MARIO: A Cognitive Radio Primary User Arrivals Data Generator

Rogers S. Cristo*, Guilherme M. D. Santana*,
Diana P. M. Osorio† and Kalinka R. L. J. C. Branco*
*Institute of Mathematics and Computer Science (ICMC) – Universidade de São Paulo (USP),
São Carlos-SP, Brazil
{rogers.cristo, guilherme.santana}@usp.br, kalinka@icmc.usp.br
†Department of Electrical Engineering (DEE) – Universidade Federal de São Carlos
São Carlos-SP, Brazil
dianamoya@ufscar.br

***Abstract*—Cognitive Radio technique has been recognized as one of the most promising solutions for the increasingly growing problem of spectrum scarcity in wireless networks, specially with the emerging of the Internet of Things. In cognitive radio networks, secondary users are allowed to intelligently access licensed bands of primary users, thus enhancing the spectrum utilization. In this context, for investigating the advantages of cognitive radio, Machine Learning techniques have been widely applied to predict primary users arrivals. However, the available simulators are usually complex and highly time consuming. Therefore, in this work, we propose a simple and intuitive primary user arrivals data generator, MARIO, that can produce random arrival data for multiple channels by employing Poisson process. This generator is validated by using the generated data to predict new sequences according to a Hidden Markov Model. Our results show that the data generator can be used to simulate various traffic patterns over different channels.**

***Index Terms*—Cognitive Radio, Data Generator, Simulation, Poisson Process, Hidden Markov Model**

I. Introduction

With the rapid growth of the Internet of Things, billions of devices are expected to be connected to wireless networks, thus spectrum scarcity becomes a critical challenge to be addressed in the near future [1]. Hence, Cognitive Radio (CR) [2] techniques have emerged as one of the most promising solutions to enhance the overall efficiency of the networks by exploiting under-utilized spectrum bands and allowing opportunistic and shared spectrum access, thus significantly increasing the overall spectrum efficiency of wireless networks [3].

In CR networks, secondary users (SUs) are allowed to dynamically access the spectrum belonging to primary users (PUs), that hold the legacy rights on the usage of a specific part of the spectrum. A CR device can act as SU by accessing the licensed spectrum in an opportunistic manner, i.e. when the channels are not occupied by PUs, or in a concurrent manner by controlling the transmission power in order to not interfere the PU. Thus, when the SU detects the arrival of a PU in its current occupied channel (spectrum sensing), it should whether reduce its transmission power to avoid interference, commonly specified as underlay spectrum sharing [4], or it should handoff to a vacant channel (i.e. spectrum handoff) [5]. The spectrum handoff process must be conducted in such a way that its delay causes minimum interference for the PU transmission [6].

Nonetheless, the handoff process introduces additional constraints to the spectrum sharing mechanism. For instance, most of the Software-defined Radio (SDR) devices [7] (i.e. hardware employed to receive and transmit data over a wide range of frequency bands) may only sense a limited radio-spectrum range at a given time. This issue prevents the SU of rapidly sense, detect and move to the most suitable band before producing interference to the PU. Then, in order to enhance the spectrum handoff process, some concepts have been introduced in the literature [8], [9].

Particularly, the spectrum handoff process has been classified in four different strategies [8], [10]: (*i*) non-handoff, i.e., the SU pauses its data transmission until the PU leaves the channel; (*ii*) reactive handoff, meaning that the spectrum search for vacant channels is applied only when the SU detects a PU arrival in the current channel; (*iii*) proactive handoff, when the SU proactively looks for backup channels trying to predict the PU traffic pattern, ideally hoping to a vacant channel before the PU arrival; and (*iv*) hybrid, the SU senses the environment looking for vacant channels and setting up a backup list, but does not try to predict the PU traffic and hops only when PU arrival is detected.

Empirically, the proactive strategy presents superior performance regarding latency reduction [8]. Besides, several studies have shown methods to forecast the PU traffic using techniques such as Autoregressive Integrated Moving Average (ARIMA) [11], Partial Periodic Pattern Mining (PPPM) [12], Artificial Neural Networks (ANN) [13], and Hidden Markov Models (HMM) [14], [15].

The data used to train the aforementioned models are

The authors would like to thank the financial support provided by the Brazilian funding agency CAPES. Research was also sponsored by the Army Research Office and was accomplished under Grant Number W911NF-18-1-0012. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

commonly collected by an SDR hardware. Although some of these devices are cheap [7], [16] (approximately dozens of dollars), some of them may reach relatively high prices (thousands of dollars) [17]. Moreover, the SDRs power consumption may discourage their employment in some critical embedded systems which require the extensive use of batteries, for instance, autonomous vehicles such as Unmanned Aerial Vehicles (UAV). These constrains may raise the difficulty in gathering data, turning it problematic to apply Machine Learning (ML) methods that demand substantial amounts of data [18].

With the purpose of mitigating these drawbacks, some complex CR simulators have been introduced into the literature, which are mainly subdivided into physical layer simulators, such as GNU Radio [19], and network simulators, such as Omnet++/INET [20], [21]. Physical layer simulators are capable of generating data including CR transmission features such as waveform generators, modulators (e.g. AM, PSK, DPSK), and signal processing analysis (e.g. FFT, Fractional Resampler, FLLBand-Edge). On the other hand, network simulators focus on generation of data regarding, e.g., protocols, mobility, and packages transmission routing.

Despite of the rich features of the available simulators, in some cases their complexity can also be considered a limitation. For instance, a PU arrival simulation concerning several channels could take hours of setup on either GNU Radio or Omnet++. According to [22], a Poisson Process could be employed to achieve this simulation, and it would feature, for example, channel traffic patterns to emulate and generate realistic data. However, even though this stochastic process has been frequently used in the context of CR [23]–[25], to the best of our knowledge, no simple data generator based on Poisson Process has been proposed in the literature.

Based on the above context, this paper aims to partially fill this gap by proposing a simple and intuitive primary user arrivals data generator (MARIO - pri**MA**ry use**R** arr**I**vals data generat**O**r), which is able to produce random arrival data for multiple channels. To verify its usability, we further setup an HMM to predict the channel traffic pattern, which is running on the simulated data.

The remaining of this paper is organized as follows. In Section II, the Poisson Process algorithm and the HMM configuration are introduced. The data generated using MARIO, as well as the validation test using the HMM are presented in Section III. Some key advantages and potential improvements in our proposal are exposed in Section IV. Finally, Section V concludes this paper.

II. Methods

The implementation of MARIO can be subdivided into two fundamental segments: (*i*) generation of PU arrivals data by using a stochastic process and (*ii*) tool validation by using a consolidated ML technique. Furthermore, this section also outlines the Python libraries utilized to implement MARIO. All code is hosted at GitHub[1] under a MIT license.

[1]https://github.com/rogerscristo/MARIO

A. MARIO

1) Poisson Process: As specified by [23]–[25], in the CR channel utilization context, licensed-users traffic pattern can be modeled as a Poisson Process. This statistical model has been widely applied in the communications area to simulate and generate time-sequential data [22].

A Poisson Process is a simple and widely used stochastic process for modeling the times at which arrivals enter a system. In this work, we adopt this process to model the PU arrivals, where the elapsed time between two arrivals is modeled as an exponential random variable. [26]:

$$\Pr\left(X_t \leq x\right) = 1 - e^{-\lambda x}, \tag{1}$$

assuming an arrival at time t, X_t stands for the time until the next arrival, whereas x corresponds to the quantity of time units and λ is the average rate of arrivals per time unit t.

To generate the arrivals during the simulation stipulated time we have used Algorithm 1. Although the literature reports the channel time use of an individual user being distributed as the negative exponential [27], for the sake of simplicity this work addresses RTD as the uniformly distributed random transmission duration. Following, RAT is the random arrival time, MSD stands for the maximum simulation duration (user defined), TD is the transmission duration, ARV accounts for the array of arrivals. $\mathbb{R} \cap [0, 1]$ corresponds to the acquisition of a random real number between 0 and 1, whereas $\mathbb{N} \cap [2, TD]$ stands for the acquisition of a random natural number between 2 and TD.

Algorithm 1: PU Arrivals data using a Poisson Process

$RTD \leftarrow 0$
while $RTD < MSD$ **do** *Generate arrival spacing*
  $RAT \leftarrow \dfrac{-\ln(\mathbb{R} \cap [0, 1])}{\lambda} + RTD$
  $RTD \leftarrow RAT + \mathbb{N} \cap [2, TD])$
  for $x \leftarrow RAT$ **to** RTD **do** *Populate result array*
    if $x < MSD$ **then** $ARV[x] \leftarrow 1$;
  end
end
return ARV

2) Tool setup: In order to provide PU arrivals data, MARIO requires the user to specify a collection of parameters. The following options will appear during the setup:

- Simulation duration, with the maximum value of 24 hours.
- Number of channels.
- For each channel:
  1) The average time between arrivals, given in minutes.
  2) The average PU transmission duration, given in minutes.

- Option to save the generated data into a *csv* file.
- Option to plot and save the generated data into a *png* file.

B. Hidden Markov Model

Hidden Markov Model is an unsupervised machine learning model mainly used on the prediction of events sequences (e.g. most probable sequences of words). Unlike the regular Markov model, where all states are observable, the HMM assumes that states can be hidden, which is an attribute that fits well for real-world problems. A formal definition [28] is presented in the following paragraphs.

The observation event sequence can be represented as:

$$\begin{aligned} O &= \{O_1, O_2, O_3, \cdots, O_T\} \\ &= \{\mathcal{P}, \mathcal{A}, \mathcal{P}, \cdots, \mathcal{P}\}, \end{aligned} \tag{2}$$

where T stands for the time on the last observation event, $\mathcal{P}$ stands for the detected PU presence and $\mathcal{A}$ stands for PU absence.

Similarly, the HMM individual hidden states S can be defined as:

$$S = \{S_1, S_2, S_3, \cdots, S_N\}, \tag{3}$$

where N is the total number of states. In this work we adopted $N = 2$.

As in a Markov chain, the transitions between the HMM hidden states are represented by a matrix of probabilities, which can be expressed as:

$$A = \{a_{i,j}\} = P[q_{t+1} = S_j \mid q_t = S_i], \quad 1 \leqslant i, j \leqslant N, \tag{4}$$

where the state at time t is denoted by q_t.

The probability of a given event associated with Sj is defined by:

$$B = \{b_j(k)\} = P[V_k \text{ at } t \mid q_t = S_j], \quad 1 \leqslant j \leqslant N, \tag{5}$$

where V_k stands for an individual symbol observed in $q_t = S_j$. In the scope of this work $V_0 = 0$ and $V_1 = 1$.

The initial state distribution π, i.e. the probability of the sequence starts at a given state S_i, can be expressed as:

$$\pi = \{\pi_i\} = P[q_1 = S_i], \quad 1 \leqslant i \leqslant N. \tag{6}$$

Commonly, an HMM may be applied whether (*i*) to estimate the probability of a given sequence happening in time $T + 1$, given a training sequence O (e.g. the probability of $\{\mathcal{A}, \mathcal{A}, \mathcal{P}\}$ given $O = \{\mathcal{P}, \mathcal{A}, \mathcal{P}\}$), or (*ii*) to generate the most likely sequence of states Q given a training sequence O (e.g., if we have $O = \{\mathcal{P}, \mathcal{A}, \mathcal{P}\}$, what is the most probable state sequence in the near future?). In this work we have validated the data generated by MARIO using (*ii*) and applying a Viterbi Algorithm (VA) to decode the unknown state sequence.

The Viterbi Algorithm was initially proposed by [29] and its integration with HMM has been applied to several areas, specially in speech recognition [30] and in bioinformatics [31]. Accordingly, due to sequential data characteristic of licensed users arrivals in CR channels, the VA have been previously adopted to both spectrum handoff and spectrum sharing enhancement [32]. Algorithm 2 presents the procedure applied in this work [28].

Algorithm 2: Viterbi algorithm

for each state $i \in \{1, 2, 3, \cdots, N\}$ **do**
  $T_1[i, 1] \leftarrow \pi_i \cdot B_{i,O_1}$
  $T_2[i, 1] \leftarrow 0$
end
for each observation $i = 2, 3, \cdots, T$ **do**
  for each state $j \in \{1, 2, 3, \cdots, N\}$ **do**
    $T_1[j, i] \leftarrow \max_n \left(T_1\left[n, i-1\right] \cdot A_{n,j} \cdot B_{j,O_i}\right)$
    $T_2[j, i] \leftarrow \arg\max_n \left(T_1\left[n, i-1\right] \cdot A_{n,j} \cdot B_{j,O_i}\right)$
  end
end
$Z_T \leftarrow \arg\max_n \left(T_1\left[k, T\right]\right)$
$Q_T \leftarrow S_{Z_T}$
for $i \leftarrow T, T-1, T-2, \cdots, 2$ **do**
  $Z_{i-1} \leftarrow T_2\left[Z_i, i\right]$
  $Q_{i-1} \leftarrow S_{Z_{i-1}}$
end
return Q

Figure 1 presents a better visualization of the HMM configuration implemented for this work. The hidden states are set to Presence or Absence, whereas the observation states are arranged as 0 or 1.

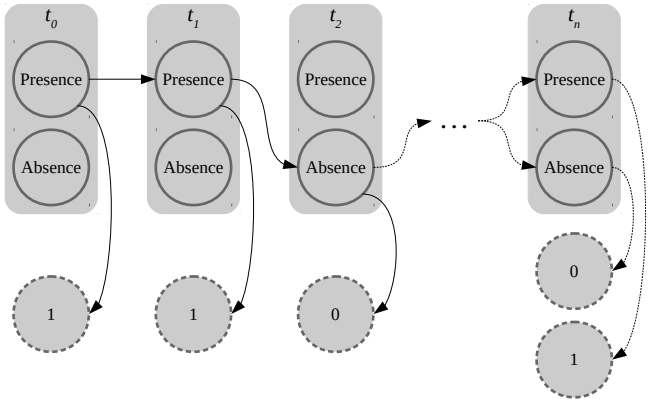

Figure 1. HMM Trellis diagram modeled for this domain.

C. Python and libraries

Python is an open-source, interpreted, dynamically-typed programming language, initially released in 1991 [33]. It was developed to be explicit, beautiful, and simple, which enabled it to become the 4th most popular programming language in the world in 2018 [34]. Due to its simplicity, Python has been commonly applied in scientific fields, such as statistics, physics and scientific computing [35].

We developed MARIO by applying four well-known Python 3.6.4 libraries to array manipulation (NumPy), data visual-

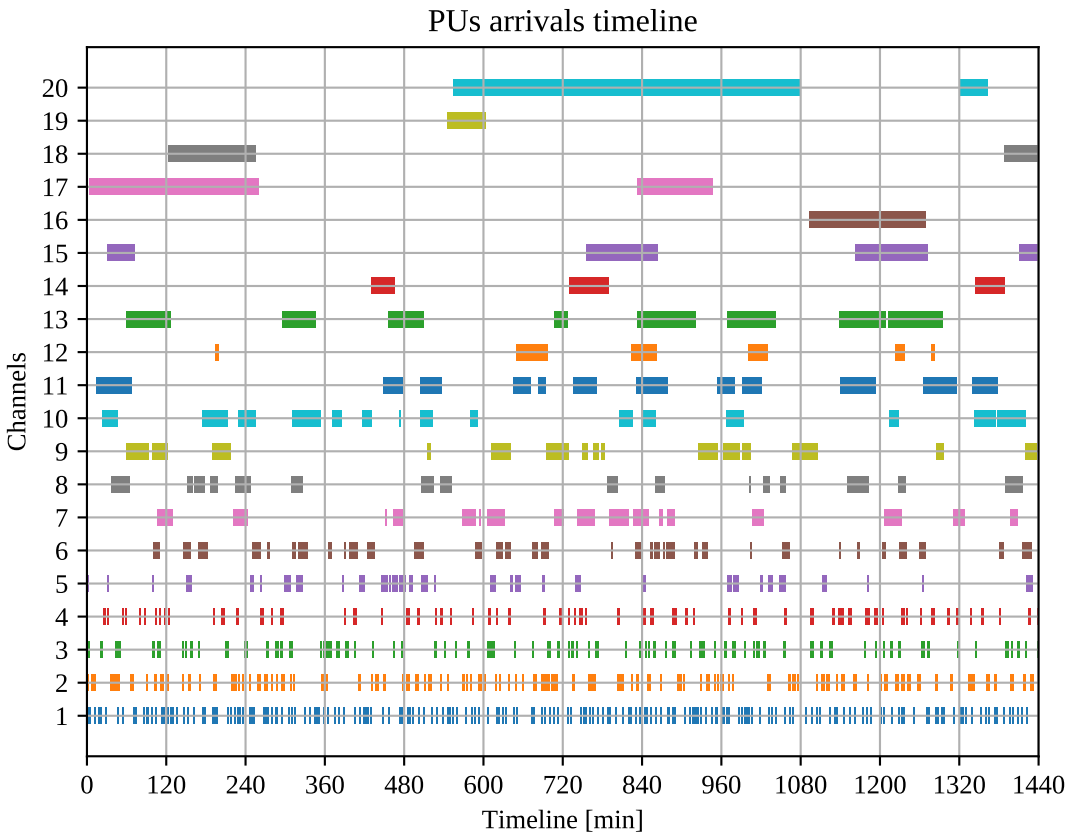

Figure 2. Simulation with 20 channels and 1440 min of duration. Each channel was individually configured. All the parameters were arbitrarily chosen.

ization (Matplotlib), data formatting (Pandas) and machine learning (TensorFlow), which are detailed below.

1) NumPy: is a Python scientific computing open-source library released in 2006. It supports the use of *n*-dimensional arrays, broadcast functions, C/C++ and Fortran integration, as well as linear algebra, Fourier transforms and random numbers capabilities [36].

2) Matplotlib: originally developed to mimic MATLAB interface, Matplotlib is a widely used plotting Python open-source library [37]. It provides several plot types such as line plots, images, contouring, histograms, paths, streamplots, pie charts, bar charts, scatter plots, including others.

3) Pandas: is an easy-to-use data manipulation open-source Python library, designed to provide intuitive data analysis on relational or labeled data [38]. It is well-suited for tabular data, ordered and unordered time series data, arbitrary matrix data with row and column labels, and any other form of statistical or observational data sets.

4) Tensorflow: was released in 2015 as an open-source machine learning library providing APIs for Python, C++, Haskell, Java, Go and Rust [39]. It was originally developed by Google Brain as a proprietary machine learning framework. In the last few years it has been applied to several real-world applications, such as Google's Inception Image Classification Model [40] and SmartReply [41], Mozilla's Deep Speech [42], as well as in machine learning scientific research [43].

III. Results

A. Data generation

Table I presents the setup parameters used to generate the data presented in this section. The simulation was arbitrarily set to 20 channels, with an average time between arrivals arbitrarily varying between 5 and 720 minutes, whereas the average PU transmission duration ranges were arbitrarily varying from 3 to 540 minutes.

Figure 2 illustrates the 20 channels data produced from the setup displayed in Table I. The *x*-axis ticks were adjusted to appear every 120 minutes. From the plot, it can be seen that each channel simulation can be configured to produce a unique traffic pattern. Hence, depending on the parameters tune, MARIO can more accurately mimic real-world channel behaviors. It is important to notice that channels with the same color does not have any parameters relation.

B. Prediction

An HMM was employed to endorse the usability of the data generated by MARIO. Figure 3 presents the results concerning HMM prediction. Channel data were arranged into 80% and 20% for training and comparison, respectively, with the forecast state sequence.

Table I
DATA GENERATION SETUP

Simulation duration: 24h		
Channel	**Avg. time between arrivals [min]**	**Avg. PU transmission duration [min]**
1	5	3
2	10	5
3	15	5
4	20	5
5	30	10
6	40	15
7	50	30
8	55	35
9	60	40
10	90	45
11	90	60
12	120	60
13	120	90
14	240	90
15	240	120
16	480	240
17	480	300
18	720	300
19	720	480
20	720	540

When comparing the predict data with the test sequence it can be observed that not every predicted arrival matches its correspondent event at the same moment in the test data. However, to avoid interference, the forecast of a PU arrival should at least be done before the event. Accordingly, the predicted sequence distribution seems to well fit test data, which testifies to the effectiveness of HMM producing unknown sequences.

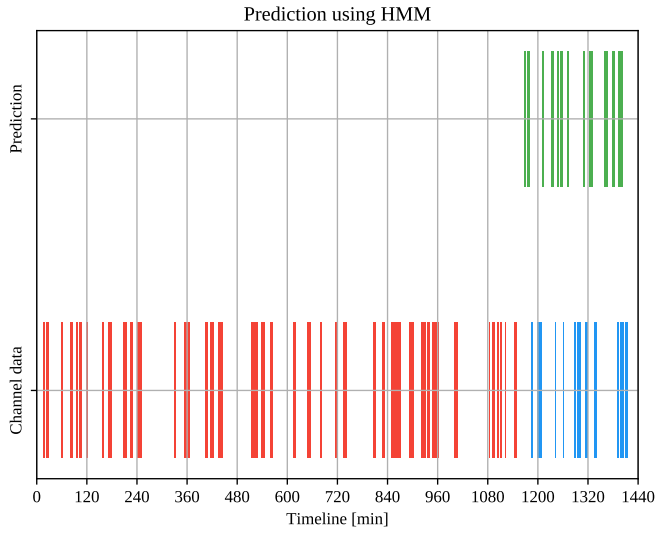

Figure 3. Prediction of one channel using HMM. Data organization: red corresponds to the 80% portion of training and blue corresponds to the 20% portion using to visually compare with forecast sequence, in green.

IV. DISCUSSION

Although the use of a Poisson Process to generate random arrival data was mentioned in the CR literature [23]–[25], as far as we know, no tool has been proposed concerning the PU arrival data generation. Thus, this work presents MARIO as an easy-to-use lightweight Python script, abstracting the algorithmic and statistical details from the user and not requiring significant computing power for this purpose. In addition, its open-source characteristic allows users to adapt it in the favor of their own needs.

Different from simulators such as Omnet++ and INET [20], [21], our proposal is unable to encompass device's mobility, and the constraints related to this (e.g. channel traffic pattern variation due to geolocation variation). This work could, therefore, benefit from the integration between geolocation and channels traffic pattern, making the simulation even more realistic and enhancing it to the use of autonomous vehicles such as UAVs.

Additionally, so far, the data produced by MARIO is narrowed to the use of a Poisson Process as a generator, not providing any support to production of arrivals sequences using real data collected by SDR devices such as HopeRF's RFM22BW [16] and Realtek's RTL2832U [7]. In this way, a method like HMM is a reasonable procedure to take real data and produce new arrivals sequences, enabling adapted simulation regarding a specific real-world frequency traffic pattern.

Finally, the first version of MARIO only provides a terminal interface. The evolution to a graphical user interface (GUI) could significantly increase its user experience.

V. CONCLUSION

This work presented a simple and intuitive PU arrivals data generator. In order to achieve the random generation of PU arrivals, we implemented an algorithm using a Poisson Process. To validate the generated data, we employed an HMM to predict unknown sequences by applying the training on the simulated data. The present study offers the opportunity of producing spectrum traffic data without the fall-downs of using a complex and, usually, counter-intuitive simulator. It is beyond the scope of this study to propose a tool which encompasses the whole set of features concerning a full-scale cognitive radio simulation. However, future work will consider the application of real-data to enhance the generated-data fidelity, mobility constrains, and a graphical user interface.

REFERENCES

[1] G. A. Akpakwu, B. J. Silva, G. P. Hancke, and A. M. Abu-Mahfouz, "A survey on 5g networks for the internet of things: Communication technologies and challenges," *IEEE Access*, vol. 6, pp. 3619–3647, 2018.

[2] J. Mitola and G. Q. Maguire, "Cognitive radio: making software radios more personal," *IEEE Personal Communications*, vol. 6, no. 4, pp. 13–18, Aug 1999.

[3] J. Ren, Y. Zhang, R. Deng, N. Zhang, D. Zhang, and X. Shen, "Joint channel access and sampling rate control in energy harvesting cognitive radio sensor networks," *IEEE Transactions on Emerging Topics in Computing*, vol. PP, no. 99, pp. 1–1, 2017.

[4] M. G. Khoshkholgh, K. Navaie, and H. Yanikomeroglu, "Access strategies for spectrum sharing in fading environment: Overlay, underlay, and mixed," *IEEE Transactions on Mobile Computing*, vol. 9, no. 12, pp. 1780–1793, Dec 2010.

[5] A. Lertsinsrubtavee, N. Malouch, and S. Fdida, "Controlling spectrum handoff with a delay requirement in cognitive radio networks," in *2012 21st International Conference on Computer Communications and Networks (ICCCN)*, July 2012, pp. 1–8.

[6] Y. Wu, Q. Yang, X. Liu, and K. S. Kwak, "Delay-constrained optimal transmission with proactive spectrum handoff in cognitive radio networks," *IEEE Transactions on Communications*, vol. 64, no. 7, pp. 2767–2779, July 2016.

[7] Realtek Semiconductor Corp. (2018, jan) Rtl2832u - dvb-t cofdm demodulator + usb 2.0. [Online]. Available: http://www.realtek.com.tw/products/productsView.aspx?Langid=1&PFid=35&Level=4&Conn=3&ProdID=257

[8] I. Christian, S. Moh, I. Chung, and J. Lee, "Spectrum mobility in cognitive radio networks," *IEEE Communications Magazine*, vol. 50, no. 6, pp. 114–121, June 2012.

[9] I. Hanif, M. Zeeshan, and A. Ahmed, "Traffic pattern based adaptive spectrum handoff strategy for cognitive radio networks," in *2016 10th International Conference on Next Generation Mobile Applications, Security and Technologies (NGMAST)*, Aug 2016, pp. 18–23.

[10] E. Hossain, D. Niyato, and D. I. Kim, "Evolution and future trends of research in cognitive radio: a contemporary survey," *Wireless Communications and Mobile Computing*, vol. 15, no. 11, pp. 1530–1564, 2015. [Online]. Available: http://dx.doi.org/10.1002/wcm.2443

[11] Z. Wang and S. Salous, "Spectrum occupancy statistics and time series models for cognitive radio," *Journal of Signal Processing Systems*, vol. 62, no. 2, pp. 145–155, Feb 2011. [Online]. Available: https://doi.org/10.1007/s11265-009-0352-5

[12] P. Huang, C. J. Liu, L. Xiao, and J. Chen, "Wireless spectrum occupancy prediction based on partial periodic pattern mining," in *2012 IEEE 20th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, Aug 2012, pp. 51–58.

[13] L. Yin, S. Yin, W. Hong, and S. Li, "Spectrum behavior learning in cognitive radio based on artificial neural network," in *2011 - MILCOM 2011 Military Communications Conference*, Nov 2011, pp. 25–30.

[14] M. T. Soleimani, M. Kahvand, and R. Sarikhani, "Handoff reduction based on prediction approach in cognitive radio networks," in *2013 15th IEEE International Conference on Communication Technology*, Nov 2013, pp. 319–323.

[15] C. Pham, N. H. Tran, C. T. Do, S. I. Moon, and C. S. Hong, "Spectrum handoff model based on hidden markov model in cognitive radio networks," in *The International Conference on Information Networking 2014 (ICOIN2014)*, Feb 2014, pp. 406–411.

[16] HOPE Microelectronics. (2018, jan) Rfm22b fsk transceiver. [Online]. Available: http://www.hoperf.com/rf_transceiver/modules/RFM22BW.html

[17] Ettus Research. (2018, jan) Usrp - universal software radio peripheral. [Online]. Available: https://www.ettus.com/product/quick-order

[18] A. R. Young and C. W. Bostian, "Simple and low-cost platforms for cognitive radio experiments [application notes]," *IEEE Microwave Magazine*, vol. 14, no. 1, pp. 146–157, Jan 2013.

[19] The GNU Radio Foundation. (2018, jan) Gnu radio - the free and open software source radio ecosystem. [Online]. Available: https://www.gnuradio.org/

[20] OpenSim Ltd. (2018, jan) Omnet++: Discrete event simulator. [Online]. Available: https://www.omnetpp.org/

[21] ——. (2018, jan) Inet framework for omnet++. [Online]. Available: https://inet.omnetpp.org/

[22] F. Baccelli and B. Blaszczyszyn, *Stochastic Geometry and Wireless Networks, Volume II - Applications*, ser. Foundations and Trends in Networking: Vol. 4: No 1-2, pp 1-312, F. Baccelli and B. Blaszczyszyn, Eds. NoW Publishers, 2009, vol. 2, stochastic Geometry and Wireless Networks, Volume I - Theory; see http://hal.inria.fr/inria-00403039. [Online]. Available: https://hal.inria.fr/inria-00403040

[23] J. DUAN and Y. LI, "An optimal spectrum handoff scheme for cognitive radio mobile ad hoc networks," *Advances in Electrical and Computer Engineering*, vol. 11, no. 3, pp. 11–16, 2011. [Online]. Available: https://doi.org/10.4316/aece.2011.03002

[24] K. A. M., F. Hu, and S. Kumar, "Intelligent spectrum management based on transfer actor-critic learning for rateless transmissions in cognitive radio networks," *IEEE Transactions on Mobile Computing*, pp. 1–1, 2017. [Online]. Available: https://doi.org/10.1109/tmc.2017.2744620

[25] S. Nejatian, S. K. Syed-Yusof, N. M. A. Latiff, V. Asadpour, and H. Hosseini, "Proactive integrated handoff management in cognitive radio mobile ad hoc networks," *EURASIP Journal on Wireless Communications and Networking*, vol. 2013, no. 1, sep 2013. [Online]. Available: https://doi.org/10.1186/1687-1499-2013-224

[26] F. Baccelli and B. Blaszczyszyn, *Stochastic Geometry and Wireless Networks, Volume I - Theory*, ser. Foundations and Trends in Networking Vol. 3: No 3-4, pp 249-449, F. Baccelli and B. Blaszczyszyn, Eds. NoW Publishers, 2009, vol. 1, stochastic Geometry and Wireless Networks, Volume II - Applications; see http://hal.inria.fr/inria-00403040. [Online]. Available: https://hal.inria.fr/inria-00403039

[27] T. Rappaport, *Wireless Communications: Principles and Practice*, 2nd ed. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[28] L. R. Rabiner, "A tutorial on hidden markov models and selected applications in speech recognition," *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257–286, Feb 1989.

[29] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, April 1967.

[30] A. V. Nefian, L. Liang, X. Pi, L. Xiaoxiang, C. Mao, and K. Murphy, "A coupled hmm for audio-visual speech recognition," in *2002 IEEE International Conference on Acoustics, Speech, and Signal Processing*, vol. 2, May 2002, pp. II–2013–II–2016.

[31] M. Remmert, A. Biegert, A. Hauser, and J. Söding, "HHblits: lightning-fast iterative protein sequence searching by HMM-HMM alignment," *Nature Methods*, vol. 9, no. 2, pp. 173–175, dec 2011. [Online]. Available: https://doi.org/10.1038/nmeth.1818

[32] X. Xing, T. Jing, W. Cheng, Y. Huo, and X. Cheng, "Spectrum prediction in cognitive radio networks," *IEEE Wireless Communications*, vol. 20, no. 2, pp. 90–96, April 2013.

[33] Python Software Foundation. (2018, mar) Python programming language. [Online]. Available: https://www.python.org/

[34] TIOBE software BV. (2018, mar) Tiobe index. [Online]. Available: https://www.tiobe.com/tiobe-index/

[35] Ž. Ivezić, A. J. Connolly, J. T. Vanderplas, and A. Gray, *statistics, data mining, and machine learning in astronomy: a practical python guide for the analysis of survey data*. Princeton University Press, 2014.

[36] T. E. Oliphant, *A guide to NumPy*. Trelgol Publishing USA, 2006, vol. 1.

[37] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science Engineering*, vol. 9, no. 3, pp. 90–95, May 2007.

[38] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference*, S. van der Walt and J. Millman, Eds., 2010, pp. 51 – 56.

[39] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org. [Online]. Available: https://www.tensorflow.org/

[40] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov *et al.*, "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014. [Online]. Available: http://arxiv.org/abs/1409.4842

[41] Greg Corrado. (2018, mar) Computer, respond to this email. [Online]. Available: https://research.googleblog.com/2015/11/computer-respond-to-this-email.html

[42] mozilla.org. (2018, mar) Project deepspeech - a tensorflow implementation of baidu's deepspeech architecture. [Online]. Available: https://github.com/mozilla/deepspeech

[43] V. Sze, Y. H. Chen, T. J. Yang, and J. S. Emer, "Efficient processing of deep neural networks: A tutorial and survey," *Proceedings of the IEEE*, vol. 105, no. 12, pp. 2295–2329, Dec 2017.