NeurWIN: Neural Whittle Index Network For Restless Bandits Via Deep RL

Anonymous Author(s)
Affiliation
Address
email

Abstract

Whittle index policy is a powerful tool to obtain asymptotically optimal solutions for the notoriously intractable problem of restless bandits. However, finding the Whittle indices remains a difficult problem for many practical restless bandits with convoluted transition kernels. This paper proposes NeurWIN, a neural Whittle index network that seeks to learn the Whittle indices for any restless bandits by leveraging mathematical properties of the Whittle indices. We show that a neural network that produces the Whittle index is also one that produces the optimal control for a set of Markov decision problems. This property motivates using deep reinforcement learning for the training of NeurWIN. We demonstrate the utility of NeurWIN by evaluating its performance for three recently studied restless bandit problems. Our experiment results show that the performance of NeurWIN is significantly better than other RL algorithms.

1 Introduction

Many sequential decision problems can be modeled as multi-armed bandit problems. A bandit problem models each potential decision as an arm. In each round, we play M arms out of a total of N arms by choosing the corresponding decisions. We then receive a reward from the played arms. The goal is to maximize the expected long-term total discounted reward. Consider, for example, displaying advertisements on an online platform with the goal to maximize the long-term discounted click-through rates. This can be modeled as a bandit problem where each arm is a piece of advertisement and we choose which advertisements to be displayed every time a particular user visits the platform. It should be noted that the reward, i.e., click-through rate, of an arm is not stationary, but depends on our actions in the past. For example, a user that just clicked on a particular advertisement may be much less likely to click on the same advertisement in the near future. Such a problem is a classic case of the *restless bandit problem*, where the reward distribution of an arm depends on its state, which changes over time based on our past actions.

The restless bandit problem is notoriously intractable [16]. Most recent efforts, such as *recovering bandits* [17], *rotting bandits* [19], and *Brownian bandits* [20], only study some special instances of the restless bandit problem. The fundamental challenge of the restless bandit problem lies in the explosion of state space, as the state of the entire system is the Cartesian product of the states of individual arms. A powerful tool to address the explosion of state space is the Whittle index policy [26]. In a nutshell, the Whittle index policy calculates a Whittle index for each arm based on the arm's current state, where the index loosely corresponds to the amount of cost that we are willing to pay to play the arm, and then plays the arm with the highest index. It has been shown that the Whittle index policy is either optimal or asymptotically optimal in many settings.

In this paper, we present Neural Whittle Index Network (NeurWIN), a principled machine learning approach that finds the Whittle indices for virtually all restless bandit problems. We note that the

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

Whittle index is an artificial construct that cannot be directly measured. Finding the Whittle index is typically intractable. As a result, the Whittle indices of many practical problems remain unknown except for a few special cases.

We are able to circumvent the challenges of finding the Whittle indices by leveraging an important mathematical property of the Whittle index: Consider an alternative problem where there is only one arm and we decide whether to play the arm in each time instance. In this problem, we need to pay a constant cost of λ every time we play the arm. The goal is to maximize the long-term discounted net reward, defined as the difference between the rewards we obtain from the arm and the costs we pay to play it. Then, the optimal policy is to play the arm whenever the Whittle index becomes larger than λ. Based on this property, a neural network that produces the Whittle index can be viewed as one that finds the optimal policy for the alternative problem for any λ.

Using this observation, we propose a deep reinforcement learning method to train NeurWIN. To demonstrate the power of NeurWIN, we employ NeurWIN for three recently studied restless bandit problems, namely, recovering bandit [17], wireless scheduling [1], and stochastic deadline scheduling [29]. We compare NeurWIN against four other reinforcement learning algorithms and the application-specific baseline policies in the respective restless bandit problems. Experiment results show that the index policy using our NeurWIN significantly outperforms other reinforcement learning algorithms. Morover, for problems where the Whittle indices are known, NeurWIN has virtually the same performance as the corresponding Whittle index policy, showing that NeurWIN indeed learns a precise approximation to the Whittle indices.

The rest of the paper is organized as follows: Section 2 reviews related literature. Section 3 provides formal definitions of the Whittle index and our problem statement. Section 4 introduces our training algorithm for NeurWIN. Section 5 demonstrates the utility of NeurWIN by evaluating its performance under three recently studied restless bandit problems. Finally, Section 6 concludes the paper.

2 Related work

Restless bandit problems were first introduced in [26]. They are known to be intractable, and are in general PSPACE hard [16]. As a result, many studies focus on finding the Whittle index policy for restless bandit problems, such as in [12, 5, 14, 22]. However, these studies are only able to find the Whittle indices under various specific assumptions about the bandit problems.

There has been a lot of studies on applying RL methods for bandit problems. [6] proposed a tool called Uniform-PAC for contextual bandits. [30] described a framework-agnostic approach towards guaranteeing RL algorithms' performance. [10] introduced contextual decision processes (CDPs) that encompass contextual bandits for RL exploration with function approximation. [18] compared deep neural networks with Bayesian linear regression against other posterior sampling methods. However, none of these studies are applicable to restless bandits, where the state of an arm can change over time.

Deep RL algorithms have been utilized in problems that resemble restless bandit problems, including HVAC control [25], cyber-physical systems [13], and dynamic multi-channel access [24]. In all these cases, a major limitation for deep RL is scalability. As the state spaces grows exponentially with the number of arms, these studies can only be applied to small-scale systems, and their evaluations are limited to cases when there are at most 5 zones, 6 sensors, and 8 channels, respectively.

An emerging research direction is applying machine learning algorithms to learn Whittle indices. [3] proposed employing the LSPE(0) algorithm [28] coupled with a polynomial function approximator. The approach was applied in [2] for scheduling web crawlers. However, this work can only be applied to restless bandits whose states can be represented by a single number, and it only uses a polynomial function approximator, which may have low representational power [21]. [8] proposed a Q-learning based heuristic to find Whittle indices. However, as shown in its experiment results, the heuristic may not produce Whittle indices even when the training converges.

3 Problem Setting

In this section, we provide a brief overview of restless bandit problems and the Whittle index. We then formally define the problem statement.

3.1 Restless Bandit Problems

A restless bandit problem consists of N restless arms. In each round t, a control policy π observes the state of each arm i, denoted by $s_i[t]$, and selects M arms to activate. We call the selected arms as *active* and the others as *passive*. We use $a_i[t]$ to denote the policy's decision on each arm i, where $a_i[t] = 1$ if the arm is active and $a_i[t] = 0$ if it is passive at round t. Each arm i generates a stochastic reward $r_i[t]$ with distribution $R_{i,act}(s_i[t])$ if it is active, and with distribution $R_{i,pass}(s_i[t])$ if it is passive. The state of each arm i in the next round evolves by the transition kernel of either $P_{i,act}(s_i[t])$ or $P_{i,pass}(s_i[t])$, depending on whether the arm is active. The goal of the control policy is to maximize the expected total discounted reward, which can be expressed as $\mathbb{E}_\pi[\sum_{t=0}^{\infty}\sum_{i=1}^{N}\beta^t r_i[t]]$ with β being the discount factor.

A control policy is effectively a function that takes the vector $(s_1[t], s_2[t], \ldots, s_N[t])$ as the input and produces the vector $(a_1[t], a_2[t], \ldots, a_N[t])$ as the output. It should be noted that the space of input is exponential in N. If each arm can be in one of K possible states, then the number of possible inputs is K^N. This feature, which is usually referred to as the curse of dimensionality, makes finding the optimal control policy intractable.

3.2 The Whittle Index

An index policy seeks to address the curse of dimensionality through decomposition. In each round, it calculates an index, denoted by $W_i(s_i[t])$, for each arm i based on its current state. The index policy then selects the M arms with the highest indices to activate. It should be noted that the index of an arm i is independent from the states of any other arms. In this sense, learning the Whittle index of a restless arm is an *auxiliary task* to finding the control policy for restless bandits.

Obviously, the performance of an index policy depends on the design of the index function $W_i(\cdot)$. A popular index with solid theoretical foundation is the Whittle index, which is defined below. Since we only consider one arm at a time, we drop the subscript i for the rest of the paper.

Consider a system with only one arm, and an activation policy that determines whether to activate the arm in each round t. Suppose that the activation policy needs to pay an activation cost of λ every time it chooses to activate the arm. The goal of the activation policy is to maximize the total expected discounted net reward, $\mathbb{E}[\sum_{t=0}^{\infty}\beta^t(r[t] - \lambda a[t])]$. The optimal activation policy can be expressed by the set of states in which it would activate this arm for a particular λ, and we denote this set by $\mathcal{A}(\lambda)$. Intuitively, the higher the cost, the less likely the optimal activation policy would activate the arm in a given state, and hence the set $\mathcal{A}(\lambda)$ would decrease monotonically. When an arm satisfies this intuition, we say that the arm is *indexable*.

Definition 1 (Indexability)**.** *An arm is said to be indexable if $\mathcal{A}(\lambda)$ decreases monotonically from the set of all states to the empty set as λ increases from $-\infty$ to ∞. A restless bandit problem is said to be indexable if all arms are indexable.*

Definition 2 (The Whittle Index)**.** *If an arm is indexable, then its Whittle index of each state s is defined as $W(s) := \sup_\lambda\{\lambda : s \in \mathcal{A}(\lambda)\}$.*

Even when an arm is indexable, finding its Whittle index can still be intractable, especially when the transition kernel of the arm is convoluted.[1] Our NeurWIN finds the Whittle index by leveraging the following property of the Whittle index: Consider the single-armed bandit problem. Suppose the initial state of an indexable arm is s at round one. Consider two possibilities: The first is that the activation policy activates the arm at round one, and then uses the optimal policy starting from round two; and the second is that the activation policy does not activate the arm at round one, and then uses the optimal policy starting from round two. Let $Q_{\lambda,act}(s)$ and $Q_{\lambda,pass}(s)$ be the expected discounted net reward for these two possibilities, respectively, and let $D_s(\lambda) := \big(Q_{\lambda,act}(s) - Q_{\lambda,pass}(s)\big)$ be their difference. Clearly, the optimal activation policy should activate an arm under state s and activation cost λ if $D_s(\lambda) \geq 0$. We then have the following:

Theorem 1. *[31, Thm 3.14] If an arm is indexable, then, for every state s, $D_s(\lambda) \geq 0$ if and only if $\lambda \leq W(s)$.*

[1] [15] described a generic approach for finding the Whittle index. The complexity of this approach is at least exponential to the number of states.

Our NeurWIN uses Thm. 1 to train neural networks that predict the Whittle index for any indexable arms. From Def. 1, a sufficient condition for indexability is when $D_s(\lambda)$ is a decreasing function. Thus, we define the concept of *strong indexability* as follows:

Definition 3 (Strong Indexability)**.** *An arm is said to be strongly indexable if $D_s(\lambda)$ is strictly decreasing in λ for every state s.*

Intuitively, as the activation cost increases, it becomes less attractable to activate the arm in any given state. Hence, one would expect $D_s(\lambda)$ to be strictly decreasing in λ for most practical problems. In the next section, we introduce an example of restless bandit problem and prove that it is strongly indexable. In Section 5, we will further use numerical results to show that all three applications that we evaluate in this paper are strongly indexable.

3.3 Example: Bandit with Strong Indexability

A recent study [29] proposes a deadline scheduling problem for the scheduling of electrical vehicle charging stations. In this problem, a charging station has N charging spots and enough power to charge M vehicles in each round. When a charging spot is available, a new vehicle may join the system and occupy the spot. Upon occupying the spot, the vehicle announces the time that it will leave the station and the amount of electricity that it needs to be charged. The charging station obtains a reward of $1 - c$ for each unit of electricity that it provides to a vehicle. However, if the station cannot fully charge the vehicle by the time it leaves, then the station needs to pay a penalty of $F(B)$, where B is the amount of unfulfilled charge. The goal of the station is to maximize its net reward, defined as the difference between the amount of reward and the amount of penalty.

In this problem, each charging spot is an arm. The state $s[t]$, action $a[t]$, reward $r[t]$, and next state $s[t+1]$ of one arm can be modeled as below:

State $s[t]$: The state of an arm is denoted by a pair of integers $(D[t], B[t])$, where $B[t]$ is the amount of electricity that the vehicle still needs and $D[t]$ is the time until the vehicle leaves the station. When a charging spot is available, its state is $(0, 0)$.

Action $a[t]$: An arm is activated, i.e. $a[t] = 1$, if the agent decides to charge the vehicle on the corresponding spot.

Reward $r[t]$: The amount of net reward $r[t]$ depends on both the reward of charging the vehicle, which occurs when $a[t] = 1$, and the penalty of failing to fully charge a vehicle by the time it leaves, which occurs when t is the last slot before the vehicle leaves and $D[t] = 1$. Hence, $r[t]$ is written as,

$$r[t] = \begin{cases} (1-c)a[t], & \text{if } B[t] > 0, D[t] > 1, \\ (1-c)a[t] - F(B[t] - a[t]), & \text{if } B[t] > 0, D[t] = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{1}$$

Next state $s[t+1]$: If there is a vehicle at the spot, then $D[t]$ decreases by one in each slot. $B[t]$ decreases by one the vehicle is being charged in time t, i.e., $a[t] = 1$. When there is no vehicle at the spot, then there can be new vehicle arriving at the spot randomly. We use $Q(D, B)$ to be the probability that a vehicle with remaining charge B and time-to-deadline D arrives at an empty spot. We then have

$$s[t+1] = \begin{cases} (D[t]-1, B[t]-a[t]) & \text{if } D[t] > 1, \\ (D, B) \text{ w.p. } Q(D, B) & \text{if } D[t] \leq 1. \end{cases} \tag{2}$$

The recent study [29] has derived the precise characterization of the Whittle index, which we refer to as the deadline Whittle index. We further demonstrate that the deadline scheduling problem is strongly indexable.

Theorem 2. *The restless bandit for the deadline scheduling problem is strongly indexable.*

Proof. See Appendix A. □

3.4 Problem Statement

We now formally describe the objective of this paper. We assume that we are given a simulator of one single restless arm as a black box. The simulator provides two functionalities: First, it allows us

to set the initial state of the arm to any arbitrary state s. Second, in each round t, the simulator takes $a[t]$, the indicator function that the arm is activated, as the input and produces the next state $s[t+1]$ and the reward $r[t]$ as the outputs.

Our goal is to derive low-complexity index algorithms for restless bandit problems by training a neural network that approximates the Whittle index of each restless arm using its simulator. A neural network takes the state s as the input and produces a real number $f_\theta(s)$ as the output, where θ is the vector containing all weights and biases of the neural network. Recall that $W(s)$ is the Whittle index of the arm. We aim to find appropriate θ that makes $|f_\theta(s) - W(s)|$ small for all s. Such a neural network is said to be *Whittle-accurate*.

Definition 4 (Whittle-accurate)**.** *A neural network with parameters θ is said to be γ-Whittle-accurate if $|f_\theta(s) - W(s)| \leq \gamma$, for all s.*

4 NeurWIN Algorithm: Neural Whittle Index Network

In this section, we present NeurWIN, a deep-RL algorithm that trains neural networks to predict the Whittle indices. Since the Whittle index of an arm is independent from other arms, NeurWIN trains one neural network for each arm independently. In this section, we discuss how NeurWIN learns the Whittle index for one single arm.

4.1 Conditions for Whittle-Accurate

Before presenting NeurWIN, we first discuss the conditions for a neural network to be γ-Whittle-accurate.

Suppose we are given a simulator of an arm and a neural network with parameters θ. We can then construct an environment of the arm along with an activation cost λ as shown in Fig. 1. In each round t, the environment takes the real number $f_\theta(s[t])$ as the input. The input is first fed into a step function to produce $a[t] = 1\big(f_\theta(s[t]) \geq \lambda\big)$, where $1(\cdot)$ is the indicator function. Then, $a(t)$ is fed into the simulator of the arm to produce $r[t]$ and $s[t+1]$. Finally, the environment outputs the net reward $r[t] - \lambda a[t]$ and the next state $s[t+1]$. We call this environment $Env(\lambda)$. Thus, the neural network can be viewed as a controller for $Env(\lambda)$. The following corollary is a direct result from Thm. 1.

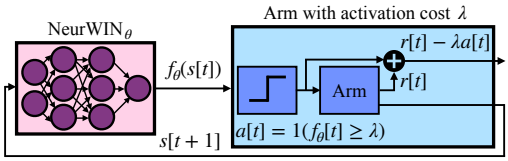

Figure 1: An illustrative motivation of NeurWIN.

Corollary 1. *If $f_\theta(s) = W(s), \forall s$, then the neural network with parameters θ is the optimal controller for $Env(\lambda)$, for any λ and initial state $s[0]$. Moreover, given λ and $s[0]$, the optimal discounted net reward is $\max\{Q_{\lambda,act}(s[0]), Q_{\lambda,pass}(s[0])\}$.*

Corollary 1 can be viewed as a necessary condition for a neural network to be 0-Whittle-accurate. Below, we establish a sufficient condition for γ-Whittle-accuracy.

Theorem 3. *If the arm is strongly indexable, then for any $\gamma > 0$ and an arbitrarily small positive constant δ, there exists a positive ϵ such that the following statement holds:*

If, for any states s_0, s_1 and any activation cost $\lambda \in [f_\theta(s_0) - \delta, f_\theta(s_0) + \delta]$, the discounted net reward of applying a neural network to $Env(\lambda)$ with initial state s_1 is at least $\max\{Q_{\lambda,act}(s_1), Q_{\lambda,pass}(s_1)\} - \epsilon$, then the neural network is γ-Whittle-accurate.

Proof. See Appendix B. □

4.2 Training Procedures for NeurWIN

Thm. 3 states that a neural network that yields near-optimal net reward for any environment $Env(\lambda)$ is also Whittle-accurate. This observation motivates the usage of deep reinforcement learning to find Whittle-accurate neural networks. To make the output of the environments differentiable with respect to the input $f_\theta(s[t])$, we replace the step function in Fig. 1 with a sigmoid function

Algorithm 1 NeurWIN Training

Input: Parameters θ, discount factor $\beta \in (0,1)$, learning rate L, sigmoid parameter m, mini-batch size R.
Output: Trained neural network parameters θ^+.
for each mini-batch b **do**
  Choose two states s_0 and s_1 uniformly at random, and set $\lambda \leftarrow f_\theta(s_0)$ and $\bar{G}_b \leftarrow 0$
  for each episode e in the mini-batch **do**
    Set the arm to state s_1, and set $h_e \leftarrow 0$
    for each round t in the episode **do**
      Choose $a[t] = 1$ w.p. $\sigma_m(f_\theta(s[t]) - \lambda)$, and $a[t] = 0$ w.p. $1 - \sigma_m(f_\theta(s[t]) - \lambda)$
      if $a[t] = 1$ **then**
        $h_e \leftarrow h_e + \nabla_\theta \ln(\sigma_m(f_\theta(s[t]) - \lambda))$
      else
        $h_e \leftarrow h_e + \nabla_\theta \ln(1 - \sigma_m(f_\theta(s[t]) - \lambda))$
      end if
    end for
    $G_e \leftarrow$ empirical discounted net reward in episode e
    $\bar{G}_b \leftarrow \bar{G}_b + G_e/R$
  end for
  $L_b \leftarrow$ learning rate in mini-batch b
  Update parameters through gradient ascent $\theta \leftarrow \theta + L_b \sum_e (G_e - \bar{G}_b) h_e$
end for

$\sigma_m(f_\theta(s[t]) - \lambda) := \big(1 + exp(-m(f_\theta(s[t]) - \lambda))\big)^{-1}$, where m is a sensitivity parameter. The environment then chooses $a[t] = 1$ with probability $\sigma_m(f_\theta(s[t]) - \lambda)$, and $a[t] = 0$ with probability $1 - \sigma_m(f_\theta(s[t]) - \lambda)$. We call this differentiable environment $Env^*(\lambda)$.

Our training procedure consists of multiple mini-batches, where each mini-batch is composed of a R episodes. At the beginning of each mini-batch, we randomly select two states s_0 and s_1. Motivated by the condition in Thm. 3, we consider the environment $Env^*(f_\theta(s_0))$ with initial state s_1 and aim to improve the empirical discounted net reward of applying the neural network to such an environment.

Our approach is based on the REINFORCE algorithm [27]. In each episode e, we set $\lambda = f_\theta(s_0)$ and initial state to be s_1. We then apply the neural network with parameters θ to $Env^*(\lambda)$ and observe the sequences of actions $\big(a[1], a[2], \dots\big)$ and states $\big(s[1], s[2], \dots\big)$. We can use these sequences to calculate their gradients with respect to θ through backward propagation, which we denote by h_e. We also observe the discounted net reward and denote it by G_e. After all episodes in the mini-batch finish, we calculate the average of all G_e as a bootstrapped baseline and denote it by $\bar{G}_b$. Finally, we do a weighted gradient ascent with the weight for episode e being its offset net reward, $G_e - \bar{G}_b$. When the step size is chosen appropriately, the neural network will be more likely to follow the sequences of actions of episodes with larger G_e after the weighted gradient ascent, and thus will have a better empirical discounted net reward. The complete algorithm is described in Alg. 1.

5 Experiments

5.1 Overview

In this section, we demonstrate NeurWIN's utility by evaluating it under three recently studied applications of restless bandit problems. In each application, we consider that there are N arms and a controller can play M of them in each round. We evaluate three different pairs of (N, M): $(4, 1)$, $(10, 1)$, and $(100, 25)$, and average the results of 50 independent runs when the problems are stochastic. Some applications consider that different arms can have different behaviors. For such scenarios, we consider that there are multiple types of arms and train a separate NeurWIN for each type. During testing, the controller calculates the index of each arm based on the arm's state and schedules the M arms with the highest indices.

The performance of NeurWIN is compared against the proposed policies in the respective recent studies. In addition, we also evaluate the REINFORCE [27], Wolpertinger-DDPG (WOLP-DDPG)

[7], Amortized Q-learning (AQL) [23], and QWIC [8]. REINFORCE is a classical policy-based RL algorithm. Both WOLP-DDPG and AQL are model-free deep RL algorithms meant to address problems with big action spaces. All three of them view a restless bandit problem as a Markov decision problem. Under this view, the number of states is exponential in N and the number of possible actions is $\binom{N}{M}$, which can be as large as $\binom{100}{25} \approx 2.4 \times 10^{23}$ in our setting. Neither REINFORCE nor WOLP-DDPG can support such a big action space, so we only evaluate them for $(N, M) = (4, 1)$ and $(10, 1)$. On the other hand, QWIC aims to find the Whittle index through Q-learning. It is a tabular RL method and does not scale well as the state space increases. Thus, we only evaluate QWIC when the size of the state space is less than one thousand. We use open-source implementations for REINFORCE [9] and WOLP-DDPG [4].

In addition, we use experiment results to evaluate two important properties. First, we evaluate whether these three applications are strongly indexable. Second, we evaluate the performance of NeurWIN when the simulator does not perfectly capture the actual behavior of an arm.

We use the same neural network architecture for NeurWIN in all three applications. The neural network is a fully connected one that consists of one input layer, one output layer, and two hidden layers. There are 16 and 32 neurons in the two hidden layers. The output layer has one neuron, and the input layer size is the same as the dimension of the state of one single arm. As for the REINFORCE, WOLP-DDPG, AQL algorithms, we choose the neural network architectures so that the total number of parameters is slightly more than N times as the number of parameters in NeurWIN to make a fair comparison. ReLU activation function is used for the two hidden layers. An initial learning rate $L = 0.001$ is set for all cases, with the Adam optimizer [11] employed for the gradient ascent step. The discount factor is $\beta = 0.99$ with an episode horizon of 300 timesteps. Each mini-batch consists of five episodes. More details can be found in the appendix.

5.2 Deadline Scheduling

This is the problem discussed in Section 3.3. We use exactly the same setting as in the recent study [29] for our experiment. In this problem, the state of an arm is denoted by a pair of integers (D, B) with $B \leq 9$ and $D \leq 12$. The size of state space is 120 for each arm.

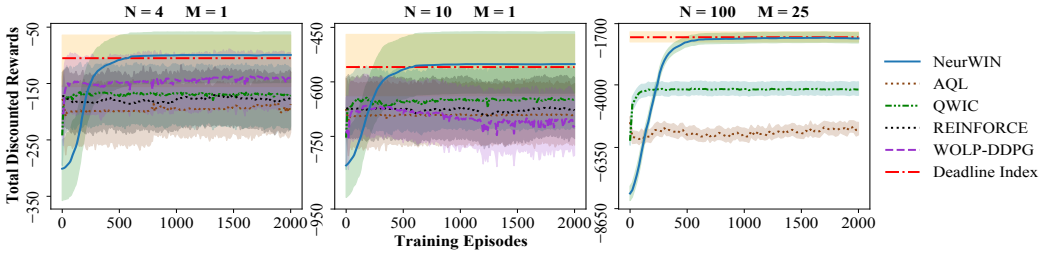

Figure 2: Average rewards and confidence bounds of different policies for deadline scheduling.

The experiment results are shown in Fig. 2. It can be observed that the performance of NeurWIN converges to that of the deadline Whittle index in 600 training episodes. In contrast, other algorithms have virtually no improvement over 2,000 training episodes and remain far worse than NeurWIN. This may be due to the explosion of state space. Even when N is only 4, the total number of possible states is $120^4 \approx 2 \times 10^8$, making it difficult for the compared deep RL algorithms to learn the optimal control in just $2,000$ episodes. QWIC performs poorly compared to NeurWIN and the deadline index. This suggests that it does not learn a good approximation to the Whittle index.

5.3 Recovering Bandits

The recovering bandits [17] aim to model the time-varying behaviors of consumers. In particular, it considers that a consumer who has just bought a certain product, say, a television, would be much less interested in advertisements of the same product in the near future. However, the consumer's interest in these advertisements may recover over time. Thus, the recovering bandit models the reward of playing an arm, i.e., displaying an advertisement, by a function $f(\min\{z, z_{max}\})$, where z is the time since the arm was last played and z_{max} is a constant specified by the arm. There is no known Whittle index or optimal control policy for this problem.

The recent study [17] considers the special case of $M = 1$. When the function $f(\cdot)$ is known, it proposes a d-step lookahead oracle. Once every d rounds, the d-step lookahead oracle builds a d-step lookahead tree. Each leaf of the d-step lookahead tree corresponds to a sequence of d actions. The d-step lookahead oracle then picks the leaf with the best reward and use the corresponding actions in the next d rounds. We extend the d-step lookahead oracle for the general case when $M \geq 1$. When $(N, M) = (4, 1)$ and $(10, 1)$, we evaluate the 8-step lookahead oracle. When $(N, M) = (100, 25)$, it is computationally infeasible to evaluate the d-step lookahead oracle due to the large action space. We modify a heuristic introduced in [17] to greedily construct a 20-step lookahead tree with 2^{20} leaves and pick the best leaf.

In our experiment, we consider different arms have different functions $f(\cdot)$. The state of each arm is its value of $\min\{z, z_{max}\}$ and we set $z_{max} = 20$ for all arms.

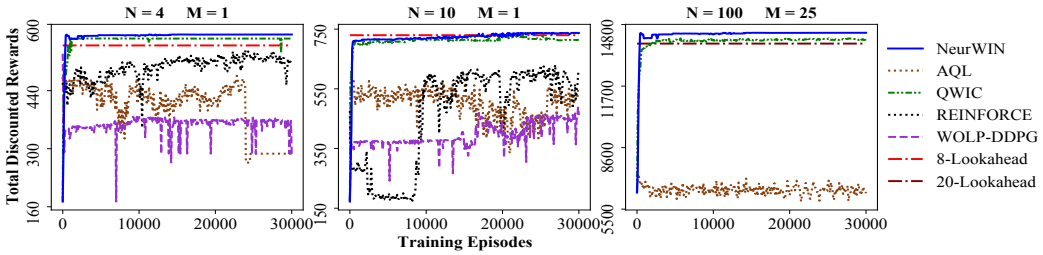

Figure 3: Experiment results for the recovering bandits.

Experiment results are shown in Fig. 3. It can be observed that NeurWIN is able to outperform 8-step and 20-step lookahead oracles in their respective settings with just a few thousands of training episodes. Other algorithms perform poorly.

5.4 Wireless Scheduling

A recent paper [1] studies the problem of wireless scheduling over fading channels. In this problem, each arm corresponds to a wireless client. Each wireless client has some data to be transmitted and it suffers from a holding cost of 1 unit per round until it has finished transmitting all its data. The channel quality of a wireless client, which determines the amount of data can be transmitted if the wireless client is scheduled, changes over time. The goal is to minimize the sum of holding costs of all wireless clients. Equivalently, we view the reward of the system as the negative of the total holding cost. Finding the Whittle index through theoretical analysis is difficult. Even for the simplified case when the channel quality is i.i.d. over time and can only be in one of two possible states, the recent paper [1] can only derive the Whittle index under some approximations. It then proposes a *size-aware index* policy using its approximated index.

In the experiment, we adopt the settings of channel qualities of the recent paper. The channel of a wireless client can be in either a good state or a bad state. Initially, the amount of load is uniform between 0 and 1Mb. The state of each arm is its channel state and the amount of remaining load. The size of state space is 2×10^6 for each arm. We consider that there are two types of arms, and different types of arms have different probabilities of being in the good state. During testing, there are $\frac{N}{2}$ arms of each type.

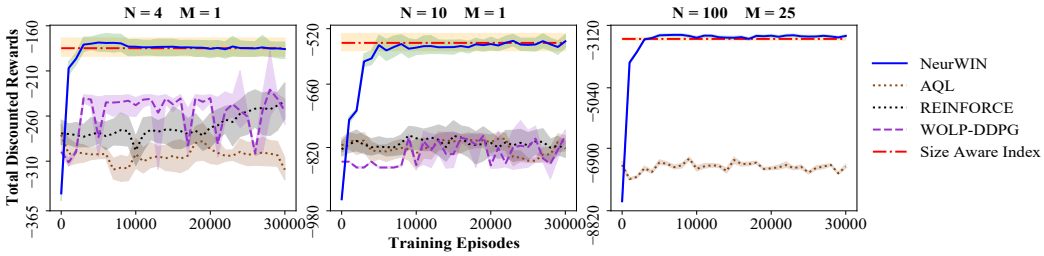

Figure 4: Average rewards and confidence bounds of different policies for wireless scheduling.

Experiment results are shown in Fig. 4. It can be observed that NeurWIN is able to perform as well as the size-aware index policy with about $5,000$ training episodes. It can also be observed that other learning algorithms perform poorly.

5.5 Evaluation of NeurWIN Limitations

A limitation of NeurWIN is that it is designed for strongly indexable bandits. Hence, it is important to evaluate whether most practical bandit problems are strongly indexable. Recall that a bandit arm is strongly indexable if $D_s(\lambda)$ is strictly decreasing in λ, for all states s. We extensively evaluate the function $D_s(\lambda)$ of different states for the three applications considered in this paper. We find that all of them are strictly decreasing in λ. Fig. 5 shows the function $D_s(\lambda)$ of five randomly selected states for each of the three applications. This result suggests that most practical bandit problems are indeed strongly indexable.

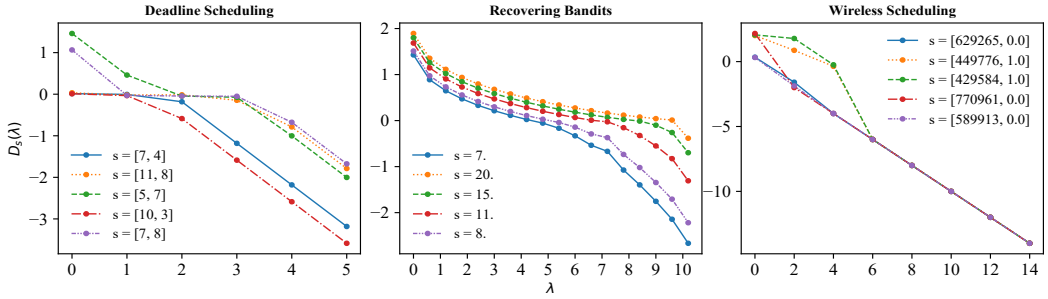

Figure 5: Difference in discounted net reward $D_s(\lambda)$ for randomly selected initial states.

Another limitation of NeurWIN is that it requires a simulator for each arm. To evaluate the robustness of NeurWIN, we test the performance of NeurWIN when the simulator is not perfectly precise. In particular, let $R_{act}(s)$ and $R_{pass}(s)$ be the rewards of an arm in state s when it is activated and not activated, respectively. Then, the simulator estimates that the rewards are $R'_{act}(s) = (1 + G_{act,s})R_{act}(s)$ and $R'_{pass}(s) = (1 + G_{pass,s})R_{pass}(s)$, respectively, where $G_{act,s}$ and $G_{pass,s}$ are independent Gaussian random variables. The variance of these Gaussian random variables correspond to the magnitude of root mean square errors in the simulator.

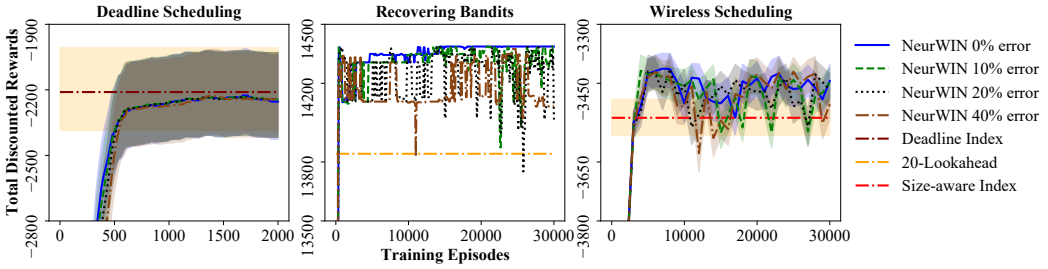

Figure 6: Experiment results for NeurWIN with noisy simulators.

We train NeurWIN using the noisy simulators with different levels of errors for the three problems. For each problem, we compare the performance of NeurWIN against the respective baseline policies. Unlike NeurWIN, the baseline policies make decisions based on the true reward functions rather than the estimated ones. The results for the case $N = 100$ and $M = 25$ are shown in Fig. 6. It can be observed that the performance of NeurWIN only degrades a little even when the root mean square error is as large as 40% of the actual rewards, and its performance remains similar or even superior to that of the baseline policies.

6 Conclusion

This paper introduced NeurWIN: a deep RL method for estimating the Whittle index for restless bandit problems. The performance of NeurWIN is evaluated by three different restless bandit problems. In each of them, NeurWIN outperforms or matches state-of-the-art control policies.

References

[1] Samuli Aalto, Pasi Lassila, and Prajwal Osti. Whittle index approach to size-aware scheduling with time-varying channels. In *Proceedings of the 2015 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Systems*, pages 57–69, 2015.

[2] K. Avrachenkov and V. S. Borkar. A learning algorithm for the whittle index policy for scheduling web crawlers. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1001–1006, 2019.

[3] V. S. Borkar and K. Chadha. A reinforcement learning algorithm for restless bandits. In *2018 Indian Control Conference (ICC)*, pages 89–94, 2018.

[4] ChangyWen. ChangyWen/wolpertinger_ddpg. `https://github.com/ChangyWen/wolpertinger_ddpg`, April 2021. original-date: 2019-06-21T02:39:45Z.

[5] Christopher R Dance and Tomi Silander. When are kalman-filter restless bandits indexable? In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 1711–1719. Curran Associates, Inc., 2015.

[6] Christoph Dann, Tor Lattimore, and Emma Brunskill. Unifying pac and regret: Uniform pac bounds for episodic reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 5717–5727, Red Hook, NY, USA, 2017. Curran Associates Inc.

[7] Gabriel Dulac-Arnold, Richard Evans, Hado van Hasselt, Peter Sunehag, Timothy Lillicrap, Jonathan Hunt, Timothy Mann, Theophane Weber, Thomas Degris, and Ben Coppin. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2016.

[8] J. Fu, Y. Nazarathy, S. Moka, and P. G. Taylor. Towards q-learning the whittle index for restless bandits. In *2019 Australian New Zealand Control Conference (ANZCC)*, pages 249–254, 2019.

[9] Christian Hubbs. Learning reinforcement learning: Reinforce with pytorch! https://towardsdatascience.com/learning-reinforcement-learning-reinforce-with-pytorch-5e8ad7fc7da0, Dec 2019.

[10] Nan Jiang, Akshay Krishnamurthy, Alekh Agarwal, John Langford, and Robert E. Schapire. Contextual decision processes with low Bellman rank are PAC-learnable. volume 70 of *Proceedings of Machine Learning Research*, pages 1704–1713, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[11] Diederick P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[12] J. Le Ny, M. Dahleh, and E. Feron. Multi-uav dynamic routing with partial observations using restless bandit allocation indices. In *2008 American Control Conference*, pages 4220–4225, 2008.

[13] Alex S. Leong, Arunselvan Ramaswamy, Daniel E. Quevedo, Holger Karl, and Ling Shi. Deep reinforcement learning for wireless sensor scheduling in cyber–physical systems. *Automatica*, 113:108759, 2020.

[14] R. Meshram, D. Manjunath, and A. Gopalan. On the whittle index for restless multiarmed hidden markov bandits. *IEEE Transactions on Automatic Control*, 63(9):3046–3053, 2018.

[15] José Niño-Mora. Dynamic priority allocation via restless bandit marginal productivity indices. *Top*, 15(2):161–198, 2007.

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

[17] Ciara Pike-Burke and Steffen Grunewalder. Recovering bandits. In *Advances in Neural Information Processing Systems*, pages 14122–14131, 2019.

[18] Carlos Riquelme, George Tucker, and Jasper Snoek. Deep bayesian bandits showdown: An empirical comparison of bayesian deep networks for thompson sampling. In *International Conference on Learning Representations (ICLR)*, 2018.

[19] Julien Seznec, Pierre Menard, Alessandro Lazaric, and Michal Valko. A single algorithm for both restless and rested rotting bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 3784–3794, 2020.

[20] Aleksandrs Slivkins and Eli Upfal. Adapting to a changing environment: the brownian restless bandits. In *COLT*, pages 343–354, 2008.

[21] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[22] V. Tripathi and E. Modiano. A whittle index approach to minimizing functions of age of information. In *2019 57th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1160–1167, 2019.

[23] Tom Van de Wiele, David Warde-Farley, Andriy Mnih, and Volodymyr Mnih. Q-learning in enormous action spaces via amortized approximate maximization. *arXiv preprint arXiv:2001.08116*, 2020.

[24] S. Wang, H. Liu, P. H. Gomes, and B. Krishnamachari. Deep reinforcement learning for dynamic multichannel access in wireless networks. *IEEE Transactions on Cognitive Communications and Networking*, 4(2):257–265, 2018.

[25] T. Wei, Yanzhi Wang, and Q. Zhu. Deep reinforcement learning for building hvac control. In *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pages 1–6, 2017.

[26] Peter Whittle. Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, pages 287–298, 1988.

[27] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Mach. Learn.*, 8(3–4):229–256, May 1992.

[28] Huizhen Yu and Dimitri P Bertsekas. Convergence results for some temporal difference methods based on least squares. *IEEE Transactions on Automatic Control*, 54(7):1515–1531, 2009.

[29] Z. Yu, Y. Xu, and L. Tong. Deadline scheduling as restless bandits. *IEEE Transactions on Automatic Control*, 63(8):2343–2358, 2018.

[30] Andrea Zanette and Emma Brunskill. Problem dependent reinforcement learning bounds which can identify bandit structure in MDPs. volume 80 of *Proceedings of Machine Learning Research*, pages 5747–5755, Stockholmsmässan, Stockholm Sweden, 10–15 Jul 2018. PMLR.

[31] Qing Zhao. Multi-armed bandits: Theory and applications to online learning in networks. *Synthesis Lectures on Communication Networks*, 12(1):1–165, 2019.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] Abstract claims an RL algorithm for learning a Whittle index policy. See section 5 for results.
   (b) Did you describe the limitations of your work? [Yes] See section 5.5.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes] See section 3 and 4.

(b) Did you include complete proofs of all theoretical results? [Yes] See appendices A and B.

3. If you ran experiments...

   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] See appendices D, E, F for experiments' details, and submitted code for implementation.

   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] See appendices D,E, F for details.

   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] Confidence intervals were plotted for 50 independent runs. See section 5 for results.

   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] See appendix C for computing resources.

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

   (a) If your work uses existing assets, did you cite the creators? [Yes]

   (b) Did you mention the license of the assets? [Yes] See code for licensing agreement.

   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See supplementary material.

   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]