A Checklist

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 appendix 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 E, F, G 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 E,F, G 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 D 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]

B Proof of Deadline Scheduling's Strong Indexability

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

Proof. Fix a state $s = (D, B)$, the function $D_s(\lambda) := (Q_{\lambda,act}(s) - Q_{\lambda,pass}(s))$ is a continuous and piece-wise linear function since the number of states is finite. Thus, it is sufficient to prove that $D_s(\lambda)$ is strictly decreasing at all points of λ where $D_s(\lambda)$ is differentiable. Let $L_{\lambda,act}(s)$ be the sequence of actions taken by a policy that activates the arm at round 1, and then uses the optimal policy starting from round 2. Let $L_{\lambda,pass}(s)$ be the sequence of actions taken by a policy that does not activate the arm at round 1, and then uses the optimal policy starting from round 2. We prove this theorem by comparing $L_{\lambda,act}(s)$ and $L_{\lambda,pass}(s)$ on every sample path. We consider the following two scenarios:

In the first scenario, $L_{\lambda,act}(s)$ and $L_{\lambda,pass}(s)$ are the same starting from round 2. Let b be the remaining job size when the current deadline expires under $L_{\lambda,act}(s)$. Since $L_{\lambda,pass}(s)$ is the same as $L_{\lambda,act}(s)$ starting from round 2, its remaining job size when the current deadline expires is $b+1$. Thus, $D_s(\lambda) = 1 - c - \lambda + \beta^{D-1}(F(b+1) - F(b))$, which is strictly decreasing in λ whenever $D_s(\lambda)$ is differentiable.

In the second scenario, $L_{\lambda,act}(s)$ and $L_{\lambda,pass}(s)$ are not the same after round 2. Let τ be the first time after round 2 that they are different. Since they are the same between round 2 and round τ, the remaining job size under $L_{\lambda,act}(s)$ is no larger than that under $L_{\lambda,pass}(s)$. Moreover, by [34], the Whittle index is increasing in job size. Hence, we can conclude that, on round τ, $L_{\lambda,pass}(s)$ activates the arm and $L_{\lambda,act}(s)$ does not activate the arm. After round τ, $L_{\lambda,act}(s)$ and $L_{\lambda,pass}(s)$ are in the same state and will choose the same actions for all following rounds. Thus, the two sequences only see different rewards on round 1 and round τ, and we have $D_s(\lambda) = (1 - c - \lambda)(1 - \beta^{\tau-1})$, which is strictly decreasing in λ whenever $D_s(\lambda)$ is differentiable.

Combining the two scenarios, the proof is complete. $\square$

C Additional NeurWIN Results For Neural Networks With Different Number of Parameters

In this section, we show the total discounted rewards' performance for NeurWIN when trained on different-sized neural networks. We compare the performance against each case's proposed baseline in order to observe the convergence rate of NeurWIN. We use the same training parameters as in section 5, and plot the results for $(4, 1), (10, 1), (100, 25)$. Fig. 7 shows the performance for a larger neural network per arm with $\{48, 64\}$ neurons in 2 hidden layers. Fig. 8 provides results for a smaller neural network per arm with $\{8, 14\}$ neurons in 2 hidden layers. For the deadline and wireless scheduling cases, the larger neural network has 3345 parameters per arm compared with 625 parameters for each network from section 5, and 165 parameters for the smaller neural network. A recovering bandits' network has a larger network parameters' count of 3297 parameters compared with 609 parameters for results in section 5, and 157 parameters for the smaller neural network.

It can be observed that a neural network with more parameters is able to converge to the baseline with fewer episodes compared with the smaller and original neural networks. In the case of deadline scheduling, the smaller network requires 1380 episodes in $(4, 1)$ to reach the Whittle index performance. The larger network, in contrast, reaches the Whittle index performance in terms of total discounted rewards with considerably fewer episodes at 180 episodes. The original network used in section 5 converges in approximately 600 episodes. The same observation is true for the wireless scheduling case, with the smaller network requiring 10,000 episodes for $(4, 1)$, and fails to outperform the baseline for $(10, 1)$.The larger network converges in fewer episodes compared to the smaller and original networks.

More interestingly, the smaller neural networks fail to outperform the baselines in the recovering bandits case for $(4, 1)$ and $(10, 1)$, which suggests that the selected neural network architecture is not rich-enough for learning the Whittle index.

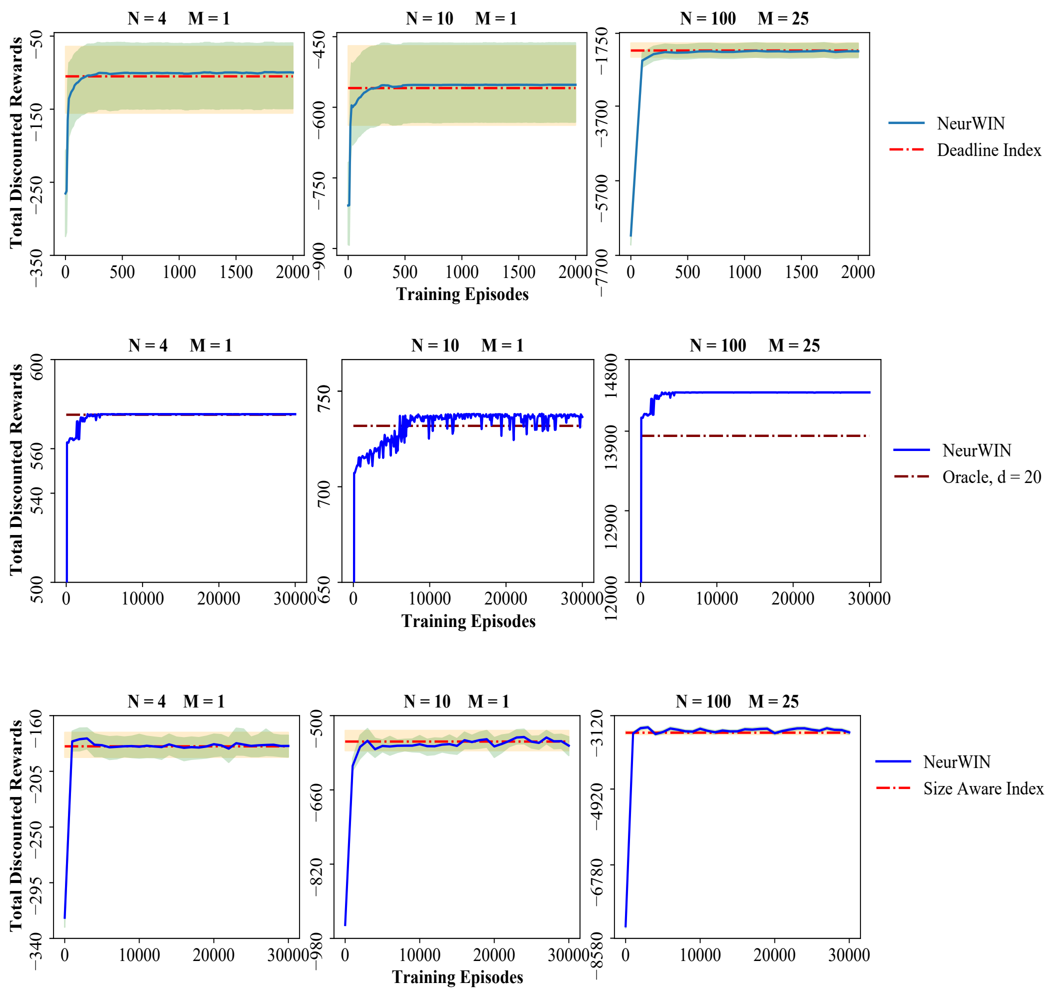

Figure 7: NeurWIN performance results when training a larger network with 3345 parameters per arm for the deadline and wireless scheduling cases, and 3297 parameters for the recovering bandits' case.

D Training and Testing Environments

For all cases, we implement NeurWIN algorithm using PyTorch, and train the agent on a single arm modelled after OpenAI's Gym API.

All algorithms were trained and tested on a Windows 10 build 19043.985 machine with an AMD Ryzen 3950X CPU, and 64 GB 3600 MHz RAM.

E Deadline Scheduling Training and Inference Details

E.1 Formulated Restless Bandit for the Deadline Scheduling Case

The state $s[t]$, action $a[t]$, reward $r[t]$, and next state $s[t+1]$ of one arm are listed below:

State $s[t]$: The state is a vector (D, B). B denotes the job size (i.e. amount of electricity needed for an electric vehicle), and D is the job's time until the hard drop deadline d is reached (i.e. time until an electric vehicle leaves).

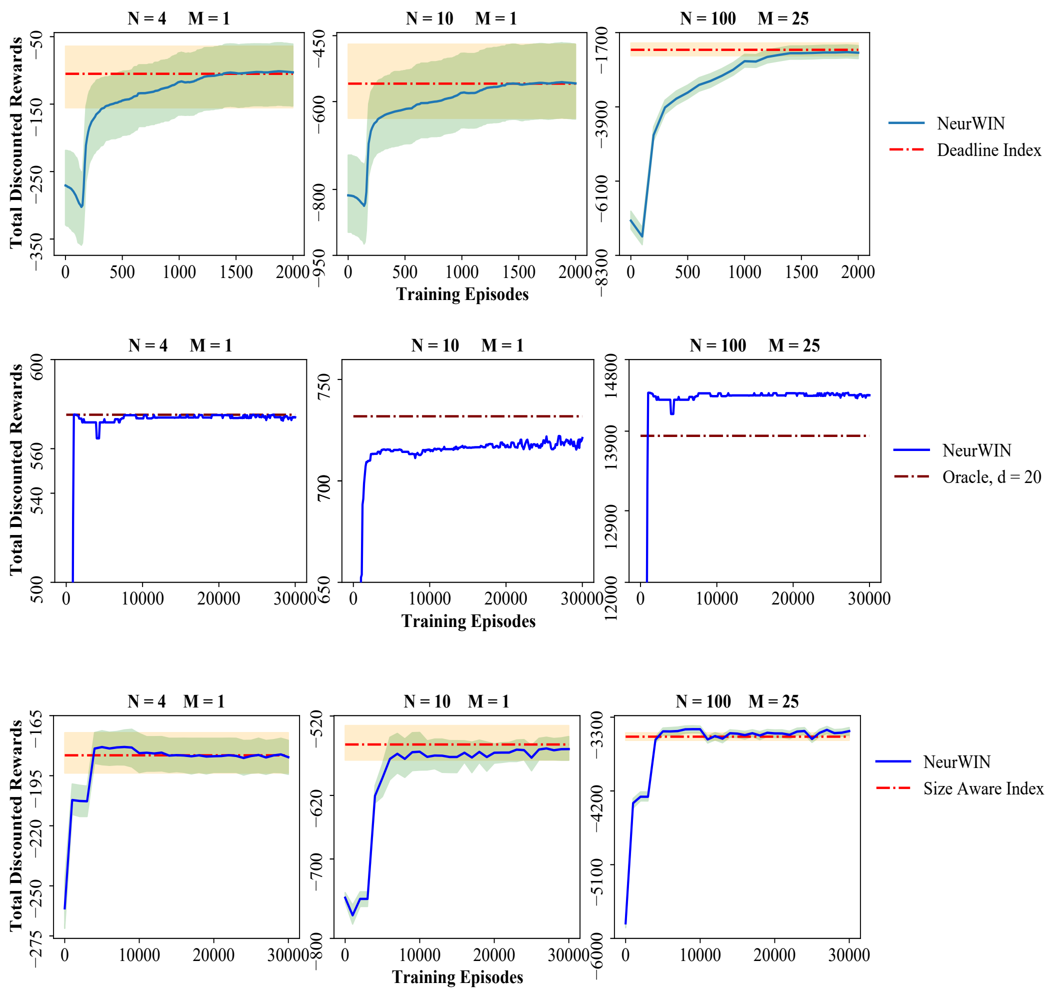

Figure 8: NeurWIN performance results for a smaller network size with 165 parameters per arm for the deadline and wireless scheduling cases, and 157 parameters for the recovering bandits' case.

Action $a[t]$**:** The agent can either activate the arm $a[t] = 1$, or leave it passive $a[t] = 0$. The next state changes based on two different transition kernels depending on the selected action. The reward is also dependent on the action at time t.

Reward $r[t]$**:** The agent, at time t, receives a reward $r[t]$ from the arm,

$$r[t] = \begin{cases} (1-c)a[t] & \text{if } B[t] > 0, D[t] > 1, \\ (1-c)a[t] - F(B[t] - a[t]) & \text{if } B[t] > 0, D[t] = 1, \\ 0 & \text{otherwise.} \end{cases} \tag{2}$$

In the equation above, c is a constant processing cost incurred when activating the arm, $F(B[t] - a[t])$ is the *penalty function* for failing to complete the job before $D = 1$. The penalty function was chosen to be $F(B[t] - a[t]) = 0.2(B[t] - a[t])^2$.

Next state $s[t+1]$**:** The next state $s[t+1]$ is given by

$$s[t+1] = \begin{cases} (D[t] - 1, B[t] - a[t]) & \text{if } D[t] > 1, \\ (D, B) \text{ w.p. } Q(D, B) & \text{if } D[t] \leq 1, \end{cases} \tag{3}$$

where $Q(D, B)$ is the arrival probability of a new job (i.e. a new electric vehicle arriving at a charging station) if the position is empty. For training and inference, we set $Q(0, 0) = 0.3$ and $Q(D, B) = 0.7/119$ for all $D > 0$, $B > 0$.

E.2 Training Setting

NeurWIN training is made for 2000 episodes on the deadline scheduling case. We save the trained model parameters at an interval of 10 episodes for inferring the control policy after training. The training produces 200 different set of parameters that output the estimated index given their respective training limit. The neural network had 625 trainable parameters given as layers $\{2, 16, 32, 1\}$, where the input layer matches the state size.

For the deadline scheduling training, we set the sigmoid value $m = 1$, episode's time horizon $T = 300$ timesteps, mini-batch size to 5 episodes, and the discount factor $\beta = 0.99$. The processing cost is set to $c = 0.5$. Training procedure follows section 4.2 from the main text. The arm randomly picks an initial state $s[t = 0] = (D, B)$, with a maximum $\bar{D} = 12$, and maximum $\bar{B} = 9$. The initial states are the same across episodes in one mini-batch for return comparison. The sequence of job arrivals in an episode's horizon is also fixed across a mini-batch. This way, the mini-batch returns are compared for one initial state, and used in tuning the estimated index value $f_\theta(\cdot)$.

At the agent side, NeurWIN receives the initial state $s[t = 0]$, sets the activation cost from a random state $\lambda = f_\theta(s_0)$. Training follows as described in NeurWIN's pseudo code.

For the MDP algorithms (REINFORCE, AQL, WOLP-DDPG), the training hyperparameters are the same as NeurWIN: Initial learning rate is $L[t = 0] = 0.001$, episode time horizon $T = 300$ timesteps, discount factor $\beta = 0.99$. Neural networks had two hidden layers with total parameters' count slightly larger than N number of NeurWIN networks for proper comparison.

QWIC was trained for the sets $\binom{4}{1}$ $\binom{10}{1}$ $\binom{100}{25}$ with a Q-table for each arm. QWIC selects from a set of candidate threshold values $\lambda \in \Lambda$ as index for each state. The algorithm learns Q function $Q \in \mathbb{R}^{\Lambda \times \mathcal{S} \times \{0,1\}}$. The estimated index $\tilde{\lambda}[s]$ per state s is determined during training as,

$$\tilde{\lambda}[s] = \underset{\lambda \in \Lambda}{\operatorname{argmin}} |Q(\lambda, s, 1) - Q(\lambda, s, 0)| \tag{4}$$

Initial timestep ϵ was selected to be $\epsilon_{max} = 1$, and at later timesteps set to be $\epsilon = \min(1, 2t^{-0.5})$. Other training hyperparameters: Initial learning rate $L[t = 0] = 0.001$, training episode time horizon of $T = 300$ timesteps, discount factor $\beta = 0.99$.

E.3 Inference Setting

In inferring the control policy, we test the trained parameters at different episode intervals. In other words, the trained models' parameters are tested at an interval of episodes, and their discounted rewards are plotted for comparison.

From the trained NeurWIN models described in E.2, we instantiate N arms, and activate M arms at each timestep based on their indices. For example, we load a NeurWIN model trained for 100 episodes on one arm, and set N arms each with its own trained agent on 100 episodes. Once the testing is complete, we load the next model trained at 110 episodes, and repeat the process for 110 episodes. The testing setting has the same parameters as the training setting with horizon $T = 300$ timesteps, and discount factor $\beta = 0.99$.

For the deadline Whittle index, we calculate the indices using the closed-form Whittle index and activate the highest M indices-associated arms. The accumulated reward from all arm (activated and passive) is then discounted with β.

For the MDP algorithms (REINFORCE, AQL, WOLP-DDPG), N arms combined states form the MDP state which is passed to the trained neural network. Reward is the sum of all rewards from the N arms. Testing is made for neural networks trained at different episode limits. The same testing parameters as NeurWIN were chosen: horizon $T = 300$ timestep, discount factor $\beta = 0.99$.

We perform the testing over 50 independent runs up to 2000 episodes, where each run the arms are seeded differently. All algorithms were tested on the same seeded arms. Results were provided in the main text for this setting.

Table 1: Θ values used in the recovering bandits' case.

Class	θ_0 Value	θ_1 Value
A	10	0.2
B	8.5	0.4
C	7	0.6
D	5.5	0.8

F Recovering Bandits' Training and Inference Details

F.1 Formulated Restless Bandit for the Recovering Bandits' Case

We list here the terms that describes one restless arm in the recovering bandits' case:

State $s[t]$**:** The state is a single value $s[t] = z[t]$ called the waiting time. The waiting time $z[t]$ indicates the time since the arm was last played. The arm state space is determined by the maximum allowed waiting time z_{max}, giving a state space $\mathcal{S} := [1, z_{max}]$.

Action $a[t]$**:** As with all other considered cases, the agent can either activate the arm $a[t] = 1$, or not select it $a[t] = 0$. The action space is then $\mathcal{A} := \{0, 1\}$.

Reward $r[t]$**:** The reward is provided by the recovering function $f(z[t])$, where $z[t]$ is the time since the arm was last played at time t. If the arm is activated, the function value at $z[t]$ is the earned reward. A reward of zero is given if the arm is left passive $a[t] = 0$. Fig. 9 shows the four recovering functions used in this work. The recovering functions are generated from,

$$f(z[t]) = \theta_0(1 - e^{-\theta_1 \cdot z[t]}) \tag{5}$$

Where the $\Theta = [\theta_0, \theta_1]$ values specify the recovering function. The Θ values for each class are provided in table 1.

Next state $s[t+1]$**:** The state evolves based on the selected action. If $a[t] = 1$, the state is reset to $s[t+1] = 1$, meaning that bandit's reward decayed to the initial waiting time $z[t+1] = 1$. If the arm is left passive $a[t] = 0$, the next state becomes $s[t+1] = \min\{z[t] + 1, z_{max}\}$.

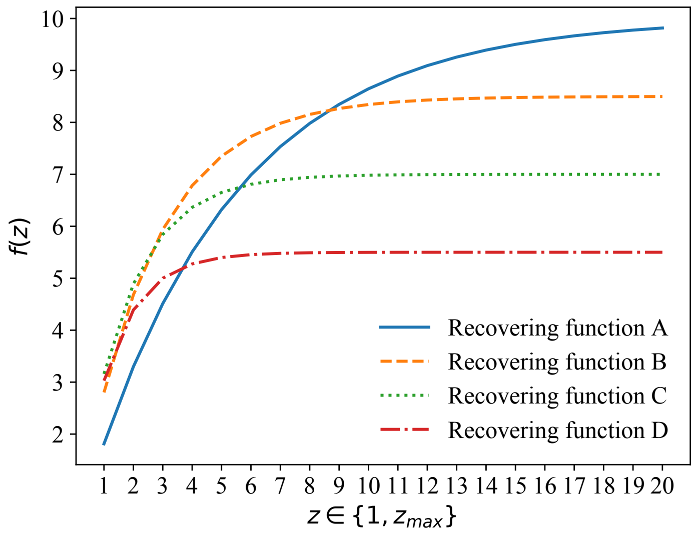

Figure 9: The selected recovering functions $f(z)$ for the recovering bandits' case.

F.2 Training Setting

Training procedure for NeurWIN algorithm follows the pseudocode in section 4. Here we discuss the parameter selection and details specific to the recovering bandits' case. We train the neural network using NeurWIN for $30,000$ episode, and save the trained parameters at an episode interval of 100 episodes. In total, for $30,000$ training episodes, we end up with 300 models for inference. The selected neural network has 609 trainable parameters with two hidden layers given as $\{1, 16, 32, 1\}$ neurons.

For training parameters, we select the sigmoid value $m = 5$, the episode's time horizon $T = 300$ timesteps, the mini-batch size to 5 episodes, and the discount factor $\beta = 0.99$. As with all other cases, each mini-batch of episodes has the same initial state $s[t = 0]$ which is provided by the arm. To ensure the agent experiences as many states in $[1, z_{max}]$ as possible, we set an initial state sampling distribution given as $Pr\{s[t = 0] = z\} = \frac{2^z}{2^1+2^2+...+2^{z_{max}}}$. Hence, the probability of selecting the initial state to be $s[t = 0] = z_{max}$ is 0.5.

At the agent side, we set the activation cost λ at the beginning of each mini-batch. λ is chosen to be the estimate index value $f_\theta(s_1)$ of a randomly selected state in $s_1 \in [1, z_{max}]$. The training continues as described in NeurWIN's pseudo code: the agent receives the state, and selects an action $a[t]$. If the agent activates the arm $a[t] = 1$, it receives a reward equal to the recovery function's value at z, and subtracts λ from it. Otherwise, the reward $r[t]$ is kept the same for $a[t] = 0$. We train NeurWIN independently for each of the four activation functions described in table 1.

For the MDP algorithms, training hyperparameters were selected as: Initial learning rate $L[t = 0] = 0.001$, discount factor $\beta = 0.99$. The algorithms are trained on the MDP representation, where the state is the combined states of the N arms.

QWIC training hyperparameters are the same. Training process is the same as in the deadline scheduling case from E.2.

F.3 Inference Setting

The inference setup measures NeurWIN's control policy for several $\binom{N}{M}$ settings. We test, for a single run, the control policy of NeurWIN over a time horizon $T = 300$ timesteps. We set N arms such that a quarter have one recovering function class from table 1. For $\binom{10}{1}$, we have three type A, three type B, two type C, and two type D arms.

At each timestep, the 8-lookahead and 20-lookahead policies rank the recovering functions reward values, and select the M arms with the highest reward values for activation. The incurred discounted reward at time t is the discounted sum of all activated arms' rewards. The total discounted reward is then the discounted rewards over time horizon $T = 300$. For inferring NeurWIN's control policy, we record the total discounted reward for each of the 300 models. For example, we instantiate N arms each having a neural network trained to $10,000$ episodes. At each timestep t, the neural networks provide the estimated index $f_{i,\theta}(s_i[t])$ for $i = 1, 2, \dots, N$. The control policy activates the M arms with the highest index values. We then load the model parameters trained on $10,100$ episodes, and repeat the aforementioned testing process using the same seed values.

With REINFORCE, AQL, WOLP-DDPG, testing happens for the same seeded arms as NeurWIN and d-lookahead policies. The MDP state is the combined states of N arms, and the total reward is the sum of all arm rewards. Testing is done for the 300 saved models, with discount factor $\beta = 0.99$ over horizon $T = 300$. QWIC and WIBQL are also tested for the 300 saved index mappings and Q-tables.

G Wireless Scheduling Training and Inference Details

G.1 Restless Arm Definition for the Wireless Scheduling Case

Here we list the state $s[t]$, action $a[t]$, reward $r[t]$, and next state $s[t + 1]$ that forms one restless arm:

State $s[t]$: The state is a vector $(y[t], v[t])$, where $y[t]$ is the arm's remaining load in bits, and $v[t]$ is the wireless channel's state indicator. $v[t] = 1$ means a good channel state and a higher transmission rate r_2, while $v[t] = 0$ is a bad channel state with a lower transmission rate r_1.

Action $a[t]$**:** The agent either activates the arm $a[t] = 1$, or keeps it passive $a[t] = 0$. The reward and next state depend on the chosen action.

Reward $r[t]$**:** The arm's reward is the negative of the *holding cost* ψ, which is a cost incurred at each timestep for not completing the job.

Next state $s[t+1]$**:** Next state is $[y[t+1], v[t+1]]$. Remaining load $y[t+1]$ equals $y[t] - a[t]r_d$, where $d = 2$ if $v[t] = 1$, and $d = 1$ if $v[t] = 0$. $v[t+1]$ is 1 with probability q, and 0, otherwise, where q is a parameter describing the probability that the channel is in a good state.

G.2 Training Setting

The neural network has 625 trainable parameters given as layers $\{2, 16, 32, 1\}$ neurons. The training happens for $30,000$ episodes, and we save the model parameters at each 1000 episodes. Hence, the training results in 30 models trained up to different episode limit.

For the wireless scheduling case, we set the sigmoid value $m = 0.75$, mini-batch size to 5 episodes, and the discount factor to $\beta = 0.99$. Maximum episode time horizon is set to $T = 300$ timesteps. The holding cost is set to $c = 1$, which is incurred for each timestep the job is not completed. We also set the good transmission rate $r_2 = 33.6$ kb, and the bad channel transmission rate $r_1 = 8.4$ kb. During training, we train NeurWIN on two different good channel probabilities, $q = 75\%$, and $q = 10\%$.

The episode defines one job size sampled uniformly from the range $y[t = 1] \sim (0, 1\text{ Mb}]$. All episodes in one mini-batch have the same initial state, as well as the same sequence of channel states $[v[t = 0], v[t = 1], \ldots, v[t = T - 1]]$.

At the agent side, NeurWIN receives the initial state $s[t = 0]$, and sets the activation cost from a random state $\lambda = f_\theta(s_1)$ for all timesteps of all mini-batch episodes. As mentioned before, we save the trained model at an interval of 1000 episodes. For $30,000$ training episodes, this results in 30 models trained up to their respective episode limit.

G.3 Inference Setting

Testing compares the induced control policy for NeurWIN with the size-aware index and learning algorithms. The algorithms' control policies are tested at different training episodes' limits. We instantiate N arms and activate M arms at each timestep t until all users' jobs terminate, or the time limit $T = 300$ is reached.

Half of the arms have a good channel probability 75%. The other half has a good channel probability 10%.

The size-aware index is defined as follows: at each timestep, the policy prioritizes arms in the good channel state, and calculates their *secondary index*. The secondary index $\hat{v}_i$ of arm i state $(y_i[t], v_i[t])$ is defined as,

$$\hat{v}_i(y_i[t], v_i[t]) = \frac{c_i r_{i,2}}{y_i[t]} \tag{6}$$

The size-aware policy then activates the highest M indexed arms. In case the number of good channel arms is below M, the policy also calculate the *primary index* of all remaining arms. The primary index v_i of arm i state $(y_i[t], v_i[t])$ is defined as,

$$v_i(y_i[t], v_i[t]) = \frac{c_i}{q_i[t](r_{i,2}/r_{i,1} - 1)} \tag{7}$$

Rewards received from all arms are summed, and discounted using $\beta = 0.99$. The inference phase proceeds until all jobs have been completed.

For NeurWIN's control policy, we record the total discounted reward for the offline-trained models. For example, we set N arms each coupled with a model trained on $10,000$ episodes. The models output their arms' indices, and the top M indexed arms are activated. In case the remaining arms are less than M, we activate all remaining arms at timestep t. timestep reward $\beta^t R[t] = \beta^t \sum_{i=1}^{N} r_i[t]$ is the sum of all arms' rewards. Once testing for the current model is finished, we load the next model $11,000$ for each arm, and repeat the process. For the MDP algorithms, the MDP state is the combined states of all N arms, with the reward being the sum of arms' rewards.

We note that the arms' initial loads are the same across runs, and that the sequence of good channel states is random. For all algorithms, we average the total discounted reward for all control policies over 50 independent runs using the same seed values.