State-Aware Value Function Approximation with Attention Mechanism for Restless Multi-armed Bandits

Shuang Wu[1] , **Jingyu Zhao**[1,2] , **Guangjian Tian**[1] , **Jun Wang**[1,3]
[1]Huawei Noah's Ark Lab
[2]The University of Hong Kong
[3]University College London
{wu.shuang1, zhao.jingyu, Tian.Guangjian, w.j}@huawei.com

Abstract

The restless multi-armed bandit (RMAB) problem is a generalization of the multi-armed bandit with non-stationary rewards. Its optimal solution is intractable due to exponentially large state and action spaces with respect to the number of arms. Existing approximation approaches, e.g., Whittle's index policy, have difficulty capturing either temporal or spatial factors such as impacts from other arms. We propose considering both factors using the attention mechanism, which has achieved great success in deep learning. Our state-aware value function approximation solution comprises an attention-based value function approximator and a Bellman equation solver. The attention-based coordination module capture both spatial and temporal factors for arm coordination. The Bellman equation solver utilizes the decoupling structure of RMABs to acquire solutions with significantly reduced computation overheads. In particular, the time complexity of our approximation is linear in the number of arms. Finally, we illustrate the effectiveness and investigate the properties of our proposed method with numerical experiments.

1 Introduction

Restless multi-armed bandit (RMAB) [Whittle, 1988] problems are non-stationary extensions of the multi-armed bandit (MAB) [Robbins, 1952] problem. Unlike the standard MAB, the arms have non-stationary reward distributions. The additional dynamics enable RMAB to model a broader range of applications than MAB, including clinical trials, aircraft surveillance, and worker scheduling [Whittle, 1988], as well as modeling robotic coordination [Argand, 1998], machine maintenance [Abad and Iyengar, 2015], recommendation system [Meshram *et al.*, 2017], multi-channel accessing [Liu and Zhao, 2010], and network resource scheduling [Kadota *et al.*, 2018].

The dynamics of RMAB are specified by Markov chains with rewards. Each arm's state transits in either an active or a passive mode defined by two different Markov chains. Each time step, a controller can at most activate M out of N arms and receive rewards depending on the current states and actions. The goal of the controller is to find a control policy to maximize the expected long-term rewards. Optimal scheduling policy requires considering both impacts across arms (spatial factors) and impacts of current action on future rewards (temporal factors).

Solving an RMAB is computationally intractable due to the exponential growth rate of the state and action spaces with respect to the number of arms, which is known as the curse of dimensionality. Papadimitriou and Tsitsiklis (1999) showed that, even under deterministic transitions, finding the optimal solution is PSPACE-hard. From a high-level perspective, coordinating the arms ideally should consider both spatial and temporal factors. The combinatorial nature of huge state-action space contributes to high spatial complexity, further amplified by the sequential structure from a temporal perspective.

Approximate solutions for RMAB need to trade off optimality for efficiency by ignoring certain factors. The approaches can be generally categorized into index policies and approximate dynamic programming. Index policies overlook impacts across arms, while general approximate dynamic programming is not sample efficient in capturing the temporal relations.

Whittle (1988) proposed an index-based heuristic for RMAB. In each time step, the controller selects the arms with the top index values, which can be roughly interpreted as future rewards. The index values are computed independently for each arm. Although this makes index policy free of the curse of dimensionality, impacts across arms are not well-captured as the impact of state transitions of other arms are overlooked. Although the index policy is optimal for the stationary MAB [Gittins, 1979], it is only asymptotically optimal for RMAB in a limiting regime [Weber and Weiss, 1990] where spatial impacts can be averaged out. Moreover, computation of indexes is restricted to an indexability condition, which is difficult to establish in general [Glazebrook *et al.*, 2005; Ouyang *et al.*, 2015].

RMAB can be alternatively formulated as a large-scale Markov decision process (MDP). Solving an MDP is not subject to the indexability restriction, but approximation quality is often compromised due to high computational costs in capturing the temporal factors. The general approximate dynamic programming framework [Bertsekas and Tsitsiklis, 1996] utilizes the state transitions as a simulator and approx-

imates the optimal solution through sampling from generated state trajectories. It is shown that the general sample complexity of this type of approach is exponential in the number of arms [Lattimore *et al.*, 2013].

In this work, we develop a method capturing both spatial and temporal factors to solve RMAB in its MDP formulation with linear complexity in the number of arms. We consider approximating the joint value function through a linear combination of each arm's value functions. We supply our approximated value function to a greedy policy to approximate the optimal solution. We propose a state-adaptive value-tuning mechanism to allow state-aware value function approximation. The complicated spatial and temporal factors in arm coordination are captured using the attention mechanism proposed in the Transformer model [Vaswani *et al.*, 2017]. We leverage the linear approximation form and the decomposable structures in RMABs to achieve linear complexity in computing the Bellman residual. These steps transform solving an RMAB into training the weights in the Transformer. We use a stochastic gradient descent (SGD)-based algorithm to train the weights, which is free of the curse of dimensionality.

The contributions of our approach are as follows.

1. Our method demonstrates a way of using the attention mechanism for capturing coupling factors in large-scale coupled systems such as RMABs.
2. We adopt SGD to Bellman residual minimization to acquire optimal value approximations. Our algorithm circumvents the curse of dimensionality faced by brute-force dynamic programming.
3. We validate the effectiveness and linear complexity of our approach for solving RMABs through experiments.

2 Related Works

Most works in RMAB focus on index-based policies, including index policies for different setups [Abad and Iyengar, 2015; Meshram *et al.*, 2017; Kadota *et al.*, 2018], optimality conditions of index policies [Weber and Weiss, 1990; Liu and Zhao, 2010], and regret bounds of learning indices for unknown models [Tekin and Liu, 2012; Jung and Tewari, 2019]. Very few works studied other approximations of RMAB solution [Guha *et al.*, 2010].

Linear value function approximation has been studied for factored MDP [Guestrin *et al.*, 2003], weakly coupled MDP [Meuleau *et al.*, 1998], and, more recently, multi-agent reinforcement learning [Sunehag *et al.*, 2018], focusing on decentralized coordination. Factored MDP is suitable for this form due to its decomposable structure described by a dynamic Bayesian network. However, this approximation form is imposed globally and is not adapted to local states. Meuleau *et al.* (1998) suggested a domain-specific heuristic to address local adaption. Our work considers a general state-dependent adaption mechanism that goes beyond the domain-specificity.

The self-attention mechanism is the backbone of the Transformer model, which has achieved remarkable successes in natural language processing [Vaswani *et al.*, 2017], and recently, computer vision tasks [Dosovitskiy *et al.*, 2021]. Recent works have applied attention mechanisms in multi-agent reinforcement learning to improve learning efficiency [Jiang and Lu, 2018]. The approaches in multi-agent reinforcement learning focus on decentralized decision making without a centralized coordinator. However, we consider a centralized scheduling setup.

3 Problem Setup

We specify the restless multi-armed bandit (RMAB) problem and relevant notations in this section.

3.1 RMAB

RMAB is related to scheduling multiple Markov decision processes (MDPs). An MDP $\{\mathbb{S}, \mathbb{A}, P, R\}$ includes a state space $\mathbb{S}$, an action space $\mathbb{A}$, a state transition law specified by a conditional probability distribution $P(s'|s,a)$ with $s, s' \in \mathbb{S}$, $a \in \mathbb{A}$, and a per-stage reward function $R(s,a)$. We consider maximizing the discounted cumulative reward over an infinite horizon criterion as $\max_{\{a^{(k)}\}} \mathbb{E}[\sum_{k=0}^{\infty} \gamma^k R(s^{(k)}, a^{(k)})]$, where $0 < \gamma < 1$ is a discount factor and the superscript k is the time step.

An RMAB with N arms consists of N heterogeneous MDPs $\{\mathbb{S}_i, \mathbb{A}_i, P_i, R_i\}$ for all $i = 1, \ldots, N$. Without loss of generality, we assume that $R_i(\cdot,\cdot) \geq 0$ and $|\mathbb{S}_i| = S$ for all i. Every arm has a binary action space $\mathbb{A}_i = \{0, 1\}$ with $a_i = 0$ and 1 for a passive and an active transition mode, respectively. One needs to maximize the cumulative individual long-term rewards $\mathbb{E}[\sum_{k=0}^{\infty} \sum_{i=1}^{N} \gamma^k R_i^{(k)}(s_i^{(k)}, a_i^{(k)})]$ by finding a sequence of collective actions $\{a^{(k)}\}_{k=0}^{\infty}$ with $a^{(k)} = [a_1^{(k)}, \ldots, a_N^{(k)}]$ subject to a cardinality constraint $\sum_{i=1}^{N} a_i^{(k)} \leq M$ for all k. An RMAB can be viewed as a combined MDP. The cardinality constraint couples the arms by restricting the action space of the corresponding MDP of the RMAB.

3.2 Value Functions and Q-Factor

Let the vectors s and a denote the aggregated state $[s_1, \ldots, s_N]$ and action $[a_1, \ldots, a_N]$, respectively. A value function refers to any real-valued function on a state space. It can be interpreted as the estimated future rewards for each state. Given a value function V, we define a Q-factor,

$$Q(s,a) := R(s,a) + \gamma \mathbb{E}_{P(s'|s,a)}[V(s')], \tag{1}$$

where $\mathbb{E}_{P(x)}[f(x)] := \int f(x) P(\mathrm{d}x)$ denotes the expectation of $f(x)$ under the probability measure P. The Q-factor is the estimated future rewards under action a. An optimal value function $V^\star(s)$ defines the maximal achievable future rewards for each state under an optimal policy. It satisfies the Bellman optimality equation,

$$V^\star(s) = \max_a Q^\star(s,a), \tag{2}$$

where $Q^\star := R(s,a) + \gamma \mathbb{E}_{P(s'|s,a)}[V^\star(s')]$. The optimal value functions of each arm and the RMAB are $V_i^\star(s_i)$ and $V^\star(s)$, respectively.

3.3 Value-Based Policy

Given a value function $V(s)$, we can produce a greedy control policy for the value function by

$$a = \pi(s) = \arg\max_{u \in \mathbb{A}} Q(s, u). \tag{3}$$

If $V = V^\star$, the greedy policy π is optimal. One can approximate an optimal policy by approximating $V^\star$. Let V^π denote the actual reward under π. There is a strict performance bound for this approach [Williams and Baird, 1993],

$$\underbrace{\|V^\star - V^\pi\|_\infty}_{\text{performance gap of greedy policy } \pi} \leq \frac{1}{1-\gamma} \underbrace{\|\max Q - V\|_\infty}_{\text{Bellman residual}}. \tag{4}$$

4 Our Approach

We approximate the optimal solution of RMAB through a value-based policy. Our proposed value function is developed upon the following general form, where it is approximated as the summation of each arm's value functions

$$V(s^{(k)}) = \sum_{i=1}^{N} V_i(s_i^{(k)}). \tag{5}$$

A straightforward approach of acquiring V_i is to minimize the Bellman residual in (4). However, V_i is restricted to every single arm, which ignores the non-stationary impacts from other arms. Contrarily, V_i should adapt to the changes of other arms' states.

To promote state-awareness, we propose a state-dependent value-tuning method to adjust V_i based on the global state $s^{(k)}$. In particular, we consider the arm value function in the following form

$$V_i(s_i^{(k)}) := V_i^w(s_i|s^{(k)}) = w_i(s_i|s^{(k)}) V_i^\star(s_i). \tag{6}$$

To capture cross-arm spatial impacts, we let the weights w_i to be dependent on the *global state*. Our approximation (6) allows the arm value functions to adapt to the changes in the states of other arms, even when s_i remains the same. And the weights will be trained to best-estimate future rewards, which seizes the temporal impacts. Thus, the representation power of our approximation is significantly stronger than the ordinary approximation (5).

The advantages of our approximation form are two-fold. **Interpretability**. The weights w_i demonstrate the importance of other states given the current state $s^{(k)}$. **Flexibility**. As $w_i(\cdot|s^{(k)})$ is determined by $s^{(k)}$, we are allowed to have distinct value approximations $V(s^{(k)}) = \sum_{i=1}^{N} V_i^w(s_i|s^{(k)})$ for different $s^{(k)}$.

4.1 System Outline

We adopt the attention mechanism to represent the state-adaptive weights and solve the network weights by a Bellman equation solver. The architecture of our approach is shown in Figure 1. The diagram consists of the attention-based approximator and a Bellman equation solver. The approximator generates re-weighted future rewards for each arm. The solver utilizes the decoupling structure in RMAB to generate

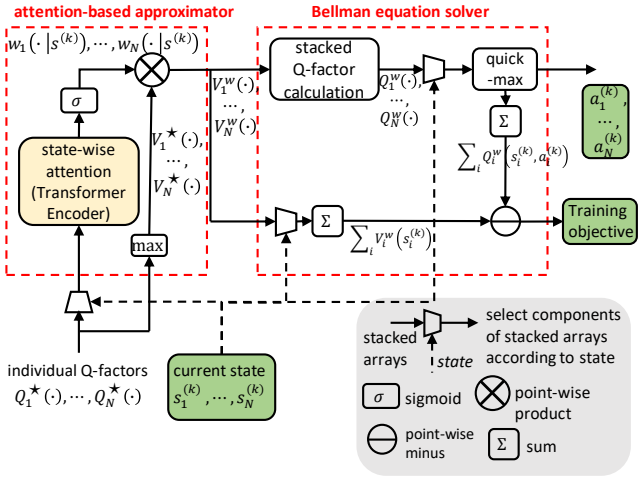

Figure 1. Attention-based linear value function approximation.

scheduling decisions and back-propagates gradient information from Bellman residual for steering the approximator to satisfy the Bellman optimality equation.

Let $s^{(k)} := [s_1^{(k)}, \ldots, s_N^{(k)}]$ be the state whose value we aim to approximate at time k. The approximator extracts weights $w_i(s_i|s^{(k)})$ for all $s_i \in \mathbb{S}_i$ for $i = 1, \ldots, N$. We then calculate Q-factors of each arm based on arms' reweighed value functions by

$$Q_i^w(s_i, a_i) = R_i(s_i, a_i) + \gamma \sum_{s_i'} P_i(s_i'|s_i, a_i) V_i^w(s_i).$$

We compute the approximate optimal action $a^{(k)}$ by solving

$$\max_a \quad Q^w(s^{(k)}, a) := \sum_i Q_i^w(s_i^{(k)}, a_i), \tag{7a}$$

$$\text{subject to} \quad \sum_i a_i \leq M, \tag{7b}$$

through a quick-max algorithm discussed later. To learn the weights in the attention module, we propose to minimize the corresponding square of the Bellman residual for $s^{(k)}$

$$B(s^{(k)}) := \Big[\sum_{i=1}^{N} V_i^w(s_i^{(k)}) - Q_i^w(s_i^{(k)}, a_i^{(k)}) \Big]^2. \tag{8}$$

We now proceed to explain how each operation works.

4.2 Attention-Based Approximator

We aim to solve RMABs with state-adaptive coordination of all arms. For the value-based approximation strategy, the scheduling decision is determined by the arms' value function. This module is responsible for adjusting the arms' value functions and coordinating the arms to achieve larger cumulative rewards. In particular, this is done by generating discounting weights $w_i(s_i|s^{(k)})$ to adjust the overestimated cumulative rewards $V_i^\star(s_i)$ for each arm.

Attention mechanism has demonstrated representational power in extracting both temporal and spatial information

and has achieved remarkable success in both natural language processing and image recognition. We use the Transformer encoder [Vaswani *et al.*, 2017] to extract state-aware information for multi-arm coordination. Coordinating multi-arms requires considering both spatial (cross-arm impact) and temporal (impact on future rewards) factors. This module intakes the arms' optimal Q-factors for the current state, i.e., $Q_i^\star(s_i^{(k)}, a)$ with $a = 0, 1$ for every arm i. Through scaled-product attention, spatial influences among the arms are captured. We then pass the extracted spatial and temporal information to the arms' value functions $V_i^\star$ to adjust the expected future rewards for each arm.

Recall that the cardinality constraint in RMAB restricts the flexibility of the arms' actions during coordination. Consequently, the sum of the arms' optimal value functions is an upper bound for all achievable value functions, i.e., $\sum_i V_i^\star(s_i) \geq V(s)$. Thus, it is necessary to ensure that the weights $w_i(\cdot|s^{(k)})$ are indeed discounting the optimal value functions for each arm. Recall that we assume non-negative rewards, which ensures that all value functions are nonnegative as well.[1] Therefore, it suffices to ensure that $0 \leq w_i \leq 1$, which is done by applying the sigmoid function $\sigma(x) = \frac{1}{1+\exp(-x)}$ to the output of the Transformer encoder.

Bellman Equation Solver

For value-based policy, we need to evaluate the Q-factor for computing greedy actions as prescribed by the value-based approach in (1) and (3). The maximum value is further used to produce losses for solving the Bellman optimality equation. However, direct computation of the Q-factor involves the combined P and R, which leads to exponential time complexity in N, and the maximization is combinatorial in N and M. We address the first issue using a value decomposition and the second one using a quick-max approximation algorithm. As a result, both computing the Q-factor and the maximization can be done with linear time complexity in N. The complexity reduction techniques are developed based on the linear approximation form (5) and independent of the state-adaptive weights w. In this subsection, we drop the related superscript w to simplify notations.

4.3 Value decomposition

We need to evaluate $Q(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s')$. For a fixed a, the size of the reward function $R(s,a)$, transition matrix $P(s'|s,a)$, and value function $V(s)$ are exponential in N. We find that the exponential complexity can be reduced to a linear one through a following decomposition.

Note that, under the linear approximation (5), there holds

$$\begin{aligned}\mathbb{E}_{s'\sim P(s|s,a)}[V(s')] =& \mathbb{E}_{s'\sim P(s|s,a)}[\sum_i V_i(s_i')] \\ =& \sum_i \mathbb{E}_{s'\sim P(s|s,a)}[V_i(s_i')] \\ =& \sum_i \mathbb{E}_{s_i'\sim P_i(s_i'|s_i,a_i)}[V_i(s_i')].\end{aligned}$$

Plug this into $Q(s,a)$, we can obtain

$$\begin{aligned}Q(s,a) =& \sum_{i=1}^N R_i(s_i,a_i) + \gamma \mathbb{E}_{s'\sim P(s|s,a)}[V(s')] \\ =& \sum_{i=1}^N R_i(s_i,a_i) + \gamma \mathbb{E}_{s_i'\sim P_i(s_i'|s_i,a_i)}[V_i(s_i')] \\ =& \sum_{i=1}^N Q_i(s_i,a_i),\end{aligned}$$

The result is straightforward as it says that the joint Q-factor can be obtained by summing the individual Q-factors under a linear approximation (5). This shows the compatibility of linear approximation for the independent transitions and rewards of RMAB. As a result, the time complexity of computing the Q-factor is now linear in N.

Contrarily, if one considers a general value function approximation such as a multi-layer neural network, decomposing the Q-factor is infeasible. This yields the curse of dimensionality if one wants to compute $Q(s,a)$ as the combined P and R are involved.

Quick maximization

A key step in evaluating the Bellman optimality operator is to solve the combinatorial optimization in (7). Enumerating all $\sum_{k=0}^M \binom{N}{k}$ feasible actions yields exponential complexity. We overcome this issue by applying a quick-max algorithm, which sequentially determines a_i with a local greedy decision under the global constraint $\sum_{i=1}^N a_i \leq M$ in a fixed order. The pseudo code of the algorithm is in the Appendix for reference. The corresponding time complexity of quick-max is at most $O(N)$.

As a side note, the quick-max algorithm is a greedy approximation for the actual maximization problem. Nevertheless, optimality can still be reserved as the value functions are adjusted to accommodate this factor through the learning procedure.

4.4 Training the Approximator

We use the square of the Bellman residual as the loss function for the attention-based approximator to adjust its weights. Recall that we assume $|\mathbb{S}_i| = S$ for all arms. There are S^N values to be approximated for $V(s)$. Consequently, the Bellman optimality equation (2) involves S^N separate equations. To promote scalability, we train the approximator using a mini-batch SGD training approach. That is, we sample a mini-batch of states in each iteration, compute the approximated Q-factors and the corresponding maximizers a, and update the Transformer encoder based on the square Bellman residuals in the mini-batch.

The curse of dimensionality caused by solving the combined MDP with exponentially increasing state and action spaces is now alleviated by learning the coordination weights in the Transformer encoder. By training with SGD, we let the Transformer acquire the weights from sampled states. As the optimal value function is determined by state transitions and rewards with a total degree of freedom linear in N, we can expect that the total sample complexity is at most polynomial

[1] The nonnegativity can be ensured by re-defining $\tilde{R}_i(s_i,a_i) = R_i(s_i,a_i) - \min_{s_i,a_i} R_i(s_i,a_i)$.

in N. In the experiments section, we verify the effectiveness of the mini-batch training approach.

4.5 Time Complexity and Error Analysis

The time complexity of solving an RMAB relies on both the approximator and the Bellman equation solver. The approximator module consists of a Transformer encoder network, a sigmoid function, and an elementwise product. Since all these sub-modules allow parallel computation, the time complexity of the coordinator module is independent of N. As for the Bellman equation solver, linear time complexity in N is ensured by the value decomposition and the quick-max algorithm. Note that the linear time complexity is the same as computing Whittle indices. The advantage of our approach is that we can capture cross-arm impacts, which is ignored when computing the indices.

The accuracy of a general numeric method involving sampling is affected by three factors: 1) approximation error due to modeling effects, 2) optimization error due to accuracy of the optimization solver, and 3) estimation error due to random sampling. By using the state-adaptive tuning mechanism, our model can capture any achievable value function, which leads to zero approximation error. Although an overparameterized neural network theoretically can reduce the optimziaiton error to zero [Allen-Zhu *et al.*, 2019], for any finite training epochs in practice, zero error is not achievable. The estimation error depends on the sample collection. Ideally, sampling the full state space reduces the error to be zero. This is not scalable due to the exponential complexity in the state space. We adopt SGD-based training to trade-off estimation error for scalability.

5 Numerical Results

We study the effectiveness of our approach for RMABs with N arms under a per-time activation constraint $\sum_{i=1}^{N} a_i^{(k)} \leq M$. We compare the performance with Whittle's index policy [Whittle, 1988] and a myopic policy. We run ablation experiments to study the effectiveness of our proposed modules. Finally, we study the practicality of our approach through experiments on hyperparameter sensitivity and scalability. Details of experiment environments are in the appendix.

5.1 Performance Comparison

Control policies. We compare our approach with myopic policy and Whittle's index policy. The myopic policy chooses the action that maximizes the per-stage reward of all arms. We use the quick-max algorithm to approximate maximization steps to ensure scalability.

Arm dynamics and rewards. We set discount factor $\gamma = 0.9$ and $S = 10$ for all arms. We consider two kinds of arms (indexable and nonindexable) defined by a parameterized restart process. In particular, the arm dynamics is described by a controlled restart process [Akbarzadeh and Mahajan, 2019] with passive and active transition respectively being

$$P_i(s_i'|s_i, 0) = \begin{cases} p_i, & s_i' = s_i, \\ (1-p_i)/(S-1), & s_i' \neq s_i, \end{cases}$$

and $P_i(s_i' = 0|s_i, 1) = 1$. To accommodate our positive rewards assumption, we set the reward function as $R_i(s_i, 0) = (S-1)^2 - (s_i - 1)^2$, and $R(s_i, 1) = 0$. For all experiments, we generate arms with randomly chosen p_i. According to [Akbarzadeh and Mahajan, 2019], the arms are indexable if $p_i \in \texttt{Uniform}[1/(S-1), 1]$. We consider both indexable and nonindexable arms with $p_i \in \texttt{Uniform}[1/(S-1), 1]$ and $p_i \sim \texttt{Uniform}[0, 1/(S-1))$, respectively.

Computation of the index values. Recall from [Whittle, 1988] that an index λ for state s_i is defined by

$$Q_i^\star(s_i, 1; \lambda) = Q_i^\star(s_i, 0; \lambda), \tag{9}$$

where

$$Q_i^\star(s_i, a_i; \lambda) := \max_{\{a_i^{(k)}\}_{k=1}^{\infty}} \mathbb{E}\Big\{ \sum_{k=0}^{\infty} \gamma^k [r_i(s_i^{(k)}, a_i^{(k)}) - \lambda \cdot a_i^{(k)}] \Big| (s_i^{(0)}, a_i^{(0)}) = (s_i, a_i) \Big\}.$$

We use a binary search to find each λ by solving (9), which does not require the indexability condition. The obtained value is not necessarily an index if the indexability cannot be verified. Nevertheless, the quantity still reflects the relative advantage of being active over passive for the current arm and is thus an indicator for arm importance, which can be used for an index policy.

Hyperparameters. We use a one-layer transformer encoder with hidden dimension 512 and 8 heads. The mini-batch size is set as 128 with 100 training epochs. The optimizer is chosen as the Adam with a 0.001 learning rate.

Metrics. We use the empirical cumulative rewards as the metrics for performance comparison and ablations study. For hyperparameter sensitivity and scalability study, we choose the square Bellman residual as the metric. We refer to it as the pseudo Bellman residual as only a subset of the states is used to compute.

Results. We simulate RMAB with three policies with both indexable and non-indexable arms. We consider three cases: $N = 10$ with $M = 3$, $N = 20$ with $M = 6$, and $N = 50$ with $M = 15$. For each case, every policy is simulated with a fixed but randomly generated 500 initial states. Table 1 shows the mean (and standard deviations in the bracket) of cumulative rewards for each policy in the first 50 time epochs. Index policy shows an advantage over the myopic policy for indexable arms, while our approach achieves the best performance among the three.

5.2 Ablation Study

We compare our proposed approach with the following cases to show the importance of our proposed submodules:

- -approximator: we remove the attention-based approximator and directly solve the Bellman residual minimization, $\min_V \|V - \max_a Q\|^2$. Note that this is equivalent to finding optimal static weights to adjust the arm value function;
- -Transformer+MLP: we replace the Transformer with a multi-layer perception, which comprises two hidden layers and the same total number of neurons as the Transformer;

	indexable		
N	10	20	50
myopic	7040.7(270.0)	13934.1(453.2)	34976.7(674.0)
index	7072.5(83.5)	14000.1(117.9)	35209.5(184.0)
ours	7476.6(147.9)	14899.9(223.4)	37262.1(326.0)
	nonindexable		
N	10	20	50
myopic	6935.2(197.3)	13719.5(316.1)	34491.0(457.3)
index	6980.6(87.0)	13799.0(123.1)	34736.8(190.4)
ours	7233.6(158.8)	14404.5(231.6)	36090.7(362.9)

Table 1. Cumulative rewards for different policies.

method	cumulative rewards mean (std. deviation)
ours	9382.1(256.1)
-approximator	4743.5(464.3)
-Transformer+MLP	8578.1(373.4)
-sigmoid	6412.1(411.2)

Table 2. Ablation study for different modules.

- -sigmoid: we keep the Transformer, but remove the sigmoid function on the output.

We test the above methods with $N = 10$, $M = 4$ for randomly generated nonindexable arms. We compare different approaches by following the same simulation setup as before. Table 2 shows the mean and the standard deviation of the cumulative rewards with 500 initial states for different cases. Our method outperforms other variants, which shows that all considered submodules contribute to performance improvements. Moreover, in the case without using the sigmoid function, both the mean and variation of accumulative rewards reward and variation degrade significantly, which indicates that discounting the future reward for each arm is critical for obtaining a better value function estimation.

5.3 Hyperparameter Sensitivity and Scalability

The major hyperparameters that potentially affect the performance of our approach include the mini-batch size, training epochs, and the hidden dimension of the transformer encoder. The mini-batch size and epoch numbers affect the training process while the hidden dimension affects the learning capacity of the model. We examine these factors by comparing the obtained pseudo Bellman residuals under $N = 10$ with $M = 5$. Relevant results are presented in Figure 2 and 3 with error bars indicating the means and standard deviations. Figure 2 shows that the Bellman residual converges quite fast and the large size of the mini-batch is critical to achieving a small residual. The left of Figure 3 shows that larger N requires a higher hidden dimension (more neurons) to learn the coordination. Moreover, larger mini-batch and hidden dimensions will hurt residual but saturate once they are large enough.

Scalability with respect to N is critical for solving RMABs. We have shown that our method yield liner time

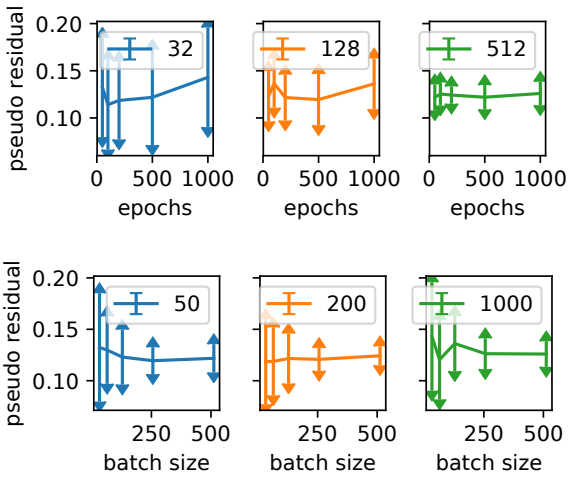

Figure 2. Sensitivity with respect to the batch sizes and the total training epochs. (Top) fixed batch size; (bottom) fixed epochs.

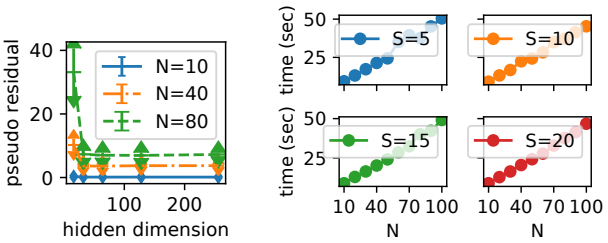

Figure 3. (Left) sensitivity in the size of the hidden dimension; (right) linear time complexity in N using our approach.

complexity. We now illustrate this through experiments. We solve RMAB with varying N with iteration epochs fixed to be 100 and 128 samples per epoch. We record the time of each instance. The time spent for solving each tested instance is shown in the right Figure 3. The time complexity increases linearly with respect to N and is not affected by S, the number of states in each arm.

6 Conclusions

We have proposed a novel approach for solving RMAB beyond the Whittle's index policy and the classic approximate dynamic programming. Our method combines a state-adaptive linear value function approximator and a Bellman equation solver to solve the combined MDP of the RMAB. The adaptation is realized by adopting the attention mechanism, which captures the complicated multi-arm coordination involving both spatial and temporal factors for multi-arm coordination. Directly solving the combined MDP is computationally intractable due to curse of dimensionality. By using the state-adaptive approximation, the difficulty is transferred to training a neural network, which can be done efficiently using SGD-based mini-batch training. Our result indicates that significant efficiency can be gained with little loss in accuracy for solving weakly coupled systems. We hope that the ideas can be extended to problems with similar weak coupling structures, such as solving the stationary distribution of multiple coupled queues.

References

[Abad and Iyengar, 2015] Carlos Abad and Garud Iyengar. A near-optimal maintenance policy for automated DR devices. *IEEE Trans. Smart Grid*, 7(3):1411–1419, 2015.

[Akbarzadeh and Mahajan, 2019] Nima Akbarzadeh and Aditya Mahajan. Restless bandits with controlled restarts: Indexability and computation of Whittle index. In *Proc. IEEE Conf. Dec. Contr.*, pages 7294–7300, 2019.

[Allen-Zhu *et al.*, 2019] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A convergence theory for deep learning via over-parameterization. In *Proc. Int. Conf. Mach. Learn.*, pages 242–252. PMLR, 2019.

[Argand, 1998] Emile Argand. Behaviors coordination using restless bandits allocation indexes. In *Proc. Int. Conf. Simul. Adapt. Behav.*, volume 5, page 159. MIT Press, 1998.

[Bertsekas and Tsitsiklis, 1996] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[Dosovitskiy *et al.*, 2021] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. Int. Conf. Learn. Represent.*, 2021.

[Gittins, 1979] John C Gittins. Bandit processes and dynamic allocation indices. *J. R. Stat. Soc. Series B*, 41(2):148–164, 1979.

[Glazebrook *et al.*, 2005] Kevin D Glazebrook, HM Mitchell, and PS Ansell. Index policies for the maintenance of a collection of machines by a set of repairmen. *Eur. J. Oper. Res.*, 165(1):267–284, 2005.

[Guestrin *et al.*, 2003] Carlos Guestrin, Daphne Koller, Ronald Parr, and Shobha Venkataraman. Efficient solution algorithms for factored MDPs. *J. Artif. Intell. Res.*, 19:399–468, 2003.

[Guha *et al.*, 2010] Sudipto Guha, Kamesh Munagala, and Peng Shi. Approximation algorithms for restless bandit problems. *J. ACM*, 58(1):1–50, 2010.

[Jiang and Lu, 2018] Jiechuan Jiang and Zongqing Lu. Learning attentional communication for multi-agent cooperation. In *Adv. Neural Inf. Process. Syst.*, pages 7254–7264, 2018.

[Jung and Tewari, 2019] Young Hun Jung and Ambuj Tewari. Regret bounds for Thompson sampling in episodic restless bandit problems. In *Adv. Neural Inf. Process. Syst.*, pages 9007–9016, 2019.

[Kadota *et al.*, 2018] Igor Kadota, Abhishek Sinha, and Eytan Modiano. Optimizing age of information in wireless networks with throughput constraints. In *Proc. IEEE Conf. Comput. Commun.*, pages 1844–1852, 2018.

[Lattimore *et al.*, 2013] Tor Lattimore, Marcus Hutter, Peter Sunehag, et al. The sample-complexity of general reinforcement learning. In *Proc. Int. Conf. Mach. Learn.*, 2013.

[Liu and Zhao, 2010] Keqin Liu and Qing Zhao. Indexability of restless bandit problems and optimality of Whittle index for dynamic multichannel access. *IEEE Trans. Inf. Theory*, 56(11):5547–5567, 2010.

[Meshram *et al.*, 2017] Rahul Meshram, Aditya Gopalan, and D Manjunath. Restless bandits that hide their hand and recommendation systems. In *Proc. Int. Conf. Commun. Syst. Netw.*, pages 206–213, 2017.

[Meuleau *et al.*, 1998] Nicolas Meuleau, Milos Hauskrecht, Kee-Eung Kim, Leonid Peshkin, Leslie Pack Kaelbling, Thomas L Dean, and Craig Boutilier. Solving very large weakly coupled Markov decision processes. In *Proc. AAAI Conf. Artif. Intell.*, pages 165–172, 1998.

[Ouyang *et al.*, 2015] Wenzhuo Ouyang, Sugumar Murugesan, Atilla Eryilmaz, and Ness B Shroff. Exploiting channel memory for joint estimation and scheduling in downlink networks—a Whittle's indexability analysis. *IEEE Trans. Inf. Theory*, 61(4):1702–1719, 2015.

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

[Robbins, 1952] Herbert Robbins. Some aspects of the sequential design of experiments. *Bull. Am. Math. Soc.*, 58(5):527–535, 1952.

[Sunehag *et al.*, 2018] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning based on team reward. In *Proc. Int. Conf. Auton. Agents Multiagent Syst.*, pages 2085–2087, 2018.

[Tekin and Liu, 2012] Cem Tekin and Mingyan Liu. Online learning of rested and restless bandits. *IEEE Trans. Inf. Theory*, 58(8):5588–5611, 2012.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Adv. Neural Inf. Process. Syst.*, pages 5998–6008, 2017.

[Weber and Weiss, 1990] Richard Weber and Gideon Weiss. On an index policy for restless bandits. *J. Appl. Probab.*, 27(3):637–648, 1990.

[Whittle, 1988] Peter Whittle. Restless bandits: Activity allocation in a changing world. *J. Appl. Probab.*, 25(A):287–298, 1988.

[Williams and Baird, 1993] Ronald J. Williams and Leemon C. Baird. Tight performance bounds on greedy policies based on imperfect value functions, 1993. Tech. rep. NU-CCS-93-14, College of Computer Science, Northeastern University.

Appendix

A Experimental Environment

All experiments are performed on a machine equipped with an 8-core Intel(R) Xeon(R) Gold 6151 CPU @ 3.00GHz, 64 GB RAM, and an NVIDIA Tesla V100 SXM2 32 GB GPU, in a docker container running EulerOS 4.8.5. The Transformer encoder and decoder comes from the official PyTorch implementation.

The relevant packages and their versions are as follows: Python == 3.6.4, torch == 1.4.0, scikit-learn == 0.19.1, numpy == 1.19.0, CUDA == 10.2.

B The Quick-Max Algorithm

We mentioned the sequential maximization. The pseudo code of the algorithm is presented in Algorithm 1.

Algorithm 1 The quick-max algorithm

Input: $Q_i(s_i, \cdot)$ for all $i = 1, \ldots, N$, and M
Output: $a = [a_1, \ldots, a_N]$
Initialization: counter $\leftarrow 0$

for all $i = 1 \ldots, N$ **do**
  $a_i \leftarrow \arg\max Q_i(s_i, \cdot)$
  counter $\leftarrow$ counter $+ a_i$
  if counter $= M$ **then**
    break
  end if
end for

C Additional Comments on Time complexity

The myopic policy does not involve offline computation and thus yields zero time overhead. We show the time spent for calculating the Whittle indices in Figure 4. We can see that the complexity is linear in the number of arms as the indices are computed independently for each arm. Moreover, the complexity scales up as the state increases as we need to compute the indices for each state. For comparison, the time spent in our approach does not scales up with respect to the number of states due to the readily deployable GPU parallelism.

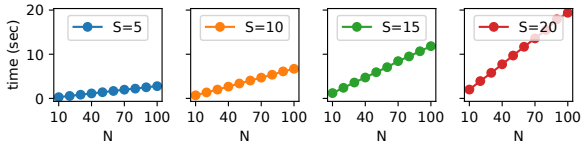

Figure 4. Time spent for calculating Whittle indices.

D Additional Performance Comparison

We additionally compare our approach with myopic and index policies for two types of arms: random shift and random shift with controlled restart.

Random shift. The active (passive) transition matrix of a random shift arm is a randomly generated lower (upper) triangular matrix. The non-zero elements in each row is a decreasing sequence.

Random shift with controlled restart. The passive transition matrix is still an upper triangular matrix, but the active transition is now a deterministic one as $P_i(s_i = 0|s_i', 1) = 1$.

The reward functions are the same as those in the paper. We use the same setup as we did for the controlled restart process in the paper to compare the performances of each policy. Table 3 shows related results. Our approach out performs the other two.

Table 3. Additional performance comparison

	random shift		
N	10	20	50
myopic	5536.3(128.7)	11343.6(215.5)	29174.8(187.8)
index	5680.4(58.5)	11790.2(97.4)	30046.8(152.7)
ours	5872.4(131.3)	12050.4(172.1)	30708.8(239.0)
	random shift with restart		
N	10	20	50
myopic	6080.6(113.5)	12551.2(126.0)	31118.4(176.9)
index	5916.0(34.7)	12460.4(57.4)	30991.2(82.9)
ours	6382.1(94.0)	13187.4(128.6)	32821.5(222.2)

E Additional Figures and Results

Figure 5 shows the transient performances of each method in the ablation study. Figure 6 shows the transient performances of each policy for a setup with $N = 100$ arms and a cardinality constraint $M = 30$.

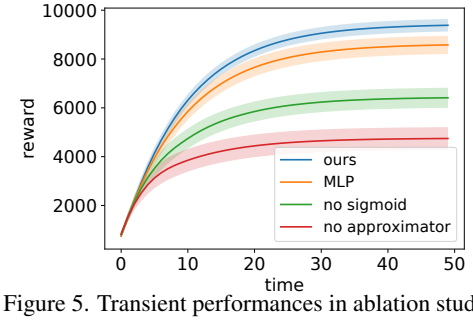

Figure 5. Transient performances in ablation study.

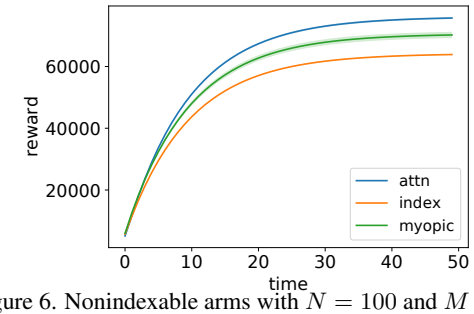

Figure 6. Nonindexable arms with $N = 100$ and $M = 30$.