

# Rate Prediction and Selection in LTE systems using Modified Source Encoding Techniques

Saishankar K.P. \*

Sheetal Kalyani \*

Narendran K. †

Department of Electrical Engineering\* Centre for Excellence in Wireless Technology,†

Indian Institute of Technology, Madras

IITM Research Park ,

Chennai, India 600036

Chennai, India 600113

{ee09d025,skalyani}@ee.iitm.ac.in

knaren@cewit.org.in

arXiv:1403.1412v5 [stat.AP] 8 Aug 2014

## Abstract

In current wireless systems, the base-Station (eNodeB) tries to serve its user-equipment (UE) at the highest possible rate that the UE can reliably decode. The eNodeB obtains this rate information as a quantized feedback from the UE at time  $n$  and uses this, for rate selection till the next feedback is received at time  $n + \delta$ . The feedback received at  $n$  can become outdated before  $n + \delta$ , because of a) Doppler fading, and b) Change in the set of active interferers for a UE. Therefore rate prediction becomes essential. Since, the rates belong to a discrete set, we propose a discrete sequence prediction approach, wherein, frequency trees for the discrete sequences are built using source encoding algorithms like Prediction by Partial Match (PPM). Finding the optimal depth of the frequency tree used for prediction is cast as a model order selection problem. The rate sequence complexity is analysed to provide an upper bound on model order. Information-theoretic criteria are then used to solve the model order problem. Finally, two prediction algorithms are proposed, using the PPM with optimal model order and system level simulations demonstrate the improvement in packet loss and throughput due to these algorithms.

## I. INTRODUCTION

4G systems, based on standards such as Long Term Evolution (LTE) offer peak data rates of upto 300 Mbps [1] and rate adaptation through adaptive modulation has played a crucial role in facilitating this. Adaptive modulation techniques exploit the variations in the wireless

channel by trying to communicate at a rate (bits per channel use), that is suited to the current channel conditions. 4G standards such as LTE supports upto 28 different rates on the downlink. The transmitter will not know the SINR at the receiver, and hence needs rate feedback from the receiver. Since we are looking at the downlink of a cellular system, the transmitter is always the Base-station/evolved NodeB (eNodeB) and the receiver is the User Equipment(UE)<sup>1</sup> [1].

The UE first measures/estimates the post-processing SINR *i.e.*, the SINR seen after receive processing such as, Minimum Mean Squared Error (MMSE) detection. Then, it calculates a rate metric which reflects the channel capacity based on standard link adaptation/abstraction techniques [1]. Typically, this rate metric is quantized, and LTE supports 4 bit quantization. The quantized feedback is called Channel Quality Indicator (CQI), and it is a number between 0 and 15 [1]. The CQI feedback is done by all UEs in the the system and each UE may use different techniques for SINR measurements and rate calculations, as, these algorithms are proprietary to each receiver. The 4 bit CQI value received at the eNodeB is mapped to a 5 bit value (28 possible states) called the Modulation and Coding Scheme index (MCS). Once the CQI feedback received at time  $n$  from a user  $u$  is mapped to an MCS value  $X_n^u$ , it will be used till the next CQI feedback is received and mapped at time  $n + \delta$  to  $X_{n+\delta}^u$ . In this work we look at prediction of the MCS indices  $X_{n+i}^u$  for times  $i = 1, 2, \dots, \delta - 1$  using the discrete sequence of past values  $\{X_n^u, X_{n-\delta}^u, X_{n-2\delta}^u, \dots\}$ . There are two reasons why prediction of MCS index is required:

- 1) The MCS available at time  $n$  may have been computed from a CQI estimated by a UE at time  $n - \gamma$ , where  $\gamma$  is the reporting delay and this shall be henceforth referred to as delayed MCS. A detailed study of the effect of CQI delay is provided in [2], [3].
- 2) The MCS available at  $n$  ( $X_n^u$ ) has to be used till time  $n + \delta$ . The channel and interference conditions can change between  $n$  and  $n + \delta$  leading to outdated MCS value  $X_n^u$ . Our focus in this work is on the effect of outdated MCS.

While the problem of delayed MCS can be addressed at the UE, the problem of outdated MCS cannot be addressed by the UE alone. This is because, if the UE were to predict and feedback the CQI meant for  $n + \delta$  at  $n$ , the eNodeB would be left with no knowledge as to what MCS is to be used for times  $n, n + 1 \dots n + \delta - 1$ . Therefore, it is necessary that the eNodeB has a prediction

<sup>1</sup>In the uplink the eNodeB knows the SINR since it is the receiver.

mechanism to handle the outdated MCS problem. There are various prediction schemes [4]–[7] that can be implemented at the UE which can correct for delayed CQI and complement the proposed prediction scheme used at the eNodeB.

The MCS  $X_n^u$  can become outdated by  $n + i$ , where  $i < \delta$ , due to the change in SINR over time because of the following reasons :

- 1) The desired signal and interference power changes gradually over time due to Doppler effect, and the change is a function of the mobility of UEs and the scattering objects.
- 2) The active set of interfering eNodeBs for a specific UE can change over time due to the following reasons:
  - a) The traffic patterns at the different eNodeBs may change over time, and when an eNodeB does not have enough data to send, it does not transmit over all sub-bands. For example, a user  $u$  scheduled in band  $i$  at time  $n$  sees eNodeBs indexed as 1,5,9 as its interferers, however by  $n + \delta$  a couple of eNodeBs from that set may have stopped transmitting and some other eNodeB which was inactive at  $n$  may have become active at  $n + \delta$  in band  $i$  leading to  $u$  seeing a different set of active interferers.
  - b) In the case of Het-Nets, in order to reduce the interference seen by pico eNodeBs, the macro eNodeBs may not transmit on certain bands on which the pico is transmitting [8], [9]. This is called sub-frame blanking and the set of active bands for an eNodeB changes dynamically when dynamic sub-frame blanking is employed [8] resulting in a change in the active set of interferers for UEs attached to neighbouring eNodeBs. The transmission power of a macro eNodeB is 46 dBm, while that of pico is only around 23-30 dBm [9]. Therefore, when the eNodeB does not transmit in some sub-frames, it ceases to be an active interferer for UEs attached to the neighboring eNodeBs and the pico power is too low for it to become a dominant interferer.

If the system is such that all eNodeBs transmit data always and the change is only due to Doppler, it is called a fully loaded system. On the other hand, if all eNodeBs do not transmit over all resources, it is referred to as partial loading <sup>2</sup>. Typically, the change in SINR due to partial loading is more abrupt, leading to higher variability in MCS values.

<sup>2</sup>Note that we are looking at reuse-one LTE system where all the frequency bands are used in all eNodeBs, and in partial loading some bands may be unoccupied

There are many CQI prediction methods, proposed in [4]–[7] with the objective of improving link adaptation. In [4] the authors perform channel prediction using Jakes and ITU models and use it for CQI updation. In [6] also channel prediction is employed to estimate the future CQI. In [5], the authors treat the CQI prediction as a filtering-prediction problem, where they treat the CQI as a real number and use a linear predictor which minimizes the Mean Square Error of the CQI estimate. It can be seen that, all the above papers, treat CQI as a continuous quantity and use filtering based prediction approaches. Furthermore, the focus is more on the effect of delayed CQI/MCS and partial loading has not been considered.

These techniques should be applied only at the UE, because, a continuous CQI viz., the actual value of SINR is available only at the UE. At each UE, the SINR-CQI mapping is done based on the receive algorithms used by it<sup>3</sup>, the transmission mode and the SINR estimation itself may be different for different users [10]. This results in different receivers computing/predicting the CQI using different techniques. Since CQI is quantized, the SINR to CQI mapping is non-invertible and furthermore, the SINR to CQI mapping employed at each UE is unknown to the eNodeB. Hence, mapping the MCS back to SINR at the eNodeB will not improve prediction accuracy. Moreover, since the eNodeB selects only a discrete rate, one can apply discrete sequence prediction, wherein, a temporal distribution of the MCS values can be built and exploited for prediction. This technique of building the MCS distribution is practically viable only if the MCS comes from a discrete set.

We assume that the feedback is periodic with time period  $\delta$  (5ms), thus the eNodeB by time  $n$  will have received a sequence  $\{X_n^u, X_{n-\delta}^u \dots X_0^u\}$  from the user  $u$ .<sup>4</sup> Our aim is to predict  $X_{n+\delta}^u$  given this discrete sequence. If the joint distribution between the future and the past, i.e.,  $P(X_0^u \dots X_n^u, X_{n+\delta}^u)$  is known, we would be able to optimally predict  $X_{n+\delta}^u$  from the previously observed sequence. However, as this distribution is not known, we propose to build the joint distribution, for each user  $u$ .

We initially propose to use algorithms from source encoding to estimate the distribution of the MCS sequence of each UE, since estimating the distribution of a source transmitting symbols, is a problem that has been studied extensively in source encoding. Certain issues in practically

<sup>3</sup>For the same SINR different receive schemes say MRC,MMSE or ML can support different rates.

<sup>4</sup>However, the approach proposed in this work can be modified and used even if the feedback is non-periodic or event triggered.

applying these algorithms are discussed, and appropriate modifications are proposed. In this paper, two source encoding algorithms namely Active Lempel Ziv (Active LeZi) and Prediction by Partial Match (PPM) [11], [12] are discussed. These algorithms build frequency trees and use these trees for prediction. The Active LeZi algorithm converges to the optimal tree depth if one has an asymptotically long MCS sequence [11], [13]. However, an asymptotically long sequence may not be available in a practical system. Two major reasons for this are a)UE sleep cycles due to Discontinuous Reception(DRX) and b)the fact that the MCS sequence may not remain stationary over very long time periods. Both are discussed in detail in Section III-A. In other words, one cannot assume very long sequence lengths and, a short sequence of MCS values at the eNodeB may not be enough, for Active LeZi to converge to the optimal tree depth. Furthermore, it is also difficult to implement Active LeZi, because of a growing memory requirement even if an asymptotically long MCS sequence was available. Therefore, we propose to use PPM which uses a fixed depth frequency tree [12].

However, we need to know the tree depth that must be traversed for prediction using PPM. The tree depth used must capture the complexity of the sequence and at the same time the distribution built must be accurate to the depth used, given an observed sequence length. These two requirements represent a trade-off in choosing the tree depth and the implications of this trade-off are discussed in Section IV. We propose to analyse the sequence complexity using a metric called sub-extensive information [14] and use it as an upper bound on tree depth as discussed in Section IV-A.

However, as the tree depth increases, the number of parameters in the distribution required to be estimated increases. Hence, one has to optimally pick a depth that will reflect the underlying sequence complexity, and at the same time will not involve estimation of too many parameters. We propose to use classical model order estimators such as Minimum Description Length (MDL), Akaike Information Criterion (AIC) based estimators in Section IV-B for finding the optimal tree depth, with the optimal model order being upper bounded by the  $k_{opt}^u$  (tree depth) given by analyzing the MCS sequence complexity. Since we have only a finite length MCS sequence available in a practical system, we focus on a finite sample corrected model order estimator to find the optimal tree depth for prediction  $\tilde{k}_{opt}^u$ . Note that  $k_{opt}^u$  is the optimal tree depth when the distribution is known, whereas  $\tilde{k}_{opt}^u$  is the optimal tree depth when the distribution also has to be estimated.

Once the tree depth is estimated, we can build the distribution to the desired order  $\tilde{k}_{opt}^u$  and use that for prediction. For the prediction step, a MAP estimator and a Bayesian Risk Minimizer are proposed for estimating  $X_{n+\delta}^u$  given the MCS sequence and the estimated distribution.

We compare the performance obtained using the proposed algorithms with a Markov predictor which uses a fixed model order across all users, the best scheme given in [7] and a naive algorithm which uses the feedback without any prediction whatsoever. The work in [7], uses order statistics such as mean, median, auto-correlation etc. to perform CQI/MCS prediction at the eNodeB while, we attempt to predict MCS at the eNodeB by building a temporal distribution.

It is possible that the CQI that has been reported may sometimes be in error as studied in [15]. In that work, they study the effect of bias in CQI reporting and correct it using the ACK/NACK reports <sup>5</sup> from the UE. Note that, while [15] can correct for bias in CQI reported, it is not a prediction technique and cannot efficiently solve the problem of outdated MCS. On the other hand, while we exploit the underlying MCS sequence complexity for efficient prediction, our techniques are not designed to handle CQI error. However, our method and the method in [15] can be easily combined in order to handle both CQI reporting error and the effect of outdated MCS.

TABLE I: List of Symbols used

$X_n^u$	MCS index $X$ for user $u$ received at time $n$
$S_n^u$	Sequence of MCS indices received upto time $n$
$I_{pred}(k)$	Predictive information in sequence with model order $k$ .
$k_{opt}^u$	Optimal Model order as estimated using $I_{pred}(k)$
$\tilde{k}_{opt}^u$	Optimal Model order when the distribution is unknown.

## II. SYSTEM MODEL

A 19 cell, 3 sectors per cell reuse-one LTE system is considered. In the system simulator, there are 19 cells and 57 sectors with wrap around, to avoid edge discontinuities [16] and UEs are distributed uniformly in each sector. LTE systems, use OFDMA in the physical layer where sub-carriers are grouped into sub-bands [17], and users are allocated a set of sub-bands for data

<sup>5</sup>These indicate whether a packet has been received successfully or not.

transmission. Each eNodeB transmits over the same set of resources, as, it is a reuse-one system. The OFDMA for the 10MHz LTE system has 1024 sub-carriers where only the 600 in the middle are used [17]. These 600 sub-carriers are grouped into 50 groups of 12 sub-carriers (SCs) each and this is done over 14 OFDM symbols. So this group of 12 SCs over 14 symbols is called one Physical Resource Block (PRB) and the 14 OFDM symbols together constitute a sub-frame [17]. There are 50 PRBs in a sub-frame and a continuous block of 3PRBs are grouped to form a sub-band. There are 17 sub-bands in LTE for the 10MHz system [17], and, scheduling and transmission is done at the sub-band level. The frame structure is provided in [18]. The set of sub-bands allocated to a user, is called a transport block and every user will be allocated one rate for the whole transport block.

There are multiple feedback techniques in LTE and here we focus on periodic feedback, where the user combines the best five sub-bands' rates and feeds back this aggregated CQI index along with the sub-band location. This estimation of the aggregated CQI is highly UE specific *i.e.*, different UEs are manufactured by different vendors and consequently, the algorithms used may vary. At the eNodeB these CQI values are converted into MCS values. Hence, our data comprises of the MCS sequences for all the users in the system. We use a full system simulator to obtain the data *i.e.*, MCS sequences for each UE used for prediction. Both, path loss exponent and shadow fading parameters are as specified in [19], [20] for an Urban Macro model. The channel model used in the simulator is the Generic Channel model as given in [19], [20]. The generic channel model is a realistic channel model for multipath channels in cellular systems. The model is such that the channel from each UE to each eNodeB is modeled using different parameters such as Angle of Arrivals and Departures of the multipath rays, distance dependent power delay profile, Line of Sight parameters and multipath profiles [19], [20]. Hence, different users see different delay spreads and even the same user sees different delay spreads from different eNodeBs *i.e.*, the multipath power delay profile of the channel between the UE and serving eNodeB can differ from the power delay profile between the UE and interfering eNodeBs. This makes a simple statistical characterization of the channel for the purposes of modeling the SINR or rate extremely difficult. Even if, one were to characterize the channel, it is to be done for all the users, and the different links between eNodeBs and UEs, making it an extremely complex system to model mathematically. Note that only the strongest 8 interferers to each user, are modeled explicitly for ease of computation. The detailed simulation parameters are given in Table II for completeness.

TABLE II: Baseline Simulation Parameters

Deployment scenario	Urban macro-cell scenario
Base station antenna height	25 m, above rooftop
Minimum distance between UT and serving cell	$\geq 25m$
Layout	19-cell Hexagonal grid with wrap around.
carrier frequency	2 GHz
Inter-site distance	500 m
UT speeds of interest	30 km/h
Total eNodeB transmit power	46 dBm for 10 MHz
Thermal noise level	-174 dBm/Hz
User mobility model	Fixed and identical speed $ v $ of all UTs, randomly and uniformly distributed direction
Inter-site interference modeling	Explicitly modeled
UT antenna gain	0 dBi
Channel Model	Urban Macro model (UMa)
Network synchronization	Synchronized
Downlink transmission scheme	1x2 Single Input Multiple Output
Downlink Scheduler	Proportional Fair with full bandwidth allocation
Downlink Adaptation Wideband CQI	sub-band Channel Quality Information (CQI) of best 5 bands for each user and for all users, at 5 ms CQI feedback periodicity, CQI delay :Ideal, CQI measurement Error: none, MCS based on LTE transport formats
Evaluated traffic profile	Full Loading and Partial loading with exponential inter-arrival time.
Simulation bandwidth	10 + 10 MHz (FDD)

The eNodeB requests MCS feedback from each user once in every  $\delta$  frames (typically  $\delta=5ms$ ), some more details are given in Table II. Since the set of MCS values are 28, this corresponds to rates varying from 0.1523 - QPSK with code rate 0.076, to 5.5547 - 64 QAM code rate 0.93, bits per symbol [1] seen in Table 10.1. The sequence received looks like  $X_{\delta}^u, X_{2\delta}^u, \dots, X_n^u, X_{n+\delta}^u, \dots$ , where the eNodeB at time instant  $n+i$  ( $i < \delta$ ) has to use a value  $X_n^u$  which was estimated at time  $n$ . As discussed earlier, there are two main reasons for  $X_{n+i}^u$  to vary from  $X_n^u$  and they are a) Mobility in the system and b) The active set of interfering eNodeBs will change.

We simulate the following traffic profiles:

- A generalized traffic distribution with exponential inter-arrival rate of 50ms and packet size 3000 bytes. (partial loading)



- A situation where all eNodeBs transmit continuously. (full loading)

To summarize, we are required to estimate, a time varying discrete value of rate, for partial and full loading. There are 57 eNodeBs with each eNodeB running scheduling algorithms independent of the other eNodeBs. These users can be scheduled over different bands, at different times, and the interfering and desired channel also changes over time. The above explained model is difficult to completely characterize mathematically and analyze, because, to do that we have to model the scheduler behavior under traffic, all the user-interferer channels which are not i.i.d and even time-varying traffic statistics. However, if one knows the joint temporal rate distribution of a user, one could predict the rate from the observed sequence. Since, the sequence to be predicted is from a discrete set, we propose to use discrete sequence prediction algorithms.

### III. COMPRESSION ALGORITHMS FOR MODEL BUILDING

In the previous sections, we explained how the MCS prediction problem for each UE could be mapped to a discrete sequence prediction problem for which a joint temporal distribution of the sequence has to be built. This problem of building a discrete distribution has been studied extensively in [11], [12], [21], [22] and we propose to apply these techniques for MCS prediction with appropriate modification. We now give algorithms, which build frequency trees, and from which the discrete distribution can be estimated.

#### A. Active LeZi

The Active LeZi builds a variable order Markov chain as proposed in [11]. This is shown in Algorithm 1. This algorithm uses a sliding window to update its contexts as will be explained in an example. We denote current window by  $W$  its length by  $W_L$  and maximum allowed window length by  $W_{Lmax}$ , the dictionary by  $D$  and current word as  $w$ .

This algorithm generates a frequency tree for  $S' = 22, 22, 22, 22, 22, 27, 27, 24, 24, 22, 24, 27, 24, 24, 22$  as in Fig. 1 and we provide an illustrative example on its working as follows: (i)

- 1) Initialization,
  - a)  $W_L = 0$ ;
  - b)  $W = \emptyset$ ;
  - c)  $D = \emptyset$
  - d)  $w = \emptyset$

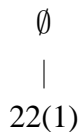
---

**Algorithm 1** Active LeZi Algorithm
 

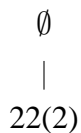
---

- 1:  $W_L = 0, W = \emptyset, D = \emptyset$
  - 2: Assign  $w = \emptyset$
  - 3: Append incoming character  $v$  to  $w$  and  $W$ , i.e.,  $W = (W, v), w = \{w, v\} W_L = W_L + 1$
  - 4: If  $w$  is part of  $D$  do not add  $w$  to  $D$ .
  - 5: If  $w$  is not part of dictionary add  $w$  to the dictionary  $D = D, w$ .
  - 6:  $W_{L_{max}}$  = Maximum word length in dictionary
  - 7: If  $W_L > W_{L_{max}}$  delete  $W[0]$
  - 8: Update frequency tree based on all contexts in the  $W$ .
  - 9: Repeat from Step 2
- 

- 2) Getting incoming character:  $v = 22$
- 3) From Step 3:  $w = \{22\}, W = \{22\}, W_L = 1$  and since  $(w \notin D)$
- 4) From Step 5:  $D = \{\{22\}\}$ ,
- 5) From Step 6 :  $W_{L_{max}} = 1$
- 6) Step 7:  $W_L \not> W_{L_{max}}$
- 7) Step 8: Update tree based on  $W$  as follows:

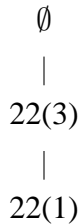


- 8) Step 9 - Repeating from Step 2:  $w = \emptyset$ ,
- 9) Getting incoming character:  $v = 22$
- 10) From Step 3:  $w = \{22\}, W = \{22, 22\}, W_L = 2$  and  $(w \in D)$
- 11) Step 6:  $W_{L_{max}} = 1$
- 12) Step 7:  $W_L > W_{L_{max}}$ : Delete  $W[0]$  thus obtaining  $W = \{22\}$
- 13) Step 8: Update tree based on  $W$  as follows:



- 14) Step 9 - Repeating from Step 2:  $w = \emptyset$ ,

- 15) Getting incoming character:  $v = 22$
- 16) Step 3:  $w = \{22, 22\}$ ,  $W = \{22, 22\}$ ,  $W_L = 2$  and  $(w \notin D)$
- 17) From Step 5:  $D = \{\{22\}, \{22, 22\}\}$
- 18) From Step 6:  $W_{L_{max}} = 2$
- 19) At Step 7:  $W_L = W_{L_{max}}$
- 20) In Step 8: Update tree using  $W$  as follows:



- 21) Repeat for whole sequence.

The full tree for the above example is shown in Fig. 1.

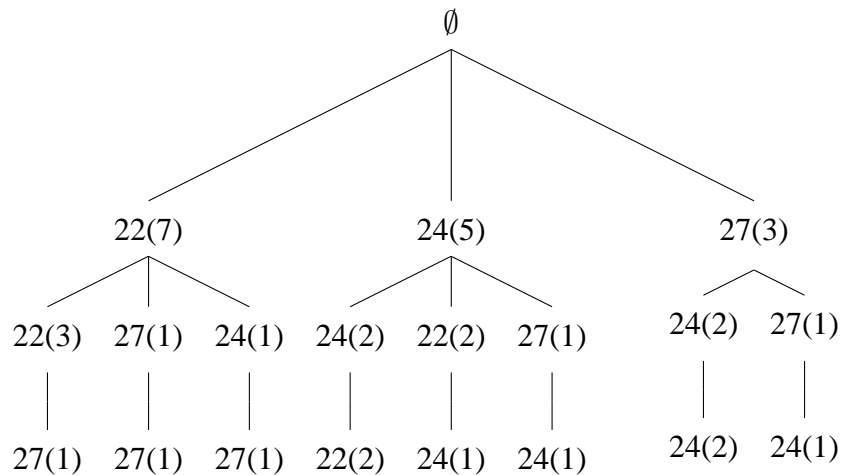


Fig. 1: Active LeZi Example Tree

The nodes in the tree in Fig. 1 gives information about the MCS index and the number of times it has occurred in a certain MCS sub-sequence. For example, in Fig.1 if one looks at the left most node in the bottom most generation a value 27(1) is seen. This implies that the subsequence  $\{22,22\}$  has been followed by a  $\{27\}$  i.e.,  $\{22,22,27\}$  has occurred once and from the parent of that node  $\{22,22\}$ , has occurred thrice , and  $\{22\}$  itself has occurred seven times.

However, this algorithm suffers from certain implementation difficulties. The maximal word

length in this algorithm grows with sequence length, thereby, requiring an ever-increasing memory to store the words and frequency trees. Since the channel correlations are typically of the order of only a few milliseconds, the correlations in the MCS sequences does not extend much in time and, it is unnecessary to learn very long contexts to predict.

Furthermore, this predictor converges to the optimal model order only asymptotically [13]. However, due to the effect of UE sleep cycle, we would never see an asymptotically long sequence to learn the data [1]. In order to save battery, when the user is idle it stops measuring/sensing the channel and hence there is no feedback during this time. There are two types of sleep cycles viz. short DRX or long DRX. First, the UE senses the control channel, to know, if there is any data to be received and if there is no data to be received it goes into a short sleep cycle, where the UE does not sense the channel or feedback MCS. Then, it again senses the channel at the end of the short DRX and if there is still no data it goes for another short DRX and after  $N$  such short DRX, if there is no data the UE goes into long DRX. The length and duration of short and long DRX and  $N$  are configurable, and are configured according to traffic type that the UE is receiving. Furthermore, the assumption of stationarity may not hold over very long time periods or sequence lengths. Hence in a practical system, one has to assume that the sequence length is limited.

Since Active LeZi requires a high amount of memory and also an asymptotically long sequence, both of which are not practical, we propose to use the PPM method of a fixed tree depth with appropriate modifications.

### *B. Prediction by Partial Match*

Most online predictors are based on the short memory principle, in which the recent past is more important for prediction i.e. prediction is done by observing the previous  $k$  symbols. Here, we plan to build a frequency tree of fixed depth  $k^{max}$  which may depend on the sequence length available. The PPM uses the Active LeZi algorithm with the  $W_{L_{max}}$  fixed to some  $k^{max}$ . Now using PPM, with fixed tree depth  $k^{max}$ , one can compute all models of the form  $P(X_n^u | X_{n-\delta}^u \dots X_{n-k\delta}^u)$  with  $k = 1, \dots, k^{max} - 1$ . Note that, while one can build the tree upto depth  $k^{max}$ , the depth used for prediction can be different. This depth used for prediction will depend on the sequence complexity and the number of parameters one needs to estimate to learn the distribution (details given in Section IV). The example tree given in Fig. 1 has  $k^{max} = 3$

and from this tree, the models  $P(X_n^u|X_{n-\delta}^u)$  and  $P(X_n^u|X_{n-\delta}^u, X_{n-2\delta}^u)$  can be computed and either of them can be used for prediction.

### C. Estimation of $P(X_n^u|X_{n-\delta}^u \dots X_{n-k\delta}^u)$ using the Frequency Trees

Using the techniques presented above, Markov models upto order  $k^{max} - 1$  can be built. In order to use a  $k$ th order model to predict, each state needs to be assigned a probability of occurrence, given the model and previous  $k$  states. This has to be done using the models of order 1 to  $k$  which are recursively built. This recursion is because even if a  $k$ th order model returns the probability of a particular state as zero, there might be a lower order context in which the state could have occurred. For instance, if one looks at the example sub-sequence given in Section III-A and its corresponding tree in Fig. 1, from the second order model alone, the next value being 22 is zero because, 24,22 has never been followed by a 22. However, if one looks at the first order model, 22 has succeeded a 22, 3 out of 7 times. Therefore, the information upto depth  $k + 1$  must be blended to give the probability of occurrence of a state under model order  $k$ . Typical blending methods are given in [11], [23]. Given the frequencies of all contexts and given that the previous  $k - 1$  alphabets were  $X_n^u \dots X_{n-k+2}^u$  then the probability that the next state is  $X_{n+\delta}^u = t_i$  is given by a recursive computation.

$$P_0(X_{n+\delta}^u = t_i) = \frac{\sum_{i=1}^n \mathbf{1}(X_i^u = t_i)}{n} \quad (1)$$

$$\begin{aligned} P_k(X_{n+\delta}^u = t_i) &= P(X_{n+\delta}^u = t_i | X_n^u, \dots, X_{n-(k-1)\delta}^u = t_{j_1} \dots t_{j_k}) \\ &= \frac{\sum_{i=1}^n \mathbf{1}(X_{(i+k)\delta}^u, \dots, X_{i\delta}^u = t_j \dots t_{j_k})}{\sum_{i=1}^n \mathbf{1}(X_{(i+(k-1))\delta}^u, \dots, X_{i\delta}^u = t_{j_1} \dots t_{j_k})} \\ &\quad + P_{k-1}(X_{n+\delta}^u = t_i) \cdot \left( 1 - \frac{\sum_{t_j} \sum_{i=1}^n \mathbf{1}(X_{(i+k)\delta}^u \dots X_{i\delta}^u = t_j \dots t_{j_k})}{\sum_{i=1}^n \mathbf{1}(X_{(i+(k-1))\delta}^u \dots X_{i\delta}^u = t_{j_1} \dots t_{j_k})} \right) \end{aligned} \quad (2)$$

where  $\mathbf{1}$  is the indicator function, indicating the occurrence of the event, and,  $\sum_{i=1}^n \mathbf{1}(X_{(i+k)\delta}^u, \dots, X_{i\delta}^u = t_j \dots t_{j_k})$  is the frequency of occurrence of the sequence  $\{t_{j_k}, t_{j_{k-1}} \dots t_{j_1}, t_j\}$  where  $n$  is the sequence length that has been observed. As an example let us use the tree given in Section III-A to compute the probability that the next value of the sequence  $S'$  is 24.

The last seen values are 24,22 . The number of times 24,22,24 has occurred given 24,22 has occurred is 1 and the number of times that 24,22 has occurred is 2. The number of times 24,22 has occurred with no future stored context is also 1 which is the second term in (2). This is the proba-

bility by which the lower order model is weighed. Therefore  $P(24|24, 22) = \frac{1}{2} + (1 - \frac{1}{2})P(24|22)$  and  $P(24|22) = \frac{1}{7}$ . Thus, the probability that  $P(24|24, 22) = \frac{1}{2} + (1 - \frac{1}{2})\frac{1}{7} = \frac{4}{7}$

To summarize this section, we saw three algorithms which built frequency trees and a method to evaluate the  $k$ th order probability. It can be seen that, to build a  $k^u$ th order model for user  $u$  viz.  $P(X_n^u | X_{n-k^u+1}^u, X_{n-k^u+2}^u, \dots, X_{n-1}^u)$ , one must use the data upto depth  $k^u + 1$  from the tree. Our next problem is finding out, the optimal  $k^u$  that can used for prediction for each user  $u$  (different users can have different values of  $k^u$ ) called the **model order** selection problem. In the next section, we shall discuss the model order problem in detail and propose methods to find the optimal order.

#### IV. MODEL ORDER SELECTION BASED ON SEQUENCE COMPLEXITY AND AIC

The algorithms which built frequency trees and evaluated probabilities using them were discussed in detail in the previous section, and now we want to find out the depth of the tree upto which one has to traverse, to obtain a 'reasonable model'.

A model used for prediction must satisfy two properties:

- The model used must capture the complexity of the sequence.
- The frequency tree built, must be 'reasonably' accurate to the required depth, given an observed sequence length.

The first property is intrinsic to the sequence, i.e. a sequence comes from a particular distribution  $P(X_{(N-k+1)\delta}^u, X_{(N-k+2)\delta}^u, \dots, X_{(N-k+i)\delta}^u \dots X_{N\delta}^u)$  such that given the previous  $k^u - 1$  values, any knowledge of values further in the past does not improve the prediction accuracy. The second property arises due to the fact that the distribution is being estimated, and with increasing  $k^u$ , the number of parameters to be estimated increase and to estimate a large number of parameters a correspondingly large sequence must be observed. In other words, if the model that best fits a given sequence is  $k^*$ , it could be that the number of parameters to be estimated for building a  $k^*$  model might be so large that estimating the required parameters accurately from a fixed length MCS sequence may not be possible. Hence, the optimal model order is that, which achieves the right balance, in the trade-off between, finding a model which is complex enough to capture the sequence complexity, but not so complex that it requires a large number of parameters to be estimated. These two properties are explained in detail in the next subsections. For the sake of notational simplicity, we henceforth drop  $\delta$  from the subscript i.e.,  $X_{i\delta}^u = X_i^u$

### A. Sub-Extensive Information as a metric for Sequence Complexity

We first focus on a metric which characterizes the underlying complexity/ learnability/ predictability of a sequence called sub-extensive information [14]. We had mentioned earlier that, sequence prediction is similar to source encoding and hence, it is only natural that, we study the model order through complexity and entropy of the sequences. The absolute entropy of a sequence increases with volume per se because complexity scales with volume [24]. Since, sequence prediction involves predicting the future, having observed the past, one is more interested in the mutual information between the past and the future than the absolute entropy. This mutual information is also called sub-extensive information or predictive information in sequence prediction literature in physics [14]. The total information/entropy in a sequence is a sum of extensive and sub-extensive information components. The total entropy at time  $n$  is given by:

$$H(X_{total}) = H(X^u_1, X^u_2, X^u_3, \dots, X^u_n) \quad (3)$$

$$= H(X^u_n | X^u_{n-1} \dots X^u_1) + H(X^u_1, X^u_2, X^u_3, \dots, X^u_{n-1}) \quad (4)$$

The first term on the RHS of (4) is the sub-extensive component and the second term is the extensive component of entropy. It can be seen that, as  $n \rightarrow \infty$  the total entropy and the extensive component will tend to infinity linearly with  $n$ , while the sub-extensive component will grow at a less than linear rate. The average sub-extensive/mutual information is given by:

$$I(X^u_n, (X^u_1, X^u_2, X^u_3, \dots, X^u_{n-1})) = \left\langle \log_2 \left( \frac{P(X^u_n | (X^u_1, X^u_2, X^u_3, \dots, X^u_{n-1}))}{P(X^u_n)} \right) \right\rangle \quad (5)$$

where,  $\langle \rangle$  denotes expectation over the joint distribution,  $P(X_1 \dots X_n)$ . Another way of writing this is:

$$\begin{aligned} I(X^u_n, (X^u_1, X^u_2, X^u_3, \dots, X^u_{n-1})) &= H(X^u_n) + H(X^u_1, X^u_2, X^u_3, \dots, X^u_{n-1}) \\ &\quad - H(X^u_1, X^u_2, X^u_3, \dots, X^u_n) \end{aligned} \quad (6)$$

$$I(X^u_n, (X^u_1, X^u_2, X^u_3, \dots, X^u_{n-1})) = H(X^u_n) - H(X^u_n | X^u_1, X^u_2, X^u_3, \dots, X^u_{n-1}) \quad (7)$$

Calculating the sub-extensive part of information requires the knowledge of joint probability distributions. This sub-extensive component of information, is also called predictive information

and is denoted as:

$$I_{pred}(T, T') = \left\langle \log_2 \left( \frac{P(X^u_{future} | X^u_{past})}{P(X^u_{future})} \right) \right\rangle \quad (8)$$

where  $T$  is the time for which the sequence has been observed in the past and  $T'$  is the future time for which the sequence is to be predicted. Computing the  $I_{pred}(T, T')$  as in (8) requires the knowledge of the joint distribution of the entire sequence. However, in practical systems one may not have the complete joint distribution of  $\{X^u_n, X^u_{n-1} \dots X^u_1\}$  and due to memory constraints, it will be possible to estimate and use only the joint distribution of  $\{X^u_n, X^u_{n-1} \dots X^u_{n-k}\}$ .

In our problem the focus is on finding the best  $k^u$ -th order Markov model for each user  $u$ , to use in PPM for prediction, and the predictive information in a sequence while using a model of order  $k$  is denoted by  $I_{pred}(k)$ . The value of  $k$  can be varied from 1 to  $K$  and  $I_{pred}(k)$  can be obtained as follows:

$$I_{pred}(k) = \left\langle \log_2 \left( \frac{P(X^u_n | (X^u_{n-1} \dots X^u_{n-k}))}{P(X^u_n)} \right) \right\rangle \quad (9)$$

$$= H(X^u_n) - H(X^u_n | (X^u_{n-1} \dots X^u_{n-k})) \quad (10)$$

Since, the sequence that we are studying is a sequence of MCS indices and the dependence on the past is of a decreasing nature i.e.  $X^u_n$  to ‘depends more’ on  $X^u_{n-k}$  than  $X^u_{n-(k+1)}$ , where  $k > 0$ , we can expect  $I_{pred}(k)$  as a function of  $k$  to grow at a rate slower than linear increase.  $I_{pred}(k)$  will be monotone non-decreasing in  $k$  because the mutual information is not going to decrease as the number of observations increase. As, the number of observations used for prediction increases i.e. between using  $k$  past values and using one more value in the farther past can only either increase, or retain the existing information about the future. For  $I_{pred}(k)$  to have a linear growth rate it would require  $X^u_n$  to ‘depend equally’ on  $X^u_{n-l}$  and  $X^u_{n-(l+1)}$  which will not happen, because, both desired and interference channel correlations decrease over time and the MCS sequence depends on both. Sub-linear rate of increase can mean either a rate of increase of  $O(k^\alpha)$  where  $\alpha < 1$  or a rate of increase of  $O(\log(k))$ . Another possibility is that the sub-extensive information is constant despite increasing the number of observations. This can happen when the underlying process is a simple Markov process. While trying to predict a simple Markov process it is enough that we observe the immediate past, i.e.,  $X^u_{n-1}$  [24]–[26].



1) *Sub-Linear  $O(k^\alpha)$  rate of increase:* The generalized form  $I_{pred}(k)$ , is [14]:

$$I_{pred}(k) = C_0 + C_1 k^\alpha \quad (11)$$

$$L(k) = I_{pred}(k) - I_{pred}(k-1) \quad (12)$$

$$L(k) \approx \frac{\partial I_{pred}(k)}{\partial k} = \alpha C_1 k^{\alpha-1} \quad (13)$$

where  $0 < \alpha < 1$ . The term  $L(k)$  is called the learning curve, and is a metric which gives the rate at which the predictive information increases when the model order is increased, and this is a decreasing function in  $k$  from (13). This implies that increasing  $k$  more and more gives only diminishing returns in prediction performance. A sub-linear rate of increase as shown in (13), implies that the number of parameters to be learnt for predicting the sequence is infinite [14]. In the problem studied here, since the sequence to be predicted itself is discrete, only finite parameters will be required to be estimated and hence, sub-linear increase will never be seen.

2) *Logarithmic  $O(\log(k))$  rate of increase:* The generalized form  $I_{pred}(k)$ , is [14]:

$$I_{pred}(k) = C_0 + C_1 \log(k) \quad (14)$$

$$L(k) = I_{pred}(k) - I_{pred}(k-1) \quad (15)$$

$$L(k) \approx \frac{\partial I_{pred}(k)}{\partial k} = \frac{C_1}{k} \quad (16)$$

A log-rate of increase in predictive information implies that the number of parameters to be estimated is finite [14]. The MCS sequences can at most have only a logarithmic rate of increase, since in predicting discrete sequences, it is required to predict only a finite number of parameters to characterize these sequences.

We now compute the  $I_{pred}(k)$  for all the users and a few users' behaviour is captured in Fig. 2. This computation is performed by empirically averaging the term  $\log_2 \left( \frac{P(X_n^u | (X_{n-1}^u \dots X_{n-k}^u))}{P(X_n^u)} \right)$  as shown in (9). The results seem to show a logarithmic behaviour, but, instead of continuously diverging the  $I_{pred}(k)$  saturates at a constant value. This can be understood better by looking at (10). The value of  $H(X_n^u | X_1^u, X_2^u, X_3^u, \dots, X_{n-k}^u)$  is bounded from above by  $H(X_n^u)$  and below by 0 and  $H(X_n^u)$  itself is bounded above by  $\log(p)$  where  $p$  is the number of possible states that  $X_n^u$  can take [25]. This is expressed concisely as:

$$0 \leq H(X_n^u | X_{n-1}^u \dots X_{n-k}^u) \leq H(X_n^u) \leq \log(p) \quad (17)$$

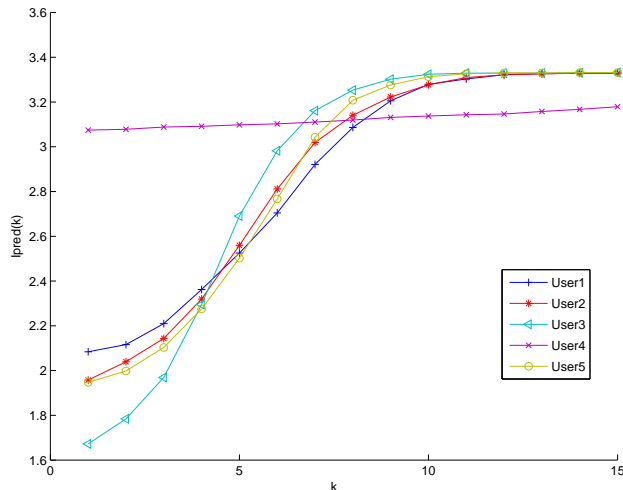


Fig. 2: Plot of  $I_{pred}(k)$  as a function of  $k$

From (10) and (17) it is apparent that:

$$0 \leq I_{pred}(k) \leq \log(p) \quad (18)$$

It can be argued that, by picking a value of  $k$  for which  $I_{pred}(k)$  achieves its maximum possible value would give us an optimal prediction performance. However, the distribution is unknown to us and, as  $k$  increases, the number of parameters needed to estimate the unknown distribution also increase and hence, the  $I_{pred}(k)$  that has been computed may not be accurate given the sequence of limited length. For example, in Fig. 2, despite the sequence of User 4 having only a slowly increasing value of  $I_{pred}(k)$  when compared to the other users, it is the sequence that has the best prediction performance. This is because, User 4 requires only a simple Markov model to predict its sequence, and it is significantly easier to estimate the parameters of a simple Markov model as compared to estimating a model of order 4. However, one can use the sub-extensive information to find out the maximum possible model order where the gains are substantial *i.e.*, the maximum model order  $k_{opt}^u$  can be found out as:

$$k_{opt}^u = \max(k) : L(k) > \epsilon \quad (19)$$

where  $\epsilon$  is chosen such that, the gains obtained in increasing the model order beyond  $k_{opt}^u$  is not significant. For instance the User 4, will have  $k_{opt}^u = 2$ .

The  $k_{opt}^u$ , as calculated here is optimum if the true distribution is known to us a priori. However, we have to estimate/learn the distribution and, as  $k_{opt}^u$  of a given user increases, the number of parameters required to be estimated in order to learn the distribution increase. The effect of estimating a large number of parameters, on model order is discussed in the next section. We use the  $k_{opt}^u$  obtained in the current section as an upper bound on the optimal model order when the distribution is to be estimated.

### B. Optimal Model Order when the distribution is to be estimated

Now, we are to fit a model order given the sequence and the distribution estimated from the sequence. The model order fitting problem is approached as a hypothesis testing problem, where  $\mathcal{H}_i$  is the hypothesis that the  $i$ th order Markov chain best fits the sequence. Then, the optimal value of  $i$  denoted by  $\tilde{k}_{opt}^u$  can be found out by maximizing information theoretic criteria such as Minimum Description Length (MDL) or Akaike Information Criteria (AIC) [13], [27], [28]. In these methods, the usual technique followed is to maximize the likelihood of the observations given the hypothesis, with a penalty on the number of parameters to be estimated. In the problem considered, the observation is the MCS sequence  $S_n^u = \{..X_m^u, X_{m+\delta}^u \dots X_n^u\}$  observed for each user  $u$  and the number of parameters is the number of distribution parameters to be estimated. We are interested in building a discrete probability distribution of  $i$  length sequences. The parameters required for building such a distribution is denoted by  $\theta_i$  where  $i$  is the model order and the cardinality of  $\theta_i$  is  $n_i^u$ , which is the number of parameters to be estimated.  $\theta_i$  is the  $i$ th order distribution itself. For example, in our scheme, to estimate the distribution  $P(X_{n+\delta}^u)$ , since there are 28 MCS values one needs to estimate  $28 - 1$  probabilities. To estimate  $P(X_{n+\delta}^u | X_n^u)$ , one must estimate a transition probability matrix of size  $(28 - 1)28$ . By induction, this logic can be extended to an  $i$ th order model and the number of parameters would be  $(28 - 1)28^{i-1}$ . To generalize, if one had to estimate a  $k$ th order Markov Model for an  $m$  state process, then  $(m - 1)m^{k-1}$  parameters would have to be estimated. We use the value obtained from our  $I_{pred}(k)$  calculations to determine the maximum possible model order  $k_{opt}^u$  for user  $u$  and use it as an upper bound on the model order to be determined.

The model order problem can be set-up as a multiple hypothesis testing problem as follows:

- $\mathcal{H}_1$  : Hypothesis that  $\tilde{k}_{opt}^u = 1$

- $\mathcal{H}_2$  : Hypothesis that  $\tilde{k}_{opt}^u = 2$
- $\vdots$
- $\mathcal{H}_{k_{opt}^u}$  : Hypothesis that  $\tilde{k}_{opt}^u = k_{opt}^u$

In usual hypothesis testing problems, the likelihood function of the observations given the hypothesis is found out and the hypothesis that maximizes the likelihood function is taken to be the true hypothesis. However, when the hypotheses are models of an increasing order, this technique fails because, the lower order models are always nested within the higher order models [29]. Since, we know that the error in estimating the parameters of a higher order model will also impact the performance of a system, we look at a cost function which picks a model that provides a trade-off between maximizing the likelihood and minimizing the error variance of the parameters to be estimated.

Therefore, we propose to use the Generalized Maximum Likelihood Estimator (GMLE) in [29] which tries to maximize the following cost function:

$$\xi_i^u = \ln(P(\mathbf{S}_n^u; \hat{\boldsymbol{\theta}}_i | \mathcal{H}_i)) - \frac{1}{2} \ln(\det(\mathbf{I}(\boldsymbol{\theta}_i))), \quad 1 \leq i \leq k_{opt}^u, \quad (20)$$

where the first term in (20) is the log-likelihood function and the second term is the penalty due to errors in model where  $\mathbf{I}(\boldsymbol{\theta}_i)$  is the Fisher information matrix of  $\boldsymbol{\theta}_i$ , and its inverse is the lower bound on the error covariance matrix in estimating  $\boldsymbol{\theta}_i$ , where  $\boldsymbol{\theta}_i$  is a vector of distribution parameters which are to be estimated and its cardinality is  $n_i^u$ . This set of estimates is denoted by  $\hat{\boldsymbol{\theta}}_i$  where  $\hat{\boldsymbol{\theta}}_i$  is the ML estimate of  $\boldsymbol{\theta}_i$ .

When  $i$  increases, the first term in (20) *i.e.*, the log-likelihood function increases while in the second term, because the number of parameters to be estimated increases, the  $\det(\mathbf{I}(\boldsymbol{\theta}_i))$  increases. Therefore, maximizing the above equation with respect to  $i$  ensures that, a model is chosen by optimally trading off, model likelihood with model parameter estimation error.

$$\tilde{k}_{opt}^u = \arg \max_i (\xi_i^u). \quad (21)$$

However, to implement the above solution one must know  $\mathbf{I}(\boldsymbol{\theta}_i)$ . That involves knowing the probability distribution function a priori. However, in our case the parameters to be estimated are the probabilities themselves. Therefore, instead of trying to estimate  $\mathbf{I}(\boldsymbol{\theta}_i)$ , the determinant  $\det(\mathbf{I}(\boldsymbol{\theta}_i))$  can be approximated as  $cN^{n_i^u}$  as in [29]. This is equivalent to MDL as in [13] and [29].

$$MDL_i^u = -\ln(P(\mathbf{S}_n^u; \hat{\boldsymbol{\theta}}_i | \mathcal{H}_i)) + \frac{n_i^u}{2} \ln(N), \quad 1 \leq i \leq k_{opt}^u. \quad (22)$$

The optimal model is obtained as:

$$\tilde{k}_{opt}^u = \arg \min_i (MDL_i^u). \quad (23)$$

Another option is to use the AIC which is given follows:

$$AIC_i^u = -2\ln(P(\mathbf{S}_n^u; \hat{\boldsymbol{\theta}}_i | \mathcal{H}_i)) + 2n_i^u, \quad 1 \leq i \leq k_{opt}^u. \quad (24)$$

Here again the optimal model order is obtained as:

$$\tilde{k}_{opt}^u = \arg \min_i (AIC_i^u). \quad (25)$$

AIC is an efficient model order estimator, while, MDL is a consistent estimator [30]. However, both AIC and MDL assume that the number of observations is asymptotically large *i.e.*,  $n \gg n_i^u$  [30], [31].

However, we have only finite length data sequences, and  $n_i^u$  grows nearly exponentially in  $i$ . Therefore we use a sample corrected AIC *i.e.*,  $AIC_C$  which is given as follows [30], [31] :

$$AIC_{C_i}^u = -2\ln(P(\mathbf{S}_n^u; \hat{\boldsymbol{\theta}}_i | \mathcal{H}_i)) + 2n_i^u + \frac{2n_i^u(n_i^u - 1)}{N - n_i^u - 1}, \quad 1 \leq i \leq k_{opt}^u, \quad (26)$$

$$\tilde{k}_{opt}^u = \arg \min_i (AIC_{C_i}^u). \quad (27)$$

The sample corrected AIC is derived in detailed in [32]. It can be seen that the sample corrected AIC tends to the asymptotic AIC as  $N \rightarrow \infty$ . Also, this criterion ensures that, one does not pick a higher order model initially when the sequence length is small.

Summarizing, we have proposed usage of finite sample model order determination methods to find the best model to be used in our PPM algorithm for predicting the sequence for a given user  $u$ . This is to be done for all user sequences as different sequences will have different complexity. In a system like LTE there are 28 MCS values that can occur. Therefore, to build a model of order  $i$ , it seems that one has to estimate nearly  $28^i$  probabilities for all possible sequences. However, a user  $u$  will not see all the MCS indices, in the short time frame, that we look at for sequence prediction. For instance, a user that sees MCS index 1 corresponding to rate 0.15 cannot see MCS 28 corresponding to rate 5.55 within a time frame of few seconds or

even between two sleep cycles. It may be that, a user sees only  $m_u$  MCS indices. The value of  $m_u$  is estimated from the frequency tree. For instance, consider the tree given in Section III-A. Since the only values observed in the sequence  $S$  for building the tree was 22,24,27 the value of  $m_u$  will be estimated as 3. Thus for a given user  $u$ , finally the model order is estimated by minimizing the cost function given below.

$$AIC_C(i^u) = -2\ln(P(\mathbf{S}_n^u; \hat{\theta}_i | \mathcal{H}_i)) + 2(m_u - 1)(m_u)^{i-1} + \frac{2(m_u - 1)(m_u)^{i-1}((m_u - 1)(m_u)^{i-1} - 1)}{N - (m_u - 1)(m_u)^{i-1} - 1} \quad 1 \leq i \leq k_{opt}^u, \quad (28)$$

and the optimal model order is given by:

$$\tilde{k}_{opt}^u = \arg \min_i AIC_C(i^u). \quad (29)$$

We have observed that when  $k_{opt}^u$  is 4,  $\tilde{k}_{opt}^u$  can vary from 1 to 4.

## V. PREDICTION ALGORITHMS USING THE ESTIMATED DISTRIBUTION

The model order obtained in the previous sections can be used in the PPM algorithm to fix the tree depth for prediction and the probabilities  $P(X_{n+\delta}^u | S_n^u)$  can be calculated using the (1) and (2). We now propose two prediction algorithms.

### A. MAP Estimator

The Maximum A Posteriori (MAP) estimator is an estimator that maximizes the a posteriori probability of an event given the observations *i.e.*, it picks that value which is the most likely given that the past has been observed. The MAP estimator for MCS index given the sequence observed is as follows:

$$\hat{X}_{n+1}^u = \arg \max_i P(X_{n+1}^u = i | X_n^u \dots X_{n-\tilde{k}_{opt}^u}^u) \quad (30)$$

where  $X_{n+1}^u$  is the next state which we want to predict and  $i$  s are the possible values taken by the MCS. This technique will result in maximum prediction accuracy. However, since it is optimized only for prediction accuracy, it treats all errors equally *i.e.*, estimating a rate higher than the true rate is same as estimating a lower rate. However, in the rate prediction problem, if the predicted rate is lower than the true rate, the transmission at the predicted rate will still be a success at the cost of a loss in efficiency whereas, if the predicted rate is higher

it will result in a packet loss. The MAP estimator is oblivious to this effect and therefore, will not be throughput optimal despite its prediction optimality. For instance, given a sequence  $S$ , if there are 3 rates  $r_1 < r_2 < r_3$  which are possible future candidates with probabilities  $P(r_1) = 0.3, P(r_2) = 0.3, P(r_3) = 0.4$ , then the MAP estimator will pick  $r_3$ . Now, based on the observed data, there is approximately 60% probability that  $r_3$  was a wrong prediction resulting in packet loss. If the rates  $r_1, r_2$  comparable to  $r_3$ , one could have chosen the lower rates  $r_1$  or  $r_2$ , thus decreasing the risk of packet loss. The next section proposes a method of predicting rate given the issues of packet loss and throughput efficiency.

### B. Bayesian Risk based Estimator

In this technique, a cost is assigned to the event of predicting a state and the state which has the minimum cost is picked. There are numerous ways of assigning the costs, and the cost assignment is done in order to enable the picking of the highest possible rate without resulting in failed transmission. The cost assignment used is as follows:

- If predicted rate is greater than the true rate then we lose the true rate and this is taken to be the cost of choosing the predicted rate.
- If predicted rate is less than the true rate the difference in rate is the cost of using the predicted rate.

The expected cost of transmitting at a rate  $r_j$  denoted by  $C_j$  is given by:

$$C_j = \sum_{i=1}^p C_{ij} P(X_{n+1}^u = i | X_n^u \dots X_{n-\tilde{k}_{opt}^u}^u)$$

where

$$C_{ij} = \begin{cases} r_i, & r_i < r_j \\ r_i - r_j, & r_i \geq r_j \end{cases} \quad (31)$$

Here  $P(X_{n+1}^u = i | X_n^u \dots X_{n-\tilde{k}_{opt}^u}^u)$  is the probability of the system being in state  $i$  given that the sequence  $X_n^u \dots X_{n-\tilde{k}_{opt}^u}^u$  was observed, calculated using (1),(2). The predicted value of  $X_{n+1}^u$  is given by minimizing the expected cost  $C_j$ .

$$\hat{X}_{n+1}^u = \arg \min_j C_j \quad (32)$$

It is apparent that this cost function is designed to minimize the loss in rate *i.e.*, when a rate which is lower than the true rate is picked the packet transmission will be successful but there is

an obvious loss in efficiency and this loss is the cost incurred. On the other hand, if a higher rate is picked then there is a packet loss and we lose the true rate that we could have got, entirely. This biases the predictor to pick lower values than the MAP predictor, thus leading to a lower packet loss.

## VI. SIMULATIONS, RESULTS AND INFERENCE

Two cases of loading are considered i.e. a) Partial Loading,<sup>6</sup> b) Full Loading. For both these cases, we use the MCS sequences over 5000 sub-frames obtained from the full System Simulator as discussed earlier, for 210 users. This results in 210 sequences - one for each user, of length 1000, since, CQI feedback happens only once in every 5 sub-frames as discussed.

We also analyzed the MCS sequences generated for each UE in order to understand the behaviour of the sequences, in the case of partial and full loading. From the sequences  $X^u$  we generated an absolute difference sequence by computing  $|X_{n+\delta}^u - X_n^u|$  for all  $n$  and studied the statistics of this new sequence for all UEs. For each user this sequence can indicate the extent of variability of the MCS value at  $n$  and  $n + \delta$ . It was found that 35% of the users exhibited variations greater than 3 between adjacent values ( $X_{n+\delta}^u = X_n^u \pm 3$ ) for at least 200 times in a 1000 length sequence for partial loading, while only 5% of users under full loading had ( $X_{n+\delta}^u = X_n^u \pm 3$ ) for more than 50 times in a 1000 length sequence. For example, an MCS value of 15 could change to 12 or 18 before the next feedback, i.e., from a bits per symbol rate of 1.96 one will go down to 1.33. Similarly 20% of the users had variations greater than 4 between adjacent values ( $X_{n+\delta}^u = X_n^u \pm 4$ ) for at least 200 times in a 1000 length sequence for partial loading while there was not a single user with more than 25 such events in full loading. All of this points to a high degree of variability in the MCS sequence for partial loading. Hence outdated MCS seems to be a critical issue in partial loading.

For each user sequence  $X_1^u \cdot X_2^u \dots X_{1000}^u$ , the following prediction procedure is implemented on the system simulator

- 1) We build frequency trees upto depth  $m$ , which are updated as and when the sequence arrives. We choose  $m = 5$  since we are looking only at a sequence of length thousand <sup>7</sup>.

<sup>6</sup>For more details on partial loading refer to the Section II

<sup>7</sup>We have restricted the sequence length to 1000 due to a) the presence of UE sleep cycle and, b) assumption of stationarity of sequence may not hold over a long sequence length.



This can be increased to  $m = 8$  or higher, if one has access to longer sequences.

- 2) Then, using the frequency trees the probabilities  $P(X_n^u | X_{n-1}^u \dots X_{n-k}^u)$  are calculated as discussed earlier using (1),(2) with  $k = 1 \dots 4$ .
- 3)  $I_{pred}(k)$  is then calculated online *i.e.*, as each value is received, we use the probabilities obtained in Step 2 in (9), to compute the empirical value of  $I_{pred}(k)$  using the probabilities and sequences seen so far. At time  $n$  the sequence  $X_{n-1}^u \dots X_{n-k}^u$  is used to calculate  $P(X_n^u | X_{n-1}^u \dots X_{n-k}^u)$  and these probabilities are used as follows to find the instantaneous predictive information of the sequence:

$$I_{pred}(k, n) = \log(p) - \sum_{X_n^u=1}^p P(X_n^u | X_{n-1}^u \dots X_{n-k}^u) \log(P(X_n^u | X_{n-1}^u \dots X_{n-k}^u)) \quad (33)$$

This value of  $I_{pred}(k, n)$  is then empirically averaged over  $n$ , to get the current online estimate of  $I_{pred}(k)$  as follows:

$$I_{pred}(k) = \frac{1}{n} \sum_{i=1}^n I_{pred}(k, i)$$

- 4) From the  $I_{pred}(k)$  obtained in Step 3, using (19) which is the learning curve based stopping criterion, the value of  $k_{opt}^u$  is found for each user once the sequence length reaches 100, and this step is repeated once in every 100 values<sup>8</sup> of the sequence *i.e.*  $n = 200, 300$  and so on. It will take time to build a reasonably informative frequency tree for prediction. Hence, till the sequence length reaches 100 we do prediction using a simple Markov model *i.e.*, we do not wait for a training period before starting prediction.
- 5) Using  $k_{opt}^u$  as an upper bound on the model order, the optimal model order when the distribution is unknown  $\tilde{k}_{opt}^u$ , is found out using (28), (29) once the sequence length reaches 100 ,and this is also repeated once in every 100 values of the sequence.
- 6) Then the tree is virtually truncated at depth  $\tilde{k}_{opt}^u + 1$ .
- 7) This tree is used to find the probabilities  $P(X_n^u | X_{n-1}^u \dots X_{n-\tilde{k}_{opt}^u}^u)$  which are now used in the prediction algorithm.
- 8) These probabilities  $P(X_n^u | X_{n-1}^u \dots X_{n-\tilde{k}_{opt}^u}^u)$  obtained from Step 7) are used for prediction. We compare this with probabilities obtained from a virtually truncated tree of fixed depth 4. The tree of fixed depth 4 gives us the probabilities  $P(X_n^u | X_{n-1}^u, X_{n-2}^u, X_{n-3}^u)$ . The

<sup>8</sup>The sequence should be of a sufficient length to get a reasonable average.

predictors using  $P(X_n^u | X_{n-1}^u \dots X_{n-k_{opt}^u}^u)$  and  $P(X_n^u | X_{n-1}^u, X_{n-2}^u, X_{n-3}^u)$  are henceforth referred to as Variable Order (VO) predictors and Fixed Markov (FM) predictors respectively.

We use the probabilities computed using PPM with VO and FM in the MAP predictor in (30) and in the Bayesian Risk Mimimizer (BRM) presented in Section V-B in (32) and compare the performance of the four schemes namely, FM-MAP, FM-BRM, VO-MAP and VO-BRM. In [7] nine techniques are proposed for prediction and out of those the median technique where the median of previous  $n$  CQI values is taken, performs best for vehicular users. Since we have Doppler see Table II and partial loading, we compare our schemes with the median technique in [7]. A naive algorithm with no prediction *i.e.*, when the previous value is used as it is, is also compared with the above given techniques.

We compare the various schemes based on the following metrics:

- Packet loss fraction ( $P_{loss}^u$ ): We compute the packet loss fraction for each user and it is given by:

$$P_{loss}^u = \frac{\sum_{n=1}^P 1(\hat{X}_n^u > X_n^u)}{P} \quad (34)$$

where  $P$  is the total number of packets transmitted. Packet loss occurs whenever the predicted MCS is greater than the actual MCS .

- Rate Efficiency Percentage( $r_{eff}^u$ ): The rate obtained due to the a specific prediction scheme is compared with the rate obtained if there was ideal prediction and  $r_{eff}^u$  for each user is :

$$r_{eff}^u = \begin{cases} \frac{Rate_{currentscheme}}{Rate_{ideal}}, & Rate_{currentscheme} \leq Rate_{ideal} \\ 0, & Rate_{currentscheme} > Rate_{ideal} \end{cases} \quad (35)$$

It is well known that one can reduce packet loss by reducing the MCS and transmitting at increasingly conservative rates. However, our schemes reduce the packet loss and at the same time improve rate efficiency, since they exploit the fact that one can learn/predict current MCS value by analyzing the complexity of the MCS sequence. Moreover, since MCS sequences of different UEs have varying complexities, we use independent learning mechanisms for each UE.

Since there are 210 users, for both partial and full loading, the empirical Cumulative Distribution Function (CDFs) are plotted for all the above mentioned metrics and these are discussed in detail. The packet loss fraction CDF under partial loading, is compared in Fig. 3a and here

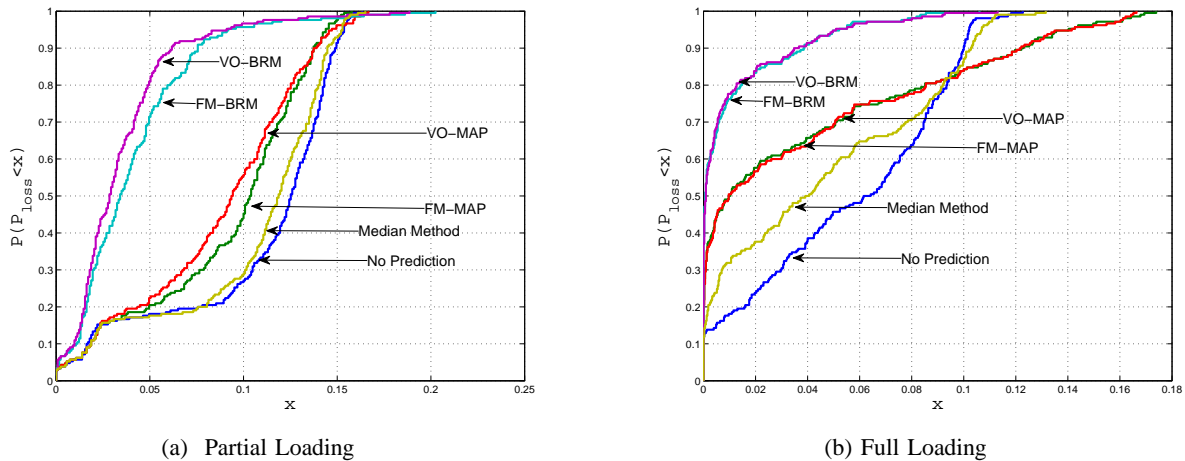


Fig. 3: Packet Loss Fraction CDFs

it can be seen that the BRM predictors significantly outperform all other methods by having the lowest percentage of failed transmissions. When the VO-BRM method is used, 90% of the users have less than 6.3% packet loss, while when FM-BRM is used the corresponding packet loss is 7.6%. In comparison the VO-MAP, FM-MAP, Median and No Prediction have only 35%, 30%, 22% and 20% users with packet loss rate less than 7.6%. At the 50-percentile point<sup>9</sup> in the packet loss distribution, VO-BRM at 2.8% packet loss, outperforms the FM-BRM by 20% and the VO-MAP and FM-MAP schemes by more than 200% and 250% respectively, median scheme proposed in [7] by 400% and the no prediction scheme by nearly 450%. This gain in packet loss performance is achieved with no loss in rate.

The rate efficiency CDF under partial loading is compared in Fig. 4a and here again it can be seen that the BRM outperforms all other methods by having the highest rate efficiency. Here, VO-BRM has 76% users achieving a rate efficiency of 90% or higher, while FM-BRM had only 69% users with this criteria. This implies that while 160 users achieve a high rate efficiency using VO-BRM, only 146 users achieve the same using FM-BRM. The corresponding percentage of users with that rate efficiency were 38%, 35%, 26% and 23% for VO-MAP, FM-MAP, median technique and scheme without prediction respectively.

When we look at full loading performance graphs in Fig. 3b and Fig. 4b we can see that

<sup>9</sup>corresponds to packet loss seen by at least 50% of the users

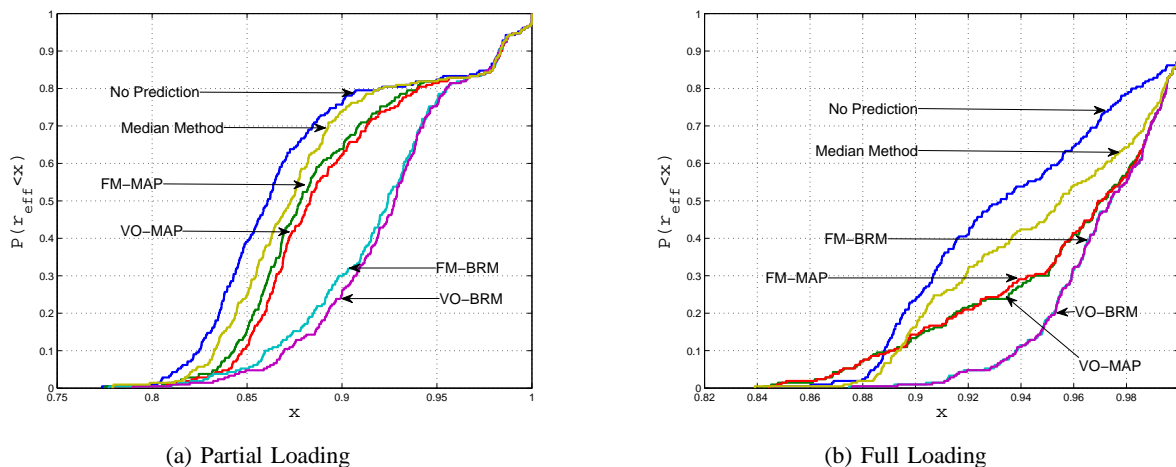


Fig. 4: Rate Efficiency CDFs

the trends of MAP versus BRM are similar *i.e.*, BRM is way better than MAP in packet loss percentage and in rate efficiency. There is a cross-over between the MAP and no prediction CDFs in packet loss percentage as seen in Fig. 3b. This is because of the behavior of the MAP predictor where all errors are treated equal. Especially, when MAP predicts an MCS that is higher than the previous fed-back value and it is also higher than the true value, a packet loss occurs. Therefore, for some users the no prediction scheme performs better than MAP prediction. This effect is seen in the full loading scenario because, the MCS variation itself is likely to be more gradual and even without prediction, sometimes the fed-back MCS works better than a predicted MCS. However, on an average the MAP is better than not predicting and BRM is far better than both. However, when one compares FM to VO, it can be seen that, there is little to choose between them across all the performance metrics considered under full loading. This implies that partial loading requires us to adapt the model order, while, full loading performance may not require us to adapt the model order. Since all practical systems see partial loading, either due to traffic or due to sub-frame blanking, VO based methods are required to fully exploit the advantages of rate adaptation.

## VII. CONCLUSIONS

The effect of outdated MCS in the presence of partial loading was investigated. Discrete sequence prediction algorithms such as PPM were proposed for MCS prediction. The optimal

tree depth that one needs to traverse for prediction using PPM was cast as a model order problem. Techniques such as MDL, AIC and Corrected AIC were proposed to estimate the model order of the sequence for each user with the sequence complexity analysis providing an upper bound on the model order. Finally, the MAP and Bayesian Risk minimization based rate predictors were proposed and implemented for MCS prediction. Simulation results indicates that, using different model order for different users, gives substantial system level gains over assuming a fixed model order for all users. The gains due to adapting the model order, were found to be substantial in partially loaded systems. Furthermore, the proposed Bayesian Risk Minimization predictor, significantly outperforms the MAP based predictor.

## REFERENCES

- [1] S. Sesia, I. Toufik, and M. Baker, *LTE: The UMTS long term evolution*. Wiley Online Library, 2009.
- [2] D. Martin-Sacristan, J. F. Monserrat, J. Gozalvez, and N. Cardona, "Effect of Channel-Quality Indicator Delay on HSDPA Performance," in *IEEE 65th Vehicular Technology Conference*,. IEEE, Apr. 2007, pp. 804–808.
- [3] A. Kuhne and A. Klein, "Throughput analysis of multi-user OFDMA-systems using imperfect CQI feedback and diversity techniques," *IEEE Journal on Selected Areas in Communications*, , vol. 26, no. 8, pp. 1440–1450, 2008.
- [4] H. Dai, Y. Wang, C. Shi, and W. Zhang, "The Evaluation of CQI Delay Compensation Schemes Based on Jakes' Model and ITU Scenarios," in *Vehicular Technology Conference (VTC Fall)*,, Sept. 2012, pp. 1–5.
- [5] T. Cui, F. Lu, V. Sethuraman, A. Goteti, S. Rao, and P. Subrahmanya, "First Order Adaptive IIR Filter for CQI Prediction in HSDPA," in *Wireless Communications and Networking Conference (WCNC)*, , Apr. 2010, pp. 1–5.
- [6] R.A.Akl, S.Valentin, G.Wunder, and S.Stanczak, , "Compensating for CQI aging by channel prediction: The LTE downlink," in *Global Communications Conference (GLOBECOM)*,, Dec 2012, pp. 4821–4827.
- [7] D. Martin-Sacristan, and J.F.Monserrat, and D.Calabuig, and N.Cardona, , "HSDPA Link Adaptation Improvement Based on Node-B CQI Processing," in *4th International Symposium on Wireless Communication Systems*, , Oct. 2007, pp. 597–601.
- [8] Al-Rawi, M. and Huschke, J. and Sedra, M., "Dynamic Protected-Subframe Density Configuration in LTE Heterogeneous Networks," in *21st International Conference on Computer Communications and Networks (ICCCN)* , Jul. 2012, pp. 1–6.
- [9] D. Lopez-Perez, I. Guvenc, G. De La Roche, M. Kountouris, T. Q. Quek, and J. Zhang, "Enhanced intercell interference coordination challenges in heterogeneous networks," *Wireless Communications, IEEE*, vol. 18, no. 3, pp. 22–30, 2011.
- [10] H.Holma and A.Toskala, *LTE for UMTS-OFDMA and SC-FDMA based radio access*. John Wiley & Sons, 2009.
- [11] K. Gopalratnam and D. J. Cook, "Online sequential prediction via incremental parsing: The Active LeZi algorithm," *IEEE Intelligent Systems*, vol. 22, no. 1, pp. 52–58, 2007.
- [12] D. Katsaros and Y. Manolopoulos, "Prediction in wireless networks by Markov chains," *IEEE Wireless Communications*, vol. 16, no. 2, pp. 56–64, 2009.
- [13] J. Rissanen, "Modeling by Shortest Data Description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.
- [14] W. Bialek, I. Nemenman, and N. Tishby, "Predictability, complexity, and learning," *Neural Computation*, vol. 13, no. 11, pp. 2409–2463, 2001.

- [15] K. Aho, O. Alanen, and J. Kaikkonen, "CQI Reporting Imperfections and their Consequences in LTE Networks," in *The Tenth International Conference on Networks*, 2011.
- [16] Le Thanh Tu. et al, "Final Version of System Level Simulator," 2007. [Online]. Available: <http://www.ict-codiv.eu/private/docs/deliverables/D5.4.pdf>
- [17] "Evolved universal terrestrial radio access (E-UTRA); Physical channels and modulation (release 8)," 2008. [Online]. Available: [www.3gpp.org](http://www.3gpp.org)
- [18] T.-T. Tran, Y. Shin, and O.-S. Shin, "Overview of enabling technologies for 3GPP LTE-advanced," *EURASIP Journal on Wireless Communications and Networking*, no. 1, pp. 1–12, 2012.
- [19] "Universal Mobile Telecommunications System (UMTS); Spatial channel model for Multiple Input Multiple Output (MIMO) simulations (3GPP TR 25.996 version 10.0.0 Release 10)."
- [20] "Evolved universal terrestrial radio access (E-UTRA); Physical layer aspects (Release 9)," 2010.
- [21] J. Cleary and I. Witten, "Data compression using adaptive coding and partial string matching," *IEEE Transactions on Communications*, vol. 32, no. 4, pp. 396–402, 1984.
- [22] J. Ziv and A. Lempel, "A universal algorithm for sequential data compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, 1977.
- [23] R. Begleiter, R. El-Yaniv, and G. Yona, "On prediction using variable order Markov models," *J. Artif. Intell. Res.(JAIR)*, vol. 22, pp. 385–421, 2004.
- [24] T. M. Cover and J. A. Thomas, *Elements of information theory*. John Wiley & Sons, 2012.
- [25] W. Feller, *An introduction to probability theory and its applications*. John Wiley & Sons, 2008, vol. 1.
- [26] A. Papoulis and S. Pillai, "Probabilities, Random Variables, and Stochastic Processes (4/e. NY: McGraw-Hill, 2002)," 1991.
- [27] H. Bozdogan, "Model selection and Akaike's information criterion (AIC): The general theory and its analytical extensions," *Psychometrika*, vol. 52, no. 3, pp. 345–370, 1987.
- [28] N. Merhav, M. Gutman, and J. Ziv, "On the estimation of the Order of a Markov chain and Universal Data Compression," *IEEE Transactions on Information Theory*, vol. 35, no. 5, pp. 1014–1019, 1989.
- [29] S. M. Kay, *Fundamentals of Statistical signal processing, Volume 2: Detection theory*. Prentice Hall PTR, 1998.
- [30] G. Claeskens and N. L. Hjort, *Model selection and model averaging*. Cambridge University Press Cambridge, 2008.
- [31] C. M. Hurvich and C.-L. Tsai, "Regression and time series model selection in small samples," *Biometrika*, vol. 76, no. 2, pp. 297–307, 1989.
- [32] J. E. Cavanaugh, "Unifying the derivations for the Akaike and corrected Akaike information criteria," *Statistics & Probability Letters*, vol. 33, no. 2, pp. 201–208, 1997.