High-Speed Sensing of the Electromagnetic Environment for Cognitive Radio Receivers

Presenters: Matt Bajor, Ron Li
Outline

• Introduction
• Background and Motivation
  ▪ High-Level System Diagram
• Problem Solution
• Electromagnetic Environment Aware (EMEA) Sensor
  ▪ Standalone Performance
• Hardware Diagram
  ▪ Modifications to the FPGA
  ▪ Compressed Sensing Implementation in GNU Radio
  ▪ MLBDE Input BER Cost Function Circuit
• Machine Learning-Based Decision Engine (MLBDE) Overview
• Multi-Armed Bandit (MAB) ML Problem Formulation
• MLBDE Proof-of-Concept Validation Results
• References
Introduction

• The EM environment is cluttered in multiple domains (frequency, time, angle, etc.) making reception with an outstation increasingly difficult.

• Current detection methods for finding available receiver whitespace do not scale well in terms of speed and energy consumption.

• We present a receiver architecture that can be used for sensing an emitter’s spectral location in a fraction of the time and energy as the current state of the art.
Problem 1: The more data available, the harder it is to rapidly exploit (sensing bottleneck).

Problem 2: Optimal decision situational dependent (decision bottleneck).

Strategy to Mitigate Bottlenecks: Combine Compressed (CS) Sensing with Machine Learning (ML)
High Level System Diagram

EMEA SENSOR
CS-Based Indication of Interferers or Disturbances
Receiver Link QoS, Channel Model, etc

ITRN
ML-BASED DECISION ENGINE

EMEA SENSOR
CS-DSP

Signals at Antenna
Preselector Filter Applied

MODEM

TX
RX

CS-DSP

Link performance feedback

Instructions to receiver (adjust LNA, use filter, switch channel, change modulation etc)

Spectral Location of Interferer or EM disturbances

ITRN (SHADED AREA)
Proposed Solution (and Assumptions about the EM Environment):

- **Compressed Sensing (CS):** By assuming sparsity (that there are much less jammers than possible signal locations \(K<<N\)), compressed sensing can be used to drastically reduce the time it takes to perform a spectrum scan. It is also much more scalable than current sensing methods.

- **Reinforcement Learning:** Using a ML based decision engine, we can make adaptive countermeasure decisions much more effectively than a lookup table (LUT) based approach. It is also more scalable than a complicated logic circuit or LUT.
Electromagnetic Environment Aware (EMEA) Sensor

- A fully custom, CS enabled RF-ASIC with a single HW branch is used, called the DSIC (Direct Space to Information Converter) [4].
- The DSIC uses LO modulation to sense the frequency spectrum on 1 or 8 antennas.

- All spectrum sensing methodologies are ultimately limited by the Nyquist rate e.g. $N$ measurements for $N$ possible signal locations.
- By exploiting sparsity in the spectrum, CS can be used to take as few as $m$ measurements where $m \ll N$ [5].

$$m = K C_o \log(N/K)$$
Standalone Performance of the EMEA Sensor

Each Point for K=1:
- 10 samples per experiment
- 1000 experiments
- 9 measurements (1 PN seq. → 9 by branch expansion)
  OMP Iterations = 2

Each Point for K=3:
- 10 samples per experiment
- 1000 experiments
- 19 measurements (1 PN seq. → 19 by branch expansion)
  OMP Iterations = 4
Hardware Diagram
FPGA Modification Goals

- Embed sync input from GPIO port into IQ data to use as a time reference
- Reformat IQ data vector to include both channels' IQ data and sync signal in 32 bits to increase throughput to GNU Radio application
• Adapted from UHD 3.15 official release
  ▪ Most changes were made in noc_block_ddc.v
• Channel 0 and 1 DDCs (noc_block_ddc) will share baseband IQ samples and repackage the IQ data and sync.
Decimation, FIFO and Data Packager are new components

- **Decimation**
  - Decimates sync signal at the same rate as the DDC and buffers the decimated sync with a FIFO to preserve time alignment with IQ data

- **FIFO**
  - Buffers baseband IQ data to be sent to the other channel's noc_block_ddc instantiation
  - Data is read out of the FIFO when the other channel's data is valid, ensuring the packaged data is valid on both channels

- **Data Packager**
  - Packages IQ data from both channels and the sync into a single 32 bit vector
• Digital branch expansion is used in GNU Radio to create \( m \) virtual branches
• Each virtual branch corresponds to a CS measurement in the frequency domain: \( m = KC_o \log(N/K) \)
• Orthogonal Matching Pursuit (OMP) is used to recover the supports (e.g., Signals) -no signal reconstruction is required
MLBDE Input BER Cost Function Circuit
The goal of the ITRN system’s Machine Learning-Based Decision Engine (MLBDE) module is to compute in real-time an optimal recommended frequency bin that is both robust and high-performing.

The MLBDE module processes spectrum sensing data streamed to it in real-time over a UDP socket interface that contains the set of frequency bins that are interfered (and estimates of channel BER values used to “score” the MLBDE recommended frequency bin).

NOTE: The MLBDE module’s ML logic is aware only of the BER value for the frequency bin it has recommended (and only after the bin has been selected). This BER value is needed to quantify the reward of its recommended frequency bin (and to update its reward scores).

NOTE: The interim average reward scores are outputted only for performance evaluation purposes to quantify the benefits of ML versus a random policy for channel selection.

MLBDE Software Architecture:
Multi-Armed Bandit (MAB) ML Problem Formulation

- The MLBDE casts the ITRN frequency bin selection challenge as an instance of the contextual multi-armed bandit (MAB) problem.
- The reward obtained by the action \( A_t \) of selecting frequency bin \( A_t = k \in \{1,2,\ldots,K\} \) at each step \( t \) is the complement of BER:
  - i.e. \( R_t(A_t = k) = 1 - q_t(k) \) where \( q_t(k) \) is the BER for frequency bin \( k \) over the interval associated with time step \( t \)
- The MLBDE performance metric evaluated over \( N \) steps is the average step-by-step reward (\( \bar{R}_N \)):
  - \( \bar{R}_N = \frac{1}{N} \sum_{t=1}^{N} R_t = \frac{1}{N} \sum_{t=1}^{N} (1 - q_t(A_t)) \)
  - The average reward yielded when selecting frequency bin \( k \):
    - \( \bar{R}_N(k) = \frac{\sum_{t=1}^{N} R_t \cdot 1_{A_t=k}}{\sum_{t=1}^{N} 1_{A_t=k}} \)
- Defining \( S_t \) as the set of frequency bins experiencing interference and \( S_N(k) \) as the fraction of steps (out of the \( N \) steps \( \{1,2,\ldots,N\} \)) for which frequency bin \( k \) was interfered, the ML decision logic selects a high-performing bin \( (A_{N+1}) \) according to \( \bar{R}_N(k) \) while avoiding bins in \( S_t \):
  - \( A_{N+1} = \max_{k \in \{1,2,\ldots,K\}, \{k\} \cap S_{N+1} = \emptyset} \bar{R}_N(k) - S_N(k) \)

- MAB policies / heuristics implemented as part of the MLBDE policy suite to process the channel information and output a recommended frequency bin:

<table>
<thead>
<tr>
<th>Policy / Heuristic</th>
<th>Description of Policy/Heuristic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Random</td>
<td>Periodically select a random frequency irrespective of the interfered frequency set or its past selections</td>
</tr>
<tr>
<td>Sticky Non-Interfered</td>
<td>A new frequency is not explored unless the current frequency is under interference</td>
</tr>
<tr>
<td>Random Non-Interfered</td>
<td>Periodically select a random frequency as long as the frequency is not under interference</td>
</tr>
<tr>
<td>( \varepsilon )-Greedy</td>
<td>Perform random exploration with probability ( \varepsilon ) (exploration) but use best frequency o/w (exploitation)</td>
</tr>
<tr>
<td>( \varepsilon )-First</td>
<td>Perform pure exploration for first ( \varepsilon N ) trials and then pure greedy exploitation for remaining ((1 - \varepsilon)N ) trials</td>
</tr>
<tr>
<td>( \varepsilon )-Decreasing</td>
<td>Similar to ( \varepsilon )-Greedy, but uses a decreasing ( \varepsilon ) value as the experiment progresses</td>
</tr>
<tr>
<td>Epoch-Greedy</td>
<td>Experiment proceeds as a sequence of epochs where, in each epoch, exploration of new frequency bin(s) is pursued first followed by exploitation of the best frequency bin for the remainder of the epoch</td>
</tr>
</tbody>
</table>

- Random policy procedures implemented primarily for comparison purposes.

\( \text{i.e. the ML decision logic uses the sensing-derived side information about interfered frequency bins as context} \)
The MLBDE software was validated in standalone mode using channel sensor input data collected offline and saved to file:

- MLBDE software compiled with `g++ 7.5.0` (Ubuntu 7.5.0-3ubuntu1~18.04)
- Tested on a “modest” Ubuntu 18.04 machine (Intel(R) Core(TM)2 Duo CPU P8700 @ 2.53GHz, 2 GB RAM)

At experiment time, the sensor data was read from file and sent at “high speed” (e.g. ~662 Hz $\leftrightarrow$ ~1.51 ms intervals) by a unit-test driver module to the MLBDE component via UDP socket API:

- Higher channel sensor input data rates are potentially supportable and are an area of further investigation

*Epoch-Greedy* was the ML heuristic enabled in the experiments behind these results

<table>
<thead>
<tr>
<th>Test Purpose</th>
<th>Result / Key Finding</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compare Random policy versus Greedy heuristic</td>
<td>ML with Epoch-Greedy heuristic in use for 2 distinct input data sets reduced avg. BER of selected frequency bin by factors of 7.76 and 26.9 versus avg. BER achieved by a baseline Random policy</td>
</tr>
<tr>
<td>Process fast channel sensor updates</td>
<td>For a representative experiment with 101 frequency bins and 10 interfering signals per sensor update (@ ~662 Hz), MLBDE processed sensor updates originated every ~1509 $\mu$s in real-time</td>
</tr>
<tr>
<td>Validate use of interference side information</td>
<td>Epoch-Greedy ML use of interfering signal side information yielded a reduction in avg. BER by a factor of ~2 (versus w/ no side information) $\Rightarrow$ Example of spectrum sensing benefit</td>
</tr>
<tr>
<td>Verify shift to robust bins by using $s_N(k)$</td>
<td>Using a compound metric ($\bar{R}<em>N(k) - s_N(k)$) to select next frequency bin ($A</em>{N+1}$) versus pure reward metric ($\bar{R}_N(k)$) shifts selected the bin preference to less-interfered bins by up to 74%</td>
</tr>
</tbody>
</table>
Simulated EM Environment

Orthogonal Matching Pursuit (OMP), GNURADIO, MLBDE

RX Section of B200

Environment Emulator

Static Jammer

Agile Jammer

EMEA Sensor and ITRN

Receiver 1 (X300) Sensor w/ Basic RX board (Baseband I/Q)

Band A

CS-EMEA Sensor (DSIC)

Band B

TX Section of B200
Demo Video
MLBDE output changes with jammer placement

MLBDE Channel selection (green) movement when static jammer added

System GUI and Example Data
### Future Work Sensor Work

<table>
<thead>
<tr>
<th>Name</th>
<th>TS-QAIC</th>
<th>DRF2IC</th>
<th>DSIC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reference</td>
<td>[26]</td>
<td>[27]</td>
<td>[67]</td>
</tr>
<tr>
<td>Year</td>
<td>2015</td>
<td>2017</td>
<td>2018</td>
</tr>
<tr>
<td>Process</td>
<td>TSMC 65nm</td>
<td>TSMC 65nm</td>
<td>TSMC 65nm</td>
</tr>
<tr>
<td>Does CS-Spectrum sensing?</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Num Antennas</td>
<td>1</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>Does CS-DoA Sensing?</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Key Features</td>
<td>Signal detector, 16 physical branches, Time Segmentation</td>
<td>Signal detector and receiver, noise canceling, 4 physical branches, branch expansion and time segmentation.</td>
<td>Signal detector and beamforming receiver, 8 antennas in 2 subarrays of 4, branch expansion, interleaved CS-DOA and CS-Spectrum sensing.</td>
</tr>
<tr>
<td>Resource Cube Visualization</td>
<td><img src="image1" alt="Visualization" /></td>
<td><img src="image2" alt="Visualization" /></td>
<td><img src="image3" alt="Visualization" /></td>
</tr>
</tbody>
</table>

---

**Name**

**TS-QAIC**

**DRF2IC**

**DSIC**

**Reference**

[26]  

[27]  

[67]  

**Year**

2015  

2017  

2018  

**Process**

TSMC 65nm  

TSMC 65nm  

TSMC 65nm  

**Does CS-Spectrum sensing?**

Yes  

Yes  

Yes  

**Num Antennas**

1  

1  

8  

**Does CS-DoA Sensing?**

No  

No  

Yes  

**Key Features**

Signal detector, 16 physical branches,  

Time Segmentation  

Signal detector and receiver, noise canceling,  

4 physical branches, branch expansion and  

Time Segmentation.  

Signal detector and beamforming receiver,  

8 antennas in 2 subarrays of 4,  

branch expansion, interleaved CS-DOA  

and CS-Spectrum sensing.  

**Resource Cube Visualization**

![Resource Cube Visualization](image1)  

![Resource Cube Visualization](image2)  

![Resource Cube Visualization](image3)
Conclusions and Future Work

• A Compressed Sensing (CS) driven receiver architecture can be used to sense the RF spectrum in a fraction of the time as current state-of-the-art techniques.

• As a proof-of-concept, a CS-enabled EM environment aware sensor was successfully integrated into the GNU Radio framework, the output of which indicates the spectral position of jammers or interferers.

• Jammer location is sent to a machine learning-based decision engine (MLBDE) which in turn takes appropriate action (retunes, switches band, etc.) by making the optimal corrective decision based on reinforcement learning.

• We are currently benchmarking the ITRN and comparing it against other EMEA sensing architectures.
References


