Optimization Techniques: Final Project
An Window-Based Solution for Solving Time-Varying Stochastic Multi-Armed Bandit Problem

Sumaiya Iqbal
Computer Science, University of New Orleans, ID # 2450707
siqbal1@uno.edu

Abstract – In this project, a time-varying stochastic multi-armed bandit (MAB) problem is considered. There are 10 slot machines. One can choose only one machine to play at each time. For playing with any machine, a reward is given. The objective is to choose the most profitable arm as much as possible and play it as often as possible. There are random reward distributions associated with all the arms which are unknown to the user. The goal is to design an algorithm to learn a good strategy by playing the multi-armed bandit problem multiple times. Here, I propose a window-based strategy to estimate the time-varying reward distribution effectively as well as dynamically update the selection model.

Introduction and Literature Review
In the classical MAB problem (Gittins et al., 2011), there are K number of arms and a player can play the arms for N times. Each of the arm is associated with an unknown reward distribution. Therefore, when an arm is played, it generates rewards according to the distribution. We considered the case when the reward distribution can vary with time. The objective is to maximize the total expected reward over N times. The performance of a sequential arm selection policy is also measured by regret (or, proportional regret or, reward loss), defined as the difference in expected total reward with respect to the ideal scenario with known reward models where the player always plays the best arm. Thus, the goal is to minimize the total regret (or the percentage of reward loss) between the ideal cumulative reward (that you always choose the best arm) and the actual reward obtained from your algorithm. Two crucial challenges are involved in this problem. Firstly, this is a classic problem to study the trade-off between exploitation (intensifying the currently available knowledge) and exploration (diversify to find new potential knowledge) while doing the optimization. Secondly, this problem includes designing a sequential decision strategy to select the best arm which should be effectively adaptable to reflect the time-variance of the reward distributions.

A rich amount of studies on MAB problem are available in the literature (Mahajan and Teneketzis, 2008, Kuleshov and Precup, 2014, Auer and Ortner, 2010, Garivier and Cappé, 2011, Audibert and Bubeck, 2010). An extensive empirical study that compares multiple algorithms for MAB problem is conducted by (Vermorel and Mohri, 2005). The available algorithms include epsilon-greedy, Boltzmann exploration (Softmax), pursuit and reinforcement comparison. The other two available algorithms are Upper Confidence Bounds (UCB) and UCB-I (tuned). Each of
these methods has its own heuristic to handle the exploitation/exploration balance. These algorithms are popular for solving MAB problem with fixed distribution of reward sequences.

However, I am motivated to solve the MAB problem instance that involves time-varying reward distributions. Therefore, I avoided the details involved in the existing algorithms and focused in proposing a new strategy to solve this problem. The main contributions of this work includes: i) extensive analysis of the time-varying reward sequence ii) step-by-step development of with three new strategies for expected reward calculation to be utilized for sequential arm selection, iii) result and performance analysis and iv) proposal of future research.

**Problem Formulation**

Let, there are $K$ number of arms. Each arm is associated with a reward distribution, $\mathcal{R}_i^{(N)} = \{r_{i(1)}, r_{i(2)}, \ldots, r_{i(N)}\}$, of length $N$. Therefore, we have a set of $K$ number of reward distributions, $\mathcal{R}^N = \{\mathcal{R}_1^N, \mathcal{R}_2^N, \ldots, \mathcal{R}_K^N\}$. At each time (trial or, turn), $t = 1, \ldots, N$, player has to select an arm, $i, i = 1, \ldots, K$. We denote the ideal reward as $r_{(i)}^{\text{ideal}}$ and actual (observed) reward as $r_{(i)}^{\text{actual}}$. In the best case, $i = \text{best arm}$. The objective is to minimize the cumulative proportional regret over all the trials computed by the following Equation 1.

$$\text{proportional regret} = \sum_{t=1}^{N} \frac{r_{(i)}^{\text{ideal}} - r_{(i)}^{\text{actual}}}{r_{(i)}^{\text{ideal}}} \quad \text{--------- (1)}$$

**Analysis of Sequential Rewards**

To understand the time-varying reward sequences well, I carried out two types of data analysis:

1. Reward distribution of the arms with mean ($\mu$) and variance ($\sigma^2$)
2. Sequential rewards per trial to spotlight the time-variant trend of the data

For this, I generated set of reward distribution, $\mathcal{R}^N = \{\mathcal{R}_1^N, \mathcal{R}_2^N, \ldots, \mathcal{R}_K^N\}$ with $K = 10, N = 1000$. TABLE I shows the output. From the distributions, we can see that arm 7 and 9 has the high expected rewards, $\mu_7 = 3.495723$ and $\mu_9 = 3.480257$, respectively. However, arm 9 ($\sigma_9^2 = 8.495855$) have higher variance compared to that of arm 7 ($\sigma_7^2 = 3.414135$). It is also clear from the reward sequence with time (right column) that arm 9’s reward distribution has an increasing trend with high rate with the time. Arm 7 has an increasing trend initially, however gives approximately consistent reward in higher number of trials. On the other hand, arm 9 has a very fluctuating reward sequence and catches up an increasing trend in later trials.

This analysis gives an intuition of the trade-off requirement between exploitation and exploration. Initially, arm 7 is effective. However, the algorithm will fail if it is not adaptable with time and keeps on intensifying the reward sequence of arm 7. Therefore, the design strategy should incorporate appropriate diversification strategy so that it can eventually detect arm 9 as an effective arm.
TABLE I: Reward distribution analysis. Left column shows the distribution with mean and variance and the right column shows the reward sequence time.

<table>
<thead>
<tr>
<th></th>
<th>Mean</th>
<th>Variance</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st Arm</td>
<td>Mean = 0.809039, Variance = 0.360424</td>
<td></td>
</tr>
<tr>
<td>2nd Arm</td>
<td>Mean = 0.673459, Variance = 0.034729</td>
<td></td>
</tr>
<tr>
<td>3rd Arm</td>
<td>Mean = 0.945802, Variance = 0.97211</td>
<td></td>
</tr>
</tbody>
</table>
4th Arm: Mean = 0.49500, Variance = 0.250225

5th Arm: Mean = 0.763748, Variance = 0.227256

6th Arm: Mean = 1.212256, Variance = 0.121911
7th Arm: Mean = 3.495723, Variance = 3.414135

8th Arm: Mean = 0.50600, Variance = 0.250214

9th Arm: Mean = 3.480257, Variance = 8.495855
Key Ideas and Sample Algorithm (simplified $\epsilon$-greedy)

The simplified $\epsilon$-greedy method is a widely used approach for sequential decision problems. With a fixed $\epsilon$ value, the algorithm selects the arm with the highest observed expected reward with probability $1 - \epsilon$. This is the greedy heuristic for exploitation, whereas the exploration strategy includes selecting a random arm with probability $\epsilon$. We are provided with a sample implementation of this algorithm in which $\epsilon$ decreases in a logarithmic fashion with time. This simulates the ETE (Exploit Then Explore) policy in which the algorithm highly fluctuates its decision of choosing arm in the initial stages. Later, it keeps on exploiting the best available knowledge. Following are the highlights about the given simplified $\epsilon$-greedy algorithm:

- **Epsilon strategy**: log decreasing
- **Selection strategy**: maximum (expected estimated reward), a greedy exploitation strategy
- **Expected reward estimation strategy**:
  - *Cumulative Moving Average on the Full data (CMFA)*
  - Let, arm $i$ has been selected at trial $t$, which is the $n^{th}$ selection of this arm. Then the expected mean reward for this arm is estimated as *(Equation 2)*,

\[
\mathbb{E}[r_{i(t)}^n] = \frac{(n - 1) \times \mathbb{E}[r_{i(t)}^{n-1}] + r_{i(t)}}{n}
\]  

- **Exploration strategy**: random selection of an arm

The greedy selection strategy, *maximum (expected estimated reward)*, of this design spotlights that we need to effectively estimate the expected reward of the arms to achieve high reward values. Based on the method of computing the expected reward (*Cumulative Moving Average on the Full data (CMFA)*), I call this simplified algorithm as “*simplified – $\epsilon$-greedy-CMAF*”. To analyze the performance, I ran simplified-$\epsilon$-greedy-CMAF algorithm on a reward distribution set with K...
7 = 10 and N = 1000. Figure 1 shows the algorithm’s selection performance by plotting the ideal and observed (or, actual) number of selection of each of the arm.

**Figure 1**: Comparison of Ideal and Actual Selection of Arms. The x-axis presents the number (index) of arms and the y-axis shows the total number of ideal selection (blue) and actual (red) selection, respectively.

Figure 1 proves that arm 9, 7 and 10 are the 1\textsuperscript{st}, 2\textsuperscript{nd} and 3\textsuperscript{rd} most profitable arms, respectively. The most important information extracted from this figure is that the simplified-\(\varepsilon\)-greedy-CMAF design highly over selects arm 7 as well highly under selects arm 9. To develop my algorithm, I wanted to overcome this weaknesses. Therefore, I further investigated the result with three illustrations focusing on arm 7 and 9.

1. **Ideal reward sequences** with time of arm 7 and 9 (Figure 2)
2. **Moving estimated expected reward** with time for arm 7 and 9 (Figure 3)
3. **Moving number of selection** with time for arm 7 and 9 (Figure 4)

Firstly, Figure 2 shows the ideal reward sequence over time. Secondly, I plotted the expected observed reward estimated by Equation 2 at each selection over time in Figure 3. I denote it as moving expected reward with time as we update it at each selection. With the greedy selection strategy, maximum (expected estimated reward), the expected reward should be computed effectively to reflect the ideal reward sequence. However, the comparative study of Figure 2 and Figure 3 shows that current full data based strategy of computing expected reward in the simplified-\(\varepsilon\)-greedy-CMAF could not reflect the time variance of ideal reward sequence well. The
key finding from this analysis is ‘to correctly estimate time-varying data, we should rely on the recent useful subset of the full data’.

**Ideal reward sequence of the arms over time**

![Ideal reward sequence of the arms over time](image)

**Figure 2**: Ideal reward sequences of arm 7 (blue) and 9 (red) over time. The x-axis and y-axis show the time (trial number) and ideal reward of the arms at that trial, respectively.

Thirdly, I focused on the trend of total selection of the arm with time in **Figure 4**. The comparative study of **Figure 3** and **Figure 4** shows that after some time the expected reward of arm 7 gets flat. However, the simplified-ε-greedy-CMAF continues to select arm 7 as the total number of selection of this arm still continues to increase. This highlights another weakness of simplified-ε-greedy-CMAF. In later trials, arm 9 becomes the most profitable arm, however this simplified algorithm cannot reflect this change with time and continues to intensify the currently available best information from arm 7. The key idea found from this analysis is that, I have ‘to track the recent trend of the data’, ‘to flush the irrelevant historical data while calculating the expected reward’ and ‘to guide the exploration step towards data with high variance’.
Moving expected observed rewards of the arms over time by simplified-\(\epsilon\)-greedy-CMAF

Figure 3: Moving expected observed reward sequences of arm 7 (blue) and 9 (red) over time. The x-axis and y-axis show the time (trial number) and expected observed reward of the arms, respectively.

Moving total number of selection of the arms over time by simplified-\(\epsilon\)-greedy-CMAF

Figure 4: Moving number of selection of arm 7 (blue) and 9 (red) over time. The x-axis and y-axis show the time (trial number) and moving total number of selection of the arms, respectively.
Development of Proposed Algorithms and Corresponding Analysis

In this section, I have shown the step-by-step development process of my proposed algorithms and corresponding improvement of the results from simplified-\(\epsilon\)-greedy-CMAF. Note that, I repeated each experiment for 100 run/cycle \((C)\) to show the best and average performance. Here, I have used one sample reward set in all the cycles in the development process of the algorithm. Therefore, it only reflect the improvement in the result, not original performance. The original performances and comparisons are shown in the following section. From the previous section, we saw that with the greedy selection strategy, \textit{maximum (expected estimated reward)}, and the expected reward should be computed effectively to reflect the ideal reward sequence. Therefore, redesigned the expected reward computation strategy.

\textit{Proposed Expected Reward Calculation Strategy - I}

In the estimation of time-varying data (reward), I propose to utilize recent subset of data instead of the full data. It involves defining a window size, \(w\). I calculated the expected observed reward by \textit{Simple Moving Average on the Windowed data (SMWA)} and call the corresponding algorithm as \textit{\(\epsilon\)-greedy-SMAW}. Moreover, I kept track of the last selection time of each of the arm \((lst_i)\). If an arm is selected long before, more than \(w\) amount of trials, I clear the irrelevant historical observed reward of that arm and update the expected observed reward as the current observed reward. If the current number of selection of an arm is less than or, equal to the \(w\) amount of time, use all the observed rewards within the window to calculate the simple average as the expected reward. On the other hand, take only the \(w\) amount of data to compute the simple average as the expected reward if the current number of selection of an arm is greater than \(w\). Let, arm \(i\) has been selected at trial \(t\) and this is the \(n^{th}\) selection of this arm. Then the expected mean reward for this arm is updated by the \textbf{Equation 3} given below:

\[
E[r_{i(t)}^n] = \begin{cases} 
    r_{i(t)}, & \text{if } (t - lst_i) \geq w \\
    \text{mean}(r_i[1, \ldots, n]), & \text{if } n \leq w \\
    \text{mean}(r_i[(n-w+1), \ldots, n]), & \text{if } n > w 
\end{cases} \tag{3}
\]

Now, to compare the performance with simplified-\(\epsilon\)-greedy-CMAF algorithm, I repeated the calculation of ideal and observed (or, actual) number of selection of each arm on the same reward distribution set as \textbf{Figure 1} for \textit{\(\epsilon\)-greedy-SMAW} algorithm (\textbf{Figure 5}). The comparative study of \textbf{Figure 1} and \textbf{Figure 5} show that the new algorithm, \textit{\(\epsilon\)-greedy-SMAW}, could solve the problem of over selection and under selection of arm 7 and 9, respectively. However, it is also visible that a gap still remains between the ideal and actual utilization of these arms.
Ideal and actual selection of arms by $\epsilon$-greedy-SMAW

**Figure 5:** Comparison of Ideal and Actual Selection of Arms by $\epsilon$-greedy-SMAW. The x-axis presents the number (index) of arms and the y-axis shows the total number of ideal selection (blue) and actual (red) selection, respectively.

**Proposed Expected Reward Calculation Strategy – II**

A more advanced way of handling and estimating the time-varying data is to apply unequal weights for each datum value in the subset to emphasize particular values in the subset. In the proposed strategy – I, I assigned identical weight (equal to 1) to all the observed reward values within the specified window. Here, I propose an improvement which assigns different weights which corresponds to the triangle series values, $\{w, w-1, ..., 2, 1\}$, as the data goes far away from the current time. This expected reward calculation is **Weighted Moving Average on the Windowed data (WMWA)** and name the corresponding algorithm as **$\epsilon$-greedy-WMAW**. Now, let arm $i$ has been selected at trial $t$ and this is the $n^{th}$ selection of this arm. Then the expected mean reward for this arm is updated by the **Equation 4** given below:

$$
\mathbb{E}[r_{i(t)}^n] = \begin{cases} 
    r_{i(t)}, & \text{if } (t - lst_i) \geq w \\
    \frac{(r_i[1] \times 1) + \cdots + (r_i[(n-1)] \times (w-1)) + (r_i[n] \times w)}{1 + \cdots + (w - 1) + w}, & \text{if } n \leq w \\
    \frac{(r_i[(n-w+1)] \times 1) + \cdots + (r_i[(n-1)] \times (w-1)) + (r_i[n] \times w)}{1 + \cdots + (w - 1) + w}, & \text{if } n > w
\end{cases}
$$

(4)
Figure 6 shows the similar analysis of Figure 1 and Figure 5 for \( \epsilon \)-greedy-WMAW algorithm. This figure shows that the weighted computation of expected reward with the windowed data can further reduce the gap of ideal and actual selection of the arms.

![Ideal and actual selection of arms by \( \epsilon \)-greedy-WMAW](image)

**Figure 6**: Comparison of Ideal and Actual Selection of Arms by \( \epsilon \)-greedy-WMAW. The x-axis presents the number (index) of arms and the y-axis shows the total number of ideal selection (blue) and actual (red) selection, respectively.

**Proposed Expected Reward Calculation Strategy - III**

In the proposed strategy – II, I used the observed reward sequence in a window \( r[n - w + 1, ..., w] \) to calculated expected reward by WMAW. Finally, I utilized the ideal rewards, consecutive with \( t \) \( r[t - w + 1, ..., w] \), within the window to compute the expected reward. This expected reward calculation is called as Weighted Ideal Moving Average on the Windowed data (WIMWA) and name the corresponding algorithm is denoted as \( \epsilon \)-greedy-WIMWA. Now, let arm \( i \) has been selected at trial \( t \) and this is the \( n^{th} \) selection of this arm. Then the expected mean reward for this arm is updated by the Equation 5 given below:

\[
\mathbb{E}[r_i^n(t)] = \begin{cases} 
  r_{i(t)}, & \text{if } (t - lst_i) \geq w \\
  \frac{(r_i[1] \times 1) + \cdots + (r_i[(t - 1)] \times (w - 1)) + (r_i[t] \times w)}{1 + \cdots + (w - 1) + w}, & \text{if } t \leq w \\
  \frac{(r_i[(t - w + 1)] \times 1) + \cdots + (r_i[(t - 1)] \times (w - 1)) + (r_i[t] \times w)}{1 + \cdots + (w - 1) + w}, & \text{if } t > w
\end{cases}
\]  

---

\( \mathbb{E} \) \( r_i^n(t) \)

---

12
Figure 7 shows the similar analysis of Figure 1, Figure 5 and Figure 6 for $\epsilon$-greedy-WIMAW algorithm. The comparative study shows that the weighted computation of expected reward with the ideal windowed reward values made the ideal and actual selection of the arms almost similar.

**Ideal and actual selection of arms by $\epsilon$-greedy-WIMAW**

![Ideal and actual selection of arms by $\epsilon$-greedy-WIMAW](image)

*Figure 7: Comparison of Ideal and Actual Selection of Arms by $\epsilon$-greedy-WIMAW.* The x-axis presents the number (index) of arms and the y-axis shows the total number of ideal selection (blue) and actual (red) selection, respectively.

Moreover, I traduced a guided exploration technique in algorithm. Selecting a random arm in the exploration stages of the simplified algorithm involves the risk of losing high reward just by random chance. To reduce this risk, I determined the arm with highest variance in the observed reward sequence at each exploration. I checked the observed reward of a random arm ($\text{candidate}_{arm} - 1$) and the expected observed reward of the arm with highest variance ($\text{candidate}_{arm} - 2$). The final best arm ($\text{best}_{arm}$) is decided by the higher reward value. I name this exploration as ‘guided exploration’ which combines the randomness. We performed preliminary investigation of the usefulness of this guided exploration. Result shows that guided exploration provides additional (6 – 8)% improvement in minimizing proportional regret over only random selection.

Following are the highlights about the new algorithm proposed in this work:

- **Epsilon strategy:** log decreasing
- **Expected reward estimation strategy:**
  - Weighted Ideal Moving Average on the Windowed data (WIMWA) – *Equation 5*
- **Arm Selection strategy:**
- **Exploitation**: maximum (expected estimated reward), a greedy strategy
- **Exploration**: guided exploration, a combined strategy of randomness and existing knowledge

The following Figure 8 and Figure 9 shows the moving expected reward and moving number of selection with time for the newly proposed algorithm (similar analysis as Figure 3 and Figure 4).

**Moving expected observed rewards of the arms over time by $\epsilon$-greedy-WIMAW**

![Graph](image)

*Figure 8. Moving expected observed reward sequences of arm 7 (blue) and 9 (red) over time. The x-axis and y-axis show the time (trial number) and expected observed reward of the arms, respectively.*

Now, the comparative study of Figure 2 and Figure 8 shows that the new windowed data based strategy of computing expected reward could effectively reflect the time variance of ideal reward sequence. The study of Figure 8 and Figure 9 shows that the new algorithm is adaptable with the time and could detect the arm 9 as the most informative arm after some trial. Initially, arm 7 is the most profitable arm which is reflected by the total number of selection in Figure 9. Later, arm 9 becomes the informative arm (Figure 8). As with the expected reward, the total number of selection of arm 7 gets flat (Figure 9) after some trial and that of arm 9 keeps on increasing (Figure 9). Therefore, the computation of expected reward is now enough effective for the greedy selection of arm with the maximum expected reward.
Results and Discussion

This section includes the final results and performance comparisons among four algorithms (simplified-\(\epsilon\)-greedy-CMAF, \(\epsilon\)-greedy-SMAW, \(\epsilon\)-greedy-WMAW, \(\epsilon\)-greedy-WIMAW) described in the previous section. For this, I generated different reward sequences with \(K = 10\) for \(C = 100\) times. I executed the simulations with \(N = 5000, 10000\) and \(20000\). This section includes the detail results with \(N = 5000\) and \(20000\). We mentioned the results with \(N = 10000\) in the next section along with the concluding discussion. The performance is compared in terms of average proportional regret computed by Equation 1 over 100 runs. I also highlighted the minimum, maximum and standard deviation of the regrets in 100 runs along with the average time (in second) needed in each run.

**Comparison Set –I (K = 10, N = 5,000, C = 100)**

TABLE II shows that my proposed three algorithms gave 44.78\%, 45.11\% and 45.18\% lower average proportional regret than that by simplified algorithm. The final algorithm (\(\epsilon\)-greedy-WIMAW) gave the lowest proportional regret, 14.91\% in an average of 100 runs which can be extended to 9.91\% if I consider the best case. TABLE II also shows that the average time needed for each complete run (5,000 trials) is also higher for my proposed design, however was reasonable as the values are in seconds. Further, Figure 10 illustrates that my proposed strategies (in blue, green and red) perform better in minimizing regret in every run than the simplified algorithm (in black).
TABLE II. Performance of regret minimization of the algorithms.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Min regret</th>
<th>Max regret</th>
<th>Avg regret</th>
<th>Improvement Over Simplified (%)</th>
<th>Std. regret</th>
<th>Avg. time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplified-e-greedy-CMAF</td>
<td>0.1010</td>
<td>0.6147</td>
<td>0.2720</td>
<td>~</td>
<td>0.1244</td>
<td>0.2456</td>
</tr>
<tr>
<td>e-greedy-SMAW</td>
<td>0.1022</td>
<td>0.2359</td>
<td>0.1502</td>
<td>44.78%</td>
<td>0.0381</td>
<td>0.5170</td>
</tr>
<tr>
<td>e-greedy-WMAW</td>
<td>0.0970</td>
<td>0.2517</td>
<td>0.1493</td>
<td>45.11%</td>
<td>0.0376</td>
<td>0.4867</td>
</tr>
<tr>
<td>e-greedy-WIMAW</td>
<td>0.0991</td>
<td>0.2454</td>
<td><strong>0.1491</strong></td>
<td><strong>45.18%</strong></td>
<td>0.0372</td>
<td>0.4855</td>
</tr>
</tbody>
</table>

Best result is indicated by bold.

Figure 10. Comparison of proportional regret in each of the cycle (run) among four algorithms.

TABLE III shows that after N = 5,000 trials, arm – 9 provides the highest expected reward both ideally (TABLE I) and by the four algorithms while arm 7 and 10 gives the 2nd and 3rd highest expected reward. And, Figure 1 shows that my algorithms can eventually select the most profitable arm in long run. The number of time my final proposed design (dark blue bar) could select the most profitable arm (9) is very close to the ideal number of selection (light blue bar). Moreover, it is far better than that of the simplified algorithm (orange/brown bar). In terms of probability, my final design could select the most profitable arm **99.7471%** of the time.
### TABLE III. Expected reward for the arms after N=5,000 trials

<table>
<thead>
<tr>
<th>Arm</th>
<th>Simplified-e-greedy-CMAF</th>
<th>e-greedy-SMAW</th>
<th>e-greedy-WMAW</th>
<th>e-greedy-WIMAW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arm – 1</td>
<td>0.794963</td>
<td>0.589324</td>
<td>0.595438</td>
<td>1.023878</td>
</tr>
<tr>
<td>Arm – 2</td>
<td>0.670753</td>
<td>0.733162</td>
<td>0.74145</td>
<td>0.69068</td>
</tr>
<tr>
<td>Arm – 3</td>
<td>1.008241</td>
<td>1.502276</td>
<td>1.258264</td>
<td>1.396177</td>
</tr>
<tr>
<td>Arm – 4</td>
<td>0.501314</td>
<td>0.998</td>
<td>1</td>
<td>0.999333</td>
</tr>
<tr>
<td>Arm – 5</td>
<td>0.747045</td>
<td>0.96993</td>
<td>0.931862</td>
<td>1.042994</td>
</tr>
<tr>
<td>Arm – 6</td>
<td>1.203161</td>
<td>1.225691</td>
<td>1.223982</td>
<td>1.268285</td>
</tr>
<tr>
<td>Arm – 7</td>
<td>3.932991</td>
<td>4.20953</td>
<td>4.6186</td>
<td>4.161025</td>
</tr>
<tr>
<td>Arm – 8</td>
<td>0.502418</td>
<td>0.99</td>
<td>1</td>
<td>0.976</td>
</tr>
<tr>
<td>Arm – 10</td>
<td>1.813949</td>
<td>2.823386</td>
<td>2.397115</td>
<td>2.413465</td>
</tr>
</tbody>
</table>

Arm with highest expected reward is indicated by bold.

---

**Figure 11.** Comparison of total number of selection of the arms ideally and by the four algorithms (N = 5,000).
Comparison Set I (K = 10, N = 20,000, C = 100)

TABLE IV: Performance of regret minimization of the algorithms. shows that my proposed three algorithms gave 42.89%, 42.59% and 43.23% lower average proportional regret than that by simplified algorithm. The final algorithm (e-greedy-WMAW) gave the lowest proportional regret, 11.05% in an average of 100 runs which can be extended to 8.72% if I consider the best case. TABLE IV: Performance of regret minimization of the algorithms. also shows that the average time needed for each complete run (20,000 trials) is also higher for my proposed design, however was reasonable as the values are in seconds. Further, Figure 12 illustrates that my proposed strategies (in blue, green and red) perform better in minimizing regret in every run than the simplified algorithm (in black).

TABLE IV: Performance of regret minimization of the algorithms.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Min regret</th>
<th>Max regret</th>
<th>Avg regret</th>
<th>Improvement Over Simplified (%)</th>
<th>Std. regret</th>
<th>Avg. time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplified-e-greedy-CMAF</td>
<td>0.0915</td>
<td>0.6820</td>
<td>0.1947</td>
<td>~</td>
<td>0.1318</td>
<td>2.8250</td>
</tr>
<tr>
<td>e-greedy-SMAW</td>
<td>0.0881</td>
<td>0.1808</td>
<td>0.1112</td>
<td>42.89%</td>
<td>0.0220</td>
<td>5.0950</td>
</tr>
<tr>
<td>e-greedy-WMAW</td>
<td>0.0903</td>
<td>0.1801</td>
<td>0.1119</td>
<td>42.59%</td>
<td>0.0226</td>
<td>4.9554</td>
</tr>
<tr>
<td>e-greedy-WIMAW</td>
<td>0.0872</td>
<td>0.1776</td>
<td><strong>0.1105</strong></td>
<td><strong>43.23%</strong></td>
<td>0.0222</td>
<td>4.9516</td>
</tr>
</tbody>
</table>

Best result is indicated by bold.

Figure 12. Comparison of proportional regret in each of the cycle (run) among four algorithms.
**TABLE V** shows that after N = 20,000 trials, arm – 9 provides the highest expected reward both ideally (**TABLE I**) and by the four algorithms while arm 7 and 10 gives the 2nd and 3rd highest expected reward. And, **Figure 13** shows that my algorithms can eventually select the most profitable arm in long run. The number of time my final proposed design (**dark blue bar**) could select the most profitable arm (9) is very close to the ideal number of selection (**light blue bar**). Moreover, it is far better than that of the simplified algorithm (**orange/brown bar**). In terms of probability, my final design could select the most profitable arm **97.08%** of the time.

**TABLE V**: Expected reward for the arms after N=20,000 trials

<table>
<thead>
<tr>
<th>Arm</th>
<th>Simplified-e-greedy-CMAF</th>
<th>e-greedy-SMAW</th>
<th>e-greedy-WMAW</th>
<th>e-greedy-WIMAW</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arm – 1</td>
<td>0.796649</td>
<td>0.705417</td>
<td>0.599675</td>
<td>0.94003</td>
</tr>
<tr>
<td>Arm – 2</td>
<td>0.675036</td>
<td>0.721165</td>
<td>0.725649</td>
<td>0.725342</td>
</tr>
<tr>
<td>Arm – 3</td>
<td>0.997231</td>
<td>1.467152</td>
<td>1.479099</td>
<td>1.468616</td>
</tr>
<tr>
<td>Arm – 4</td>
<td>0.502626</td>
<td>1</td>
<td>1</td>
<td>0.997333</td>
</tr>
<tr>
<td>Arm – 5</td>
<td>0.751944</td>
<td>0.980132</td>
<td>1.035234</td>
<td>0.970642</td>
</tr>
<tr>
<td>Arm – 6</td>
<td>1.202057</td>
<td>1.28896</td>
<td>1.30673</td>
<td>1.31099</td>
</tr>
<tr>
<td>Arm – 7</td>
<td>4.290809</td>
<td>5.309482</td>
<td>5.300439</td>
<td>5.529246</td>
</tr>
<tr>
<td>Arm – 8</td>
<td>0.50154</td>
<td>1</td>
<td>0.995333</td>
<td>0.988</td>
</tr>
<tr>
<td><strong>Arm – 9</strong></td>
<td><strong>18.85527</strong></td>
<td><strong>25.43285</strong></td>
<td><strong>25.31975</strong></td>
<td><strong>25.40591</strong></td>
</tr>
<tr>
<td>Arm – 10</td>
<td>2.187159</td>
<td>2.776826</td>
<td>2.908255</td>
<td>2.896709</td>
</tr>
</tbody>
</table>

Arm with highest expected reward is indicated by bold.
Discussion, Future Work and Conclusion
This section draws the conclusion of this work with brief overall discussion.

Case Study
As a part of the discussion, we would like to shows the result of a case study on a real-life application of stochastic multi-arm bandit problem. Assume that, you initially have $100 and selecting an arm will cost you $1.00 each time and you will get the reward from the arm that you selected. Set N to be very large and simulate your arm selection strategy until either you have used all your money or you have reached more than $200 for the first time. Under this case study, I ran the four algorithms 100 times and documented the minimum, maximum and average number of steps and the time (in second) taken by the algorithms to reach $200. I did not face any situation where the algorithm would have stopped due to lack of money. Shows that the proposed design took on an average 111 steps to reach the target $200. However, it took 68 steps in best case. For a fare comparison of this performance, I computed the ideal case which is the number of trails needed to reach $200 is always the ideal best arm is played. Ideally, approximately 63 steps are required to achieve $200.
TABLE VI: Performance of the algorithms for the case-study.

<table>
<thead>
<tr>
<th>Algorithms</th>
<th>Steps (min)</th>
<th>Steps (max)</th>
<th>Steps (avg.)</th>
<th>Time (avg)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplified-e-greedy-CMAF</td>
<td>59</td>
<td>307</td>
<td>116.7100</td>
<td>0.0944</td>
</tr>
<tr>
<td>e-greedy-SMAW</td>
<td>57</td>
<td>208</td>
<td>112.8500</td>
<td>0.1084</td>
</tr>
<tr>
<td>e-greedy-WMAW</td>
<td>59</td>
<td>238</td>
<td>118.9100</td>
<td>0.1120</td>
</tr>
<tr>
<td>e-greedy-WIMAW</td>
<td>68</td>
<td>201</td>
<td>111.3100</td>
<td>0.1072</td>
</tr>
</tbody>
</table>

**Parameter Tuning**

The proposed design of the algorithm involves only one parameter, namely window size $w$, that defines the length of the useful subset of data to be utilized in expected reward calculation. I carried out preliminary experiments with window size 3 to 100. The result shows that the window size of ~ (4 – 7) give better results than that of the higher window size. Later, I made the window size adjustable to ‘reduce the estimated variance’. The steps for this experiment are following.

For this at every 100th trial, *(step 1)* take the most profitable arm up to current time as representative arm, *(step 2)* take the ideal (or, estimated) reward sequence of this arm up to this time, *(step 3)* check all possible window sizes: (2 to (reward sequence length-1)) and *(step 4)* select w value that gives lowest estimated variance. The corresponding source code is commented within the submission materials. The motivation behind using ‘lowest estimated variance’ as the statistic for selecting window size is to detect the ‘most recent and useful trend’ of the time variant data.

However, according to my simulations, the adjustable window size does not offer very impressive results if I compare with deterministic window size. Importantly, the adjustment requires high computational time. For a similar setup (K = 10, N = 10,000, C = 100), while the deterministic window size ($w = 5$) resulted average proportional regret of 12.47%, the adjustable window size resulted 13.33% of the average proportional regret. In the same time, the deterministic window size took 1.479 second/run and the adjustable one took 14.3239 second/run (~870% more time). Therefore, I reported all the results with the tuned deterministic value of window size 5. It was also interesting to see that even when I applied adjustable window size, in later runs eventually the optimum window size that reduces the estimated variance in expected reward was ~ (3 – 6).

**Future Possible Improvement**

One possible scope of improvement over the proposed algorithm is to modify the strategy of adjusting epsilon sequentially with time. In this work, the epsilon is decreased in a logarithmic fashion with time which allows higher amount of exploration initially, higher exploitation in later trials. It would be interesting to investigate some other strategy of adjusting the window size effectively.
**Concluding Highlights of the Proposed Algorithm**

I propose a new idea to solve time-varying stochastic multi-arm bandit (MAB) problem where the unknown reward distribution of each arm can change arbitrarily over time. It involves only one parameter to be tuned. The idea of using windowed useful and recent subset of data to estimate future rewards effectively captured the time-variant reward sequence. The design also involves flushing out of irrelevant historical data. Using variable weights (triangle series, \(\frac{w(w+1)}{2}\)) on the reward values within the window makes the design more suitable to the real time scenario. All of the proposed algorithms took higher computational time, however reasonable (approximately 50 – 60% in sec) than that of simplified epsilon-greedy. The proposed designs offer approximately (44 – 46)% improvement in terms of average proportional regret over the simplified eps-greedy. The proposed design could select the most profitable arm for about approximately (97 – 99)% of the time. The proposed design resulted approximately (12 – 14)% of average proportional regret for (5,000 - 10,000) trials which can be extended to approximately (8 – 9)% if we consider the best performance.

**Acknowledgement**
The author would like to thank the course supervisor, Dr. Huimin Chen for his suggestions and guidelines.

**Reference**


