Periodic Bandits and Wireless Network Selection

Shunhao Oh
Department of Computer Science, National University of Singapore
ohoh@u.nus.edu

Anuja Meetoo Appavoo
Department of Computer Science, National University of Singapore
anuja@comp.nus.edu.sg

Seth Gilbert
Department of Computer Science, National University of Singapore
seth.gilbert@comp.nus.edu.sg

Abstract

Bandit-style algorithms have been studied extensively in stochastic and adversarial settings. Such algorithms have been shown to be useful in multiplayer settings, e.g. to solve the wireless network selection problem, which can be formulated as an adversarial bandit problem. A leading bandit algorithm for the adversarial setting is EXP3. However, network behavior is often repetitive, where user density and network behavior follow regular patterns. Bandit algorithms, like EXP3, fail to provide good guarantees for periodic behaviors. A major reason is that these algorithms compete against fixed-action policies, which is ineffective in a periodic setting.

In this paper, we define a periodic bandit setting, and periodic regret as a better performance measure for this type of setting. Instead of comparing an algorithm’s performance to fixed-action policies, we aim to be competitive with policies that play arms under some set of possible periodic patterns \( F \) (for example, all possible periodic functions with periods \( 1, 2, \ldots, P \)). We propose Periodic EXP4, a computationally efficient variant of the EXP4 algorithm for periodic settings. With \( K \) arms, \( T \) time steps, and where each periodic pattern in \( F \) is of length at most \( P \), we show that the periodic regret obtained by Periodic EXP4 is at most \( O\left(\sqrt{PKT\log K + KT\log|F|}\right) \). We also prove a lower bound of \( \Omega\left(\frac{PKT}{\log K} + \frac{KT\log|F|}{\log K}\right) \) for the periodic setting, showing that this is optimal within log-factors. As an example, we focus on the wireless network selection problem. Through simulation, we show that Periodic EXP4 learns the periodic pattern over time, adapts to changes in a dynamic environment, and far outperforms EXP3.

2012 ACM Subject Classification Theory of computation → Online learning algorithms

Keywords and phrases multi-armed bandits, wireless network selection, periodicity in environment

Supplement Material Source code: https://github.com/Ohohcakester/PeriodicEXP4-Source

Acknowledgements This project was funded by Singapore Ministry of Education Grant MOE2018-T2-1-160 (Beyond Worst-Case Analysis: A Tale of Distributed Algorithms).
1 Introduction

The multi-armed bandit problem is an online learning problem in which a player has access to a set of choices (i.e., “arms”) each of which provides some reward (i.e., “gain”). At each time step, the player chooses an arm and gets some reward. In stochastic variants, rewards are determined by some probabilistic distribution. In adversarial variants, an adversary specifies the rewards. Amazingly, even when rewards are adversarially chosen, the player can do fairly well! For example, the EXP3 algorithm [6] minimizes the player’s “regret”, ensuring that the player does almost as well as if she had selected the single fixed best arm throughout. Another fascinating property of bandit algorithms is that they work well in multi-player settings [27, 16], converging to close variants of a Nash equilibrium.

Recently, it has been shown that bandit-style algorithms can efficiently solve the wireless network selection problem, yielding good performance both in theory and in practice [1, 2, 7]. In this problem, each user has access to a collection of networks (e.g., a few different WiFi networks and a 4G connection); the goal is to pick networks with higher data rates. Selecting the best network is challenging, especially in dynamic environments where the “best” network changes over time, as users move and network bandwidth fluctuates. This can be modeled as an adversarial bandit problem and solved with EXP3 and its variants.

Bandit algorithms have one major weakness in dynamic settings (such as wireless network settings): they are designed to learn the average payoff of each arm, and to converge to the arm that provides the best average performance. In the stochastic case, this is exactly what you want. In the adversarial case, it leads to minimum regret, i.e., the user does almost as well as if they knew the best network in advance. If, however, the situation is changing over time, and especially if it is changing in some predictable manner, then learning the average payoff of each arm is not productive.

Periodic, repetitive patterns are a particularly common type of dynamic behavior. Take, for example, the problem of network selection. Network behavior is often repetitive, with user density and network quality following regular patterns: for example, office WiFi networks have no users at night, their performance drops when workers arrive in the morning, and the performance improves again during lunch hour. Other networks are clogged with streaming video during lunch hour and in the evenings. Periodic patterns are ubiquitous.

Unfortunately, bandit algorithms will fail badly in the case of periodic behavior. As an example, suppose a player is playing a slot machine with two arms. The first arm gives a reward of 1 when pulled on odd-numbered hours and 0 otherwise, while the second arm does the reverse, with a reward of 1 on even-numbered hours and 0 otherwise. In this simple case, a bandit algorithm will never learn this pattern, instead converging to the best single-action policy; and the best policy can only reap half of the maximum reward. The player will receive an average payout of only 1/2 per selection, despite a very predictable pattern. And when this case is extended to cycle among K arms, the best fixed choice of arm gives only 1/K of the total obtainable reward. Thus, algorithms like EXP3 that minimize the regret do not guarantee good performance on periodic problems.

1.1 Contributions

Our goal in this paper is to develop an efficient adversarial bandit algorithm for periodic settings, and to demonstrate the effectiveness of this algorithm in the context of the wireless network selection problem, yielding a new approach to network selection in dynamic, periodic environments. The first step is to establish the right metric by which to evaluate bandit algorithms. The performance of an adversarial bandit algorithm is heavily characterized by
the definition of “regret,” which forms the baseline that it competes against. And traditionally, the regret is computed with respect to the best fixed strategy.

For the periodic bandit setting, we define a better performance measure, ‘periodic regret’, which compares an algorithm’s performance against the best periodic choice of arms. No choice of period may match the input data perfectly, but the goal of periodic regret is to compare against the best choice. Moreover, we provide a generalized notion of periodicity, so that this notion of periodic regret can capture different types of patterned behavior.

Next, we develop an algorithm that minimizes periodic regret, Periodic EXP4, a computationally efficient variant of EXP4 (Exponential-weight algorithm for Exploration and Exploitation using Expert advice) [6]. We show that the algorithm minimizes periodic regret in the following sense: with $K$ arms, $|F|$ possible periods, with each possible period of at most length $P$, then in an execution of length $T$ the periodic regret is at most $O\left(\sqrt{PKT \log K + KT \log |F|}\right)$. We also prove a lower bound of $\Omega\left(\sqrt{PKT + KT \log |F| \log K}\right)$ on periodic regret in an adversarial setting, showing that this is optimal within log-factors. An important aspect of Periodic EXP4 is that it is a polynomial time algorithm: we leverage the structure provided by the target periodic patterns to reduce the computational complexity. This is in contrast to EXP4 which requires exponential time and space in this context.

The other major contribution of this paper is a new algorithm for network selection that is especially optimized for environments with periodic, patterned behaviors. We simulate the network selection problem, comparing Periodic EXP4 to EXP3 and to a “randomized optimal” omniscient solution. (We have previously seen in [1] that these types of simulations are reasonably predictive of real-world behavior.)

Our first observation is that Periodic EXP4 does in fact efficiently learn periodic patterns and adapts relatively quickly to changes in network data rates (both discrete and continuous). We also see that Periodic EXP4 does indeed outperform EXP3 in periodic settings, as expected, potentially yielding significant real-world improvements.

Our second question involved the robustness of Periodic EXP4 to noisy patterns. Real-world periodic patterns are rarely perfectly periodic, suffering noise and variance. We experiment with noisy patterns, and see that Periodic EXP4 continues to work well.

Finally, our third set of experiments looked at the performance of Periodic EXP4 in the context of user mobility. We simulate several scenarios where users change location over time, leading to changes in which networks they can access (and hence changes in the load on those networks). For example, we imagine a typical office scenario where users arrive at the office in the morning, take a break for lunch, return to work, and then head home at the end of the day. We observe that Periodic EXP4 can also learn this type of periodic behavior, again, learning to adapt the users’ network selection in a near-optimal fashion. In fact, we compare two versions of the algorithm: one in which the algorithm is notified when networks become unavailable, and one in which it is not—we observe that even in the latter case where it is completely oblivious to the changes, the user strategy converges to near-optimal choices.

Overall, we conclude that periodic adversarial bandit algorithms may have significant value, that Periodic EXP4 is an efficient algorithm for the problem, and that it yields a potentially interesting and useful approach to network selection.

2 Related work

In this section, we discuss relevant work done on bandit algorithms, and state-of-art wireless network selection approaches. Multi-armed bandit techniques have been successfully applied to wireless network selection [1] [2] [7]. They have also been considered for other resource
Periodic Bandits and Wireless Network Selection

selection problems, such as channel selection \cite{13,27}, selection of the right sensors to query in a sensor network \cite{14}, and selection of replica server for content distribution networks \cite{28}.

Many variations of bandit problems have been studied, in both stochastic and adversarial settings. EXP3 is the most well-known algorithm for the standard adversarial bandit problem. With $K$ arms and $T$ time steps, it establishes a pseudo-regret upper bound of $O(\sqrt{KT \log K})$, which almost matches the lower bound of $\Omega(\sqrt{KT})$ \cite{6}. The log $K$ gap in the bounds has been recently closed by \cite{5} bringing the upper bound down to $O(\sqrt{KT})$. But, these bound the regret against the best single-action policy, limiting their usefulness in a periodic setting.

A related problem is that of bandits with expert advice, defined in the same paper \cite{6}. It defines a more general notion of regret, by competing against the best policy from a set. With $K$ arms, $T$ time steps and $N$ experts, the EXP4 algorithm gives a pseudo-regret bound of $O(\sqrt{KT \log N})$. However, its possibly high running time and memory cost limit its use in practice. There are other algorithms for bandits with expert advice, like Context-FTPL. The latter is more computationally efficient, but has a weaker regret bound \cite{26}. A lower bound of $\Omega(\sum KT \log N \log K)$ \cite{23} has been shown, but the log $K$ gap in bounds has not been closed.

An equivalent formulation of our generalized periodic regret (explained later in Section 4.2) has been briefly discussed in \cite{10} Chapter 4.2.1, phrased as a contextual bandit problem where the algorithm competes against the best context set from a class of context sets. The possible use of EXP4 is mentioned, but an alternative algorithm with a weaker regret bound is instead discussed as it has a reasonable polynomial-time performance unlike EXP4.

While much of the existing literature assume a single best arm, there are other efforts to look beyond this. One approach to the stochastic version of the problem is to allow reward distributions of the arms to occasionally change \cite{9,22}. Our work on the other hand is fully adversarial, and makes no assumptions on the rewards produced by the adversary.

Numerous wireless network selection approaches have been proposed. Some are centralized \cite{3,8,18,25}; hence, not scalable and limited to managed networks. A number of distributed approaches have been proposed, with various limitations. Some rely on coordination from networks \cite{15}, while others require cooperation of wireless devices \cite{12}. Others assume global knowledge \cite{20,4,19}, or availability of some information \cite{30,11}. A continuous-time multi-armed bandit approach in a stochastic setting has been considered in \cite{29}. A similar setting to ours, though non-periodic and in the stochastic setting, is considered in \cite{7}.

3 Wireless Network Selection

Here, we describe the wireless network selection problem, discuss the periodicity of events in wireless environments, and formulate the network selection problem as a bandit problem.

3.1 Wireless network selection problem.

We consider an environment with multiple wireless devices and heterogeneous wireless networks, such as the one depicted in Figure 1. The latter illustrates four mobile users with their (active) mobile devices, and five wireless networks, namely four WiFi networks and a cellular network (represented using 3 cellular base stations). The wireless networks have limited areas of coverage. Hence, each mobile device may have access to a different set of wireless networks depending on their location, e.g. different networks are available at home and at the office. The bandwidths of wireless networks may also vary with time. Each mobile device aims to quickly identify and associate with the best network, which may vary over time, to maximize their data rates.
Mobile users tend to have daily routines that follow repetitive patterns—going to the office each morning, lunch at noon, returning home in the evening; these activities are performed at fixed times each weekday. Figure 1 broadly depicts the daily routine of a mobile user, Alice. Network behavior, which is affected by user density, is also often repetitive and follows a regular pattern. For example, the available bandwidth of office WiFi networks is likely to be higher during lunch hours, where the office is nearly empty. A good network selection protocol learns and adapts to periodic patterns in network quality for better performance.

3.2 Wireless network selection as a bandit problem

A device must be aware of the bit rate it can observe from each network to perform an optimal network selection. While this information is unknown at the time of selection, the device can estimate the achievable bit rate by exploring the networks. The network selection problem can be seen as a multi-armed bandit problem in a multi-player setting. A mobile device is a player, and each network can be considered as an arm. Every so often (e.g. once per minute), a device selects a network (analogous to pulling an arm) and observes a bit rate (gain) for that network. The gain from other networks is unknown to the device. Given that mobile devices operate in a dynamic environment, they must continuously explore and adapt to changes, by deciding which networks to select in sequence. The goal of each device is to maximize its cumulative gain over time. Since the quality of a wireless network is affected by its number of clients, other mobile devices in the environment may be considered to be adversaries. We hence use the adversarial setting. A leading bandit algorithm in this setting is EXP3.

4 Periodic Bandit Problem

In this section, we introduce the periodic bandit problem and discuss periodic regret.

We consider a general bandit problem. On each time step, an algorithm is allowed to pick any one out of $K$ possible arms, and each arm produces a certain amount of reward. These rewards are unknown to the algorithm, which can only observe the reward of the arm it picked. We aim to maximize the total reward obtained by the algorithm. We study the adversarial setting with a possibly adaptive adversary, which decides on the distribution of rewards at each time step, taking into consideration the outcomes of past random events.

Let $K$ be the number of arms. The set of arms is $[K] := \{1, 2, \cdots, K\}$. Let $x_i(t) \in [0, 1]$ be the reward earned by arm $i \in [K]$ at time step $t$. Let $a(t) \in [K]$ be the arm played by the algorithm at time $t$. Let $T$ be the total number of time steps. The set of time steps is $[T] := \{1, 2, \cdots, T\}$. Thus, the total reward earned by the algorithm after $T$ iterations is $\sum_{t=1}^{T} x_{a(t)}(t)$. The commonly used performance measure for bandit algorithms is regret. Regret compares the total reward obtained by the algorithm against a “best possible” reward.
“OPT” after some number of time steps $T$. Different types of regret compare the algorithm’s result to different notions of the optimal result.

We can define a form of regret where OPT is allowed to pick any arm in $[K]$ at each time step. For later reference we will refer to this as full regret, defined as follows:

$$R_{\text{full}}(T) = T \sum_{t=1}^{T} \max_{i \in [K]} E[x_i(t)] - E \left[ \sum_{t=1}^{T} x_{a(t)}(t) \right]$$

The above definition uses what is commonly known as pseudo-regret, rather than expected regret. For the rest of this paper, we will often refer to pseudo-regret as simply “regret”. Expectations are taken over the possible randomness of the algorithm and adversary.

In most studies of adversarial bandits, a weaker definition of regret is used. This is because full regret uses too powerful an adversary, and it is impossible to achieve better than linear expected full regret in the worst case (we include a proof in Appendix A.1). Therefore, it is common to define a notion of regret where OPT is required to use the same arm for all $T$ time steps. We refer to this as weak regret, defined as follows:

$$R_{\text{weak}}(T) = \max_{i \in [K]} T \sum_{t=1}^{T} E[x_i(t)] - E \left[ \sum_{t=1}^{T} x_{a(t)}(t) \right]$$

Weak regret however, severely limits what OPT can do, and being competitive with an algorithm that can only pick one arm and stick to it may not be a very strong result.

### 4.1 Periodic Regret

We can bridge the two with a periodic definition of regret. Taking the idea that a periodic choice of arms is likely to perform well in situations with periodic patterns, we can define a regret function which measures how competitive an algorithm is with the best periodic choice of arms. For example, we can say OPT is forced to play the same arm every $\tau \in \mathbb{N}$ steps. This defines a regret function as follows,

$$R_{\tau}(T) = \tau \sum_{\ell=1}^{\lfloor T/\tau \rfloor} \max_{i \in [K]} \sum_{t=0}^{\tau-1} E[x_i(t\tau + \ell)] - E \left[ \sum_{t=1}^{T} x_{a(t)}(t) \right]$$

As OPT may optionally still pick the same arm on all time steps, this is a generalization of weak regret. This makes for a regret value in between weak regret and full regret.

If we were competing against the regret for a specific, known value of $\tau$, this would be equivalent to playing $\tau$ independent instances of the adversarial bandits problem over approximately $T/\tau$ time steps each. By playing $\tau$ separate instances of an algorithm for weak regret, and by Theorem 2 in Section 6.1, we have an upper/lower bound of $\Theta(\sqrt{\tau KT})$.

However, if we were to consider that the “best possible” period $\tau$ may not be known (for example, if OPT were to consist of the best periodic function for any of the possible periods $\tau \in \{1, \cdots, P\}$), these bounds do not apply as easily.

### 4.2 Generalized Periodic Regret

A generalization of the periodic case is the use of partition functions. Fix a maximum number of labels $P$. We define this upper bound $P$ for use in our analysis later on. A partition function $f : [T] \to [P]$ is a function that assigns every time step a label from 1 to $P$. We consider two partition functions the same if their choice of label assignments are permutations
of each other. The regret under function $f$ would be when OPT is forced to play the same arm for all timesteps with the same label as assigned by $f$.

$$R_f(T) = \sum_{\ell \in [P]} \max_{\ell \in [K]} \sum_{t \in f^{-1}(\ell)} E\left[ x_i(t) \right] - E\left[ \sum_{t=1}^T x_{a(t)}(t) \right]$$

Consider a set of partition functions $F \subseteq \{ f : [T] \to [P] \}$ for some $P \in \mathbb{N}$. $F$ is necessarily finite. The regret under the function set $f$ would be when OPT can choose to play using any of the partition functions in $F$. This gives the following regret definition:

$$R_F(T) = \max_{f \in F} \sum_{\ell \in [P]} \max_{\ell \in [K]} \sum_{t \in f^{-1}(\ell)} E\left[ x_i(t) \right] - E\left[ \sum_{t=1}^T x_{a(t)}(t) \right]$$

This definition of periodic regret gives us more choice in how we want to define our potential periodic patterns to learn, through deciding on the labels on each time step for each function. We demonstrate this with our choice of partition functions in Section 7.

To model the example described earlier with periods $\tau \in \{1, 2, \ldots, P\}$, we can use the set of partitions $F = \{ f_1, f_2, \cdots, f_P \}$, where $f_\tau(t) := (t \mod \tau) + 1$ for each $t \in [T]$, $\tau \in [P]$.

5 The Periodic EXP4 Algorithm

We discuss the relationship between our generalized periodic setting and the problem of bandits with expert advice, and hence the applicability of EXP4 to the problem. We use this to introduce Periodic EXP4, an efficient algorithm for generalized periodic regret.

5.1 Applying Bandits with Expert Advice to Periodic Bandit Problems

Periodic bandit problems can be reduced to the problem of bandits with expert advice. In the problem of bandits with expert advice, we are given a set $\Pi \subseteq \mathbb{N}$ experts. Each expert predicts an arm on each time step. We fix the number of time steps $T$. Thus an expert can be seen as a function $\pi : [T] \to [K]$. An algorithm to solve this problem would make use of each expert’s predictions on each time step, to obtain a reward competitive with the best expert in the set. This gives us the following regret definition:

$$R_\Pi(T) = \max_{\pi \in \Pi} \sum_{t=1}^T x_{\pi(t)}(t) - E\left[ \sum_{t=1}^T x_{a(t)}(t) \right]$$

This can be used to model all of the above notions of regret. For full regret, we have $\Pi := \{ \pi : [T] \to [K] \}$, the set of all possible functions from $[T]$ to $[K]$. For weak regret, $\Pi$ is the set of all constant functions from $[T]$ to $[K]$.

In the generalized periodic setting, let $F$ be the set of partition functions $f : [T] \to [P]$. For each function $f \in F$, let $\Theta_f$ be the set of all possible mappings $\theta : f([T]) \to [K]$ from the image set $f([T])$ of $f$ to the set of arms $[K]$ (thus $|\Theta_f| = K^{|f([T])|}$). Each composition $\theta \circ f$, $f \in F$, $\theta \in \Theta_f$ thus represents a possible mapping of the time steps $[T]$ to arms. Thus, for the generalized periodic setting, $\Pi = \{ \theta \circ f \mid f \in F, \theta \in \Theta_f \}$.

We note that when $\Pi_1 \subseteq \Pi_2$, we will have $R_{\Pi_1}(T) \leq R_{\Pi_2}(T)$. Let $\Pi_{full}$, $\Pi_{weak}$ and $\Pi_F$ be the sets of functions corresponding to full regret, weak regret and generalized periodic regret under some function set $F$ respectively. Thus, for any nonempty set $F$ of partition functions, we have $R_{\Pi_{weak}}(T) \leq R_{\Pi_F}(T) \leq R_{\Pi_{full}}(T)$. 
An existing algorithm for this problem is the EXP4 algorithm [6], which achieves a regret upper bound of \(O(\sqrt{KT\log N})\), where \(N := |\Pi|\). We can thus apply EXP4 directly to our problem. However, a commonly cited drawback of the EXP4 algorithm is that its running time and memory cost are at least linear in \(N\). This is an issue as \(N\) is often very large. For example, in the generalized periodic setting, the size of \(N\) could easily be on the order of \(|F|KP\), which is exponential in \(P\). However, we show below that in the generalized periodic setting, we can devise an algorithm that is distributionally equivalent to EXP4 and can be made to run in time polynomial in \(|F|, K\) and \(P\).

The EXP4 algorithm works by assigning a weight \(w_\pi\) (with initial value 1) to each expert \(\pi \in \Pi\). The probability \(p_i(t)\) of playing an arm \(i \in [K]\) would then be \(\sum_{\pi(t)=i} w_\pi(t)/\sum_{\pi} w_\pi(t)\), the ratio of the combined weights of the experts agreeing to play arm \(i\) to the total weight of the experts. Whenever an arm \(i \in [K]\) is played, each expert who suggested arm \(i\) will have their weight adjusted by some factor \(\exp(\gamma x_i(t)/p_i(t))\). More details on EXP4 are given in [6]. Note that it discusses a more general form of expert advice where each expert suggests a probability vector on the arms. However, we only require the case where at each time step, each expert suggests one arm with probability 1, and all other arms with probability 0.

5.2 Periodic EXP4, Memory and Running Time Costs

Periodic EXP4 (Algorithm [1]) is distributionally equivalent to the EXP4 algorithm when run with the set of experts \(\Pi = \{\theta \circ f \mid f \in F, \theta \in \Theta_f\}\). The key intuition behind this algorithm is that the generalized periodic setting produces many symmetries in the weight computation for each expert. Specifically, we take advantage of how for each partition function \(f\), the set of experts contains every possible combination of arm assignments to labels in the image set \(f([T])\). This allows us to compute the probabilities that EXP4 would play each arm at each time step without computing the individual weights of every expert.

For brevity, let \(P_f := |f([T])|\) be the number of labels used by the function \(f\). Necessarily \(P_f \leq P\). The memory requirement is \(O(K \sum_{f \in F} P_f)\), which is at most \(O(KP|F|)\). A naive implementation of the algorithm gives a running time of \(O(K^2 \sum_{f \in F} P_f)\) per time step, but with some pre-computation, the running time can be lowered as shown in Appendix A.2.

5.3 Correctness of Periodic EXP4

To show correctness, we show that our algorithm produces the same probability distribution over arms as EXP4 in every time step. Define \(\pi_{\theta,f}\) as the expert which at time \(t\) recommends arm \(\theta \circ f(t)\) with probability 1 and all other arms with probability 0. We show this algorithm is distributionally equivalent to EXP4, where \(\Pi = \{\pi_{\theta,f}\mid f \in F, \theta \in \Theta_f\}\). In EXP4, each expert \(\pi_{\theta,f}\) would have some weight \(w_{\theta,f}(t)\) at time step \(t\). At time step \(t\), EXP4 plays arm \(i\) with probability \(p_i(t)\) represented by the following expression:

\[
p_i(t) = \frac{\sum_{f \in F, \theta \in \Theta_f, \theta \circ f(t) = i} w_{\theta,f}(t)}{\sum_{f \in F, \theta \in \Theta_f} w_{\theta,f}(t)}
\]

Thus, to show that the two algorithms are distributionally equivalent, as \(p_i(t) := r_i(t)/\sum_{j=1}^K r_j(t)\) in our algorithm, for each successive time step \(t\), we only need to show the following:

\[
r_i(t) = \sum_{f \in F, \theta \in \Theta_f, \theta \circ f(t) = i} w_{\theta,f}(t)
\]
Algorithm 1 Periodic EXP4

1: procedure Initialization
2: for each \( f \in F \) do
3:     for each \( \ell \in f([T]) \) do
4:         for each \( i \in [K] \) do
5:             Initialize \( b^{0,f}_{i(t)}(1) = 1 \)
6: end
7: procedure Algorithm
8: for each time step \( t = 1, 2, \cdots, T \) do
9:     for each \( i \in [K] \) do
10:        \( r_{i(t)} := \sum_{f \in F} b^{f(t),f}_{i}(t) \prod_{\ell \in f([t]) \setminus f(t)} \sum_{j=1}^{K} b^{f}_{j}(t) \)
11:        for each \( i \in [K] \) do
12:           \( p_{i(t)} = \frac{r_{i(t)}}{\sum_{i=1}^{K} r_{i(t)}} \)
13:    for each \( f \in F \) do
14:       for each \( \ell \in f([T]) \) do
15:           for each \( i \in [K] \) do
16:               if \( i = i_{t} \) and \( \ell = f(t) \) then
17:                  \( b^{f}_{i}(t+1) = b^{f}_{i}(t) \exp \left( \frac{x_{i}(t)}{p_{i}(t)} \right) \)
18:               else
19:                  \( b^{f}_{i}(t+1) = b^{f}_{i}(t) \)

The details of this derivation is given in Appendix A.3. We can thus formally state a regret upper bound as follows (Theorem 1). This upper bound comes directly from EXP4’s regret bound of \( O(\sqrt{KT \log N}) \), where the number of experts \( N = \sum_{f \in F} K^{|f([T])|} \leq |F|K^{P} \).

\begin{center}
| Theorem 1. | With \( K \) arms, \( T \) time steps, \( |F| \) partition functions, with every function having at most \( P \) labels, Periodic EXP4 gives a regret upper bound of \( O(\sqrt{PKT \log K} + KT \log |F|) \).
\end{center}

6 Lower Bounds

In this section, we provide lower bounds for the case of a single partition and for a set of partitions. We demonstrate that the upper and lower bounds differ by a factor of \( \log K \).

The existing regret lower bound for the problem of bandits with expert advice \cite{23} is \( \Omega(\sqrt{KT \log N / \log K}) \). This lower bound is derived by dividing the time steps \( [T] \) into \( \log N / \log K \) equal parts. For the generalized periodic setting, as this lower bound uses an instance that can be modeled with a single partition function, it does not give immediate insight into whether having multiple different periods or partition functions increases the difficulty of the problem.

6.1 Lower Bound for a Single Partition

We consider the case with only a single partition function \( f : [T] \rightarrow [P] \), which partitions the time steps into \( P \) labels \( 1, 2, \cdots, P \). The sizes of the partitions are \( |f^{-1}(1)|, |f^{-1}(2)|, \cdots, |f^{-1}(P)| \) respectively. It seems like intuitively, by seeing this as \( P \) separate instances of the
weak regret setting, and by the existing $\Theta(\sqrt{KT})$ upper/lower bounds on weak regret \cite{27,28}, we would have an upper/lower bound of $\Theta(\sum_{p=1}^{P} \sqrt{K[f^{-1}(p)]})$. For equally sized partitions of size approximately $T$ each, this bound would be $\Theta(\sqrt{PKT})$.

However, while the upper bound is clearly met by running $P$ independent instances of an algorithm for weak regret, the lower bound is less clear. Even when considering it as $P$ separate instances, there is a possibility of an algorithm “reacting” to losses in other instances to play differently in the current instance, obtaining a higher total reward as a result. For completeness, we include a proof for the lower bound (Theorem 2) in Appendix A.4.

\begin{theorem}
Fix a partition function $f : [T] \rightarrow [P]$ which assigns a label to each time step. Assume that for each $\ell \in f([T])$, there are at least $K/(4 \ln \frac{4}{3})$ time steps with label $\ell$. Then the minimax pseudo-regret $\inf_a \sup_R \left( \max_{\ell \in f(T)} E \left[ \sum_{t \in [T]} x_{a_f(t)}(t) \right] - E \left[ \sum_{t \in [T]} x_{a(t)}(t) \right] \right)$ is at least $\sqrt{\alpha KT}$ for some positive constant $\alpha$.
\end{theorem}

If we consider the simple case where OPT may play only periodic functions from any period $\tau \in \{1, 2, \cdots, P\}$, it can do no worse than if it were only allowed to play at period $P$. We thus obtain a lower regret bound of $\sqrt{PKT}$.

### 6.2 Lower Bound for the Generalized Periodic Setting

Let $F$ be the set of partitions, so $|F|$ is the number of partitions. Let $P$ be the maximum number of labels of any partition in $F$. For sufficiently large $T$ and $K \leq P$, we obtain a pseudo-regret $\Omega(\sqrt{PKT})$ lower bound of $\Omega(\sqrt{PKT} + \sqrt{KT\log\frac{|F|}{\log K}})$. It is proved in Appendix A.5.

If $P < K$ instead, a simple lower bound can be obtained by using only $P$ out of the $K$ arms, so we obtain a problem with $P$ arms and maximum partition size $P$. This gives us a lower bound of $\Omega\left(\sqrt{PKT} + \sqrt{PT\log\frac{|F|}{\log P}}\right)$. We can then merge these two lower bounds into a single expression $\Omega\left(\sqrt{PKT} + \min(P, K)T\frac{\log|F|}{\log\min(P,K)}\right)$.

### 6.3 Analysis of Bounds

A conclusion we can make from Section 6.2 is that having multiple periods indeed increases the difficulty of the problem - we have obtained a lower bound higher than the known upper bound of $O(\sqrt{PKT})$ had only one partition function of the maximum period $P$ been used.

With $K$ arms, $T$ time steps, $|F|$ partition functions, with every function having at most $P$ labels, Periodic EXP4 gives an upper bound of $O(\sqrt{PKT \log K} + KT \log|F|)$. On the other hand, we have a lower bound of $\Omega(\sqrt{PKT + KT \log|F|})$ in the case where $K \leq P$. This gives a gap of $\sqrt{\log K}$ between the two bounds. Interestingly, this log-factor is the same as the current gap between the upper and lower bounds in the problem of bandits with expert advice. This is possibly because we use a similar lower bound proof to the problem of bandits with expert advice \cite{23}, as well as a similar algorithm for the upper bound.

### 7 Experimental Evaluation

In this section, we discuss the implementation details of Periodic EXP4 and parameter values chosen, evaluate the algorithm via simulation, and compare its performance to EXP3 \cite{29}. We show how Periodic EXP4 (a) learns periodic patterns over time under both discrete and
continuous changes in network data rates, (b) outperforms EXP3, (c) is robust to noisy patterns, and (d) adapts to changes due to mobility of users.

We benchmark against “Optimal Random”, a player with prior knowledge of the actual bandwidths of each network. In each time slot, it picks a network from a probability distribution equal to the ratios of the bandwidths. For example, with network bandwidths 4, 10 and 6, the probability of picking the networks will be 0.2, 0.5 and 0.3, respectively.

All the algorithms are implemented in Python, using SimPy [24], while the core algorithm is written in C++. We use a time-varying learning rate $\gamma = t^{-\frac{1}{2}}$ [17] for both Periodic EXP4 and EXP3; $\gamma$ slowly tends to zero to ensure convergence [27] while at the same time ensures that the algorithm does not take too long to learn (it learns slowly when $\gamma$ is very small). Although they are not pre-requirements of Periodic EXP4, for simplicity, we assume that (a) a network’s bandwidth is equally shared among its clients, and (b) devices are time-synchronized. To reduce numerical error in our simulations, we substitute computations of $\sum_{x \in Y} \exp(x)$ with $\exp(\max_{x \in Y} x)$. In nearly all cases, sums of exponentials in our algorithm are heavily dominated by a single term, making the values of the two expressions approximately equal. Experimentally, we find that this has negligible effects on the values computed within the algorithm.

We do simulations on synthetic data. We consider setups with 20 mobile devices and 3 wireless networks, unless otherwise specified. While the number of devices remain constant throughout the simulation run, the data rates and availability of networks may change. We assume that a network selection is performed once every minute; hence, 1440 time slots is one simulated day. All results presented are from 20 simulation runs, of 86,400 time slots each (i.e., 2 simulated months). The pattern of network behavior and/or user mobility over the first 1440 time slots is repeated 60 times; we refer to each repetition as an ‘iteration’.

We apply Periodic EXP4 in the generalized periodic setting. We define a partition function of period $\tau$ as one which divides each iteration of 1440 time slots into $\tau$ equal contiguous segments, labeled 1 to $\tau$ in chronological order. The same labels are used for each successive repetition. Unless otherwise specified, we use the period set $\{1, \ldots, 24\}$. This refers to using 24 partition functions, of periods 1 to $\tau$ respectively.

### 7.1 Evaluation Criteria

Good assignments of devices to networks divide the available bandwidth evenly among the devices. We thus evaluate the performance of the algorithms based on the lowest data rate observed by any of the devices. We compare this to the optimal allocation of devices, which maximizes the lowest data rate observed by any device. If a device with the lowest data rate observes 3Mbps, but the optimal’s lowest is 5Mbps, we say it loses 40% of its achievable gain. We refer to this percentage loss as the “distance to optimal minimum” in our results.

We do not use average cumulative gain as a performance measure because in our problem setting, average gain is maximized as long as there is at least one user in each network.

### 7.2 Performance Comparison of Algorithms

We consider two setups, both at an office with two WiFi networks and a cellular network. The data rates of these networks vary over time. The first setup involves discrete changes in network bandwidths at fixed time intervals (Figure 2a). In the second setup, the data rates vary continuously with time (Figure 2b). Figures 3a and 3b show that in both setups, the distance to optimal minimum of Periodic EXP4 drops over time while EXP3 shows no noticeable improvement with time.
Periodic Bandits and Wireless Network Selection

(a) Discrete changes in network data rates.  

(b) Continuous changes in network data rates.

Figure 2 Changes in network data rates over one iteration (this is repeated 60 times).

(a) Performance under discrete setup.  

(b) Performance under continuous setup.

Figure 3 Distance to optimal minimum of Periodic EXP4 and EXP3 over 60 iterations.

(a) EXP3: Combined probabilities for each network over first 10 iterations.  

(b) Bandwidth ratio

(c) Periodic EXP4: Combined probabilities for each network over first 10 iterations.

Figure 4 Area chart showing the time variation of combined probabilities in the continuous setup. Figure 1b shows the actual ratio of the bandwidths of the three networks within any one iteration.

Figure 4 for the continuous setup explains this improvement. The figure for the discrete setup is in Appendix B.1. At each time step, each user has a probability of picking each of the networks. If we consider the combined probability of picking each network, we can see that in Periodic EXP4, these probabilities converge towards the ratios of the bandwidths of the networks (Figures 4c). This is despite the continuous setup having no obvious best period. On the other hand, EXP3’s probabilities slowly flatten out (Figure 4a). This is consistent with what we would expect, as EXP3 seeks to be competitive with the best fixed-action policy, meaning that it only seeks out the best fixed arm to play.

Figure 5 shows that while EXP3 initially learns more quickly, Periodic EXP4 eventually...
outperforms EXP3 (which converges to the network with the best average performance), with a performance similar to Optimal Random. From our experiments, we find that while all algorithms have similar total cumulative gains, we may note that Periodic EXP4 is fairer than EXP3, with significantly lower variance. We present these results in Appendix B.4.

7.3 Other Experiments

In Appendix B, we discuss a few more experiments, the results of which are briefly summarized as follows:

1. **Performance in Noisy Settings**: On each time step, we apply a 10% Gaussian noise to each of the networks’ data rates. We find that our algorithms are largely unaffected by noise in the data, giving similar results with and without noise.

2. **Comparison of Period Sets**: We do a comparison between different possible period sets $F$. We find that the algorithm learns more slowly with larger period sets (e.g. \{1, 2, \cdots , 45\}, as compared to \{1, 2, \cdots , 15\}), but can converge to better results on more complex instances (instances where the bandwidth may fluctuate more wildly).

3. **Mobility of Users**: We consider a setup where users move around and have access to different sets of networks at different times. We compare Vanilla Periodic EXP4, which is oblivious to networks possibly becoming unavailable, against an optimized version, which selects only from the set of currently available networks. While the optimized version initially yields a better performance, they eventually perform equally well when the Vanilla Periodic EXP4 algorithm learns the pattern.

8 Conclusion

In this paper, we develop an efficient variant of EXP4 for the periodic bandit problem, give nearly matching upper and lower bounds for it, and demonstrate its advantages in learning periodic behavior in the context of the network selection problem.

An interesting issue raised in contrasting this paper and \[9, 22\] is whether non-stationary bandit problems are better modeled stochastically or adversarially. While these papers address non-stationary rewards primarily in a stochastic setting with some adversarial aspects, we tackle the periodic bandit problem in a fully adversarial setting. Using the adversarial setting has the benefit of not placing any constraints on the adversary; we adapt to the periodic setting only through our definition of regret. A proper comparison of stochastic and adversarial methods for network selection is a possible future line of work.
References

1. Anuja Meetoo Appavoo, Seth Gilbert, and Kian-Lee Tan. Shrewd selection speeds surfing: Use smart exp3! In 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS), pages 188–199. IEEE, 2018.
2. Anuja Meetoo Appavoo, Seth Gilbert, and Kian-Lee Tan. Cooperation speeds surfing: Use co-bandit! arXiv preprint arXiv:1901.07768, 2019.
3. E. Aryafar, A. Keshavarz-Haddad, C. Joe-Wong, and M. Chiang. Max-min fair resource allocation in hetnets: Distributed algorithms and hybrid architecture. In ICDCS, 2017, pages 857–869. IEEE, 2017.
4. E. Aryafar, A. Keshavarz-Haddad, M.1 Wang, and M. Chiang. Rat selection games in hetnets. In INFOCOM, pages 998–1006. IEEE, 2013.
5. Jean-Yves Audibert and Sébastien Bubeck. Minimax policies for adversarial and stochastic bandits. In COLT, pages 217–226, 2009.
6. P. Auer, N. Cesa-Bianchi, Y. Freund, and R. E. Schapire. The nonstochastic multiarmed bandit problem. SIAM Journal on Computing, 32(1):48–77, 2002.
7. O. Avner and S. Mannor. Multi-user lax communications: A multi-armed bandit approach. In IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications, pages 1–9, April 2016. doi:10.1109/INFOCOM.2016.7524557.
8. Y. Bejerano, S.-J. Han, and L. E. Li. Fairness and load balancing in wireless lans using association control. In MobiCom, pages 315–329. ACM, 2004.
9. Omar Besbes, Yonatan Gur, and Assaf Zeevi. Stochastic multi-armed-bandit problem with non-stationary rewards. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 199–207. Curran Associates, Inc., 2014. URL: http://papers.nips.cc/paper/5378-stochastic-multi-armed-bandit-problem-with-non-stationary-rewards.pdf.
10. Sébastien Bubeck, Nicolo Cesa-Bianchi, et al. Regret analysis of stochastic and nonstochastic multi-armed bandit problems. Foundations and Trends® in Machine Learning, 5(1):1–122, 2012.
11. M. H. Cheung, F. Hou, J. Huang, and R. Southwell. Congestion-aware distributed network selection for integrated cellular and wi-fi networks. arXiv preprint arXiv:1703.00216, 2017.
12. S. Deng, A. Sivaraman, and H. Balakrishnan. All your network are belong to us: A transport framework for mobile network selection. In HotMobile. ACM, 2014.
13. Yi Gai, Bhaskar Krishnamachari, and Rahul Jain. Learning multiuser channel allocations in cognitive radio networks: A combinatorial multi-armed bandit formulation. In New Frontiers in Dynamic Spectrum, 2010 IEEE Symposium on, pages 1–9. IEEE, 2010.
14. D. Golovin, M. Faulkner, and A. Krause. Online distributed sensor selection. In Proceedings of the 9th ACM/IEEE International Conference on Information Processing in Sensor Networks, pages 220–231. ACM, 2010.
15. B. Kauffmann, F. Baccelli, A. Chaintreau, V. Mhatre, K. Papagiannaki, and C. Diot. Measurement-based self organization of interfering 802.11 wireless access networks. In INFOCOM 2007, pages 1451–1459. IEEE, 2007.
16. R. Kleinberg, G. Piliouras, and E. Tardos. Multiplicative updates outperform generic no-regret learning in congestion games. In ACM STOC, pages 533–542. ACM, 2009.
17. S. Maghsudi and S. Stanczak. Relay selection with no side information: An adversarial bandit approach. In WCNC, pages 715–720. IEEE, 2013.
18. A. Mishra, V. Brik, S. Banerjee, A. Srinivasan, and W. A. Arbaugh. A client-driven approach for channel management in wireless lans. In Infocom, 2006.
19. E Monsef, A. Keshavarz-Haddad, E. Aryafar, J. Saniie, and M. Chiang. Convergence properties of general network selection games. In INFOCOM, pages 1445–1453. IEEE, 2015.
20. D. Niyato and E. Hossain. Dynamics of network selection in heterogeneous wireless networks: An evolutionary game approach. TVT, 58(4):2008–2017, 2009.
21 Basil Cameron Rennie and Annette Jane Dobson. On stirling numbers of the second kind. Journal of Combinatorial Theory, 7(2):116–121, 1969.

22 Allesiardo Robin, Raphaël Féraud, and Odalric-Ambrym Maillard. The non-stationary stochastic multi-armed bandit problem. International Journal of Data Science and Analytics, 03 2017. doi:10.1007/s41060-017-0050-5

23 Yevgeny Seldin and Gábor Lugosi. A lower bound for multi-armed bandits with expert advice. In 13th European Workshop on Reinforcement Learning (EWRL), 2016.

24 SimPy. SimPy - Event discrete simulation for Python, 2016. https://simpy.readthedocs.io/, accessed 2018-19-12.

25 K. Sui, M. Zhou, D. Liu, M. Ma, D. Pei, Y. Zhao, Z. Li, and T. Moscibroda. Characterizing and improving wifi latency in large-scale operational networks. In MobiSys, pages 347–360. ACM, 2016.

26 Vasilis Syrgkanis, Akshay Krishnamurthy, and Robert Schapire. Efficient algorithms for adversarial contextual learning. In International Conference on Machine Learning, pages 2159–2168, 2016.

27 C. Tekin and M. Liu. Performance and convergence of multi-user online learning. In GAMENETS, pages 321–336. Springer, 2011.

28 H. A. Tran, S. Hoceini, A. Mellouk, J. Perez, and S. Zeadally. Qoe-based server selection for content distribution networks. IEEE Transactions on Computers, 63(11):2803–2815, 2014.

29 Q. Wu, Z. Du, P. Yang, Y.-D. Yao, and J. Wang. Traffic-aware online network selection in heterogeneous wireless networks. TVT, 65(1):381–397, 2016.

30 K. Zhu, D. Niyato, and P. Wang. Network selection in heterogeneous wireless networks: Evolution with incomplete information. In WCNC, pages 1–6. IEEE, 2010.
Appendix: Theoretical Results and Proofs

A.1 Lower bound on Worst Case Full Regret

We construct a proof using a deterministic oblivious adversary for full generality. Proofs using a randomized oblivious adversary or adaptive adversary are simpler. A deterministic adversary must select the full sequence of rewards prior to the first round. This is in contrast to an adaptive adversary, a more powerful adversary which is allowed to select rewards each round with full knowledge of the outcomes of random events occurring prior to the round.

Let $T$ be the number of time steps and $K$ be the number of arms. Fix an algorithm $a$. We show that there exists a problem instance (a predetermined sequence of rewards for each arm) such that algorithm $a$ obtains an expected full pseudo-regret of at least $T \frac{K-1}{K}$.

We construct a problem instance which, for each time step $t$ from 1 to $T$, has one arm $b_t \in [K]$ which gives a reward of 1, while all other arms give a reward of 0. We construct each $b_t$ based on the algorithm (but not on the algorithm’s choices) inductively as follows:

At the start of the algorithm, the algorithm plays arms with probabilities $p_1, \ldots, p_K$ respectively. Define $b_1$ to be the arm with the lowest probability of being played.

We maintain the following invariant with parameter $\tau \in [T]$ - when running the algorithm $a$ on the constructed problem instance from time steps 1 to $\tau$, the expected total reward $E[\sum_{t=1}^{\tau} x_{a(t)}(t)]$ by the algorithm is at most $\frac{\tau-1}{K}$. With the definition of $b_1$ above, we can see that this invariant holds for $\tau = 1$.

Now fix any later time step $\tau \in [T]$. The algorithm’s decision on time step $\tau$ can only be based on past rewards and the sequence of arms played by the algorithm on time steps up to $\tau - 1$. As our choices of $b_1$ to $b_{\tau-1}$ are fixed, the only randomness comes from the algorithm’s choices of arms up to this point. Let $\alpha = (a_1, a_2, \ldots, a_\tau)$ represent a possible sequence of arms played by the algorithm for the first $\tau - 1$ time steps. This event occurs with probability $Q_\alpha$, and the algorithm would accumulate a total reward of $R_\alpha$. Assuming by induction that the invariant holds on time steps up to $\tau - 1$, we can state that $\sum_{\alpha} Q_\alpha R_\alpha \leq \frac{\tau-1}{K}$.

Now on time step $\tau$, based on its past choices of arms $\alpha$, the algorithm constructs a probability vector $p_1^\tau, p_2^\tau, \ldots, p_K^\tau$ representing the probabilities of playing each arm on time step $\tau$. Now we construct $b_\tau \in [K]$ independently of $\alpha$. For any fixed $b_\tau$, the expected reward by the algorithm after time step $\tau$ is given by:

$$E[\sum_{t=1}^{\tau} x_{a(t)}(t)] = \sum_{\alpha} Q_\alpha (R_\alpha + 1 \times p_{b_\tau}^\alpha) = \sum_{\alpha} Q_\alpha R_\alpha + \sum_{\alpha} Q_\alpha p_{b_\tau}^\alpha$$

As we have the following:

$$\sum_{\alpha} Q_\alpha p_{b_1}^\alpha + \sum_{\alpha} Q_\alpha p_{b_2}^\alpha + \cdots + \sum_{\alpha} Q_\alpha p_{b_K}^\alpha = \sum_{\alpha} Q_\alpha (p_{b_1}^\alpha + p_{b_2}^\alpha + \cdots + p_{b_K}^\alpha) = \sum_{\alpha} Q_\alpha \times 1 = 1$$

there must exist a $b_\tau \in [K]$ such that $\sum_{\alpha} Q_\alpha p_{b_\tau}^\alpha \leq \frac{1}{K}$. By selecting such a $b_\tau$, we can conclude that:

$$E[\sum_{t=1}^{\tau} x_{a(t)}(t)] = \sum_{\alpha} Q_\alpha R_\alpha + \sum_{\alpha} Q_\alpha p_{b_\tau}^\alpha \leq \frac{\tau - 1}{K} + \frac{1}{K} = \frac{\tau}{K}$$

completing the inductive proof. Thus, we can lower bound the full pseudo-regret as follows:

$$R_{\text{full}}(T) = \sum_{t=1}^{T} \max_{a \in [K]} E_a [x_i(t)] - E_a \left[ \sum_{t=1}^{T} x_{a(t)}(t) \right] \geq \left( \sum_{t=1}^{T} 1 \right) - \frac{T}{K} = \frac{T(K - 1)}{K}$$
A.2 Optimized Periodic EXP4

In this section, we give an optimized implementation of Periodic EXP4 to show that the running time bounds can be improved upon with some pre-computation. With this optimization, we have a running time of $O(K \sum_{f \in F} P_f)$ for pre-computation, and $O(K|F|)$ per time step later on. However, we note that such an implementation can potentially increase the amount of numerical error. The reduction in running time largely comes from optimizing the following computation via the introduction of variables $S_f^j(t)$ and $B_f(t)$:

$$r_i(t) := \sum_{f \in F} \left( b_i^{f(t)},j(t) \prod_{\ell \in f(t) \setminus \{f(t)\}} b_{i,\ell}^{f(t)}(t) \right)$$

**Algorithm 2** Periodic EXP4

1: procedure **Initialization**
2:   for each $f \in F$ do
3:     for each $\ell \in f([T])$ do
4:     for each $i \in [K]$ do
5:         Initialize $b_i^{f,j}(1) = 1$
6:     Initialize $S_f^j(1) = K$ \> we maintain $S_f^j(t) = \sum_{j=1}^K b_i^{f,j}(t)$
7:     Initialize $B_f(1) = K|f([T])|$ \> we maintain $B_f(t) = \prod_{\ell \in f([t])} S_f^j(t)$

8: procedure **Algorithm**
9:   for each time step $t = 1, 2, \cdots, T$ do
10:      for each $i \in [K]$ do
11:         $r_i(t) := \sum_{f \in F} \left( b_i^{f(t)},j(t) \times B_f(t)/S_f^j(t) \right)$ \> running time $O(K|F|)$
12:      for each $i \in [K]$ do
13:         $p_i(t) = \frac{r_i(t)}{\sum_{j=1}^K r_j(t)}$ \> running time $O(K)$
14:         Play arm $i \in [K]$ from the probabilities $p_1(t), p_2(t), \cdots, p_K(t)$
15:         Obtain reward $x_i(t)$
16:      for each $f \in F$ do
17:         for each $\ell \in f([T])$ do
18:            for each $i \in [K]$ do
19:               if $i = i_\ell$ and $\ell = f(t)$ then
20:                  $b_i^{f,j}(t+1) = b_i^{f,j}(t) \exp(\frac{2}{\gamma} x_i(t)/p_i(t))$ \> running time $O(|F|)$
21:               else
22:                  $b_i^{f,j}(t+1) = b_i^{f,j}(t)$ \> implemented as no-op
23:         $S_f^j(t) := S_f^j(t) + b_i^{f,j}(t) - b_i^{f,j}(t-1)$ \> running time $O(|F|)$
24:         $B_f(t) := B_f(t-1) \times S_f^j(t)/S_f^j(t-1)$ \> running time $O(|F|)$

A.3 Correctness of Periodic EXP4

In this section, we complete the proof of correctness of Periodic EXP4 as mentioned in Section 5.3. As described before, we show that our algorithm produces the same probability...
distribution over arms as EXP4 in every time step. In EXP4, \( \pi_{\theta,f} \) is the expert which at time \( t \) recommends arm \( \theta \circ f(t) \) with probability 1 and all other arms with probability 0. We show that Periodic EXP4 is distributionally equivalent to EXP4, where \( \Pi = \{ \pi_{\theta,f} | f \in F, \theta \in \Theta_f \} \). In EXP4, Each expert \( \pi_{\theta,f} \) would have some weight \( w_{\theta,f}(t) \) at time step \( t \). At time step \( t \), EXP4 plays arm \( i \) with probability \( p_i(t) \) represented by the following expression:

\[
p_i(t) = \frac{\sum_{f \in F, \theta \in \Theta_f, \theta \circ f(t) = i} w_{\theta,f}(t)}{\sum_{f \in F, \theta \in \Theta_f} w_{\theta,f}(t)}
\]

To show that the two algorithms are distributionally equivalent, as \( p_i(t) = r_i(t)/\sum_{j=1}^{K} r_j(t) \) in Periodic EXP4, for each successive time step \( t \), we only need to show the following:

\[
r_i(t) = \sum_{f \in F, \theta \in \Theta_f, \theta \circ f(t) = i} w_{\theta,f}(t)
\]

\( r_i(t) \) is defined in Algorithm 1.

We first note that for each \( f \in [F], \ell \in f([T]), i \in [K] \), from the way \( b^{\ell,f}_i(t+1) \) is defined in Periodic EXP4 (Algorithm 1), we have the following expression:

\[
b^{\ell,f}_i(t+1) = \prod_{s \in f^{-1}((\ell) \cap [t])} \exp\left(\frac{\gamma}{K} \hat{x}_i(s)\right) = \exp\left(\frac{\gamma}{K} \sum_{s \in f^{-1}((\ell) \cap [t])} \hat{x}_i(s)\right)
\]

We then note that in EXP4,

\[
w_{\theta,f}(t+1) = \exp\left(\frac{\gamma}{K} \sum_{s \in [t]} \hat{x}_{\theta \circ f(s)}(s)\right)
\]

\[
= \exp\left(\frac{\gamma}{K} \sum_{f \in f([t]) \subseteq f^{-1}((\ell) \cap [t])} \sum_{s \in f(s)} \hat{x}_{\theta \circ f(s)}(s)\right)
\]

\[
= \exp\left(\frac{\gamma}{K} \sum_{f \in f([t]) \subseteq f^{-1}((\ell) \cap [t])} \sum_{s \in f(s)} \hat{x}_{\theta \circ f(s)}(s)\right)
\]

\[
= \prod_{f \in f([t]) \subseteq f^{-1}((\ell) \cap [t])} \exp\left(\frac{\gamma}{K} \sum_{s \in f(s)} \hat{x}_{\theta \circ f(s)}(s)\right)
\]

\[
= \prod_{f \in f([t]) \subseteq f^{-1}((\ell) \cap [t])} b^{\ell,f}_{\theta | f}(t+1)
\]

where \( b^{\ell,f}_{\theta | f}(t) \) comes from Periodic EXP4 (Algorithm 1). We then note that:

\[
\sum_{f \in F, \theta \in \Theta_f, \theta \circ f(t) = i} w_{\theta,f}(t) = \sum_{f \in F} \sum_{\theta \in \Theta_f, \theta \circ f(t) = i} w_{\theta,f}(t)
\]

and that, (the last step is as \( \Theta_f \) contains every function \( \theta : f([T]) \rightarrow [K] \))

\[
\sum_{\theta \in \Theta_f, \theta \circ f(t) = i} w_{\theta,f}(t) = \sum_{\theta \in \Theta_f, \theta \circ f(t) = i} \prod_{f \in f([t]) \subseteq f^{-1}((\ell) \cap [t])} b^{\ell,f}_{\theta | f}(t)
\]

By (3)

\[
= \sum_{\theta \in \Theta_f, \theta \circ f(t) = i} b^{f(t),f}_{\theta \circ f(t)}(t) \prod_{f \in f([t]) \subseteq f^{-1}((\ell) \cap [t])} b^{\ell,f}_{\theta | f}(t)
\]

Extract current label

\[
= b^{f(t),f}_{\theta \circ f(t)}(t) \prod_{f \in f([t]) \subseteq f^{-1}((\ell) \cap [t])} b^{\ell,f}_{\theta | f}(t)
\]

\[
= b^{f(t),f}_{\theta \circ f(t)}(t) \prod_{f \in f([t]) \subseteq f^{-1}((\ell) \cap [t])} b^{\ell,f}_{\theta | f}(t)
\]

\[
= b^{f(t),f}_{\theta \circ f(t)}(t) \prod_{f \in f([t]) \subseteq f^{-1}((\ell) \cap [t])} b^{\ell,f}_{\theta | f}(t)
\]
Thus we have,
\[
\sum_{f \in F, \theta \in \Theta, \theta \circ f(t) = i} w_{\theta,f}(t) = \sum_{f \in F} \left( b_0^f(t) \cdot f(t) \prod_{t \in f([\theta]) \setminus \{f(t)\}} \sum_{j=1}^K b_j^f(t) \right) = r_i(t)
\]
This shows that the expression for \( r_i(t) \) we have defined in Periodic EXP4 corresponds to the sum of weights of all the “experts” \( \pi_{\theta,f} \) from EXP4 which agree to play arm \( i \) on time step \( t \). Thus this concludes the proof of distributional equivalence between the algorithms.

### A.4 Worst Case Regret Lower Bound on a Single Partition

This is based on the pseudo-regret in the setting with a single partition function (1). We make use of a modified formulation by [23] of a theorem originally presented in [6].

- **Theorem 3.** [23, 6] Assume that the number of time steps \( T \geq K/(4 \ln 4) \). Then there exists a randomized oblivious adversary \( R \), such that for algorithms \( a \),

\[
\inf_{a} \left( \max_{t \in [K]} E \left[ \sum_{t \in [T]} x_i(t) \right] - E \left[ \sum_{t \in [T]} x_{a(t)}(t) \right] \right) \geq \sqrt{cKT}
\]

We note that this randomized oblivious adversary \( R \) picks arms independently of the choices made by algorithm \( a \). Details on the construction of this adversary \( R \) are given in [23]. \( c = (\sqrt{2} - 1)/\sqrt{32 \ln(4/3)} \) is a constant independent of any parameter.

We now proceed with the proof of Theorem 3. For each label \( \ell \in [P] \), we consider a bandit problem of length \( T_{\ell} := |f^{-1}(\ell)| \). If we assume each \( T_{\ell} \geq K/(4 \ln 4) \), then by Theorem 3 there exists an adversary \( R_{\ell} \) such that for any algorithm \( a_{\ell} \) running on a bandit problem of length \( T_{\ell} \),

\[
\max_{t \in [K]} E \left[ \sum_{t \in [T_{\ell}]} x_i(t) \right] - E \left[ \sum_{t \in [T_{\ell}]} x_{a_{\ell}(t)}(t) \right] \geq \sqrt{cKT_{\ell}}
\]

We now construct an adversary \( R \) for a bandit problem of length \( T \) on partition function \( f \). For each time step \( t \), let \( \ell := f(t) \). The adversary \( R \) takes \( R_{\ell} \)'s advice to generate a randomized reward vector for time step \( t \).

Now consider any algorithm \( a \) for a bandit problem of length \( T \) on partition function \( f \), and run it against the adversary \( R \). Suppose there exists a label \( \ell \in [P] \) such that:

\[
\max_{t \in [K]} E \left[ \sum_{t \in f^{-1}(\ell)} x_i(t) \right] - E \left[ \sum_{t \in f^{-1}(\ell)} x_{a(t)}(t) \right] < \sqrt{cKT_{\ell}}
\]

We can then consider a “restriction” \( a_{\ell} \) of algorithm \( a \) that plays on a bandit problem of length \( T_{\ell} := |f^{-1}(\ell)| \). This algorithm \( a_{\ell} \) would play exactly what algorithm \( a \) would play on the \( T_{\ell} \) time steps of label \( \ell \), while simulating \( a \)'s plays against adversary \( E \) internally on all other time steps. Therefore, against the adversary \( R_{\ell} \), the algorithm \( a_{\ell} \) would achieve a pseudo-regret under \( \sqrt{cKT} \), which contradicts our choice of adversary \( R_{\ell} \).
We can thus conclude that:

$$\max_{\theta \in \Theta} \mathbb{E} \left[ \sum_{t \in [T]} x_{\theta \circ f(t)}(t) \right] - \mathbb{E} \left[ \sum_{t \in [T]} x_{\pi}(t) \right]$$

$$= \max_{(i_1, i_2, \cdots, i_P) \in [K]^P} \left( \mathbb{E}\left[ \sum_{t \in f^{-1}(i_1)} x_i(t) \right] - \mathbb{E}\left[ \sum_{t \in f^{-1}(i_2)} x_i(t) \right] \right)$$

$$= \sum_{i \in [K]} \max_{P \in [P]} \mathbb{E}\left[ \sum_{t \in f^{-1}(i)} x_i(t) \right] - \mathbb{E}\left[ \sum_{t \in f^{-1}(P)} x_{\pi}(t) \right]$$

$$\geq \sum_{P \in [P]} \sqrt{cKT}$$

### A.5 Lower Bound on Worst Case Generalized Periodic Regret

We show a lower bound on the worst-case pseudo-regret in the generalized periodic setting based on $T$, $K$, $|F|$ and $P$. In order to make use of Theorem 2 later on in the proof, we first make the base assumption that $T$ is sufficiently large. Specifically, we require that:

$$T \geq \left( \frac{\log |F|}{\log K} + \frac{0.5 \log 0.5}{\log K} \right) \times K/(4 \ln \frac{3}{4})$$

Fix any integer $M \geq P$. Suppose $K \leq P$ (more labels than arms). We split the time steps $[T]$ into $M$ equally sized sections. Now we let $F$ be the set of all partitions of $|M| = \{1, 2, \cdots, M\}$ into $K$ parts. As $K \leq P$, we can define $F$ this way. Each partition in $F$ assigns one label in $[K] = \{1, 2, \cdots, K\}$ to each of the $M$ sections.

#### Lemma 4.

This set of partitions $F$ covers all possible ways to assign a different arm to each of the $M$ parts.

**Proof.** Consider any assignment $\pi : [M] \rightarrow [K]$, representing each possible assignment of arms to the $M$ parts. We have a partition function $f \in F$ that partitions $[M]$ into the pre-image sets $\pi^{-1}(1), \pi^{-1}(2), \cdots, \pi^{-1}(K)$ of $\pi$ by assigning each a separate label. The labels can then be assigned to arms accordingly to represent $\pi$. ⊣

As OPT can choose an arm for each of the $M$ sections independently, we obtain a regret lower bound of $\Omega(M \times \sqrt{\frac{K}{M}}) = \Omega(\sqrt{MKT})$ by Theorem 2. Now we express $M$ in terms of $|F|$. As $|F|$ is the number of partitions of $[M]$ into $K$ parts, we have $|F| = S(M, K)$, where $S(M, K)$ refers to the Stirling numbers of the second kind. Using the upper and lower bounds for $S(M, K)$ from [21], we can bound $|F|$, and thus $M$, as follows:

$$\frac{1}{2} \left( \begin{array}{c} M \\ K \end{array} \right) K^{M-K} \leq |F| \leq \frac{1}{2} \left( K^2 + K + 2 \right) K^{M-K-1} - 1$$

$$\Rightarrow \frac{1}{2} K^{M-K} \leq |F| \leq \frac{1}{2} \left( K^2 + K^2 + 2K^2 \right) K^{M-K-1} = 2K^{M-K+1}$$

$$\Rightarrow \log |F| \log K + (K-1) - \frac{\log 2}{\log K} \leq M \leq \frac{\log |F|}{\log K} + K - \frac{\log 0.5}{\log K}$$

This expression validates our use of Theorem 2 as this with our assumption implies that $T/M \geq K/(4 \ln \frac{3}{4})$, and thus for any $f \in F$, there are at least $K/(4 \ln \frac{3}{4})$ time steps associated with each label. From the bounds on $M$, we have that $M = \Theta\left( \frac{\log |F|}{\log K} + K \right)$. This thus gives us a lower bound of $\Omega\left( \sqrt{\frac{KT}{\log K} + K} \right)$.

Now, we can add a single extra partition function to $F$, that partitions the time steps $[T]$ into $P$ equal parts. With the use of Theorem 2 again (note that $T/P \geq T/M \geq K/(4 \ln \frac{3}{4})$),
the presence of this partition function in $F$ makes our lower bound no smaller than $\Omega(\sqrt{P KT})$. Therefore, we obtain a final lower bound of:

$$\Omega\left(\sqrt{P KT} + \sqrt{KT \frac{\log |F|}{\log K}}\right)$$

$$= \Omega\left(\sqrt{P KT} + \sqrt{KT \frac{\log |F|}{\log K}}\right) \text{ (as } K \leq P\).$$

If $P < K$ instead, a simple lower bound can be obtained by using only $P$ out of the $K$ arms, so we obtain a problem with $P$ arms and maximum partition size $P$. This gives us a lower bound of $\Omega\left(\sqrt{P KT} + \sqrt{PT \frac{\log |F|}{\log P}}\right)$. We can then merge these two lower bounds into a single expression $\Omega\left(\sqrt{P KT} + \sqrt{\min(P, K)T \frac{\log |F|}{\log \min(P, K)}}\right)$.

**B** Appendix: Experiments

**B.1** Learning Patterns

Figure 6 illustrates how the average probabilities of the three networks vary with time in the discrete setup. The conclusions we can draw are the same as with the continuous setup. We only include this figure here for completeness.

![EXP3: Combined probabilities of each network over first 10 iterations.](image1)

![Bandwidth ratio](image2)

Figure 6b shows the actual ratio of the bandwidths of the three networks within any one iteration.

**B.2** Noisy Settings

In this experiment, we apply a 10% Gaussian noise to each of the networks’ data rates in the instances given in Figure 2. These noisy instances are illustrated in Figure 7. Figure 8 shows that Periodic EXP4 is largely unaffected by noise in the data, giving similar results with and without noise.
Periodic Bandits and Wireless Network Selection

![Data rate vs Time slot](image1)

(a) Continuous, 10% gaussian noise.

(b) Discrete, 10% gaussian noise.

**Figure 7** Variation of data rates with within one iteration in each setup with noise. The original instances without noise are given in Figure 2.

![Distance to Optimal minimum vs Iteration](image2)

(a) Continuous, no noise.

(b) Continuous, 10% gaussian noise.

(c) Discrete, no noise.

(d) Discrete, 10% gaussian noise.

**Figure 8** Distance to Optimal minimum plotted over each iteration for each setup.

### B.3 Comparison of Different Periods

It is useful to run Periodic EXP4 with multiple periods, especially in cases where the best period is not obvious. The is the case if, for example, the bit rates (or more generally, rewards of pulling arms) vary continuously with time.

Figures 9 and 10 compares the performance of Periodic EXP4 when run with different period sets. The period sets chosen are \{1\}, \{4\}, \{1, 2, \cdots, 15\}, \{1, 2, \cdots, 24\} and \{1, 2, \cdots, 45\}. Running Periodic EXP4 with period set \{1\} is equivalent to running EXP3, and running with period set \{4\} is equivalent to running four separate instance of EXP3, switching between the four instances every quarter of an iteration. Period set \{4\} is used for comparison as the discrete setting we use has a known “best period” of 4. It is important to note, however, that the “best period” is often not known prior to the experiment in practice. We also include the Optimal Random player as a point of comparison.

From the results on the discrete setting (Figure 9a), we can see period set \{4\} immediately taking the lead, and having significantly better performance than other period sets. This is to be expected, as in this case, the algorithm does not need to figure out what the “right period” is. Period set \{4\} however performs poorly on the continuous settings.

To better illustrate the utility of larger period sets, we run another experiment on a more complex continuous setting (Figure 10a), with bandwidths that fluctuate more than
Figure 9 Comparison of different period sets on the discrete and continuous settings.

(a) Discrete setting from Figure 2a.

(b) Continuous setting from Figure 2b.

Figure 10 Comparison of different period sets on a continuous setting with more bandwidth fluctuations. This is run for 100 iterations, a little over three months.

(a) “Harder” continuous setting with more bandwidth fluctuations.

(b) Comparison on the “harder” continuous setting.

In summary, larger period sets have greater utility when more fluctuations in bandwidths (or rewards) are expected, as it allows the algorithm to more closely match the target pattern. However, a larger period set means the algorithm will take a longer time to learn. This is in line with the usual trade-offs between flexibility and efficiency in machine learning problems.

B.4 Performance Comparison of Algorithms.

Figure 11 provides the complete distribution of the cumulative gain each device observes when using Periodic EXP4, EXP3 or Optimal Random. Table [1] lists the median and standard deviation for each algorithm. The median and standard deviation values are computed over 20 runs with 20 devices each, for a total of 400 data points. All three have very similar median cumulative gains. We do note however, that the median cumulative gain is unlikely to indicate much, as it is likely to be close to the average cumulative gain, which as mentioned before, is not very useful as a metric as it is maximized whenever there is at least one device
in each network. On the other hand, we can see that Periodic EXP4 has a lower variance than EXP3, which suggests that Periodic EXP4 divides the bandwidths more evenly between the devices.

Table 1: Median and standard deviation of cumulative download (GB) per device, when using each of the three algorithms, namely Periodic EXP4, EXP3 and Optimal Random.

|            | Periodic EXP4 | EXP3 | Optimal Random |
|------------|---------------|------|----------------|
| Median (GB)| 34.78         | 35.08| 34.67          |
| Standard deviation (GB) | 0.20 | 2.72 | 0.04 |

Figure 11: Cumulative gain (GB) of a single device when using each of the algorithms, taking into account all devices across all runs — the box represents the interquartile range (middle 50% cumulative gains), the vertical line in the box denotes the median, the whiskers show the range of the remaining cumulative gains, excluding outliers shown as dots.

B.5 Mobility of Users

We consider a similar setting to Figure 1, with 20 mobile users (each with a mobile device) and 9 networks. Devices have access to a different set of networks over time depending on their locations. We divide an iteration into 6 phases, based on the mobility of users, as listed in Table 2. In phase 1, all the mobile users are at home (same building with common networks); 10 devices have access to networks 1, 2 and 3 while the remaining have access to networks 1 and 2. We assume 5 users are always at home (whose devices have access to networks 1, 2 and 3 — hence, they might need to switch networks when the others go out). The rest spend 13 hours at home. In phase 2, they travel (one hour) together to office, during which they have access to networks 4 and 5. In phase 3, they have access to networks 6, 7 and 8 at the office. After 3 hours of work, in phase 4, 10 users go out for a 1-hour lunch during which they have access to networks 8 and 9. After lunch, they spend another 5 hours at the office (phase 5), before travelling home for an hour in phase 6.

Table 2: Phases during one iteration, the time slots delimiting each phase (relative to the first time slot of the iteration), and the list of networks available to each device during every phase.

| Phase(s) | 1 | 2 and 6 | 3 and 5 | 4 |
|----------|---|---------|---------|---|
| Time slots delimiting each phase | 1 - 780 | 781 - 840 | 841 - 1020 | 1021 - 1080 |
| 1381 - 1440 | 1081 - 1380 |
| Device(s): their list of available networks | 1 - 10: 1, 2, 3 | 1 - 5: 1, 2, 3 | 1 - 5: 1, 2, 3 | 1 - 5: 1, 2, 3 |
| 11 - 20: 1, 2 | 6 - 20: 4, 5 | 6 - 20: 6, 7, 8 | 6 - 11: 6, 7, 8 |
| 11 - 20: 8, 9 |

Vanilla Periodic EXP4 is oblivious to networks possibly becoming unavailable, and we
give it a gain of zero whenever it decides to select an inaccessible network, in the hope that it also learns the pattern of network availability. We compare it against an optimized version of Periodic EXP4 which selects only from the set of currently available networks. Figure 12 shows that, as expected, the optimized version initially yields a better performance. However, after the Vanilla Periodic EXP4 algorithm learns the pattern, they both perform equally well. The intuition for a slightly better performance compared to Optimal Random is that when using Periodic EXP4, many of the devices eventually converge to selecting a single network with high probability, and are thus less likely to make a random ‘bad’ selection. On the other hand, Optimal Random has every device running the exact same (though distributionally optimal) probability distribution, leading to the occasional random event where too many devices select the same network simultaneously.

![Figure 12](image-url) Distance to Optimal minimum in the first and last repetitions of the mobility setting. Vertical lines indicate the time of any change in the environment.