

Minimal Time Broadcasting in Cognitive Radio Networks*

Chanaka J. Liyana Arachchige, S. Venkatesan, R. Chandrasekaran, and Neeraj Mittal

Erik Jonsson School of Engineering and Computer Science, The University of Texas at Dallas,
800 West Campbell Road, Richardson, TX 75080, USA
{chanaka.liyana, venky, chandra, neerajm}@utdallas.edu

Abstract. This paper addresses time-efficient broadcast scheduling problem in Cognitive Radio (CR) Networks. Cognitive Radio is a promising technology that enables the use of unused spectrum in an opportunistic manner. Because of the unique characteristics of CR technology, the broadcast scheduling problem in CR networks needs unique solutions. Even for single channel wireless networks, finding a minimum-length broadcast schedule is an NP-hard problem. In addition, the multi-channel nature of the CR networks, especially the non-uniform channel availability, makes it a more complex problem to solve. In this paper, we first present an Integer Linear Programming formulation (ILP) to determine the minimum broadcast schedule length for a CR network. We then present two heuristics to construct minimal length broadcast schedules. Comparison of optimal results (found by solving the ILP formulation) with the result of the heuristics through simulation shows that both heuristics produce schedules of either optimal or very closer to optimal lengths.

Keywords: cognitive radio networks, time efficient broadcasting, broadcast scheduling.

1 Introduction

Cognitive Radio (CR) is a technology that enables the use of spectrum in an efficient manner. CR nodes can sense the spectrum over a wide range of frequency bands and utilize unused or underutilized licensed frequency bands. Since CR nodes do not own these bands, they cannot use these bands indefinitely. When the owners of the frequency bands (primary users) start using their bands, CR nodes need to vacate the band and move to another band. Since the availability of free channels at a node depends on factors such as the physical location of the node and the hardware capability of its transceiver (frequency range supported by the radio hardware), even neighboring CR nodes may have different channels available to them.

Broadcasting is a fundamental operation in computer networks. Inherent broadcast nature of the wireless medium makes it more appealing for wireless networks to support broadcast as a primary network operation. In this paper, we focus on constructing

* This work was partly supported by a grant from Research In Motion (RIM) Ltd.

TDMA-based broadcast schedules that enable a given node to send a message to all other nodes in a CR network. A TDMA-based schedule consists of multiple timeslots. In each time slot of the schedule, every node transmits on some channel, listens on some channel or stays idle in a pre-determined manner.

1.1 Motivation

CR technology introduces new complexity to the broadcast scheduling problem due to its non-uniform channel availability and multi-channel nature. Solutions to the broadcast scheduling problem in single channel wireless networks do not need to be concerned about which channel to use in broadcasting. Also traditional multi-channel networks can utilize single channel algorithms, since all the nodes have uniform channel availability. But in a CR network, channel availability is not uniform and the availability of a network-wide common channel is not guaranteed. Therefore any solution to the broadcast scheduling problem in CR networks needs to consider channel availability at each node.

To further illustrate the challenges of broadcast scheduling in a CR network, consider a simple star network with *node A* as the center node and N edge nodes. In the single channel case (figure 1 (a)), *node A* can broadcast a message to all of its N neighbors using a single time slot. In a traditional multi-channel network where channel availability is uniform, *node A* can again broadcast to all its neighbors using a single timeslot by transmitting on any channel, provided all its neighbors listen on that channel. However, in a CR network, *node A* may not be able to broadcast a message using only a single time slot; it may have to transmit the message multiple times using a different channel each time. For example in figure 1 (b) *node A* needs at least three timeslots to transmit the message to all its neighbors. In the worst case, each neighboring node may have a share a different channel with *node A* and broadcasting will take N time slots (see figure 1 (c)).

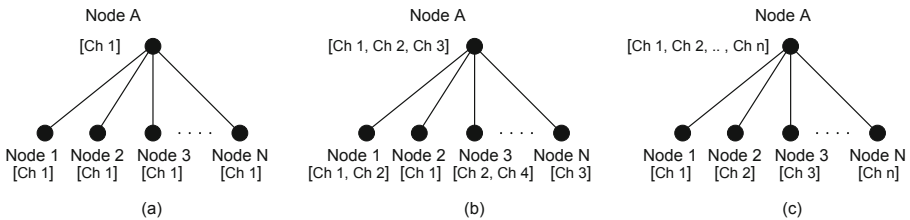


Fig. 1. Star network topology. Available channels for each node are shown in the square brackets. (a) Single channel network. (b) CR network 1. (c) CR network 2 (each node has a unique common channel with node A).

1.2 Related Work

Time-efficient broadcast scheduling is a widely studied problem for single channel networks. Chlamtac and Kutten [1] proved that finding a time-optimal broadcast schedule is an NP-hard problem. This has led to the development of approximation algorithms and global bounds for minimum-time broadcast scheduling problem [2]. Peleg [2] presented a comprehensive literature review on time-efficient broadcast

scheduling algorithms for single channel networks under different models and assumptions. The non-uniform channel availability (specially the possibility of unavailability of globally common channel) makes it hard to use single channel broadcast scheduling heuristics in CR networks.

Qadir et al. [6] studied the minimum-latency broadcast scheduling problem for multi-channel, multi-hop radio networks. They have proposed a set of centralized heuristic solutions to the problem. But they assume each node has multiple interfaces (or transceivers) and that the number of interfaces is equal to the number of channels.

Broadcast scheduling in CR networks is a relatively new problem. To the best of our knowledge, the only published work in this area is due to Kondareddy and Agrawal [8]. They proposed a broadcast algorithm for the exchange of control information based on flooding the message using a limited set of channels. Constructing a time-efficient broadcast schedule was not considered in their work.

1.3 Our Contributions

Our focus is on finding a minimal-length schedule for transmitting a message from a node to all other nodes in a multi-hop CR network. To the best of our knowledge, this is the first paper that addresses time-efficient broadcast scheduling problem for CR networks. Assuming time is slotted, we propose to find a schedule that minimizes the number of time slots in the schedule. Our main contributions are as follows:

- An Integer Linear Programming solution to find an optimal broadcast schedule for a CR network.
- Two polynomial-time heuristic solutions to find a time-efficient broadcast schedule for a CR network.
- Simulation study to compare heuristic results with the optimal schedule length.

The rest of the paper is organized as follows. Section 2 presents the system model used in our solution. Section 3 presents the ILP formulation and section 4 presents two heuristics. Simulation and results are presented and discussed in section 5. Section 6 concludes the paper.

2 System Model

2.1 Node

We consider a CR network consisting of N nodes. Each node is assigned a unique identifier from the range $1 \dots N$. Each node knows its own ID and is equipped with one wireless transceiver (transmitter and receiver). Thus each node is capable of either transmitting or receiving (but not both) at any given moment.

2.2 Medium

The CR network has M channels available for potential use. This is termed as the global channels set ($C_{global} = \{1, 2, \dots, M\}$). Each node knows the global channel set. A node can operate on a subset of this global channel set depending on the channel availability at that node. We assume that each node knows its channel availability set;

finding and maintaining the channel availability set is outside the scope of this work. We also assume channel availability set to be relatively stable with respect to the algorithm execution time.

2.3 Network Operation

We assume a multi-hop wireless network formed by CR nodes that operates in a time slotted manner [9]. Broadcast schedule is calculated by the broadcast initiating node (source node). Source node knows about the network topology and channel availability information at each node. Information about the schedule can be sent to other nodes using either a common signaling channel or other signaling mechanisms. Note that as long as the physical topology and the channel availability sets do not change, the same broadcast schedule can be used.

We represent the CR network as an undirected graph. Vertices represent nodes. Edges represent connectivity between nodes. There is an edge between two nodes, if there is at least one common channel between two nodes and if they are within the communication range of each other. All edges are bidirectional. (If A can send a message to B on channel x , B can also send a message to A on channel x .)

3 ILP Formulation

In this section, we present an Integer Linear Programming (ILP) formulation to determine the optimal (minimum) broadcast schedule length for CR networks.

Consider an undirected graph $G = (V, E)$, where $V = (1, \dots, N)$ is a set of nodes and E is a set of edges. Each node can operate on a subset of C_{global} channels. Available channel set at node i is denoted by C_i , where $C_i \subseteq C_{global}$. L is the schedule length in terms of number of timeslots. Initially node 1 has the message. Our aim is to determine whether message can be propagated to all the nodes in the network using L timeslots. First we define following variables:

For $i \in V$, $k \in C_i$ and $t \in L$ define $y_{i,k,t}$ as follows:

$$y_{i,k,t} = \begin{cases} 1 & \text{If node } i \text{ transmits on channel } k \text{ during timeslot } t \\ 0 & \text{Otherwise} \end{cases}$$

For $i \in V$, $k \in C_i$ and $t \in L$ define $z_{i,k,t}$ as follows:

$$z_{i,k,t} = \begin{cases} 1 & \text{If node } i \text{ receives on channel } k \text{ during timeslot } t \\ 0 & \text{Otherwise} \end{cases}$$

The necessary constraints for the ILP are as follows:

1. Since a node has only one transceiver, it cannot simultaneously transmit and receive during a single time slot. Also a node cannot transmit or receive on more than one channel during a timeslot. This is ensured by the following constraint:

$$\sum_{k \in C_i} (y_{i,k,t} + z_{i,k,t}) \leq 1 \quad \forall i \text{ and } \forall t$$

2. A node cannot transmit the message before receiving it, except for the first node.

$$y_{i,k,t} \leq \sum_{k' \in C_i} \sum_{t' < t} z_{i,k',t'} \quad \forall i \neq 1, \forall k \in C_i \text{ and } \forall t$$

3. A node cannot receive a message unless a neighbor transmits the message.

$$z_{i,k,t} \leq \sum_{j:(i,j) \in E, i \neq j} y_{j,k,t} \quad \forall i, \forall k \in C_i \cap C_j \text{ and } \forall t$$

4. A node cannot receive correctly if two or more neighbors transmit during the same timeslot on the same channel. This constraint deals with the collisions.

$$y_{j,k,t} + y_{j',k,t} \leq 2 - z_{i,k,t} \quad \forall (i,j) \in E \text{ and } (i,j') \in E, i \neq j \neq j', \\ \forall k \in C_i \cap C_j \cap C_{j'} \text{ and } \forall t$$

5. All nodes except the source node should receive the message.

$$\sum_{k \in C_i} \sum_t z_{i,k,t} \geq 1 \quad \forall i \neq 1$$

This formulation does not directly provide the optimal broadcast schedule length. Instead, for a given schedule length L , it gives the feasibility of the solution. If a solution is feasible, we can get the schedule by observing which of the $y_{i,k,t}$ and $z_{i,k,t}$ values are equal to 1.

For any given graph, we cannot produce a broadcast schedule shorter than the minimum hop distance from the source node to the farthest node. Thus the graph radius of the source node acts as a lower bound on the broadcast schedule length. To find the optimal schedule length, we can start from this lower bound and keep incrementing it until ILP gives a feasible solution.

4 Heuristic Algorithm

Even though ILP can be used to get the optimal broadcast schedule length, it takes a substantial amount of time to terminate. Therefore we present two centralized heuristics to find a time-efficient broadcast schedule.

4.1 Heuristic 1

The idea behind heuristic 1 is to give priority to nodes that have paths to farthest nodes (from the source node) in the network. Whenever we have to choose between several nodes, we give priority to nodes that have the shortest paths to farthest nodes.

The heuristic consists of two phases. In the first phase, we assign a level number to each node based on its distance from the source node and then select transmitting instances (transmitting node, receiving nodes and channels) to serve each level. During the second phase we assign these transmitting instances to the time slots.

When assigning levels, the source node gets level 0, while the farthest node gets the highest level. When selecting transmitting instances, we start from the highest level (farthest from source node) and assign transmitting nodes to serve all the nodes in each level using nodes from the immediate lower level (e.g. level 4 is served using nodes from level 3). Transmitting (Tx) node selection is done as follows.

Consider the nodes at levels k and $k-1$. Our aim is to find a smallest set of nodes from level $k-1$ to cover all the nodes in the level k . This is equivalent to the set cover problem which is known to be NP-hard [10]. We use the following simple greedy heuristic to solve this problem:

1. Find a node and a channel from level $k-1$, which covers the maximum number of nodes at level k . If more than one channel can be used by Tx node to cover the same receiving (Rx) node set, we keep all such channels as possible options when we move to the next phase.
2. Remove the covered nodes from the level k . If no nodes are remaining in level k , selection is complete for the level. Else go to step 1.

To prioritize the nodes that have paths to farthest nodes, we use a rank. Initially, each node has rank 0. Each time we select a Tx instance, rank of the Tx node is updated based on the ranks of nodes it covers (Tx node rank = maximum rank of Rx nodes + 1). For example if a selected Tx node covers a set of nodes with ranks 0,3 and 1, then the Tx node gets 4 as the rank.

At the end of the first phase we have a set of Tx instances consisting of Tx node, its rank, the corresponding Rx nodes and possible channels for transmission. Essentially this set forms the broadcasting tree. Also note that a node can appear in more than one Tx instance and can have different rank values. When a node has multiple rank values, maximum rank will be used when computing the Tx node rank.

During the second phase, we assign Tx instances to the time slots. For the first time slot, we start with the source node. If we have more than one Tx instance with source node as the Tx node, we select the Tx instance with the highest rank. From the second time slot onwards, we select Tx instances based on the following criteria:

1. From the Tx instances set which already has the message, select a Tx instance with the highest rank. Schedule it to transmit on current timeslot. In case of a tie, we select the Tx instance which has the most number of Rx nodes. If number of Rx nodes is also equal, select one randomly.
2. Try to schedule another Tx instances to the same timeslot. From the Tx instances set which already has the message and not already scheduled to transmit during current time slot, pick a Tx instance with the highest rank.
3. Check whether the selected Tx instance causes collisions with already scheduled transmissions. If it does not cause collisions, schedule it to transmit on current time slot and go back to step 2. If it causes collision go to step 4.
4. Check next highest rank Tx instance which already has the message and not already scheduled to transmit during current time slot. If such Tx instance can be found go to step 3. If such Tx instance cannot be found, scheduling for this current slot is done. Move to the next time slot and start from step 1. Once all nodes are scheduled to receive the message, algorithm will be done.

Checking for collisions is done by checking the following two conditions:

1. Candidate Tx node's transmission on selected channel interferes with already scheduled Rx nodes.
2. Candidate Rx node's receptions on selected channel are interfered by the already scheduled Tx nodes.

4.2 Heuristic 2

The idea behind the second heuristic is to select a set of Tx nodes and channels for each time slot such that the number of Rx nodes served is maximum. While selecting Tx nodes and channels, we make sure collisions are avoided.

At each time slot, we consider the set of nodes that have the message (covered), and the set of nodes that do not have the message (uncovered). Then, in each timeslot, we try to cover the maximum number of nodes from the uncovered set using the covered node set, without causing collisions. This is again equivalent to the set cover problem which is NP-hard [10]. We use the following greedy approach to solve the problem.

Consider following sets of nodes (i = current timeslot):

$$A_i = \{\text{Nodes who has the message at end of timeslot } i-1\}$$

$$A_1 = S = \{\text{Source node}\}$$

$$B_i = \{\text{Single hop neighbors of nodes in } A_i, \text{ who do not have the message at the end of timeslot } i-1\}$$

$$\text{txSet}_i = \{\text{Nodes supposed to transmit in timeslot } i\}$$

$$\text{rxSet}_i = \{\text{Nodes supposed to receive in timeslot } i\}$$

Scheduling is done as follows:

1. Start from first time slot ($i=1$).
2. Select a node and a channel (node should be in the set A_i) which covers the maximum number of nodes from B_i (break the ties randomly). Add selected Tx node to txSet_i and corresponding Rx nodes to rxSet_i .
3. Find another node and a channel (node should be in the set $\{A_i - \text{txSet}_i\}$) which covers maximum number of nodes from the set $\{B_i - \text{rxSet}_i\}$ and does not cause collisions. Collisions are identified as described in the heuristic 1.
 - a. If such a node can be found, add it to txSet_i and its Rx nodes to the rxSet_i . Then repeat step 3.
 - b. If no such nodes can be found, scheduling for this timeslot is done. If all the nodes are covered scheduling is done. Otherwise move to next time slot ($i+1$), update the sets A_i and B_i , and go to step 2.

4.3 Time Complexity Analysis – Heuristic 1

Heuristic 1 consists of two phases; Phase 1 consists of assigning levels based on the distance from the source node and selecting Tx instances. Level assignment is similar to finding shortest path to all nodes from the source and can be done in $O(N^2)$ time.

Selection of Tx instances for each level is done using a greedy heuristic. Here we assume each node in a Tx level keeps a list of nodes that it can cover from a Rx level for each available channel. This lists can be constructed in $O(MN^2)$ time. Selecting the

node and the channel that cover the maximum number of nodes from Rx level takes $O(MN)$ time. (Recall that M is the number of available channels.) Once a Tx instance is selected, we need to update the Rx node lists of each Tx node based on the already scheduled Tx instance information. Updating one list takes $O(N \log N)$ time. (We have two lists of size $O(N)$ and we need to check whether a node in already selected Tx list present in the Rx node's list. This can be done in $O(N \log N)$ time.) Since we have $O(N)$ Tx instances and $O(M)$ channels, updating all the lists takes $O(MN^2 \log N)$ time. Therefore the time required to find one Tx instance and update the Tx node details is $O(MN + MN^2 \log N) = O(MN^2 \log N)$. For each level, we may have to select up to $O(N)$ nodes. Therefore to complete each level we need $O(MN^3 \log N)$ time. If the network radius is R , we need $O(RMN^3 \log N)$ time to complete the Tx instance selection procedure over all levels. Therefore phase one takes $O(RMN^3 \log N)$ time.

In the second phase, we assign selected Tx instances to the time slots. Selection of the Tx instance with highest rank takes $O(N)$ time. Then, we try to schedule another Tx instance for same time slot. Here, we have to check for collisions. The first step in checking for collisions is to check for interference to already scheduled Rx nodes. For each scheduled Rx node, we need to check whether it is a neighbor of candidate Tx node and whether they use the same channel. This can be done in $O(N \log N)$ time. The second step in checking for collisions is to check whether candidate Rx node receptions are interfered by the already selected Tx node. Here we check whether any of the Tx nodes, which are scheduled to use the same channel as candidate Rx nodes, are present in the neighbor lists of the candidate Rx nodes. This can take $O(N^2 \log N)$ time. Hence, the total time for collision checking is $O(N^2 \log N + N \log N) = O(N^2 \log N)$.

The total time for scheduling one node is $O(N + N^2 \log N) = O(N^2 \log N)$. We might have to schedule up to $O(N)$ nodes in one timeslot. Therefore, the time required to schedule one time slot is $O(N^3 \log N)$. If the schedule length is L time slots, the time complexity of phase two is $O(LN^3 \log N)$. Therefore the time complexity of heuristic 1 is $O(RMN^3 \log N + LN^3 \log N)$.

4.4 Time Complexity Analysis – Heuristic 2

Time complexity of the second heuristic follows from the first one. In the second heuristic, we try to pack as many Tx nodes as possible to each time slot while avoiding collisions. This is similar to the greedy method used in heuristic 1 to select Tx instances. Therefore we can use same time complexity here. Also the time complexity of collision checking procedure is the same. Therefore the time complexity of scheduling one timeslot is $O(MN^3 \log N + N^2 \log N) = O(MN^3 \log N)$. If the schedule length is L time slots, then we need $O(LMN^3 \log N)$ time. So the total time complexity of the heuristic 2 is $O(LMN^3 \log N)$.

5 Simulation and Results

5.1 Simulation Setup

To evaluate the performance of our heuristics, we compared the optimal schedule length with the lengths of the schedule lengths generated by our heuristics. The

optimal schedule length was obtained using the ILP formulation presented in the section 3. We implemented the ILP formulation using GNU Linear Programming Kit (GLPK) [11]. Schedule lengths for the two heuristics were obtained using computer simulations. We ran the simulations on a wide range of network topologies and scenarios.

Network topologies were generated by creating random graphs as follows: first we generated a predefined number of random points (x,y coordinates) within a 1000 x 1000 square. Each point represents a node in the network. To avoid nodes being too close to each other (clustering problem) we rejected new points that are within a threshold distance to existing nodes. Edges were then added to the graph. This was done using a circle of radius R centered on each point. Whenever another point is found within the circle, an edge is added between them. Since we require a connected graph, whenever no points are found within the circle, we increased the radius by *r* percentage until we find points within the circle. Once edges are added for each node, there is still a possibility that the graph is disconnected. In such cases, we added an edge between nearest two points of such disconnected components. This way we make sure final graph is connected. We used JGraphT [12] graph library for the implementation of graphs.

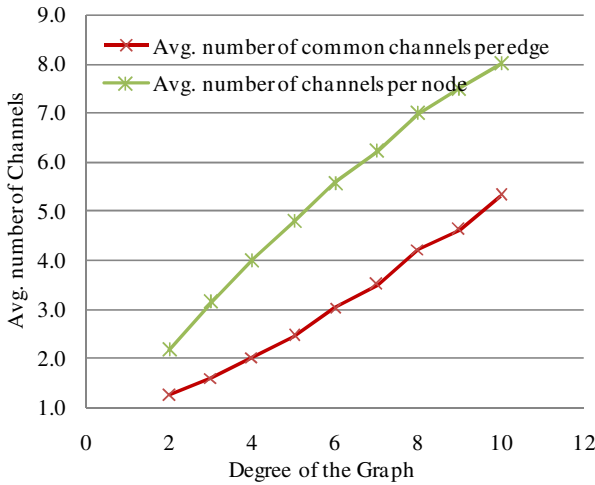


Fig. 2. Channel availability among the nodes and average number of common channels per edge. (Total number of available channels = 15)

Once we have the network topology, the next question was how to assign channels for each node. We use following method for assigning channels:

First, we assigned *k* ($k = 1..M$) channels to each edge randomly. Then based on assigned channels for each edge, the channel set for each node was determined by taking union of the channels assigned to each edge. Note that even though we assign *k* channels for each edge, the final number of channels available for each edge may be higher than *k*.

From the simulation results, we have observed that when there are multiple common channels available for edges, the schedule lengths given by both heuristics were either optimal or very close to the optimal. To simulate worst case scenarios, we decided to keep the number of common channels per edge to a small value. Therefore we selected $k = 1$. Average statistics for the channel distribution among the nodes and edges are given in figure 2.

For the final simulation we have used 100 nodes in a 1000 X 1000 area. The total number of available channels was set to 15. By changing the value of R (within the range 30 to 200) we obtained graphs with degree ranging from 2 to 10.

5.2 Results

Simulation results (broadcast schedule length) were collected by running both heuristics and the ILP on randomly generated network topologies. We varied the graph topology from sparse to dense by changing the average graph degree from 2 to 10. For each average degree value, we ran both heuristics and ILP on 10 randomly generated graphs. The reason that we limit only to 10 graphs is the time taken to run the ILP. Table 1 summarizes the main results. Figure 3 presents the comparison between optimal and heuristic schedule lengths.

Table 1. Results for the simulation and ILP (Note that all the results shown are averaged over 10 graphs)

Average Graph Degree	Average Graph Radius	Average Schedule Length		
		Optimal (ILP)	Heuristic 1	Heuristic 2
2	22.7	22.8	22.8	26.2
3	21.4	22.1	22.3	24.8
4	19.7	19.9	20.2	21.7
5	15	15.2	15.5	16
6	12.4	12.4	13.1	13.1
7	10.8	11	11.3	11.5
8	10	10.3	10.5	10.6
9	9.1	9.1	9.2	9.3
10	8.5	8.5	9	8.6

5.3 Discussion

From the results, we can clearly see that the heuristic 1 performed very well compared to the optimal results. The average difference is only 2.41%. Actually we have not seen it deviating more than 2 timeslots from the optimal schedule length. The maximum difference between the optimal length and the length of the heuristic 1 is 5.88% (for degree of 10).

One possible explanation for the performance of the heuristic 1 to be very close to the optimal is as follows:

Assume a scenario where a node has to deliver a message to n neighboring nodes. Even if the node could not send the message to all n nodes during the timeslot t , there is a high possibility that it can deliver the message in the subsequent time slots (e.g.

$t+1, t+2, etc.$) while the recipient nodes of the time slot t can also transmit on the timeslots $t+1, t+2, etc.$ This is because of the availability of multiple channels which enable simultaneous communication among neighboring nodes without causing collisions. (Note that in a single channel network, this is not possible.) Therefore in CR networks, there is a high possibility that a node can transmit a message to all of its neighbors fairly quickly (within successive timeslots). Now the question is to select the order of transmissions. Heuristic 1 takes care of this by assigning priorities to the nodes that have paths to the farthest nodes. Therefore the combination of availability of multiple channels and giving priority to nodes that have paths to the farthest nodes works well to produce faster schedules.

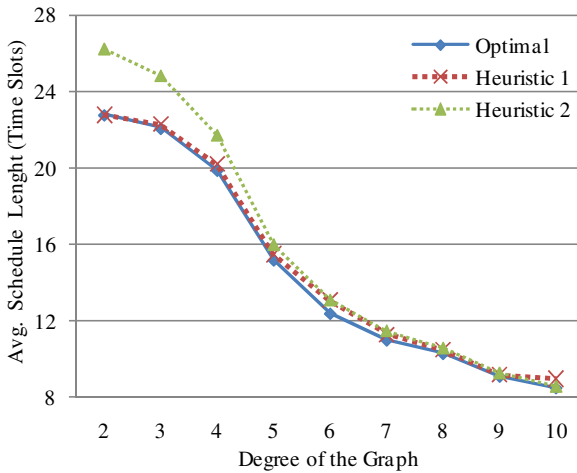


Fig. 3. Comparison between the optimal schedule length and heuristic schedule lengths (Number of nodes = 100, Number of available channels = 15)

Heuristic 2 also performed very well when the graph was not sparse. But for sparse graphs performance of the heuristic 2 was not as good as heuristic 1. When the graph is sparse the number of available paths from any given node to other nodes is limited. Since heuristic 2 does not give priority to nodes that have paths to farthest nodes, this performance difference is understandable. But when the graph is dense, there are many paths of equal length, from any given node to all other nodes, thus giving priority to nodes that have paths to farthest nodes does not make a big difference.

An important observation that we have made is the relationship between the graph radius (hop distance from source node to the farthest node) and the optimal schedule length; these two values were very close. (See table 1.) The average difference between two values was only 1.21%. In terms of time slots, difference was always either 1 or 0. Therefore, we believe that the radius of the network can be taken as a good approximation for the optimal broadcast schedule length in CR networks. This also leads to the conclusion that the characteristics of the CR networks (i.e. availability of multiple channels and non-uniform channel availability) do not hinder achieving a short broadcast schedule.

6 Conclusion

In this paper, we addressed the time-efficient broadcast scheduling problem in Cognitive Radio Networks. We have presented an Integer Linear Programming (ILP) formulation for finding an optimal broadcast schedule for a CR network. We have also presented two centralized heuristics to find minimal-length broadcast schedules.

In the first heuristic, we start from the farthest nodes (from source node) in the network and select a set of transmitting nodes/channels to cover the entire network. Then scheduling is done by giving priority to the node/channel pairs that cover nodes who have paths to the farthest nodes. In the second heuristic, we start from the broadcast initializing node and pick the best node/channel pair at each stage, considering the number of nodes covered by each potential transmitting node in each available channel as the selection criteria. In both heuristics, we try to pack as many transmitting node/channel pairs as possible to each time slot, while avoiding collisions.

We have implemented the ILP to obtain the optimal schedule. Also we implemented our heuristics and performed simulations on different graph topologies. Comparison of optimal results with simulation results show that both heuristics produce schedule lengths that are very close to the optimal schedule lengths.

References

1. Chlamtec, I., Kutten, S.: On broadcasting in radio networks - problem analysis and protocol design. *IEEE Transactions on Communications* 33, 1240–1246 (1985)
2. Peleg, D.: Time-Efficient Broadcasting in Radio Networks: A Review. In: Janowski, T., Mohanty, H. (eds.) *ICDCIT 2007*. LNCS, vol. 4882, pp. 1–18. Springer, Heidelberg (2007)
3. Chlamtac, I., Weinstein, O.: The Wave Expansion Approach to Broadcasting in Multihop Radio Networks. *IEEE Transaction on Communications* 30(3) (1991)
4. Kowalski, D., Pelc, A.: Optimal deterministic broadcasting in known topology radio networks. *Distributed Computing* 19(3), 185–195 (2007)
5. Gasieniec, L., Peleg, D., Xin, Q.: Faster communication in known topology radio networks. *Distributed Computing* 19, 289–300 (2007)
6. Qadir, J., Chou, C.T., Misra, A.: Minimum Latency Broadcasting in Multi-Radio Multi-Channel Multi-Rate Wireless Mesh Network. In: *Proc. of IEEE Sensor and Ad Hoc Communications and Networks* (September 2006)
7. Li, L., Qin, B., Zhang, C., Li, H.: Efficient Broadcasting in Multi-radio Multi-channel and Multi-hop Wireless Networks Based on Self-pruning. In: Perrott, R., Chapman, B.M., Subhlok, J., de Mello, R.F., Yang, L.T. (eds.) *HPCC 2007*. LNCS, vol. 4782, pp. 484–495. Springer, Heidelberg (2007)
8. Kondareddy, Y.R., Agrawal, P.: Selective Broadcasting in Multi-Hop Cognitive Radio Networks. In: *IEEE Sarnoff Symposium* (April 2008)
9. Krishnamurthy, S., Thoppian, M., Kuppa, S., Chandrasekaran, R., Mittal, N., Venkatesan, S., Prakash, R.: Time-efficient Distributed Layer-2 Auto-configuration for Cognitive Radio Networks. *Computer Networks* 52(4) (2008)
10. Corman, T.H., Leiserson, C.E., Rivest, R.L., Stein, C.: *Introduction to Algorithms*, 2nd edn.
11. *GNU Linear Programming Kit (GLPK)*, <http://www.gnu.org/software/glpk/>
12. *JGraphT*, <http://www.jgrapht.org>