Wireless Spectrum Occupancy Prediction Based on Partial Periodic Pattern Mining

Pei Huang, Chin-Jung Liu, Xi Yang, Li Xiao, Jin Chen

Abstract—Cognitive radio appears as a promising technology to allocate wireless spectrum between licensed and unlicensed users in an efficient way. The availability of spectrum holes vastly affects the throughput and delay of unlicensed users. Prediction models for inferring the availability of spectrum holes can help to improve the spectrum extraction rate and reduce the collision rate. In this paper, a spectrum occupancy prediction model based on Partial Periodic Pattern Mining (PPPM) is introduced. The mining aims at identifying frequent spectrum occupancy patterns that are hidden in the spectrum usage of a channel. The mined frequent patterns are then used to predict future channel states (i.e., busy or idle). Based on the prediction, unlicensed users are able to extract spectrum holes aggressively without introducing significant interference to licensed users. PPPM outperforms traditional Frequent Pattern Mining (FPM) by considering real patterns that do not repeat perfectly due to noise, sensing errors, and irregular behaviors. Using real life network activities we show a significant reduction on miss rate in channel state prediction. With the proposed prediction mechanism, the performance of Dynamic Spectrum Access (DSA) is substantially improved. Further, we extend the three-state PPPM to an N-state PPPM to predict the duration of high/low utilization in a channel. The frequent patterns of channel utilization duration are critical in optimizing channel switch strategies. The high prediction accuracy is validated with data collected in the paging bands.

Index Terms—Cognitive radio, dynamic spectrum access (DSA), spectrum occupancy prediction, partial periodic pattern mining

1 Introduction

The rapid growth of wireless applications has resulted in a dense allocation of frequency bands. Traditional static frequency allocation does not consider variations of usage in both spatial and temporal domains. Recent measurements have revealed that many licensed bands are underutilized [1], whereas many services such as WLAN, Bluetooth, and ZigBee are confined to crowded unlicensed bands. To exploit the temporarily unused spectrum, the concept of Cognitive Radio (CR) has been introduced. CR is an intelligent radio that can sense and learn from its surrounding environment and adjust its operating parameters accordingly [2]. In a CR network (CRN), unlicensed users known as secondary users (SUs) sense the licensed channels and use the slots (called spectrum holes) that are not occupied by primary users (PUs). The dynamic spectrum access (DSA) increases the available spectrum resources for SUs, but SUs must immediately vacate the channels upon PUs' return [3].

Dynamic behaviors of PUs require accurate sensing and fast switch strategies to convince service providers to open their licensed bands. Due to the difficulty of accurate sensing for low power signals, FCC decided to adopt the geo-location and database access method, eliminating the spectrum sensing requirement for unlicensed TV band devices [4]. However, it is hard to determine the accurate TV signal reception range due to the tropospheric propagation [1]. In addition, some service bands (e.g., personal communication service bands and paging bands) have lots of short and frequent sessions, preventing the use of database access method, which may incur high overhead for frequent updating. Even between SUs, spectrum sharing is a tough problem. Due to power asymmetry or different physical layer standards, some SUs may act as virtual PUs towards the other SUs. With spectrum occupancy prediction, CR devices can aggressively extract spectrum holes but avoid time slots of high collision probabilities. If collision rate can be upper bounded by prediction-assisted DSA, more service providers will be willing to open their licensed bands for additional return on investment.

Wireless communications are designed to conform to certain protocols. Patterns are expected to be observed in spectrum use. Recently a Frequent Pattern Mining (FPM) method has been introduced to find frequent patterns and encouraging prediction performance has been demonstrated [5]. To discover more useful patterns, this paper introduces a Partial Periodic Pattern Mining (PPPM) based spectrum occupancy prediction (i.e., channel state prediction) model. Different from full periodic patterns, partial periodic patterns identify regularity of behaviors at some but not all points of time [6]. For example, a pattern disclosing that the channel utilization of a channel is high during certain hours every day but not regular in other time is a partial periodic pattern.

A spectrum occupancy pattern is affected by various factors such as noise, sensing errors, and irregular user behaviors. Therefore, the periodicity may be partially observable. PPPM algorithm is tailored for identification of real patterns that are irregular in nature. Using PPP-

• *P. Huang, C.-J. Liu, X. Yang, L. Xiao, and J. Chen are with the Department of Computer Science and Engineering, Michigan State University, 3115 Engineering Building, East Lansing, MI 48824.*
E-mail: {huangpe3, liuchinj, yangxi1, lxiao, jinchen}@cse.msu.edu

M, more occupancy patterns are identified, leading to extra channel state prediction rules that can reduce the number of unpredictable slots.

Besides accurate prediction, timely estimation of spectrum availability is also critical. Hence, we speed up the mining process in two phases. First, for each candidate frequent pattern, we reduce the number of subsequences that need to be examined in a time series by introducing an index list structure. Using the index list structure, counting the occurrences of a pattern does not need to scan the entire database once, improving mining expedition. Second, we reduce the number of candidate frequent patterns by identifying patterns that cannot yield longer frequent patterns so as to stop mining on them early. Moreover, by checking whether a pattern will be absorbed by another pattern, we avoid redundant mining on two branches that produce identical frequent patterns. Finally, we extend the prediction mechanism introduced early in [7] to predict the remaining time of high/low utilization in different channels. This helps to optimize channel switch strategies of CR devices. In summary, the contributions of this work are as follows.

- An efficient Partial Periodic Pattern Mining (PPPM) algorithm is proposed to mine rules for predicting the channel state in the next time slot.
- An index list structure along with an Apriori-like property and a backward-extension rule are introduced to speed up the mining process.
- An efficient channel utilization duration mining algorithm is developed for predicting the high/low utilization duration of a channel.

We compared our PPPM with the FPM and the hidden Markov model (HMM) using real life network activities and data collected in the Personal Communication Service (PCS) bands. The results show that the proposed PPPM-based prediction can predict more slots than FPM-based prediction, showing that many spectrum occupancy patterns only exhibit partial periodicity. The PPPM also outperforms HMM for skipping uncertain slots. In addition, we particularly observed that distinguishing low utilization periods from high utilization periods and mine rules in corresponding utilization periods will substantially improve the prediction performance.

In order to show the advantages of integrating prediction in DSA, we compared our prediction-assisted DSA with a statistical knowledge-based DSA. The results show that the prediction of channel state significantly improves the **spectrum extraction rate**, which is defined as the ratio between the number of idle slots that are actually obtained by SUs and the total number of available idle slots left by PUs.

2 Related work

Various models have been developed to provide prediction for signal power, duty cycle, and channel state. We give a brief review over them in this section and also discuss related work on partial periodic pattern mining.

Several papers [8] [9] [10] model the variation of signal power in a channel. In [8], a second-order autoregressive (AR-2) process is used to model the channel variation. Once the parameters of the AR-2 model are computed, the signal power can be predicted and the channel state can be estimated via a Kalman filter. In [9], a moving average model (MA) is introduced and it is integrated with the AR model to create an autoregressive moving average model (ARMA). The results show that the time series of all TV channels fall into the moving average model. An ARMA model requires that the time series is stationary, which means the statistical properties of a time series is similar to those of time shifted series [11]. An autoregressive integrated moving average (ARIMA) model is introduced in [11] to handle non-stationary time series. It models the changes of band occupancy, which is defined as the ratio between the number of busy channels and the total number of channels in a given band. An inconvenience of these models is that a time series must be converted to a stationary and periodic time series before analyzing it.

Instead of predicting signal power, many papers consider that the channel is either detected as occupied or unoccupied and the channel states constitute a binary time series. Most studies assume that a Markov chain exists in such a binary time series. Assuming that the PU's traffic follows a Poisson process, a hidden Markov model (HMM) based DSA scheme is introduced in [12]. In [13], several deterministic traffic patterns are studied for the HMM-based channel state predictor. A more detailed analysis of the HMM-based predictor design is presented in [14], where a multilayer perceptron (MLP) based channel state predictor is also analyzed. The existence of Markov chain in spectrum occupancy by PUs has been validated in [15] with data collected in the paging bands. Introducing one more state named "fuzzy" (unknown availability), a 3-state higher-order HMM channel state predictor is introduced in [16]. Recently, a frequent pattern mining (FPM) based method [5] demonstrates that it provides better channel state prediction over the first-order HMM-based predictor. In this paper, we show that PPPM can generate more prediction rules, leading to fewer unpredictable slots.

Mining patterns with gap constraints has been studied for sequence databases [17] [18] [19] [20]. Their methods, however, are not suitable for our problem. The number of entries in a sequence database is fixed. On the contrary, the number of sequences in a time series is changing along with the length of a sequence. Partial periodic pattern mining in time series databases has been studied in [6] [21], but the algorithm aims at enumerating all partial periodic patterns of a single period. It has to construct a complex max-subpattern tree for a single period. For a set of periods, the max-subpattern hit algorithm incurs high overhead for looping over the single period mining. In spectrum occupancy prediction, there is no need to enumerate all partial periodic patterns of a single period. Many of them are inappropriate for

use because they include too many uncertain symbols. A pattern with a lot of uncertain symbols has very low prediction power. Therefore, we use a gap constraint (introduced in Section 3) to filter out useless patterns. We also introduce conditional entropy (in Section 4) to generate partial periodic patterns incrementally, identifying patterns of unknown lengths without using complex structures of high overhead. We further study how to extract prediction rules from mined patterns and utilize them to improve communication efficiency.

The formalization of our problem uses terms (e.g., symbol, alphabet, pattern) that are used in language theory [22] [23] [24] [25], but our study is not to parse a sentence and derive the meaning. Our work is to check whether there exist frequent patterns in spectrum use even when signals of multiple devices are mixed.

In duration prediction, many studies focus on finding a good statistical model to specify the distribution of idle durations [5] [26] [27]. However, the assumed underlying distribution may not actually hold in unknown traffic and thus the parameters may be hard to estimate. Moreover, the distribution may be different when different patterns are observed. Our mining discovers the subtle changes of idle duration distribution in different periods.

3 Problem Formulation

In this section, we give a formal definition of our partial periodic pattern mining in a spectrum usage database.

In CRNs, the SUs are responsible for sensing the channel before commencing any transmission so as to guarantee that the interference caused by them is under a certain limit. Multiple PUs can be viewed as a single virtual PU. If the spectrum use of the virtual PU exhibits some patterns, SUs can learn from the sensing results and utilize the frequent patterns to predict the availability of spectrum holes.

Let '1' denote busy and '0' represent idle. The channel states in a period can be represented by a binary time series. Let I be the alphabet of all possible values that occur in the time series, we have $I = \{0, 1\}$. A **wild card** (denoted by a single $*$) is a special character that matches any value in I, which means it does not specify the channel state of a slot because the uncertainty in the slot is high. A **gap** is a sequence of wild cards, and the *size* of a gap is the number of wild cards in it. We use g^m to represent a gap whose size is within the range $[0, m]$, which means the wild card $*$ can be repeated at most m times in a gap. Here, m is the **gap constraint**. If a pattern has too many wild cards, any sequence of observed channel states can match the pattern. The prediction power of the pattern is weak. Therefore, we stop developing a pattern if it violates the gap constraint. Same as other PPPM algorithms [17] [18] [19] [20], it is a user-defined parameter in the mining algorithm.

A **symbol** in a pattern is a **value** in I or a wild card. Given a pattern P, we use $P[i]$ to represent the ith symbol of P and p_i to represent the ith value of P. A pattern $P = p_1 g^m p_2 g^m ... p_{q-1} g^m p_q$ is a set of values and gaps. We define the *length* of a pattern P (denoted by $l = |P|$) as the number of symbols in P, which means the wild cards are also counted towards the pattern's length. If $l \geq 2$, the substring that contains the first $l-1$ symbols is called the *prefix* of P (denoted by *prefix(P)*), and the substring that contains the last $l-1$ symbols is the *suffix* of P (denoted by *suffix(P)*).

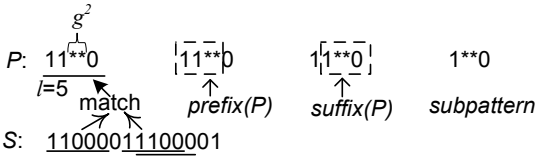

Fig. 1: Terms being used. Note that $\langle 1000 \rangle$ is not regarded as a subpattern of $\langle 11**0 \rangle$ even though $*$ is a wild card that matches any value.

A pattern $P = P[1]P[2]...P[l_1]$ is a *subpattern* of $C = C[1]C[2]...C[l_2]$ if $l_2 > l_1$ and there exists an integer $1 \leq i \leq l_2 - l_1 + 1$ such that $P[1] = C[i], P[2] = C[i+1], ..., P[l_1] = C[i+l_1-1]$. C is thus a *superpattern* of P. An example is illustrated in Fig. 1. Note that $\langle 1000 \rangle$ is not regarded as a subpattern of $\langle 11**0 \rangle$ because it violates the Apriori property in data mining (i.e., the support of a pattern cannot exceed the support of any of its subpatterns [28]). The pattern $\langle 11**0 \rangle$ may have more matches than $\langle 1000 \rangle$ in a time series and thus $\langle 1000 \rangle$ cannot be regarded as a subpattern of $\langle 11**0 \rangle$.

Our goal is to identify frequent patterns in a time series $S = s_1 s_2 ... s_n$ where s_i is the ith value of S. We need to define the term *frequency* and how often a pattern should occur before we can consider it *frequent* in S. We define the *frequency* of a pattern P by the probability of observing P if we randomly pick a subsequence of length l in S. The time series S can be divided into N_l subsequences of equal length l that start at different starting points: $S_i = s_i s_{i+1} ... s_{i+l-1}$ where $1 \leq i \leq n-l+1$ and $N_l = n - l + 1$. A subsequence S_i *matches* a pattern P of length l if for each position $1 \leq j \leq l$, $s_{i+j-1} \in P[j]$ (a wild card matches any value). The **support** of P in S (denoted by $sup(P)$) is the number of subsequences in S that match the pattern. The **confidence** of P (denoted by $conf(P)$) is defined as the division of its support by the total number of subsequences of length l in S, which reflects the probability of observing P in S.

We say that a pattern is a *frequent partial periodic pattern* in a time series if its confidence exceeds a user-specified threshold ρ_c, which is defined as the **confidence constraint**. For example, if a pattern $P = \langle 0*1 \rangle$ appears 5 times in a time series $S = \langle \mathbf{00}11\mathbf{00}110\mathbf{0}01 \rangle$, the confidence of P is $conf(P) = sup(P)/N_l = 5/10 = 1/2$. If $\rho_c = 0.5$, P can be considered as a frequent pattern. We use confidence instead of support to measure the frequency because the supports of long patterns are usually lower than that of short patterns. This is because the total number of sequences of length l is monotonically decreasing along with the increasing length l. The support is thus normalized by N_l to correctly reflect the frequency.

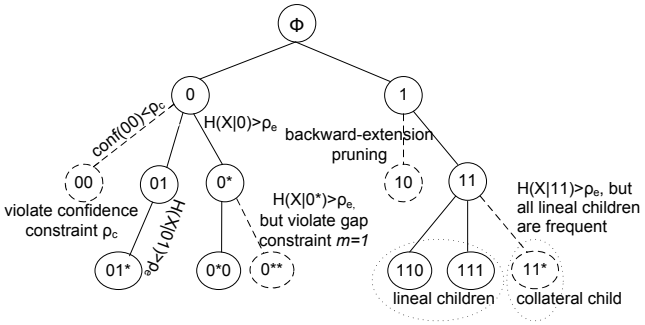

Fig. 2: An illustration of the frequent partial periodic pattern tree.

4 Partial Periodic Pattern Mining

Frequent pattern mining is to enumerate candidate frequent patterns and count the support of each pattern to determine whether a pattern is frequent. The general steps of mining include *generating candidate patterns*, *counting the support*, and *pruning candidate patterns*. To adapt to different applications, different techniques are designed to discover different useful patterns. This paper introduces conditional entropy to determine whether a wild card is needed in generating partial periodic patterns. An index list structure is designed to speed up support counting. In addition, an Apriori-like property is derived for confidence-based candidate pattern pruning and a backward-extension checking is introduced to further make the pruning process efficient.

4.1 Frequent Pattern Enumeration

The complete search space of frequent partial periodic patterns forms a tree as shown in Fig. 2. Starting at a root node that is labeled with Φ, we can recursively extend a node of level L on the tree by concatenating a value in I or a wild card that satisfies the gap constraint to get a child node on the next level $L+1$. Whether a parent node Q will develop a child node by concatenating a wild card largely depends on the conditional entropy $H(X|Q)$ defined in Definition 2.

Entropy is a measure of the uncertainty inherent in the distribution of a random variable. When the entropy of a random variable is large, it implies that the uncertainty as to the value of that random variable is large. We hence consider it as a noise, but the noise should not affect the discovery of long patterns. We give the definitions of entropy and conditional entropy as follows.

Definition 1 *The entropy of a discrete random variable X is defined as $H(X) = -\sum_{x \in I} p(x) log_2 p(x)$ where $p(x)$ is the occurrence probability of x and $I = \{0, 1\}$.*

Definition 2 *For a given node Q, the conditional entropy of a discrete random variable X after node Q is defined as $H(X|Q) = -\sum_{x \in I} p(x|Q) log_2 p(x|Q)$ where $p(x|Q)$ denotes the fraction of subsequences with prefix Q in S that match $P = Q \oplus x$ and $\oplus$ denotes concatenation.*

If the conditional entropy of a node $H(X|Q)$ is higher than a user-specified threshold ρ_e, the node will develop a child node by extending itself with a wild card if both the gap constraint and the confidence constraint are not violated and at least one child with a value extension cannot exceed the confidence threshold. The ρ_e specifies how users define the uncertainty. In our case, if no value (i.e., either 0 or 1) appears after a pattern with a probability higher than 75%, we consider the next slot following the pattern is uncertain. Consequently, a child with the extension of $*$ may be generated and it is the *collateral child* of the pattern. The other two children with a value extension are considered as the *lineal children* of the pattern. The children of a node are generated and arranged according to a lexicographical order. If a node is not frequent, we stop developing its subtree. To save memory, we grow a pattern in a depth-first search way so that we can remove a branch when it is done. Algorithm 1 summarize the mining.

Algorithm 1: Partial periodic pattern mining.

```
let C be a stack;
C.push(1);
C.push(0);
while C is not empty do
    P ← C.pop();
    P_0 ← P ⊕ 0;
    P_1 ← P ⊕ 1;
    P_2 ← P ⊕ *;
    if P_0 or P_1 is infrequent and H(X|P) > ρ_e then
        if P_2 meets gap constraint then
            C.push(P_2);
    if P_1 is frequent then
        C.push(P_1);
    if P_0 is frequent then
        C.push(P_0);
```

The introduction of the collateral child has two advantages. First, channel sensing is not perfect. Misdetections will reduce the observed occurrences of a pattern. The wild cards can save some frequent patterns that are near the decision boundary. For example, two patterns $\langle \underline{00011} \rangle$ and $\langle \underline{00010} \rangle$ may occur 987 and 324 times respectively. However, the confidence constraint requires that a pattern of length 5 must occur at least 1000 times to be considered frequent. If most occurrences of the two patterns are followed by a '0', we miss an important frequent pattern $\langle \underline{0001} * 0 \rangle$ and any frequent pattern that has the prefix of $\langle 0001 * 0 \rangle$. Noise and sensing errors may make the occurrences of a pattern fall below the support threshold. Therefore, when the lineal children of a node cannot pass the confidence constraint while its collateral child can, we go on developing the subtree of this node so as to avoid missing any possible frequent pattern. Mining on the subtree will terminate soon either due

to the violation of gap constraint or due to the violation of confidence constraint. In the example, the support of $P = \langle \underline{0001}* \rangle$ is as low as its parent pattern $\langle \underline{0001} \rangle$ (i.e., 1311). If we extend the P with values in I, the support is reduced as it splits into two parts. Mining on the subtree will terminate in several levels when superpatterns fail to meet the confidence constraint. If we keep extending the P with wild cards, the support will not be reduced as a wild card matches either 1 or 0. However, suppose the gap constraint is m. We do not allow m consecutive wild cards. Mining on the subtree will terminate in m levels due to the violation of gap constraint.

Second, even when one lineal child of a node is frequent, we may want to develop an additional pattern that accounts for the irregularity of behaviors. Suppose one of the two patterns described above passes the confidence constraint with some irregular slots (e.g., $\langle \underline{0001}0 \rangle$ occurs 1328 times and $\langle \underline{0001}1 \rangle$ occurs 964 times). The irregularity will be detected by the entropy of their parent (i.e., $H(X \mid 0001)$). An additional pattern $\langle \underline{0001}* \rangle$ is developed along with $\langle \underline{0001}0 \rangle$. The reason is that the regularity of behaviors during a certain period of time may not be identified. For example, people make phone calls in a pure random manner. It may be unable to identify regular channel activities during certain periods of time but it is desirable to discover frequent patterns like $\langle \underline{0001} * * * *0001 \rangle$ so that we can predict channel states regardless of observed channel states in some time slots. These partial periodic patterns specify regularity of behaviors at some but not all points of time. Considering that partial periodicity exists ubiquitously in real life, more patterns can be discovered and more useful prediction rules can be extracted.

If all of the lineal children are frequent, we consider them as two different frequent patterns and do not generate the collateral child even though the entropy of their parent is higher than ρ_e.

4.2 Support Counting

For each candidate pattern, we have to check whether it is frequent by counting its support. This requires us to scan the time series S once. Some algorithms traverse the entire time series as many times as needed [6] [20]. Since we perform depth-first search, we can use an index list structure to reduce the number of subsequences that need to be checked.

Given a time series S and a pattern P of length l, the *head index list* of P (denoted by $HIL(P)$) is a list of indices where the P appears in S. For example, if $S = \langle \mathbf{00}1101\mathbf{0}0010 \rangle$ and $P = \langle 0 * 1 \rangle$, then $HIL(P) = 1, 2, 8$. Given $HIL(P)$, it is easy to count the supports of patterns that have the prefix P.

A pattern Q of length $l-1$ has three children: $P_0 = Q \oplus 0$, $P_1 = Q \oplus 1$, and $P_2 = Q \oplus *$. Their HILs can be computed by examining subsequences that start at positions specified by $HIL(Q)$.

For each head index x in $HIL(Q)$, if there exists a subsequence of length l that starts at x in S, we check the last value of the subsequence. If the value is equal to '0', the head index x is inserted to $HIL(P_0)$, otherwise the head index x is inserted to $HIL(P_1)$. The HIL of the collateral child P_2 is given by

$$HIL(P_2) = HIL(Q). \quad (1)$$

Since we do depth-first search, the length of the HIL is monotonically decreasing until we reach a pattern that is not frequent and we stop developing the subtree. The method speeds up the mining process quite a lot in practice because the number of subsequences that need to be examined is significantly reduced, and in each comparison a single character comparison is much faster than a string comparison.

Algorithm 2: Head index list construction.

```
for x ∈ HIL(Q) do
    if (x + l − 1) ≤ n then
        if S[x + l − 1] == '0' then
            Insert x in HIL(P_0);
            sup(P_0) ← sup(P_0) + 1;
        else
            Insert x in HIL(P_1);
            sup(P_1) ← sup(P_1) + 1;
```

4.3 Candidate Pattern Pruning

A common problem in data mining is that the number of candidate patterns is huge. There exist about 3^l candidate patterns for a certain length l in our problem. It is infeasible to enumerate all candidate frequent patterns and count their supports. In this section, we introduce two constraints for efficient candidate pattern pruning.

4.3.1 Confidence constraint

For efficient pruning, most mining algorithms adopt the *Apriori* property, which states that any subpattern of a frequent pattern must also have the minimum support [28]. We can extend the *Apriori* property to the concept of confidence if the total number of sequences is fixed. However, the number of sequences of length l in a time series is changing with l. For example, in a time series $S = \langle \underline{001}\underline{001} \rangle$, a pattern $P = \langle 001 \rangle$ and its subpattern $Q = \langle 00 \rangle$ both have $sup(P) = sup(Q) = 2$, but $N_3 = 4$ and $N_2 = 5$. As a result, $conf(P) = \frac{2}{4}$ and $conf(Q) = \frac{2}{5}$, which shows that the confidence of a pattern may exceed the confidence of its subpattern. Therefore, we cannot prune a candidate pattern even though its confidence is lower than the user-specified threshold ρ_c; otherwise we are unable to discover some long patterns. To achieve efficient pruning, we derive an *Apriori-like* property in Theorem 1.

Theorem 1 *Given a length l pattern P and a length $m < l$ subpattern $Q = P[i]P[i+1]...P[i+m-1]$ of P, where $1 \le i \le l-m+1$, we have $conf(Q) \ge \frac{N_l}{N_m} \rho_c$.*

Proof: Because Q is a subpattern of P, we have $sup(Q) \geq sup(P)$. If a length l pattern P is frequent, then by definition, we have $conf(P) = sup(P)/N_l \geq \rho_c$. Now, consider a length $m < l$ subpattern Q of P, we have

$$conf(Q) = \frac{sup(Q)}{N_m} \geq \frac{sup(P)}{N_m} \geq \frac{N_l}{N_m}\rho_c = \lambda_{l,m} \cdot \rho_c \quad (2)$$

□

Theorem 1 implies that any length-m subpattern Q of a frequent length-l pattern P must retain a confidence that is greater than or equal to $\lambda_{l,m} \cdot \rho_c$. This Apriori-like property allows us to prune a large number of candidate patterns from consideration. Suppose the length of the longest frequent pattern in the time series S is l_m. If the confidence of a length i pattern is less than $\lambda_{l_m,i} \cdot \rho_c$, we can stop developing the pattern. This requires that the user has a rough idea about the value of the l_m, and we can guarantee that all frequent patterns of length less than or equal to l_m will be discovered. There exist some periodicity detection methods [29] [30] [31] [32] that can be used to determine the length of the longest pattern. Users can also estimate the pattern length that is meaningful (e.g., a length of 500 ms). Since a CR device needs to estimate the channel state instantly, the rule set cannot be too large. A maximum length of tens of bits is usually sufficient. Section 5 also introduces a duration mining method that can be used to estimate the l_m.

4.3.2 Backward-extension constraint

To further reduce the number of candidate frequent patterns that need to be examined, we identify patterns that can be absorbed by other patterns. Suppose $P = P[1]P[2]...P[l]$ is a pattern of length l in a time series S. Given another pattern $C = p_0P$ where $p_0 \in I$, if $sup(P) = sup(C)$, we say p_0 is a *backward-extension item* of P. If P has a backward-extension item p_0, it can be absorbed by C. For example, if there is a '0' before any occurrence of '110', the pattern $\langle 110 \rangle$ can be absorbed by $\langle 0110 \rangle$ because any pattern developed under the tree of '110' has the the prefix '0110'. It is redundant to develop both of them (i.e., '110' and '0110').

To check the backward-extension item of a pattern P, we extract the head indices from $HIL(P)$ and examine whether the values located one symbol before all occurrences of P are the same value in I. Since $I = \{0, 1\}$, we use summation to check the condition.

As shown in Algorithm 3, when we count the support of a pattern P, we also add up the value that is one symbol ahead of each matched subsequence. If the sum is 0, it implies that there is a '0' before any occurrence of P. If the sum is equal to $len(HIL(P))$, it implies that there is a '1' before any occurrence of P. If there exists a $x = 1$ in $HIL(P)$, we reach the boundary and we can regard that there is a '1' before any occurrence of P if the sum is equal to $len(HIL(P))-1$. In any of the three cases, the pattern P can be absorbed by another pattern and we skip developing the subtree of P. The pruning method

Algorithm 3: Backward-extension check.

```
Cnt ← 0;
boundary ← False;
for x ∈ HIL(P) do
    if x ≥ 2 then
        Cnt ← sum (Cnt, S[x − 1])
    else                          /* reach boundary */
        boundary ← True
if Cnt == 0 or Cnt == len (HIL(P)) or (Cnt == len
(HIL(P))-1 and boundary) then
    P can be safely pruned;
```

is very efficient because if we use a simple candidate-maintenance-and-test method, in a case where $\langle 10 \rangle$ can be absorbed by $\langle 110 \rangle$, $\langle 110 \rangle$ must be mined before $\langle 10 \rangle$ so that $\langle 10 \rangle$ can be compared with $\langle 110 \rangle$. However, the subtree of $\langle 10 \rangle$ is developed before $\langle 11 \rangle$ as shown in Fig. 2. Therefore, a candidate-maintenance-and-test method will fail to remove the redundancy. On the contrary, in our method, we can discover that $\langle 10 \rangle$ can be absorbed by $\langle 110 \rangle$ even though we do depth-first search for $\langle 10 \rangle$ ahead of $\langle 11 \rangle$. The redundant development of $\langle 10 \rangle$ can be avoided as desired.

4.4 Channel State Prediction

The mining algorithm outputs frequent patterns and generates prediction rules at the same time. A prediction rule is defined as $P \Rightarrow C$, where C is a superpattern of P. The confidence of a rule is defined as $conf(P \Rightarrow C) = sup(C)/sup(P)$. For example, if $\langle 0001 \rangle$ appears 1000 times and $\langle \underline{0001}0 \rangle$ appears 926 times, the confidence of the prediction rule $\langle 0001 \rangle \Rightarrow \langle \underline{0001}0 \rangle$ is 92.6%, which declares that the channel state is predicted to be idle in the next slot with a confidence of 92.6% when channel states $\langle 0001 \rangle$ are observed. In the following discussions, we present a rule $P \Rightarrow C$ as $P \Rightarrow c$, where $C = P \oplus c$.

Both partial periodic patterns and full periodic patterns are discovered in PPPM. To improve prediction efficiency by reducing the number of rules, we examine whether a rule can be absorbed by another rule. Suppose two rules $P_1 \Rightarrow c$ and $P_2 \Rightarrow c$ yield the same prediction of c. If they have the same length, and $P_1[i] \in P_2[i]$ for each position i, and $conf(P_1 \Rightarrow c) \leq conf(P_2 \Rightarrow c)$, the first rule can be absorbed by the second rule. Fig. 3 shows an example. Note that the fusion cannot be performed if $conf(R_1) > conf(R_2)$; otherwise if there exist a rule $conf(R_3) > conf(R_2)$, we will incorrectly adopt the rule R_3 when $\langle 01010 \rangle$ is observed because R_3 has a higher confidence than R_2. We should keep the rule R_1 so that we can get the correct prediction of '1' from R_1 instead of '0' from R_3.

In the prediction stage we start with patterns of the longest length. If observed channel states match the pattern P in a rule $P \Rightarrow c$, we predict the channel state in the next slot as c. Because a wild card $*$ matches

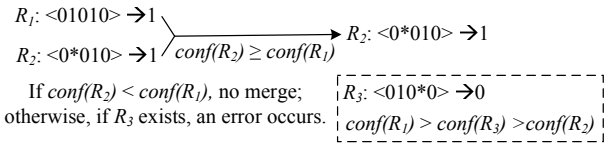

Fig. 3: Rules can be merged under certain conditions.

any value, it is possible that patterns from multiple rules are matched. In such a case, we adopt the rule with the largest number of matched values. If several patterns have the same number of matched values, we adopt the rule with the highest confidence. We stop searching for rules of a shorter length once a prediction can be made. The reason is that a longer match usually leads to a more accurate prediction. A shorter pattern may provide a prediction of a higher confidence, but it includes contributions from multiple long patterns. For example, $\langle 010 \rangle$ appears whenever $\langle 1\underline{010} \rangle$ or $\langle 0\underline{010} \rangle$ appears. When channel states '1010' are observed, it is better to use the rule $\langle 1010 \rangle$ instead of $\langle 010 \rangle$.

If no pattern can match the observed channel states, we do not predict and regard the slot as a miss. We define **prediction accuracy** as the ratio between the number of correctly predicted slots and the total number of predicted slots. The ratio between the number of unpredictable slots and the total number of slots is defined as the **miss rate**.

5 Prediction of Duration

The channel state prediction predicts the channel state in the next slot. The prediction accuracy drops quickly when we try to predict more slots because we use predicted slots to perform further prediction. Therefore, channel state prediction is limited to answer whether the next slot is available. To view a bigger picture, we can make slot size much larger but this sacrifices spectrum efficiency as any activity detected in a slot marks the slot as occupied. We thus convert a binary time series of small slots to a duration sequence.

We count the number of consecutive '1' and use a positive number to represent it. Similarly, we count the number of consecutive '0' and use a negative number to represent it. As values besides '0' and '1' are contained in the converted sequence, we extend our 3-state (i.e., busy, idle, uncertain) mining algorithm to an N-state mining algorithm as follows.

5.1 Duration pattern mining

Because any number may appear in the converted duration sequence, a specific pattern usually has a very low support. We cannot use a wild card that matches any duration to address the variation because two durations of a great disparity may belong to two different patterns (e.g., we cannot regard $\langle 3, -14, 7 \rangle$ and $\langle 3, -14, 170 \rangle$ as one pattern $\langle 3, -14, * \rangle$). Therefore, we define a variation threshold ε to aggregate similar sequences, making frequent patterns rise quickly.

When we count the support for a pattern P of length l, we consider that a subsequence S_i matches P as long as the value difference in each position is less than ε. For example, if $\varepsilon = 1$, subsequence $S_i = \langle 3, -14, 7 \rangle$ matches the pattern $\langle 3, -14, 8 \rangle$. Due to the variation tolerance, several close values have the similar supports. As an example, when we are developing the subtree of a pattern $\langle 3, -14 \rangle$, we may find that the value of the next position can be either '7' or '8'. Because we count occurrences of '7' as occurrences of '8' and vice versa, $\langle 3, -14, 7 \rangle$ and $\langle 3, -14, 8 \rangle$ will have the same support. To avoid redundancy, we only develop the pattern with the largest actual occurrences. For example, if $\langle 3, -14, 7 \rangle$ appears 10 times and $\langle 3, -14, 8 \rangle$ appears 26 times, we only develop the pattern $\langle 3, -14, 8 \rangle$ which has the largest actual occurrences. The ε allows some patterns to pass the confidence threshold with one representative.

In duration pattern mining, we stop developing a pattern if its occurrences are less than min_sup. Because each value in a duration pattern represents a number of slots, even a short duration pattern can span a long period of time. In a finite period of observation, duration patterns that can be observed multiple times are usually short patterns. Assuming a sufficiently large value (e.g., $l_m = 100$) for the length of the longest duration pattern, we have $min\_sup \geq N_{l_m} \times \rho_c$, which ensures that the longest frequent pattern ($l <= l_m$) will be identified. The mining outputs all frequent duration patterns that are less than or equal to l_m. The sum of the absolute values in a duration pattern gives the period of a possible micro pattern. The largest sum can be used as the l_m defined in Theorem 1 for micro channel state pattern mining.

Algorithm 4 summarizes the duration pattern mining. The algorithm scans the duration sequence S once to identify all length-1 frequent patterns. In the ith iteration, the algorithm generates a length-$(i+1)$ candidate pattern P by joining a length-i frequent pattern Q with a length-1 frequent pattern p. Because the depth-first search is used, the algorithm just checks the subsequences where the length-i pattern Q occurs. If the $(i+1)$th value s_{i+1} of a subsequence is within the variation threshold of p (i.e., $||s_{i+1}| - |p|| \leq \varepsilon$), the subsequence is a match for P.

Algorithm 4: Duration pattern mining.

```
scan S to find all length-1 frequent patterns;
F_1 ← the set of all length-1 frequent patterns;
let C be a stack;
for each pattern p in F_1 do
    C.push(p);
while C is not empty do
    Q ← C.pop();
    for each pattern p in F_1 do
        P ← Q ⊕ p;
        if P is frequent then
            C.push(P);
```

5.2 State duration prediction

With the mined duration patterns, the remaining idle time of a channel can be predicted. A CR device can adapt its packet size according to the estimation so as to reduce short yet high interference upon PUs' return.

The duration pattern mining can also be used to predict high/low utilization duration. A CR device may want to know the remaining low utilization duration of current channel so that it can plan its channel switch before the current channel becomes busy. A CR device may also want to know the low utilization duration of a candidate channel before it switches to the channel. To maintain statistical information about backup channels, a CR device can check backup channels periodically or use a secondary radio to sweep a large band of spectrum. Because two subsequent measurements in a backup channel can be several milliseconds or seconds apart, we cannot detect activities between two subsequent measurements but we can use the rough measurements to estimate the channel utilization.

Suppose a channel is considered good for use when the channel utilization is below a certain threshold δ. We calculate the channel utilization of a channel every minute. If the channel utilization is greater than δ, we output a '1', otherwise we output a '0'. The binary time series is then converted to a duration sequence and the duration patterns are mined as described above.

Because there are too many possible values following a duration pattern, we cannot predict a specific value with high confidence. Therefore, we predict the longest duration whose confidence is higher than a threshold ρ_d. The confidence of a certain duration x is defined as the probability of obtaining such a duration, which is $P(X \geq x) = \sum_{|x_i| \geq x} P(X = x_i) = \sum_{|x_i| \geq x} p(x_i)$. For example, following $\langle 3, -14 \rangle$, '7' appears 3 times, '8' appears 60 times, and '9' appears 7 times. The probability of obtaining a duration of 9 is $P(X \geq 9) = 10\%$, the probability of obtaining a duration of 8 is $P(X \geq 8) = 96\%$, and the probability of obtaining a duration of 7 is 100%. If $\rho_d = 0.9$, we predict there exists a duration of 8 following the observation of $\langle 3, -14 \rangle$ because it is the longest duration whose confidence is higher than the ρ_d. Since a CR device is able to estimate the low utilization duration of a candidate channel, it can choose the channel with the longest low utilization duration to use. This reduces the frequency of channel switch.

6 Performance Study

In this section, we compared the performance of our PPPM with the FPM [5] where patterns must be full periodic. We show that more prediction rules are extracted from patterns mined by PPPM. We also compared the PPPM with the HMM model [13] to demonstrate better prediction performance. Further, comparing PPPM with the optimal statistical knowledge-based dynamic spectrum access strategy [33], we show that the spectrum extraction rate is significantly improved with prediction.

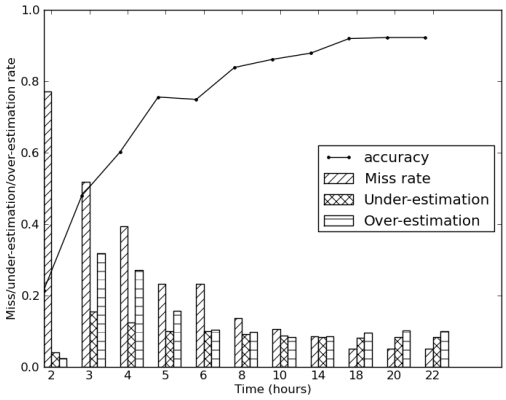

Fig. 4: The duration prediction results in the paging bands.

To evaluate the performance, we measured network activities in the Personal Communication Service (PCS) bands and the paging bands. Data collection and preprocessing are presented in the supplement.

6.1 Macro patterns

To mine duration patterns, we use the slow changing signals measured in the paging bands for study. These signals appear for hundreds of milliseconds and vanish for up to tens of seconds. The slow changing signals are used to simulate patterns that may span a long period of time (e.g., utilization changes). One day's data are available for training and one day's data are used for test. We check whether there exist frequent patterns and whether they can be identified with data collected in several hours. The prediction performance based on mined rules is presented in Fig. 4.

Prediction rules: When a pattern is observed, we predict that there is a length l_e of consecutive busy or idle states. If the actual length is l_t and it is equal to or greater than the predicted l_e, we count it as a correct estimation because there exists such a length (l_e) of consecutive busy or idle states. If the actual length l_t is longer than the predicted l_e, we count it as an underestimation and the underestimation rate is defined as $(l_t - l_e)/l_t$. If the actual length l_t is shorter than the l_e, we count it as an error and an overestimation with an overestimation rate of $(l_e - l_t)/l_e$. The underestimation and overestimation rates indicate the degree of errors, otherwise prediction accuracy can be increased by preferring predicting shorter lengths. If no rule can be found, we count it as a miss and an error.

A pattern may span several seconds or minutes. The occurrences are rare in a short time observation. Therefore, only a few frequent patterns can be identified with data collected in several hours. Because the number of prediction rules is small, the miss rate is high. Since a miss is also counted as an error, the prediction accuracy is low. The underestimation rate and the overestimation rate are low because only a few predictions are made.

If the observation time is long enough, frequent pattern may have appeared many times and made them

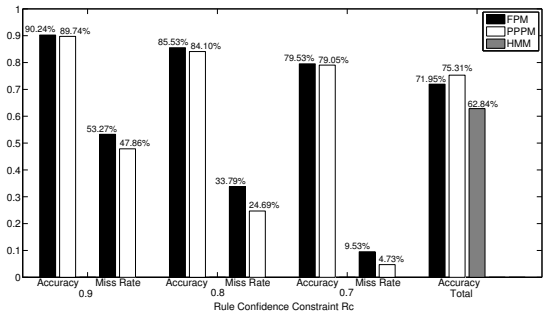

Fig. 5: The channel state prediction results under different rule confidence constraints. The total accuracy takes all missed slots as wrong predictions.

distinguishable. With data collected for more than 18 hours, most rules can be mined and the prediction accuracy becomes stable at 92% with an underestimation rate of 8%, an overestimation rate of 10%, and a miss rate of 5%. We can add more training data, but the impact is trivial as shown in Fig. 4.

Although we need data of at least 18 hours here, only the short duration sequence instead of the long binary time series is needed. The memory requirement is not high. The duration pattern mining gives us a way to predict the idle duration of a channel. It is also useful for discovering the maximum pattern length of micro patterns. Any macro pattern $\langle a, b, c\rangle$ indicates the existence of a micro pattern of length $l = |a| + |b| + |c|$. By examining macro patterns, we can set an appropriate value for l_m that is needed in Theorem 1.

6.2 Micro patterns

To get real-life network activities, we use the network traffic trace collected in the SIGCOMM'08 [34] as a reference. We divide the time into slots of 20 ms and convert the trace into a binary time series of channel states: if channel activities are detected in a slot, we output a '1'; otherwise we output a '0'. There is a beacon every 100 ms and thus 5 bits cover a beacon interval. Depending on the application, the slot size may vary. When a CR device utilizes a spectrum hole, it may desire that the slot is long enough for at least one transmission. Therefore, from a practical point of view, an entire slot may be considered as busy if channel activities are detected. The duration pattern mining shows that it is sufficient to set $l_m = 30$ in micro pattern mining, which covers six beacon intervals.

To investigate the prediction accuracy and the miss rate, we use one day's data for training and one day's data for test. Fig. 5 summarizes the prediction results under different rule confidence constraints (R_c). In order to bound the prediction accuracy, only rules of confidence that is higher than R_c are used. Generally, PPPM reduces the miss rate by around $5\% \sim 9\%$ with a sacrifice of about 1% on prediction accuracy. The lower miss rate implies that more useful rules are extracted from patterns mined by PPPM, which validates that some

```
P 00001000010000110  S0   0.939169  len 17
P 00001101011***1***0  S1   0.765957  len 19
P 00001 0000 *10***        S1   0.814815  len 15
```

Fig. 6: Examples of frequent patterns.

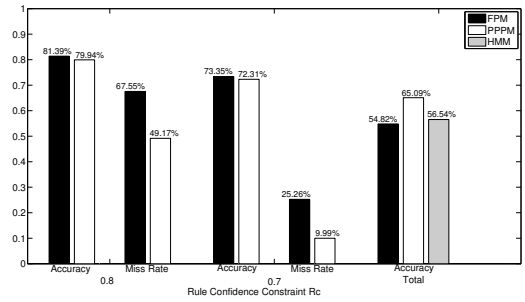

Fig. 7: The channel state prediction results in the Personal Communication Service (PCS) bands.

patterns exhibit partial periodicity only and they cannot be discovered by FPM.

Fig. 6 shows some frequent pattern examples. The 1st one is a full periodic pattern. The 2nd pattern shows that the beaconing is quite regular regardless of channel activities. However, FPM may miss these patterns as they have uncertainty in some slots. The 3rd pattern shows that the beaconing may not occur exactly every 100 ms. It may be delayed by some activities. If this does not happen often, FPM may not take the offset of beaconing as a frequent pattern. When $\langle 0000100000\rangle$ is observed, there is no rule for prediction. In PPPM, a partial periodic pattern $\langle 000010000 * 1\rangle$ may provide a correct prediction.

Fig. 5 also shows that the miss rate is significantly reduced when rules of lower confidences are used. It implies that most rules have confidences between 0.7 and 0.9. The prediction accuracy thus will be pulled down to around 80% if most slots are predicted. Because the unpredictable slots cannot be utilized, we may consider the accuracy as the ratio between the number of correctly predicted slots and the total number of slots (instead of predicted slots). HMM always gives a prediction. If the probability of obtaining 1 is higher than that of 0, the channel state is predicted to be busy, otherwise the channel state is predicted to be idle. The corresponding **total accuracy** shows that PPPM is the best as it masks the irregularity with wild cards, whereas both FPM and HMM try to capture the channel state in every slot.

Fig. 7 summarizes the prediction results in the PCS bands. Different from Wi-Fi channels, there is no explicit periodicity of channel activities. Therefore, few rules have confidence of prediction higher than 0.9 and we have to loosen the rule confidence constraint. The miss rate is reduced by around 40% if rules of confidence between 0.7 and 0.8 are allowed to use. This indicates that there exist some patterns in activities that seem to be random. They are just obscure and hard to identify. PPPM shows its advantage in discovering patterns in true traffic. The miss rate of PPPM is $16\% \sim 18\%$ lower than that of FPM. The total accuracy of PPPM is higher

than that of FPM because PPPM predicts much more slots with only a slight drop on accuracy.

In the supplement, we investigate factors that impact prediction performance, provide analysis of overhead, and discuss advantages of integrating prediction in dynamic spectrum access.

7 Conclusion

In the paper, a spectrum occupancy prediction model based on Partial Periodic Pattern Mining (PPPM) is introduced. The mining takes into account the irregularity of spectrum use and thus is more suitable for true traffic. The proposed PPPM algorithm combines the gap-constrained pattern growth, the head index list structure, the Apriori-like property, and the backward-extension pruning to achieve fast and reliable partial periodic pattern mining. To estimate channel utilization duration, the three-state mining algorithm is extended to an N-state mining algorithm.

The partial periodicity of spectrum use and the performance of PPPM are validated with real Wi-Fi network activities and data collected in the PCS bands. PPPM extracts more channel state prediction rules, leading to a significant reduction on miss rate in spectrum occupancy prediction in comparison with traditional FPM. The PPPM also outperforms HMM for being more robust to irregularity. We particularly observed that distinguishing low utilization periods from high utilization periods and mining rules in corresponding utilization periods will substantially improve the prediction performance.

Although there are many underutilized spectrum chunks, the actual number of spectrum holes that SUs can utilize are few due to the PUs' tight interference constraints. We compared the PPPM-assisted DSA with a statistical knowledge based DSA to demonstrate that prediction of channel states significantly improves the spectrum extraction rate without introducing significant interference to PUs. We also demonstrated that the prediction of high/low channel utilization duration reaches an accuracy of 92% with a miss rate of 5% using data collected in the paging bands.

References

[1] "General survey of radio frequency bands (30 MHz to 3 GHz): Vienna, Virginia, September 1-5, 2009," http://www.sharedspectrum.com/papers/spectrum-reports/.

[2] S. Haykin, "Cognitive radio: brain-empowered wireless communications," *IEEE J. Sel. Areas Communications*, vol. 23, no. 2, pp. 201–220, Feb. 2005.

[3] I. F. Akyildiz, W.-Y. Lee, M. C. Vuran, and S. Mohanty, "Next generation/dynamic spectrum access/cognitive radio wireless networks: A survey," *Computer Networks*, vol. 50, no. 13, pp. 2127–2159, Sep. 2006.

[4] "FCC-10-174," http://www.fcc.gov/Daily_Releases/Daily_Business/2010/db0923/FCC-10-174A1.pdf.

[5] D. Chen, S. Yin, Q. Zhang, M. Liu, and S. Li, "Mining spectrum usage data: a large-scale spectrum measurement study," in *Proc. MobiCom*, 2009, pp. 13–24.

[6] J. Han, G. Dong, and Y. Yin, "Efficient mining of partial periodic patterns in time series database," in *Proc. ICDE*, 1999, pp. 106–115.

[7] P. Huang, C.-J. Liu, L. Xiao, and J. Chen, "Wireless spectrum occupancy prediction based on partial periodic pattern mining," in *Proc. MASCOTS*, 2012, pp. 51–58.

[8] Z. Wen, T. Luo, W. Xiang, S. Majhi, and Y. Ma, "Autoregressive spectrum hole prediction model for cognitive radio systems," in *Proc. ICC Workshops*, 2008, pp. 154–157.

[9] J. Su and W. Wu, "Wireless spectrum prediction model based on time series analysis method," in *Proc. CoRoNet*, 2009, pp. 61–66.

[10] C.-J. Yu, Y.-Y. He, and T.-F. Quan, "Frequency spectrum prediction method based on EMD and SVR," in *Proc. ISDA*, 2008, pp. 39–44.

[11] Z. Wang and S. Salous, "Spectrum occupancy statistics and time series models for cognitive radio," *J. of Signal Processing Systems*, vol. 62, no. 2, pp. 145–155, Feb. 2011.

[12] I. A. Akbar and W. H. Tranter, "Dynamic spectrum allocation in cognitive radio using hidden Markov models: Poisson distributed case," in *Proc. SoutheastCon*, 2007, pp. 196–201.

[13] C.-H. Park, S.-W. Kim, S.-M. Lim, and M.-S. Song, "HMM based channel status predictor for cognitive radio," in *Proc. APMC*, 2007, pp. 1–4.

[14] V. K. Tumuluru, P. Wang, and D. Niyato, "Channel status prediction for cognitive radio networks," *Wireless Commun. and Mobile Comput.*, vol. 12, no. 10, pp. 862–874, Jul. 2010.

[15] C. Ghosh, C. Cordeiro, D. P. Agrawal, and M. B. Rao, "Markov chain existence and hidden Markov models in spectrum sensing," in *Proc. PerCom*, 2009, pp. 1–6.

[16] Z. Chen and R. C. Qiu, "Prediction of channel state for cognitive radio using higher-order hidden Markov model," in *Proc. SoutheastCon*, 2010, pp. 276–282.

[17] C. Li and J. Wang, "Efficiently mining closed subsequences with gap constraints," in *Proc. SDM*, 2008, pp. 313–322.

[18] B. Ding, D. Lo, J. Han, and S.-C. Khoo, "Efficient mining of closed repetitive gapped subsequences from a sequence database," in *Proc. ICDE*, 2009.

[19] M. Zhang, B. Kao, D. W. Cheung, and K. Y. YIP, "Mining periodic patterns with gap requirement from sequences," *ACM Trans. Knowledge Discovery from Data*, vol. 1, no. 2, Aug. 2007.

[20] X. Zhu and X. Wu, "Mining complex patterns across sequences with gap requirments," in *Proc. IJCAI*, 2007, pp. 2934–2940.

[21] W. G. Aref, M. G. Elfeky, and A. K. Elmagarmid, "Incremental, online, and merge mining of partial periodic patterns in time-series databases," *IEEE Trans. Knowledge and Data Engineering*, vol. 16, no. 3, pp. 332–342, Mar 2004.

[22] N. Chomsky, "Three models for the description of language," *IRE Transactions on Information Theory*, vol. 2, no. 3, pp. 113–124, 1956.

[23] M. A. Harrison, *Introduction to Formal Language Theory*, 1st ed. Addison-Wesley Longman Publishing Co., Inc., 1978.

[24] M. Nowak, *Evolutionary Dynamics: Exploring the Equations of Life*. Belknap Press, 2006.

[25] D. Jurafsky and J. Martin, *Speech And Language Processing: An Introduction to Natural Language Processing , Computational Linguistics, and Speech Recognition*, ser. Prentice Hall Series in Artificial Intelligence. Prentice Hall, 2009.

[26] S. Geirhofer, L. Tong, and B. Sadler, "A measurement-based model for dynamic spectrum access in WLAN channels," in *Proc. MILCOM*, 2006, pp. 1–7.

[27] J. Huang, G. Xing, G. Zhou, and R. Zhou, "Beyond co-existence: Exploiting WiFi white space for ZigBee performance assurance," in *Proc. ICNP*, 2010.

[28] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," in *Proc. VLDB*, 1994, pp. 487–499.

[29] M. G. Elfeky, W. G. Aref, and A. K. Elmagarmid, "Periodicity detection in time series databases," *IEEE Trans. Knowledge and Data Engineering*, vol. 17, no. 7, pp. 875–887, Jul. 2005.

[30] C. Berberidis, W. G. Aref, M. Atallah, I. Vlahavas, and A. K. Elmagarmid, "Multiple and partial periodicity mining in time series databases," in *Proc. ECAI*, 2002.

[31] T. Clancy and B. Walker, "Predictive dynamic spectrum access," in *Proc. SDR Forum Technical Conference*, 2006.

[32] M. Wellens, A. de Baynast, and P. Ma ho nen, "Exploiting historical spectrum occupancy information for adaptive spectrum sensing," in *Proc. WCNC*, 2008, pp. 717–722.

[33] S. Huang, X. Liu, and Z. Ding, "Optimal transmission strategies for dynamic spectrum access in cognitive radio networks," *IEEE Trans. Mobile Computing*, vol. 8, no. 12, pp. 1636–1648, Dec. 2009.

[34] "Anonymized packet traces and AP syslog," http://www.cs.umd.edu/projects/wifidelity/sigcomm08_traces/.

Pei Huang received his BS and MS degrees in Electrical Engineering from Southwest Jiaotong University, China, in 2005 and 2008, respectively. He is currently working towards the PhD degree in Computer Science and Engineering at Michigan State University. His research interests include wireless communication, wireless ad hoc and sensor networks, and cognitive radio networks.

Chin-Jung Liu received his BS degree in Computer Science from National Tsing Hua University in 2007 and his MS degree in Computer Science and Engineering from Michigan State University in 2011. He is currently working towards the PhD degree in CSE at Michigan State University. His research interests include wireless networking, cognitive radio networks, and wireless sensor networks.

Xi Yang received her BS degree in Electrical Engineering and MS degrees in Communication and Information System from Southwest Jiaotong University, China, in 2005 and 2008, respectively. She is currently working towards the MS degree in Computer Science and Engineering at Michigan State University. Her research interests include wireless networking, data mining, and computer vision.

Li Xiao received the BS and MS degrees in Computer Science from Northwestern Polytechnic University, China, and the PhD degree in Computer Science from the College of William and Mary in 2002. She is an associate professor of Computer Science and Engineering at Michigan State University. Her research interests are in the areas of distributed and networking systems, overlay systems and applications, and wireless networks. She is a senior member of the IEEE.

Jin Chen is an Assistant Professor in the MSU-DOE Plant Research Laboratory and the Department of Computer Science and Engieering at Michigan State University. He received his Bachelor degree in Computer Science from Southeast University in China in 1997, and received his Ph.D. in Computer Science from the National University of Singapore in 2007. Before joining MSU, he was a postdoctoral research associate in the bioinformatics laboratory at the Carnegie Institution for Science at Stanford University. His general research interests are in computational biology, as well as its interface with machine learning and algorithms. He is particularly interested in developing data mining and graph mining algorithms for deciphering genomic data, especially algorithms for genome-level analysis of protein/gene functions and interactions.