2018 IEEE 59th Annual Symposium on Foundations of Computer Science

Recharging Bandits

Nicole Immorlica
Microsoft Research
nicimm@microsoft.com

Robert Kleinberg
Cornell University
rdk@cs.cornell.edu

***Abstract*—We introduce a general model of bandit problems in which the expected payout of an arm is an increasing concave function of the time since it was last played. We first develop a PTAS for the underlying optimization problem of determining a reward-maximizing sequence of arm pulls. We then show how to use this PTAS in a learning setting to obtain sublinear regret.**

Absence makes the heart grow fonder.

I. Introduction

Sequential decision problems in unknown environments require decision makers to balance the twin objectives of learning about the environment, called *exploration*, and reaping the reward, called *exploitation*. The prototypical example of such a decision problem is that faced by a gambler who tries to maximize her winnings by sequentially choosing between a casino's slot machines. Slot machines are differentiated in their probability of payout, and the goal of the gambler is play the best machine indefinitely.

Multi-armed bandit algorithms provide a framework for navigating these exploration-exploitation tradeoffs. The average performance guarantee of a bandit algorithm is often expressed as a linear function of the reward of the best option. The multiplicative factor is called the *approximation*, and the additive factor is called the *regret*. Ideal bandit algorithms have optimal multiplicative factors of one and sublinear (in the length of the sequential process) regret. Such algorithms, when played indefinitely, approach an average payout equal to that of a sequential decision maker who always selects the optimal option.

Multi-armed bandits have numerous applications in domains such as providing restaurant recommendations to a diner with unknown tastes. Here each cuisine corresponds to a slot machine, and the reward upon recommending a restaurant is the diner's satisfaction, modeled as a random draw from the diner's unknown preference distribution for the restaurant's cuisine. The algorithm, once it develops sufficient confidence in the diner's tastes, maximizes reward by always recommending restaurants from the diner's favorite cuisine.

In some applications, it is questionable whether the optimal decision sequence always selects the same option. Instead, there may be some benefit to waiting as the payout of an option may improve as the option lies dormant. That is, options recharge over time. In the recommendation domain, for example, a diner always enjoys eating his favorite cuisine, but his desire for the cuisine grows the longer he goes without it. Similar issues arise in other domains as well, such as web crawling where the crawler must cycle among pages to note updates, education where learning is maximized by spaced repetition of multiple concepts, and law enforcement where cycling among potential targets detects and stops a greater amount of illicit activity than repeatedly patrolling a single target.

Motivated by these applications, we introduce a new bandit problem, called *Recharging Bandits*, which models the idea that rewards accrue to arms over time. In Recharging Bandits, each arm is parameterized by an unknown reward function, specifying its expected payout as a function of the delay since it was last played. We assume these rewards are increasing and concave (i.e., it is never harmful to insert extra pulls of an arm in a schedule). Each time step, the bandit algorithm pulls a single arm and receives a sample from the payout function, depending on the delay since the arm was last pulled. The goal is to find a sequence of arm pulls that maximizes the total reward in expectation. While our motivating applications certainly involve additional complications, the Recharging Bandits problem is a common fundamental element.

A key feature of our problem that makes it challenging, in contrast to existing bandit models, is that even if all the input parameters were known (so that there is no learning and no exploration-exploitation tradeoff) the underlying optimization problem is quite complex. As the following example shows, greedily selecting the arm with the largest expected payout in each time step, which achieves the known-parameter optimum in typical bandit models, is at best a constant approximation in our setting.

Example I.1. *Suppose there are two arms. The first arm has a payout of t if it was last pulled t days ago. The second has a payout of $1-\varepsilon$ for some $\varepsilon > 0$ whenever it is pulled. The greedy algorithm always pulls the first arm for an average payout of 1 per time step. Consider the following periodic schedule: for a large fixed T, pull the first arm on time steps equal to $0 \pmod T$, and the second arm on all other time steps. Then, in each period, this schedule gets a value of T from the first arm, and $(1-\varepsilon)(T-1)$ from the second arm,*

2575-8454/18/$31.00 ©2018 IEEE
DOI 10.1109/FOCS.2018.00037

for an average payout approaching $(2-\varepsilon)$ as T grows large which is nearly twice that of greedy.

A second example, analyzed in the full version, illustrates the complex set of interdependencies that arise when many recurrent schedules (one for each arm) must be interleaved on the same timeline without ever scheduling two arms at the same time.

Example I.2. *There are three arms with $H_1(t) = \min\{t, 2\}$, $H_2(t) = \min\{t, 3\}$, $H_3(t) = \min\{t, 6\}$. If it were possible, it would be most efficient to pull arm 1 at least once in every time window of length 2, arm 2 at least once in every time window of length 3, and arm 3 at least once in every time window of length 6, because then no arm would have a time step when it sits idly without storing up an additional unit of payoff, resulting in an average payout of 3 per time step. There is no obvious "packing obstruction" to designing such a schedule (because $1/2 + 1/3 + 1/6 = 1$) but there is a number-theoretic obstruction: arithmetic progressions with gap sizes 2 and 3 inevitably collide, because 2 and 3 are coprime. In fact, the maximum average payout achievable in this example is $\frac{17}{6}$.*

The bulk of our paper is devoted to designing algorithms that approximately solve this optimization problem, which we call the *Recurrent Scheduling problem*. We then leverage these optimization algorithms in our learning context to solve the Recharging Bandits problem.

Recurrent Scheduling: Our main optimization result is a PTAS for the Recurrent Scheduling problem. As explained below, Recurrent Scheduling is a special case of the Restless Bandit problem, a notoriously hard class of dynamic programming problems that generalizes the standard multi-armed bandit problem, but whose general case is PSPACE-hard to even appoximate [22]. To the best of our knowledge, our algorithm is the first PTAS for any restless bandit problem more general than the original multi-armed bandit problem.

Our PTAS divides arms into two groups: the *big* arms and the *small* arms. Intuitively, big arms are those that the optimal schedule pulls frequently, say at least an ε^2 fraction of time for some constant $\varepsilon > 0$, whereas small arms are those that the optimal schedule pulls infrequently, say at most an ε^2 fraction of time. First note if all arms are big, then there can be at most $1/\varepsilon^2$ of them and so the optimal periodic schedule can be found in polynomial time by brute force search.[1] Next, imagine all arms are small. In this case, we develop a concave programming relaxation for the scheduling problem. The variables of this program represent target frequencies for the arms. A naive rounding scheme selects, independently on each day, to play an arm with probability equal to its target frequency. While this is a $(1-\frac{1}{e})$-approximation in general, its performance restricted to small arms is insufficient for our purposes.

The pitfall of the independent rounding scheme is the large variance in delay between successive pulls of an arm. To counteract this, we develop a dependent rounding scheme which we call *interleaved rounding*. This scheme schedules arms in continuous time at intervals corresponding to their target frequencies. It then interleaves these arms by shifting each arm's schedule by a random offset. Finally, it produces a discrete-time schedule by pulling arms in the same order as the continuous-time schedule. To show that this scheme has sufficiently low variance in the delays experienced by the arms when pulled, we prove the delays are "more concentrated than" Poisson random variables. Formally, we show that in the convex stochastic order, the delay distributions are bounded above by Poisson random variables with the same expectation. As our objective function is a concave function of the delays, we can bound our approximation factor by studying the expected reward obtained by these Poisson random variables. Following this strategy, we prove that interleaved rounding is a $(1-\frac{1}{2e})$-approximation in general, and a $(1-\delta)$-approximation when the target frequencies of all arms are at most δ^2.

To get a PTAS for general settings, we use partial enumeration to find a good set of big arms, and a periodic schedule for pulling big arms that leaves sufficient gaps for small arms. We then fit small arms into the gaps using interleaved rounding of our concave relaxation. To make this work, we must overcome two key difficulties.

First, the gaps in the periodic schedule of big arms may not be evenly spaced, which interferes with the guarantee of the interleaved rounding scheme. To handle this, we select for each small arm just one congruence class (mod p) of eligible gaps, where $p = O_\varepsilon(1)$ is the period length of the big arms' schedule. We then bin-pack the small arms into these congruence classes and round each congruence class separately. This of course works only if small arms have frequencies less than ε^2/p whereas big arms have frequencies at least $1/p$. In other words, we must eliminate some set of arms with intermediate frequencies in order to drive a wedge between small and large frequencies, and we must argue these do not contribute much to the optimal value.

Second, we must argue there is a periodic schedule of big arms with period $p = O_\varepsilon(1)$ that both approximates the optimal value obtained from big arms and leaves sufficient gaps for small arms. Accordingly, we associate two parameters to the schedules: the value per time unit and the fraction of unallocated steps. Through a sequence of modifications to the optimal periodic schedule of all arms, we argue there is a length-p periodic schedule that obtains high value from big arms and a (potentially different) length-p periodic schedule that leaves sufficient gaps for small arms. By interleaving

[1]The optimal schedule may not be periodic; however periodic schedules with sufficient length are arbitrarily good approximations.

these, we can create a new length-$O(p)$ periodic schedule that obtains the convex combination of their parameters of interest.

Recharging Bandits: We design algorithms for Recharging Bandits by reducing to Recurrent Scheduling using the *optimism under uncertainty* principle. Algorithms designed according to this principle schedule arm pulls under optimistic assumptions about their payouts and then update their assumptions based on the results. The analysis argues that in each step, either the assumption was justified, in which case the reward is high (exploitation), or the assumption was overly optimistic, in which case the updated assumption is more realistic (exploration).

The main barrier in applying this principle to our reduction is that Recurrent Scheduling algorithms assume the payout functions are concave whereas the upper confidence bounds (UCBs) on the payout functions, defining the optimistic assumptions of the learning algorithm, might not be concave. Broadly speaking, we manage this by *ironing* the UCBs, i.e., using the concave envelope of the UCBs as our optimistic estimate of the payout function. If the Recurrent Scheduling algorithm returns a schedule which pulls arms at delays in the interior of an ironing interval, then the algorithm might get predictably low reward. In such time steps, the algorithm neither exploits (because the reward was low) nor explores/learns (because the reward was predictable), and therefore suffers regret. To overcome this, the Recharging Bandits algorithm perturbs the schedule returned by the Recurrent Scheduling algorithm, targeting a delay for each arm that is at an endpoint of an ironing interval. Because of the concavity of the underlying payout functions, an appropriate mixture of these endpoints either achieves an expected reward close to the target point on the ironed UCB curve (exploitation), or the estimated UCBs at the endpoints of the ironing interval are improved (exploration).

Using this approach in combination with our PTAS for the Recurrent Scheduling problem, we develop a learning algorithm for the Recharging Bandits problem which gets an average expected payout of

$$(1-\varepsilon)\mathrm{OPT} - O\left(\frac{n^2 \log n}{\varepsilon^{O(1/\varepsilon)}}\sqrt{\frac{\log T}{T}}\right),$$

where n is the number of arms, T is the number of time steps (length of the schedule), and $\varepsilon > 0$ governs the desired multiplicative appoximation. We note the multiplicative factor in this regret bound is the best we know how to achieve in polynomial time even for the easier Recurrent Scheduling problem, and the additive factor is the best achievable (in terms of its dependence on T) even for the standard two-armed bandit problem in which the expected payout functions are constant.

We now summarize some related work.

Restless bandits: Restless bandit problems [27] are a broad class of dynamic programming problems that generalize the multi-armed bandit problem by allowing the payout distributions of the arms to change while other arms are being pulled. As noted earlier, it is PSPACE-hard to even approximate restless bandits, even in special cases when state transitions are deterministic [22] as is the case in the Recurrent Scheduling problem. The first constant-factor approximation algorithm for a class of restless bandit problems was discovered by Guha and Munagala in 2007 [14] and generalized soon thereafter by Guha, Munagala, and Shi [15]. We are not aware of a PTAS for any non-trivial restless bandit problem prior to our result on Recurrent Scheduling.

The Recharging Bandits problem is a special case of Markov decision processes (MDPs) with deterministic transitions and uncertain rewards, which is in turn a special case of reinforcement learning. Regret minimizing algorithms for MDPs with deterministic transitions and stochastic rewards were developed by Ortner [20]; the more difficult setting of deterministic transitions and adversarial rewards was tackled by Arora et al. and Dekel et al. in [2, 12]. Generalizing in a different direction, Jaksch et al. [7, 18] obtained near-optimal regret bounds for reinforcement learning in MDPs with stochastic transitions and rewards. All of the aforementioned problems generalize Recharging Bandits, but do so in a way that produces an exponential (in the number of arms) blow-up in the size of the state space. Any of these algorithms may be applied to solve Recharging Bandits when the number of arms is constant, with polynomial bounds on the regret and running time. Finally, Ortner et al. [21] present an algorithm with $\tilde{O}(\sqrt{T})$ regret for *ε-structured MDPs*, a class of problems that includes restless bandits, assuming one has an oracle for solving the (generally computationally hard) problem of computing an optimal policy for the ε-structured MDP. Our Recharging Bandits algorithm pertains to a much narrower class of restless bandit problems, but eliminates the exponential-time optimal policy computation.

Periodic scheduling and low-discrepancy coloring: Our Recurrent Scheduling problem contributes to the literature on periodic scheduling of jobs. Our work differs from previously studied periodic scheduling problems (e.g. [23]) in that the jobs themselves do not have periodic release times (in effect, our model assumes a new job is released as soon as one of the same type completes) and it does not impose hard constraints on the time intervals between consecutive occurrences of a job; instead, our concave payout function in effect penalizes scheduling jobs at irregular intervals. To our knowledge, the only other prior work on periodic scheduling with a maximization objective is by Sgall et al. [24], a paper which designs constant-factor approximation algorithms for a scheduling problem with hard constraints on the minimum amount of idle time after jobs complete and before they can be scheduled again. Among the periodic scheduling

problems studied in prior work, the closest to ours is the *pinwheel problem* [10, 11, 13, 16], in which one is given an n-tuple $(v_1, \ldots, v_n)$ and must decide whether there exists an n-coloring of the integers such that every color i is used at least once in every interval of v_i consecutive numbers. This decision problem belongs to PSPACE, but questions of membership in NP, and of NP-hardness, remain open. Jacobs and Longo [17] have shown that the problem does not admit a pseudo-polynomial time algorithm unless SAT can be solved by a randomized algorithm with expected running time $n^{O(\log n \log\log n)}$. The *chairman assignment problem*, whose solution by Tijdeman in the 1970's [26] was independently rediscovered by Angel et al. [1] in 2009, poses a similar-looking problem about low-discrepancy colorings of the natural numbers, but with the crucial distinction that the discrepancy constraint is enforced only on intervals of the form $[1, k]$ rather than on all intervals.

Periodic scheduling problems in which multiple jobs may be scheduled at once, motivated by various problems in networking and multimedia distribution, were studied in [4, 5, 6]. In our PTAS when we pack "small arms" into the gaps of a p-periodic "big arm" schedule by assigning each small arm to a congruence class mod p, we are effectively treating the congruence classes as distinct "servers" that run in parallel, which builds a thematic link between our work and the *windows scheduling* problem [4, 5], but approximation guarantees for the algorithms in those works are not strong enough to be of use in designing our PTAS.

Optimal schedules for security games: In a paper that bears a similarity to ours in terms of both motivation and techniques, Kempe, Schulman, and Tamuz [19] formulate an optimal scheduling problem for a security game in which there are n targets and a defender commits to a (randomized) schedule for visiting targets, to minimize an attacker's utility. Their motivation is similar to our law enforcement motivation listed above, but unlike our model (and appropriately for the law enforcement application) theirs is explicitly game-theoretic. As in our analysis of LP rounding algorithms in §III, the focus of attention in their paper is on rounding a probability distribution over arms (targets, in their terminology) to a type of low-discrepancy sequence that they call a *K-quasi-regular sequence*. It seems possible that their construction of 2-quasi-regular random sequences could furnish an alternative rounding scheme that attains an approximation factor $\alpha > 1 - \frac{1}{2e}$ to the value of the concave relaxation (1). On the other hand, as shown in the full version, the integrality gap is bounded away from 1, so a PTAS for Recurrent Scheduling cannot be based on improved rounding schemes alone. A final point of connection between their paper and ours is their Lemma 4.3, which furnished the inspiration for our learning algorithm that rounds all but one arm to an extreme point of its "ironing interval."

II. Model

There is a set $\mathcal{N}$ of n arms. The expected payout of arm i, if it was last pulled t days ago, is $H_i(t)$. We assume $H_i(t)$ is weakly concave and weakly increasing, and that $H_i(0) = 0$.

Let $\mathcal{N}_+ = \mathcal{N} \cup \{\perp\}$, where $\perp$ is a special "null symbol". A schedule is a function $\iota : \mathbb{N} \to \mathcal{N}_+$, where $\iota(s) \in N$ denotes the arm that is pulled at time step s, and $\iota(s) = \perp$ indicates that no arm was pulled at time s. A schedule is *periodic*, with period T, if the sequence $\iota(1), \iota(2), \ldots$ satisfies $\iota(s + T) = \iota(s)$ for all $s \in \mathbb{N}$. It will often be convenient to work with an alternative representation of schedules as functions $\sigma : \mathcal{N} \times \mathbb{N} \to (\mathbb{N} \cup \{\infty\})$, where $\sigma(i, k)$ denotes the time step at which the k^{th} pull of arm i occurs, or ∞ if arm i is pulled strictly fewer than k times in the sequence. For notational convenience, we set $\sigma(i, 0) = 0$. In the sequel, we will not distinguish between a schedule ι and the function σ that represents it.

The value of a schedule is the limiting average of its value per time step. More formally, for any subset of time steps $S \subseteq \{1, \ldots, T\}$, define the value $v_{i,T}(S)$ of arm i to be its contribution if pulled at time steps S. That is, if $s_1, s_2, \ldots, s_k$ are the elements of S in increasing order, then, letting $s_0 = 0$ for convenience, $v_{i,T}(S) = \sum_{j=1}^{k} H_i(s_j - s_{j-1})$. Note there is a one-to-one correspondence between partitions $S_1, \ldots, S_n$ of $\{1, \ldots, T\}$ and the length-T prefix of schedules in which $\iota(s) = i$ if $s \in S_i$. Let $\sigma|_T$ be the length-T prefix of schedule σ and $S_1(\sigma|_T), \ldots, S_n(\sigma|_T)$ be the partition of its time steps to arms. Then the value σ is $v(\sigma) = \limsup_{T\to\infty} \frac{1}{T} \sum_{i=1}^{n} v_{i,T}(S_i(\sigma|_T))$.

The optimum solution to our problem is a schedule with maximum value. We overload notation and refer to both the optimal schedule and its value by OPT. A schedule is an α-approximation to OPT for $0 \le \alpha \le 1$ if its value is at least $\alpha \cdot \text{OPT}$. Throughout our paper, we will leverage the following useful properties of our setting, proved in the full version.

Lemma II.1. *The value $v_{i,T}(S)$ of an arm i is weakly monotone increasing and submodular.*

The following two lemmas describe conditions under which periodic or finite schedules approximate the value of the optimal schedule.

Lemma II.2. *Given $\epsilon > 0$, for all $T \ge n/\epsilon$, there is a length-T periodic schedule of value at least $(1 - \epsilon)$OPT.*

Lemma II.3. *For all $T > 3n$ there is a schedule σ^T whose length-T prefix value, $\frac{1}{T} \sum_{i=1}^{n} v_{i,T}(S_i(\sigma^T|_T))$, is at least $(1 - 3n/T)$OPT.*

First, we design optimization algorithms that compute approximately-optimal schedules when the reward functions $H_i(t)$ are known to the algorithm. A natural greedy approach would be to always pull the arm with the highest expected reward. There are multiple proofs that such an algorithm is

a $(1/2)$-approximation. Example I.1 shows that the greedy algorithm is, however, suboptimal. Instead, we use a concave relaxation with an interleaved rounding scheme to obtain a constant approximation in Section III, and then combine this with partial enumeration in Section IV to obtain a PTAS. Finally, in Section V, we design a learning algorithms with low regret when the reward functions $H_i(t)$ are unknown to the algorithm. All missing proofs are deferred to the full version.

III. Concave Relaxation

To approximate the optimal schedule, we first describe an upper bound. Let the rate of return function, $R_i(x)$, denote the maximum long-run payoff achievable by scheduling arm i in at most an x fraction of time steps, i.e., $R_i(x) = \limsup\{\frac{1}{T}\sum_{j=1}^{k} H_i(s_j - S_{j-1})\}$ where $T \in \mathbb{N}$, $k \leq x \cdot T$, $0 = s_0 < s_1 < \ldots < s_k \leq T$.
We argue the following concave program is an upper bound on the value of the optimal schedule. Variable x_i indicates the fraction of time spent pulling arm i. The constraint $\sum_{i\in\mathcal{N}} x_i \leq 1$ bounds the total fraction of allocated time steps. The main difficulty is showing that the objective is an upper bound on the value of the optimal schedule; this involves some careful manipulations of the limits. For a formal proof, see the full version.

Lemma III.1. *The following concave program is an upper bound on the value of an optimal schedule.*

$$\begin{array}{lll} \max & \sum_{i=1}^{n} R_i(x_i) & \\ \text{subject to} & \sum_i x_i \leq 1 & (1) \\ & x_i \geq 0 \qquad\qquad & \forall i. \end{array}$$

To see that (1) is efficiently solvable, in the full version we prove the following useful lemma showing that the functions R_i are piecewise-linear concave functions. The key step in the proof is to show that given a budget of T time steps, k of which may be used for pulling arm i, the way to maximize the combined payout obtained from arm i is to make the timing of those pulls as evenly spaced as possible, meaning that the set of gap sizes between consecutive pulls is a set of at most two (consecutive) integer values.

Lemma III.2. *The rate of return function $R_i(x)$ is piecewise linear on $(0, 1]$. It satisfies the formula $R_i(\frac{1}{k}) = \frac{1}{k}H_i(k)$ for every positive integer k, and its restriction to each interval $[\frac{1}{k+1}, \frac{1}{k}]$ is a linear function.*

Given this lemma, it is apparent that at every point $x \in (0, 1)$ the function $R_i(x)$ has a well-defined left derivative and right derivative, which are equal unless $x \in \{1/2, 1/3, \ldots\}$. We denote these two derivatives by $R_i'(x^-)$ and $R_i'(x^+)$, respectively. At the optimal point x^* it must be the case that there exists a threshold θ such that $R_i'(x_i^{*+}) \leq \theta$ for all i, while $R_j'(x_j^{*-}) \geq \theta$ for all j such that $x_j^* > 0$. Otherwise, we could improve the objective value of (1) by increasing x_i^* to $x_i^* + \delta$ and decreasing x_j^* to $x_j^* - \delta$ for small enough δ. Hence, to solve (1) it suffices to calculate the piecewise-constant functions $a_i(\theta) = \sup\{x \mid R_i'(x^-) \geq \theta\}$ and $b_i(\theta) = \inf\{x \mid R_i'(x^+) \leq \theta\}$, and then use binary search to find a value of θ such that $\sum_i a_i(\theta) \leq 1 \leq \sum_i b_i(\theta)$. The optimal solution (x_i^*) is then obtained by finding any point x in the set $\prod_{i\in\mathcal{N}}[a_i(\theta), b_i(\theta)]$ that satisfies the constraint $\sum_i x_i = 1$.

The algorithm for finding this optimal solution is efficient. Assuming that the input to the Recurrent Scheduling problem is specified by representing each function H_i as a piecewise-linear concave function on $\mathbb{R}_+$ with at most T pieces, the most time-consuming step in solving the concave program is the $O(nT\log(nT))$ step of sorting the set of slopes of segments of the piecewise linear functions $\{R_i\}_{i\in\mathcal{N}}$, before commencing the binary search. Thus, if $T = O(n/\varepsilon)$, which is sufficient for $(1-\varepsilon)$-approximation of the optimum concave program solution, the running time is $\tilde{O}(n^2/\varepsilon)$.

A rounding scheme takes a vector $\mathbf{x} = (x_i)$ that is feasible for concave program (1) and outputs a (potentially randomized) schedule σ. The rounding scheme is an α-approximation if its expected value $\mathbb{E}[v(\sigma)]$ is at least an α fraction of the objective function of (1), evaluated at $\mathbf{x}$. Proving such a bound requires: a) arguing about the interchange of an expectation and a limit superior in calculating the value of the rounding, and b) arguing about the expected delays for each pull of each arm. These two arguments are significantly simplified if the rounding scheme is *well-behaved*. Let $d_i^k = \sigma(i,k) - \sigma(i,k-1)$ be a random variable indicating the delay of arm i the k^{th} time it is pulled, let $(x_{it}^T(\sigma))_{i\in\mathcal{N},\, t\in[T]}$ denote the fraction of times arm i is pulled after a delay of at least t by schedule σ in the first T time steps, and let $\bar{x}_{it}^T = \mathbb{E}_\sigma\big[x_{it}^T(\sigma)\big]$.

Definition III.3. *A rounding scheme is* well-behaved *if for every feasible input vector $\mathbf{x}$ it produces a distribution over schedules σ such that:*

1) *The distributions of the variables d_i^k converge in total variation to a limit distribution as $k \to \infty$. Henceforth, we let d_i be a random variable with this limit distribution.*
2) *The expectation of the delay distribution satisfies $\mathbb{E}[d_i] = \frac{1}{x_i}$.*
3) *There is a sequence of constants $\varepsilon_1, \varepsilon_2, \ldots$ converging to 0, such that $\forall i \in \mathcal{N}, \forall 1 \leq t \leq T$,*

$$\frac{\bar{x}_{it}^T}{x_i} \geq (1 - \varepsilon_T)\Pr[d_i \geq t].$$

The distributions $\{d_i\}_{i\in\mathcal{N}}$ are called the limit distributions *of the rounding scheme.*

To develop intuition for well-behaved rounding schemes

and their approximation ratios, it is helpful to think about the case of well-behaved schemes that always allocate x_i fraction of time steps (in the limit) to pulling arm i, i.e. the sequence $x_{i1}^T(\sigma)$ almost surely converges to x_i. Since the average payout of each of these pulls (in the limit) tends to $\mathbb{E}[H_i(d_i)]$, the net contribution of arm i to the value of the rounded schedule, $v(\sigma)$, is $x_i\mathbb{E}[H_i(d_i)]$. Meanwhile, the contribution of arm i to the concave relaxation is $R_i(x_i)$. If we interpret the equation $R_i(1/k) = (1/k)H_i(k)$ to hold at every $k \in \mathbb{R}_+$ and not just at integer k (i.e., treat this equation as the definition of $H_i(k)$ for fractional k) then the contribution of arm i to the concave relaxation is expressed as $R_i(x_i) = x_iH_i(1/x_i) = x_iH(\mathbb{E}d_i)$. So, the difference between the value of arm i in the rounded schedule and in the concave relaxation is the difference between $x_i\mathbb{E}[H_i(d_i)]$ and $x_iH_i(\mathbb{E}d_i)$. Since H_i is concave, the latter quantity is larger. The approximation ratio of the well-behaved rounding scheme is related to the worst-case factor by which $H(\mathbb{E}d_i)$ can exceed $\mathbb{E}[H(d_i)]$ for a concave non-decreasing function H.

Proposition III.4. *For a well-behaved rounding scheme and feasible solution (x_i) to concave program* (1)*, if for all arms i and all concave, non-negative, non-decreasing functions H,*

$$\mathbb{E}[H(d_i)] \geq \alpha H(\mathbb{E}[d_i]), \tag{2}$$

then the rounding scheme is an α-approximation.

To the extent that the formal proof in the full version differs from the intuition sketched above, it is because of the need to deal with convergence issues, and also because the definition of a well-behaved rounding scheme is not strong enough to guarantee that $x_{i1}^T(\sigma)$ almost surely converges to x_i, and in cases when it does not there will be correlation between the limiting frequency of pulling arm i and the limiting distribution of delays.

A first attempt at rounding an optimal solution would be to choose, independently in each time step, to pull arm i with probability x_i^*. As we show in the full version, this simple rounding scheme is a $(1 - \frac{1}{e})$-approximation. However, the large variance in arm delays introduced by this scheme harms the approximation factor. Using dependent rounding, we can make sure that the delays between consecutive pulls of arm i are more concentrated around their expectation of $\frac{1}{x_i}$. Our scheme, called *interleaved rounding*, assigns a continuous clock time (positive real number) to each arm pull, and from that, creates a schedule as follows. Given vector (x_i) feasible for concave program (1),

1) Assume $\sum_{i\in\mathcal{N}} x_i = 1$ by creating a *dummy arm* 0, if necessary, and setting $x_0 = 1 - \sum_{i\neq 0} x_i$.
2) For each arm i, choose an $a_i \in [0, 1]$ uniformly at random and then label all reals of the form $\frac{a_i+k}{x_i}$ for $k = 1, 2, \ldots$ with arm i.
3) Let ι be the sequence of pulls obtained by reading off labels in increasing order.
4) If a dummy arm 0 was created in step 1, modify ι by changing all occurrences of arm 0 into gaps in the schedule, i.e. modifying each occurrence of $\iota(t) = 0$ to $\iota(t) = \perp$.

We show this rounding scheme achieves a $(1 - \delta)$-approximation if the optimal variables satisfy $x_i^* \leq \delta^2$ for all i, a key ingredient in our PTAS. We additionally show in the full version that this rounding scheme achieves a $(1 - \frac{1}{2e})$-approximation in general and is optimal for $n = 2$ arms.

Theorem III.5. *When all $x_i^* \leq \delta^2$, interleaved rounding is a $(1-\delta)$-approximation. In general, interleaved rounding is a $(1 - \frac{1}{2e})$-approximation.*

In the full version, we show interleaved rounding is well-behaved, and we describe the form of the limiting distributions $\{d_i\}_{i\in\mathcal{N}}$. Letting $\mathrm{frac}(z)$ denote the fractional part of a number z, i.e. the unique $\tilde{z} \in [0, 1)$ such that $z - \tilde{z}$ is an integer, the limiting distribution d_i is a sum of independent Bernoulli random variables such that each arm $j \in \mathcal{N}$ contributes $\lfloor x_j/x_i \rfloor$ summands that are deterministically equal to 1, and one summand with expected value $\mathrm{frac}(x_j/x_i)$.

Next we discuss the key ingredient in the approximation bound. To derive Theorem III.5 from Proposition III.4 one must prove lower bounds on the ratio between $\mathbb{E}[H(d_i)]$ and $H(\mathbb{E}d_i)$ when d_i is a sum of independent Bernoulli random variables and H is concave and non-decreasing. Our tool for doing so is the *convex stochastic ordering*, which allows us to reduce this to a question about Poisson random variables. The convex stochastic ordering is a partial ordering of random variables defined by specifying that $X \leq_{\mathrm{cx}} Y$ if and only if $\mathbb{E}\varphi(X) \leq \mathbb{E}\varphi(Y)$ for every convex function $\varphi : \mathbb{R} \to \mathbb{R}$ [8, 9, 25]. We use the following lemma, proved in the full version.

Lemma III.6. *If $X = \sum_{i=1}^n X_i$ is a sum of independent Bernoulli random variables and Y is a Poisson random variable such that $\mathbb{E}Y = \mathbb{E}X$ then $X + 1 \leq_{\mathrm{cx}} Y + 1$.*

This lemma, together with Proposition III.4, implies that it suffices to argue about the expected value of a concave, non-decreasing H on a Poisson random variable that approximates the true delay.

Lemma III.7. *If y is a Poisson random variable with expected value $(1/x_i) - 1$ and H is a concave, non-negative, non-decreasing function, then*

$$\mathbb{E}[H(d_i)] \geq \mathbb{E}[H(1 + y)]. \tag{3}$$

Proof: As d_i is a sum of independent Bernoulli random variables, at least one of which is deterministically equal to 1, Lemma III.6, implies $d_i \leq_{\mathrm{cx}} 1 + y$. Applying the definition

of the convex stochastic ordering to the convex function $\varphi = -H$ yields (3). ∎

The following lemma, proved in the full version, assembles facts about concave functions of Poisson random variables that are required to complete our proof.

Lemma III.8. *If H is non-negative, non-decreasing, and concave, and y is a Poisson random variable, then $\mathbb{E}[H(1+y)] \geq \left(1 - \frac{1}{2e}\right) H(1+\mathbb{E}y)$. If, furthermore, $\mathbb{E}[y] \geq \frac{1}{\delta^2}$, then $\mathbb{E}[H(1+y)] \geq (1-\delta)H(1+\mathbb{E}y)$.*

IV. Polynomial-Time Approximation Scheme

To construct a polynomial-time approximation scheme for the problem of computing an optimal schedule, we define a set of *structured schedules* and argue that there exists at least one structured schedule that closely approximates the optimum. Then we describe a polynomial-time algorithm for approximately optimizing over the set of structured schedules. A structured schedule partitions arms into a constant number of "big" arms that are played frequently, according to a periodic schedule with bounded period length. The remaining arms are scheduled into the gaps in the big-arm schedule at a much lower frequency.

Definition IV.1. *For a positive integer p and a real number $\delta > 0$, a (p, δ)-*structured schedule *is a partial schedule such that the arms can be partitioned into two sets B, S, called the "big arms" and "small arms", such that:*

1) *Big arms are played with period p: if $i(t) \in B$ then $i(t+p) = i(t)$.*
2) *Small arms are played with asymptotic frequency at most δ^2/p: if $i \in S$ then*

$$\frac{\delta^2}{p} \geq \limsup_{T\to\infty} \frac{1}{T} \cdot |\{t \mid 1 \leq t \leq T \text{ and } i(t) = i\}|.$$

Our PTAS is based on partial enumeration combined with rounding a concave program. Given a desired approximation ratio $1-\varepsilon$, where $1/\varepsilon$ is assumed, without loss of generality, to be an integer, it sets $\delta = \varepsilon/6$ and $p = \delta^{2-4/\delta}(1+\delta)^2$. The value of p is chosen to ensure that the set of (p,δ)-structured schedules includes at least one element whose value is a $(1-\delta)^4$-approximation to that of the optimium schedule. We show this is possible in the full version.

To approximately optimize over (p,δ)-structured schedules, our PTAS performs brute-force enumeration of partial schedules of period length p; the arms not appearing in the partial schedule are assumed to be small and are scheduled into the gaps in the partial schedule using a variant of the interleaved rounding technique introduced in §III. We will show that the outcome of the rounding step is a schedule whose value is at least $(1-\delta)^2$ times that of the optimal (p,δ)-structured schedule. Therefore, the overall approximation guarantee of our algorithm is $(1-\delta)^4 \cdot (1-\delta)^2 = (1-\delta)^6$, which is greater than $1-\varepsilon$, by our choice of δ.

Algorithm 1 Polynomial-time approximation scheme for Recurrent Scheduling

$\delta \leftarrow \varepsilon/6$
for $1 \leq p \leq \delta^{2-4/\delta}(1+\delta)^2$ **do**
  for all p-periodic partial schedules σ_p **do**
    $B \leftarrow \{$arms occurring at least once in $\sigma_p\}$
    $S \leftarrow \mathcal{N} \setminus B$
    $y \leftarrow$ fraction of steps allocated to gaps in σ_p
    Compute $g(y, \delta^2/p, S)$ by solving (4).
    Compute $\bar{v}(\sigma_p)$ using (5).
  end for
end for
$(\sigma', p) \leftarrow$ the periodic partial schedule that maximizes $\bar{v}(\sigma')$, and its period length
Define B, S, y for σ' as before. Let $\mathbf{x_1}$ denote the solution of (4) with $z = \delta^2/p$.
$G \leftarrow \{j \mid \sigma'(j) = \perp\}$
Partition S into $\{S_j\}_{j\in G}$ by packing bins of size 1 with items of sizes $p \cdot x_i$ $(i \in S)$.
for all $j \in G$ **do**
  $\iota_j \leftarrow$ schedule obtained by rounding $p \cdot x_i$ $(i \in S_j)$ using arithmetic progressions as in §III.
end for
for $t = 1, 2, \ldots$ **do**
  Write $t = j + ps$ with $j, s \in \mathbb{N}$, $j < p$.
  $\iota(t) = \begin{cases} \sigma'(j) & \text{if } \sigma'(j) \neq \perp \\ \iota_j(s) & \text{otherwise} \end{cases}$
end for

To analyze the rounding step, we use a concave-program-based upper bound on the value of structured schedules with a given period p for the big arms, motivated by the program in §III. The proof of the following lemma is in the full version.

Lemma IV.2. *For a set S of arms, a positive integer T, and real numbers $y \geq z > 0$, define $g(y, z, S)$ to be the value of the optimum of the concave program*

$$\max_{\mathbf{x_1}\in[0,z]^n, \sum_{i\in S} x_{i1} \leq y} \left\{ \sum_{i\in S} R_i^T(x_{i1}) \right\}. \tag{4}$$

The value $g(y,z,S)$ is an upper bound on the value of any T-periodic partial schedule σ that uses only arms in S, assigns at most zT steps per period to any single arm in S, and assigns at most yT steps per period to all arms in S, combined. Furthermore, for any fixed S, the function $g(y,z,S)$ is non-decreasing in y, non-decreasing in z, and jointly concave in (y,z).

Definition IV.3. *For a periodic partial schedule σ with period ℓ, let B denote the set of arms that are played at least once in σ, let S denote the set of arms that are never played, and let y denote the fraction of time steps that are*

allocated to gaps. The relaxed value *of σ is the quantity*

$$\bar{v}(\sigma) = \frac{1}{T}\sum_{i \in B} v_{i,\ell}(\Sigma_i) \; + \; g\left(y, \tfrac{\delta^2}{\ell}, S\right) \tag{5}$$

where Σ_i denotes the set $\{\sigma(i,k) \pmod{\ell} \mid k \in \mathbb{N}\}$ of time steps $\pmod{\ell}$ allocated to arm i.

Note that Lemma IV.2 implies that for every (p,δ)-structured schedule σ, if σ_B denotes the schedule obtained from σ by replacing every small arm with a gap, then $\bar{v}(\sigma_B) \geq v(\sigma)$. In the full version, we show how to fill the gaps in any p-periodic partial schedule σ' to produce a (p,δ)-structured schedule σ such that $v(\sigma) \geq (1-\delta)^2\bar{v}(\sigma')$. We are then able to show that our algorithm (summarized in pseudocode as Algorithm 1 below) computes a $(1-\varepsilon)$-approximation to the optimal schedule for any instance of Recurrent Scheduling. The running time of the algorithm is $(n+1)^{(1/\varepsilon)^{O(1/\varepsilon)}} \cdot (n^2/\varepsilon)\log(n/\varepsilon)$ due to the loop that iterates over p-periodic partial schedules, solving program 4 in each loop iteration in time $(n^2/\varepsilon)\log(n/\varepsilon)$ using the techniques of §III.

V. Learning Results

We now turn to designing algorithms for the Recharging Bandits problem itself; unlike in §§III-IV, we assume that the values $\{H_i(t) \mid i \in N, t \in \mathbb{N}\}$ are not known *a priori* but must be learned from samples. When arm i is pulled t steps after the preceding pull, the expected payout $H_i(t)$ is assumed to be a concave, non-decreasing function satisfying $H_i(t) - H_i(t-1) \leq 1$ as in earlier sections. We also need to specify our assumptions about the payout distributions. First, we assume, conditional on the sequence $\iota(1), \iota(2), \ldots$ of arms pulled, the payouts received in different time steps are mutually independent. Next, we assume if two consecutive pulls of arm i occur at times $\sigma(i,k-1) = s-t$ and $\sigma(i,k) = s$, then the payout $\pi_i(s)$ observed at time s has expected value $H_i(t)$, and the random variable $\pi_i(s)$ is supported on $[0,t]$.

We will design algorithms for Recharging Bandits by reducing to Recurrent Scheduling. Broadly speaking, our reduction is based on the "optimism under uncertainty" principle that has been successfully applied in the past to design low-regret algorithms for other stochastic multi-armed bandit problems, e.g. the UCB1 algorithm of [3]. However, our reduction must confront several difficulties.

1) Since the entities that our algorithm samples are arm-delay pairs rather than merely arms, the intent to obtain a sample corresponding to arm i and delay τ requires planning at least τ steps into the future; typical stochastic bandit algorithms do not do any future planning.
2) Recurrent scheduling algorithms may not target any particular delay τ with high frequency.[2] Furthermore, due to rounding error, randomized recurrent scheduling algorithms may target a delay that does not equal the realized delay. These issues can result in a small number of samples for any particular arm i and delay τ, hindering the ability to learn about $H_i(\tau)$.
3) Due to sampling error, our empirical estimates of the functions $H_i(\tau)$, as well as the upper confidence bounds derived from those estimates, may not be concave increasing functions of τ. This interferes with our ability to plug the upper confidence bounds into a black-box scheduling algorithm that is designed for concave increasing functions.

The first difficulty is relatively easy to deal with; we simply partition time into phases of pre-specified length ϕ, plan a schedule at the start of each phase, and revise our estimates of $H_i(\tau)$ only at the end of the phase.

The second and third difficulties turn out to be much trickier to deal with, and require us to introduce additional requirements for recurrent scheduling algorithms. To overcome the second difficulty, we require the scheduling algorithm to acquire useful training data for future phases at a sufficiently high rate. By grouping delays into "buckets" with geometrically-increasing widths, we limit the number of delays we need to learn about (the β-geometric property below). The representative delay in each bucket (say its endpoint) is the *hallucinated delay*, the one that we imagine the algorithm earns when its target delay according to its schedule falls in the bucket. We first require that when the algorithm hallucinates delay τ, the realized delay (which may differ from the target delay due to rounding error) is within a constant factor of τ at least a γ fraction of the time (Property 3 below). This allows us to guarantee we obtain sufficiently many samples of $H_i(\tau)$. However, we do not want to lose this γ in our approximation, and so we must attribute the reward from all samples, including those that fall outside their desired bucket, to the bucket itself. Thus we additionally require that the average reward of samples attributed to a bucket is at least a constant fraction of the reward of its hallucinated delay (Property 2 below).

To overcome the third difficulty, we require the scheduling algorithm to satisfy a stronger type of approximation guarantee, which says that even when its input functions $\bar{H}_i$ are potentially non-concave and non-increasing, it still outputs an α-approximation of the value of the optimal schedule, provided that the optimal schedule is evaluated according to concave increasing functions H_i that are pointwise less than or equal to $\bar{H}_i$ but may be unknown to the scheduling algorithm (Property 1 below).

In the remainder of this section, we formalize the required properties. We then sketch a reduction that transforms any

[2]The target delay is the one the algorithm would choose if there were no rounding errors or scheduling conflicts, e.g. the delay from the LP in the interleaved rounding algorithm.

algorithm satisfying these assumptions into a low-regret algorithm. Finally, we sketch an argument demonstrating that the PTAS of §IV can be modified to satisfy the required assumptions, and state the regret of the learning algorithm that results from applying our reduction to this modified PTAS. All details in this section are deferred to the full version.

An *augmented scheduling algorithm* receives, as input, a time interval I consisting of ϕ consuctive integers, and a tuple of functions $\{\bar{H}_i(\cdot) \mid i \in \mathcal{N}\}$. We will interpret $\bar{H}_i(\tau)$ as the upper confidence bound of the true reward $H_i(\tau)$. The algorithm outputs two functions $\iota : I \to \mathcal{N}_+$ and $\tau : I \to \mathbb{N}$ where $\iota(t)$ is interpreted as the arm scheduled at time t and $\tau(t)$ is interpreted as the "hallucinated delay" between time t and the previous time that the arm was scheduled. The purpose of the hallucinated delays is to group together samples with similar but non-identical delays into a single set to be used for estimating $H_i(\tau)$.

Let $d(t)$ denote the length of the time interval between t and the previous time step when the same arm was scheduled, or $d(t) = t$ if there is no such previous time step. Also let $A(i,\tau)$ be the set of time steps when i was scheduled after a hallucinated delay of τ, and $B(i,\tau)$ is the subset of $A(i,\tau)$ consisting of time steps when the hallucinated delay differed from the actual delay by a factor of at most β; these will be the only time steps in the learning algorithm's training data.

We say that an augmented scheduling algorithm is (α,β,γ)-strong if it satisfies the following three properties with respect to any tuple of concave non-decreasing functions $\{H_i(\cdot) \mid i \in \mathcal{N}\}$ satisfying $H_i(0) = 0\ \forall i$ and $H_i(\tau) \le \bar{H}_i(\tau)$ for all i,τ.

1) **[α-approximate optimality]** The hallucinated value of the schedule ι is at least α times the actual value of the optimal schedule ι^*.

$$\mathbb{E}\left[\sum_{t\in I} \bar{H}_{\iota(t)}(\tau(t))\right] \ge \alpha \cdot \sum_{t\in I} H_{\iota^*(t)}(d^*(t)). \quad (6)$$

2) **[β-approximate attribution]** The expected average reward accumulated by arm i in steps when its hallucinated delay was τ is at least $\beta H_i(\tau)$.

$$\mathbb{E}\left[\sum_{t\in A(i,\tau)} H_i(d(t))\right] \ge \beta \cdot \mathbb{E}|A(i,\tau)| \cdot H_i(\tau) \quad (7)$$

3) **[sampling rate at least γ]** In expectation, we use at least γ fraction of all time steps as training examples: for all i,τ,

$$\mathbb{E}|B(i,\tau)| \ge \gamma \cdot \mathbb{E}|A(i,\tau)|. \quad (8)$$

We say an augmented scheduling algorithm is β-geometric if the ratio between two distinct hallucinated delays is never in the interval (β, β^{-1}).

Suppose we are given an (α,β,γ)-strong augmented scheduling algorithm. To transform it into a low-regret learning algorithm for Recharging Bandits, we subdivide time into contiguous intervals (called *phases*) of length ϕ and run the scheduling algorithm once at the start of each phase s, to assign the time steps of that phase to arms. To do so, we must provide "hallucinated reward" functions $\bar{H}_i$ for each arm i. These rewards will be the upper-confidence bound of the arm/delay pair $U(i,\tau,s) = \frac{q}{k} + \tau\cdot\sqrt{\frac{4\ln(knT)}{k}}$ from the previous phase (where k is the total number of samples of the arm/delay pair in $\cup_{s'=1}^{s-1} B(i,\tau,s')$ and q is their sum). Note that this value is defined at the start of a phase and does not vary during the phase. At the end of the phase, we update the functions $\bar{H}_i$ using the data from $B(i,\tau,s)$. This completes the description of the reduction. Its analysis appears in the full version and results in the following theorem.

Theorem V.1. *For any (α,β,γ)-strong and β-geometric augmented scheduling algorithm, the reduction produces a Recharging Bandits algorithm whose expected average payout is at least*

$$\left(1-\frac{3n}{\phi}\right)\alpha\beta^3\mathrm{OPT} \;-\; \frac{4\phi}{\gamma}\left(\frac{n\ln\phi}{1-\beta}+\sqrt{\frac{2n}{1-\beta}}\right)f(T)$$

where $f(T) = \sqrt{\frac{2\ln(nT)}{T}}$.

To transform Algorithm 1, the PTAS from §IV, into an (α,β,γ)-strong and β-geometric augmented scheduling algorithm, recall that it distinguishes two types of arms, big and small. It performs brute-force enumeration of p-periodic partial schedules for the big arms while using the interleaved progression rounding technique of §III (Theorem III.5) to schedule the small arms. The analysis of brute-force enumeration for big arms i does not require $H_i(\tau)$ to be a concave function of τ, but the analysis of the LP rounding for small arms i relies on concavity. This constitutes the main difficulty in transforming Algorithm 1 into an (α,β,γ)-strong augmented scheduling algorithm. To overcome this difficulty, we will modify two parts of the PTAS.

First, instead of the function $g(y,\delta^2/p,S)$ we will substitute an alternative function $\hat{g}(y,\delta^2/p,S)$ that is computed by solving a concave program derived from an "ironed" version $\hat{R}_i$ of the (not necessarily concave) functions $\bar{R}_i(x) = x\bar{H}_i(1/x)$. The function $\hat{R}_i$ is defined to be the non-decreasing concave hull of $\bar{R}_i$, i.e. the pointwise minimum of all concave non-decreasing functions $F : [0,1] \to \mathbb{R}$ that satisfy $F(x) \ge \bar{R}_i(x)$ for all $x \in \mathcal{X}$. This pointwise minimum is itself a non-decreasing concave piecewise linear function. Since the modified PTAS is given access to the functions $\{\bar{H}_i\}$ but not $\{H_i\}$, it substitutes $\hat{R}_i$ in place of the role played by R_i in the original PTAS. In particular, analogous to the function $g(y,z,S)$ defined in Lemma IV.2,

function $\hat{g}(y, z, S)$ is defined as the solution of the optimization problem

$$\begin{array}{ll} \max & \sum_{i=1}^{n} \hat{R}_i(x_i) \\ \text{subject to} & y - z \geq \sum_{i \in S} x_i \\ & z \geq x_i \geq \frac{\varepsilon}{pn} \qquad \forall\, i \in S \end{array} \tag{9}$$

and we use the value $\hat{g}(y, \delta^2/p, S)$ in place of $g(y, \delta^2/p, S)$ in line 7 of the PTAS. For a p-periodic structured schedule σ we define $\hat{v}(\sigma)$ analogously to $\bar{v}(\sigma)$ but using $\hat{g}$ in place of g.

The reason for using $y - z$ rather than y on the left side of the sum constraint in (9) is to allow ourselves the freedom to adjust any single coordinate of the feasible solution to any value in the range $[0, z]$, without violating the constraint $y \geq \sum_{i \in S} x_i$. Such adjustments are important in our modified PTAS for a reason to be explained shortly. The reason we require $x_i \geq \frac{\varepsilon}{pn}$ for each $i \in S$ is to ensure that each small arm is sampled at least $1/\varepsilon$ times during a phase, a fact which is needed in the proof of the β-approximate attribution property.

The second modification of the PTAS concerns the way that small arms are scheduled when rounding the solution to the concave program (9). Recall the structure of the set of optimal solutions to (9) as derived in §III. Each function $\hat{R}_i(x)$ is a piecewise-linear concave non-decreasing function of $x \in (0, z]$, hence $(0, z]$ is partitioned into a finite number of subintervals on which $\hat{R}_i$ has constant slope. We will call these *ironing intervals*. For every $\theta \geq 0$ we can define $a_i(\theta), b_i(\theta)$ to be the endpoints of the interval on which $\hat{R}_i$ has slope θ, if such an interval exists, and otherwise $a_i(\theta) = b_i(\theta)$ is the rightmost endpoint of a subinterval of $(0, z]$ on which $\hat{R}_i$ has slope greater than θ. For any θ such that $\sum_{i \in S} a_i(\theta) \leq y - z \leq \sum_{i \in S} b_i(\theta)$ the set of optimal solutions of (9) is the subset of $\prod_{i \in S} [a_i(\theta), b_i(\theta)]$ consisting of points x such that $\sum_{i \in S} x_i = y - z$; denote this set by P.

In our augmented scheduling algorithm, the small arms will be scheduled by choosing $x^* = (x_i^*)_{i \in S}$ to be an extreme point of P, randomly modifying x^* to obtain a slightly different vector x', and then applying the dependent rounding scheme of §III to x'. Our use of an extreme point of P is inspired by, and analogous to, Lemma 4.3 of [19], although our application doesn't necessitate using a distribution over extreme points, as theirs does. Notice that the extreme points x^* of P have the property that x_i^* is an endpoint of the interval $[a_i(\theta), b_i(\theta)]$ for all but at most one value of i. Indeed, if $x \in P$ has two coordinates in the interior of their respective intervals, then we can vary x while remaining within P, by increasing either of these coordinates by a small amount while decreasing the other by the same amount. This defines a line segment contained in P having x in its interior, which demonstrates that x cannot be an extreme point.

Let $s \in S$ be the unique arm such that $a_s(\theta) < x_s^* < b_s(\theta)$, or if there is no such arm then choose $s \in S$ arbitrarily. In the sequel we will refer to this s as the *special arm*. To understand why special arms are problematic for our algorithm, and to motivate the special treatment we will give them in our rounding scheme, note that there are two sources of error that arise when we use $\hat{R}_s(x)$ to bound $R_s(x)$ from above. The first is the difference between $\bar{R}_s$ and R_s, which is due to sampling errors and confidence radii, both of which tend to decrease as we draw more samples. The other is the difference between $\hat{R}_s$ and $\bar{R}_s$, which arises from "ironing out" the non-concavity of $\bar{R}_s$. These differences tend to arise from overestimating the value of $\bar{R}_s$ at the endpoints of an ironing interval, and the only way to correct them is to obtain more samples at the endpoints. Thus, since x_s^* may belong to the interior of an interval $[a_s(\theta), b_s(\theta)]$, special care must be taken to obtain samples that allow us to narrow the confidence bounds corresponding to the endpoints of that interval.

To deal with this issue, we adjust the vector x^* by replacing its s^{th} coordinate with a randomly sampled endpoint of its ironing interval. Specifically, let $q \in [0, 1]$ be the solution to

$$x_s^* = qa + (1 - q)b. \tag{10}$$

Sample a random vector x' defined by setting $x'_i = x_i^*$ when $i \neq s$ while setting x'_s equal to a with probability q, and to b with probability $1 - q$. Note that

$$\sum_{i \in S} x'_i = x'_s + \sum_{i \in S \setminus \{s\}} x_i^* \leq z + (y - z) = y$$

hence the vector x' is feasible for the concave program (4) used in our original PTAS. The remainder of the modified PTAS is the same as the original one presented in Algorithm 1: we partition the small arms into sets S_j by greedy bin packing, then apply interleaved arithmetic progression rounding to schedule the arms in S_j into the time steps in the congruence class $j \pmod p$.

To conclude the description of our augmented scheduling algorithm we must specify how the values $\tau(t)$ are defined. If a big arm is pulled at time t, we set $\tau(t)$ to be the smallest element of $\mathcal{L}$ that is greater than or equal to $d(t)$. If small arm i is pulled at time t, we set $\tau(t) = 1/x'_i$.

In the full version, we show this modified PTAS is (α, β, γ)-strong and β-geometric for $\alpha = \beta = 1 - O(\varepsilon)$ and $\gamma = 3/4$, implying an expected average payout of

$$(1 - \varepsilon)\text{OPT} - O\left(\frac{n^2 \log n}{\varepsilon^{O(1/\varepsilon)}} \sqrt{\frac{\log T}{T}}\right)$$

by Theorem V.1.

VI. Acknowledgments

We thank Boaz Barak, Anna Karlin, David Kempe, Madhu Sudan, Karthik Sridharan, Vasilis Syrgkanis, Milind Tambe, and Omer Tamuz for useful discussions. Robert Kleinberg gratefully acknowledges the support of Microsoft Research New England and of NSF grant CCF-1512964.

References

[1] Omer Angel, Alexander E Holroyd, James B Martin, and James Propp. Discrete low-discrepancy sequences. *arXiv preprint arXiv:0910.1077*, 2009.

[2] Raman Arora, Ofer Dekel, and Ambuj Tewari. Deterministic MDPs with adversarial rewards and bandit feedback. In *Proc. 28th UAI*, pages 93–101, 2012.

[3] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. Finite-time analysis of the multi-armed bandit problem. *Machine Learning*, 47:235–256, 2002.

[4] Amotz Bar-Noy and Richard E Ladner. Windows scheduling problems for broadcast systems. *SIAM Journal on Computing*, 32(4):1091–1113, 2003.

[5] Amotz Bar-Noy, Richard E Ladner, and Tami Tamir. Windows scheduling as a restricted version of bin packing. *ACM Transactions on Algorithms (TALG)*, 3(3):28, 2007.

[6] Amotz Bar-Noy, Alain Mayer, Baruch Schieber, and Madhu Sudan. Guaranteeing fair service to persistent dependent tasks. *SIAM journal on Computing*, 27(4):1168–1189, 1998.

[7] Peter L Bartlett and Ambuj Tewari. Regal: A regularization based algorithm for reinforcement learning in weakly communicating MDPs. In *Proc. 25th UAI*, pages 35–42. AUAI Press, 2009.

[8] David Blackwell. Comparison of experiments. In *Proc. 2nd Berkeley Symposium on Mathematical Statistics and Probability*, pages 93–102, Berkeley, Calif., 1951. University of California Press.

[9] David Blackwell. Equivalent comparisons of experiments. *Ann. Math. Stat.*, 24(2):265–272, 1953.

[10] Mee Yee Chan and Francis Chin. Schedulers for larger classes of pinwheel instances. *Algorithmica*, 9(5):425–462, 1993.

[11] Mee Yee Chan and Francis Y. L. Chin. General schedulers for the pinwheel problem based on double-integer reduction. *IEEE Transactions on Computers*, 41(6):755–768, 1992.

[12] Ofer Dekel and Elad Hazan. Better rates for any adversarial deterministic MDP. In *ICML*, pages 675–683, 2013.

[13] Peter C Fishburn and JC Lagarias. Pinwheel scheduling: Achievable densities. *Algorithmica*, 34(1):14–38, 2002.

[14] Sudipto Guha and Kamesh Munagala. Approximation algorithms for partial-information based stochastic control with Markovian rewards. In *Proc. 48th FOCS*, pages 483–493. IEEE, 2007.

[15] Sudipto Guha, Kamesh Munagala, and Peng Shi. Approximation algorithms for restless bandit problems. *Journal of the ACM (JACM)*, 58(1):3, 2010.

[16] Robert Holte, Al Mok, Louis Rosier, Igor Tulchinsky, and Donald Varvel. The pinwheel: A real-time scheduling problem. In *System Sciences, 1989. Vol. II: Software Track, Proceedings of the Twenty-Second Annual Hawaii International Conference on*, volume 2, pages 693–702. IEEE, 1989.

[17] Tobias Jacobs and Salvatore Longo. A new perspective on the windows scheduling problem. *CoRR*, abs/1410.7237, 2014.

[18] Thomas Jaksch, Ronald Ortner, and Peter Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

[19] David Kempe, Leonard J Schulman, and Omer Tamuz. Quasi-regular sequences and optimal schedules for security games. In *Proc. 29th SODA*, pages 1625–1644. SIAM, 2018.

[20] Ronald Ortner. Online regret bounds for markov decision processes with deterministic transitions. *Theor. Comput. Sci.*, 411(29-30):2684–2695, 2010.

[21] Ronald Ortner, Daniil Ryabko, Peter Auer, and Rémi Munos. Regret bounds for restless markov bandits. In *International Conf. on Algorithmic Learning Theory*, pages 214–228. Springer, 2012.

[22] Christos H Papadimitriou and John N Tsitsiklis. The complexity of optimal queuing network control. *Mathematics of Operations Research*, 24(2):293–305, 1999.

[23] Thomas Rothvoss. *On the computational complexity of periodic scheduling*. PhD thesis, EPFL, Lausanne, 2009.

[24] Jiří Sgall, Hadas Shachnai, and Tami Tamir. Periodic scheduling with obligatory vacations. *Theor. Comput. Sci.*, 410(47-49):5112–5121, 2009.

[25] M Shaked and J Shantikumar. Stochastic orderings, 2007.

[26] Robert Tijdeman. The chairman assignment problem. *Discrete Mathematics*, 32(3):323–330, 1980.

[27] Peter Whittle. Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, 25(A):287–298, 1988.