Spectrum Opportunity-Based Control Channel Assignment in Cognitive Radio Networks

Loukas Lazos, Sisi Liu, and Marwan Krunz
Dept. of Electrical and Computer Engineering, University of Arizona
Email: {llazos, sisimm, krunz}@ece.arizona.edu

***Abstract*—We address the problem of dynamic assignment of coordination (control) channels in cognitive radio networks (CRNs) by exploiting time- and space-varying spectrum opportunities. Motivated by the inherent grouping of Cognitive Radio (CR) users according to channel availability, we propose a cluster-based architecture for control-channel assignment in a CRN. CRs are grouped in the same cluster if they roughly sense similar idle channels and are within communication range, either directly or via a clusterhead. We formulate the clustering design as a *maximum edge biclique* problem. A distributed cluster agreement algorithm called Spectrum-Opportunity Clustering (SOC) is proposed to solve this problem. SOC provides a desirable balance between two competing factors: the set of common idle channels within each cluster and the cluster size. A large set of common idle channels within each cluster allows graceful migration from the current control channel should primary radio (PR) activity appear on that channel. Hence, SOC provides a stable network partition with respect to local coordination, with no need for frequent reclustering. Moreover, when reclustering has to be performed (due to CR mobility or PR activity), CRs agree on new clusters after the broadcast of only three messages, thus incurring low communication overhead.**

I. Introduction

Cognitive radios (CRs) have been proposed as an enabling technology for opportunistic spectrum access [1]. These radios are capable of sensing the spectrum for idle frequency bands (channels) and dynamically hop between them to avoid interfering with primary radios (PRs). Reliable spectrum sensing can be achieved through cooperation of neighboring CRs, whose individual sensing observations are combined to determine a common set of idle channels [8], [9], [13]. Using these idle channels, CRs can be organized into a cognitive radio network (CRN) that implements network functions such as multi-channel access, and multi-hop routing.

Collaborative CRN functions, such as cooperative sensing and channel assignment, require a broadcast control channel for local coordination. In here, we define the control channel as a frequency band that is used to disseminate coordination information locally (i.e., within the one-hop neighborhood), or globally via multiple hops. For example, nodes may use a predefined control channel to negotiate the data channel assignment and to reserve the medium for future transmissions [15]. However, in a CRN, the set of channels sensed by each CR depends on its relative location to PR, and varies with the temporal and spatial variations of the spectrum availability [17]. Therefore, CRs may not be able to communicate over a predefined and globally common channel.

One possible solution to the control-channel problem is to license a frequency band for carrying control traffic [5]. However, such a design conflicts with the opportunistic nature of CRNs. Moreover, almost all available spectrum is already leased. Alternatively, an unlicensed band, such as the industrial, scientific and medical (ISM) band, or an unlicensed ultra-wide band (UWB) can be used for control [4], [5]. In this case, the reliability of the control channel cannot be guaranteed, as unlicensed bands are already over-crowded and they suffer from strong interference from other unlicensed users. In the absence of a dedicated control channel, control traffic has to be carried in-band. In such a case, the control channel is subject to PR activity, and hence has to vary in frequency depending on the locally perceived spectrum availability [17]. Such allocation must also adapt to the temporal variations of PR activity, i.e., the control channel should be migrated to another frequency band should CRs sense new PR activity on the current one.

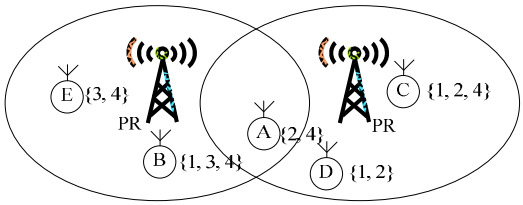


Fig. 1. Spatial variation of channel availability in a CRN due to PR activity.

In this paper, we address the problem of *control channel assignment based on time- and space-varying spectrum opportunities.* This problem becomes particularly challenging when considered in combination with cooperative sensing due to the following circular dependency. Cooperative sensing requires coordination via a control channel to determine channel availability, but the control channel is assigned after channel availability has been determined. The problem of control channel assignment is illustrated in the CRN of Figure 1. Each CR senses the set of idle channels. For example, CR A senses channels $\{2, 4\}$ as idle while CR B senses channels $\{1, 3, 4\}$ to be idle. Note that there is no idle channel common to all CRs. Hence, distinct control channels have to be allocated in different neighborhoods. This inherently leads to partitioning of the CRN into clusters, whereby the CRs of a given cluster share a common control channel. In the example of Figure 1, such partitioning can be done by forming cluster $\{A, B, E\}$, whose nodes overlap in channel 4, and cluster $\{C, D\}$, whose nodes overlap in channels $\{1, 2\}$. Note, however, that if PR activity emerges on channel 4 in cluster $\{A, B, E\}$, no other common channel exists and hence, reclustering has to be conducted.

On the other hand, if the partitioning yields clusters $\{B, E\}$ and $\{A, C, D\}$, two idle channels are available for control use in each cluster, allowing control-channel migration without reclustering if PR activity emerges on the control channel.

The example of Figure 1 highlights several critical problems relevant to the dynamic nature of the control channel. First, CRs must agree on a control channel based on the local spectrum availability, leading to the natural partitioning of the network into clusters. While many partitions may be possible, not all of them have the same stability in the light of PR dynamics. For example, Zhao et. al. considered partitioning the network into the least number of clusters by selecting the idle channel common to the highest number of CRs for control [17]. Their approach limits the overhead for inter-cluster coordination, however, frequent reclustering may be required due to the small number of common channels in each cluster. Second, CRs must efficiently coordinate the (re)clustering process in the absence of a control channel, since this channel is allocated after the common set of idle channels has been determined. Previous approaches assume that if a PR appears on the current control channel, CRs can "quickly" exchange control messages on the current channel and migrate to a new common channel [7], [17]. However, message exchanges on a channel occupied by a PR may not be feasible, due to collisions with the PR signal.

A. Our Contributions

To cope with the time- and space-varying nature of channel availability in CRNs, we consider a cluster-based assignment of the control channel. We formulate the clustering problem as a *maximum edge biclique graph problem* and show a graceful tradeoff between the cluster size and the set of common idle channels within each cluster. We develop a distributed clustering algorithm called Spectrum Opportunity-based Clustering (SOC), which groups neighboring CRs with similar channel availability in the same cluster. According to SOC, nodes agree on cluster membership in a distributed fashion after the exchange of exactly three rounds of broadcast messages. Hence, SOC is suitable for CRN environments in which reclustering occurs frequently due to PR activity or node mobility. To locally assign the control channel in the absence of a current one, we present a neighbor discovery and coordination mechanism suitable for CRNs. We also propose mechanisms for: (a) dynamic migration of the control channel within each cluster based on PR activity, and (b) inter-cluster coordination.

The remainder of this paper is organized as follows: In Section II we present our system model. In Section III, we present a neighbor discovery scheme suitable for CRNs. Section IV, describes our distributed spectrum opportunity-based clustering algorithm. We present our dynamic control channel allocation method in Section V. In Section VI, we evaluate the performance of SOC. Section VII presents related work. In Section VIII, we summarize our contributions.

II. System Model

We consider the co-existence of PRs and CRs in the same geographical area. PRs are licensed to use a fixed spectrum, which can be divided into a set $\mathcal{M} = \{1, 2, \ldots, M\}$ of M *non-overlapping* orthogonal channels. For simplicity, we assume that all channels have the same capacity. However, our algorithms can be easily extended to channels of different capacities by considering a weighted version of the clustering problem. CRs can access licensed bands if they do not interfere with ongoing PR transmissions. To prevent interference, CRs are capable of sensing spectrum opportunities using energy detectors, cyclostationary feature extraction, pilot signals, or cooperative sensing [1]. CRs are equipped with half-duplex transceivers that can either receive or transmit on a single channel at a time. Each CR can "swiftly" hop between channels using software defined radio (SDR) technology.

We assume a time-slotted system for communication. Further, we represent the CRN by a *connectivity graph* $\mathcal{G}(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of vertices corresponding to CR nodes and $\mathcal{E}$ denotes the set of edges corresponding to possible communication links. In this model, an edge exists between two CRs if they can directly communicate in the absence of any PR activity. For the i^{th} CR, denoted by CR_i, its one-hop neighborhood N_i is the set of all CRs it can communicate with based on $\mathcal{G}(\mathcal{V}, \mathcal{E})$. We also define $C_i = \{i_1, i_2, \ldots, i_{K_i}\}$ to indicate the set of idle channels sensed by CR_i. Here, $K_i = |C_i|$ and i_j denotes the j^{th} channel in the set C_i. Following the spectrum sensing process, the connectivity graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ reduces to a *communication graph* $\mathcal{G}(\mathcal{V}, \mathcal{E}')$, where an edge between two CRs x and y exists if they are within communication range and $C_x \bigcap C_y \neq \emptyset$. Note that $\mathcal{E}' \subseteq \mathcal{E}$. The definition of $\mathcal{G}(\mathcal{V}, \mathcal{E}')$ implicitly assumes the existence of a mechanism for the discovery of common channels among neighbors. We now describe such a mechanism in the absence of a known control channel.

III. Neighbor Discovery

To agree on a common set of control channels in each neighborhood, CRs must discover the network topology and exchange relevant information such as spectrum availability. This is typically achieved during the neighbor discovery phase. In the absence of a predefined control channel, CRs may be tuned to different channels during neighbor discovery, preventing them from discovering all neighbors. To address this problem, we propose a neighbor discover protocol that relies on the rotation among the set of channels $\mathcal{M}$ in a scheduled fashion. The steps of the protocol are as follows.

1) Each CR_i individually determines the list of idle channels $C_i = \{i_1, i_2, \ldots, i_{K_i}\}$.
2) CR_i beacons its channel list C_i on channel $i_j \in C_i$ during slots $t = 1, 2, ...$ if $i_j = [(t-1) \pmod M]+1$, and stays silent otherwise. Any CR_ℓ that hears CR_i's transmission places CR_i in N_ℓ.
3) CR_i communicates with $\text{CR}_\ell \in N_i$ using the channel schedule derived from C_ℓ, until a common control channel is setup.

In Step 1, each CR determines the set of idle channels. In Step 2, a universal time schedule for channel access is followed regardless of the individual views of channel availability. Each time slot t is mapped to a channel $j \in \mathcal{M}$ by a modulo M

operation. For example, for the CRN of Figure 1, if (t, j) denotes the (slot, channel) pair, the universal schedule is (1,1), (2,2) (3,3), (4,4), (5,1),... CR node A can communicate with all others CRs on channel 3 at time slots $t = 3, 7, 11$,... Using this universal schedule, CRs can discover their neighbors.

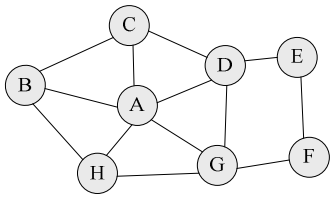
Our neighbor discovery protocol requires all CRs to be synchronized to the same time-slotted system. The CRs can use PR signals to initially acquire a common time reference. For the purpose of neighbor discovery, the time slot unit can be set to relatively large values compared to typical slotted systems, to allow the discovery of all CRs that are tuned to the same channel. Hence, the neighbor discovery phase can end after at most M time slots, in which case all possible channels will have been scanned. While in general channel availability is not expected to change during the neighbor discovery phase, should a PR appear on channel j at time slot t, the CRs can simply vacate j, update their channel lists, and use other idle channels to continue the neighbor discovery. Note that due to the long time slot unit, loose time synchronization is sufficient for our neighbor discovery protocol, with CRs resynchronizing periodically. After the one-hop neighbors are discovered, CRs use the time schedules of their neighbors to coordinate the clustering and control channel assignment process.

IV. Spectrum-Opportunity Clustering

In this section, we present a novel clustering algorithm for CRNs called Spectrum-Opportunity Clustering (SOC). Clustering in SOC is performed based on the set of idle channels that are common to all cluster members. The control channel from a given cluster is selected from this set. The goal of SOC is to group CRs with similar spectrum opportunities so as to increase the availability of common idle channels in each cluster. The reason for pursuing such a goal is twofold. First, CRs with similar channel availability can select the control channel from a larger set of channels. Hence, the control channel can migrate to another channel if the current one becomes occupied by a PR without the need for reclustering. Second, grouping CRs with similar channel lists implicitly implements hard-decision cooperative sensing [8], [9], [13].

SOC is designed with the following properties in mind: (a) distributed implementation of clustering, (b) uniqueness: CRs individually converge to the same membership decisions, (c) delay and communication efficiency: cluster agreement and control channel assignment are achieved with the exchange of a small number of messages, and (d) stability: reclustering is repeated infrequently to reduce the associated delay and bandwidth overhead for control channel re-assignment.

To satisfy these properties, we formulate the clustering problem as a *maximum edge biclique graph problem*, and use a distributed clustering algorithm to determine cluster membership. The clustering process is inspired by the algorithm proposed by Sun et. al. [16]. Their method partitions a network into cliques (fully connected components), and clustering decisions are made purely based on connectivity. Our method groups CRs based on channel availability, and it may result in clusters that do not form a clique.



$C_A = \{1, 2, 3, 4, 5, 6, 10\}$, $C_B = \{1, 2, 3, 5, 7\}$, $C_C = \{1, 2, 3, 4, 10\}$, $C_D = \{1, 2, 3, 5, 7\}$, $C_E = \{2, 3, 5, 7\}$, $C_F = \{2, 4, 5, 6, 7, 10\}$, $C_G = \{1, 2, 3, 4, 8\}$, $C_H = \{1, 2, 5, 8\}$.

Fig. 2. Connectivity graph of an 8-node CRN, and the list of idle channels individually sensed by each CR.

A. The SOC Algorithm

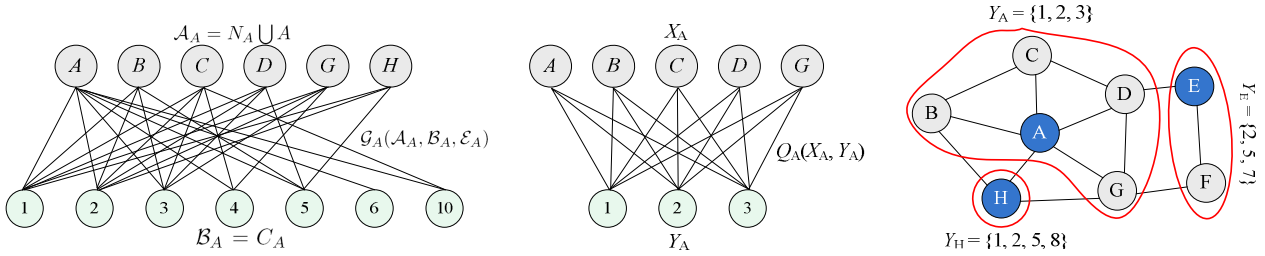
The SOC algorithm is outlined in the following three steps:

1) CRs individually compute their cluster memberships by solving the maximum edge biclique problem [14].
2) CRs broadcast the computed cluster membership information to their neighbors, and update cluster memberships accordingly. New cluster information is rebroadcasted.
3) CRs compute the final and unique cluster membership information and broadcast the final clusters to ensure consistency with their neighbors in cluster and control-channel assignments.

All broadcast operations are performed using the time schedule of the aforementioned neighbor discovery protocol, since the control channel has not yet been established. We now describe each step of SOC.

Step 1: *Maximum edge biclique computation* – In the first step, each CR$_i$ constructs an undirected bipartite graph $\mathcal{G}_i(\mathcal{A}_i, \mathcal{B}_i, \mathcal{E}_i)$, based on its neighbor list N_i and the idle channel list of each neighbor. A graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$ is called bipartite if the set of vertices $\mathcal{V}$ can be partitioned into two disjoint sets $\mathcal{A}$ and $\mathcal{B}$ with $\mathcal{A} \cup \mathcal{B} = \mathcal{V}$, such that all edges in $\mathcal{E}$ connect vertices from $\mathcal{A}$ to $\mathcal{B}$. For our purposes, $\mathcal{A}_i = CR_i \cup N_i$, and $\mathcal{B}_i = C_i$. An edge (x, y) exists between vertices $x \in \mathcal{A}_i$ and $y \in \mathcal{B}_i$ if $y \in C_x$, i.e., channel y is in the channel list of CR$_x$. In Figure 2, we show the connectivity graph of an 8-node CRN along with the idle channel list for each CR. In Figure 3(a), we show the bipartite graph $\mathcal{G}_A(\mathcal{A}_A, \mathcal{B}_A, \mathcal{E}_A)$ constructed by CR$_A$. The set of vertices $\mathcal{A}_A$ corresponds to the one-hop neighbors $N_A = \{B, C, D, G, H\}$ plus A, while the set of vertices $\mathcal{B}_A$ corresponds to the set of channels $C_A = \{1, 2, 3, 4, 5, 6, 10\}$. Note that A is connected to all vertices in $\mathcal{B}_A$, since $\mathcal{B}_A = C_A$,

Using its bipartite graph $\mathcal{G}_i$, each CR$_i$ constructs the maximum edge biclique graph $Q_i^*(X_i, Y_i, E_i^*)$. A bipartite graph $Q(X, Y, E)$ is called a biclique if for each $x \in X$ and $y \in Y$ there exists an edge between x and y, that is $E = \{(x, y) \mid x \in X, y \in Y\}$. The edge set E can be completely determined by X and Y and hence, is omitted from the biclique notation. For CR$_i$, a biclique graph $Q_i(X_i, Y_i)$ represents a clustering X_i of the one-hop neighbors of CR$_i$ plus CR$_i$, that have channels $Y_i \subseteq C_i$ in common. In Figure 3(b), we show a biclique Q_A for the bipartite graph $\mathcal{G}_A$. $Q_A(X_A, Y_A)$ represents a clustering $X_A = \{A, B, C, D, G\}$ with channels $Y_A = \{1, 2, 3\}$ common to all CRs in X_A.



(a) (b) (c)

Fig. 3. (a) Bipartite graph constructed by CR_A, (b) maximum edge biclique constructed by CR_A according to Algorithm 1, (c) three clusters constructed using SOC (CR_A, CR_D, CR_H become the CHs).

By maximizing the number of edges of Q_i, we provide a graceful tradeoff between the cluster size and the number of idle channels common to all CRs in a cluster. To illustrate this tradeoff, consider a cluster of size $|X_i|$ with $|Y_i|$ common channels. The number of common channels in the cluster can increase only if the cluster size decreases. Let δ be the increase in number of common channels, when the cluster size decreases by β. According to the principle of maximizing the number of edges in the biclique, the cluster size will decrease by β if,

$$\frac{\delta}{|Y_i|} > \frac{\beta}{|X_i| - \beta}, \quad |Y_i| + \delta \leq M. \tag{1}$$

Inequality (1) states that the current cluster size $|X_i|$ will decrease by β if the fractional increase $\delta/|Y_i|$ in common channels is larger than the fractional decrease $\beta/(|X_i| - \beta)$, computed over the new cluster size. This inequality is constrained by the total number of channels M. This rule allows for a graceful balance between the cluster size and the set of common channels in a cluster. If CRs in a one-hop neighborhood have similar channels lists, our clustering rule will be fairly inclusive, resulting in large clusters. On the other hand, if the channel lists between CRs vary significantly, our clustering rule will reduce the cluster size to increase the common channel availability within each cluster. We further investigate this tradeoff in Section VI. Note that in general, the maximum edge biclique may contain any subset of vertices of the original bipartite graph. However, our bipartite graph construction guarantees that: (a) any channel common to all neighbors of CR_i will be part of the maximum edge biclique Q_i^*, and (b) CR_i will also be part of Q_i^*. This is shown in the following lemma.

Lemma 1: Let a vertex $x \in \mathcal{A}$ of a bipartite graph $\mathcal{G}(\mathcal{A}, \mathcal{B}, \mathcal{E})$ be connected to all vertices in set $\mathcal{B}$. Then, x belongs to the maximum edge biclique $Q^*(X, Y)$.

Proof: We prove Lemma 1 by contradiction. Let $x \in \mathcal{A}$ be a vertex of a bipartite graph $\mathcal{G}(\mathcal{A}, \mathcal{B}, \mathcal{E})$ such that there exists an edge (x, y), $\forall y \in \mathcal{B}$. Let $Q^*(X, Y)$ be the maximum edge biclique, and assume that $x \notin X$. By adding x to the graph Q^*, we obtain graph $Q'(X \bigcup x, Y)$, which is still a biclique since x is connected to every vertex of $\mathcal{B}$ and hence, every vertex of Y. The number of edges of the biclique Q' is $(|X| + 1) \times |Y| > |X| \times |Y|$. This contradicts our initial assumption that Q^* is a maximum edge biclique. The same result can be shown for any vertex $y \in \mathcal{B}$ connected to all vertices in $\mathcal{A}$. ■

Finding the maximum edge biclique of a bipartite graph is an NP-complete problem [14]. For bipartite graphs of small size, an exhaustive search may be feasible. However, the space of possible solutions grows exponentially with the cardinality of the vertex set. Hence, we develop a greedy heuristic (Algorithm 1) for finding a biclique with a large number of edges.

Algorithm 1 Greedy Heuristic for Computing the Maximum Edge Biclique

INPUT $\mathcal{G}_i(\mathcal{A}_i, \mathcal{B}_i, \mathcal{E}_i)$
$Y_i \leftarrow \mathcal{B}_i$
for $j = 1$ to $|\mathcal{A}_i|$ **do**
  Find $\text{CR}_k \in \mathcal{A}_i$ such that $|Y_i \bigcap C_k|$ is max
  if $Y_i \bigcap C_k = \emptyset$ **then**
    break
  else
    $S_i[j] = k$;
    $\mathcal{A}_i \leftarrow \mathcal{A}_i - \text{CR}_k$, $X_i \leftarrow X_i \bigcup \text{CR}_k$, $Y_i \leftarrow Y_i \bigcap C_k$,
    $P_i[j] = |X_i| \times |Y_i|$
  end if
end for
Find $j^* = \arg\max_j P_i[j]$
return $Q^*(X_i, Y_i)$; $X_i = \{\text{CR}_{S_i[1]}, \ldots, \text{CR}_{S_i[j^*]}\}$ $Y_i = \bigcap_{k=1}^{k=j^*} C_{S_i[k]}$

Algorithm 1 iteratively examines a single CR in each round. Vector S_i holds the indexes of the CRs that have already been examined, while Y_i holds the list of common channels among CRs already in S_i. Initially, S_i is empty while $Y_i = \mathcal{B}_i$. In each iteration, we find the CR_k whose channel list C_k has the highest overlap with Y_i. We then remove CR_k from $\mathcal{A}_i$, add k to S_i, and repeat the process until either $\mathcal{A}_i$ is empty or the intersection of the common channel list with the remaining CRs is an empty set. Note that in each iteration, $Q(X_i, Y_i)$ is a biclique with $|X_i| \times |Y_i|$ edges. We record the number of edges of each constructed biclique in the vector P_i, and then find the biclique with the maximum number of edges. Algorithm 1 guarantees the property expressed in Lemma 1. That is, a CR with all the channels C_i on its channel list will be included in the maximum edge biclique Q_i^*, as computed by Algorithm 1. This is formalized in the following lemma.

Lemma 2: Any $x \in A_i$ with $C_i \subseteq C_x$ will be included in $Q_i^*(X_i, Y_i)$, as computed by Algorithm 1.

Proof: Let $x \in \mathcal{A}_i$ be a vertex of a bipartite graph $\mathcal{G}_i(\mathcal{A}_i, \mathcal{B}_i, \mathcal{E}_i)$ such that there exists an edge (x, y) $\forall y \in \mathcal{B}_i$. Assume that the maximum edge biclique $Q_i^*(X_i, Y_i)$ is com-

puted at the j^{th} iteration of Algorithm 1. Then any CR added to X_i in the previous iterations will be part of Q_i^*. Hence, it is sufficient to show that x will be added in X_i before or on the j^{th} iteration. If $Y_i = C_i$ then $x \in X_i$, since addition of x can only increase the number of edges of Q_i^*, by C_i. If $Y_i \subset C_i$, there exist some $x' \in X_i$ such that $C_{x'} \bigcap C_i \subset C_i$. Since on initialization $Y_i = C_i$, and $C_x \bigcap C_i = C_i$ according to line 4 of Algorithm 1, x will be added to X_i before x'. Hence, Q_i^*, will contain x. ■

Lemma 2 is essential for guaranteeing that the maximum edge biclique computed by CR_i using the Algorithm 1 will include CR_i in the cluster membership X_i. We now illustrate the application of Algorithm 1 to the CRN of Figure 2, and for node A. The bipartite graph $\mathcal{G}_A$ for CR_A is shown in Figure 3(a). In the first iteration, node A is selected since it has the highest overlap with $Y_A = C_A$. The number of edges of the biclique is recorded to be $P_A[1] = 7$. In the second iteration, node C is selected, resulting in a cluster membership $X_A = \{A, C\}$, a set of common channels $Y_A = \{1, 2, 3, 4, 10\}$, and $P_A[2] = 10$. Further iterations result in the addition of CRs D, B, G, H in that particular order, and the number of edges is $P_A[3] = 9$, $P_A[4] = 12$, $P_A[5] = 15$, and $P_A[6] = 12$. The biclique Q_A^* with the maximum number of edges is selected. The final cluster is $X_A = \{A, B, C, D, G\}$ and the common channel list is $Y_A = \{1, 2, 3\}$. This biclique is depicted in Figure 3(b).

Step 2: *Updating cluster membership information* – Let $Q_i^1(X_i^1, Y_i^1)$ denote the maximum edge biclique computed by CR_i in step 1. The cluster membership X_i^1 is likely to be different at each CR_i due to the difference in the one-hop neighbor sets N_i. In step 2, each CR_i informs its neighbors of X_i^1 and Y_i^1. When CR_i receives this information, it checks if there is a biclique Q_j^1 with $\text{CR}_i \in X_j^1$ that provides *better* clustering than Q_i^1. Before we can show the convergence to the same cluster membership/common channel list independently by various CRs, we define the inequality operator "$<$" between Q_i and Q_j.

Definition 1: For two bicliques $Q_i(X_i, Y_i)$ and $Q_j(X_j, Y_j)$, $Q_i < Q_j$ if and only if

(a) $|X_i| \times |Y_i| < |X_j| \times |Y_j|$ or,
(b) $|X_i| \times |Y_i| = |X_j| \times |Y_j|$ and $|X_i| < |X_j|$ or,
(c) $|X_i| \times |Y_i| = |X_j| \times |Y_j|$, $|X_i| = |X_j|$, $|Y_i| = |Y_j|$, and $i < j$.

In the above definition, we first compare the number of edges in the two bicliques. If two bicliques have the same number of edges, we then compare the cluster sizes. If the cluster sizes are also equal, we compare the cardinality of the common channel sets. We break the tie by selecting the biclique of the CR with the highest id. Definition 1 imposes a total ordering between different bicliques (i.e., two bicliques can never be equal).

CR_i selects the biclique $Q_j^1(X_j^1, Y_j^1)$ with $\text{CR}_i \in X_j^1$, that is best according to the relation operator given in Definition 1, and updates its maximum edge biclique to $Q_i^2 = Q_j^1$. After computing Q_i^2, each CR_i informs its neighbors of the updated cluster membership X_i^2 and the common channel list Y_i^2.

We now illustrate the execution of step 2 for the CRN of Figure 2. Node A receives the following updates from its one hop neighbors: (a) Q_B^1 with $X_B = \{A, B, H\}$ and $Y_B = \{1, 2, 5\}$, (b) Q_C^1 with $X_C = \{A, B, C, D\}$ and $Y_C = \{1, 2, 3\}$, (c) Q_D^1 with $X_D = \{A, C, D, E, G\}$ and $Y_D = \{2, 3\}$, (d) Q_G^1 with $X_G = \{A, D, G, H\}$ and $Y_G = \{1, 2\}$, and (e) Q_H^1 with $X_H = \{A, B, G, H\}$ and $Y_H = \{1, 2\}$. Ordering the bicliques according to Definition 1 yields, $Q_G^1 < Q_D^1 < Q_H^1 < Q_B^1 < Q_D^1 < Q_C^1 < Q_A^1$. Then A sets $Q_A^2 = Q_A^1$ since Q_A^2 has the maximum number of edges. Similarly, $Q_B^2 = Q_A^1$, $Q_C^2 = Q_A^1$, $Q_D^2 = Q_A^1$, $Q_G^2 = Q_A^1$, and $Q_H^2 = Q_A^1$.

Step 3: *Finalizing cluster membership* – In step 3, each CR_i examines the cluster membership X_i^2. For each $\text{CR}_j \in X_i^2$, if $\text{CR}_i \notin X_j^2$ then CR_i removes CR_j from its biclique Q_i^2. Completion of this step yields the final bicliques Q_i^3, which provide the final cluster memberships X_i^3 and the common channels Y_i^3. If a CR_i has adopted the clustering proposed by a neighboring CR_j during step 2, it may be the case that X_j^2 may contain CRs not within the range of CR_i. For those CRs, CR_i will not have biclique information. To provide cluster information in this case, CR_j will replay all received biclique info if its biclique is selected by any of its neighbors in step 2.

We now illustrate step 3 for the CRN of Figure 2. Node A goes through its list X_A^2 and checks if it is included in the bicliques of all CRs in X_A^2. Given that all neighboring CRs have adopted Q_A^1, A concludes that $X_A^3 = X_A^2$ and $Y_A^3 = Y_A^2$. Similarly, B goes through its list $X_B^2 = \{A, B, C, D, G\}$. Since D and G are not within range of B, the only way that B can know about Q_D^2 and Q_G^2 is if A has replayed their updates. According to our algorithm, A replays Q_D^2 and Q_G^2, since both D and G have adopted the Q_A^1. Hence, B can see that it is included in the bicliques of B and G, and thus set $X_B^3 = \{A, B, C, D, G\}$. Note that H is not included in any of the biclique updates of A, B, and G and therefore H forms its own cluster. The final clustering and the list of common channels are shown in Figure 3(c).

B. Correctness of the SOC Algorithm

We now prove that the SOC algorithm leads to consistent cluster memberships. The correctness proof follows similar steps to the proof of the method in [16]. Several modifications are made to adjust the proofs to the use of bicliques with cluster memberships that possibly do not form cliques.

Lemma 3: If $\text{CR}_i \in X_j^2$ and $\text{CR}_j \in X_i^2$, then $Q_i^2 = Q_j^2$.

Proof: After step 1, CR_i and CR_j will have received the updates from their one-hop neighbors. Suppose that CR_i selects $Q_i^2 = Q_k^1$ where CR_k is a neighbor of CR_i or is CR_i itself. Since $\text{CR}_j \in X_i^2$, this means that $\text{CR}_j \in X_k^1$, and hence, CR_j is a neighbor of CR_k. Following a similar argument, we can show that for the decision $Q_j^2 = Q_m^1$ to be made, the selected Q_j^2 must be constructed by a CR_m which is a neighbor of CR_i given that $\text{CR}_i \in X_j^2$. Since CR_k and CR_m are neighbors of both CR_i and CR_j, CR_i and CR_j must have received both Q_k^1 and Q_m^1 in step 1, before updating their own bicliques. Due to the total ordering imposed, CR_i concludes that $Q_m^1 < Q_k^1$, and CR_j concludes that $Q_k^1 < Q_m^1$. This is true only if $k = m$. ■

According to Lemma 3, two CRs that include each other on their respective bicliques after step 2, must have agreed on the same bicliques. We utilize this result in Lemma 4.

Lemma 4: Suppose that for CR_i, CR_j, and CR_k, $CR_k \in X_i^2$ and $CR_k \in X_j^2$ with $Q_i^2 = Q_j^2$. Then if $CR_i \notin X_k^2$, it must also hold that $CR_j \notin X_k^2$.

Proof: Lemma 4 can be proved by contradiction. Assume that $CR_j \in X_k^2$. By Lemma 3, since $CR_j \in X_k^2$ and $CR_k \in X_j^2$, then $Q_j^2 = Q_k^2$. However, by assumption we also have $Q_i^2 = Q_j^2$, and hence $Q_i^2 = Q_k^2$. Since $CR_k \in X_i^2$ and $Q_i^2 = Q_k^2$, this also means that $CR_i \in X_k^2$, which is a contradiction. Hence $CR_j \notin X_k^2$. ■

Based on Lemmas 3 and 4, we now show that SOC guarantees that CRs will have consistent cluster membership information. It also follows that CRs will agree on the set of common channels.

Theorem 1: For any $CR_j \in X_i^3$, $Q_i^3 = Q_j^3$.

Proof: Q_i^3 is a pruned version of Q_i^2, i.e., $X_i^3 \subseteq X_i^2$. Therefore, any $CR_j \in X_i^3$ must have also been a member of X_i^2. Also for any $CR_j \in X_i^3$, we have $CR_i \in X_j^2$ since otherwise, CR_j would have been removed from X_i^3. Using Lemma 3 it follows that $Q_i^2 = Q_j^2$. Now consider any $CR_k \in X_i^2$ which is removed from X_i^2 in step 3, i.e., $CR_k \notin X_i^3$. This happens only if $CR_i \notin X_k^2$, which also means by Lemma 2 that $CR_j \notin X_k^2$, and CR_k will also be removed from X_j^2 in step 3. Hence, every CR that is removed from X_i^2 is also removed from X_j^2 making $X_i^3 = X_j^3$. For two bicliques with the same membership, it follows that $Y_i^3 = Y_j^3$, and hence $Q_i^3 = Q_j^3$. ■

C. Clusterhead Election

Many cluster-based protocols require the election of a clusterhead (CH) for facilitating the protocol operation. A typical requirement for any CH election process is that the CH is able to communicate with all members of its cluster. SOC algorithm inherently implements CH election. Though CRs of a cluster are not guaranteed to form a clique (for example in Figure 3(c), CRs B, D are not within communication range while in the same cluster), in the following lemma we prove that at least one CR is guaranteed to be within one-hop of every cluster member. This CR can be identified in step 3 of SOC.

Lemma 5: In every cluster produced by SOC, there is at least one CR that is one-hop away from all other CRs of that cluster.

Proof: Consider a cluster expressed by the biclique $Q_i^3(X_i^3, Y_i^3)$. According to Theorem 1, all $CR_j \in X_i^3$ converge to the same cluster membership in step 3. For any CR_i and $CR_j \in X_i^3$, it holds that $CR_i \in X_j^2$ and $CR_j \in X_i^2$. Otherwise, CR_i would have removed CR_j from X_i^2 in step 2, and similarly CR_j would have removed CR_i from X_j^2. According to Lemma 3, if $CR_i \in X_j^2$ and $CR_j \in X_i^2$, it holds that $Q_i^2 = Q_j^2$. This means that all members of a cluster formed after step 3, must have computed the same biclique in step 2. But the biclique Q_i^2 of any CR in step 2 is the best biclique Q_j^1 with $CR_j \in N_i$ or $j = i$. Hence, the only way that all CRs of a cluster would chose Q_j^1 as the best biclique in step 2 is if CR_j is a neighbor to all. Therefore, there exists at least one CR in the cluster that is one-hop from all other CRs. This CR announces the best biclique in step 2. ■

For the CRN in Figure 3(a), A, E and H are elected as CHs.

V. Dynamic Control Channel Allocation

After clustering and CH election have been completed, the CH can select one channel from the common list for control. From an architectural standpoint, the assignment of different control channels to various clusters poses two major challenges:

Inter-cluster coordination problem: Consider the clustering in Figure 3(c). Suppose that CH A selects channel 1 for control whereas CH E selects channel 5. Assume that CR_D wants to broadcast a control message to all its neighbors. Broadcasting on channel 1 will only reach CRs A, C and G, but not E, who may be tuned to control channel 5. In this case, CR_D would have to repeat its broadcast on channel 5 to ensure its reception by CR_E. Note, however, that communication would not be possible if channel 7 was selected for control within cluster $\{E, F\}$, since channel 7 is not in C_D.

Control channel migration problem: Consider now the process of migrating the control channel. For the CRN in Figure 3(a), assume that a PR occupies control channel 1, and only CR_B senses the PR activity (due to fading or being out of range of the transmitting PR). CR_B needs to notify other CRs in the cluster that channel 1 is no longer free. Since other CRs are tuned to channel 1, a notification sent on this channel may interfere with the PR transmission. Even if the notification is sent, the CH A must notify all CRs in the cluster of the new control channel selection. Again this has to occur on channel 1, thus possibly interfering with the PR transmission and not being correctly received.

A. Control Channel Hopping

To allow for inter-cluster communication and to coordinate control channel migration, we propose the following slow channel-hopping mechanism. Rather than using the same channel for control until PR activity appears on it, the control channel hops among the common channels within each cluster. Let $Y_i = \{i_1, i_2, \ldots, i_{K_i}\}$ denote the set of common channels in cluster i with $K_i = |Y_i|$. For time slots $t = 1, 2, \ldots$, CRs within cluster i use channel i_j where $j = [(t-1) \pmod{K_i}] + 1$. The channel hopping mechanism is similar to the hopping of the neighbor discovery mechanism. However, CRs hop only through the list of common channels within their cluster and channel schedules between clusters may be different.

To illustrate the above control-channel hopping procedure, consider the CRN of Figure 3(a). For the cluster with CH A, the set of common channels is $Y_A = \{1, 2, 3\}$. The (slot, control channel) pairs for the first few slots are (1,1), (2,2), (3,3), (4,1)... Similarly, for cluster with CH D the (slot, control channel) pairs are (1,2), (2,5), (3,7), (4,2)... Note that various clusters may operate on different control channels at a given time slot, depending on the PR activity. CRs at the boundary between clusters are aware of the control channel schedules as long as the list of common channels within each cluster is known.

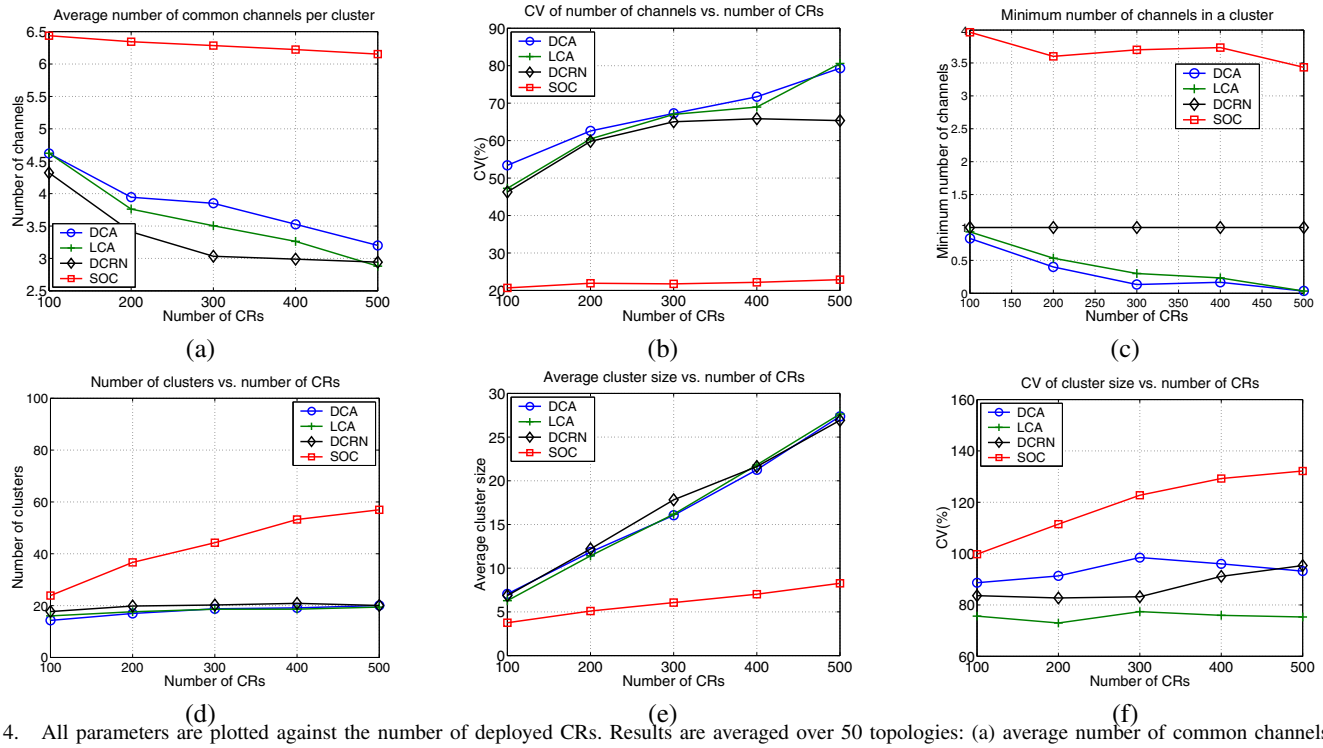


Fig. 4. All parameters are plotted against the number of deployed CRs. Results are averaged over 50 topologies: (a) average number of common channels per cluster, (b) CV coefficient for the average number of common channels, (c) minimum number of channels in a cluster, (d) average number of clusters in the CRN, (e) average cluster size, (f) CV coefficient for the average cluster size.

When a CR senses PR activity on the current control channel, it can wait until the control channel is migrated to an idle one before notifying other CRs within the same cluster. For example, in the CRN of Figure 3(a), suppose that CR_B senses PR activity on channel 1. When the control channel hops to channel 2, CR_B notifies CH A of its new idle channel list. The CH updates the new list of common channels to $Y'_A = \{2, 3\}$ and broadcasts Y'_A to all CRs in the cluster, using either channel 2 or 3. The control channel now hops only between channels 2 and 3.

Channel hopping also addresses the problem of inter-cluster coordination. A CR that is within the communication range of a cluster can communicate with that cluster as long as one of the common channels of that cluster is in its idle channel list. For example, if node D is aware of the common channel list $Y_E = \{2, 5, 7\}$ of the cluster with CH E, it can use time slots $(t-1) \equiv 1 \pmod 3 + 1$ to communicate control information on channel 2. To enable inter-cluster coordination, CRs use the broadcast of Q_j^3 from their neighbors to obtain the common channel list of adjacent clusters and derive their channel schedule.

B. Impact on Reclustering

Although SOC converges after only three rounds of message exchanges, it is necessary to limit frequent recomputation of clusters. By updating the common channel list, clustering need not be repeated every time the channel availability changes at a CR. Instead, each cluster progressively adapts its set of common channels to spectrum variations. Reclustering can be avoided as long as at least one common channel is still available among the CRs of the same cluster at any given time. A reclustering operation can be triggered periodically to account for the long-term dynamics of PR activity and changes in the CRN topology.

VI. Performance Evaluation

In this section, we use simulations to demonstrate the agility of SOC in adapting to variations in channel availability. We also explore the tradeoff between the cluster size and the set of common channels in a cluster. To perform our evaluation, we randomly deployed CRNs of different sizes within a 100x100m area. We set the communication range of each CR to 10 meters and the available PR channels equal to 10. Our results are averaged over 50 different CRN topologies.

As a point of reference, we compare the performance of SOC against three schemes: (a) the distributed clustering algorithm (DCA) [3], (b) the lowest-id clustering algorithm (LCA) [2], and (c) the distributed coordination scheme proposed in [17]. We will refer to the latter as DCRN. In DCA, a node is elected as a CH if it has the highest weight among its neighbors. Every CR selects the neighboring CH with the highest weight and joins its cluster. We set the weight of a CR to be its degree of connectivity on the connectivity graph. In LCA, a node becomes a CH if it has the lowest id within its neighborhood. CRs join the cluster of the lowest id neighboring CH. Note that both DCA and LCA do not take into account the channel availability. Finally, in DCRN, neighboring CRs are grouped in the same cluster as long as they have at least one channel in common.

The metrics used to compare the performance of various algorithms are: (a) number of clusters in the network, (b) average cluster size, (c) average number of common channels in each cluster, (d) minimum number of common channels, (e) coefficient of variation (CV) for the common channels, and (f) CV for the cluster size. The CV is a normalized measure of the dispersion of a probability distribution, defined as the ratio of the standard deviation over the mean value.

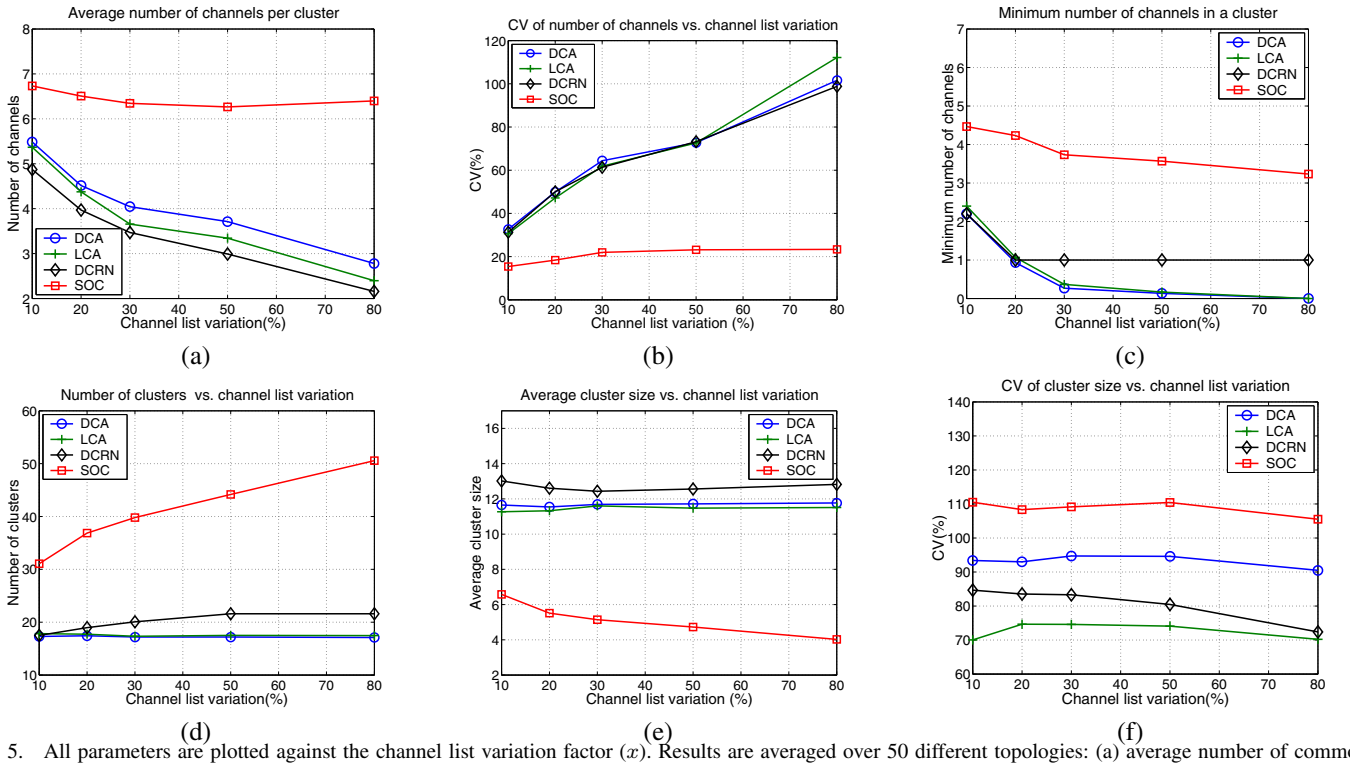


Fig. 5. All parameters are plotted against the channel list variation factor (x). Results are averaged over 50 different topologies: (a) average number of common channels per cluster, (b) CV coefficient for the average number of common channels, (c) minimum number of channels in a cluster, (d) average number of clusters in the CRN, (e) average cluster size, (f) CV coefficient for the average cluster size.

A. *Variation of Deployment Density*

In the first set of experiments, we vary the node density by increasing the number of deployed CRs. To emulate the variation in the channel lists between CRs, we partition the network area to a grid of 10x10 squares. CRs within each square has a fraction $(100 - x)\%$ of channels common while the remaining $x\%$ of channels are randomly selected. The cardinality of the set of idle channels at each CR is also varied. Adjacent squares in the grid have a $(100 - x)\%$ overlap in common channels while the rest are randomly selected. For our first set of experiments, we set x to 30.

In Figure 4(a), we show the average number of common channels in a cluster as a function of the number of CRs. We observe that SOC results in clusters with 50%-100% more common channels compared to the other clustering methods. Even DCRN that takes into account the channel availability, results in clusters with a small set of common channels, due to the goal of minimizing the number of clusters. In fact, as shown in Figure 4(b), the CV for SOC is very low (around 20%) and remains constant with the variations in node density. This behavior is essential to guaranteeing that there are sufficient idle channels for migration when a PR appears on the current control channel. Hence, frequent reclustering is avoided. On the other hand, other methods show high variability in terms of number of common channels in each cluster. In Figure 4(c), we show the minimum number of channels within a cluster. We observe that DCA, LCA, and DCRN can construct clusters with very few common channels. This trend is further exacerbated as the density increases. This is because DCA, LCA, and DCRN tend to create large size clusters, where CRs channel lists have small overlap. On the other hand, SOC consistently creates clusters with several common idle channels.

In Figure 4(d), we show the number of clusters in the CRN as a function of the number of deployed CRs. We observe that under SOC, the number of clusters increases almost linearly with the number of deployed CRs. This is due to the fact that the average cluster size is kept relatively constant to allow for multiple common channels within each cluster. The tendency of SOC to keep a constant cluster size is illustrated in Figure 4(e). In Figure 4(f), we show the CV as a function of the number of CRs. SOC exhibits a larger variability in cluster size, due to its adaptation to variations in spectrum opportunities.

B. *Variation of Channel List*

In the next set of experiments, we var the channel list variation parameter (x). A low channel list variation indicates that CRs within a given neighborhood detect roughly the same set of idle channels, whereas a high value indicates uncorrelated channel lists. In Figure 5(a), we show the average number of common channels in a cluster for different values x. We observe that as the channel lists sensed by neighboring CRs become uncorrelated, SOC adjusts the cluster size to maintain an almost constant number of common channels. This is verified in conjunction with Figures 5(d) and (e), which respectively show the number of clusters in the CRN and the average cluster size as a function of x. We observe that the average cluster size decreases slowly as x increases to compensate for the reduction in the number of common channels in a cluster. The reduction in the average cluster size causes an increase in the number of clusters in the CRN, as shown in Figure 5(d). On the other hand, the average number of common channels per

cluster decreases to a value less than three for the DCA, LCA, and DCRN algorithms. The number of clusters, the average cluster size, and the CV for the cluster size, as shown in Figures 5(d),(e), and (f), remain constant. In Figure 5(b), we show the CV for the common channel list within each cluster. SOC has the lowest CV values yielding common channel sets of similar cardinality within each cluster. On the other hand, the cardinality of the common channel set significantly increases for DCA, LCA, and DCRN indicating a fairly uneven distribution in the number of common channels. In Figure 5(c), we show the minimum number of common channel in a cluster. SOC adjusts the cluster size and ensures that at least three channels are present in each cluster. On the other hand, when the channel list variation increases, the other clustering algorithms construct clusters with zero common channels.

VII. Related Work

Architectures for the control channel assignment in CRNs can be classified into two types. The first type relies on the existence of a dedicated static control channel known to all CRs (e.g. [4], [5], [10]). Čabrić et. al. proposed the CORVUS system, in which control traffic is carried over static unlicensed ultra wide bands (UWBs) [5]. Brown proposed the use of the unlicensed ISM band for control in CRNs [4]. Han et. al. proposed an OFDM-based scheme to allow for long-range transmission of control messages with small bit error rates [10]. The use of unlicensed bands cannot guarantee the reliability of control transmissions, which are critical for the CRN operation, as such bands suffer from interference from unlicensed users.

In the second type of architectures, the control channel is no longer static or universal. Zhao et. al. proposed the distributed coordination of CRs via a locally computed control channel that changes dynamically in response to PR activity [17]. The control channel available to the largest set of one-hop neighbors is selected for control in each neighborhood, implementing a partition of the CRN into clusters. Their approach minimizes the set of distinct frequency bands used for control within the CRN (i.e., the number of clusters), but also reduces the common set of channels within each cluster. This can lead to the frequent reclustering due to variations in PR activity. SOC follows the same philosophy as [17], but with the goal of achieving a balance between the cluster size and the set of common channels in the cluster, so that reclustering will rarely be needed.

The problem of distributed clustering in ad hoc networks with fixed number of channels has been extensively studied in the literature (e.g. [2], [3], [6], [11], [12], [16]). Clustering methods for such networks can be classified into: (a) leader-first cluster formation, in which a CH is first selected based on metrics such as node connectivity, mobility, residual energy, and id (e.g. [2], [3], [6]), and (b) cluster-first, in which the cluster is formed before a CH is elected (e.g. [11], [12], [16]). SOC belongs to the latter category, and is closely related to cluster-first clique formation clustering methods [11], [16].

Acknowledgments

This research was supported by the National Science Foundation (under grant CNS-0721935), BAE, Raytheon, and Connection One (an I/UCRC NSF/industry/university consortium). Any opinions, findings, conclusions, or recommendations expressed in this paper are those of the author(s) and do not necessarily reflect the views of NSF.

VIII. Conclusion

We addressed the problem of control channel allocation in CRNs. We argued that in CRNs, the control channel cannot be statically allocated due to the dynamics of the spectrum availability. We proposed a cluster-based architecture, in which the control channel is dynamically allocated within each cluster. We formulated the clustering problem as a maximum-edge biclique problem and showed that this formulation allows for a graceful tradeoff between the cluster size and the set of common channels in each cluster. We proposed a distributed clustering algorithm called SOC that takes into account the channel availability in deciding cluster memberships. We proved that CRs converge to the same clusters after only three rounds of message exchanges. We further proposed a control channel rotation scheme that enables control channel migration in case of PR activity, inter-cluster communication, and adaptation to the temporal variations of spectrum availability.

References

[1] I. Akyildiz, W. Lee, M. Vuran, and S. Mohanty. NeXt generation/dynamic spectrum access/cognitive radio wireless networks: A survey. *Computer Networks*, 50(13):2127–2159, 2006.

[2] D. Baker, A. Ephremides, and J. Flynn. The Design and Simulation of a Mobile Radio Network with Distributed Control. *Selected Areas in Communications, IEEE*, 2(1):226–237, 1984.

[3] S. Basagni. Distributed clustering for ad hoc networks. In *I-SPAN 1999*, pages 310–315, 1999.

[4] T. Brown. An analysis of unlicensed device operation in licensed broadcast service bands. In *Proc. of DySpan*, pages 11–29, 2005.

[5] D. Čabrić, S. Mishra, D. Willkomm, R. Brodersen, and A. Wolisz. A Cognitive Radio Approach for Usage of Virtual Unlicensed Spectrum. *14th IST Mobile and Wireless Communications Summit, June*, 2005.

[6] M. Chatterjee, S. Das, and D. Turgut. WCA: A Weighted Clustering Algorithm for Mobile Ad Hoc Networks. *Cluster Computing*, 5(2):193–204, 2002.

[7] T. Chen, H. Zhang, G. Maggio, and I. Chlamtac. CogMesh: A Cluster-Based Cognitive Radio Network. In *Proc. of DySPAN*, pages 168–178, 2007.

[8] G. Ganesan and Y. Li. Cooperative spectrum sensing in cognitive radio networks. In *Proc. of DySpan*, pages 137–143, 2005.

[9] A. Ghasemi and E. Sousa. Collaborative spectrum sensing for opportunistic access in fading environments. In *Proc. of DySPAN*, pages 131–136, 2005.

[10] C. Han, J. Wang, Y. Yang, and S. Li. Addressing the control channel design problem: OFDM-based transform domain communication system in cognitive radio. *Computer Networks*, 2007.

[11] P. Krishna, N. Vaidya, M. Chatterjee, and D. Pradhan. A cluster-based approach for routing in dynamic networks. *ACM SIGCOMM Computer Communication Review*, 27(2):49–64, 1997.

[12] C. Lin and M. Gerla. Adaptive clustering for mobile wireless networks. *Selected Areas in Communications, IEEE*, 15(7):1265–1275, 1997.

[13] S. Mishra, A. Sahai, and R. Brodersen. Cooperative sensing among cognitive radios. In *Proc. of ICC*, 2006.

[14] R. Peeters. The maximum edge biclique problem is NP-complete. *Discrete Applied Mathematics*, 131(3):651–654, 2003.

[15] J. So and N. Vaidya. Multi-channel mac for ad hoc networks: handling multi-channel hidden terminals using a single transceiver. In *Proc. of MOBIHOC*, pages 222–233, 2004.

[16] K. Sun, P. Peng, P. Ning, and C. Wang. Secure distributed cluster formation in wireless sensor networks. In *Proc. of ACSAC 2006)*, 2006.

[17] J. Zhao, H. Zheng, and G.-H. Yang. Spectrum sharing through distributed coordination in dynamic spectrum access networks. *Wireless Communications and Mobile Computing*, 7(9):1061–1075, 2007.