Alternative Index Policy

BTP Report: Pratik Chetan Shah (200100121)

February 20, 2024

The formulation for restless multi arm bandits is:

$$max\ E\ [\lim_{m\to\infty} \frac{1}{m} \sum_{n=1}^{n=m} \sum_{i=1}^{i=N} r^i(s_n^i, a_n^i)] \tag{1}$$

$$\text{subject to: } \sum_{i=1}^{i=N} a_n^i = M, \quad M < N \tag{2}$$

The Whittle Relaxation used is:

$$E\ [\lim_{m\to\infty} \frac{1}{m} \sum_{n=1}^{n=m} \sum_{i=1}^{i=N} a_n^i] = M \tag{3}$$

The relaxed LP formulation for restless multi arm bandits for N arms out of which M can be activated at a timestep can be written as:

$$max\ E\ [\lim_{m\to\infty} \frac{1}{m} \sum_{n=1}^{n=m} \sum_{i=1}^{i=N} r^i(s_n^i, a_n^i) + \lambda^*(\sum_{i=1}^{i=N} a_n^i - M)] \tag{4}$$

We propose an alternative index policy for restless multi arm bandits based on the difference of the Q values for the state (k) in which the arm is present in i.e.

$$\delta = Q_{\lambda^*}(k, 1) - Q_{\lambda^*}(k, 0) \tag{5}$$

Where λ^* is the common subsidy for passivity learnt by gradient ascent in the dual space:

$$\lambda_{n+1}^* = \lambda_n^* + \beta(n)(\sum_{i=1}^{i=N} a_n^i - M) \tag{6}$$

The Q values are updated according to the RVI Q-Learning algorithm as:

$$Q_{n+1}(s_n, a_n) \leftarrow (1 - \alpha(n))Q_n(s_n, a_n) + \alpha(n)((1 - a_n)(r_o(s_n) + \lambda_n^*) + a_n r_1(s_n) + \max_{v\epsilon\{0,1\}} Q_n(s_{n+1}, v) - \frac{1}{2d} \sum_{k\in S} Q_n(k, 0) + Q_n(k, 1)) \tag{7}$$

By using a common subsidy for passivity the problem complexity is reduced as we don't have to maintain and update Lagrangian multiplier for every arm (which also acts as the Whittle Index) instead we just have to learn the common subsidy
The alternative indexes are learnt offline using the following algorithm:

Algorithm 1 An Alternative Index Policy for Restless Multi Arm Bandits

1: Initialize λ^*, Q(s,a) for all states and actions, ϵ=1.
2: **for** n = 1: n_{end} **do**
3: Update $\alpha(n)$ and $\beta(n)$ as: $\alpha(n) = \frac{1}{\lceil n \backslash 5000 \rceil + 1}$ $\beta(n) = \frac{1}{\lceil nlog(n) \backslash 5000 \rceil + 1}$
4: Choose action a_n^i for each arm i in an ϵ-greedy fashion
5: Update s_{n+1}^i and reward r_n^i from s_n^i and a_n^i for every arm i
6: Update (s_n^i,a_n^i) Q-values for each arm i as:
$Q_{n+1}(s_n, a_n) \leftarrow (1-\alpha(n))Q_n(s_n, a_n)+\alpha(n)((1-a_n)(r_o(s_n)+\lambda_n^*)+a_n r_1(s_n)+ \max_{v \epsilon \{0,1\}} Q_n(s_{n+1}, v)-\frac{1}{2d}\sum_{k \in S} Q_n(k,0)+ Q_n(k,1))$
7: Update common subsidy for passivity for all arms λ^* as: $\lambda_{n+1}^* = \lambda_n^* + \beta(n)(\sum_{i=1}^{i=N} a_n^i - M)$
8: Decrement ϵ as $\epsilon_{n+1} = max(0.01, \epsilon_n * 0.99)$
9: **end for**
10: Calculate the new index for each arm i where the current state of arm is represented by k:
$\delta_{n+1}(k) = Q_{\lambda^*}(k,1) - Q_{\lambda^*}(k,0)$

Once the alternative indexes are learned they can be used online to select the top M arms to be pulled according to the descending order of the indices.
We present 3 approaches to implement Algorithm 1 for finding the alternative indices:

- **A**. Using a single Q-value matrix shared across all homogenous arms of the same type
- **B**. Using a different Q-value matrix for each arm, and hence the updates in step 5 of the algorithm only change the Q-values of the corresponding arm only
- **C**. Allowing a maximum budget number of arms only to be pulled even during the training phase. This is useful in certain applications where pulling more than the budget number of arms can have adverse effects.

Approach A is the default approach as it is the most efficient one, and the performance of Approach A and B are almost always the same, but in some examples, as we will show in the numerical experiments, Approach B works better.

1 Example with Restart

Now we consider a "restart problem" with state space $S = \{0, 1, 2, 3, 4\}$ where the active action forces an arm to restart from some state. Specifically, we consider an example with 5 states, where in the passive mode (u = 0) an arm has tendency to go up the state space whereas in the active mode (u = 1) the arm restarts from state 1 with probability 1, i.e.,

$$P_0 = \begin{bmatrix} 1/10 & 9/10 & 0 & 0 & 0 \\ 1/10 & 0 & 9/10 & 0 & 0 \\ 1/10 & 0 & 0 & 9/10 & 0 \\ 1/10 & 0 & 0 & 0 & 9/10 \\ 1/10 & 0 & 0 & 0 & 9/10 \end{bmatrix} \text{ and } P_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{8}$$

The rewards in the passive mode are given by $R(k, 0) = a^k$ (in this numerical experiment, we have taken a = 0.9) and the rewards in the active mode are all zero. As in the previous example, we consider the scenario with N = 100 arms, out of which M = 20 are active at each time step. the exact values of the Whittle indices are given by: $\lambda_1 = -0.9, \lambda_2 = -0.73, \lambda_3 = -0.5, \lambda_4 = -0.26$ and $\lambda_5 = -0.01$. We initialize our algorithm with $\delta_i = 0$ and $Q(i, u) = r(i, u), \forall i \in S$

1.1 Approach A

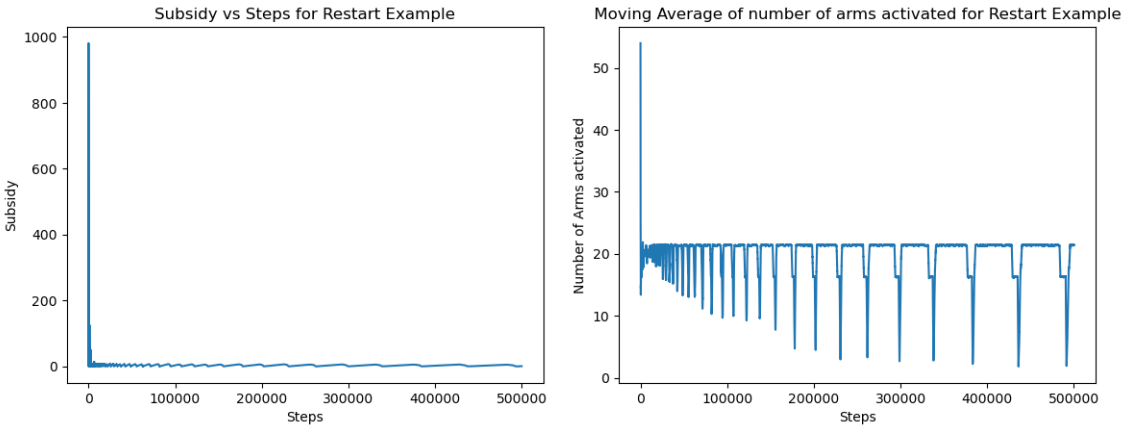


Figure 1: Above two plots are during the training phase

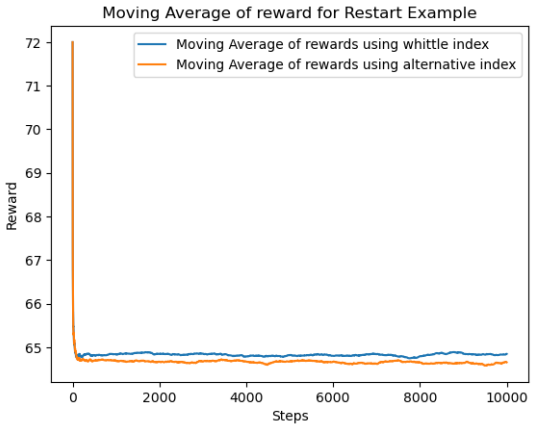


Figure 2: Above plot compares performances during online selection

1.2 Approach B

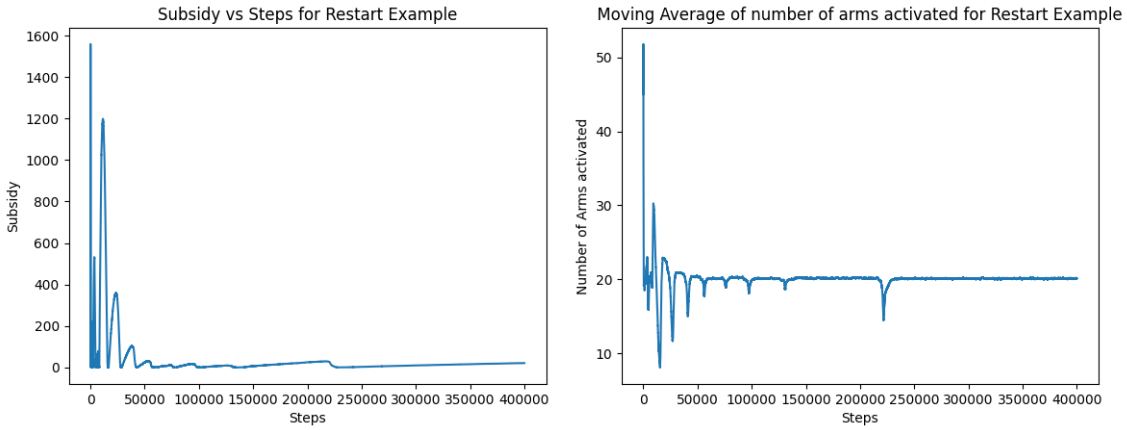


Figure 3: Above two plots are during the training phase

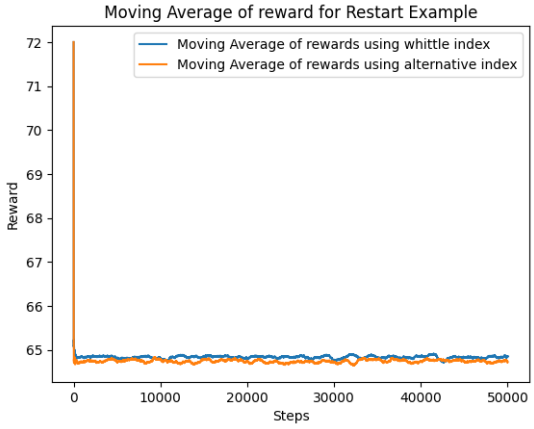
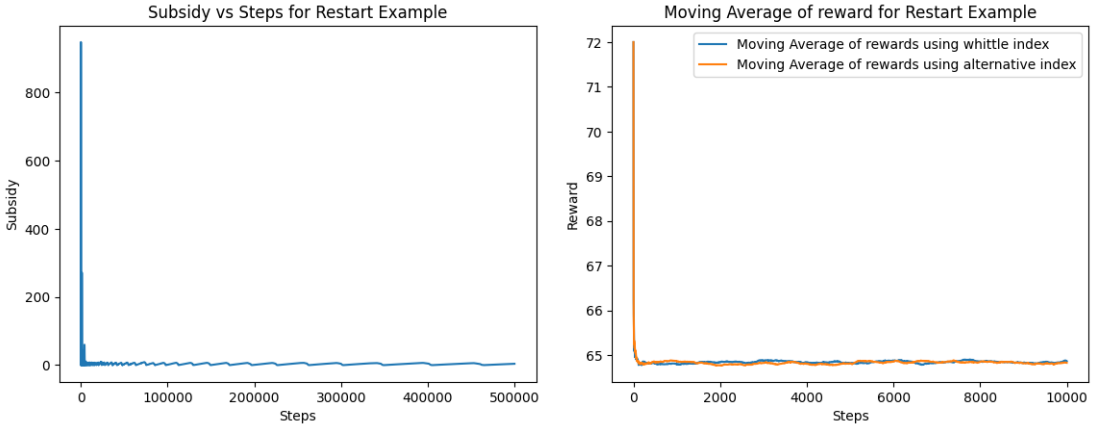


Figure 4: Above plot compares performances during online selection

1.3 Approach C



2 Heterogeneous Arms with Restart

In this example, we demonstrate the advantage of the alternative index policy. We take 5 arms as described in the second example, all with different transition probabilities. The transition probabilities are expressed as functions of a variable x, which takes different values for each arm, making them heterogeneous.

$$P_0 = \begin{bmatrix} 1-x & x & 0 & 0 & 0 \\ 1-x & 0 & x & 0 & 0 \\ 1-x & 0 & 0 & x & 0 \\ 1-x & 0 & 0 & 0 & x \\ 1-x & 0 & 0 & 0 & x \end{bmatrix} \text{ and } P_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{9}$$

here x takes values from $\{0.5, 0.6, 0.7, 0.8, 0.9\}$ corresponding to the 5 arms. From the 5 arms we want to pull 1 arms i.e. N=5 and M=1. The rewards in the passive mode are given by $R(k, 0) = x^k$, and the rewards in the active mode are all zero. In this example the Whittle indices will have to be calculated for each state of every different arm i.e., all 5 arms, whereas the common subsidy of passivity is shared across all the arms and hence needs to be learned only once, thus substantially reducing the problem complexity. In this case as all arms are different approach A and B are the same.

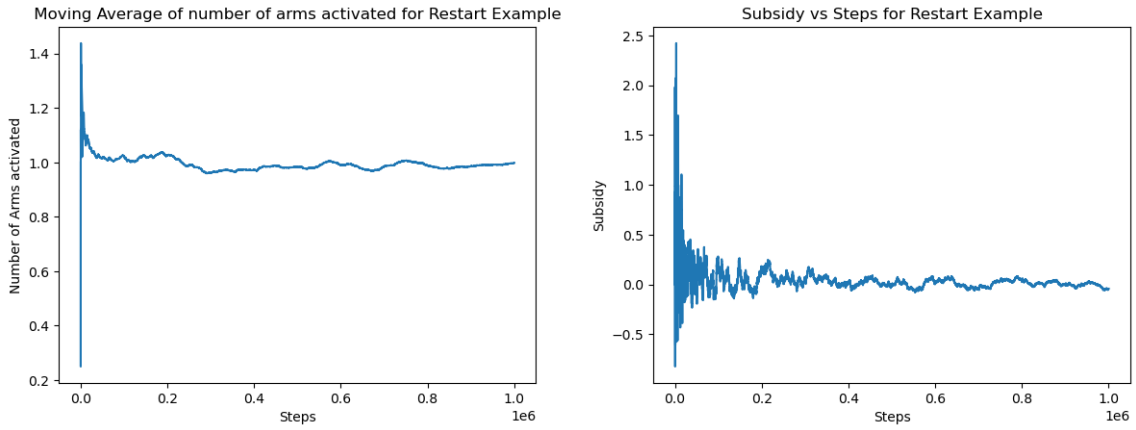


Figure 5: During training phase

We calculate the Whittle indices for comparison of performance during online selection for all the states of each arm using the Q-learning for Whittle Index (QWI) Algorithm. Below are the graphs showing the convergence of the Whittle index for all 5 arms. The Y axis is the Whittle index, and the X axis is the number of steps.

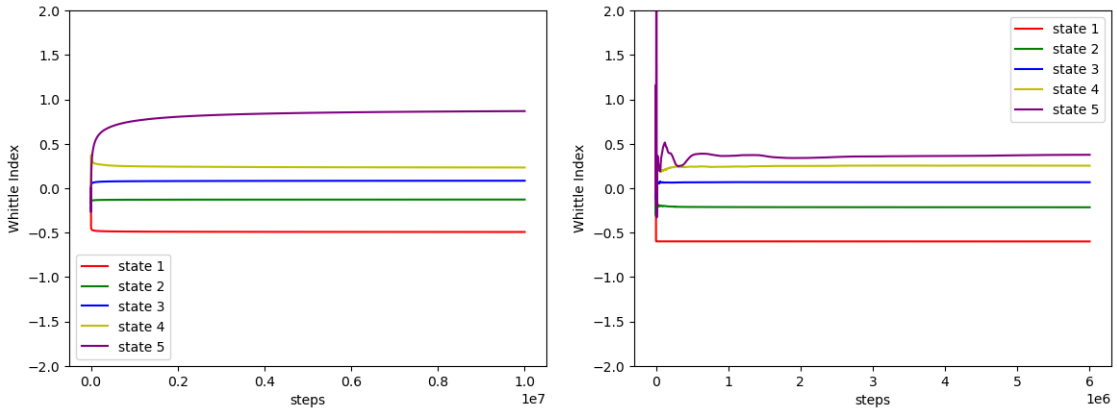


Figure 6: Above two plots are for arm 1 and arm 2

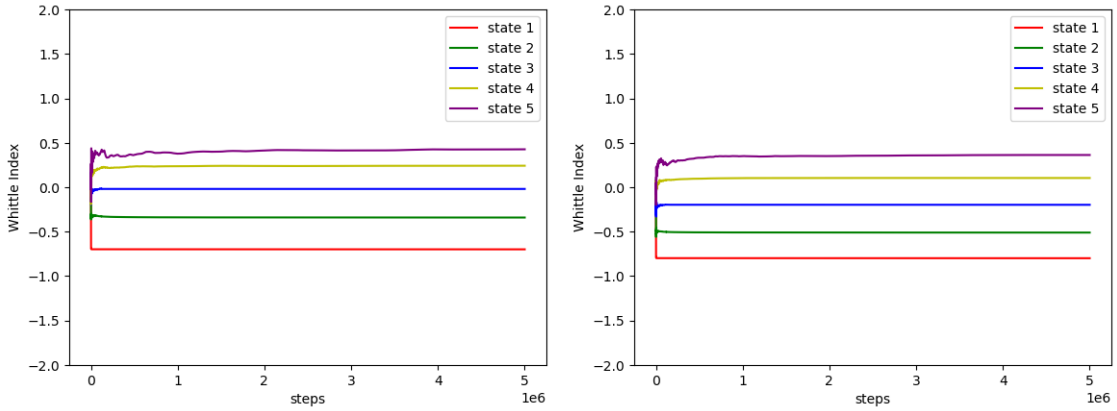


Figure 7: Above two plots are for arm 3 and arm 4

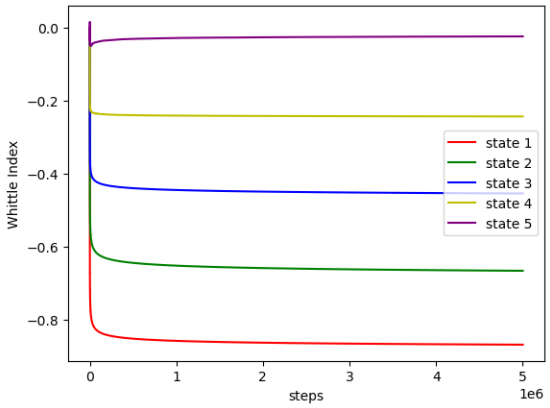


Figure 8: Above plot is for arm 5

Once the whittle indexes are calculated we compare the rewards obtained when choosing arms according to the proposed alternative index vs choosing arms according to the calculated whittle indexes.

Figure 9: Above plot compares performances during online selection

To show the advantage of the new policy, we analyzed the index calculation time for both the Whittle Index Policy and the Alternative Policy:

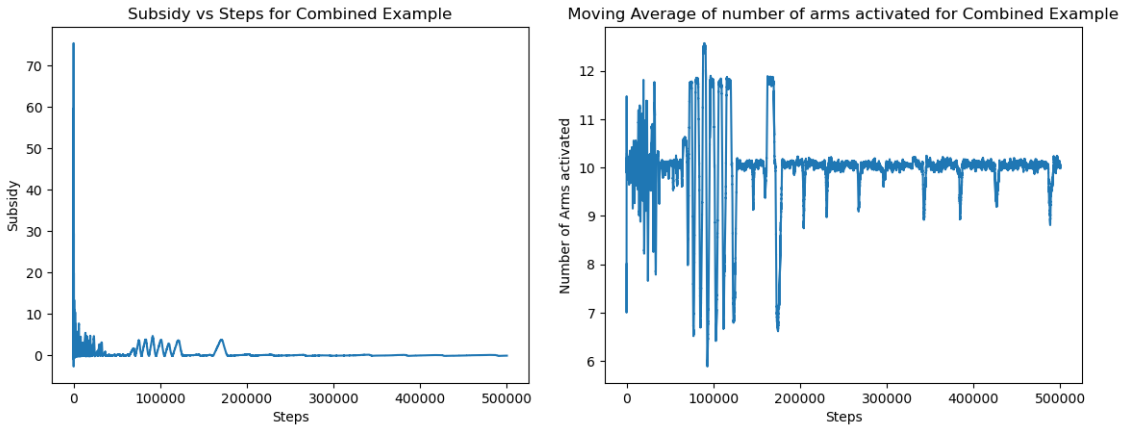
- The Time taken to calculate the Whittle index for each type of arm is around 2 minutes. Therefore, it takes around 10 minutes to calculate the Whittle indices for all the arms.
- The Time taken to calculate the new index for all arms is around 1.5 minutes.

Hence, the Alternative Policy significantly reduces the index calculation time and performs similarly to the Whittle Index Policy.

3 Multiple Heterogeneous Restart Arms

In this example, we take a set of 50 heterogeneous restart arms, as described in example 4, with a budget of 10 arms. The variable x still takes values from $\{0.5, 0.6, 0.7, 0.8, 0.9\}$, but now for each value of x we take 10 arms. For all the 50 arms, we still learn just one common subsidy of passivity, thus signifying the advantage of our approach. For comparison during online selection, we use the Whittle Indices calculated in section 4.

3.1 Approach A



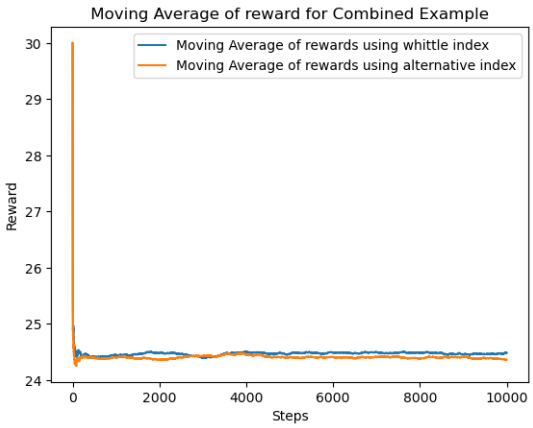
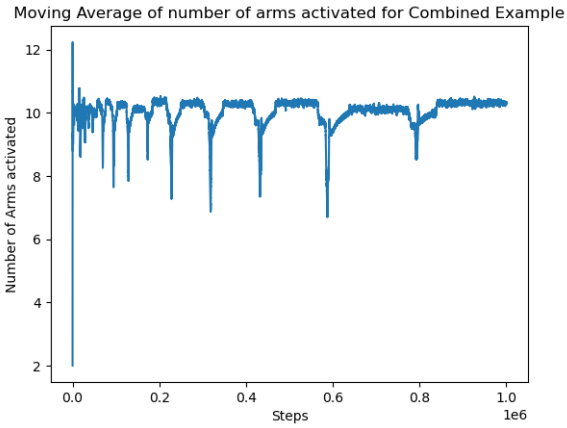
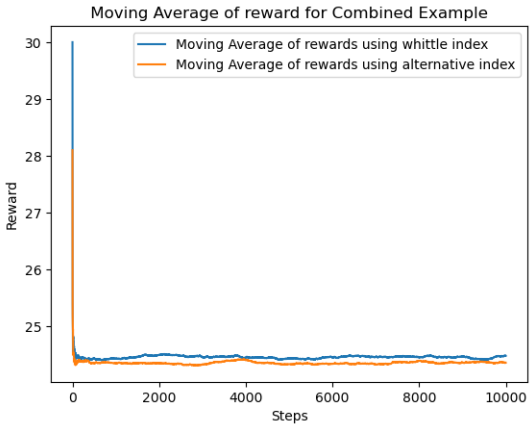


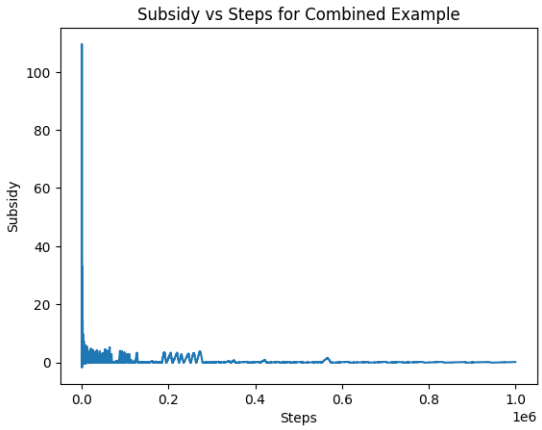
Figure 1 above shows the number of arms being pulled converges to 10 during the training phase. Figure 2 shows the comparison of rewards during online selection

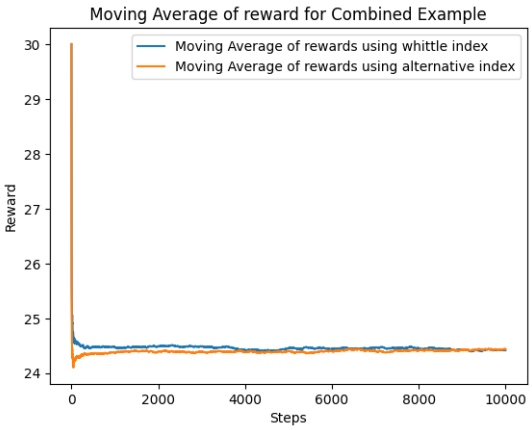
3.2 Approach B





3.3 Approach C





4 Using DQN instead of tabular Q learning for alternative indices

Till now we have used tabular Q learning for learning Q values and ultimately the alternative indices. Now, we use the Double Deep Q Learning Network (DDQN) for approximation of the Q values. This will especially be better in cases where the state space is large. A complete scheme of how our algorithm works can be found in Alg. 2.

In the case of homogeneous arms, one-dimensional state space suffices, which will be the state of the arm. In the case of heterogeneous arms, a two-dimensional state space is used. The first dimension is used to give information about the type of arm, and the second dimension specifies the state of the arm. The neural network consists of a dense network of three hidden layers of (16,32,16) neurons connected through ReLU activation function. The neural network outputs are the Q-values for both possible actions. The actions are taken in an ϵ-greedy fashion. ϵ, i.e. the probability for exploration, is initialized as 1 and then at the end of every iteration, is reduced by the exploration-decay factor. The min value of ϵ is set to be 0.01.

The equation for the Q target used in training DQN is:

$$Q_{target}(s_n, a_n) \leftarrow ((1-a_n)(r_o(s_n)+\lambda_n^*) + a_n r_1(s_n) + \max_{v\epsilon\{0,1\}} Q_n(s_{n+1}, v) - \frac{1}{2d}\sum_{k\in S} Q_n(k,0) + Q_n(k,1)) \tag{10}$$

In DDQN we employ a second neural network, called the target model, for the computation of $\max_{v\epsilon\{0,1\}} Q_n(s_{n+1}, v)$. The reason for this decision is due to the maximisation bias of Q-learning: overestimating the Q-values results in this error increase over time, due to the target being $r + \gamma \ max_a Q(s,a)$. The use of a second neural network for the target helps control this bias. This second neural network copies the parameter values of the main network, the policy model, after every iteration.

We store the $(s_n^i, a_n^i, r_n^i, s_{n+1}^i, \lambda_n^*)$ tuples in an experience replay buffer, from which a batch of random samples is taken every time to train the main neural network. For each tuple $(s_n^i, a_n^i, r_n^i, s_{n+1}^i, \lambda_n^*)$ we calculate Q target values using equation 11. Once we have calculated Q-values for all the examples in batch through $Q_\theta(.)$ and its targets $Q_{target}(.)$, we compute the loss function as the mean square error between the two and train the main neural network, through a standard Adam optimiser using a learning rate of $lr = 2.5e-4$.

The alternative indexes are learnt offline using the following algorithm:

Algorithm 2 An Alternative Index Policy for Restless Multi Arm Bandits using DDQN

Initialize λ^*, DQN policy, DQN target, Batch Size,ϵ=1.
for n = 1: n_{end} **do**
    Update $\beta(n)$ as: $\beta(n) = \frac{1}{\lceil nlog(n) \backslash 5000 \rceil + 1}$
    for arm $i \in N$ **do**
        Choose action a_n^i for each arm i in an ϵ-greedy fashion
        Update s_{n+1}^i and reward r_n^i from s_n^i and a_n^i for every arm i
        Store $(s_n^i, a_n^i, r_n^i, s_{n+1}^i, \lambda_n^*)$ in experience replay buffer
        Train DQN policy on one batch of $(s_n^i, a_n^i, r_n^i, s_{n+1}^i, \lambda_n^*)$ tuples from experience replay buffer
    end for
    Update the DQN target model
    Update common subsidy for passivity for all arms λ^* as: $\lambda_{n+1}^* = \lambda_n^* + \beta(n)(\sum_{i=1}^{i=N} a_n^i - M)$
    Decrement ϵ as $\epsilon_{n+1} = max(0.01, \epsilon_n * 0.99)$
end for
Calculate the new index for every state s of each arm i by:
    $\delta(s) = Q_{\lambda^*}(s,1) - Q_{\lambda^*}(s,0)$

We now show the performance of the DDQN in case of the "restart problem" as described in example 2. As stated earlier, as this is a homogeneous problem, we use a 1-dimensional state space. Below are the results obtained in the case of using N=10 restart arms, out of which M=2 have to be activated. For the DDQN, a batch size of 32 was used, the size of the experience replay buffer was kept as 100 tuples and the exploration decay factor used was 0.98.

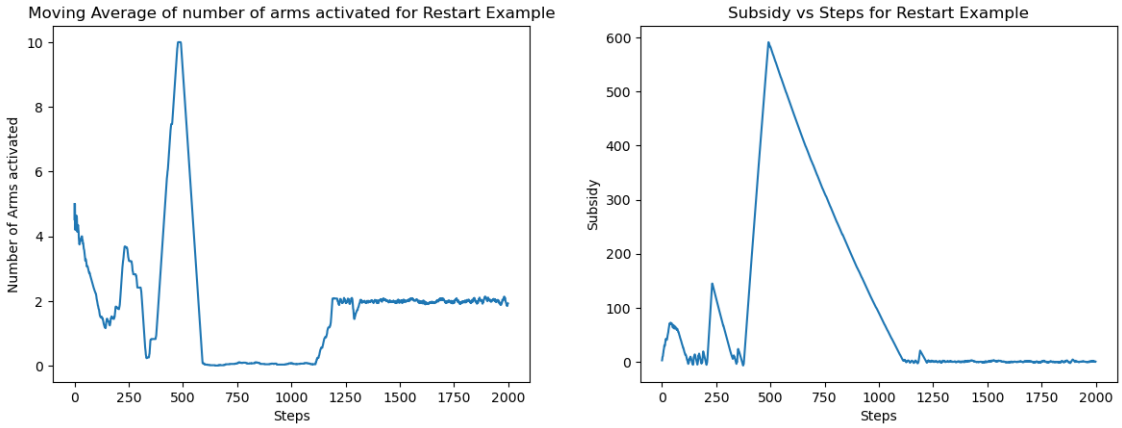


Figure 10: Above two plots are during the training phase

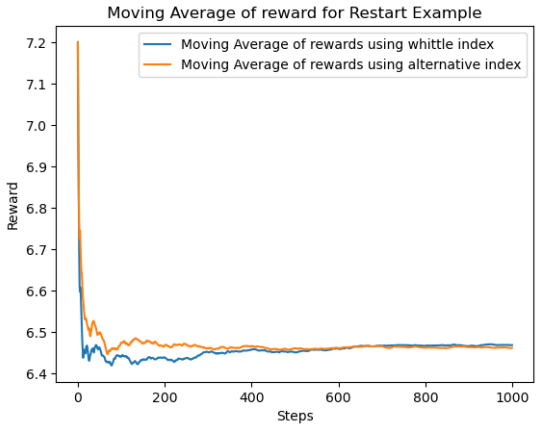


Figure 11: Above plot compares performances during online selection

As seen in the left graph of Figure 12, even when utilizing DDQN, the number of arms being activated converges to 2 during the training phase. Figure 13 shows that the new indices learned utilizing DDQN perform as well as actual Whittle Indices during online selection.

5 Using DDQN in heterogenous restart bandits case

Here, we demonstrate the use of DDQN for calculating the new indices in a hetergeneous case. As described in example 4 we take 5 heterogeneous restart arms with the variable x taking values from $\{0.5, 0.6, 0.7, 0.8, 0.9\}$, and the budget is of 1 arm. In this case, we use a two-dimensional state space. The first dimension is used to give information about the type of arm and the second dimension is the state of the arm. The first dimension used are 0,1,2,3,4 for x=0.5, x=0.6, x=0.7, x=0.8 and x=0.9 respectively. So for example, if it is the arm corresponding to x=0.5 and it is in the 2^{nd} state, the state given to DDQN will be [0,2].

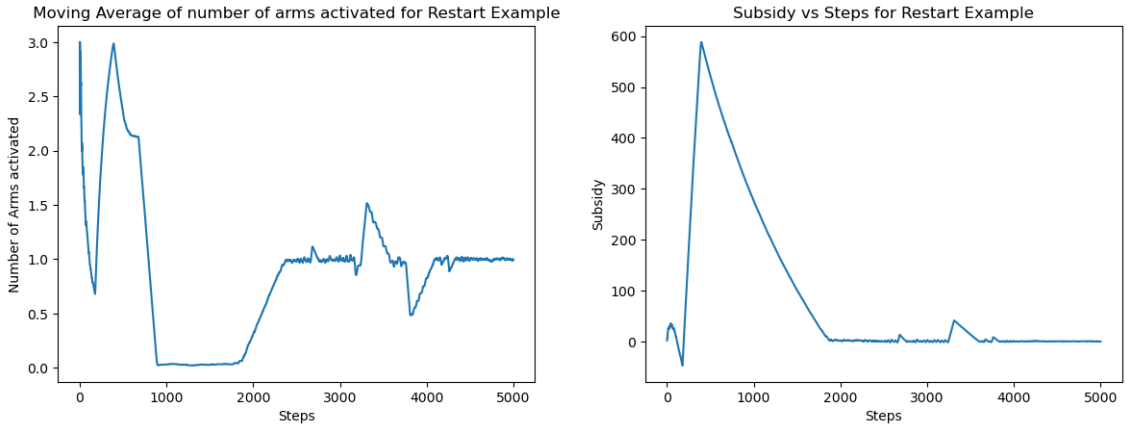


Figure 12: Above two plots are during the training phase

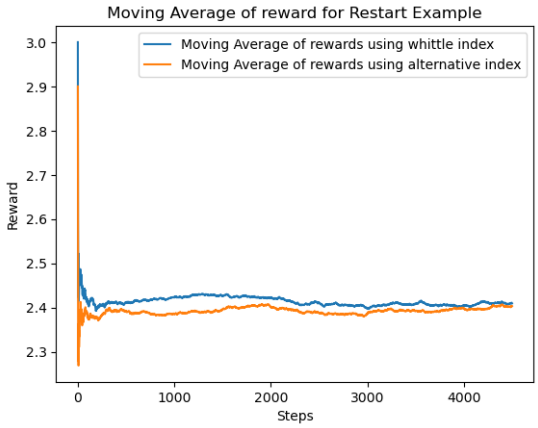


Figure 13: Above plot compares performances during online selection

6 Non Whittle Indexable Problem

Here, we check the performance of our index in a Non-Whittle Indexable Problem as described in the Appendix A.4 of this paper.
A problem is said to be Whittle Indexable if as the passivity subsidy λ increases from $-\infty$ to $+\infty$ the set of project states where it is optimal to rest the project increases monotonically from the empty set to the full project state space.
The transition probability matrices and the reward matrix for this problem are:

$$P_0 = \begin{bmatrix} 0.005 & 0.793 & 0.202 \\ 0.027 & 558 & 0.415 \\ 0.736 & 0.249 & 0.015 \end{bmatrix} \text{ and } P_1 = \begin{bmatrix} 0.718 & 0.254 & 0.028 \\ 0.347 & 0.097 & 0.556 \\ 0.015 & 0.956 & 0.029 \end{bmatrix} \tag{11}$$

$$\text{Reward Matrix: } [[0, 0.699], [0, 0.362], [0, 0.715]]$$

In many cases even when the problem is Non Whittle Indexable, Whittle Indices are used empirically. Hence even we calculate the Whittle Indices for this problem and compare the performance of our indices against them. The Whittle Indices are again calculated using the QWI algorithm.

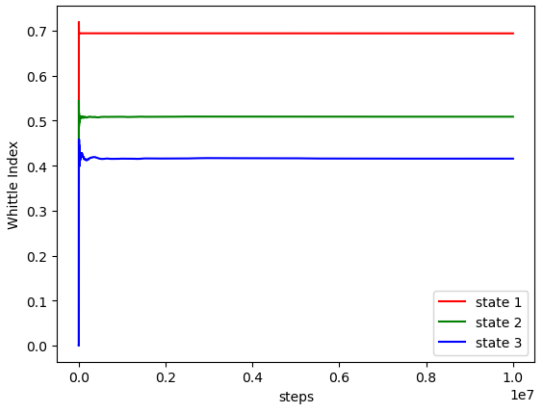
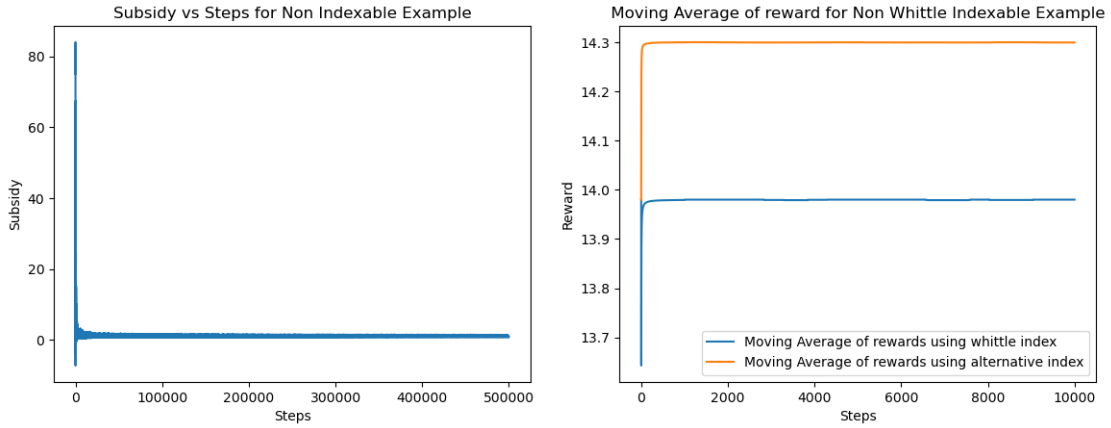


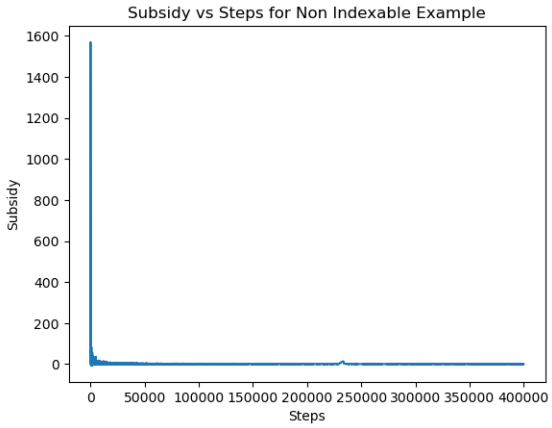
Figure 14: Whittle Indices calculated using QWI

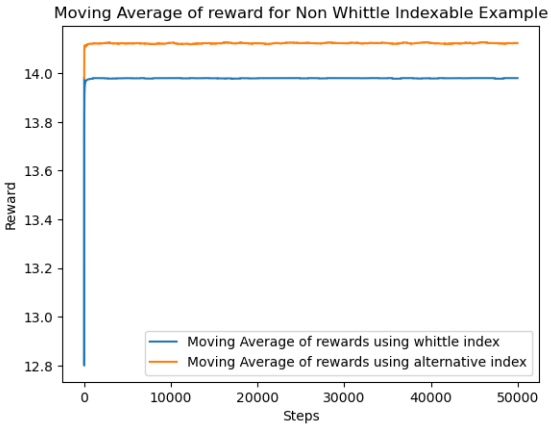
6.1 Approach A



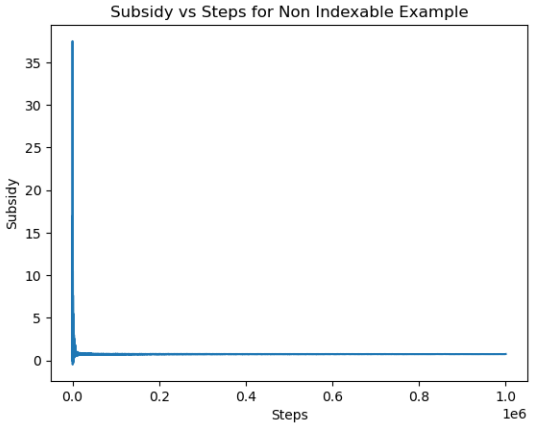
As seen in Figure (right), the new indices result in a higher reward as compared to the Whittle indices, thus showing that the new index can also be used in Non-Whittle Indexable cases as well.

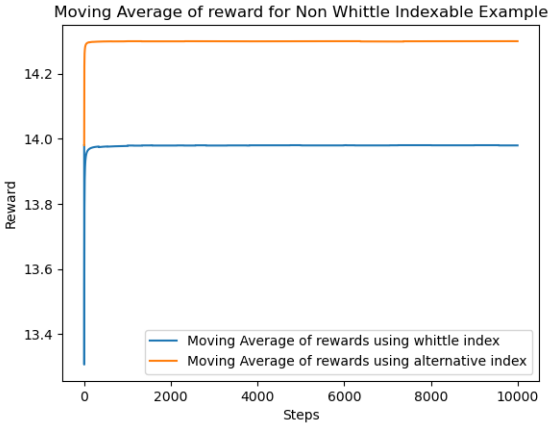
6.2 Approach B





6.3 Approach C





7 Comparison against LP Priority Policy

Here, we borrow the experiment as described in section G.2 of the Appendix of this paper. Figure 3 of the same paper shows that the LP Priority policy doesn't work well in this example.
The experiment details are as follows:
This is a homogeneous case where each arm can take one of 8 states $\{0, 1, 2, 3, 4, 5, 6, 7\}$. The budget M is set to be N/2, where N is the total number of arms. The transition probability matrices are:

$$P_0 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.48 & 0.52 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.47 & 0.53 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.9 & 0.1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.9 & 0.1 \\ 0.1 & 0 & 0 & 0 & 0 & 0 & 0 & 0.9 \end{bmatrix} \quad P_1 = \begin{bmatrix} 0.9 & 0.1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.9 & 0.1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.9 & 0.1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.9 & 0.1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.46 & 0.54 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.45 & 0.55 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.44 & 0.56 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.43 & 0.57 \end{bmatrix} \tag{12}$$

Reward Matrix: $[[0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0, 0], [0.1, 0]]$

7.1 Approach A

In this example we compare two different ways:

- The constrained way: To pull exactly N/2 arms during the online selection based on the descending order of the new indices
- The relaxed way: To pull only those arms for which the new index corresponding to the state the arm is in is positive. At max, N/2 arms can only be pulled in this case. The ordering of arms is done again according to the descending order of the new indices.

For the case N=1000, M=500:
The alternative indices we got were: {0.0079, 0.1880, 0.1024, 0.0987, -0.1087, -0.0566, -0.0506, -0.0299}
Giving the preference order as: $1 > 2 > 3 > 0 > 7 > 6 > 5 > 4$ This is the same order as mentioned in the paper
Relaxed Way:

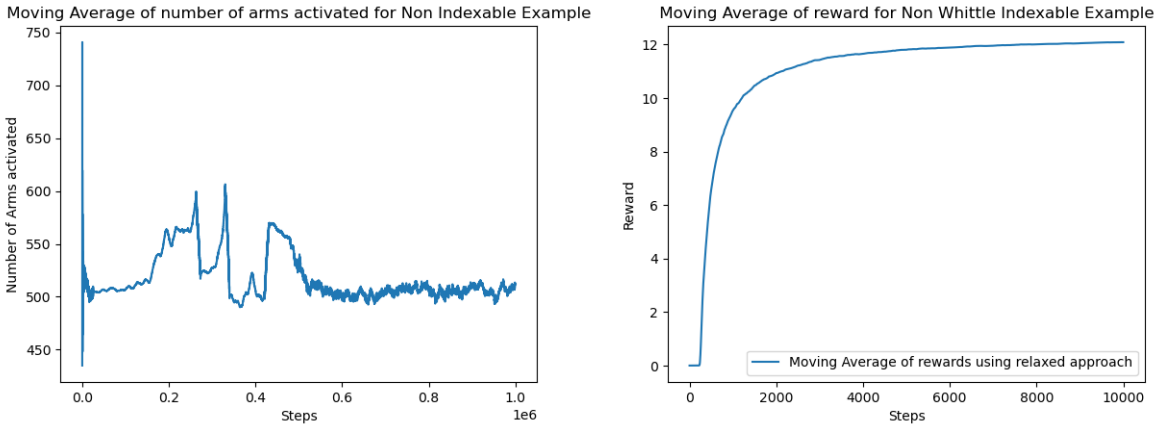


Figure 15: Moving Average of number of arms pulled and reward using relaxed approach

As seen in the graphs, the average rewards obtained using the relaxed way asymptotically tend to the optimal value of 0.012 (12/1000)
Constrained Way for N=100 M=50:

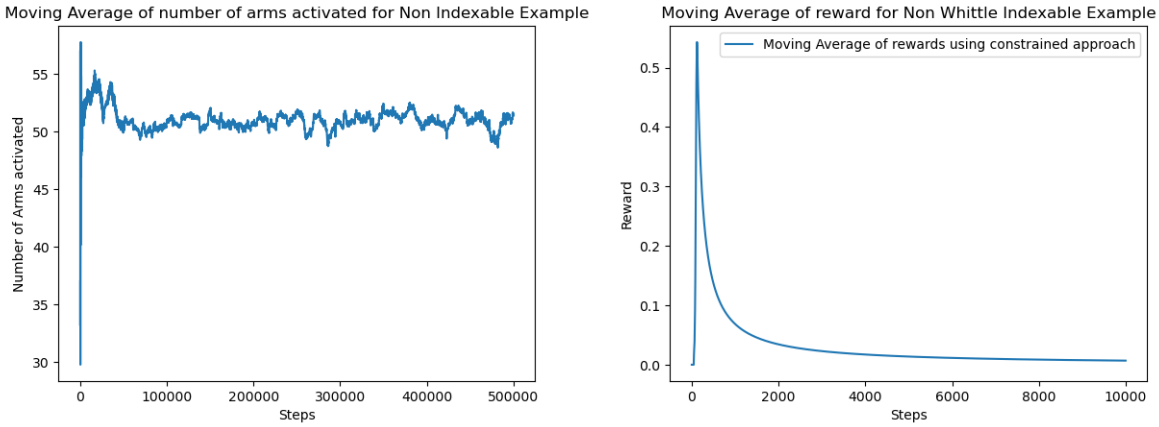


Figure 16: Moving Average of number of arms pulled and reward using relaxed approach

In the constrained way, the reward drops down to close to 0 and is in accordance with Figure 3 of the paper, which shows that the rewards in the case of LP-Priority are close to 0

7.2 Approach B

Again, we compare two different ways:

- The constrained way: To pull exactly N/2 arms during the online selection based on the descending order of the new indices
- The relaxed way: To pull only those arms for which the new index corresponding to the state the arm is in is positive. At max, N/2 arms can only be pulled in this case. The ordering of arms is done again according to the descending order of the new indices.

For the case N=50, M=25:

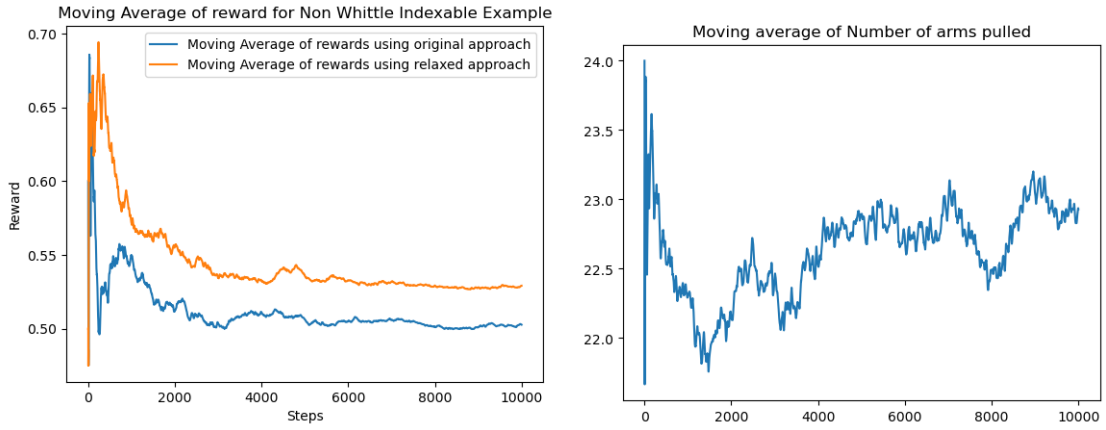


Figure 17: Left: Moving Average of number of arms pulled during online selection using relaxed approach

For the case N=200, M=100:

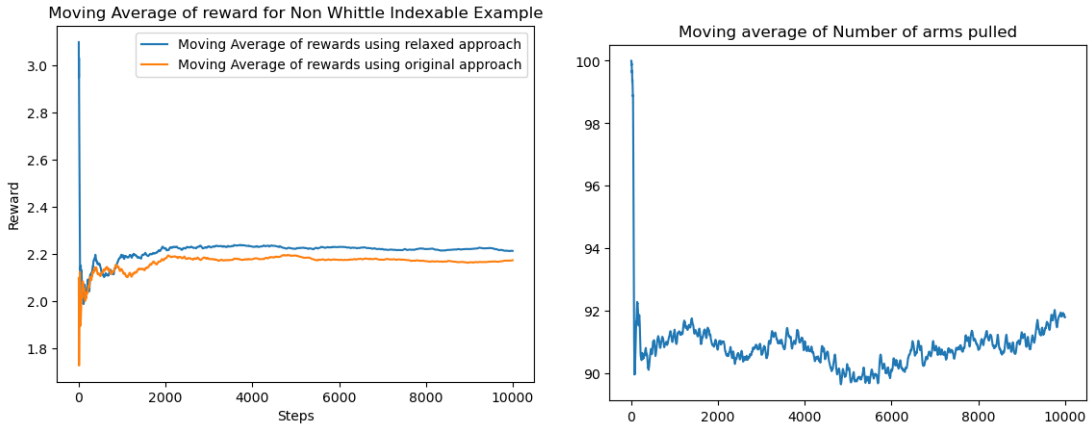


Figure 18: Left: Moving Average of number of arms pulled during online selection using relaxed approach

For the case N=500, M=250:

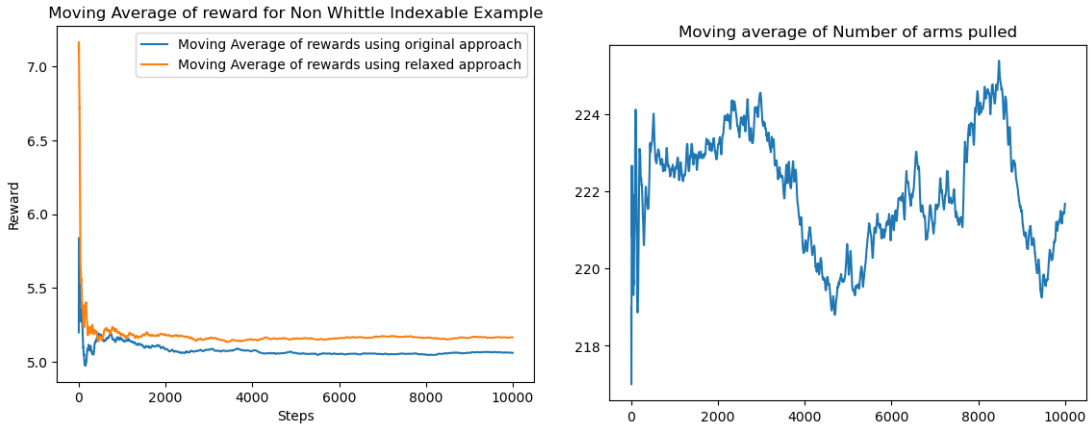


Figure 19: Left: Moving Average of number of arms pulled during online selection using relaxed approach

As seen from the graphs, the relaxed way gives better results than the constrained way in all the cases. Moreover, both ways give an average reward in the range of 0.01 to 0.011, which is much higher than the LP Priority policy, as seen in Figure 3 of the above-mentioned paper