Delayed Reward Bernoulli Bandits: Optimal Policy and Predictive Meta-Algorithm PARDI

Sebastian Pilarski, *Graduate Student Member, IEEE*, Slawomir Pilarski, *Member, IEEE*, and Dániel Varró, *Member, IEEE*

***Abstract*—Bernoulli multi-armed bandits are a reinforcement learning model used to optimize the sequences of decisions with binary outcomes. Well-known bandit algorithms, including the optimal policy, assume that before a decision is made the outcomes of previous decisions are known. This assumption is often not satisfied in real-life scenarios. As demonstrated in this article, if decision outcomes are affected by delays, the performance of existing algorithms can be severely affected. We present the first practically applicable method to compute statistically optimal decisions in the presence of outcome delays. Our method has a predictive component abstracted out into a meta-algorithm, predictive algorithm reducing delay impact (PARDI), which significantly reduces the impact of delays on commonly used algorithms. We demonstrate empirically that PARDI-enhanced Whittle index is nearly optimal for a wide range of Bernoulli bandit parameters and delays. In a wide spectrum of experiments, it performed better than any other suboptimal algorithm, e.g., UCB1-tuned and Thompson sampling. PARDI-enhanced Whittle index can be used when computational requirements of the optimal policy are too high.**

***Impact Statement*— Bernoulli multi-armed bandit algorithms are used to optimize sequential binary decisions. Oftentimes, decisions must be made without knowing the results of some previous decisions, e.g., in clinical trials where finding out treatment outcomes takes time. Well-known bandit algorithms are ill-equipped to deal with still unknown (delayed) decision results, which may translate into significant losses, e.g., the number of unsuccessfully treated patients. We present the first method of determining the optimal strategy for these type of situations and a meta-algorithm PARDI that drastically improves the quality of decisions by well-known algorithms—lowers regret by up to 3×. This is achieved by a 6× reduction in excess regret caused by delay. By addressing delays, this work can improve the quality of decisions in various applications. It opens new applications of Bernoulli bandits.**

***Index Terms*—Clinical trials, delayed feedback, multi-armed Bernoulli bandits, optimal policy, partially observable Markov decision process, Thompson sampling (TS), upper confidence bound, Whittle index (WI).**

Manuscript received May 2, 2021; revised July 7, 2021 and August 19, 2021; accepted September 18, 2021. Date of publication October 20, 2021; date of current version March 24, 2022. This work was supported in part by Versyn Inc. and in part by NSERC Discovery under Grant RGPIN-2016-04573. This paper was recommended for publication by Associate Editor Christian Wagner upon evaluation of the reviewers' comments. *(Corresponding author: Sebastian Pilarski.)*

Sebastian Pilarski and Dániel Varró are with the Department of Electrical and Computer Engineering, McGill University, Montreal, QC H3A 0G4, Canada (e-mail: sebastian.pilarski@mail.mcgill.ca; daniel.varro@mcgill.ca).

Slawomir Pilarski is with Versyn Inc., Vancouver, WA 98660 USA (e-mail: slawomir@versyn.com).

Digital Object Identifier 10.1109/TAI.2021.3117743

I. Introduction

MAKING choices is an integral part of everyday life. In most situations, the outcome of a decision is uncertain. If a series of decisions has to be made, it may be possible to optimize their outcomes using some probabilistic model and an algorithm that learns from past decisions.

Multi-Armed Bandits: A probabilistic model used in reinforcement learning is referred to as a *multi-armed bandit* [1]. It assumes that decisions are sequential, and at each time, one of a finite number of options is selected. Its name reflects the quandary of a casino gambler, or a *player*, who attempts to maximize his total winnings, or a *cumulative reward*, when facing a row of slot machines called 1-arm bandits. The model assumes that each arm, when pulled, produces a random reward according to its own probability distribution, which is unknown to the player.

Bernoulli Multi-Armed Bandits: A *Bernoulli bandit* [2] is a multi-armed bandit used to model processes where the outcome of a decision is strictly binary: success/failure, yes/no, or 1/0; each arm is characterized by its own probability of success. Bernoulli bandits can be used, for example, in clinical trials [2]–[8], portfolio management [9], and A/B testing (e.g., news headline selection and click feedback) [10]–[12]. A survey of practical applications of multi-armed bandits, including dynamic pricing and telecommunications, can be found in [13].

Algorithm Assumptions: A number of bandit algorithms have been proposed. They differ in assumptions and performances [8], [14], [15]. While some algorithms are general, others were designed specifically for Bernoulli bandits [2], [4], [16]. Some algorithms assume that the game goes forever. Some assume a finite *time horizon*, i.e., a situation where the total number of arm pulls is finite and predetermined.

Fundamental bandit algorithms were developed under the assumption that the rewards are *immediate*, i.e., known to the algorithm at the time of subsequent decision. This may be a serious limitation in practical applications [4], [14] and consequently it gets an increasing attention in research [17].

Delayed Rewards: If a reward associated with an arm pull is unknown until d subsequent decisions have been taken, we say that the reward has a *delay* of d decisions. Delayed rewards are also referred to as *delayed feedback* [14]. In this article, we often use the term **delay** as a synonym for *delayed rewards*. We also use the term **no-delay** meaning *immediate rewards*. All applications discussed earlier can operate in environments where rewards are subject to delay.

A. Related Work and Motivation

There exists a wealth of published work on multi-armed bandits under a variety of assumptions [8], [14], [15], [18]–[20] with some work focusing exclusively on Bernoulli bandits [2]–[4], [6], [16], [21]–[23]. However, as stressed in [14], few articles have been published on any variation of bandits with delayed rewards. It is well known that an assumption of no delay is often not applicable in practice [3], [4], [14].

A clear example where delay is most-often inevitable is in medical trials. Kuleshov and Precup [8] examined medical logs where the response of a patient to a treatment was unavailable when, on average, 24 subsequent patients were given their treatments. As shown in Section III, this delay can deteriorate algorithm performance significantly—up to $4\times$.

Effect of delayed feedback: Liu *et al.* [17] provided a survey and empirical evaluation of several basic algorithms and a novel adaptive greedy algorithm in the context of both stationary and nonstationary bandits with delay. A meta-algorithm for delays is presented and upper bounds on the impact of feedback delay is provided in [24]. However, Joulani *et al.* [24] did not performed any empirical evaluation. Pike-Burke *et al.* [25] developed an algorithm for bandits with delayed, aggregated, anonymous feedback, meaning that a reward received at a given time step can be the sum of multiple rewards from multiple arm pulls. Interestingly, their algorithm empirically matched the worst case regret bound in [24] in a much more complex setting than the nonaggregated, nonanonymous assumption in [24]. Tyo *et al.* [26] presented a new algorithm in the aggregated, anonymous feedback setting, which appears to determine the best arm faster than the algorithm in [25].

The aforementioned works [17], [8], and [24] assumed that bandit algorithms make decisions solely on revealed rewards, while [25] and[26] used additional knowledge of expected delays and delay bounds.

Algorithms for delayed feedback: There is a growing interest in designing algorithms that take into account delayed feedback. Grover *et al.* [27] analyzed multi-armed bandits that provide partial (contextual) feedback in the context of several real-world examples. A version of UCB for delays in contextual bandits is studied in [28]. A delayed exponential-based algorithm is proposed in [29] for adversary multi-armed bandits applied to fog computing offloading, which yielded performance near the nondelayed feedback setting. Ito *et al.* [30] presented an algorithm for online linear optimization with delayed bandit feedback for fixed constant delays and provided proofs on bounds.

Problem statement: Practically feasible methods to compute the optimal policy in the context of Bernoulli bandits for immediate rewards were recently proposed in [16], where the level of suboptimality of well-known algorithms was gauged. However, to the best of our knowledge, no computation of the optimal policy under delays has ever been published or proposed. The suboptimality of performance of well-known algorithms dealing with delays has never been assessed.[1] Also, no algorithmic solutions have been proposed to reduce the delay impact on common bandit algorithms under the general Bernoulli bandit model, i.e., without any additional assumptions.

[1]Known bounds on performance [14], [24] are too loose in practice [8].

B. Objectives, Contributions, and Significance

Objectives: The main goal of this article is to present practical methodologies for computing the optimal policy under any delay and reducing the negative impact of delay on well-known bandit algorithms. The article aims to showcase the significant negative impact of delay as well as the extent to which this impact can be reduced.

Contributions:

1) *A novel and first practical method to compute the optimal policy under any delay* based on the no-delay optimal policy [16] and classic probability theory (see Section IV).
2) *Computation and empirical evaluation of the optimal policy under delay* to provide performance limits for Bernoulli bandits under various delays and arm priors (see Section IV).
3) *Predictive algorithm reducing delay impact (PARDI) is a predictive meta-algorithm for reducing negative effects of delay* on suboptimal algorithms (see Section V).
4) *Evaluation of PARDI-enhanced common algorithms* for a range of parameters, including delays (see Section VI).
5) *Demonstrating that PARDI-enhanced Whittle index (WI) provides nearly optimal regret results* (see Section VI).

Significance: By extending efficient computation methods of the optimal policy to delayed-reward Bernoulli bandits, this article opens it to new applications where delays were an obstacle. For example, the optimal policy under delay can now be deployed in clinical trials where treatment results were available only after 24 subsequent decisions on average.

This article provides an empirical study of the best achievable regret for Bernoulli bandits under delay, allowing an assessment of suboptimality of various algorithms. Such an empirical assessment gives more information about algorithms' performance than a loose theoretical bound [8].

The predictive meta-algorithm, PARDI, can easily improve any existing Bernoulli bandit application where suboptimal algorithm suffers performance loss due to delays.

Finally, this article demonstrates that in situations where the optimal policy under delay is too expensive to deploy, e.g., because of a large number of arms, the Whittle index (WI) may be an excellent choice.

II. Preliminaries

A. Definitions and Notation

In this article, we use common notions in probability theory, such as *expected value*, *(arithmetic) mean*, *standard deviation*, and *variance* [31]. Much of this article deals with *cumulative* rewards, which is the arithmetic sum of rewards over a set period of time. *Cumulative* is applied to several random variables. We use the following standard notation (see [16]).

$\mathbb{P}[A]$ is the probability of an event A.
$\mathbb{E}[X]$ denotes the expected value of a random variable X.
σ denotes standard deviation of a random variable.
σ^2 denotes variance of a random variable.
$\hat{\sigma}$ denotes deviation of observed values.

$\mathbb{Z}$ is the set of integers.
$\text{argmax}_{i=1,\ldots,k}(g(i))$ selects i with max value of $g(i)$.
$\text{argmin}_{i=1,\ldots,k}(g(i))$ selects i with min value of $g(i)$.

The *Bernoulli multi-armed bandit*, BMAB(k, H), has k arms and time horizon H. In this single player game, the player (or algorithm) chooses one of k arms at each time (step), $0 \leq t < H$, and receives binary reward 1 (success) or 0 (failure) after some delay d. Each arm has its success probability drawn from some *prior* distribution at the start of each game.

When presenting or discussing bandit algorithms, we use the following symbols.

k is the number of arms.
H is the time horizon; number of pulls in a single game.
t is time, $0 \leq t \leq H$.
n_i is the number of times arm i was pulled.
s_i is the number of times arm i produced a success.
f_i is the number of times arm i produced a failure.
μ_i is the expected value of reward for arm i.
$\hat{\mu}_i$ is the observed mean reward for arm i.

Note that μ_i, $\hat{\mu}_i$, and n_i apply to a single experiment, i.e., drawing arm probabilities and playing one *game* that consists of a sequence of single arm pulls.

Also, note that for a single game, at any time t, $\Sigma_{i=1}^{k} n_i = t$. In a single game, $n_i = s_i + f_i$ is a function of t, and $n_i(t)$ denotes the number of times arm i has been pulled by time t.

Best arm: When analyzing the performance of an algorithm, it is convenient to have a symbol for the arm with the highest probability of success. It is called the *best arm*.

best_arm $= \text{argmax}_{i=1,\ldots,k}(\mu_i)$.
Symbols associated with the best arm are labeled with $*$.
$\mu^* = \max_{i=1,\ldots,k}(\mu_i)$ is best expected reward.
$\hat{\mu}^* = \max_{i=1,\ldots,k}(\hat{\mu}_i)$ is best observed mean reward.

Note that in a single game, μ^* is a constant, while $\hat{\mu}_i^*$ is a random variable.

Computed over multiple games, $\mathbb{E}[\mu^*]$ is an obvious upper bound on the mean reward of any player's algorithm at any time t. $\mathbb{E}[\mu^*]$ is a function of the distribution of arm success probabilities and the number of arms k.

Regret: Whenever the player pulls an arm with $\mu < \mu^*$, he or she experiences a loss of opportunity [5]. Such a potential loss is called *regret* and is measured by the difference between best arm's success rate and the selected arm success rate [14]. Formally, regret$(t) = \mathbb{E}[\mu^* - \mu(t)]$, where $\mu(t)$ represents the expected reward of the arm selected by the player at time t in a single game, and μ^* represents a constant in a single game, but a random variable when multiple games are run.

Minimization of cumulative regret is equivalent to maximizing cumulative reward.

In our simulation experiments, each simulation run starts with assigning each arm with its success probability. This probability is a random variable drawn from a probability distribution, a *prior*. In all our simulations, we use the *Beta distribution* [32]. The probability density function of the Beta distribution is expressed by the following equation: $\frac{x^{\alpha-1}(1-x)^{\beta-1}}{\int_0^1 y^{\alpha-1}(1-y)^{\beta-1}dy}$. Fig. 1 shows the plots of the Beta distribution for selected values of parameters α and β.

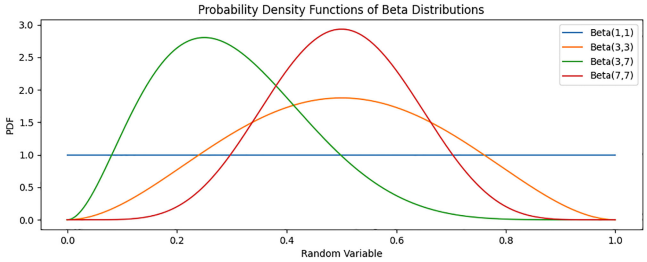

Fig. 1. Examples of the Beta distribution [16].

Property 1 (Conjugate Prior): Pulling an arm changes its distribution from Beta(α, β) to Beta$(\alpha + 1, \beta)$ if the outcome is a success, or to Beta$(\alpha, \beta + 1)$ if the outcome is a failure [32].

Property 1 and Equation (1) play a vital role in some algorithms discussed in this article

$$\mathbb{E}\left(\text{Beta}(\alpha, \beta)\right) = \frac{\alpha}{\alpha + \beta}. \tag{1}$$

B. Suboptimal Algorithms

To study the impact of delays, we selected three different algorithms from those analyzed in [16]. Thompson sampling (TS) and UCB1-tuned (UCBT) have been widely studied and recognized for their effectiveness. Under the assumption of no-delay, WI was shown to be nearly optimal.

- **Thompson Sampling (TS)** [33]–[35]
  This is a randomized algorithm presented here in the context of Bernoulli bandits. In essence, it generates random numbers according to each arms' Beta distribution, and picks the arm with the largest random number.

$$\text{choose}(t) = \underset{i=1,\ldots,k}{\text{argmax}} \left(\text{random}\left(\text{Beta}\left(\alpha_i(t), \beta_i(t)\right)\right)\right).$$

- **UCB1-Tuned (UCBT)** [14], [36]
  The UCBT algorithm is an improved version of the also well-known UCB1.

$$\text{choose}(t) = \underset{i=1,\ldots,k}{\text{argmax}} \left(\hat{\mu}_i(t) + c\sqrt{\frac{\ln t}{n_i(t)} \min\left(\frac{1}{4}, V_i(t)\right)}\right)$$

$$V_i(t) = \hat{\sigma}_i^2(t) + \sqrt{\frac{2 \ln t}{n_i(t)}}$$

  where $\hat{\sigma}_i^2$ is the variance of the success rate, and c is a constant. The constant c controls the degree of exploration [16]. In this article, we use $c = 1$ for all experiments—the common default value used in literature.
- **Whittle Index (WI)** [3], [4], [20], [37] **(See Appendix A)**
  WI is a modification of Gittins index [16], [38], which was a milestone in developing Bernoulli bandit algorithms [3], [4], [38], [39]. It is modified for finite time horizons

$$\text{choose}(t) = \underset{i=1,\ldots,k}{\text{argmax}} \left(W_I(s_i(t) + \alpha_i, f_i(t) + \beta_i, H - t)\right)$$

  where W_I is the WI, H is the time horizon, s_i and f_i are the numbers of successes and failures observed for arm

i, and Beta(α_i, β_i) is the distribution from which success probability of arm i was drawn.

It was shown empirically in [16] that WI offers nearly optimal regret for a variety of numbers of arms and their priors regardless of the time horizon.

C. Optimal Policy

In the context of Bernoulli bandits, there exists a statistically optimal deterministic policy. It can be computed for a given number of arms and their respective Beta priors.

- **Optimal Policy (OPT)** [4], [5], [15], [16]

$$\text{choose}(t) = O_{P_H}\big((s_1(t), f_1(t)), \ldots, (s_k(t), f_k(t))\big)$$

where O_{P_H} is the optimal policy for time horizon H and a set of Beta priors—one for each arm.

Optimal Policy Principle: For any configuration of successes and failures on all arms, the optimal policy chooses the arm with best expected reward until the end of the game.

This principle together with Property 1 and Equation (1) leads to the following equations [16]. They compute *best expected value* O_{V_H} and the *optimal policy* O_{P_H} for time horizon H by dynamic programming iterating backward from $t = H$ down to $t = 0$.

1) The game finishes at $t = H$, and there are no more rewards:

$$O_{V_H}((s_1, f_1), \ldots, (s_k, f_k)) = 0, \text{ if } \sum_{i=1}^{k}(s_i + f_i) = H.$$

2) For a discount factor γ (assumed later to be $\gamma = 1$), using (1), the *expected value* V_{H_i} of pulling an arm i is given by

$$\begin{aligned} &V_{H_i}((s_1, f_1), \ldots, (s_k, f_k)) \\ &\quad = \frac{\alpha_i + s_i}{\alpha_i + s_i + \beta_i + f_i}(1 + \gamma \cdot O_{V_H}(\ldots, (s_i + 1, f_i), \ldots)) \\ &\quad + \frac{\beta_i + f_i}{\alpha_i + s_i + \beta_i + f_i}(0 + \gamma \cdot O_{V_H}(\ldots, (s_i, f_i + 1), \ldots)). \end{aligned}$$

3) For $0 \le t < H$ (where $t = \Sigma_{i=1}^{k}(s_i + f_i)$), the optimal strategy selects the *best expected value* according to the following rule:

$$\begin{aligned} &O_{V_H}((s_1, f_1), \ldots, (s_k, f_k)) \\ &\quad = \max_{i=1,\ldots,k}(V_{H_i}((s_1, f_1), \ldots, (s_k, f_k))) \\ &\quad \text{if } 0 \le \sum_{i=1}^{k}(s_i + f_i) < H. \end{aligned}$$

4) *Optimal* policy selects the arm with maximal V_H

$$\begin{aligned} &O_{P_H}((s_1, f_1), \ldots, (s_k, f_k)) \\ &\quad = \operatorname*{argmax}_{i=1,\ldots,k}(V_{H_i}((s_1, f_1), \ldots, (s_k, f_k))) \\ &\quad \text{if } 0 \le \sum_{i=1}^{k}(s_i + f_i) < H. \end{aligned}$$

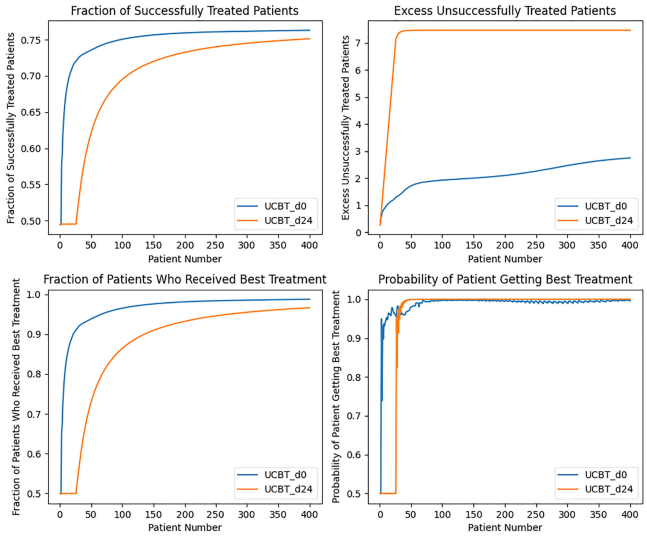

Fig. 2. Clinical trial simulation results. 2-arm Bernoulli bandit with arm probabilities of success 0.77 and 0.22. UCBT: no delay versus delay of 24.

The optimal policy can be easily computed using the memory-indexing scheme proposed in [16]. We use this scheme as a fundamental tool in a large number of simulation experiments presented in this article.

The optimal policy is organized as a 1-D array. In the case of 2-arm bandits, it can be indexed by the following expression:

$$\begin{aligned} \text{Index}((s_1, f_1), (s_2, f_2)) &= \frac{1}{24}\,t\,(t+1)(t+2)(t+3) \\ &+ \frac{1}{6}(s_1 + f_1)(s_1 + f_1 + 1)(2s_1 + 2f_1 - 3t - 5) \\ &+ s_1(t - s_1 - f_1 + 1) + s_2. \end{aligned}$$

Details of index expressions for three and more arms can be found in [16]. Efficient implementations of these expressions and procedures to compute the no-delay optimal policy can be downloaded from [40] and [41].

III. Delay Impact

Motivating Clinical Trial Example: To illustrate the impact of delays on algorithm effectiveness, we resimulated the 2-arm Bernoulli clinical trial model examined by Kuleshov *et al.* [8]. The number of arms and their reward probabilities were extracted from logs of a real medical trial with two treatments that had empirically determined 77% and 22% probabilities of success. During the trial, 360 patients received treatments sequentially. On average, before the treatment result for a given patient was available, decisions for 24 other patients also had to be made. Due to a lack of exact delay data, in our experiments, we modeled the process as a bandit with a constant reward delay of 24 decisions. The algorithm used in our resimulation is UCBT [36], which in [8] produced the best average number of successfully treated patients. The principles of this algorithm are explained in Section II-B. Our resimulation results represent averages of one million runs.

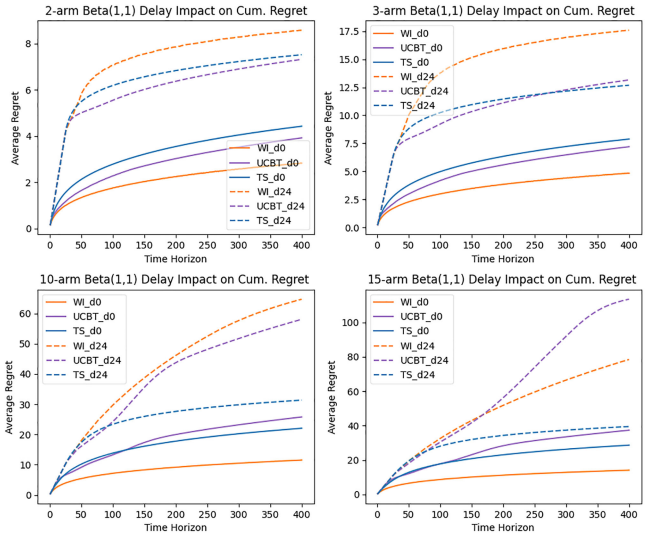

Fig. 3. Regret of common algorithms. No delay versus delay of 24.

Fig. 2 compares how a delay of 24 compares to no-delay using four different measures of algorithm effectiveness. Introducing delay results in a significantly smaller fraction of successfully treated patients during the trials and especially impacts the probability of successful treatment for early patients in the trial. If always providing the best treatment (77% success) to all patients is the baseline, the delay of 24 results in an additional (excess) seven unsuccessfully treated patients (83 versus 90) over the course of the entire trial (360 patients). With no delay, the excess is just over two. We consider such an impact of delay to be significant.[2]

Rigorous Evaluation: In the clinical trial example, we saw that the delay had a significant impact on the effectiveness of UCBT. It was only a simple example for two arms, fixed arm success probabilities, one delay value, and one algorithm.

We now present a more rigorous study examining various numbers of arms; we consider arm priors rather than fixed success probabilities, and we examine three algorithms: WI, TS, and UCBT. This selection is inspired by the results presented in [16] where a wider variety of algorithms were studied.

In a single experiment, the algorithm, time horizon, number of arms, and delay are preselected; then, arm success probabilities are drawn from their Beta priors, and the algorithm plays one game-making decisions solely on already known rewards. Fig. 3 presents simulation results for time horizons up to 400. Each data point is the average of one million simulation runs.

Observation 1: Reward delays significantly deteriorate performance of all algorithms, and the impact of delays increases with the number of arms.

Observation 2: TS is least sensitive to reward delays while the WI, which is nearly optimal in the absence of delays [16], performs poorly.

The last two observations lead us to the following research questions that guide this article.

[2]Our data appear to disagree with an observation in [8], which states that the impact of delays on algorithm effectiveness was minimal. No quantification in support of this statement was given.

Question 1: Can regret due to delay be significantly reduced without making additional assumptions on the general Bernoulli bandit model? If so, how?

Question 2: Is it possible to compute the optimal policy for Bernoulli bandits with delayed rewards?

We will address these questions in the reverse order.

IV. Optimal Policy Under Delay

The UCBT algorithm used in [8] and other algorithms presented in Section III make decisions solely from fully observed rewards. This means that these algorithms do not take into account arm pulls which still have not returned a reward value due to delay. Unfortunately, it is not possible to use the optimal policy in such a way. The optimal policy works under the strict assumption that all rewards are always known before any subsequent decision.

This relates to Question 2.

In this section, we show that starting from the principles of the optimal policy computations presented in Section II-C and using classic probability theory, it is possible *to compute the optimal policy under delay, which makes optimal decisions at any stage of the game for any set of still unknown rewards.* We also show that the optimal policy indexing scheme presented in [16] *makes such computations practically feasible for 2-arm and 3-arm bandits.*

A. Unknown-Rewards—Outcome Analysis

Actual (yet unknown) arm states: Consider a single arm and let (s, f) be its currently observed and known successes and failures. Assume that the arm has been pulled additional u times, and the corresponding rewards are still unknown due to delay (there is $s + f + u$ total arm pulls). Given that each unknown reward is either a success or a failure, the arm must be in one of the following actual success/failure states:

$$(s, f{+}u), (s{+}1, f{+}u{-}1), (s{+}2, f{+}u{-}2), \ldots, (s{+}u, f)$$

or using set-builder notation

$$\{(s + \delta, f + u - \delta) \mid \delta \in \mathbb{Z} \text{ and } 0 \leq \delta \leq u\}.$$

The probability of each state can be calculated via classic probability theory using priors derived from the known rewards.

Probabilities of unknown states: Fig. 4 presents an example of all possible reward changes for $u = 3$. This is a typical finite Markov chain model [1], [31]. The current observed and known reward state, by definition, has probability $\mathbb{P}[(s, f)] = 1$. The probabilities of possible current reward states $\mathbb{P}[(s + \delta, f + u - \delta)]$, $0 \leq \delta \leq u$, can be easily computed using well-known methods [1], [31]. Since the expected value of a binary random variable is equal to the probability of success, state transition probabilities can be determined using Equation (1) and Property 1.

Using the notation in Fig. 4, after taking into account the arm's prior, Beta(α, β), the probability of state $(s + 1, f)$ is

$$\mathbb{P}[(s + 1, f)] = \frac{s'}{s' + f'} \mathbb{P}[(s, f)] = \frac{s'}{s' + f'}$$

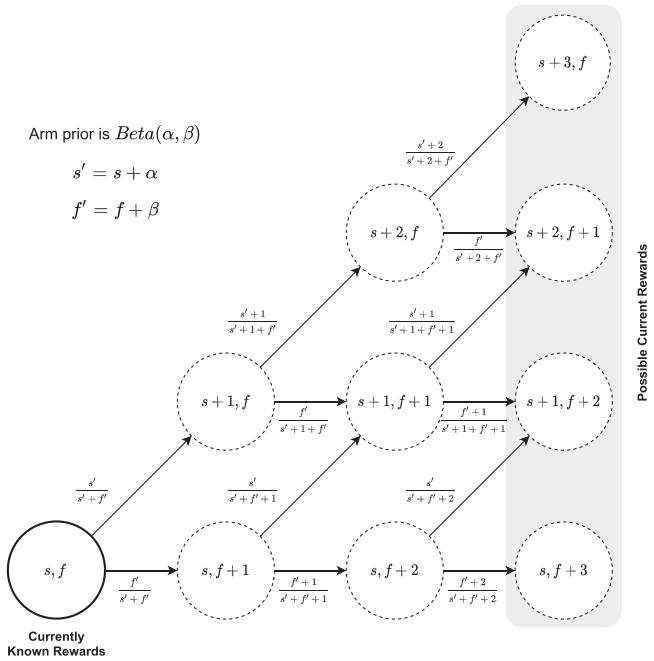

Fig. 4. Probabilistic analysis of reward outcomes. 1-arm and three unknown rewards.

TABLE I
NUMBER OF FLOATS IN THE PRECOMPUTED PART OF (2)

	Time Horizon				
	100	**200**	**300**	**400**	**1000**
2-arms	2.3 M	35.2 M	175 M	548 M	21 G
3-arms	297 M	16.8 G	184 G	1 T	—

and the probability of state $(s+1, f+1)$ is

$$\mathbb{P}[(s+1, f+1)] = \frac{f'}{(s'+1)+f'}\mathbb{P}[(s+1, f)] + \frac{s'}{s'+(f'+1)}\mathbb{P}[(s, f+1)].$$

Note that the outcome analysis does not assume any particular sequence in which rewards are revealed to the player. It only looks at how many rewards are still unknown. This means it is valid when two different pulls of the same arm produce rewards with arbitrarily different delays.

B. Expression

Once currently possible rewards and their probabilities have been computed for each arm, one can examine all possible configurations of successes and failures on all arms.

A single *configuration* is

$$((s_1+\delta_1, f_1+u_1-\delta_1), \ldots, (s_k+\delta_k, f_k+u_k-\delta_k))$$

where for any arm i, $0 \le i \le k$, $0 \le \delta_i \le u_i$. Its probability is $\prod_{j=1}^{k} \mathbb{P}[(s_j+\delta_j, f_j+u_j-\delta_j)]$.

For each configuration, the expected remaining reward value of pulling arm i equals $V_{H_i}((s_1+\delta_1, f_1+u_1-\delta_1), \ldots, (s_k+\delta_k, f_k+u_k-\delta_k))$, computed using equations given in Section II-C.

Consequently, taking into account all possible configurations and their probabilities, the expected remaining reward value of pulling arm i can be expressed as

$$\sum_{\substack{\delta_1=0 \\ \cdots \\ \delta_k=0}}^{\substack{u_1 \\ \cdots \\ u_k}} \left(V_{H_i}((s_1+\delta_1, f_1+u_1-\delta_1), \ldots) \prod_{j=1}^{k} \mathbb{P}[(s_j+\delta_j, f_j+u_j-\delta_j)] \right)$$

where

$$\sum_{\substack{\delta_1=0 \\ \cdots \\ \delta_k=0}}^{\substack{u_1 \\ \cdots \\ u_k}} (\text{arg}) = \sum_{\delta_1=0}^{u_1} \left(\sum_{\delta_2=0}^{u_2} \left(\ldots \left(\sum_{\delta_k=0}^{u_k} (\text{arg}) \right) \right) \right).$$

Now *the optimal policy under delay* can be expressed as

$$O_{\text{PUD}_H}((s_1, f_1, u_1), \ldots, (s_k, f_k, u_k)) = \underset{i=1,\ldots,k}{\text{argmax}} \sum_{\substack{\delta_1=0 \\ \cdots \\ \delta_k=0}}^{\substack{u_1 \\ \cdots \\ u_k}} \left(V_{H_i}((s_1+\delta_1, f_1+u_1-\delta_1), \ldots) \times \prod_{j=1}^{k} \mathbb{P}[(s_j+\delta_j, f_j+u_j-\delta_j)] \right). \quad (2)$$

The optimal policy under delay is a generalization of the optimal policy for the classic Bernoulli bandit problem, where all rewards are known before the next decision is made.

Note that we select the arm with best expected value. *This is, by definition, the best achievable regret performance by any algorithm.*

C. Computation

Our expression for $O_{\text{PUD}_H}((s_1, f_1, u_1), \ldots, (s_k, f_k, u_k))$ combines elements of the classic no-delay optimal policy computations (V_{H_i}) with results of Markov chain analysis ($\mathbb{P}[(s_i+\delta_i, f_i+u_i-\delta_i)]$). Taking into account, delays add a significant computational cost.

We assume that V_H (see Section II-C) is precomputed as in [16], and $\mathbb{P}[(s_i+\delta_i, f_i+u_i-\delta_i)]$ are evaluated on the fly. V_H storage requirements for small numbers of arms and various time horizons are presented in Table I. They were determined using expressions derived in [16], where practical efficiency of V_H computation for 2-arm and 3-arm bandits is also demonstrated.

As can be seen in Fig. 4, memory requirements for Markov chain computations are linear with respect to the number of unknown rewards, u; the number of floating point multiplications and divisions for one arm is $2\sum_{i=1}^{u} 2i = 2(u^2+u)$, and Markov chain computations can be easily parallelized for bigger values of u.

Observation 3: By using the memory-indexing method presented in [16], the optimal policy under delay can be used in practice for up to three arms.

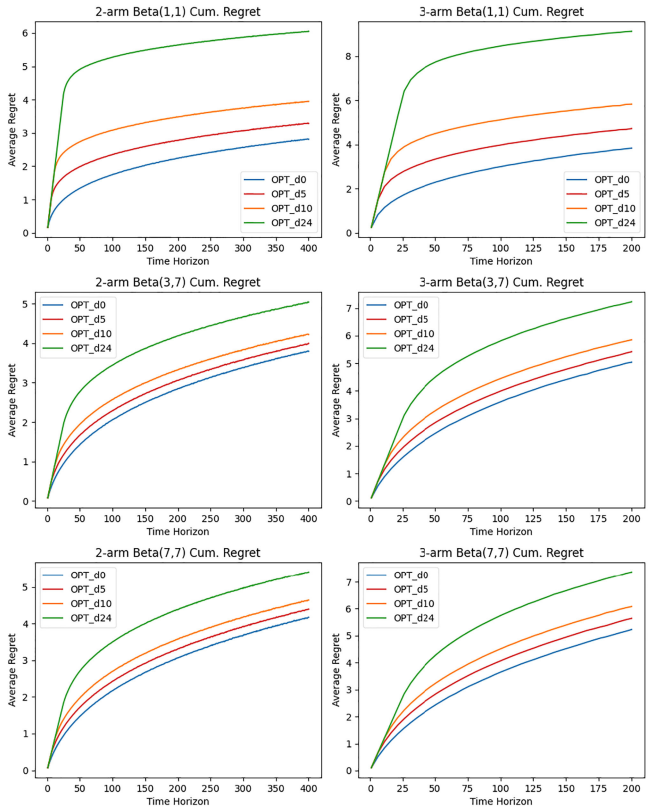

Fig. 5. Optimal policy regret for various delays. This is the ultimate lower bound on regret for any algorithm.

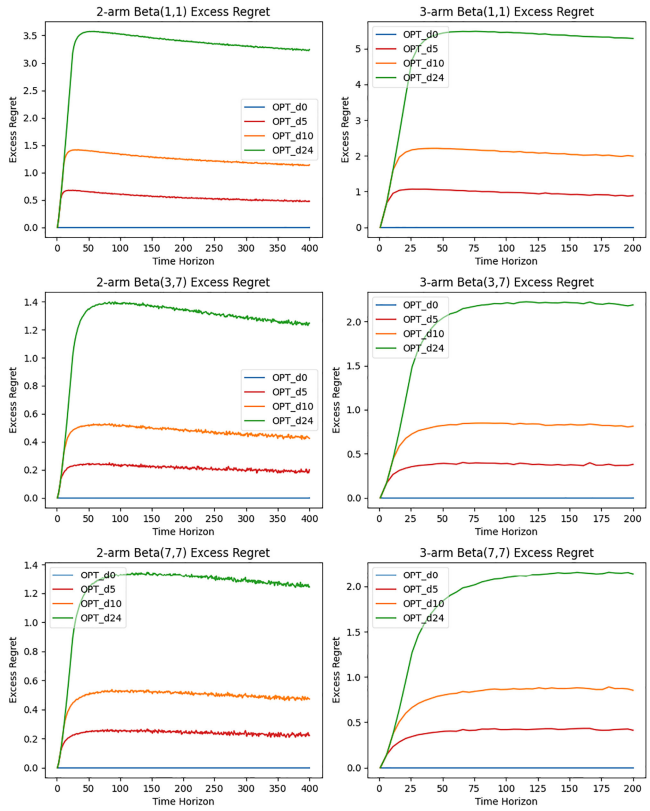

Fig. 6. Optimal policy: excess regret caused by delay.

D. Empirical Evaluation

We calculated the V_H tables for 2-arms and 3-arms with various priors and all time horizons up to 400 and 200, respectively. Then we evaluated O_{PUD_H} under these conditions by averaging the results of one million simulation runs. Results are shown in Fig. 5.

Observation 4: Delays cause significant excess regret even under optimal decisions, but the optimal-policy-under-delay regret is substantially lower than the regret for suboptimal algorithms (compare Fig. 5 with Fig. 3).

Excess regret caused by delay can be visualized by plotting the difference with the corresponding no-delay values. Fig. 6 presents such plots.

Observation 5: The optimal policy's excess regret caused by delay is largest for the uniform prior, i.e., Beta$(1, 1)$*, and increases with delay.*

V. Meta-Algorithm PARDI

The optimal policy under delay becomes difficult to use efficiently in practice when the number of arms is greater than three. Computational and storage requirements become prohibitive on standard computing machines. This brings us back to Question 1 in the context of suboptimal algorithms.

Abstraction: Every suboptimal algorithm presented in Section II-B, as well as other published algorithms, calculates some value measure of selecting a given arm (eval_arm) and applies argmax, just like the optimal policy. This similarity suggests that the principles of the expression in Equation (2), given in Section IV-B, which calculates the optimal policy under delay, $O_{\text{PUD}_H}((s_1, f_1, u_1), \ldots, (s_k, f_k, u_k))$, can be applied to algorithms other than the optimal policy. The key element of the optimal policy under delay is the consideration (prediction) of all possible—yet unknown—current rewards and their probabilities at a given time. *Such probabilities are a result of existing priors and remain independent of the valuation function.* Thus, the valuation part of the optimal policy under delay, V_{H_i}, can be replaced by the valuation function from any other algorithm, e.g., from UCBT. Consequently, the principles of the optimal policy under delay can be abstracted out into a meta-algorithm, which takes a valuation function (eval_arm) as an input. We call this meta-algorithm the PARDI. Fig. 7 shows how the PARDI meta-algorithm is applied to the optimal policy's valuation function using mathematical notation. As previously discussed, this is the optimal policy under delay.

Although our reasoning followed a different path, this meta-algorithm could be classified as a partially observable partially observable Markov decision process (POMDP)—a well-known technique in reinforcement learning [1].

Observation 6: The predictive approach derived for computing the optimal policy under delay can be extended to many common suboptimal algorithms, e.g., UCBT.

Simplification: For most algorithms, including UCBT, TS, and WI, the meta-algorithm can be simplified as they evaluate each arm independently of others. This means that other arms do not affect the valuation of a given arm i, and variables associated

$$\begin{aligned}
\textbf{no-delay optimal policy: } choose(t) &= \operatorname*{argmax}_{i=1,\ldots,k}\Big(eval_arm_i\big((s_1,f_1),\ldots,(s_k,f_k)\big)\Big), where\\
eval_arm_i\big((s_1,f_1),\ldots,(s_k,f_k)\big) &= V_{H_i}\big((s_1,f_1),\ldots(s_k,f_k)\big) \quad \text{// see Section II-C}\\
\textbf{PARDI meta algorithm: } choose(t,eval_arm) &= \operatorname*{argmax}_{i=1,\ldots,k}\Big(meta_eval_arm_i\Big(\big((s_1,f_1,u_1),\ldots\big),eval_arm_i\Big)\Big)\\
meta_eval_arm_i\Big(\big((s_1,f_1,u_1),\ldots\big),eval_arm_i\Big) &= \sum_{\substack{\delta_1=0\\ \cdots\\ \delta_k=0}}^{\substack{u_1\\ \cdots\\ u_k}}\Big(\underbrace{eval_arm_i\big((s_1+\delta_1,f_1+u_1-\delta_1),\ldots\big)}_{\text{No-delay arm evaluation}}\underbrace{\prod_{j=1}^{k}\mathbb{P}[(s_j+\delta_j,f_j+u_j-\delta_j)]}_{\text{predicts configuration}}\Big)
\end{aligned}$$

Fig. 7. Optimal policy under delay implemented using meta-algorithm predictive algorithm reducing delay impact (PARDI).

$$\begin{aligned}
\textbf{no-delay algorithm: } choose(t) &= \operatorname*{argmax}_{i=1,\ldots,k}\Big(eval_arm(s_i,f_i,t,H)\Big) \quad \text{// e.g., } \textit{WhittleIndex(s, f, H - t)}\\
\textbf{PARDI-S meta algorithm: } choose(t,eval_arm) &= \operatorname*{argmax}_{i=1,\ldots,k}\Big(meta_eval_arm(s_i,f_i,u_i,t,H,eval_arm)\Big)\\
meta_eval_arm_i(s_i,f_i,u_i,t,H,eval_arm) &= \sum_{\delta=0}^{u_i}\Big(\underbrace{eval_arm(s_i+\delta,f_i+u_i-\delta,t,H)}_{\text{No-delay arm evaluation}}\underbrace{\mathbb{P}[(s_i+\delta,f_i+u_i-\delta)]}_{\text{predicts configuration}}\Big)
\end{aligned}$$

Fig. 8. PARDI simplified (PARDI-S) for algorithms which evaluate arms individually. It reduces computational complexity.

with them can be eliminated. Thus, in the simplified meta-algorithm (PARDI-S, Fig. 8), $\text{eval\_arm}_i((s_1,f_1),\ldots,(s_k,f_k))$ from PARDI is replaced with simpler $\text{eval\_arm}(s_i+\delta, f_i+u_i-\delta, t, H)$, which has probability $\mathbb{P}[(s_i+\delta,f_i+u_i-\delta)]$. Consequently, as we no longer need to examine all possible configurations

$$\sum_{\substack{\delta_1=0\\ \cdots\\ \delta_k=0}}^{\substack{u_1\\ \cdots\\ u_k}} \text{ is replaced with } \sum_{\delta=0}^{u_i}.$$

Observation 7: When arms are evaluated independently (practically by all algorithms), PARDI-S is equivalent to PARDI as it merely eliminates redundant calculations.

Note that PARDI is still required for algorithms that do not evaluate arms independently (e.g., the optimal policy).

Algorithm 1 presents PARDI-S in an algorithmic manner. PARDI-S has to keep track of all statistics required for valuation by the applied bandit algorithm. It must internally maintain and update priors as a result of successes and failures to calculate probabilities. The algorithm iterates over each arm. For each possible arm state, it calculates the state's probability (see Section IV-A) and calculates the expected valuation. Finally, it selects the arm with the largest valuation.

For each arm, with a number of unknown rewards u, the number of floating point multiplications and divisions is $2\sum_{i=1}^{u} 2i = 2(u^2+u)$. For algorithms, which cannot use the PARDI-S optimization, computational cost grows (worst case) exponentially with the number of arms. When PARDI-S is used, the total number of multiplications grows (worst case) linearly with the number of arms. As PARDI-S is merely a computational complexity optimization of PARDI available for some algorithms, *we will not distinguish between PARDI and PARDI-S throughout the remainder of the article.*

Algorithm 1: PARDI-S.

Result: Elimination or reduction of delay impact
Input: Context: priors, time horizon H, time t
Input: Current (s_i, f_i, u_i) for each arm $i, 1 \le i \le k$.
Input: $eval\_arm$ function of a bandit algorithm, e.g.,

$$eval\_arm = WhittleIndex(s, f, H-t)$$

Output: Arm with best valuation under delay
Data: $E[i], 1 \le i \le k$, arms' valuations under delay
foreach *arm* i **do**
  $E[i] = 0$;
  foreach *state* $(s_i+\delta, f_i+u_i-\delta), 0 \le \delta \le u_i$ **do**
    $p = \mathbb{P}[(s_i+\delta, f_i+u_i-\delta)]$; /* Markov */
    $e = eval\_arm(s_i, f_i, H-t)$;
    $E[i] = E[i] + p \cdot e$;
  end
end
return $\operatorname{argmax}_{i=1}^{k} E[i]$.

VI. PARDI: Empirical Evaluation

In this section, we apply PARDI to UCBT, WI, and TS algorithms and evaluate their performance. Each algorithm is evaluated and results are averaged over a simulation study of one

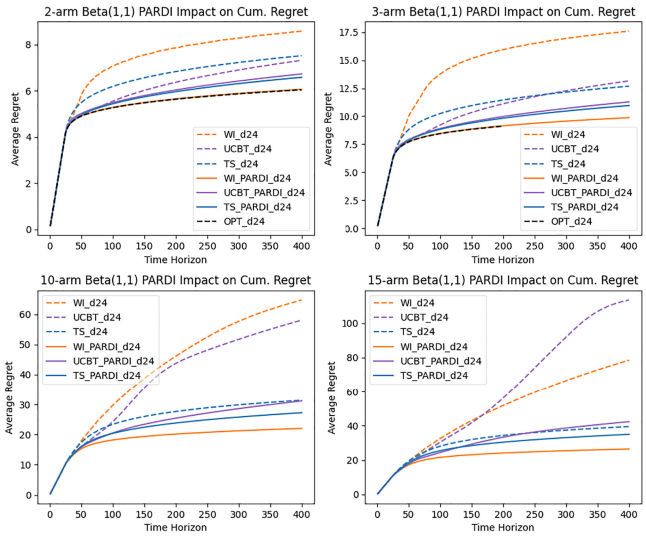

Fig. 9. PARDI: Reduction of regret for common algorithms, delay of 24.

TABLE II
PARDI: REDUCTION OF EXCESS-REGRET-DUE-TO-DELAY AND REGRET-DECREASE-FACTOR FOR TIME HORIZON 400; BETA(1,1)

		without PARDI		with PARDI		
Alg.	**arms**	**delay 0**	**delay 24**	**delay 24**	**reduct.**	**decr.**
WI	2	2.8315	8.5814	6.0661	44%	1.41x
	3	4.8406	17.5802	9.8815	64%	1.78x
	10	11.5299	64.7024	22.0047	80%	2.94x
	15	14.1046	78.4476	26.4493	81%	2.97x
UCBT	2	3.9263	7.3225	6.7346	17%	1.08x
	3	7.2051	13.1538	11.2846	31%	1.17x
	10	25.7976	58.0500	31.1195	83%	1.87x
	15	37.3594	113.4912	42.3696	93%	2.68x
TS	2	4.4300	7.5226	6.5866	30%	1.14x
	3	7.8876	12.6814	10.9589	36%	1.16x
	10	22.0499	31.3632	27.2134	45%	1.15x
	15	28.6613	39.4877	34.9860	42%	1.13x

million runs.[3] All data presented in this article were produced by a single extensively tested simulator[4] (e.g., recreated results published by others).

A summary of regret performance for delay of 24 can be found in Fig. 9 and Table II. Each algorithm with and without PARDI is presented for 2-arms, 3-arms, 10-arms, and 15-arms with probabilities drawn from the Beta(1,1) distribution.

Observation 8: PARDI significantly improved the performance of all tested algorithms—it eliminated up to 93% of excess regret and decreased cumulative regret by up to 3×.

Observation 8 is further supported by plots in Appendix B, where results for more delay values and various priors are also presented.

Note that 2-arm and 3-arm plots in Fig. 9 also show optimal-policy-under-delay results. They are labeled OPT_d24 and overlap plot WI_PARDI_d24. An examination of data in Fig. 5

[3]We repeated simulation studies of one million runs 100 times. The standard deviation of regret between such studies is below 0.005 for any setting and would not move plots/results in any significant or visible way.

[4]Simulation tools and experimental data related to this article can be found at https://github.com/SebastianPilarski/Bernoulli_bandits.

TABLE III
WI_PARDI IS NEARLY OPTIMAL

		2-arm		3-arm	
Prior	**Delay**	**OPT**	**WI_PARDI**	**OPT**	**WI_PARDI**
Beta(1,1)	5	2.77352	2.78650	4.71790	4.73984
	10	3.47832	3.48888	5.82506	5.84402
	24	5.64396	5.65896	9.11806	9.14856
Beta(3,7)	5	3.05990	3.06526	5.41926	5.41572
	10	3.32584	3.34036	5.85242	5.87978
	24	4.18788	4.20548	7.22888	7.26536
Beta(7,7)	5	3.31386	3.31916	5.63938	5.65076
	10	3.58572	3.59082	6.07974	6.11108
	24	4.38830	4.41610	7.36972	7.40542

Regret for time horizon 200.

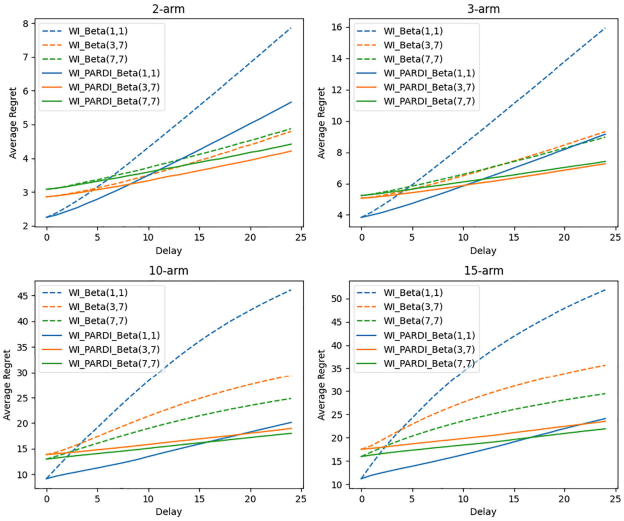

Fig. 10. WI_PARDI: Reduction of regret as a function of delay. $H = 200$.

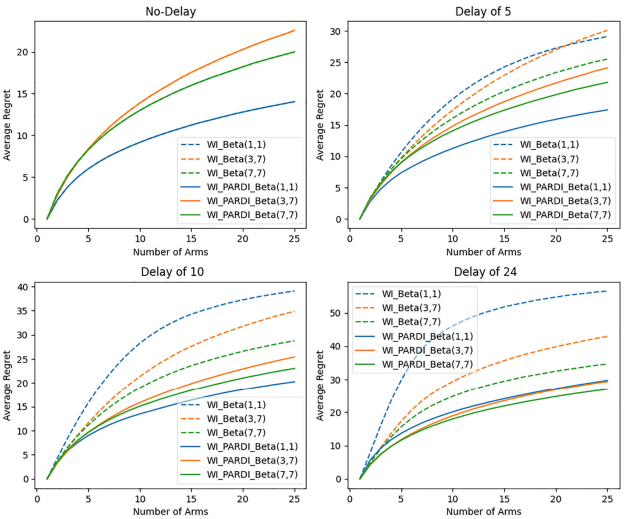

Fig. 11. WI_PARDI: Reduction of regret versus the number of arms. $H = 200$.

and the corresponding data in Fig. 12(a) leads to the following observation.

Observation 9: For 2-arm and 3-arm delayed reward bandits, WI_PARDI offers nearly optimal regret performance.

Data in Table III support this claim by comparing the optimal regret to WI_PARDI regret for various delays and arm

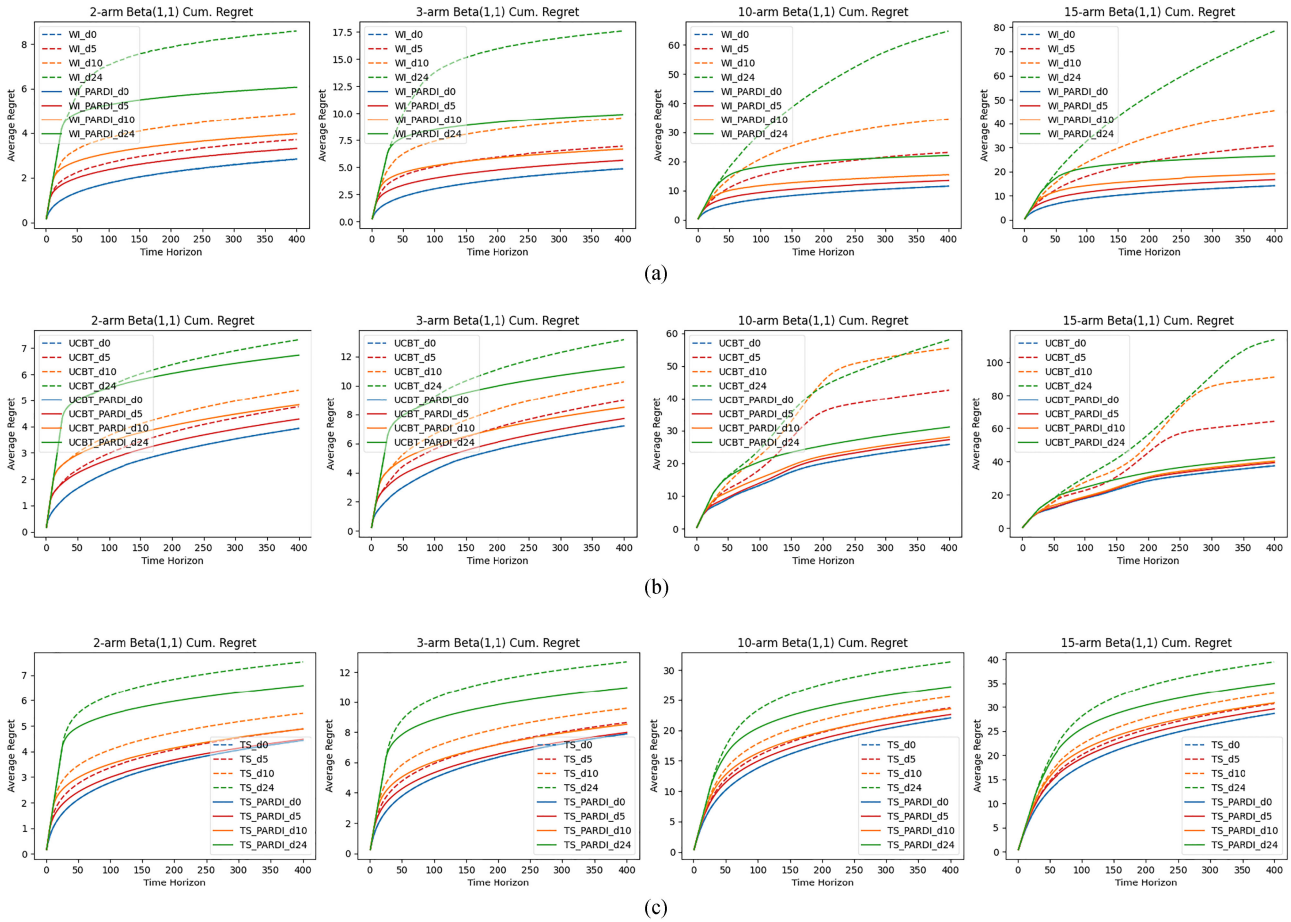

Fig. 12. PARDI: Regret reduction for various algorithms, delays, and numbers of arms. (a) WI: Regret reduction by PARDI. (b) UCBT: Regret reduction by PARDI. (c) TS: Regret reduction by PARDI.

priors. Observation 9 is a generalization of a similar observation made in [16], where no delays were considered, but a larger scope of number of arms and time horizons were examined.

Fig. 10 shows WI regret and WI_PARDI regret as a function of delay for three different priors. WI_PARDI regret growth vs delay appears to be slightly faster than linear.

Fig. 11 shows WI regret and WI_PARDI regret as a function of the number of arms.

Observation 10: WI_PARDI regret increases with greater delays and number of arms. Varying the prior Beta distribution has a significant and complex effect.

VII. Summary and Conclusion

This article shows that delayed rewards may cause a dramatic decrease of regret-related performance of well-known algorithms, and render the no-delay optimal policy useless. It presents not only a novel, but first, computational method that determines the general optimal policy under delays. The main idea is to use a classic probability theory, namely finite Markov chains, to predict yet unknown rewards and their probabilities, and fall back on no-delay optimal policy procedures. The memory-indexing scheme, proposed in [16], makes the generalized optimal policy practically applicable if the number of arms does not exceed three. In the case of 2-arm bandits, it can be easily computed for time horizons of 1000.

This article also demonstrated that the predictive part of the generalized optimal policy can be abstracted out as a meta-algorithm, PARDI, applicable to well-known suboptimal bandit algorithms. PARDI brings very significant improvements to their performance measured by regret, e.g., up to more than $3\times$ in the case of 15-arm bandits with delay of 24.

Finally, this article shows that PARDI applied to the WI offers nearly optimal regret when 2-arm and 3-arm bandits are examined. This observation holds regardless of delays, arms' priors, and examined time horizons. In experiments with 10-arm and 15-arm bandits, PARDI-enhanced WI was performing better than any other suboptimal algorithm.

PARDI does not make any additional assumptions on the general Bernoulli bandit model. It does not make any assumptions on reward delays associated with individual arm pulls. PARDI is an effective practical solution to reward delay-related performance loss observed in well-known suboptimal algorithms.

APPENDIX A
WHITTLE INDEX

The WI is closely related to the Gittins index [16], [38]. It considers a finite time horizon rather than an infinite one.

"The main idea is as follows. Consider playing just one arm with an unknown probability of success. For any state of the game, it is possible to replace the arm with an arm of known probability if it improves expected reward. The goal is to find the smallest known probability p for any state of the game, which is the value of the index" [16].

Equations that precisely define the WI are as follows:

$$V^*_{\gamma,H}(s,f,p,0) = 0$$

$$V^*_{\gamma,H}(s,f,p,j) = \max\left\{ p\sum_{i=0}^{j-1}\gamma^i, \right.$$
$$\frac{s}{s+f}\left(1+\gamma\cdot V^*_{\gamma,H}(s+1,f,p,j-1)\right)$$
$$\left. +\frac{f}{s+f}\left(0+\gamma\cdot V^*_{\gamma,H}(s,f+1,p,j-1)\right)\right\}$$
$$j=1,\ldots,H$$

$$W_I(s,f,j) = \min p:\ p\sum_{i=0}^{j-1}\gamma^i \geq$$
$$\frac{s}{s+f}(1+\gamma\cdot V^*_{\gamma,H}(s+1,f,p,j-1))$$
$$+\frac{f}{s+f}(0+\gamma\cdot V^*_{\gamma,H}(s,f+1,p,j-1))$$

where j represents the remaining playing time, and we assume the discount factor $\gamma = 1$ in this article.

The first equation says that there are no more rewards after the last pull. The second and third equations, which use Property 1 and Equation (1), say that WI is computed "sequentially backward," i.e., starting from the last pull.

Both equations are an application of Property 1 and Equation (1).

APPENDIX B
PARDI: ADDITIONAL EMPIRICAL EVALUATION

Fig. 12 presents regret reduction by WI_PARDI, UCBT_PARDI, and TS_PARDI for various delay values and number of arms. The plots in this figure support our observations and can serve as a guide to select the most effective algorithm for a particular application.

Additional plots for various priors can be found online[5].

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. Cambridge, MA, USA: MIT Press, 2018.

[2] D. A. Berry, "Modified two-armed bandit strategies for certain clinical trials," *J. Amer. Stat. Assoc.*, vol. 73, no. 362, pp. 339–345, 1978.

[3] S. S. Villar, J. Bowden, and J. Wason, "Multi-armed bandit models for the optimal design of clinical trials: Benefits and challenges," *Statist. Sci.*, vol. 30, no. 2, pp. 199–215, 2015.

[4] S. S. Villar, "Bandit strategies evaluated in the context of clinical trials in rare life-threatening diseases," *Probability Eng. Inf. Sci.*, vol. 32, no. 2, pp. 229–245, 2018.

[5] D. A. Berry and B. Fristedt, *Bandit Problems: Sequential Allocation of Experiments* (Monographs on Statistics and Applied Probability), vol. 5. London, U.K.: Chapman and Hall, no. 71/87, 1985, pp. 7–7 .

[6] Y. Cheng, F. Su, and D. A. Berry, "Choosing sample size for a clinical trial using decision analysis," *Biometrika*, vol. 90, no. 4, pp. 923–936, 2003.

[7] T. Friede *et al.*, "Recent advances in methodology for clinical trials in small populations: The inspire project," *Orphanet J. Rare Dis.*, vol. 13, no. 1, pp. 186–186, Oct. 2018.

[8] V. Kuleshov and D. Precup, "Algorithms for multi-armed bandit problems," *J. Mach. Learn. Res.*, vol. 1, pp. 1–48, 2014.

[9] M. Zhu, X. Zheng, Y. Wang, Y. Li, and Q. Liang, "Adaptive portfolio by solving multi-armed bandit via thompson sampling," 2019, *arXiv:1911.05309*.

[10] E. Kaufmann, O. Cappé, and A. Garivier, "On the complexity of A/B testing," in *Proc. Conf. Learn. Theory*, 2014, pp. 461–481.

[11] Y. Mao, M. Chen, A. Wagle, J. Pan, M. Natkovich, and D. Matheson, "A batched multi-armed bandit approach to news headline testing," in *Proc. IEEE Int. Conf. Big Data*, 2018, pp. 1966–1973.

[12] S. Katariya, B. Kveton, C. Szepesvári, C. Vernade, and Z. Wen, "Bernoulli rank-1 bandits for click feedback," in *Proc. 26th Int. Joint Conf. Artif. Intell.*, 2017, pp. 2001–2007.

[13] D. Bouneffouf and I. Rish, "A survey on practical applications of multi-armed and contextual bandits," 2019, *arXiv:1904.10040*.

[14] G. Burtini, J. Loeppky, and R. Lawrence, "A survey of online experiment design with the stochastic multi-armed bandit," 2015, *arXiv:1510.00757*.

[15] T. Lattimore and C. Szepesvári, *Bandit Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2020.

[16] S. Pilarski, S. Pilarski, and D. Varro, "Optimal policy for Bernoulli bandits: Computation and algorithm gauge," *IEEE Trans. Artif. Intell.*, vol. 2, no. 1, pp. 2–17, Feb. 2021.

[17] L. Liu, R. Downe, and J. Reid, "Multi-armed bandit strategies for non-stationary reward distributions and delayed feedback processes," 2019, *arXiv:1902.08593*.

[18] L. Zhou, "A survey on contextual multi-armed bandits," 2015, *arXiv:1508.03326*.

[19] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," in *Proc. Int. Conf. Neural Inf. Process. Syst.*, 2011, pp. 2249–2257.

[20] N. Akbarzadeh and A. Mahajan, "Conditions for indexability of restless bandits and an algorithm to compute Whittle index," 2020, *arXiv:2008.06111*.

[21] K. Ronoh, R. Oyamo, E. Milgo, M. Drugan, and B. Manderick, "Bernoulli bandits an empirical comparison," in *Proc. 23rd Eur. Symp. Artif. Neural Netw., Comput. Intell. Mach. Learn.*, 2015, pp. 59–64.

[22] P. Jacko, "BinaryBandit: An efficient Julia package for optimization and evaluation of the finite-horizon bandit problem with binary responses," Lancaster Univ. Manage. School, Lancaster, U.K., Working Paper, 2019. [Online]. Available: https://eprints.lancs.ac.uk/id/eprint/136340

[23] P. Jacko, "The finite-horizon two-armed bandit problem with binary responses: A multidisciplinary survey of the history, state of the art, and myths," 2019, *arXiv:1906.10173*.

[24] P. Joulani, A. Gyorgy, and C. Szepesvári, "Online learning under delayed feedback," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 1453–1461.

[25] C. Pike-Burke, S. Agrawal, C. Szepesvari, and S. Grunewalder, "Bandits with delayed, aggregated anonymous feedback," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, vol. 80, pp. 4105–4113.

[26] J. Tyo, O. Neopane, J. Byrd, C. Gupta, and C. Igoe, "Multi-armed bandits with delayed and aggregated rewards," DEVCOM Army Res. Lab., White Oak, MD, USA, Tech. Rep. ARL-TR-8754, 2019.

[27] A. Grover *et al.*, "Best arm identification in multi-armed bandits with delayed feedback," in *Proc. Mach. Learn. Res.*, 2018, vol. 84, pp. 833–842.

[28] Z. Zhou, R. Xu, and J. Blanchet, "Learning in generalized linear contextual bandits with stochastic delays," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, vol. 32, pp. 5197–5208.

[29] M. Yang, H. Zhu, H. Wang, Y. Koucheryavy, K. Samouylov, and H. Qian, "Peer to peer offloading with delayed feedback: An adversary bandit approach," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, 2020, pp. 5035–5039.

[5] [Online]. Available: https://github.com/SebastianPilarski/Bernoulli_bandits

[30] S. Ito *et al.*, "Delay and cooperation in nonstochastic linear bandits," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 4872–4883, 2020.
[31] J. Doob, *Stochastic Processes* (Ser. Probability and Statistics Series). Hoboken, NJ, USA: Wiley, 1953.
[32] J. M. Bernardo and A. F. Smith, *Bayesian Theory*. Hoboken, NJ, USA: Wiley, 2009.
[33] W. R. Thompson, "On the likelihood that one unknown probability exceeds another in view of the evidence of two samples," *Biometrika*, vol. 25, no. 3/4, pp. 285–294, 1933.
[34] W. R. Thompson, "On the theory of apportionment," *Amer. J. Math.*, vol. 57, no. 2, pp. 450–456, 1935.
[35] E. Kaufmann, N. Korda, and R. Munos, "Thompson sampling: An asymptotically optimal finite-time analysis," in *Proc. Int. Conf. Algorithmic Learn. Theory*, 2012, pp. 199–213.
[36] P. Auer, N. Cesa-Bianchi, and P. Fischer, "Finite-time analysis of the multiarmed bandit problem," *Mach. Learn.*, vol. 47, no. 2/3, pp. 235–256, 2002.
[37] P. Whittle, "Restless bandits: Activity allocation in a changing world," *J. Appl. Probability*, vol. 25, pp. 287–298, 1988.
[38] J. C. Gittins, "Bandit processes and dynamic allocation indices," *J. Roy. Stat. Soc.: Ser. B. (Methodological)*, vol. 41, no. 2, pp. 148–164, 1979.
[39] J. Chakravorty and A. Mahajan, "Multi-armed bandits, Gittins index, and its calculation," in *Methods and Applications of Statistics in Clinical Trials: Planning, Analysis, and Inferential Methods.* Hoboken, NJ, USA: Wiley, 2014, ch. 24, pp. 416–435.
[40] S. Pilarski *et al.*, "Bernoulli bandit data and tools repository." 2021. [Online]. Available: https://github.com/SebastianPilarski/Bernoulli_bandits
[41] S. Pilarski *et al.*, "Bernoulli bandit code capsule," 2021, doi: 10.24433/CO.1300732.v1.

Sebastian Pilarski (Graduate Student Member, IEEE) is currently working toward the Ph.D. degree with the Department of Electrical and Computer Engineering, McGill University, Montreal, QC, Canada.

He has been investigating applications of artificial intelligence in gas turbine design and control with Siemens Energy. His research interests include software engineering, machine learning, reinforcement learning, and their applications to systems engineering.

Slawomir Pilarski (Member, IEEE) received the M.Sc. degree from the Warsaw University of Technology, Poland, and the Ph.D. degree from the Institute of Electron Technology, Warsaw, Poland. He was a tenured Professor with Simon Fraser University, Burnaby, BC, Canada. He held Director-level positions with Synopsys, Mountain View, CA, USA, and Magma DA, San Jose, CA, USA, where he led advanced formal verification R&D teams. He founded two startups—one acquired by Magma. His work on built-in self-test of VLSI circuits was recognized by the IEEE Design and Test of Computers as a milestone in test technology and was deployed by several major chip manufacturing companies. He developed core algorithms and architected two industry-leading formal verification tools including second-generation formality—an equivalence checker. He has authored or coauthored more than 40 research papers and an IEEE monograph, and has also coauthored three patents in three different research domains. His research interests include computer architecture, various aspects of VLSI design, distributed databases, formal verification, and logic synthesis to theoretical computer science and reinforcement learning.

Dániel Varró (Member, IEEE) received the Ph.D. degree from the Budapest University of Technology and Economics. He is currently a Full Professor with McGill University, Montreal, QC, Canada. He is a coauthor of more than 170 scientific papers.

Mr. Varró was the recipient of seven Distinguished Paper Awards and three Most Influential Paper Awards. He is on the Editorial Board of *Software and Systems Modeling* and *Journal of Object Technology* periodicals, and was a Program Co-Chair of MODELS 2021, SLE 2016, ICMT 2014, and FASE 2013 conferences. He delivered keynote talks at numerous conferences, including CSMR, SOFSEM, and SAM, and International Summer Schools. He is a Co-Founder of the VIATRA open source model query and transformation framework, and IncQuery Labs, Budapest, Hungary, a technology-intensive Hungarian company.