Optimal Algorithm for Cognitive Spectrum Decision Making

Liao Ming-Xue, He Xiao-Xin, Jiang Xiao-Hong
Institute of Software
Chinese Academy of Sciences
Beijing, China
{mingxue, xiaoxin, xiaohong}@iscas.ac.cn

Abstract—**In this paper, an optimal algorithm of spectrum decision making is presented for a real cognitive radio network with tree-based topology. All nodes of a subnet in such network have the capability in being aware of spectrum information by both energy detector based sensing and centralized cooperative sensing. After gathering sensed information, the master node will decide which frequency can be used by the subnet and which slave nodes should leave the subnet if there is no common frequency among all nodes. The problem is how to keep nodes staying in the subnet as many as possible. Traditionally, this is a combination-optimization problem. By mapping the node set and frequency set to be both parts of a bipartite graph respectively, the problem can be turned into a special case of searching for maximal bicliques. Based on a well-known LCM (Linear time Closed itemset Miner) algorithm, and using some new techniques in terms of dynamic thresholds and efficient management of closeness states, we have solved this problem for our application requiring real-time performance. For some special cases where nodes in a subnet may have different weights, our algorithm can also find an optimal solution with maximal weights in real time.**

Keywords-cognitive radio network; spectrum decision making; maximal biclique; dynamic threshold.

I. INTRODUCTION

In this paper, we propose a new application in CRAHNs (**C**ognitive **R**adio **A**d **H**oc **N**etworks) [1]. Suppose that there is a simple network consisting of a master node and several slave nodes. The constraint in such network is that the slave nodes only communicate with the master node directly and they must use the same channel parameters decided by the master node, such as frequency and power. There are two main steps for the network to complete its spectrum sensing process. According to a dedicated energy threshold, the first step is that all nodes begin a search for idle channels of local electromagnetic environment through energy detection method [2]. Then the idle information will be sent to the master by a control channel. The master then selects the most reliable channels for further examination by a waveform based bidirectional channel test with each slave. This test is a process of centralized cooperative sensing [3] to identify channels of false usefulness and capture truly useable channels. By a channel quality threshold, the master gathers all useful frequency (channel) sets from slave nodes, and then decides to use which frequency for communication and to backup several frequencies for use in future because of high cost of the sensing process. Certainly, the master does not need to backup too many frequencies as these selected frequencies will become stale over a certain long period. However, maybe there is no common frequency to be useful for all nodes since the electromagnetic surroundings, especially in variant terrains, are different here and there. Under this condition, how to choose the slave nodes and the frequency set is the problem called CSDM (**C**ognitive **S**pectrum **D**ecision **M**aking). It is obvious that we need solve CSDM twice during spectrum sensing process in our application.

Traditionally, CSDM is a combination-optimization problem. We can adopt an exhausting algorithm of enumerating all combinations of nodes. If there are n nodes and each slave node have at most m useful frequencies with the mast node, then the algorithm may take time complexity of $O(2^n * m)$ to run. For bigger n or m, such algorithms have no chance to meet real-time requirements of applications.

We can model CSDM by enumerating maximal bicliques from a bipartite graph in which one part represents a node set while the other part represents a frequency set useful for the node set to communicate [4].

Let $G = <V, E>$ be a graph with vertex set V and edge set E. A pair of disjoint nonempty subsets V_1 and V_2 of V is called a biclique if $(u, v) \in E$ for all $u \in V_1$ and $v \in V_2$. Define $\beta(v)$ as the set of all vertices in G that are adjacent to v, i.e., $\beta(v) = \{u | (u, v) \in E\}$. For a nonempty subset X of vertices of a graph, $\beta(X)$ is the set of common neighborhoods of all vertices in X. For an existing biclique sub-graph $H = < V_1 \cup V_2, E >$, the biclique is a maximal biclique if $\beta(V_1) = V_2$ and $\beta(V_2) = V_1$.

Enumerating maximal bicliques from a graph can be one-to-one correspondence with the enumeration of closed pattern pairs [5]. A closed pattern pair is composed of two parts: a frequent closed item set and its support set. Many real-life applications can be modeled by the both conceptions such as associating rule mining, life science data analysis and inductive database [6]. One example is given here. For social relation, common characters of persons can be modeled by maximal bicliques which is useful in commercial activities. Surprisingly this idea has similar scenario in wireless communication filed.

Either enumerating frequent closed item sets or enumerating maximal bicliques are long studied. There are several algorithms for these problems at present, such as CLOSED+ [7], LCM [8][9][10] and [11]. However, all these algorithms are enumerating all either maximal bicliques or closed pattern pairs. For CSDM problem, we are only interested in the best solution defined later in this paper.

Considering good performance of LCM, we choose it as the base of our algorithm.

What discussed above is based on a hypothesis that all nodes have the same importance or that all items are interchangeable. But in real applications, some nodes may play more important role in the network. Some importance is originated from the fact that different nodes have different functions in a concrete application context. Also in a view of a tree-formed network, some node inherently has more importance than other nodes.

Take a two-layer network shown in Figure 1 for example. All nodes in the network have the same importance. Both node B and C want to connect node A, while B has three sub nodes at this time. After a cognitive process, A and B have found two common frequencies available, and A, C have three useful frequencies. But there are no common frequencies among A, B, C. Therefore, the master node A has two options to make: select either B or C. Obviously A will make a decision of deleting B and only keep C in its network, because it choose larger amount of common frequencies when the both options have the same number of node. Now the whole network has only two nodes, A and C. Certainly, this is not the best solution for the two-layer network as a whole. And by keeping B and deleting C, the whole network can have five nodes, which is larger. In this paper we solve this problem through adding a weight property to each node.

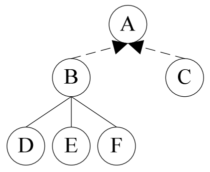


Figure 1. Two-layer network

The remainder of this paper is organized as follows. In Section II, we describe the details of CSDM. In Section III and Section IV, we introduce the algorithm LCM and our algorithm EMBS (**E**xtreme **M**aximal **B**iclique **S**earcher). In Section V, we talk about how to process the case of weighted nodes. The experiments and results are listed in Section VI. The last section will conclude this paper.

II. CSDM Problem

Let $Net=<N \cup F, E_N>$ be a network with a node set N, a frequency set F and a relationship E between both nodes. A pair $(n, f) \in E$ if and only if a node n can use the frequency f to communicate with the master node. Any frequency by a node is detected by a bidirectional wave detection process between the node and the master node. A subset $<N_i \subseteq N, F_i \subseteq F, E_i>$ is a solution to CSDM if all conditions below are satisfied. The condition (2) declares that each node that can use all frequencies in F_i should be in the solution. (3) expresses a similar meaning: each frequency that can be used by all nodes in N_i should be in the solution.

$$N_i \times F_i = E_i \subseteq E_N \tag{1}$$

$$((\forall f \in F_i)\ (n, f) \in E_i) \rightarrow n \in N_i \tag{2}$$

$$((\forall n \in N_i)\ (n, f) \in E_i) \rightarrow f \in F_i \tag{3}$$

Let $G = < V_1 \cup V_2, E_G >$ be a bipartite graph with vertex set V_1, V_2 and edge set E_G. To model the network, let the node set N be V_1 and the frequency set F be V_2. If there is a pair of (n, f) in E_N, add a corresponding edge into E_G. Therefore, the graph can be modified to $G = < N \cup F, E_N >$ and each solution to CSDM is a maximal biclique in G because the conditions satisfied by the solution are the same as to those satisfied by maximal bicliques.

Proof.

First, a maximal biclique $<N_i \subseteq N, F_i \subseteq F, E_i>$ must be a solution to CSDM.

If $(n, f) \in E_i$ for $\forall f \in F_i$, then $n \in \beta(F_i)$, by $\beta(F_i) = N_i$, $n \in N_i$ follows.

If $(n, f) \in E_i$ for $\forall n \in N_i$, then $f \in \beta(N_i)$, by $\beta(N_i) = F_i$, $f \in F_i$ follows.

Second, a solution to CSDM $<N_i \subseteq N, F_i \subseteq F, E_i>$ must be a maximal biclique.

If $f \in \beta(N_i)$, then $\forall n \in N_i$, $(n, f) \in E_i$, $f \in F_i$ follows.

If $f \in F_i$, by $N_i \times F_i = E_i$, then $\forall n \in N_i$, $(n, f) \in E_i$, $f \in \beta(N_i)$.

So $\beta(N_i) = F_i$.

Similarly, $\beta(F_i) = N_i$. □

However, we are only interested in the best solution $B_m=<N_m \cup F_m, E_B>$, i.e., the best maximal biclique (also called **extreme maximal biclique**), satisfying (4) and (5). The condition (4) is a condition to restrict the size of a solution. It states that the solution should have at least n_m nodes and each node should have at least f_m frequencies for communication with the master node. An optimal solution is defined by (5). Naturally, we hope that more nodes can be kept in the network. If two solutions have the same number of nodes, then the solution with more common frequencies is much better.

$$|N_i| \geq T_n,\ |F_i| \geq T_f \tag{4}$$

$$\forall <N_i, F_i>,\ |N_i| < |N_m| \vee (|N_i| = |N_m| \wedge |F_i| \leq |F_m|) \tag{5}$$

III. LCM Algorithm

In this section, LCM algorithm is described in graph format while it is described in the database format in original paper [8]. This work has been done in detail by [5] and here we only list the information needed.

Let $G=<V_1 \cup V_2, E_G>$ be a bipartite graph. For a biclique sub-graph $B = < X \cup Y, E_B>$, the set X and Y are called closed sets if and only if B is a maximal biclique, or else they are called unclosed sets. For a vertex $v \in V_1$, id(v) is the index of v in V_1 which is sorted by $|\beta(v)|$ in decreasing order.

Algorithm LCM()

```
Global:
    a bipartite graph $G$ with vertex set $V_1$ and $V_2$
    $p$ is the threshold of one part in a maximal biclique
    $q$ is the threshold of the other part in the biclique
Description:
1:  sort $\{v \in V_1\}$, $\{v \in V_2\}$ by $|\beta(v)|$ in decreasing order
2:  for each $v \in V_1$, set $flag(v)$=0
3:  $T \leftarrow \Phi$
4:  for each $v \in V_1$
5:      $X \leftarrow \Phi$
6:      if $\beta(v) \geq q$ and LCM_CLOSED($X$, $v$) = 0
7:          then LCM_Iter($X$, $V_2$, $v$) /* $\beta(\Phi)= V_2$ */
```

Algorithm LCM_Iter()

```
Input:
    a vertex set $X$ and $\beta(X)$
    a vertex $v$ to be added to $X$
Description:
1:  $Y \leftarrow X \cup \{v\}$
2:  for each $u \in \{\omega|\omega \in V_1 \wedge \omega \notin Y \wedge \mathrm{id}(\omega) < \mathrm{id}(v)\}$
3:      if $\beta(u) \supseteq \beta(Y)$ then $Y \leftarrow Y \cup \{u\}$
4:  $Z \leftarrow \{\omega|\omega \in V_1 - Y \wedge \mathrm{id}(\omega) < \mathrm{id}(v) \wedge |\beta(\omega) \cap \beta(Y)| \geq q\}$
5:  if $|Y| \geq p$ then output $(Y, \beta(Y))$
6:  if $|Y| + |Z| < p$ then return
7:  for each $\omega \in Z$
8:      if $flag(\omega)$=0 then $r \leftarrow$LCM_CLOSED($Y$, $\omega$)
9:          if $r = 0$ then LCM_Iter($Y$, $\beta(Y)$, $\omega$)
10:         else $flag(\omega) \leftarrow r$
```

Algorithm LCM_CLOSED()

```
Input:
    $X$ is a vertex set and $v$ is a vertex to be added to $X$
Description:
1:  for each $u \in V_1$, $u \notin X$, $u \neq v$
2:      if $\beta(\{u\}) \supseteq \beta(X \cup \{v\})$ then return id($u$)
3:  return 0
```

The pseudo code of LCM is rebuilt from a program [12]. For the bipartite graph G, LCM algorithm will recursively list all size-qualified maximal bicliques in G.

IV. Our Algorithm: EMBS

In this section, we transform LCM into a new algorithm EMBS to search extreme maximal bicliques in a bipartite graph.

A. Dynamic thresholds

In our algorithm, we introduce two new parameters p_m and q_m representing the best maximal biclique to be found currently. In LCM, the thresholds are constant and the algorithm enumerates all maximal biclique not less than the thresholds. In EMBS, only the best maximal biclique will be saved and the thresholds will be dynamically updated according to the maximal biclique found. Figure 2 will show this difference.

In Figure 2, M is a matrix sorted, and m and n are the amount of rows and columns of M respectively. An element at column j of row i means that a node i can use the frequency j to communicate with the master node. (a) of Figure 2 is the pruning tree of LCM and (b) is that of EMBS. The sets in an italic style are the leaves or pruned branches while the boldfaced sets are maximal bicliques found.

From Figure 2 we can see that the enumeration tree of EMBS is smaller than that of LCM. The difference is caused by dynamic thresholds (p_m, q_m). The pair (p_m, q_m) is always (2, 2) in (a) of Figure 2. But in (b) of Figure 2 from EMBS, (p_m, q_m) is changed to (2, 3) when the first maximal biclique is found and it is changed to (3, 3) when the second one is found. Because of the increasing thresholds, more nodes are pruned in (b).

$$M = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}, p=2, q=2, m=6, n=6$$

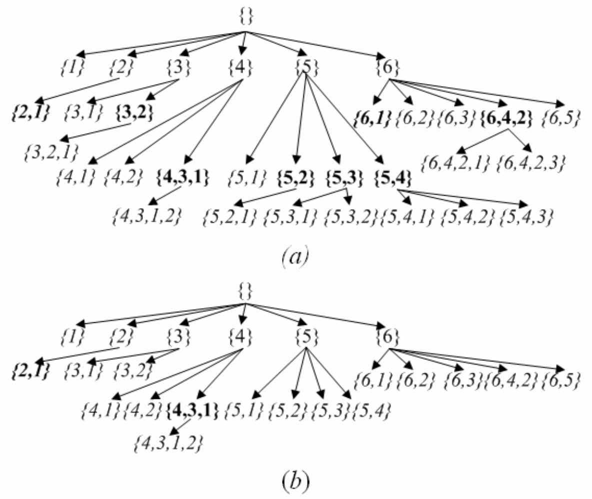


Figure 2. Different pruning tree of LCM and that of EMBS

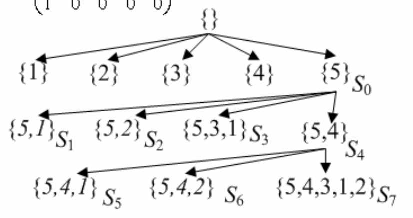
B. Improve the judgment of closed state

Let X be the first parameter of the function LCM_Iter. According to the pseudo code, $|X|$ is increasing continuously in recursive process while $|\beta(X)|$ is decreasing. For parameter v, if there is a vertex u not included by X, id(u) > id(v) and $\beta(X\cup\{u\}) \supseteq \beta(X\cup\{v\})$, then $X\cup\{v\}$ is not closed. If id(u) < id(v) and $\beta(X\cup\{u\}) \supseteq \beta(X\cup\{v\})$, the vertex will be inserted to X with v together. So we do not iterate on vertices with id less than that of v. For each step S' iterated from a step S, let Y be the set of the vertices selected from S to S' and let u be the vertex making X unclosed in S. If $u \notin Y$, $X\cup Y\cup\{v\}$ is also not closed because $\beta(X\cup Y\cup\{u\}) = \beta(X\cup\{u\})\cap\beta(Y)$, $\beta(X\cup\{v\})\cap\beta(Y) = \beta(X\cup Y\cup\{v\})$ and $\beta(X\cup\{u\}) \supseteq \beta(X\cup\{v\})$. So at step S, we can record the vertex making v unclosed and keep this information valid for S' to skip some evaluations on closed states until S returned. But if the vertex u has been inserted into Y, $X\cup Y\cup\{v\}$ may be closed. At this condition, the closeness state of $X\cup Y\cup\{v\}$ should be recomputed.

LCM algorithm uses two arrays *unclosed_u* and *unclosed_v* to manage the closeness states. If u makes v unclosed at step S, set *unclosed_u*[v] = u and *unclosed_v*[*length*] = v while a variable *length* is used to indicate the length of *unclosed_v*. All elements put to *unclosed_v* by step S will be sorted by the corresponding values in *unclosed_u*. When iterating on a vertex i at step S', LCM deletes all values which are not bigger than i from the tail of *unclosed_v* and clear the corresponding values in *unclosed_u* also.

Figure 3 is an example to show this clearly. In Figure 3, M is the input graph and p, q are two thresholds. From S_0 to S_7 is the recursion starting from set {5}.The sets marked italic are unclosed sets. Table in Figure 2 shows the process of transformation of *unclosed_u* and *unclosed_v*. *unclosed_u*[0] is set to 3 at step S_1 and is cleared at step S_3. But at step S_5, *unclosed_u*[1] is recomputed and reset. In fact, the value 3 of *unclosed_u*[1] should be ignored only when the vertex 3 has been in the selected set.

$$M = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}, p = 1, q = 1, m = 6, n = 5$$



Process of transformation of arrays

step	*unclosed_u*	*unclosed_v*	*length*
S_0	0,0,0,0,0	0,0,0,0,0	0
S_1	3,0,0,0,0	1,0,0,0,0	1
S_2	3,4,0,0,0	2,1,0,0,0	2
S_3	0,4,0,0,0	2,0,0,0,0	1
S_4	0,0,0,0,0	0,0,0,0,0	0
S_5	3,0,0,0,0	1,0,0,0,0	1
S_6	3,3,0,0,0	1,2,0,0,0	2
S_7	0,0,0,0,0	0,0,0,0,0	0

Figure 3. An example of the recursion process

In EMBS algorithm, each vertex in V_1 has a stack to manage closed states. Firstly zero is pushed into every stack and supposes that u is the vertex making v unclosed in S_i, u will be pushed into stack of v and it will be valid until u is selected or S_i returns. If S_j is an offspring step iterated from S_i and only if u is selected in S_j, closed state of v will be recomputed, or else there is no any calculation on closed state about v in step S_j. When step S_i returns, every stack changed in S_i will pop the top element. While using stacks to manage closed states, each operation except computing closed state can be completed in $O(1)$ time and the result of computation can be used more effectively. After optimizing, the process of sorting for *unclosed_v* is cut while all redundant recomputation of closed states is reduced.

C. Reduce the size circularly

In LCM algorithm, only the vertices in one part of the graph are reduced by the relationship of the neighborhoods.

In EMBS algorithm, all vertices in both parts of the graph are reduced. After reducing the vertices in one part, the neighborhoods of the vertices in the other part are changed simultaneity. So we should reduce the other part again until all the two parts can not be reduced any more.

Figure 4 shows two matrices reduced by LCM and EMBS. M is the original matrix. M_1 is the matrix reduced by LCM and M_2 is the matrix reduced by EMBS. It shows that the matrix reduced by EMBS is much smaller than the matrix done by LCM. The sixth column of M is eliminated by LCM while the last two columns and the last two rows of M are removed by EMBS.

$$M = \begin{pmatrix} 1 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{pmatrix}, p = 2, q = 2, m = 6, n = 6$$

$$M_1 = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}, p = 2, q = 2, m = 6, n = 5$$

$$M_2 = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, p = 2, q = 2, m = 4, n = 4$$

Figure 4. Matrices processed by LCM and EMBS

D. The algorithm

Detailed algorithm is described below.

Global variables:

R is one part of the result biclique.

$\beta(R)$ is the other part of the result biclique.

p_m is the current threshold of R(main threshold), corresponding to T_n in Section II.

q_m is the current threshold of $\beta(R)$, corresponding to T_f in Section II.

p is the initial threshold of R(main threshold)

q is the initial threshold of $\beta(R)$

Algorithm EMBS()

Input:

A bipartite graph G with vertex sets V_1, V_2

Description:

```
1:    $p_m \leftarrow p$ , $q_m \leftarrow q$
      /* reduce matrix circularly */
2:    while $\exists v \in V_1 \rightarrow |\beta(v)| < p_m \vee \exists v' \in V_2 \rightarrow |\beta(v)| < q_m$
```

```
3:          then $V_1 \leftarrow V_1 - \{v\}$, $V_2 \leftarrow V_2 - \{v'\}$
4:    sort $\{v \in V_1\}$, $\{v \in V_2\}$ by $|\beta(v)|$ in decreasing order
5:    initialize a stack for every $v \in V_1$, $stack[v]$.push(0)
6:    for each $v \in V_1$
7:       $X \leftarrow \Phi$
8:       if $\beta(v) \geq q_m$ and LCM_CLOSED($X$, $v$) = 0
9:       then EMBS_Iter($X$, $V_2$, $v$) /* $\beta(\Phi) = V_2$ */
10:   return $<R, \beta(R)>$
```

Algorithm EMBS_Iter()

Input:

X is a vertex set

$\beta(X)$ is the neighbourhood of X

v is the vertex to be added to X

Description:

```
1:    $Y \leftarrow X \cup \{v\}$
2:    for each $u \in \{\omega | \omega \in V_1 - Y \wedge \mathrm{id}(\omega) < \mathrm{id}(v)\}$
3:        if $\beta(u) \supseteq \beta(Y)$   then $Y \leftarrow Y \cup \{u\}$
4:    $Z \leftarrow \{\omega | \omega \in V_1 - Y \wedge \mathrm{id}(\omega) < \mathrm{id}(v) \wedge |\beta(\omega) \cap \beta(Y)| \geq q_m\}$
5:    if $|Y| + |Z| < p_m$   then return
6:    if $(|Y| > p_m) \vee (|Y| = p_m \wedge |\beta(Y)| \geq q_m)$ /* see (5) */
      /* Dynamic thresholds */
7:    then $<R,\beta(R)> \leftarrow <Y,\beta(Y)>$, $<p_m,q_m> \leftarrow <|Y|,|\beta(Y)|+1>$
8:    $T \leftarrow \Phi$
9:    for each $\omega \in Z$
10:     if $stack[\omega]$.peek() = 0 $\vee$ $stack[\omega]$.peek() $\in Y$
11:       then $r \leftarrow$ LCM_CLOSED($Y$, $\omega$)
12:         if $r = 0$  then EMBS_Iter($Y$, $\beta(Y)$, $\omega$)
13:         else $stack[\omega]$.push($r$), $T \leftarrow T \cup \{\omega\}$
14:   for each $\omega \in T$   $stack[\omega]$.pop()
```

The 7th line of function EMBS_Iter is to update thresholds dynamically. In EMBS algorithm, the both line 2 and 3 are to reduce the size of the matrix circularly, and lines between 9th and 14th of function EMBS_Iter represent the improvement for judgment of closed state.

V. WEIGHTED CASE OF CSDM

In this section, we talk about how to process the case of weighted nodes in CSDM problem. We call such case *w*CSDM (weighted CSDM).

A. *Description of w*CSDM

In the $Net = <N \cup F, E_N>$, we can put a weight property to each node $n \in N$, denoted by $w(n)$. And we denote the sum of weight of nodes in N as $w(N)$. A best solution to *w*CSDM $B_m = <N_m \cup F_m, E_B>$ should satisfy the condition (6) besides (1)~(4) in Section II.

$$\begin{aligned} &\forall <N_i \cup F_i, E_i> w(N_i) < w(N_B) \vee \\ &(w(N_i) = w(N_B)) \wedge |N_i| < |N_m|) \vee \\ &((w(N_i) = w(N_B) \wedge |N_i| = |N_m| \wedge |F_i| \leq |F_m|) \end{aligned} \tag{6}$$

By definition, the best solution is also a maximal biclique. Therefore, the algorithm EMBS for CSDM is suitable for *w*CSDM except for some different techniques to prune the enumeration tree.

B. *Strategy for sorting nodes*

Different from the line 4 of the algorithm EMBS, a new strategy for sorting nodes is presented as below.

4: Sort $\{v \in V_1\}$ by $w(v)$ in decreasing order, and sort $\{v \in V_2\}$ by $|\beta(v)|$ in decreasing order

When weight differences between nodes are big, sorting by $w(v)$ can speed up pruning process because a solution with big weights will be found earlier. If the weight differences are not very obvious, and if we still sort them such a way, then the solution with more nodes will not be found earlier because they may have almost the same weight as that of other solutions. Therefore, we need sort $\{v \in V_1\}$ by $|\beta(v)|$ in decreasing order if there is no noticeable weight differences among nodes.

C. *Pruning Strategy*

Forecasted weight strategy

For an enumeration over a node combination $X = \{v \in V_1\}$, if $w(X \cup Z) < W$ where W is the sum of weights of the solution found earlier and Z is the set of all nodes in V_1 after X, then we need no further depth-first enumerations branched from X.

Closeness strategy

For an enumeration over a node combination $X = \{v \in V_1\}$, if X is unclosed, then we only need execute a calculation over $X \cup Z$ where Z is the set of all nodes that make X unclosed. If $X \cup Z$ is a solution better than the solution found before, then we can update the current solution and all thresholds. After this calculation, we rapidly return with no more iteration on $X \cup Z$.

If X is closed, whether we need iterate over a superset of X is determined firstly by forecasted weight strategy. Then if $|\beta(X) \leq q|$, it is useless to iterate because any superset of X will violate the rule by the q threshold defined in algorithm EMBS.

VI. EXPERIMENT AND RESULT

We evaluate the efficiency of EMBS with dynamic thresholds by running on different size of graphs and evaluate that of EMBS without dynamic thresholds by comparing it to LCM algorithm. The experiments are conducted on randomly generated matrices representing bipartite graphs. Our computer for experiment is a PC with a 3.0GHz CPU and 1GB of memory.

Table I shows the performance of EMBS with dynamic thresholds on randomly generated bipartite graphs in different size. Row one is the size of the graphs and m+n means that there are m vertices in the part representing nodes and n vertices in the other part representing frequencies. Row two is edge density of the graphs. If there are m+n vertices and w edges in the graph, the edge density will be calculated by $w/(m*n)$. The first column is the threshold of frequencies while the threshold of nodes is 1. The number in the table is time in milliseconds and data of first two size graphs are in integral number while others maintain two digits after decimal point.

To evaluate the efficiency of EMBS with dynamic thresholds, we use four different sizes of graphs to represent different size of subnet. The biggest value of threshold of frequencies is eight because we only need to keep at most eight frequencies to use over a certain long period. The threshold of nods means that one node is needed at least. We can find that the running time of EMBS with dynamic thresholds is below one second at most times. This performance meets the real-time requirements of our applications. In some cases, the running time is still very long. However, these cases are very rare in real applications.

Table II shows performance of both LCM and EMBS without dynamic thresholds on different number of vertices and edge density. At each row of the table, the performance is averaged over five randomly generated graphs of the same vertices and edge density. The thresholds are both one in this case. Note that both LCM and EMBS here are searching for all maximal bicliques (complete bipartite graphs) not only for the extreme maximal biclique. The first column of the table is the amount of nodes in each part of the graph. The second column is the edge density in the graph. The third column is the amount of all maximal bicliques ever found. The maximal bicliques found by LCM and those found by EMBS without dynamic thresholds are the same. The forth and fifth columns are running time of LCM and EMBS while the sixth column is the ratio of data in the fifth column and data in the forth column. The last column denotes the performance improvement of EMBS, and obviously EMBS without dynamic thresholds performs better than LCM according to Table II. The first reason is that we reduce the time for judgment of closed state, though the pruning tree of EMBS and that of LCM are the same. The second reason is that EMBS can reduce the graph better than LCM, especially when the edge density of graph becomes little.

For the case of weighted CSDM, we slightly transform EMBS to a new version called *w*EMBS benefiting from pruning condition appeared in the Section V. With more pruning conditions but with more calculations related to weight, it is not a surprise that the performance of *w*EMBS is only somewhat faster than EMBS, as shown in Table III. However, this performance is sufficient for our application as that of EMBS and the performance instability problem in [13] is also solved.

TABLE I. PERFORMANCE OF EMBS WITH DYNAMIC THRESHOLDS ON RANDOM BIPARTITE GRAPHS.

Vertices	**64+512**					**32+256**					**16+128**					**8+64**				
Density / Threshold	***10***	***30***	***50***	***70***	***90***	***10***	***30***	***50***	***70***	***90***	***10***	***30***	***50***	***70***	***90***	***10***	***30***	***50***	***70***	***90***
1	40	34	47	55	87	29	4	7	11	17	1.73	1.39	1.80	2.55	2.82	0.50	0.64	0.77	0.81	0.89
2	5	34	47	55	87	2	5	7	12	17	0.77	1.47	1.79	2.42	4.46	0.49	0.66	0.76	0.82	0.86
3	8	52	92	76	89	2	4	9	15	17	0.77	1.51	2.11	3.18	2.89	0.52	0.70	0.71	0.77	1.49
4	15	144	493	489	95	3	11	16	24	17	0.99	2.02	3.30	5.36	2.81	0.52	1.98	0.78	0.77	0.85
5	18	356	2,099	3,486	109	3	13	51	34	17	1.06	1.79	3.66	5.08	2.72	0.51	0.75	0.75	0.84	0.85
6	32	668	8,251	13,477	149	5	26	101	63	18	0.99	2.20	6.56	6.57	3.44	0.52	0.77	0.81	0.83	0.89
7	36	1,083	19,450	68,684	551	4	34	193	194	17	1.09	2.82	8.31	13.65	3.13	0.52	0.80	0.85	0.83	0.82
8	42	2,049	45,016	340,940	11,295	6	62	432	969	17	1.09	2.88	14.70	16.84	2.74	0.50	0.91	0.94	0.86	0.86

TABLE II. PERFORMANCE OF LCM AND EMBS WITHOUT DYNAMIC THRESHOLDS ON RANDOM BIPARTITE GRAPHS.

Vertices	**Edge density**	**Maximal biclique**	**Time of LCM (milliseconds)**	**Time of EMBS (milliseconds)**	**Ratio**	**Performance Improvement (1-Ratio)**
100+100	0.10	1,371	120	80	67%	33%
100+100	0.20	11,340	102	95	93%	7%
100+100	0.30	96,809	896	848	95%	5%
100+100	0.50	11,264,781	120,075	113,920	95%	5%
200+200	0.10	13,640	132	126	95%	5%
300+300	0.10	59,296	787	731	93%	7%
400+400	0.10	178,732	3,282	2,908	89%	11%
500+500	0.10	433,874	10,156	8,672	85%	15%
1000+1000	0.01	4,233	45	43	96%	4%
2000+2000	0.01	35,322	511	417	82%	18%
4000+4000	0.01	419,076	9,399	6,964	74%	26%
6000+6000	0.01	1,823,122	60,598	44,910	74%	26%

TABLE III. PERFORMANCE (IN MILLISECONDS) OF EMBS VS *w*EMBS WITH 64 NODES AND 462 FREQUENCIES.

T	1	2	4	6	8	10	12	14	16	18	20	22	24	26	28	30	32	Average
EMBS	1.308	15	540	1588	618	331	211	259	178	158	147.8	139	132	128	123	125	116	283.4651
*w*EMBS	1.309	16	747	1295	491	273	183	222	166	153	147.2	137	129	124	120	124	117	262.0086

Data in each column of Table III are from 10 experiments. For each experiment, the same frequency threshold configuration (T) is set for both *w*EBMS and EBMS. Moreover, we have setup a computer simulation platform to test spectrum sensing process in network scenarios with more than 100 nodes. Especially, we have executed a formal verification [14] on the cooperative spectrum sensing protocol used by our application. The simulation shows real time performance of EMBS & *w*EMBS and the verification guarantees high reliability.

VII. CONCLUSION AND FUTURE WORK

In this paper, we discussed a new application named CSDM in cognitive radio networks. Based on a well-known algorithm LCM for frequent item set mining, the CSDM problem has been solved by our algorithm EMBS perfectly through introducing the idea of dynamic thresholds. Benefiting from dynamic thresholds, EMBS can prune small maximal bicliques efficiently to find the extreme maximal biclique. Therefore, most CSDM problems can be solved in real time. We also improved the performance of LCM algorithm itself in two aspects: reduce the size of the graph and reduce the time for judgment of closed state. We found that the performance of EMBS with dynamic thresholds relates to the thresholds while the performance of EMBS without dynamic thresholds relates to the edge density of the graph. And the experiments show that EMBS outperforms much more than LCM.

EMBS solves CSDM problem perfectly in one subnet. However, in some real-life applications, nodes in a subnet may have different importance or weights. Thus another problem *w*CSDM is presented in this paper and an improved version of *w*EMBS is proposed for *w*CSDM. And the performance of *w*CSDM is somewhat better than that of EMBS because we can combine those pruning techniques for CSDM with an extra pruning strategy in terms of weight.

Meanwhile, we should develop more efficient algorithm to achieve real-time performance in some very large wireless networks, though such networks are very rare in current applications. Still, the future work also includes those related applications with different definitions of extreme maximal bicliques. For example, some applications may be interested in maximal bicliques which includes the most nodes in both parts of a biclique. Moreover, some nodes in a special scene may have infinite weights and thus they must not be removed. In such cases, the current *w*EMBS can not fulfill its work because the infinite weights require a totally different strategy for calculating sum of weights. Therefore, an adaptive *w*EMBS is required for the future.

Currently, the algorithm EMBS has been put to use in a real cognitive radio network (CRN) with tree based topology. As this network has a limit in its capacity, EMBS gains surprising performance of no more than 1 millisecond for optimal solutions. Furthermore, we developed a platform for simulating with more than one hundred nodes and for verifying the protocol of cooperative spectrum sensing. In this platform, EMBS accomplished its task in real-time too and the protocol runs well after a few bugs are removed. Now, *w*EMBS is also ready to be used as more requirements for applications contribute more complexity to our CRNs.

ACKNOWLEDGMENTS

The authors wish to thank both Fan Zhong-Ji and Ji Pan-Pan who implemented the algorithm of this paper, and to thank both Liu Jin-Bo and Fan Lin-Lin who developed formal verification tools for this paper.

REFERENCES

[1] I. F. Akyildiz, W. Y. Lee, and K. Chowdhury, “CRAHNs: Cognitive Radio Ad Hoc Networks,” Ad Hoc Net. J., vol. 7, no. 5, July 2009.

[2] D. Cabric, A. Tkachenko, and R. Brodersen, “Spectrum sensing measurements of pilot, energy, and collaborative detection,” in Proc. IEEE Military Commun. Conf., 2006, pp. 1–7.

[3] T. Yücek and H. Arslan, “A Survey of Spectrum Sensing Algorithms for Cognitive Radio Applications,” IEEE Communications Surveys & Tutorials, vol. 11, no. 1, 2009, pp. 116-130.

[4] Z. J. Fan, M. X. Liao, X. X. He, H. H. Hu, X. Zhou., “Efficient Algorithm for Extreme Maximal Biclique Ming in Cognitive Spectrum Decision Making”, In IEEE ICCSN, 2011, pp. 25-30.

[5] J. Li, G. Liu, H. Li, and L. Wong, “Maximal Biclique Subgraphs and Closed Pattern Pairs of the Adjacency Matrix: A One-to-One Correspondence and Mining Algorithms,” IEEE Trans. Knowledge and Data Engineering, vol. 19, No. 12, pp. 1625-1637, Dec. 2007.

[6] L. Ji, K. L. Tan, and K. H. Tung, “Compressed Hierarchical Mining of Frequent Closed Patterns from Dense Data Sets,” IEEE Trans. on Knowledge and Data Engineering, Vol 19, No.9, Sept 2007.

[7] J. Wang, J. Han, and J. Pei, “CLOSET+: Searching for the best strategies for mining frequent closed itemsets,” in Proc. of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2003, pp. 236–245.

[8] T. Uno, M. Kiyomi, and H. Arimura, “LCM ver.2: Efficient Mining Algorithms for Frequent/closed/maximal Itemsets,” In Proc. IEEE ICDM’04 Workshop FIMI’04, 2004.

[9] T. Uno, T. Asai, Y. Uchida, and H. Arimura, “LCM: an Effcient Algorithm for Enumerating Frequent Closed Item Sets,” In Proc. IEEE ICDM’03 Workshop FIMI’03, 2003.

[10] T. Uno, M. Kiyomi, and H. Arimura, “LCM ver.3: Collaboration of Array, Bitmap and Prefix Tree for Frequent Itemset mining,” In Proc. of the 1st International Workshop on Open Source Data Mining, 2005, pp. 77–86.

[11] G. Alexe, S. Alexe, Y. Crama, S. Foldes, etc., “Consensus algorithms for the generation of all maximal bicliques,” Discrete Applied Mathematics, vol. 145(1), pp. 11–21, 2004.

[12] http://fimi.cs.helsinki.fi/src/, 24.02.2012.

[13] P. P. Ji, M. X. Liao, X. X. He, Y. Deng. “Extreme Maximal Weighted Frequent Itemset Mining for Cognitive Spectrum Decision Making,” in IEEE ICCSNT, 2011, pp. 267-271.

[14] J. B. Liu, M. X. Liao, X. X. He, X. H. Hu. “Formal Verification on Distributed Spectrum Sensing Protocol,” in IEEE ICCSNT, 2011, pp. 190-194.