis, and S. Shakkottai, "On wireless scheduling with partial channel-state information," *IEEE Transactions on Information Theory*, vol. 58, no. 1, pp. 403–420, 2012.

[30] M. J. Neely, S. T. Rager, and T. F. La Porta, "Max weight learning algorithms for scheduling in unknown environments," *IEEE Transactions on Automatic Control*, vol. 57, no. 5, pp. 1179–1191, 2012.

[31] E. Altman, *Constrained Markov decision processes*. CRC Press, 1999, vol. 7.

[32] M. J. Neely, "Dynamic optimization and learning for renewal systems," *IEEE Transactions on Automatic Control*, vol. 58, no. 1, pp. 32–46, 2013.

[33] C. W. Chan, M. Armony, and N. Bambos, "Maximum weight matching with hysteresis in overloaded queues with setups," *Queueing Systems*, vol. 82, no. 3-4, pp. 315–351, 2016.

[34] G. Celik, S. C. Borst, P. A. Whiting, and E. Modiano, "Dynamic scheduling with reconfiguration delays," *Queueing Systems*, vol. 83, no. 1-2, pp. 87–129, 2016.

[35] J. He and X. Yao, "Drift analysis and average time complexity of evolutionary algorithms," *Artificial intelligence*, vol. 127, no. 1, pp. 57–85, 2001.