# Tabular and Deep Learning of Whittle Index

Francisco Robledo, Vivek S Borkar, Urtzi Ayesta, Konstantin Avrachenkov

# Tabular and Deep Learning of Whittle Index

**Francisco Robledo** frrobledo96@gmail.com
*Basque Country University,*
*Barrio Sarriena,*
*48940, Leioa,*
*Spain*

**Vivek Borkar** borkar.vs@gmail.com
*Indian Institute of Technology*
*Main Gate Rd, IIT Area*
*400076, Mumbai*
*India*

**Urtzi Ayesta** urtzi.ayesta@irit.fr
*Institut de Recherche en Informatique de Toulouse*
*118 Route de Narbonne*
*31062, Toulouse,*
*France*

**Konstantin Avrachenkov** konstantin.avratchenkov@inria.fr
*INRIA Sophia Antipolis,*
*2004 Route des Lucioles,*
*06902, Sophia Antipolis,*
*France*

## Abstract

The Whittle index policy is a heuristic that has shown remarkable good performance (with guaranted asymptotic optimality) when applied to the class of problems known as Restless Multi-Armed Bandit Problems (RMABP). In this paper we present QWI and QWINN, two algorithms capable of learning the Whittle indices for the total discounted criterion. The key feature is the usage of two time-scales, a faster one to update the state-action $Q$-values, and a relatively slower one to update the Whittle indices. In our main theoretical result we show that QWI, which is a tabular implementation, converges to the real Whittle indices. We then present QWINN, an adaptation of QWI algorithm using neural networks to compute the $Q$-values on the faster time-scale, which is able to extrapolate information from one state to another and scales naturally to large state-space environments. Numerical computations show that QWI and QWINN converge much faster than the standard $Q$-learning algorithm, neural-network based approximate Q-learning and other state of the art algorithms.

**Keywords:** Machine learning, Reinforcement Learning, Whittle Index, Markov Decision Problem, Multi-armed Restless Bandit

## 1. Introduction

Markov decision processes (MDPs) provide a mathematical framework to model sequential decision problems. Formally, an MDP is a stochastic control process where the objective is to maximise a long-run payoff. At each time step, depending on the state and action taken by the MDP, the decision maker receives a reward and reaches a new random state. Due to their broad applicability, MDPs are found in many areas, including artificial intelligence, economics, communications and operations research.

An MDP can be solved via dynamic programming, however, this is a computationally intractable task for realistic model sizes. As a result, classes of MDPs that are analytically tractable or possess good approximations have received a lot of attention. In this paper we focus on one such class, namely, the Restless Multi-Armed Bandit Problem (RMABP), introduced in Whittle (1988). In an RMABP there are $N$ concurrent projects or bandit's arms. The decision maker

knows the states of all arms and the reward in every state, and aims at maximizing the long-term reward. At every decision epoch, the decision maker activates $M < N$ arms, and the state of active and passive arms evolve stochastically. We note that RMABP are a generalization of multi-arm bandit problems (MABP), in which only one arm can be activated at a time, and the state of passive states does not evolve. It was shown by Gittins that the optimal solution to an MABP is an index policy, nowadays known as Gittins' index policy, see Gittins et al. (2011).

RMABPs have become extremely popular over the years, and have been applied in many contexts, including inventory routing, machine maintenance, health-care systems, networking, etc. RMABPs cannot be solved exactly, except for some toy examples, and are PSPACE-hard Papadimitriou and Tsitsiklis (1999). In Whittle (1988) Whittle developed a methodology to obtain a heuristic by solving a relaxed version of the RMABP in which $M < N$ arms are activated *only* on average. The obtained heuristic, known as Whittle index policy, relies on calculating Whittle index for each of the arms, and activating in every decision epoch the $M$ arms with the highest Whittle indices. It has been reported on numerous instances that Whittle index policy provides strikingly good performance, and it has been shown to be asymptotically optimal as the number of arms grows large Weber and Weiss (1990). As expected, Whittle index reduces to Gittins index when applied to an MABP.

In recent years there has been a surge in the interest of developing algorithms to calculate Whittle index. The first paper interested in learning index policies is probably Duff (1995), which considered an MABP and developed an algorithm that learns Gittins' indices. Regarding Whittle index, one of the first papers was Fu et al. (2019), which proposed an algorithm that did not converge to the real values. The paper Avrachenkov and Borkar (2020) considers the time-average criterion, and presents a tabular algorithm that converges to Whittle index. We finally describe the NeurWIN algorithm, see Nakhleh et al. (2020), which is a gradient based reinforcement learning algorithm that aims at solving RMABP's, without any guarantee of convergence to Whittle indices.

In this paper, we present QWI and QWINN, two reinforcement learning algorithms capable of learning Whittle indices under the discounted reward criterion. A key aspect of both algorithms is the use of two time-scales, a faster one to update the state-action $Q$-values, and a relatively slower one to update the Whittle indices. In our main theoretical result, we show that QWI, which is a tabular implementation, converges to the Whittle indices of any RMABP. In our second main contribution we present QWINN, an adaptation of QWI algorithm using neural networks, which naturally scales to large state-space environments. We compare numerically the performance of QWI and QWINN with several other algorithms, including the standard $Q$-learning algorithm, a vanilla implementation of DQN and NeurWIN. Our results show that both QWI and QWINN outperform the other algorithms both in terms of rate of convergence as well as performance. QWINN stands out in its ability to obtain good Whittle indices with far fewer samples than QWI, and thanks to the extrapolation of the information performed by the neural networks, it is able to obtain good predictions of the indices of these states even with few or no samples. On the other hand, QWI, given enough samples in each state/action pair, is able to reliably converge to the correct values of the Whittle indices.

## 2. Restless Markovian Bandits

### 2.1 Problem formulation and relaxation

We consider an RMABP with $N$ arms. We denote by $S_n^i$ and $A_n^i$ the state and action space of the $i-$th arm at the $n$-th time step, respectively, and we let $r^i(S_n^i, A_n^i)$ denote the conditional expected reward obtained by the $i$-th arm at step $n$. We consider the total discounted reward criterion with a discounting factor $0 < \gamma < 1$. Arms with an active action change state with a transition probability $p(S_{n+1}^i|S_n^i, 1)$ while the passive ones can change state with a different transition probability $p(S_{n+1}^i|S_n^i, 0)$.

At each time step $n$, the control policy $\pi$ observes the state of all the arms, and activates at most $M$ arms ($A_n^i = 1$), whereas the rest remain passive with $A_n^i = 0$. The objective is to determine the optimal control policy $\pi^*$ that solves:

$$V_{\pi^*}(S_0, \ldots, S_N) = \max_{\pi} E\left[\sum_{n=1}^{\infty}\sum_{i=1}^{N}\gamma^n r^i\left(S_n^i, A_n^i\right)\right], \tag{1}$$

subject to:

$$\sum_{i=1}^{N} A_n^i \leq M, \quad n \geq 0, \tag{2}$$

where $V_{\pi^*}$ is known as the value function.

Following Whittle development Whittle (1988), we relax the constraint (2) so that it only has to be satisfied on average, i.e., $E\left[\sum_{n=0}^{\infty}\sum_{i=1}^{N}\gamma^n A_n^i\right] \leq M/(1-\gamma)$. The problem then has an equivalent *unconstrained* formulation:

$$\max_{\pi} E\left[\sum_{n=1}^{\infty}\sum_{i=1}^{N}\gamma^n\left(r\left(S_n^i, A_n^i\right) + \lambda\left(1 - A_n^i\right)\right)\right] \tag{3}$$

where $\lambda$ is a Lagrange multiplier associated with the constraint. We note that as $\lambda$ increases, we expect that the passive action to become attractive in more states, and as a consequence, the multiplier $\lambda$ can be seen as a subsidy for passivity.

The key observation by Whittle is that problem (3) can be decomposed, and its solution is obtained by combining the solution to $N$ independent problems. In other words, for each arm $i$ we need to solve the associated Bellman equation given by:

$$V_{\pi^i}^i(s) = \max_{a\in\{0,1\}}\left[Q^i(s,a)\right] \tag{4}$$

where

$$Q^i(s,a) = a\left(r^i(s,1) + \gamma\sum_j p^i(j|s,1)V(j)\right) + (1-a)\left(r^i(s,0) + \lambda + \gamma\sum_j p^i(j|s,0)V^i(j)\right) \tag{5}$$

The functions $V^i(s)$ and $Q^i(s,a)$ are known as the value function and the state-action function resp. for arm $i$.

### 2.2 Whittle index

The optimal action in (4) will be to activate a given arm $i$ if $r^i(k,1) + \gamma\sum_j p^i(j|k,1)V^i(j) > r^i(k,0) + \lambda + \gamma\sum_j p(j|k,0)V^i(j)$, while otherwise the optimal action will be to keep it passive. For a given $\lambda$, the optimal policy $\pi^*$ is then characterized by $\mathcal{S}(\lambda)$, the set of states in which the optimal action is to activate.

We assume throughout that the problems are *indexable*. Formally, a problem is indexable if and only if subset of states $\mathcal{S}(\lambda) \subset S$ where active action is optimal decreases monotonically as we increase the subsidy $\lambda$. For a given state $k$, Whittle index is then defined as the value $\lambda(k)$ such that

$$r^i(k,1) + \gamma\sum_j p^i(j|k,1)V^i(j) = r^i(k,0) + \lambda(k) + \gamma\sum_j p(j|k,0)V^i(j). \tag{6}$$

It thus follows that Whittle indices characterize the optimal solution to the relaxed problem 3, which will simply activate all arms whose Whittle index is larger than $\lambda$.

In its seminal paper, Whittle then introduced the heuristic policy for the problem (1), known nowadays as Whittle index policy, which is defined as the policy that at every time step $n$ activates the $M$ arms with the $M$ highest Whittle indices. As explained in the introduction, this heuristic has shown to have a close to optimal performance, and to be asymptotically optimal as the values of both $N$ and $M$ tend to infinity, see Weber and Weiss (1990) and **?**.

## 3. Learning the Whittle indices

In this section we present QWI and QWINN, the tabular and deep algorithms to learn Whittle indices. Throughout this section we drop the dependence on the arm from the notation.

### 3.1 Tabular QWI algorithm

From equation (6) we see that the Whittle index of a given state $x$ can be written in terms of the state-action function (5) as the solution to:

$$\lambda(x) : Q(x,1) - Q(x,0) = 0. \tag{7}$$

We note that this is a fixed point equation, as the state-action function does depend on the value of the multiplier $\lambda(x)$. We thus see that the Whittle index is defined by coupled fixed point equations (5) and (7).

To solve the joint fixed point equations, we can intuitively argue that the equilibrium $\lambda(x)$ can be found by starting in a given value of the Whittle index, then estimate the state-action functions, update the value of the Whittle index, and start all over again. In the ensuing, we demonstrate that this can be done online, and that the algorithm converges to the real Whittle index.

For this purpose, in QWI we invoke the theory of stochastic approximation with multiple time scales, see Chapter 6 in Borkar (2009), which leads to the iterations:

$$Q^x_{n+1}(s_n,a_n) = (1-\alpha(n))Q^x_{n+1}(s_n,a_n) + \alpha(n)\left((1-a_n)(r_0(s_n)+\lambda_n(x)) + a_n r_1(s_n) + \gamma \max_{v\in\{0,1\}} Q^x_n(s_{n+1},v)\right) \tag{8}$$

and

$$\lambda_{n+1}(x) = \lambda_n(x) + \beta(n)\left(Q^x_n(x,1) - Q^x_n(x,0)\right). \tag{9}$$

Here, $s_n$ denotes the state visited at time step $n$, $x$ is a reference state for which we wish to learn Whittle index, $r_0(s_n)$ and $r_1(s_n)$ are the sampled rewards for actions passive and active, respectively, and $\alpha(n)$ and $\beta(n)$ are learning parameters. As usual in the literature, the learning parameters need to satisfy $\sum_n \alpha(n) = \infty$, $\sum_n \alpha(n)^2 < \infty$, $\sum_n \beta(n) = \infty, \sum_n \beta(n)^2 < \infty$. In addition, we require $\beta(n) = o(\alpha(n))$ in order to implement the two distinct time scales, namely, the relatively faster time scale for the updates of the state-action function, and the slow one for the Whittle indices.

The pseudocode implementation of QWI can be seen in Alg. 1 in appendix B. For a given value of the discount parameter $\gamma$ and exploration parameter $\epsilon$, we initialise the $N$ arms with a random state. At every decision epoch $n$, with probability $1-\epsilon$, we choose the greedy action, i.e, we activate ($a^i_n = 1$) the $M$ arms with largest $\lambda_n(x)$, while with probability $\epsilon$ we choose $M$ arms at random. Rewards $r^i_n$ are collected, the new states $s^i_{n+1}$ are observed, and the values of the state-action function and the Whittle indices are updated by (8) and (9). We note that QWI learns simultaneously the Whittle indices of all the states of every arm on-line.

The next result is the main theoretical contribution of the paper, and shows that QWI converges to Whittle indices for any RMABP. The proof is in the appendix A.

**Theorem** (Convergence of QWI) Given learning steps $\alpha(n)$ and $\beta(n)$ such that $\sum_n \alpha(n) = \sum_n \beta(n) = \infty$, $\sum_n \alpha(n)^2 < \infty$, $\sum_n \beta(n)^2 < \infty$ and $\beta(n) = o(a(n))$, iterations (8) and (9) converge to the state-action function of Whittle index policy, denoted by $Q_W(s,a)$, and to the Whittle indices $\lambda(s)$, i.e., $\lambda_n(s) \to \lambda(s)$ and $Q_n(s,a) \to Q_W(s,a)$ a.s. $\forall s \in S, a \in A$ as $n \to \infty$.

### 3.2 QWINN algorithm

Our QWINN algorithm is based on Whittle index heuristic introduced in section 3. The main difference lies in the use of a neural network for the approximation of the $Q$-values of the fast time scale (8). A complete scheme of how our algorithm works can be found in appendix 2. With QWINN, we maintain the distinction between the visited state $s$ and the reference state $x$ of the Whittle index $\lambda(x)$, so that our state-action value function becomes $Q^x_\theta(s) = [Q^x_\theta(s,0) \quad Q^x_\theta(s,1)]$, making those two state variables the input of the neural network while the outputs are the $Q$-values for both possible actions. This neural network consists of three hidden layers of (100,200,100) neurons using a ReLU activation function. One of the main changes with respect to the tabular algorithm in section 3.1 is the use of Double Q-Learning Van Hasselt et al. (2016) Hasselt (2010), where we employ a second neural network for the calculation of the $\max_{v\in A} Q^x_{\theta'}(s_{n+1},v)$ term in equation 10. This second neural network copies the parameter values of the main network $Q_\theta$ every 50 iterations.

$$Q^x_{target}(s_n,a_n) = (1-a_n)(r_0(s_n)+\lambda_n(x)) + a_n r_1(s_n) + \gamma \max_{v\in A} Q^x_{\theta'}(s_{n+1},v). \tag{10}$$

Once we have calculated the $Q$-values of the batches through $Q_\theta(\cdot)$ and its targets $Q_{target}(\cdot)$, we compute the loss function and we update the $\theta$ parameters of the neural network through a standard Adam optimiser. For each state $x \in S$, we update Whittle index in a similar way to QWI, that is:

$$\lambda_{\theta,n+1}(x) = \lambda_{\theta,n}(x) + \beta(n)\left(Q^x_{\theta,n}(x,1) - Q^x_{\theta,n}(x,0)\right) \tag{11}$$

Once we have enough samples in memory, we create in each iteration a batch of random samples with which to train the main neural network $Q_\theta$. For each sample $(s_n, a_n, r_n, s_{n+1})$, we create different new target $Q$-Values for the neural network using each of the different states $x \in S$ of the environment, such that the target $Q$-values we want our $Q^x_\theta(s_n)$ neural network to approximate are: $Q^x_{target}(s_n, a_n) = (1-a_n)(r_0(s_n)+\lambda_n(x))+a_n r_1(s_n)+\gamma \max_{v\in A} Q^x_{\theta'}(s_{n+1}, v)$.

Once we have calculated the $Q$-values of the batches through $Q_\theta(\cdot)$ and its targets $Q_{target}(\cdot)$, we compute the loss function and we update the $\theta$ parameters of the neural network through a standard Adam optimiser.

### 3.3 Other algorithms

We describe now the other algorithms from the literature that we will consider in the numerical simulations.

#### 3.3.1 NEURWIN ALGORITHM

The NeurWIN algorithm, introduced in Nakhleh et al. (2020), is a neural network-based algorithm designed to directly compute the Whittle indices estimates $\lambda_\theta(s_n)$ of a problem using as an input to the neural network only the state whose index is to be computed.

Their training procedure consists of multiple mini-batches made of $R$ episodes. At the beginning of each mini-batch, random states $s_0$ and $s_1$ are selected, where $s_0$ defines the environment's fixed activation cost for such mini-batch as $Env^*(\lambda_\theta(s_0))$, whereas $s_1$ is the initial state of that mini-batch. In each episode $e$, a sequence of states $(s_1, s_2, \dots)$ and actions $(a_1, a_2, \dots)$ are recorded and used to compute the gradients $h_e$ with respect to $\theta$ through backward propagation. For each state $s_n$, given a fixed activation cost $\lambda = \lambda_\theta(s_0)$, the algorithm performs an active action with probability

$$\sigma_m(\lambda_\theta(s_n) - \lambda) = \frac{1}{1 + e^{-m(\lambda_\theta(s_n)-\lambda)}},$$

where $m$ is a sensitivity parameter. In addition to this, the discounted net rewards $G_e$ is observed and averaged as a bootstrapped baseline denoted $\bar{G}_b$, and then used for the weighted gradient ascent using the offset reward $G_e - \bar{G}_b$.

Unlike the other algorithms discussed in this paper, NeurWIN does not update the parameters of its estimator with each transition of the Markov chain, but at the end of several mini-batches of $R$ episodes each. In order to compare the convergence speed of all these algorithms to the desired policy, we have decided to represent each NeurWIN update as a unique transition, so that at each iteration all algorithms are updated simultaneously. It is also important to highlight that, unlike in QWI and QWINN, the training and execution phrases are separate in NeurWIN, i.e., we first need to learn the indices of each arm separately.

#### 3.3.2 $Q$-LEARNING AND DQN ALGORITHMS

We also consider the vanilla implementations of $Q$-learning and DQN Sutton and Barto (2018). $Q$-learning is known to converge to the optimal policy, Watkins and Dayan (1992), but it suffers from the "curse of dimensionality". DQN does not have performance guarantees, but in practice performs better thanks to the interpolation of the state information.

As with QWINN, DQN training is performed through batches of samples stored in memory, where the input to the neural network is the combination of arm states, while the outputs are the $Q$-values for each possible combination of arm activations. In the following results, $Q$-learning uses $\alpha = \frac{1}{\#N(s,a)}$ As the learning step size, where $\#N(s, a)$ is the number of times that said state $(s, a)$ has been visited, while DQN has been trained using a fixed learning rate $lr = 0.001$.

## 4. Numerical results

In this section we compare the performance of our algorithms QWI and QWINN with respect to $Q$-learning, DQN and NeurWIN. We consider three different RMABPs used in previous literature: the "restart problem" proposed in Avrachenkov and Borkar (2020), the "deadline scheduling problem" studied in Yu et al. (2018) and the "circular environment problem" in Fu et al. (2019). An interesting feature of these examples is that the Whittle index can be

calculated in closed-form, which allows us to assess the accuracy of the estimates of the Whittle indices obtained with the various algorithms.

We recall that Whittle index policy is a heuristic, i.e., it does not characterize the optimal solution to an RMABP, even though it has often been reported that its sub-optimality gap is very small. For this reason, we consider as a baseline the performance of the optimal policy $\pi^*$. We recall that the optimal policy $\pi^*$ solves Bellman's Optimality Equation, see for example **?**,

$$V_{\pi^*}(s) = \max_{a\in\{0,1\}^N} \left( r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_{\pi^*}(s') \right), \tag{12}$$

where $s = (s_1, \ldots, s_N)$ denotes the states of all the arms, $r(s,a) = \sum_{i=1}^N r^i(s_i, a_i)$, and $p(s'|s,a) = \prod_{i=1}^N p^i(s'_i|s_i, a_i)$. We solve equation (12) by Value Iteration.

We also use Bellman's equation in order to assess the performance of each algorithm during its training. Let $\pi$ denote the current policy of each one of the algorithms at a given time. Then, the value function $V_\pi(s)$ characterizing its performance is the solution of Bellman's equation

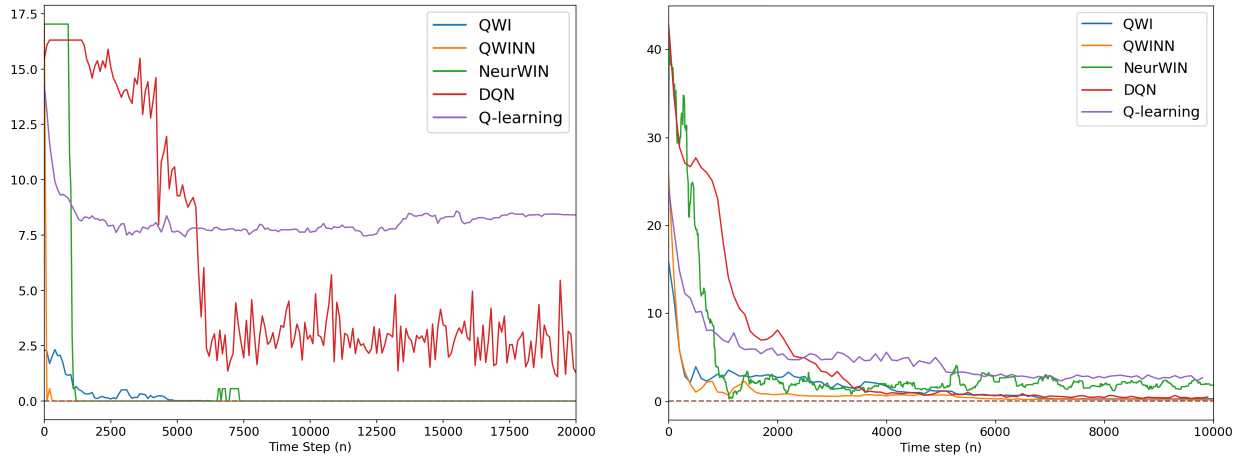
$$V_\pi(s) = \sum_a \pi(a|s) \left( r(s,a) + \gamma \sum_{s'} p(s'|s,a) V_\pi(s') \right), \tag{13}$$

where $\pi(a|s)$ is the probability of taking action $a \in \{0,1\}^N$ in state $s$. We also solve equation (13) with Value Iteration.

In order to compare the performance of a policy $\pi$ with respect to the optimal policy $\pi^*$ we use the Bellman Relative Error defined as

$$BRE(\pi, \pi^*) = \frac{1}{|S|} \sum_{s\in S} \frac{|V_\pi(s) - V_{\pi^*}(s)|}{V_{\pi^*}(s)}. \tag{14}$$

Throughout this section, we will denote by $\pi_n^P$, $P \in \{\text{QWI}, \text{QWINN}, \text{DQN}, \text{Q}\}$, the Whittle index heuristic estimated by algorithm $P$ at time $n$. In the case of QWI and QWINN we have used the following learning step sizes: $\alpha(n) = \frac{1}{\lceil \frac{n}{5000} \rceil}$, and $\beta(n) = \frac{1}{1+\lceil \frac{n \log n}{5000} \rceil} I\{n(\text{mod } 100) \equiv 0\}$, which in particular imply that $\beta(n) \neq 0$ is updated once every 100 iterations. Throughout our simulations, we consider a discount factor $\gamma = 0.9$ and an exploration parameter $\epsilon = 1$.




(a) Homogeneous restart problem

(b) Heterogeneous restart problem

Figure 1: Bellman Relative Error $BRE(\pi_n^P)$, $P \in \{\text{QWI}, \text{QWINN}, \text{NeurWIN}, \text{DQN}, \text{Q-learning}\}$ of the algorithms during training.

### 4.1 Restart problem (homogeneous arms)

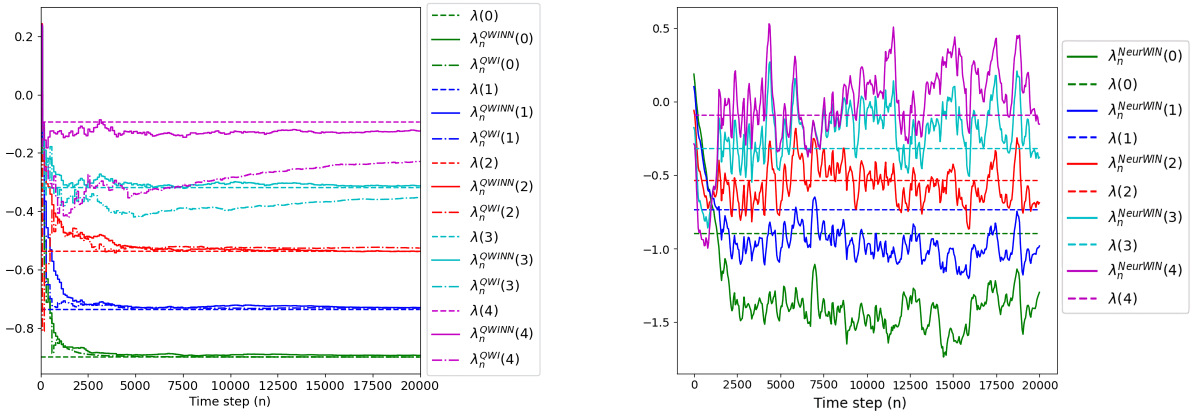
In this section we consider a "restart problem" with state space $S = \{0, 1, 2, 3, 4\}$. From any state, the active action ($a = 1$) brings the arm to the initial state with probability 1. i.e., $p(0|s, 1) = 1$, for all $s$. The transitions probability matrix in the case of passive action is:

$$P_0 = \begin{pmatrix} 1-x & x & 0 & 0 & 0 \\ 1-x & 0 & x & 0 & 0 \\ 1-x & 0 & 0 & x & 0 \\ 1-x & 0 & 0 & 0 & x \\ 1-x & 0 & 0 & 0 & x \end{pmatrix}.$$

The expected conditional reward is given by $r(s, 0) = y^{s+1}$ for passive actions, while $r(s, 1) = 0$ for active actions ($a = 1$). We consider the following setting: $N = 5$ arms; one arm $M = 1$ is activated every time epoch; and $x = y = 0.9$. The Whittle index can be analytically calculated in this problem, yielding $\lambda(0) = -0.9$, $\lambda(1) = -0.7371$, $\lambda(2) = -0.5373$, $\lambda(3) = -0.3188$ and $\lambda(4) = -0.0939$ while using a discount factor $\gamma = 0.9$. We note that the Whittle index is increasing as the state increases, which is consistent with the reward structure. We note that from the description of the problem, a good policy will tend to keep all arms around state 0, as the rewards are larger there.

In Figure 1a we depict the Bellman Relative Error for all the algorithms. Namely, we depict $BRE(\pi_n^P, \pi^*)$, $P \in \{\text{QWI}, \text{QWINN}, \text{NeurWIN}, \text{Q-learning}, \text{DQN}\}$, for all the time steps $n$. We observe that QWI, NeurWIN and QWINN learn the optimal policy, QWINN being the fastest. We note that even though Whittle index policy need not be optimal, it might happen in such a simple problem. On the other hand, $Q$-learning does not learn the optimal policy, even though we know from theory that as $n \to \infty$, it is the only algorithm for which convergence to the optimal policy is guaranteed Watkins and Dayan (1992). The vanilla DQN implementation does better, but it requires longer to converge to the optimal policy.

An important observation is that the performance of an index policy, in particular the values plotted in Figure 1a, only depend on the relative ordering between the states, and not on the precise value of the Whittle index. We thus explore the accuracy of the estimates of QWI/QWINN (see Figure 2a) and NeurWIN (see Figure 2b) during their training. More specifically, in Figure 2a we plot the averaged (across the 5 arms) estimates of Whittle indices. In the case of NeurWIN, there is a single agent cloned for the 5 arms, and therefore, in Figure 2b we depict the estimate of this single agent.




(a) QWI (solid lines) and QWINN (dashdot lines)

(b) NeurWIN

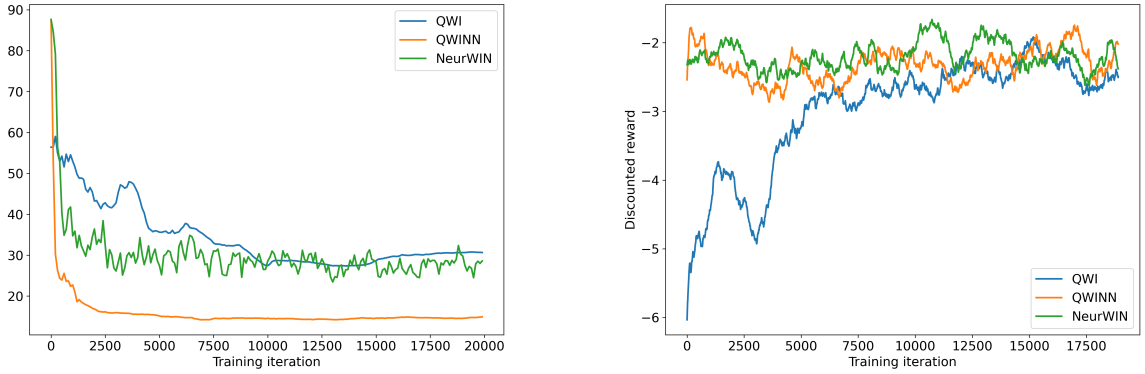
Figure 2: Evolution of the Whittle index estimates for the restart problem

In Figure 2a we note that the QWI's estimates for the indices of states 0,1 and 2 converge, whereas the estimates of states 3 and 4 need longer simulations due to the fact that these states are visited less frequently, while QWINN converges rapidly to the theoretical values of all the indeces. It is not surprising that the estimates for states 3 and 4 require

longer as they are visited less frequently. We also observe that the estimates are stable, i.e, there are no fluctuations. In Figure 2b we note that the ordering with NeurWIN is correct, but that the fluctuations do not vanish. The latter might suggest that in the case of heterogeneous arms, NeurWIN might swap the ordering of the states, which might yield a sub optimallity gap larger than with QWI and QWINN. In Appendix C we show that this is indeed the case.

### 4.2 Deadline scheduling problem

In this section we consider the deadline scheduling problem studied in Nakhleh et al. (2020), and for which the source code is publicly available. The states of this problem are defined by two different variables, the service time duration $B \in [0, 9]$ and the deadline $T \in [0, 12]$, leading to a total of 130 states (although several of these states are not accessible by the Markov chain as detailed in Yu et al. (2018)). The full description of this problem, including the state transition rules, the reward functions and the theoretical values of the indeces, is detailed in the appendix D. For this problem we will consider the homogeneous case using $N = 3$ and $M = 1$, with a processing cost $c = 0.5$ for all arms and a discount factor $\gamma = 0.9$.




(a) Percentage of states in which an optimal action is not performed in the "deadline scheduling" problem with homogeneous arms

(b) Discounted reward using the Whittle index estimates of QWI, QWINN and NeurWIN in the homogeneous "deadline scheduling" problem at each training time as a policy

Figure 3: Performance graphs for "deadline scheduling problem".

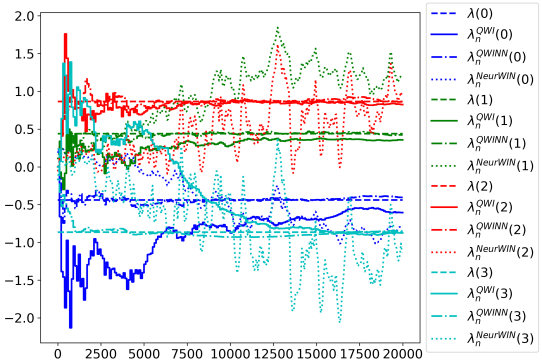
In Figure 3a we plot the percentage of state combinations in which the action of an algorithm would diverge from what prescribed by Whittle index policy. In other words, this happens when the algorithm fails to identify the best arm. We observe that the QWINN is the algorithm with the smallest number of misorderings, and that both QWI and NeurWIN fail to identify the best arm in around 30% of the states. By inspection, we have observed that QWI's estimates are poor for states with deadline $T > 9$, which are rarely visited. This is a clear example of the limitations of a tabular algorithm when solving a problem with a large state space: by not having enough samples of these states, the tabular QWI algorithm is not able to obtain good predictions of the Whittle indices. On the other hand, we have not identified a particular set of states in which NeurWIN and QWINN fails.

However, it is important to highlight that these misorderings do not necessarily imply a significant loss of performance. In Figure 3b we plot the performance of the learnt algorithms as time goes on. We note that given the size of the state space, solving Bellman's equation (13) at every time step is computationally very demanding. Thus, for any given time step and every algorithm, we evaluate the performance of the learnt algorithms using Monte Carlo, choosing one initial state at random. We note that the performance obtained with QWI, QWINN and NeurWIN do not significantly differ. This indicates that the states in which QWI and NeurWIN miss to identify the best arm do not have a significant impact on the performance.

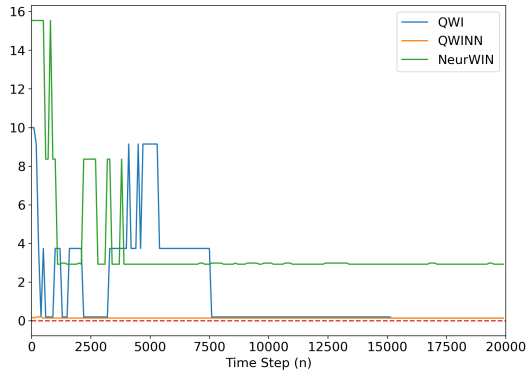
### 4.3 Circular problem

Finally, we consider here the "circular problem" problem, with a state space $S = \{0, 1, 2, 3\}$ Fu et al. (2019). In this problem, with an active action the process remains in its current state with probability $0.6$, or increments positively with probability $0.4$. Similarly, with a passive action the process remains in its current state with probability $0.6$, or decrements negatively with probability $0.4$. It is called circular, as an increment (decrement) from state 3 (0) brings the process to state 0 (3). The reward function does not depend on the action performed, but only on the state, being $R(0) = -1, R(1) = R(2) = 0, R(3) = 1$. Using a discount parameter of $\gamma = 0.9$, the theoretical values of Whittle indices for each state are $\lambda(0) = -0.4390, \lambda(1) = 0.4390, \lambda(2) = 0.8652, \lambda(3) = -0.8652$, and we note that the best state is 1, and the second best 2.

In Figure 4a, we plot the Whittle index estimates for the QWI, QWINN and NeurWIN algorithms for this problem. We note that QWI and QWINN order correctly all states, but that with NeurWIN the estimate of the index of state 2 is consistently larger than that of state 1. We also note that QWINN's estimates converge the fastest. In Figure 4b, we plot the Bellman Relative Error $BRE(P)$ for the QWI, QWINN and NeurWIN algorithms, using a discount factor $\gamma = 0.9$ and $N = 3$ and $M = 1$. As can be seen, none of the algorithms achieves an optimal policy, since for this problem with this number of arms the optimal policy is not Whittle indices. Nevertheless, QWI and QWINN are able to achieve very good performance with respect to the optimal policy, especially QWINN which achieves this policy in the first few hundred iterations. On the other hand, due to NeurWIN's misordering of the indices, it shows a larger suboptimality gap throughout the training.




(a) Whittle index estimates for QWI (solid lines), QWINN (dash-dots) and NeurWIN (dotted line)




(b) Bellman Relative Error $BRE(\pi_n^P)$, $P \in \{\text{QWI, QWINN, NeurWIN}\}$ of the algorithms during training

Figure 4: Performance graphs for "circular problem".

## 5. Conclusions

In this paper we have developed two algorithms, QWI and QWINN, capable of learning Whittle indices for the total discounted criterion. Through numerical simulations, we have compared the performance of QWI, QWINN with respect to other relevant algorithms, namely Q-larning, DQN and NeurWIN. Our results show that QWI, QWINN, NeurWIN are all more efficient than $Q$-learning in obtaining a policy with near-optimal performance. QWI estimates very accurately Whittle indices in problems of small to moderate size. QWINN is able to accurately learn Whittle indices for problems with larger state spaces, and it is typically the fastest algorithm to converge. NeurWIN's estimates are not always accurate, but the latter does not necessarily imply a loss in performance. On the other hand, a simple problem like the circular shows that a wrong ordering can lead to a substantial performance degradation. In future work we plan to carry out a more comprenhenvise study to investigate the advantages and disadvantages of each algorithm.

## References


Jinane Abounadi, Dimitrib Bertsekas, and Vivek S Borkar. Learning algorithms for markov decision processes with average cost. *SIAM Journal on Control and Optimization*, 40(3):681–698, 2001. URL https://epubs.siam.org/doi/abs/10.1137/S0363012999361974?casa_token=1DzfRZBygwIAAAAA:1R6PB1DKopfx50oH5D9xAql38TAL8XbUL1TY3k8FJkc5NrkemLARZoiEd1paX3bA6zMAiHiklZWi-w.

Konstantin Avrachenkov and Vivek S Borkar. Whittle index based q-learning for restless bandits with average reward. *arXiv preprint arXiv:2004.14427*, 2020. URL https://arxiv.org/abs/2004.14427.

V. S. Borkar. Stochastic approximation: a dynamical systems viewpoint. *Springer*, 48, 2009. doi: 10.1007/978-93-86279-38-5.

Michael O Duff. Q-learning for bandit problems. In *Machine Learning Proceedings 1995*, pages 209–217. Elsevier, 1995. URL https://www.sciencedirect.com/science/article/pii/B9781558603776500347.

Jing Fu, Yoni Nazarathy, Sarat Moka, and Peter G Taylor. Towards q-learning the whittle index for restless bandits. In *2019 Australian & New Zealand Control Conference (ANZCC)*, pages 249–254. IEEE, 2019. doi: 10.1109/anzcc47194.2019.8945748.

J.C. Gittins, K. Glazebrook, and R. Weber. *Multi-armed Bandit Allocation Indices*. Wiley, 2011.

Hado Hasselt. Double Q-learning. In *Advances in Neural Information Processing Systems*, volume 23. Curran Associates, Inc., 2010. URL https://proceedings.neurips.cc/paper/2010/hash/091d584fced301b442654dd8c23b3fc9-Abstract.html.

Chandrashekar Lakshminarayanan and Shalabh Bhatnagar. A stability criterion for two timescale stochastic approximation schemes. *Automatica*, 79:108–114, 2017. doi: 10.1016/j.automatica.2016.12.014.

Khaled Nakhleh, Santosh Ganji, Ping-Chun Hsieh, I-Hong Hou, and Srinivas Shakkottai. Neurwin: Neural whittle index network for restless bandits via deep rl. 2020. URL https://openreview.net/forum?id=QpT9Q_NNfQL.

C.H. Papadimitriou and J.N. Tsitsiklis. The complexity of optimal queueing network. *Mathematics of Operations Research*, 24(2):293–305, 1999.

Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016. URL https://ojs.aaai.org/index.php/AAAI/article/view/10295.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992. URL https://link.springer.com/content/pdf/10.1007/BF00992698.pdf.

Richard R Weber and Gideon Weiss. On an index policy for restless bandits. *Journal of applied probability*, pages 637–648, 1990. doi: 10.2307/3214547.

Peter Whittle. Restless bandits: Activity allocation in a changing world. *Journal of applied probability*, pages 287–298, 1988. doi: 10.2307/3214163.

Zhe Yu, Yunjian Xu, and Lang Tong. Deadline scheduling as restless bandits. *IEEE Transactions on Automatic Control*, 63(8):2343–2358, 2018. URL https://ieeexplore.ieee.org/abstract/document/8295041/.

## Appendix A. Proof of convergence of QWI index

For the demonstration of the convergence of the estimates of the tabular QWI algorithm to Whittle indices, both the RMABP and the agent itself must meet a number of prerequisites to ensure convergence to the theoretical values of the Whittle indices.

(C0) Unichain property. There must exist a state $s \in S$ reachable with strictly positive probability from any other state under any stationary policy.

(C1) Step sizes $\{\alpha(n)\}$ satisfy, for $x \in (0,1)$,

$$\sup_n \frac{\alpha(\lfloor xn \rfloor)}{\alpha(n)} < \infty,$$
$$\sup_{y \in [x,1]} \left| \frac{\sum_{m=0}^{\lfloor yn \rfloor} \alpha(m)}{\sum_{m=0}^{n} \alpha(m)} - 1 \right| \to 0$$

(C2) The problem is Whittle indexable, that is, given the subset of states $\Pi(\lambda)$ where the optimal action is the passive action due to the extra reward $\lambda$ of this action, the number of states within this subset $\Pi(\lambda)$ must grow monotonically as $\lambda$ increases.

We will follow the general scheme proposed in Lakshminarayanan and Bhatnagar (2017) and Borkar (2009). Consider a generalised version of equations (8) and (9) such that:

$$x_{n+1} = x_n + a(n)\left[h(x_n, y_n) + M_{n+1}^{(1)}\right] \tag{15}$$
$$y_{n+1} = y_n + b(n)\left[g(x_n, y_n) + M_{n+1}^{(2)}\right] \tag{16}$$

where (15) represents equation (8), the fast time-scale where we compute the $Q$-values, and (16) represents equation (9), the slower time-scale in which the Whittle indices are updated. In these equations, the functions $h$ and $g$ are continuous Lipschitz functions, the $M_n$ are martingale difference sequences representing noise terms, and $a(n)$ and $b(n)$ are step-size terms satisfying $\frac{b(n)}{a(n)} \to 0$ as $n \to \infty$, in addition to the usual conditions $\sum_n a(n) = \sum_n b(n) = \infty$ and $\sum_n a(n)^2, \sum_n b(n)^2 < \infty$.

First, let us define $F_{su}^{\lambda}(\Psi(j,b))$ and $M_{n+1}(s,u)$ such that:

$$F_{su}^{\lambda}(\Psi(j,b)) = (1-u)(R_0(s) + \lambda) + uR_1(s) + \gamma \sum_j p(j|i,u) \max_{v \in \{0,1\}} \Psi(j,v) \tag{17}$$

$$M_{n+1}(s,u) = (1-u)(R_0(s) + \lambda_n(x)) + uR_1(s) + \max_{v \in \{0,1\}} Q_n(x_{n+1}, v) - F_{su}^{\lambda_n(x)}(Q_n) \tag{18}$$

From these terms, we can rewrite the equation (8) as:

$$Q_{n+1}^x(s,u) = Q_n^x(s,u) + \alpha(n)\left[F_{su}^{\lambda_n(x)}(Q_n) - Q_n + M_{n+1}(s,u)\right] \tag{19}$$

Comparing equations (8) and (19) we make the correspondence $a(n) = \alpha(n)$, $h(x_n, y_n) = F_{su}^{\lambda_n(x)}(Q_n) - Q_n$, where $x_n = Q_n$ and $y_n = \lambda_n$ are the $Q$-value and Whittle index estimate respectively and $M_{n+1}(s,u)$ is the martingale difference sequence $M_{n+1}^{(1)}$. On the other hand, equations (9) and (16) correspond to $b(n) = \beta(n)$, $g(x_n, y_n) = Q_n^x(x,1) - Q_n^x(x,0)$ and a martingale difference sequence $M_{n+1}^{(2)} = 0$.

As in Borkar (2009), we will consider three necessary conditions for equations (15) and (16) to be stable and converge to their optimal values $x_n \to x^*$ and $y_n \to y^*$.

A1 $h$ and $g$ must be Lipschitz continuous.

A2 $\{M_n^{(1)}\}$ and $\{M_n^{(2)}\}$ are martingale difference sequences.

A3 $\{a(n)\}$ and $\{b(n)\}$ satisfy:

- $a(n) > 0, b(n) > 0$
- $\sum_n a(n) = \sum_n b(n) = \infty$, $\sum_n (a(n)^2 + b(n)^2 < \infty$
- $\frac{b(n)}{a(n)} \to \infty$

The full demonstration of **A1** can be found on page 687 of Abounadi et al. (2001). In our notation, $M_{n+1}(s,u)$ and 0 are respectively the martingale sequences $M_{n+1}^{(1)}$ and $M_{n+1}^{(2)}$ thus satisfying condition **A2**. Finally, **A3** is also verified given that $\beta(n) = o(\alpha(n))$.

Let us assume, for the demonstration of the convergence of the Whittle indices to their theoretical values, that the equations (8) and (9) are bounded. We will prove this condition later. First we will rewrite the equation for the calculation of the indices (9) as:

$$\lambda_{n+1}(x) = \lambda_n(x) + \alpha(n)\left(\frac{\beta(n)}{\alpha(n)}\right)\left(Q_n^x(x,1) - Q_n^x(x,0)\right) \tag{20}$$

Let $\tau(n) = \sum_{m=0}^{n} \alpha(m), m \geq 0$. We define $\bar{Q}(t), \bar{\lambda}(t)$ as the interpolation of the trajectories of $Q_n^x$ and $\lambda_n(x)$ on each interval $[\tau(n), \tau(n+1)], n \geq 0$ as:

$$\bar{Q}(t) = Q(n) + \left(\frac{t-\tau(n)}{\tau(n+1)-\tau(n)}\right)\left(Q(n+1) - Q(n)\right) \tag{21}$$

$$\bar{\lambda}(t) = \lambda(n) + \left(\frac{t-\tau(n)}{\tau(n+1)-\tau(n)}\right)\left(\lambda(n+1) + \lambda(n)\right) \tag{22}$$

$$t \in [\tau(n), \tau(n+1)]$$

which track the asymptotic behavior of the coupled o.d.e.s

$$\dot{Q}(t) = h(Q(t), \lambda(t)), \dot{\lambda} = 0$$

where the latter is a consequence of $\frac{\beta(n)}{\alpha(n)} \to 0$ in (20). From the reference frame of $Q(t)$, $\lambda(\cdot)$ is a constant of value $\lambda'$. Because of this, the first o.d.e. becomes $\dot{Q} = h(Q(t), \lambda')$, which is well posed and bounded, and has an asymptotically stable equilibrium at $Q_\lambda^*$ (Theorem 3.4, p. 689 in Abounadi et al. (2001)). This implies that $Q_n^x - Q_{\lambda_n}^* \to 0$ as $n \to \infty$.

On the other hand, for $\lambda(t)$, let us consider a second trajectory on another time scale, such that:

$$\begin{aligned} \tilde{\lambda}(t) &= \lambda(n) + \left(\frac{t-\tau'(n)}{\tau'(n+1)-\tau'(n)}\right)\left(g(n+1) - g(n)\right) \\ t &\in [\tau'(n), \tau'(n+1)], \tau'(n) = \sum_{m=0}^{n} \beta(m), n \geq 0 \end{aligned} \tag{23}$$

which tracks the o.d.e.:

$$\dot{\Lambda}(t) = Q_{\Lambda(t)}^*(x,1) - Q_{\Lambda(t)}^*(x,0)$$

If $\Lambda(t) > \lambda(x)$ (excess subsidy), the passive mode is preferred, i.e., $Q_{\Lambda(t)}^*(x,0) > Q_{\Lambda(t)}^*(x,1)$, making the r.h.s of the previous equation $< 0$ and $\Lambda(t)$ decreases. Likewise, if the opposite strict inequality holds, the r.h.s. is $> 0$ and $\lambda(t)$ increases. Thus, the trajectory of $\Lambda(\cdot)$ remains bounded. As in the previous case, being a well-defined bounded scalar o.d.e., it converges to an asymptotically stable equilibrium where $\Lambda$ satisfies $Q_\Lambda^*(x,1) = Q_\Lambda^*(x,0)$. This point is where both policies are equally desirable, i.e. $\Lambda$ is the Whittle index.

## Appendix B. QWI/QWINN algorithm scheme

**Algorithm 1** Tabular QWI Algorithm

**Input:** Discount parameter $\gamma \in (0,1)$, exploration parameter $\epsilon \in [0,1]$,
**Output:** Whittle index matrix for all states in each arm
Initialize $s_0$ for all arms
**for** $n = 1 : n_{end}$ **do**
  Define action $a_n^i$ through $\epsilon$-greedy policy for each arm
  Get new states $s_{n+1}^i$ and rewards $r_n^i$ from states $s_n^i$ and actions $a_n^i$
  Update learning rate $\alpha(n)$, $\beta(n)$
  Update $(s_n^i, a_n^i)$ $Q$-values as (8)
  Update Whittle estimates as (9)
**end for**

**Algorithm 2** QWINN Algorithm

**Input:** Discount parameter $\gamma \in (0,1)$, exploration parameter $\epsilon \in [0,1]$,
**Output:** Whittle index vector for all states
Initialize $s_0$ for all arms
**for** $n = 1 : n_{end}$ **do**
  Define action $a_n^i$ through $\epsilon$-greedy policy for each arm
  Get new states $s_{n+1}^i$ and rewards $r_n^i$ from states $s_n^i$ and actions $a_n^i$
  Save each arm's data into separated memories
  **if** memory $>$ threshold **then**
    Update Whittle index learning rate $\beta(n)$
    **for** arm $i \in N$ **do**
      **for** $x \in S$ **do**
        Predict $Q$-values of sample batch $(s_n^i, a_n^i)$ using reference state $x$ for Whittle index with $Q_\theta$
        Compute target $Q$-value for sample $(s_n^i, a_n^i, r_n^i, s_{n+1}^i, x)$ as (10)
      **end for**
      Compute loss function between $Q_\theta$ and $Q_{target}$
      Update the $\theta$ parameters of the $Q_\theta$ regressor through backpropagation
      Update every Whittle index as (11)
    **end for**
  **end if**
**end for**

## Appendix C. Restart problem (heterogeneous arms)

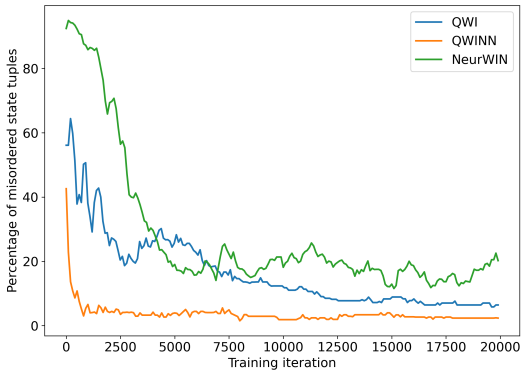
In this subsection we want to investigate the impact that errors in the estimation of Whittle indices has on the performance. In order to do so, we consider a restart problem with $N = 3$ arms, each one having a different set of parameters, and as before we consider $M = 1$. As in the previous section, we assume that for every arm $x = y$, and that this value is taken uniformly at random from the set $\{0.4, 0.5, 0.6, 0.7, 0.8, 0.9\}$. We simulate 20 such instances, and to assess the sub-optimality gap we consider the Bellman Averaged Relative Error of algorithm $P$ defined as:

$$BRE(P) = \frac{1}{20} \sum_{k=1}^{20} BRE_k(\pi_k^P, \pi_k^*), \tag{24}$$

where $BRE_k(\pi_k^P, \pi_k^*)$ denotes the Bellman Relative Error for instance $k$, $\pi_k^*$ is the optimal policy for instance $k$, and $\pi_k^P$ is the Whittle Index policy for instance $k$ obtained with algorithm $P$.

In Figure 1b we plot the evolution of $BRE(P)$ for each of the algorithms during the training. We note that the state space is smaller than in the previous section, and as a consequence $Q$-learning converges faster, and that DQN

essentially learns the optimal policy. Regarding QWI and QWINN, we observe that whereas the relative errors of QWI and QWINN are negligible and indistinguishable from DQN, NeurWIN incurs in a tangible error of around $3\%$. The latter is due to lack of accuracy in the estimation of Whittle index, which can induce a wrong ordering among the states. One way to show convergence to the correct index policy is to plot, for all possible combinations of states, what percentage of these do not perform the same action that Whittle index policy would render. In other words, during each instant of training, and for each possible combination of states among the different arms of the problem, we evaluate what action each algorithm's policy would perform against the Whittle index policy corresponding to that problem. In figure 5a we plot the percentage of state combinations in which an optimal action is not taken. For the restart problem with heterogeneous arms, using $N = 3$, the total number of state combinations is $5^3$. As we can see, while QWINN obtains an optimal policy for most of the states in the first iterations, both QWI and NeurWIN keep an error of $30\%$, the former due to the difficulty of visiting the higher states of the problem, while the latter due to inaccurate index predictions, leading to suboptimal policies especially in the heterogeneous case.




(a) Percentage of states in which an optimal action is not performed in the "restart" problem for heterogeneous arms.

Figure 5: Performance graphs for the QWI, QWINN and NeurWIN algorithms: assignment of optimal policies in the heterogeneous "restart" problem (a) and discounted rewards for the homogeneous "deadline scheduling" problem (b).

## Appendix D. "Deadline scheduling" problem properties

The deadline scheduling problem, proposed in ref, has states formed by two different variables: The service time $B \in [0, 9]$ and the deadline $T \in [0, 12]$. Therefore, the variable $B$ represents the amount of workload pending to complete a certain job while the variable $T$ represents the remaining time available to perform it. When there is no job at the $i$th position, the state is $(0, 0)$, while otherwise it is $(T_n^i, B_n^i)$. In each iteration, the state transition $s_n^i = (T_n^i, B_n^i)$ depends on the action $a_n^i$ performed:

$$s_{n+1}^i = \begin{cases} (T_n^i - 1, (B_n^i - a_n^i)^+) & \text{if } T_n^i > 1, \\ (T, B) \text{ with prob. } Q(T, B) & \text{if } T_n^i \leq 1, \end{cases}$$

where $b^+ = \max(b, 0)$. When $T = 1$, the deadline for completing the job ends, and the new state (including the empty state $(0, 0)$) is chosen uniformly at random. The value of $B$, the workload to be completed, is only reduced if the action is active. If the scheduler reaches the state $(T = 1, B > 0)$, the job could not be finished on time, and an extra penalty $F(B_n^i - a_n^i) = 0.2(B_n^i - a_n^i)^2$ is incurred. In addition, activating the arms involves a fixed cost $c = 0.5$, resulting in the following rewards:

$$r_n^i(s_n^i, a_n^i, c) = \begin{cases} (1 - c)a_n^i & \text{if } B_n^i > 0, T_n^i > 1, \\ (1 - c)a_n^i - F(B_n^i - a_n^i) & \text{if } B_n^i > 0, T_n^i = 1, \\ 0 & \text{otherwise} \end{cases}$$

In Yu et al. (2018) the authors provide an expression for Whittle index, given by:

$$\lambda(T,B,c)=\begin{cases}0 & \text{if } B=0,\\ 1-c & \text{if } 1\leq B\leq T-1,\\ \gamma^{T-1}F(B-T+1)-\gamma^{T-1}F(B-T)+1-c & \text{if } T\leq B\end{cases}$$