Step-by-Step Framework: MAB Dwell-Time Optimization Using the Sense-Bandits Architecture

Team 15: RL Team

November 2025

Contents

1. Phase 1: Offline Model Construction

The offline phase constructs a structured representation of the environment prior to deployment. The system operates with eleven channels, ten discrete dwell-time actions, and a fixed per-epoch time budget.

1.1 Experience Data Collection

A simulation framework generates diverse interference conditions such as microwave noise, Bluetooth activity, and Wi-Fi congestion. For each scenario, extensive trials are executed in which a dwell-time action is sampled, the corresponding sensing fingerprint is obtained, and the resulting interference measurement is recorded. Each trial yields a tuple (s, a, r), and the complete dataset forms the basis of the offline model.

NOTE: In this document, we assume that the epoch duration is 1000 ms,sensing phase is 10ms per channel, the number of actions is 10, and the number of dwelling channels is 11.

Algorithm 1 Experience Data Collection

```
for each interference scenario do
    for trial = 1 to N do
        Sample action a uniformly from action set.
        Obtain sensing fingerprint s.
        Measure reward r.
        Store tuple (s, a, r).
    end for
end for
```

1.2 State Clustering

All fingerprints in the dataset are clustered using the K-Means algorithm with Euclidean distance. This produces K representative centroids and a deterministic assignment of each fingerprint to its nearest centroid.

Algorithm 2 State Clustering

```
Extract all fingerprints {s_i}.
Apply K-Means to obtain clusters {C_k} and centroids {μ_k}.
Compute assignment map A : s_i ↦ k.
Output {μ_k} and A.
```

1.3 Cluster Radius Determination

For each cluster, the Euclidean distance between its centroid and all assigned fingerprints is computed. The minimal distance is defined as the cluster radius, establishing the spatial

validity region of each cluster.

Algorithm 3 Cluster Radius Computation

for each cluster k **do**
  $R_k \leftarrow \min_{s_i \in C_k} \|s_i - \mu_k\|_2$
end for
Output radii $\{R_k\}$.

1.4 Cluster-Based Utility Estimation

A reward accumulator is maintained for each pair (k, a), where k is a cluster and a a dwell-time action. After inserting all dataset rewards into their respective accumulators, the arithmetic mean is computed, forming the $K \times 10$ offline utility table.

Algorithm 4 Utility Table Construction

Initialize accumulators $\text{Acc}_{k,a}$ for all (k, a).
for each tuple (s, a, r) **do**
  $k \leftarrow A(s)$ {cluster assignment}
  Append r to $\text{Acc}_{k,a}$.
end for
for each (k, a) **do**
  $\overline{\mu}_k(k, a) \leftarrow \text{mean}(\text{Acc}_{k,a})$
end for
Output utility table $\overline{\mu}_k$.

1.5 Model Packaging

The centroids, radii, and utility table are serialized into model files. These constitute the complete offline model required for real-time operation.

2. Phase 2: Online Learning and Real-Time Allocation

The online phase performs continuous adaptation under real-time constraints. Each epoch lasts 1000 ms and follows a structured sequence of sensing, model updating, optimization, execution, and storage.

2.1 Agent Initialization

Each of the eleven channels is associated with an agent instance. An initial fingerprint is sensed, classified into the nearest cluster, and the corresponding statistical parameters are loaded. Fine-tuning matrices are initialized for online regression.

Algorithm 5 Agent Initialization

for each channel c **do**
  Sense initial fingerprint s.
  $k \leftarrow \text{FindNearestCluster}(s)$.
  Load centroid μ_k, radius R_k, and utilities $\overline{\mu}_k(k, \cdot)$.
  Initialize $b_{\text{regression}} = 0$, $X_{\text{regression}} = I$.
end for

2.2 Epoch Structure

Each epoch consists of five main steps described below.

2.2.1 Sensing Phase

A rapid scan of all channels is performed, generating a fresh vector of fingerprints that characterizes the current environmental state.

Algorithm 6 Sensing Phase

for each channel c **do**
  Obtain new fingerprint $s_t^{(c)}$.
end for

2.2.2 Estimation and Model Update

Each agent evaluates whether the environment has changed relative to its previously assigned cluster. If a change is detected, the agent resets its fine-tuning model and reassigns itself. Otherwise, it updates its online regression variables. Finally, action scores are computed.

Algorithm 7 Agent Update and Score Computation

for each agent c **do**
  Retrieve previous fingerprint s_{t-1}, action vector x_{t-1}, and reward r_{t-1}.
  Compute $d \leftarrow \|s_{t-1} - \mu_{k_c}\|_2$.
  if $d > R_{k_c}$ **then**
    $k_c \leftarrow \text{FindNearestCluster}(s_{t-1})$.
    Reset $b_{\text{regression}} \leftarrow 0$, $X_{\text{regression}} \leftarrow I$.
    Load $\overline{\mu}_k(k_c, \cdot)$.
  else
    $b_{\text{regression}} \leftarrow b_{\text{regression}} + x_{t-1} \cdot r_{t-1}$.
    $X_{\text{regression}} \leftarrow X_{\text{regression}} + x_{t-1} x_{t-1}^{\top}$.
  end if
  $\tilde{\mu}_{t,e} \leftarrow X_{\text{regression}}^{-1} b_{\text{regression}}$.
  for each action a **do**
    $\text{Score}(a) \leftarrow \overline{\mu}_k(k_c, a) + \tilde{\mu}_{t,e}(a) + CB_{t,c}$.
  end for
end for
Output score matrix $S \in \mathbb{R}^{11 \times 10}$.

2.2.3 Dwell-Time Allocation

Given the remaining execution budget, a knapsack-like optimization problem is solved to determine the dwell-time assignment for all channels.

Algorithm 8 Knapsack-Based Allocation

Define budget B (e.g.,epoch budget-channels*10ms=890 ms).
Solve:

$$\max_{\{a_c\}} \sum_{c=1}^{11} S(c, a_c) \quad \text{s.t.} \quad \sum_{c=1}^{11} \text{Duration}(a_c) \leq B.$$

Output selected actions $\{a_1, \ldots, a_{11}\}$.

2.2.4 Execution Phase

The selected dwell times are executed sequentially, and the corresponding interference measurements are collected.

2.2.5 Transition Storage

All fingerprints, chosen actions, and observed rewards are stored to serve as the previous-step data for the next epoch.

2.3 Epoch Continuation

After completion of storage, the system returns to the sensing step, and the next epoch begins. This cycle repeats indefinitely, enabling continuous adaptation.