Distributed Stochastic Optimization via Correlated Scheduling

Michael J. Neely
University of Southern California
http://www-bcf.usc.edu/~mjneely

***Abstract*—This paper considers a problem where multiple users make repeated decisions based on their own observed events. The events and decisions at each time step determine the values of a utility function and a collection of penalty functions. The goal is to make distributed decisions over time to maximize time average utility subject to time average constraints on the penalties. An example is a collection of power constrained sensors that repeatedly report their own observations to a fusion center. Maximum time average utility is fundamentally reduced because users do not know the events observed by others. Optimality is characterized for this distributed context. It is shown that optimality is achieved by correlating user decisions through a commonly known pseudorandom sequence. An optimal algorithm is developed that chooses pure strategies at each time step based on a set of time-varying weights.**

I. Introduction

Consider a multi-user system that operates over discrete time with unit time slots $t \in \{0, 1, 2, \ldots\}$. There are N users. At each time slot t, each user i observes a *random event* $\omega_i(t)$ and makes a *control action* $\alpha_i(t)$ based on this observation. Let $\boldsymbol{\omega}(t)$ and $\boldsymbol{\alpha}(t)$ be vectors of these values:

$$\begin{aligned}\boldsymbol{\omega}(t) &= (\omega_1(t), \omega_2(t), \ldots, \omega_N(t)) \\ \boldsymbol{\alpha}(t) &= (\alpha_1(t), \alpha_2(t), \ldots, \alpha_N(t))\end{aligned}$$

For each slot t, these vectors determine the values of a *system utility* $u(t)$ and a collection of *system penalties* $p_1(t), \ldots, p_K(t)$ (for some non-negative integer K) via real-valued functions:

$$\begin{aligned}u(t) &= \hat{u}(\boldsymbol{\alpha}(t), \boldsymbol{\omega}(t)) \\ p_k(t) &= \hat{p}_k(\boldsymbol{\alpha}(t), \boldsymbol{\omega}(t)) \ \forall k \in \{1, \ldots, K\}\end{aligned}$$

The functions $\hat{u}(\cdot)$ and $\hat{p}_k(\cdot)$ are arbitrary and can possibly be negative. Negative penalties can be used to represent desirable *system rewards*.

The goal is to make distributed decisions over time that maximize time average utility subject to time average constraints on the penalties. Central to this problem is the assumption that each user i can only observe $\omega_i(t)$, and cannot observe the value of $\omega_j(t)$ for other users $j \neq i$. Further, each user i only knows its own action $\alpha_i(t)$, but does not know the actions $\alpha_j(t)$ of others. Therefore, each user only knows a portion of the arguments that go into the functions $\hat{u}(\boldsymbol{\alpha}(t), \boldsymbol{\omega}(t))$ and $\hat{p}_k(\boldsymbol{\alpha}(t), \boldsymbol{\omega}(t))$ for each slot t. This uncertainty fundamentally restricts the time averages that can be achieved.

Specifically, assume the random event vector $\boldsymbol{\omega}(t)$ is independent and identically distributed (i.i.d.) over slots (possibly correlated over entries in each slot). The vector $\boldsymbol{\omega}(t)$ takes values in some abstract *event space* $\Omega = \Omega_1 \times \Omega_2 \times \cdots \times \Omega_N$, where $\omega_i(t) \in \Omega_i$ for all $i \in \{1, \ldots, N\}$ and all slots t. Similarly, assume $\boldsymbol{\alpha}(t)$ is chosen in some abstract *action space* $\mathcal{A} = \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \times \mathcal{A}_N$, where $\alpha_i(t) \in \mathcal{A}_i$ for all $i \in \{1, \ldots, N\}$ and all slots t. Let $\overline{u}$ and $\overline{p}_k$ be the time average expected utility and penalty incurred by a particular algorithm:

$$\begin{aligned}\overline{u} &= \lim_{t\rightarrow\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\left[u(\tau)\right] \\ \overline{p}_k &= \lim_{t\rightarrow\infty} \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\left[p_k(\tau)\right]\end{aligned}$$

The following problem is considered:

$$\begin{aligned}&\text{Maximize:} && \overline{u} && (1)\\ &\text{Subject to:} && \overline{p}_k \leq c_k \ \ \forall k \in \{1, \ldots, K\} && (2)\\ & && \text{Decisions are distributed} && (3)\end{aligned}$$

where c_k are a given collection of real numbers that specify constraints on the time average penalties.

The constraint that decisions must be distributed, specified in (3), is not mathematically precise. This constraint is more carefully posed in Section III. Without the distributed scheduling constraint, the problem (1)-(2) reduces to a standard problem of stochastic network optimization and can be solved via the *drift-plus-penalty method* [1]. Such a centralized approach would allow users to coordinate to form an action vector $\boldsymbol{\alpha}(t)$ based on full knowledge of the event vector $\boldsymbol{\omega}(t)$. The time average utility achieved by the best centralized algorithm can be strictly larger than that of the best distributed algorithm. An example of this gap is given in Section II.

A. Applications to sensor networks

The above formulation is useful for problems where distributed agents make their own decisions based on partial system knowledge. An important example is a network of wireless sensors that repeatedly send reports about system events to a fusion center. The goal is to make distributed decisions that maximize time average *quality of information*. This scenario was previously considered by Liu et al. in [2]. There, sensors can provide reports every slot t using one

This work is supported in part by one or more of: the NSF Career grant CCF-0747525, NSF grant 1049541, the Network Science Collaborative Technology Alliance sponsored by the U.S. Army Research Laboratory W911NF-09-2-0053.

of multiple *reporting formats*, such as text, image, or video. Sensors can also choose to remain idle. Thus, the action spaces $\mathcal{A}_i$ are the same for all sensors i:

$$\alpha_i(t) \in \mathcal{A}_i \triangleq \{\text{idle}, \text{text}, \text{image}, \text{video}\} \quad \forall i \in \{1, \ldots, N\}$$

where the notation "$\triangleq$" represents *defined to be equal to*.

Each format requires a different amount of power and provides a different level of quality. Define $p_i(t) = \hat{p}_i(\alpha_i(t))$ as the power incurred by sensor i on slot t, where:

$$0 = \hat{p}_i(\text{idle}) < \hat{p}_i(\text{text}) < \hat{p}_i(\text{image}) < \hat{p}_i(\text{video})$$

Assume that $\omega_i(t)$ represents the *quality* that sensor i would bring to the fusion center if it reports the event it observes on slot t using the video format. Define $f_i(\alpha_i(t))$ as the fraction of this quality that is achieved under format $\alpha_i(t)$, where:

$$0 = f_i(\text{idle}) < f_i(\text{text}) < f_i(\text{image}) < f_i(\text{video}) = 1$$

The prior work [2] considers the problem of maximizing time average utility subject to a time average power constraint:

$$\sum_{i=1}^N \overline{p}_i \leq c$$

where c is some given positive number. Further, that work restricts to the special case when the utility function is a *separable sum* of functions of user i variables, such as:

$$u(t) = \sum_{i=1}^N f_i(\alpha_i(t))\omega_i(t)$$

Such separable utilities cannot model the realistic scenario of *information saturation*, where, once a certain amount of utility is achieved on slot t, there is little value of having additional sensors spend power to deliver additional information on that slot. The current paper considers the case of *arbitrary, possibly non-separable* utility functions. An example is:

$$u(t) = \min\left[\sum_{i=1}^N f(\alpha_i(t))\omega_i(t), 1\right]$$

This means that once a total quality of 1 is accumulated from one or more sensors on slot t, there is no advantage in having other sensors report information on that slot. This scenario is significantly more challenging to solve in a distributed context.

B. Applications to wireless multiple access

The general formulation of this paper can also treat simple forms of distributed multiple access problems. Again suppose there are N wireless sensors that report to a fusion center. For each $i \in \{1, \ldots, N\}$, define $\omega_i(t)$ as the *quality* that a transmission from sensor i would bring to the system if it transmits on slot t. Define $\alpha_i(t)$ as a binary value that is 1 if sensor i transmits on slot t, and 0 else. Assume the network operates according to a simple collision model, where a transmission from sensor i is successful on slot t if and only if it is the only sensor that transmits on that slot:

$$u(t) = \sum_{i=1}^N \omega_i(t)\left[\alpha_i(t)\prod_{j\neq i}(1-\alpha_j(t))\right] \tag{4}$$

The above utility function is non-separable. Concurrent work in [3] considers a similar utility function for wireless energy harvesting applications.

C. Contributions and related work

The framework of partial knowledge at each user is similar in spirit to a *multi-player Bayesian game* [4][5]. There, the goal is to design competitive strategies that lead to a Nash equilibrium. The current paper is not concerned with competition or equilibrium. Rather, it seeks distributed strategies for maximizing the time average of a single utility function subject to additional time average penalty constraints.

This paper shows that an optimal distributed algorithm can be designed by having users correlate their decisions through an independent source of common randomness (Section III). Related notions of commonly shared randomness are used in game theory to define a *correlated equilibrium*, which is typically easier to compute than a standard Nash equilibrium [6][7][5][4]. For the current paper, the shared randomness is crucial for solving the distributed optimization problem. This paper shows that optimality can be achieved by using a shared random variable with $K + 1$ possible outcomes, where K is the number of penalty constraints. The solution is computable through a linear program. Unfortunately, the linear program can have a very large number of variables, even for 2-user problems. A reduction to polynomial complexity is shown to be possible in certain cases (Section IV). This paper also develops an online algorithm that chooses pure strategies every slot based on a set of weights that are updated at the end of each slot (Section V). The online technique is based on Lyapunov optimization concepts [1][8][9].

Much prior work on network optimization treats scenarios where it is possible to find distributed solutions with no loss of optimality. For example, network flow problems that are described by linear or separable convex programs can be optimally solved in a distributed manner [10][11][12][9]. Work in [13] uses distributed agents to solve for an optimal vector of parameters associated with a Markov decision problem. Work in [14][15][16] develops distributed multiple access methods that converge to optimality. However, the above problems do not have random events that create a fundamental gap between centralized and distributed performance.

Recent work in [17] derives structural results for distributed optimization in Markov decision systems with delayed information. Such problems *do* exhibit gaps between centralized and distributed scheduling. The use of *private information* in [17] is similar in spirit to the assumption in the current paper that each user observes its own random event $\omega_i(t)$. The work [17] derives a sufficient statistic for dynamic programming. It does not consider time average constraints and its solutions do not involve correlated scheduling via a pseudorandom sequence. Recent work in [3] considers distributed reporting of events with different qualities, but considers a more restrictive class of policies that do not use correlated scheduling. The current paper treats a different model than [17] and [3], and shows that correlated scheduling is necessary in systems with constraints. The current paper also provides complexity reduction results (Section IV), and provides an online algorithm that does not require knowledge of event probabilities (Section V).

II. Example sensor network problem

This section illustrates the benefits of using a common source of randomness for a simple example. Consider a network with two sensors that operate over time slots $t \in \{0, 1, 2, \ldots\}$. Every slot, the sensors observe the state of a particular system and choose whether or not to report their observations to a fusion center. Let $\omega_i(t)$ be a binary variable that is 1 if sensor i observes an event on slot t, and 0 else. Let $\alpha_1(t)$ and $\alpha_2(t)$ be the slot t decision variables, so that $\alpha_i(t) = 1$ if sensor i reports on slot t, and $\alpha_i(t) = 0$ otherwise. Suppose the fusion center trusts sensor 1 more than sensor 2. Consider the following example utility function:

$$u(t) = \min[\omega_1(t)\alpha_1(t) + \omega_2(t)\alpha_2(t)/2, 1]$$

so that $\hat{u}(\cdot)$ is given by:

$$\hat{u}(\alpha_1, \alpha_2, \omega_1, \omega_2) = \min[\omega_1\alpha_1 + \omega_2\alpha_2/2, 1] \qquad (5)$$

Therefore, $u(t) \in \{0, 1/2, 1\}$ for all slots t. If $\omega_1(t) = 1$ and sensor 1 reports on slot t, there is no utility increase if sensor 2 also reports.

Each report uses one unit of power. Let $p_i(t)$ be the power incurred by sensor i on slot t, being 1 if it reports its observation, and 0 otherwise. The power penalties for $i \in \{1, 2\}$ are:

$$p_i(t) = \alpha_i(t) \qquad (6)$$

so that $\hat{p}_i(\alpha_1, \alpha_2, \omega_1, \omega_2) = \alpha_i$ for $i \in \{1, 2\}$. Each sensor i can choose *not* to report an observation in order to save power. The difficulty is that neither sensor knows what event was observed by the other. Therefore, a distributed algorithm might send reports from *both* sensors on a given slot. A centralized scheduler would avoid this because it wastes power without increasing utility.

Suppose that $\omega_1(t)$ and $\omega_2(t)$ are independent of each other and i.i.d. over slots, with:

$$Pr[\omega_1(t) = 1] = 3/4, \quad Pr[\omega_1(t) = 0] = 1/4$$
$$Pr[\omega_2(t) = 1] = 1/2, \quad Pr[\omega_2(t) = 0] = 1/2$$

For a specific numeric example, consider the problem:

$$\begin{aligned} \text{Maximize:} \quad & \overline{u} & (7)\\ \text{Subject to:} \quad & \overline{p}_1 \leq 1/3 \ , \ \overline{p}_2 \leq 1/3 & (8)\\ & \text{Decisions are distributed} & (9) \end{aligned}$$

A. Independent reporting

Consider the following class of *independent scheduling* algorithms: Each sensor i independently decides to report with probability θ_i if it observes $\omega_i(t) = 1$ (it does not report if $\omega_i(t) = 0$). Since $\boldsymbol{\omega}(t)$ is i.i.d. over slots, the resulting sequences $\{u(t)\}_{t=0}^{\infty}$, $\{p_1(t)\}_{t=0}^{\infty}$, $\{p_2(t)\}_{t=0}^{\infty}$ are i.i.d. over slots. The time averages are:

$$\overline{p}_1 = \tfrac{3}{4}\theta_1 \quad , \quad \overline{p}_2 = \tfrac{1}{2}\theta_2$$

$$\begin{aligned} \overline{u} \;=\; & \mathbb{E}\left[u(t)|\omega_1(t) = 1, \omega_2(t) = 0\right] \tfrac{3}{4}\tfrac{1}{2} \\ & + \mathbb{E}\left[u(t)|\omega_1(t) = 0, \omega_2(t) = 1\right] \tfrac{1}{4}\tfrac{1}{2} \\ & + \mathbb{E}\left[u(t)|\omega_1(t) = \omega_2(t) = 1\right] \tfrac{3}{4}\tfrac{1}{2} \\ =\; & \tfrac{3}{4}\tfrac{1}{2}\theta_1 + \tfrac{1}{4}\tfrac{1}{2}(\theta_2/2) + \tfrac{3}{4}\tfrac{1}{2}(\theta_1 + (1-\theta_1)\theta_2/2) \end{aligned}$$

For this class of algorithms, utility is maximized by choosing θ_1 and θ_2 to meet the power constraints with equality. This leads to $\theta_1 = 4/9$, $\theta_2 = 2/3$. The resulting utility is:

$$\overline{u} = 4/9 \approx 0.44444$$

B. Correlated reporting

As an alternative, consider the following three strategies:

- Strategy 1: $\omega_1(t) = 1 \implies \alpha_1(t) = 1$ (else, $\alpha_1(t) = 0$). Sensor 2 always chooses $\alpha_2(t) = 0$.
- Strategy 2: $\omega_2(t) = 1 \implies \alpha_2(t) = 1$ (else, $\alpha_2(t) = 0$). Sensor 1 always chooses $\alpha_1(t) = 0$.
- Strategy 3: $\omega_1(t) = 1 \implies \alpha_1(t) = 1$ (else, $\alpha_1(t) = 0$). $\omega_2(t) = 1 \implies \alpha_2(t) = 1$ (else, $\alpha_2(t) = 0$).

The above three strategies are *pure strategies* because $\alpha_i(t)$ is a deterministic function of $\omega_i(t)$ for each sensor i. Now let $X(t)$ be an external source of randomness that is commonly known at both sensors on slot t. Assume $X(t)$ is independent of everything else in the system, and is i.i.d. over slots with:

$$Pr[X(t) = m] = \theta_m \quad , \ \forall m \in \{1, 2, 3\}$$

where $\theta_1, \theta_2, \theta_3$ are probabilities that sum to 1. Consider the following algorithm: On slot t, if $X(t) = m$ then choose strategy m, where $m \in \{1, 2, 3\}$. This algorithm can be implemented by letting $X(t)$ be a pseudorandom sequence that is installed in both sensors at time 0. The resulting time averages are:

$$\overline{p}_1 = (\theta_1 + \theta_3)\tfrac{3}{4} \ , \ \overline{p}_2 = (\theta_2 + \theta_3)\tfrac{1}{2}$$
$$\overline{u} = \theta_1\tfrac{3}{4} + \theta_2\tfrac{1}{2}\tfrac{1}{2} + \theta_3(\tfrac{3}{4} + \tfrac{1}{4}\tfrac{1}{2}\tfrac{1}{2})$$

A simple linear program can be used to compute the optimal θ_m probabilities for this algorithm structure. The result is $\theta_1 = 1/3$, $\theta_2 = 5/9$, $\theta_3 = 1/9$. The resulting utility is:

$$\overline{u} = 23/48 \approx 0.47917$$

This is strictly larger than the time average utility of 0.44444 achieved by the independent reporting algorithm. Thus, performance can be strictly improved by correlating reports via a common source of randomness. Alternatively, the same time averages can be achieved by *time sharing*: The two sensors agree to use a periodic schedule of period 9 slots. The first 3 slots of the period use strategy 1, the next 5 slots use strategy 2, and the final slot uses strategy 3.

C. Centralized reporting

Suppose sensors coordinate by observing $(\omega_1(t), \omega_2(t))$ and then cooperatively selecting $(\alpha_1(t), \alpha_2(t))$. It turns out that an optimal centralized policy is as follows [1]: Every slot t, observe $(\omega_1(t), \omega_2(t))$ and choose $(\alpha_1(t), \alpha_2(t))$ as follows:

- $(\omega_1(t), \omega_2(t)) = (0, 0) \implies (\alpha_1(t), \alpha_2(t)) = (0, 0)$.
- $(\omega_1(t), \omega_2(t)) = (0, 1) \implies (\alpha_1(t), \alpha_2(t)) = (0, 1)$.
- If $(\omega_1(t), \omega_2(t)) = (1, 0)$, independently choose:

$$(\alpha_1(t), \alpha_2(t)) = \begin{cases} (1, 0) & \text{with probability } 8/9 \\ (0, 0) & \text{with probability } 1/9 \end{cases}$$

- If $(\omega_1(t), \omega_2(t)) = (1, 1)$, independently choose:

$$(\alpha_1(t), \alpha_2(t)) = \begin{cases} (0,1) & \text{with probability } 5/9 \\ (0,0) & \text{with probability } 4/9 \end{cases}$$

The resulting optimal centralized time average utility is:

$$\overline{u} = 0.5$$

This is larger than the value 0.47917 achieved by the distributed algorithm of the previous subsection.

The question remains: Is it possible to construct some other distributed algorithm that yields $\overline{u} > 0.47917$? Results in the next section imply this is impossible. Thus, the correlated reporting algorithm of the previous subsection optimizes time average utility over all possible distributed algorithms that satisfy the constraints. Therefore, for this example, there is a *fundamental gap* between the performance of the best centralized algorithm and the best distributed algorithm.

III. Characterizing optimality

Consider the general N user problem. Recall that:

$$\boldsymbol{\omega}(t) \in \Omega \triangleq \Omega_1 \times \cdots \times \Omega_N \ \ , \ \ \boldsymbol{\alpha}(t) \in \mathcal{A} \triangleq \mathcal{A}_1 \times \cdots \times \mathcal{A}_N$$

where the vectors $\boldsymbol{\omega}(t)$ are i.i.d. over slots (possibly correlated over entries in each slot). Assume that the sets Ω_i and $\mathcal{A}_i$ are finite with sizes denoted $|\Omega_i|$ and $|\mathcal{A}_i|$. For each $\boldsymbol{\omega} \in \Omega$ define:

$$\pi(\boldsymbol{\omega}) = Pr[\boldsymbol{\omega}(t) = \boldsymbol{\omega}]$$

Define the *history* $\mathcal{H}(t)$ by:

$$\mathcal{H}(t) \triangleq \{(\boldsymbol{\omega}(0), \boldsymbol{\alpha}(0)), \ldots, (\boldsymbol{\omega}(t-1), \boldsymbol{\alpha}(t-1))\}$$

This section considers all distributed algorithms, including those where users know the history $\mathcal{H}(t)$. This might be available through a feedback message that specifies $(\boldsymbol{\alpha}(t), \boldsymbol{\omega}(t))$ at the end of each slot t. Theorem 1 shows that optimality can be achieved *without* this history information.

A. The distributed scheduling constraint

An algorithm for selecting $\boldsymbol{\alpha}(t)$ over slots $t \in \{0, 1, 2, \ldots\}$ is *distributed* if:

- There is an abstract set $\mathcal{X}$, called a *common information set*.
- There is a sequence of *commonly known random elements* $X(t) \in \mathcal{X}$ such that $\boldsymbol{\omega}(t)$ is independent of $X(t)$ for each $t \in \{0, 1, 2, \ldots\}$.
- There are deterministic functions $f_i(\omega_i, X)$ for each $i \in \{1, \ldots, N\}$ of the form $f_i : \Omega_i \times \mathcal{X} \to \mathcal{A}_i$.
- The decisions $\alpha_i(t)$ satisfy the following for all slots t:

$$\alpha_i(t) = f_i(\omega_i(t), X(t)) \ \text{ for all } i \in \{1, \ldots, N\} \tag{10}$$

Intuitively, the random elements $X(t)$ can be designed as any source of common randomness on which users can base their decisions. For example, $X(t)$ can have the form:

$$X(t) = (t, \mathcal{H}(t), Y(t))$$

where $Y(t)$ is a random element with support and distribution that can possibly depend on $\mathcal{H}(t)$ as well as past values $Y(\tau)$ for $\tau < t$. The only restriction is that $X(t)$ is independent of $\boldsymbol{\omega}(t)$. Because the $\boldsymbol{\omega}(t)$ vectors are i.i.d. over slots, $X(t)$ can be based on any events that occur before slot t.

B. The optimization problem

For notational convenience, define:

$$p_0(t) \triangleq -u(t) \ \ , \ \ \hat{p}_0(\boldsymbol{\alpha}(t), \boldsymbol{\omega}(t)) \triangleq -\hat{u}(\boldsymbol{\alpha}(t), \boldsymbol{\omega}(t))$$

Maximizing the time average expectation of $u(t)$ is equivalent to minimizing the time average expectation of $p_0(t)$. For each $k \in \{0, 1, \ldots, K\}$ and each slot $t > 0$ define:

$$\overline{p}_k(t) \triangleq \tfrac{1}{t} \textstyle\sum_{\tau=0}^{t-1} \mathbb{E}\left[p_k(\tau)\right]$$

The goal is to design a distributed algorithm that solves:

$$\begin{aligned} \text{Minimize:} \quad & \limsup_{t\to\infty} \overline{p}_0(t) && (11) \\ \text{Subject to:} \quad & \limsup_{t\to\infty} \overline{p}_k(t) \leq c_k \ \ \forall k \in \{1, \ldots, K\} && (12) \\ & \text{Condition (10) holds } \forall t \in \{0, 1, 2, \ldots\} && (13) \end{aligned}$$

It is assumed throughout this paper that the constraints (12)-(13) are *feasible*. Define p_0^{opt} as the infimum of all limiting $\overline{p}_0(t)$ values (11) achievable by algorithms that satisfy the constraints (12)-(13). The infimum is finite because $p_0(t)$ takes values in the same bounded set for all slots t.

C. Optimality via correlated scheduling

A *pure strategy* is defined as a vector-valued function:

$$\boldsymbol{g}(\boldsymbol{\omega}) = (g_1(\omega_1), g_2(\omega_2), \ldots, g_N(\omega_N))$$

where $g_i(\omega_i) \in \mathcal{A}_i$ for all $i \in \{1, \ldots, N\}$ and all $\omega_i \in \Omega_i$. The function $\boldsymbol{g}(\boldsymbol{\omega})$ specifies a distributed decision rule where each user i chooses α_i as a deterministic function of ω_i. Specifically, $\alpha_i = g_i(\omega_i)$. The total number of pure strategy functions $\boldsymbol{g}(\boldsymbol{\omega})$ is $\prod_{i=1}^{N} |\mathcal{A}_i|^{|\Omega_i|}$. Define M as this number, and enumerate these functions by $\boldsymbol{g}^{(m)}(\boldsymbol{\omega})$ for $m \in \{1, \ldots, M\}$. For each $m \in \{1, \ldots, M\}$ and $k \in \{0, 1, \ldots, K\}$ define:

$$r_k^{(m)} \triangleq \textstyle\sum_{\boldsymbol{\omega} \in \Omega} \pi(\boldsymbol{\omega}) \hat{p}_k(\boldsymbol{g}^{(m)}(\boldsymbol{\omega}), \boldsymbol{\omega}) \tag{14}$$

The value $r_k^{(m)}$ is the expected value of $p_k(t)$ given that users implement strategy $\boldsymbol{g}^{(m)}(\boldsymbol{\omega})$ on slot t.

Consider a randomized algorithm that, every slot t, independently uses strategy $\boldsymbol{g}^{(m)}(\boldsymbol{\omega})$ with probability θ_m. For each $k \in \{0, 1, \ldots, K\}$, this gives an expected penalty $\mathbb{E}\left[p_k(t)\right]$ of:

$$\begin{aligned} \mathbb{E}\left[p_k(t)\right] &= \textstyle\sum_{m=1}^{M} \theta_m \mathbb{E}\left[\hat{p}_k\left(\boldsymbol{g}^{(m)}(\boldsymbol{\omega}(t)), \boldsymbol{\omega}(t)\right)\right] \\ &= \textstyle\sum_{m=1}^{M} \theta_m r_k^{(m)} \end{aligned}$$

The following linear program solves for θ_m probabilities that minimize $\mathbb{E}\left[p_0(t)\right]$ over all algorithms that satisfy this specific randomized structure, subject to the constraints $\mathbb{E}\left[p_k(t)\right] \leq c_k$ for $k \in \{1, \ldots, K\}$:

$$\begin{aligned} \text{Minimize:} \quad & \textstyle\sum_{m=1}^{M} \theta_m r_0^{(m)} && (15) \\ \text{Subject to:} \quad & \textstyle\sum_{m=1}^{M} \theta_m r_k^{(m)} \leq c_k \quad \forall k \in \{1, \ldots, K\} && (16) \\ & \theta_m \geq 0 \quad \forall m \in \{1, \ldots, M\} && (17) \\ & \textstyle\sum_{m=1}^{M} \theta_m = 1 && (18) \end{aligned}$$

Such a randomized algorithm does not use the history $\mathcal{H}(t)$. The next theorem shows this algorithm structure is optimal.

Theorem 1: Suppose the problem (11)-(13) is feasible. Then the linear program (15)-(18) is feasible, and the optimal objective value (15) is equal to p_0^{opt}. Furthermore, there exist probabilities $(\theta_1, \ldots, \theta_M)$ that solve the linear program and satisfy $\theta_m > 0$ for at most $K+1$ values of $m \in \{1, \ldots, M\}$.

Proof: See Appendix. ■

The above theorem can be used to prove that the correlated reporting algorithm given in Section II, which uses $K+1 = 3$ pure strategies, is optimal for that example (see [18] for details).

IV. Reduced complexity

The linear program (15)-(18) uses variables $(\theta_1, \theta_2, \ldots, \theta_M)$, where M is the number of pure strategies. The value of M can be very large. This section shows that, if certain conditions hold, the set of strategy functions can be pruned to a smaller set without loss of optimality. For example, consider a two-user problem with binary actions, so that $|\mathcal{A}_i| = 2$ for $i \in \{1, 2\}$. Then:

$$M = 2^{|\Omega_1| + |\Omega_2|}$$

If certain conditions hold, strategies can be restricted to a set of size $\tilde{M}$, where:

$$\tilde{M} = (|\Omega_1| + 1)(|\Omega_2| + 1)$$

Thus, an exponentially large set is pruned to a smaller set with polynomial size.

A. The preferred action property

Suppose the sets $\mathcal{A}_i$ and Ω_i for each user $i \in \{1, \ldots, N\}$ are given by:

$$\mathcal{A}_i = \{0, 1, \ldots, |\mathcal{A}_i| - 1\} \quad (19)$$
$$\Omega_i = \{0, 1, \ldots, |\Omega_i| - 1\} \quad (20)$$

For notational convenience, for each $i \in \{1, \ldots, N\}$ let $[\boldsymbol{\alpha}_{\bar{i}}, \alpha_i]$ denote the N-dimensional vector $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_N)$, where $\boldsymbol{\alpha}_{\bar{i}}$ is the $(N-1)$-dimensional vector of α_j components for $j \neq i$. This notation facilitates comparison of two vectors that differ in just one coordinate. Define $\mathcal{A}_{\bar{i}}$ and $\Omega_{\bar{i}}$ as the set of all possible $(N-1)$-dimensional vectors $\boldsymbol{\alpha}_{\bar{i}}$ and $\boldsymbol{\omega}_{\bar{i}}$, respectively.

Definition 1: A penalty function $\hat{p}(\boldsymbol{\alpha}, \boldsymbol{\omega})$ has the *preferred action property* if for all $i \in \{1, \ldots, N\}$, all $\boldsymbol{\alpha}_{\bar{i}} \in \mathcal{A}_{\bar{i}}$, and all $\boldsymbol{\omega}_{\bar{i}} \in \Omega_{\bar{i}}$, one has:

$$\hat{p}([\boldsymbol{\alpha}_{\bar{i}}, \alpha], [\boldsymbol{\omega}_{\bar{i}}, \omega]) - \hat{p}([\boldsymbol{\alpha}_{\bar{i}}, \beta], [\boldsymbol{\omega}_{\bar{i}}, \omega])$$
$$\geq \hat{p}([\boldsymbol{\alpha}_{\bar{i}}, \alpha], [\boldsymbol{\omega}_{\bar{i}}, \gamma]) - \hat{p}([\boldsymbol{\alpha}_{\bar{i}}, \beta], [\boldsymbol{\omega}_{\bar{i}}, \gamma])$$

whenever α, β are values in $\mathcal{A}_i$ that satisfy $\alpha > \beta$, and ω, γ are values in Ω_i that satisfy $\omega < \gamma$.

Intuitively, the above definition means that if user i compares the difference in penalty under the actions $\alpha_i(t) = \alpha$ and $\alpha_i(t) = \beta$ (where $\alpha > \beta$), this difference is non-increasing in the user i observation $\omega_i(t)$ (assuming all other actions and events $\boldsymbol{\alpha}_{\bar{i}}$ and $\boldsymbol{\omega}_{\bar{i}}$ are held fixed).

For example, any function $\hat{p}(\boldsymbol{\alpha}, \boldsymbol{\omega})$ that does not depend on $\boldsymbol{\omega}$ trivially satisfies the preferred action property. This is the case for the $\hat{p}_1(\cdot)$ and $\hat{p}_2(\cdot)$ functions in (6) used to represent power expenditures for the sensor network example of Section II. Further, the utility function (5) in that example yields $\hat{p}_0(\cdot) = -\hat{u}(\cdot)$ that satisfies the preferred action property, as shown by the next lemma.

Lemma 1: Suppose $\mathcal{A}_i = \{0, 1\}$ and Ω_i satisfies (20) for $i \in \{1, \ldots, N\}$. Define:

$$\hat{u}(\boldsymbol{\alpha}, \boldsymbol{\omega}) = \min\left[\sum_{i=1}^N \phi_i(\omega_i)\alpha_i, b\right]$$

for some (real-valued) constant b and some (real-valued) non-decreasing functions $\phi_i(\omega_i)$. Then the penalty function $\hat{p}_0(\boldsymbol{\alpha}, \boldsymbol{\omega}) = -\hat{u}(\boldsymbol{\alpha}, \boldsymbol{\omega})$ has the preferred action property.

Lemma 2: Suppose $\mathcal{A}_i = \{0, 1\}$ and Ω_i satisfies (20) for $i \in \{1, \ldots, N\}$. Define the utility function $\hat{u}(\boldsymbol{\alpha}, \boldsymbol{\omega})$ according to the multi-access example equation (4). Then the penalty function $\hat{p}_0(\boldsymbol{\alpha}, \boldsymbol{\omega}) = -\hat{u}(\boldsymbol{\alpha}, \boldsymbol{\omega})$ has the preferred action property.

Lemma 3: Suppose $\mathcal{A}_i$ and Ω_i satisfy (19)-(20). Define $\hat{p}(\boldsymbol{\alpha}, \boldsymbol{\omega})$ by:

$$\hat{p}(\boldsymbol{\alpha}, \boldsymbol{\omega}) = \prod_{i=1}^N \phi_i(\omega_i)\psi_i(\alpha_i)$$

where $\phi_i(\omega_i)$, $\psi_i(\alpha_i)$ are non-negative functions for all $i \in \{1, \ldots, N\}$. Suppose that for each $i \in \{1, \ldots, N\}$, $\phi_i(\omega_i)$ is non-increasing in ω_i and $\psi_i(\alpha_i)$ is non-decreasing in α_i. Then $\hat{p}(\boldsymbol{\alpha}, \boldsymbol{\omega})$ has the preferred action property.

Lemma 4: Suppose $\mathcal{A}_i$ and Ω_i satisfy (19)-(20), and $\hat{p}_1(\boldsymbol{\alpha}, \boldsymbol{\omega}), \ldots, \hat{p}_R(\boldsymbol{\alpha}, \boldsymbol{\omega})$ are a collection of functions that have the preferred action property (where R is a given positive integer). Then for any non-negative weights $w_1, \ldots, w_R$, the following function has the preferred action property:

$$\hat{p}(\boldsymbol{\alpha}, \boldsymbol{\omega}) = \sum_{r=1}^R w_r \hat{p}_r(\boldsymbol{\alpha}, \boldsymbol{\omega})$$

The proofs of Lemmas 1-4 are given in [18].

B. Independent events and reduced complexity

Consider the special case when the components of $\boldsymbol{\omega}(t) = (\omega_1(t), \ldots, \omega_N(t))$ are mutually independent, so that:

$$\pi(\boldsymbol{\omega}) = \prod_{i=1}^N q_i(\omega_i) \quad (21)$$

where $q_i(\omega_i) \triangleq Pr[\omega_i(t) = \omega_i]$. Without loss of generality, assume $q_i(\omega_i) > 0$ for all $i \in \{1, \ldots, N\}$ and all $\omega_i \in \Omega_i$. Recall that a pure strategy $\boldsymbol{g}(\boldsymbol{\omega})$ is composed of individual *strategy functions* $g_i(\omega_i)$ for each user i:

$$\boldsymbol{g}(\boldsymbol{\omega}) = (g_1(\omega_1), \ldots, g_N(\omega_N))$$

Theorem 2: (Non-decreasing strategy functions) If all penalty functions $\hat{p}_k(\boldsymbol{\alpha}, \boldsymbol{\omega})$ for $k \in \{0, 1, \ldots, K\}$ have the preferred action property, and if $\boldsymbol{\omega}(t)$ satisfies the independence property (21), then it suffices to restrict attention to strategy functions $g_i(\omega_i)$ that are non-decreasing in ω_i.

Proof: Fix $m \in \{1, \ldots, M\}$, $i \in \{1, \ldots, N\}$, and fix two elements ω and γ in Ω_i that satisfy $\omega < \gamma$. Suppose the linear program (15)-(18) places weight $\theta_m > 0$ on a strategy function $\boldsymbol{g}^{(m)}(\boldsymbol{\omega})$ that satisfies $g_i^{(m)}(\omega) > g_i^{(m)}(\gamma)$ (so the non-decreasing requirement is violated). The goal is to show this can be replaced by new strategies that do not violate the non-decreasing requirement for elements ω and γ, without loss of optimality.

Define $\alpha = g_i^{(m)}(\omega)$ and $\beta = g_i^{(m)}(\gamma)$. Then $\alpha > \beta$. Define two new functions:

$$\begin{aligned} g_i^{(m),low}(\omega_i) &= \left\{ \begin{array}{ll} g_i^{(m)}(\omega_i) & \text{if } \omega_i \notin \{\omega, \gamma\} \\ \beta & \text{if } \omega_i \in \{\omega, \gamma\} \end{array} \right. \\ g_i^{(m),high}(\omega_i) &= \left\{ \begin{array}{ll} g_i^{(m)}(\omega_i) & \text{if } \omega_i \notin \{\omega, \gamma\} \\ \alpha & \text{if } \omega_i \in \{\omega, \gamma\} \end{array} \right. \end{aligned}$$

Unlike the original function $g_i^{(m)}(\omega_i)$, these new functions satisfy:

$$\begin{aligned} g_i^{(m),low}(\omega) &\leq g_i^{(m),low}(\gamma) \\ g_i^{(m),high}(\omega) &\leq g_i^{(m),high}(\gamma) \end{aligned}$$

Define $\boldsymbol{g}^{(m),low}(\boldsymbol{\omega})$ and $\boldsymbol{g}^{(m),high}(\boldsymbol{\omega})$ by replacing the ith component function $g_i^{(m)}(\omega_i)$ of $\boldsymbol{g}^{(m)}(\boldsymbol{\omega})$ with new component functions $g_i^{(m),low}(\omega_i)$ and $g_i^{(m),high}(\omega_i)$, respectively. Let $p_k^{old}(t)$ be the kth penalty incurred in the (old) strategy that uses $\boldsymbol{g}^{(m)}(\boldsymbol{\omega})$ with probability θ_m. Let $p_k^{new}(t)$ be the corresponding penalty under a (new) strategy that, instead of using $\boldsymbol{g}^{(m)}(\boldsymbol{\omega})$ with probability θ_m, uses:

- $\boldsymbol{g}^{(m),low}(\boldsymbol{\omega})$ with probability $\theta_m q_i(\gamma)/(q_i(\omega) + q_i(\gamma))$.
- $\boldsymbol{g}^{(m),high}(\boldsymbol{\omega})$ with probability $\theta_m q_i(\omega)/(q_i(\omega) + q_i(\gamma))$.

Define $\mathcal{E}_0$ as the event that the old strategy uses $\boldsymbol{g}^{(m)}(\boldsymbol{\omega})$. Let $\boldsymbol{\omega}_{\bar{i}}(t)$ denote the $(N-1)$-dimensional vector of components $\omega_j(t)$ for $j \neq i$. Fix any vector $\boldsymbol{\omega}_{\bar{i}} \in \Omega_{\bar{i}}$. Define $\boldsymbol{\alpha}_{\bar{i}}$ as the corresponding $(N-1)$-dimensional vector of $g_j^{(m)}(\omega_j)$ values for $j \neq i$. Then:

- If $\mathcal{E}_0$ holds, $\boldsymbol{\omega}_{\bar{i}}(t) = \boldsymbol{\omega}_{\bar{i}}$, $\omega_i(t) = \omega$, and $\boldsymbol{g}^{(m),low}(\boldsymbol{\omega})$ is used by the new strategy, then $\boldsymbol{\omega}(t) = [\boldsymbol{\omega}_{\bar{i}}, \omega]$ and:

  $$\begin{aligned} p_k^{new}(t) &= \hat{p}_k\left(\boldsymbol{g}^{(m),low}\left([\boldsymbol{\omega}_{\bar{i}}, \omega]\right), [\boldsymbol{\omega}_{\bar{i}}, \omega]\right) \\ &= \hat{p}_k\left([\boldsymbol{\alpha}_{\bar{i}}, \beta], [\boldsymbol{\omega}_{\bar{i}}, \omega]\right) \end{aligned}$$

  Further, since the old strategy used $g_i^{(m)}(\omega) = \alpha$:

  $$\begin{aligned} p_k^{old}(t) &= \hat{p}_k\left(\boldsymbol{g}^{(m)}\left([\boldsymbol{\omega}_{\bar{i}}, \omega]\right), [\boldsymbol{\omega}_{\bar{i}}, \omega]\right) \\ &= \hat{p}_k\left([\boldsymbol{\alpha}_{\bar{i}}, \alpha], [\boldsymbol{\omega}_{\bar{i}}, \omega]\right) \end{aligned}$$

- If $\mathcal{E}_0$ holds, $\boldsymbol{\omega}_{\bar{i}}(t) = \boldsymbol{\omega}_{\bar{i}}$, $\omega_i(t) = \gamma$, and $\boldsymbol{g}^{(m),high}(\boldsymbol{\omega})$ is used by the new strategy, then $\boldsymbol{\omega}(t) = [\boldsymbol{\omega}_{\bar{i}}, \gamma]$ and:

  $$\begin{aligned} p_k^{new}(t) &= \hat{p}_k\left(\boldsymbol{g}^{(m),high}\left([\boldsymbol{\omega}_{\bar{i}}, \gamma]\right), [\boldsymbol{\omega}_{\bar{i}}, \gamma]\right) \\ &= \hat{p}_k\left([\boldsymbol{\alpha}_{\bar{i}}, \alpha], [\boldsymbol{\omega}_{\bar{i}}, \gamma]\right) \end{aligned}$$

  Further, since the old strategy used $g_i^{(m)}(\gamma) = \beta$:

  $$\begin{aligned} p_k^{old}(t) &= \hat{p}_k\left(\boldsymbol{g}^{(m)}\left([\boldsymbol{\omega}_{\bar{i}}, \omega]\right), [\boldsymbol{\omega}_{\bar{i}}, \omega]\right) \\ &= \hat{p}_k\left([\boldsymbol{\alpha}_{\bar{i}}, \beta], [\boldsymbol{\omega}_{\bar{i}}, \gamma]\right) \end{aligned}$$

- Suppose $\boldsymbol{\omega}_{\bar{i}}(t) = \boldsymbol{\omega}_{\bar{i}}$, but neither of the above two events are satisfied on slot t. That is, neither of the events $\mathcal{E}_1$ or $\mathcal{E}_2$ are true, where:

  $$\begin{aligned} \mathcal{E}_1 &\triangleq \mathcal{E}_0 \cap \{\omega_i(t) = \omega\} \cap \{\boldsymbol{g}^{(m),low}(\boldsymbol{\omega}) \text{ is used}\} \\ \mathcal{E}_2 &\triangleq \mathcal{E}_0 \cap \{\omega_i(t) = \gamma\} \cap \{\boldsymbol{g}^{(m),high}(\boldsymbol{\omega}) \text{ is used}\} \end{aligned}$$

  Then $p_k^{new}(t) - p_k^{old}(t) = 0$.

It follows that:

$$\begin{aligned} &\mathbb{E}\left[p_k^{new}(t) - p_k^{old}(t) | \boldsymbol{\omega}_{\bar{i}}(t) = \boldsymbol{\omega}_{\bar{i}}\right] \\ &= \theta_m q_i(\omega)\left(\frac{q_i(\gamma)}{q_i(\omega) + q_i(\gamma)}\right) \times \\ &\quad [\hat{p}_k\left([\boldsymbol{\alpha}_{\bar{i}}, \beta], [\boldsymbol{\omega}_{\bar{i}}, \omega]\right) - \hat{p}_k\left([\boldsymbol{\alpha}_{\bar{i}}, \alpha], [\boldsymbol{\omega}_{\bar{i}}, \omega]\right)] \\ &\quad + \theta_m q_i(\gamma)\left(\frac{q_i(\omega)}{q_i(\omega) + q_i(\gamma)}\right) \times \\ &\quad [\hat{p}_k\left([\boldsymbol{\alpha}_{\bar{i}}, \alpha], [\boldsymbol{\omega}_{\bar{i}}, \gamma]\right) - \hat{p}_k\left([\boldsymbol{\alpha}_{\bar{i}}, \beta], [\boldsymbol{\omega}_{\bar{i}}, \gamma]\right)] \end{aligned} \tag{22}$$

where the above uses the fact that $\omega_i(t)$ is independent of $\boldsymbol{\omega}_{\bar{i}}(t)$, so conditioning on $\boldsymbol{\omega}_{\bar{i}}(t) = \boldsymbol{\omega}_{\bar{i}}$ does not change the distribution of $\omega_i(t)$. Because $\hat{p}_k(\cdot)$ satisfies the preferred action property and $\alpha > \beta$, $\omega < \gamma$, one has:

$$\begin{aligned} &[\hat{p}_k\left([\boldsymbol{\alpha}_{\bar{i}}, \alpha], [\boldsymbol{\omega}_{\bar{i}}, \omega]\right) - \hat{p}_k\left([\boldsymbol{\alpha}_{\bar{i}}, \beta], [\boldsymbol{\omega}_{\bar{i}}, \omega]\right)] \\ &\geq [\hat{p}_k\left([\boldsymbol{\alpha}_{\bar{i}}, \alpha], [\boldsymbol{\omega}_{\bar{i}}, \gamma]\right) - \hat{p}_k\left([\boldsymbol{\alpha}_{\bar{i}}, \beta], [\boldsymbol{\omega}_{\bar{i}}, \gamma]\right)] \end{aligned}$$

and hence (22) is less than or equal to zero. This holds when conditioning on all possible values of $\boldsymbol{\omega}_{\bar{i}}(t)$, and so:

$$\mathbb{E}\left[p_k^{new}(t) - p_k^{old}(t)\right] \leq 0$$

This holds for all penalties $k \in \{0, 1, \ldots, K\}$, and so the modified algorithm still satisfies all constraints with an optimal value for $\mathbb{E}[p_0(t)]$. The interchange can be repeated until all strategy functions are non-decreasing. ■

In the special case of binary actions, so that $\mathcal{A}_i = \{0, 1\}$ for all $i \in \{1, \ldots, N\}$, all non-decreasing strategy functions $g_i(\omega_i)$ have the following form:

$$g_i(\omega_i) = \left\{ \begin{array}{ll} 0 & \text{if } \omega_i < h_i^* \\ 1 & \text{if } \omega_i \geq h_i^* \end{array} \right. \tag{23}$$

for some threshold $h_i^* \in \{0, 1, \ldots, |\Omega_i|\}$. There are $|\Omega_i| + 1$ such threshold functions, whereas the total number of strategy functions for user i is $2^{|\Omega_i|}$. Restricting to the threshold functions significantly decreases complexity.

V. Online optimization

The randomized policy of Theorem 1 solves the problem (11)-(13) with no interaction amongst users. However, it requires θ_m values to be computed at time 0 based on knowledge of the $\pi[\boldsymbol{\omega}]$ probabilities. This section presents a dynamic algorithm that does not require these probabilities, and that can adapt if they change. Such an algorithm requires feedback messages. However, for distributed implementation, it is assumed throughout that all feedback takes place after a fixed delay of at least one slot. Theorem 1 ensures the linear program (15)-(18) still characterizes optimality in this context. The dynamic algorithm can also be viewed as an online solution to this linear program.

Let $\tilde{M}$ be the number of pure strategies required for consideration in the linear program (where $\tilde{M}$ is possibly smaller than M, as discussed in the previous section). Reorder the functions $g^{(m)}(\boldsymbol{\omega})$ if necessary so that every slot t, the system chooses a strategy function in the set $\{g^{(1)}(\boldsymbol{\omega}), \ldots, g^{(\tilde{M})}(\boldsymbol{\omega})\}$.

Suppose all users receive feedback specifying the values of the penalties $p_1(t), \ldots, p_K(t)$ at the end of slot $t + D$, where D is a non-negative integer that represents a system delay.

Any mechanism for delivering this feedback can be used. For each constraint $k \in \{1, \ldots, K\}$, define a *virtual queue* $Q_k(t)$ and initialize $Q_k(0)$ to a commonly known value (typically 0). For each $t \in \{0, 1, 2, \ldots\}$ the queue is updated by:

$$Q_k(t+1) = \max[Q_k(t) + p_k(t-D) - c_k, 0] \qquad (24)$$

Each user can iterate the above equation based on information available at the end of slot t. Thus, all users know the value of $Q_k(t)$ at the beginning of each slot t. If $D > 0$, define $p_k(-1) = p_k(-2) = \cdots = p_k(-D) = 0$. Stabilizing the above virtual queues ensures the time average penalties satisfy their constraints [1].

Define $\boldsymbol{Q}(t) = (Q_1(t), \ldots, Q_K(t))$. Define $L(t)$ by:

$$L(t) \triangleq \frac{1}{2}\sum_{k=1}^{K} Q_k(t)^2$$

Define $\Delta(t) \triangleq L(t+1) - L(t)$, called the *Lyapunov drift*. Motivated by the theory in [1], the approach is to choose probabilities every slot to greedily minimize a bound on the *drift-plus-penalty expression* $\mathbb{E}\left[\Delta(t+D) + Vp_0(t)|\boldsymbol{Q}(t)\right]$, where V is a non-negative weight that affects a performance tradeoff. The D-shifted drift term $\Delta(t+D)$ is different from [1] and is used because of the delayed feedback structure of the queue update (24). The intuition is that minimizing $\Delta(t+D)$ maintains queue stability, while adding the weighted penalty term $Vp_0(t)$ biases decisions in favor of lower penalties. The following *drift-plus-penalty* algorithm results from this greedy minimization (see detailed development in [18]). Every slot t:

- Users observe the queue vector $\boldsymbol{Q}(t)$.
- Users apply the pure strategy $\boldsymbol{g}^{(m)}(\boldsymbol{\omega})$, where m is the index in $\{1, \ldots, \tilde{M}\}$ that minimizes the expression:

$$Vr_0^{(m)} + \sum_{k=1}^{K} Q_k(t) r_k^{(m)} \qquad (25)$$

- The delayed penalty information $p_k(t-D)$ is observed and queues are updated via (24).

A. Performance Analysis

Theorem 3: If the problem (11)-(13) is feasible, then under the drift-plus-penalty algorithm for any $V \geq 0$:

- All desired constraints (12)-(13) are satisfied.
- For all $t > 0$, the time average expectation of $p_0(t)$ satisfies:

$$\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\left[p_0(\tau)\right] \leq p_0^{opt} + \frac{B(1+2D)}{V} + \frac{\mathbb{E}\left[L(D)\right]}{Vt} \qquad (26)$$

where the constant B is defined:

$$B \triangleq \max_{m \in \{1, \ldots, \tilde{M}\}} \frac{1}{2}\sum_{k=1}^{K}\sum_{\boldsymbol{\omega}\in\Omega}\pi(\boldsymbol{\omega})\left|\hat{p}_k\left(\boldsymbol{g}^{(m)}(\boldsymbol{\omega}), \boldsymbol{\omega}\right) - c_k\right|^2$$

- For all $t > 0$, the time average expectation of $p_k(t)$ satisfies the following for all $k \in \{1, \ldots, K\}$:

$$\frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\left[p_k(\tau)\right] \leq c_k + O(\sqrt{V/t}) \qquad (27)$$

Proof: See [18]. ■

The above theorem shows the time average expectation of $p_0(t)$ is within $O(1/V)$ of optimality. It can be pushed as close to optimal as desired by increasing the V parameter. The tradeoff is in the amount of time required for the time average expected penalties to be close to their desired constraints.

B. The approximate drift-plus-penalty algorithm

The online algorithm assumes perfect knowledge of the $r_k^{(m)}$ values. These can be computed by (14) if the event probabilities $\pi(\boldsymbol{\omega})$ are known. Suppose these probabilities are unknown, but delayed samples $\boldsymbol{\omega}(t-D)$ are available at the end of each slot t. Let W be a positive integer that represents a *sample size*. The $r_k^{(m)}$ values can be approximated by:

$$\tilde{r}_k^{(m)}(t) = \frac{1}{W}\sum_{w=0}^{W-1}\hat{p}_k\left(\boldsymbol{g}^{(m)}(\boldsymbol{\omega}(t-D-w)), \boldsymbol{\omega}(t-D-w)\right)$$

The approximate algorithm uses $\tilde{r}_k^{(m)}(t)$ values in replace of $r_k^{(m)}$ in the expression (25). Analysis in [19] shows that the performance gap between exact and approximate drift-plus-penalty implementations is $O(1/\sqrt{W})$.

VI. Simulations

A. Ergodic performance for a 2 user system

First consider the 2 user sensor network example of Section II. The approximate drift-plus-penalty algorithm of Section V-B is used with a delay of $D = 10$ slots and a moving average window size of $W = 40$ slots. The algorithm is not aware of the system probabilities. The objective of this simulation is to find how close the achieved utility is to the optimal value $u^{opt} = 23/48 \approx 0.47917$ computed in Section II-B. Recall that the desired power constraints are $\overline{p}_i \leq 1/3$ for each user $i \in \{1, 2\}$. The table in Fig. 1 gives results for various values of V. For $V \geq 50$ the achieved utility differs from optimality only in the fourth decimal place.

V	$\overline{u}$	$\overline{p}_1$	$\overline{p}_2$
1	0.344639	0.259764	0.219525
5	0.454557	0.333158	0.267161
10	0.472763	0.333335	0.300415
25	0.478186	0.333346	0.326948
50	0.479032	0.333369	0.332873
100	0.479218	0.333406	0.333334

Fig. 1. Algorithm performance over $t = 10^6$ slots ($D = 10$, $W = 40$). Recall that $u^{opt} = 23/48 \approx 0.47917$.

B. Ergodic performance for a 3 user system

Consider a network of 3 sensors that communicate reports to a fusion center. The event processes $\omega_i(t)$ for each sensor $i \in \{1, 2, 3\}$ take values in the same 10 element set $\mathcal{B}$:

$$\mathcal{B} \triangleq \{0, 1, 2, 3, \ldots, 9\}$$

Consider binary actions $\alpha_i(t) \in \{0,1\}$, where $\alpha_i(t) = 1$ corresponds to sensor i sending a report, and incurs a power cost of 1 for that sensor. The penalty and utility functions are:

$$\begin{aligned} \hat{p}_i(\alpha_i, \omega_i) &= \alpha_i \quad \forall i \in \{1,2,3\} \\ \hat{u}(\boldsymbol{\alpha}, \boldsymbol{\omega}) &= \min\left[\frac{\alpha_1\omega_1}{10} + \frac{\alpha_2\omega_2 + \alpha_3\omega_3}{20}, 1\right] \end{aligned}$$

Thus, sensor 1 brings more utility than the other sensors.

Assume $\omega_1(t), \omega_2(t), \omega_3(t)$ are mutually independent and uniformly distributed over $\mathcal{B}$. The requirements for Theorem 2 hold, and so one can restrict attention to the 11 threshold functions $g_i(\omega_i)$ of the type (23). As it does not make sense to report when $\omega_i(t) = 0$, the functions $g_i(\omega) = 1$ for all ω can be removed. This leaves only 10 threshold functions at each user, for a total of $10^3 = 1000$ strategy functions $\boldsymbol{g}^{(m)}(\boldsymbol{\omega})$ to be considered every slot. The approximate drift-plus-penalty algorithm of Section V-B is simulated over $t = 10^6$ slots with a delay $D = 10$ and for various choices of the moving average window size W and the parameter V. All average power constraints were met for all choices of V and W. The achieved utility is shown in Fig. 2. The utility increases to a limiting value as V is increased. This limiting value can be improved by adjusting the number of samples W used in the moving average. Increasing W from 40 to 200 gives a small improvement in performance. There is only a negligible improvement when W is further increased to 400 (the curves for $W = 200$ and $W = 400$ look identical).

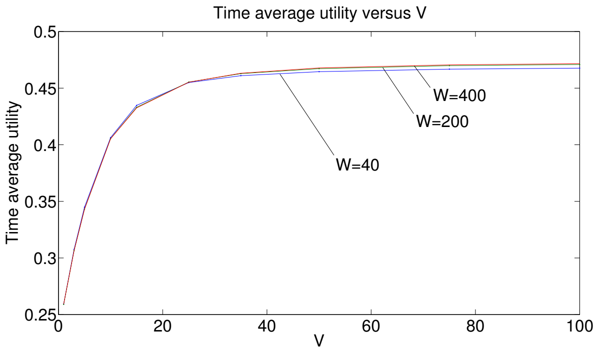

Fig. 2. Achieved utility $\overline{u}$ versus V for various choices of W.

Fig. 3 demonstrates how the V parameter affects the rate of convergence to the desired constraints. The window size is fixed to $W = 40$ and the value $\max[\overline{p}_1(t), \overline{p}_2(t), \overline{p}_3(t)]$ is plotted for $t \in \{0, 1, \ldots, 2000\}$ (where $\overline{p}_i(t)$ is the empirical average power expenditure of user i up to slot t). This value approaches the constraint $1/3$ more slowly when V is large.

C. Adaptation to non-ergodic changes

The initial queue state determines the coefficient of an $O(1/t)$ transient in the performance bounds of the system (consider the $\mathbb{E}\left[L(D)\right]/(Vt)$ term in (26)). Thus, if system probabilities change abruptly, the system can be viewed as restarting with a different initial condition. Thus, one expects the system to react robustly to such changes.

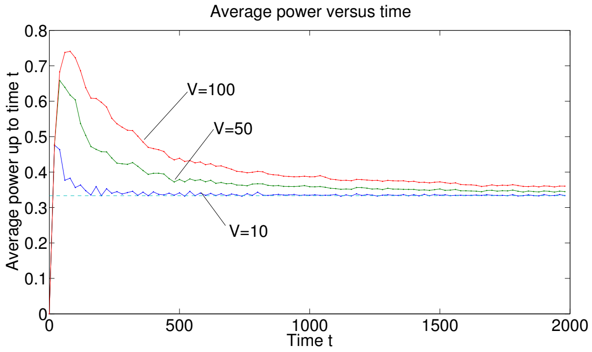

Fig. 3. An illustration of the rate of convergence to the desired constraint $1/3$ for various choices of V. The curves plot $\max[\overline{p}_1(t), \overline{p}_2(t), \overline{p}_3(t)]$ versus t.

To illustrate this, consider the same 3-user system of the previous subsection, using $V = 50, W = 40$. The event processes $\omega_i(t)$ have the same probabilities as given in the previous subsection for slots $t < 4000$ and $t > 8000$. Call this *distribution type 1*. However, for slots $t \in \{4000, \ldots, 8000\}$, the $\omega_i(t)$ processes are independently chosen with a different distribution as follows:

- $Pr[\omega_1(t) = 0] = Pr[\omega_1(t) = 9] = 1/2$.
- $Pr[\omega_2(t) = k] = 1/4$ for $k \in \{6, 7, 8, 9\}$.
- $Pr[\omega_3(t) = k] = 1/4$ for $k \in \{6, 7, 8, 9\}$.

This is called *distribution type 2*.

Fig. 4 shows average utility and average power over the first 12000 slots. Values at each slot t are averaged over 2000 independent system runs. The two dashed horizontal lines in the top plot of the figure are long term time average utilities achieved over 10^6 slots under probabilities that are fixed at distribution type 1 and type 2, respectively. It is seen that the system adapts to the non-ergodic change by quickly adjusting to the new optimal average utility. The figure also plots average power of user 1 versus time, with a dashed horizontal line at the power constraint $1/3$. A noticeable disturbance in average power occurs at the non-ergodic changes in distribution.

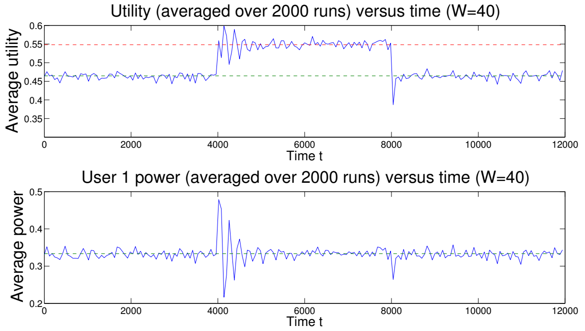

Fig. 4. A sample path of average utility and power versus time. Values at each time slot t are obtained by averaging the actual utility and power used by the algorithm on that slot over 2000 independent simulation runs.

VII. Conclusions

This paper treated distributed scheduling in a multi-user system where users know their own observations and actions, but not those of others. Optimal distributed policies were constructed by correlating decisions via a source of common randomness. The optimal policy is computable via a linear program if all system probabilities are known, and through an online algorithm with virtual queues if probabilities are unknown. The online algorithm assumes there is delayed feedback about previous penalties and rewards. The algorithm was shown in simulation to adapt when system probabilities change. If the penalty and utility functions satisfy a *preferred action property*, a complexity reduction result was shown to reduce the number of pure strategies required for consideration. In some cases, this reduces an exponentially complex algorithm to one that has only polynomial complexity.

Appendix — Proof of Theorem 1

Define the $(K+1)$-dimensional *penalty vectors*:

$$\begin{aligned}\boldsymbol{p}(t) &= (p_0(t), p_1(t), \ldots, p_K(t)) \\ \hat{\boldsymbol{p}}(\boldsymbol{\alpha}, \boldsymbol{\omega}) &= (\hat{p}_0(\boldsymbol{\alpha}, \boldsymbol{\omega}), \hat{p}_1(\boldsymbol{\alpha}, \boldsymbol{\omega}), \ldots, \hat{p}_K(\boldsymbol{\alpha}, \boldsymbol{\omega}))\end{aligned}$$

For each $m \in \{1, \ldots, M\}$, define:

$$\boldsymbol{r}^{(m)} \triangleq \sum_{\boldsymbol{\omega}\in\Omega} \pi(\boldsymbol{\omega})\hat{\boldsymbol{p}}(\boldsymbol{g}^{(m)}(\boldsymbol{\omega}), \boldsymbol{\omega}) = (r_0^{(m)}, r_1^{(m)}, \ldots, r_K^{(m)})$$

Define $\mathcal{R}$ as the convex hull of these vectors:

$$\mathcal{R} \triangleq Conv\left(\{\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(M)}\}\right)$$

Lemma 5: Let $\boldsymbol{\alpha}(t)$ be decisions of an algorithm that satisfies the distributed scheduling constraint (10) every slot. Then:
(a) $\mathbb{E}\left[\boldsymbol{p}(t)\right] \in \mathcal{R}$ for all $t \in \{0, 1, 2, \ldots\}$.
(b) $\overline{\boldsymbol{p}}(t) \in \mathcal{R}$ for all $t \in \{1, 2, 3, \ldots\}$, where

$$\overline{\boldsymbol{p}}(t) \triangleq \frac{1}{t}\sum_{\tau=0}^{t-1} \mathbb{E}\left[\boldsymbol{p}(\tau)\right]$$

Proof: Part (b) follows immediately from part (a) together with the fact that $\mathcal{R}$ is convex. To prove part (a), fix a slot $t \in \{0, 1, 2, \ldots\}$. By (10), the users make decisions:

$$\boldsymbol{\alpha}(t) = (f_1(\omega_1(t), X(t)), \ldots, f_N(\omega_N(t), X(t)))$$

For each $X(t) \in \mathcal{X}$ and $\boldsymbol{\omega} \in \Omega$, define:

$$\boldsymbol{g}_{X(t)}(\boldsymbol{\omega}) = (f_1(\omega_1, X(t)), \ldots, f_N(\omega_N, X(t)))$$

Then, given $X(t)$, the function $\boldsymbol{g}_{X(t)}(\boldsymbol{\omega})$ is a pure strategy. Hence, $\boldsymbol{g}_{X(t)}(\boldsymbol{\omega}) = \boldsymbol{g}^{(m)}(\boldsymbol{\omega})$ for some $m \in \{1, \ldots, M\}$. Define $m_{X(t)}$ as the value $m \in \{1, \ldots, M\}$ for which this holds. Thus, $\boldsymbol{g}_{X(t)}(\boldsymbol{\omega}) = \boldsymbol{g}^{(m_{X(t)})}(\boldsymbol{\omega})$, and:

$$\begin{aligned}\mathbb{E}\left[\boldsymbol{p}(t)|X(t)\right] &= \mathbb{E}\left[\hat{\boldsymbol{p}}(\boldsymbol{\alpha}(t), \boldsymbol{\omega}(t))|X(t)\right] \\ &= \mathbb{E}\left[\hat{\boldsymbol{p}}\left(\boldsymbol{g}^{(m_{X(t)})}(\boldsymbol{\omega}(t)), \boldsymbol{\omega}(t)\right)|X(t)\right] \\ &= \sum_{\boldsymbol{\omega}\in\Omega} \pi(\boldsymbol{\omega})\hat{\boldsymbol{p}}\left(\boldsymbol{g}^{(m_{X(t)})}(\boldsymbol{\omega}), \boldsymbol{\omega}\right) \\ &= \boldsymbol{r}^{(m_{X(t)})}\end{aligned}$$

Taking expectations of both sides and using the law of iterated expectations gives:

$$\mathbb{E}\left[\boldsymbol{p}(t)\right] = \sum_{m=1}^{M} Pr[m_{X(t)} = m]\boldsymbol{r}^{(m)}$$

The above is a convex combination of $\{\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(M)}\}$, and hence is in $\mathcal{R}$. ■

Lemma 6: There exist real numbers $r_1, r_2, \ldots, r_K$ that satisfy the following:

$$r_k \leq c_k \quad \forall k \in \{1, \ldots, K\} \tag{28}$$

$$(p_0^{opt}, r_1, r_2, \ldots, r_K) \in \mathcal{R} \tag{29}$$

Furthermore, the vector in (29) is on the *boundary* of $\mathcal{R}$.

Proof: The proof is an application of Lemma 5 and is omitted for brevity (see [18]). ■

Because $\mathcal{R} = Conv(\{\boldsymbol{r}^{(1)}, \ldots, \boldsymbol{r}^{(M)}\})$, Lemma 6 implies there are probabilities θ_m that sum to 1 such that:

$$(p_0^{opt}, r_1, \ldots, r_K) = \sum_{m=1}^{M} \theta_m \boldsymbol{r}^{(m)}$$

where the r_k values satisfy (28). Because $\mathcal{R}$ is a $(K+1)$-dimensional set and the above vector is on the *boundary* of $\mathcal{R}$, a simple extension of Caratheodory's theorem ensures the vector can be expressed with at most $K+1$ non-zero θ_m values. This proves Theorem 1.

References

[1] M. J. Neely. *Stochastic Network Optimization with Application to Communication and Queueing Systems*. Morgan & Claypool, 2010.

[2] B. Liu, P. Terlecky, A. Bar-Noy, R. Govindan, M. J. Neely, and D. Rawitz. Optimizing information credibility in social swarming applications. *IEEE Trans. on Parallel and Distributed Systems*, vol. 23, no. 6, pp. 1147-1158, June 2012.

[3] N. Michelusi and M. Zorzi. Optimal random multiaccess in energy harvesting wireless sensor networks. *Proc. IEEE International Conference on Communications*, to appear.

[4] M. J. Osborne and A. Rubinstein. *A Course in Game Theory*. MIT Press, Cambridge, MA, 1994.

[5] Y. Shoham and K. Leyton-Brown. *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, NY, NY, 2009.

[6] R. Aumann. Subjectivity and correlation in randomized strategies. *Journal of Mathematical Economics*, vol. 1, pp. 67-96, 1974.

[7] R. Aumann. Correlated equilibrium as an expression of bayesian rationality. *Econometrica*, vol. 55, pp. 1-18, 1987.

[8] L. Georgiadis, M. J. Neely, and L. Tassiulas. Resource allocation and cross-layer control in wireless networks. *Foundations and Trends in Networking*, vol. 1, no. 1, pp. 1-149, 2006.

[9] M. J. Neely, E. Modiano, and C. Li. Fairness and optimal stochastic control for heterogeneous networks. *IEEE/ACM Transactions on Networking*, vol. 16, no. 2, pp. 396-409, April 2008.

[10] X. Lin and N. B. Shroff. Joint rate control and scheduling in multihop wireless networks. *Proc. of 43rd IEEE Conf. on Decision and Control, Paradise Island, Bahamas*, Dec. 2004.

[11] L. Xiao, M. Johansson, and S. P. Boyd. Simultaneous routing and resource allocation via dual decomposition. *IEEE Transactions on Communications*, vol. 52, no. 7, pp. 1136-1144, July 2004.

[12] S. H. Low and D. E. Lapsley. Optimization flow control, i: Basic algorithm and convergence. *IEEE/ACM Transactions on Networking*, vol. 7 no. 6, pp. 861-875, Dec. 1999.

[13] C. C. Moallemi and B. Van Roy. Distributed optimization in adaptive networks. *Advances in Neural Information Processing Systems*, vol. 16, MIT Press, 2004.

[14] L. Jiang and J. Walrand. A distributed csma algorithm for throughput and utility maximization in wireless networks. *Proc. Allerton Conf. on Communication, Control, and Computing*, Sept. 2008.

[15] S. Rajagopalan and D. Shah. Reversible networks, distributed optimization, and network scheduling: What do they have in common? *Proc. Conf. on Information Sciences and Sytems (CISS)*, 2008.

[16] L. Jiang and J. Walrand. *Scheduling and Congestion Control for Wireless and Processing Networks*. Morgan & Claypool, 2010.

[17] A. Nayyar. *Sequential Decision Making in Decentralized Systems*. PhD thesis, University of Michigan, 2011.

[18] M. J. Neely. Distributed stochastic optimization via correlated scheduling. *ArXiv technical report, arXiv:1304.7727v2*, May 2013.

[19] 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, May 2012.