EURASIP Journal on
Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

ZAP: a distributed channel assignment algorithm for cognitive radio networks

Paulo Roberto Walenga Junior[1], Mauro Fonseca[1*], Anelise Munaretto[2], Aline Carneiro Viana[3] and Artur Ziviani[4]

Abstract

We propose ZAP, an algorithm for the distributed channel assignment in cognitive radio (CR) networks. CRs are capable of identifying underutilized licensed bands of the spectrum, allowing their reuse by secondary users without interfering with primary users. In this context, efficient channel assignment is challenging as ideally it must be simple, incur acceptable communication overhead, provide timely response, and be adaptive to accommodate frequent changes in the network. Another challenge is the optimization of network capacity through interference minimization. In contrast to related work, ZAP addresses these challenges with a fully distributed approach based only on local (neighborhood) knowledge, while significantly reducing computational costs and the number of messages required for channel assignment. Simulations confirm the efficiency of ZAP in terms of (i) the performance tradeoff between different metrics and (ii) the fast achievement of a suitable assignment solution regardless of network size and density.

Keywords: Cognitive Radio Networks, Wireless Networks, Channel Selection, Distributed Solution

1. Introduction

The unlicensed portion of the spectrum becomes increasingly overloaded because of the growing number of wireless nodes and mobile users. While a small portion of the frequency spectrum is overloaded, a large part of the frequency spectrum licensed to primary users is being underutilized or never used at all [1].

Cognitive radios (CRs) [2-4] allow the reuse of underutilized portions of the frequency spectrum by secondary users (SUs) in a non-interfering manner with primary users (PUs). To achieve this capability, a CR should be able to investigate the spectrum applying an adaptive learning approach based on historical observations of the channel behavior. Through this investigation a CR is able to identify *white holes*, i.e., non-utilized frequency channels in a specific timeslot that are available for communication. Once the white holes are identified, the CR should distribute the available non-utilized channels to similar network nodes in range. This problem, known as *channel assignment*, aims at allocating a single channel to each network link to maximize the network capacity [5]. The channel assignment problem for cognitive radio networks (CRNs) has been recently addressed by both centralized [6] and distributed [7,8] approaches.

On the one hand, while a centralized approach to the channel assignment problem in CRNs usually obtains best results considering solely the utilization of network capacity, the proposals based on this strategy typically incur a high communication overhead. Considering that the channel availability is frequently time-varying, a centralized approach becomes less efficient because the information on which this allocation was based may have already become outdated when the channel assignment solution is defined. On the other hand, distributed approaches [7,8] to the channel assignment problem in CRNs provide solutions that are less costly, more fault tolerant, and more competitive than centralized approaches in terms of overall results, even if not reaching the best channel assignment. Although such decentralized approaches present promising results, reducing communication overhead and dealing with frequent changes in the network are in general disregarded by them.

An efficient channel assignment algorithm for CRNs should ideally use the least possible amount of communication resources to allow CRs to reuse underutilized

* Correspondence: mauro.fonseca@ppgia.pucpr.br
[1]Pontifical Catholic University of Paraná (PUC-PR), Brazil
Full list of author information is available at the end of the article

channels by SUs without interfering with PUs. In this context, providing an efficient channel assignment is challenging as it requires that it must be simple, incur acceptable communication overhead, provide timely response, and be adaptive to accommodate frequent changes in the network (e.g., because of the PUs resuming operation or node's mobility). Moreover, achieving a distributed optimization of the network capacity utilization while mitigating the interference can be an additional challenge.

In this article, we propose the ZAP algorithm providing a fully distributed channel assignment in CRNs. ZAP operates in the common control channel (CCC), which allows avoiding competition between control and data messages' exchange. In contrast to related study (see Section 5), ZAP addresses the afore-mentioned challenges in channel assignment for CRNs with a fully distributed approach based only on local (neighborhood) knowledge. An efficient solution is shown to be achieved while significantly reducing computational cost, mitigating the interferences among simultaneous transmissions, and decreasing the number of exchanged control messages.

A simulation study confirms the efficiency of ZAP's channel assignment in terms of the performance tradeoff between different metrics as compared to both a random channel assignment (lower bound) and a centralized channel assignment (upper bound). Results show that ZAP outperforms a random channel assignment in about 10% for increasing values of available channels, network sizes, and network densities. When compared to the upper bound results of the centralized approach, ZAP is able to correctly imitate its behavior, with a decrease of about only 7% in performance for varying average network densities and network sizes, and about only 5% for varying number of channels. In addition, results indicate that the six first interactions of the ZAP algorithm achieve 99% of the performance that could be achieved if an infinite number of interactions, regardless of the network size or network density, were carried out. Similar performance is perceived even under 5% of message losses.

This result contributes to a timely response in ZAP as well as to its scalability. Moreover, as ZAP uses only local knowledge (neighborhood) of each node, it significantly reduces the number of messages as compared to a centralized algorithm and then makes ZAP network size independent.

The organization of the article is as follows. In Section 2, the tackled problem is specified and models for CR, network, and interference are defined. Section 3 presents the proposed ZAP algorithm. The performance evaluation results are presented and analyzed in Section 4. In Section 5, related study is discussed. Finally, Section 6 concludes the article and discusses future work.

2. The channel assignment problem

This section presents the considered models for CR, network, and interference, which are used for the problem specification and proposal definition.

A. CR model

According to the model proposed in [9], the cognitive cycle consists of four functions for spectrum management: sensing, decision, sharing, and mobility. Only the first two functions (sensing and decision) are directly related to the problem of assigning channels; therefore, the other functions (sharing and mobility) will not be considered in this study.

The sensing function is responsible for the search of underutilized frequency bands (*out-of-band sensing*) and also for the monitoring of bands to be employed in the communication of the unlicensed node itself (*in-band sensing*). The aim is to detect a possible use of the band by the licensed PUs and then immediately interrupt its use by the unlicensed SU.

Contrary to the IEEE 802.11 standard [10], in CRNs, it is assumed that the channels are orthogonal to each other, so that the frequency bands do not overlap. It is reasonable to assume that a node initially spends some time looking for underutilized channels to create a list of available channels. As soon as the licensed PUs begin to occupy again some of the available channels, these channels are removed from the list until only a few channels (e.g., 2 or 3) remain. At this moment, a new sensing has to be performed, allowing a new search of underutilized frequency bands.

The decision function can be divided into three modules: characterization, selection, and reconfiguration. Considering channel characteristics (such as interference, path loss, error rate, and propagation delay) as parameters and the historical behavior of PUs, the characterization module has the ranking of channels found by the sensing function as its aim. The selection module performs the channel assignment to SU nodes based on the ranked list provided by the characterization module and it is the focus of this study. Finally, the reconfiguration module adapts higher layer protocols to the channel parameters and is beyond the scope of this study.

In terms of equipment, the presence of two radio interfaces is assumed: one permanently tuned to a CCC [11] and another able to quickly–compared to the transmission time of frames–switch between channels. The use of a CCC allows the design of a distributed system architecture, where no central controller is required. In the literature, several studies consider a portion of the spectrum to be reserved to form a CCC for exchanging control information [11-13].

B. Network model

The network model considers a wireless mesh network with wireless and static mesh routers, each one equipped with a CR. The network is modeled by a network graph, vertices (nodes) of whivh represent CR routers and the edges represent the links between two neighboring nodes. As each node has its own list of available channels, two nodes are considered as neighbors if and only if they are within the communication range of each other and the intersection of their lists of available channels is not empty. Nodes are considered within the range of each other if they can communicate through the CCC.

C. Interference model

When analyzing a scenario with multiple hops, one of the factors limiting the performance of the network is the interference– i.e., two nodes cannot perform successful communication if they are using the same channel at the same time and if they are on the interference range of each other.

In this study, we adopt a modified model of the two-hop interference [14], in which two nodes are considered to be interfering if they are exactly two hops away from each other. We justify the choice of this model by the existence of only one radio interface for data communication, so that a node can communicate with only one of its neighbors at each given time. We thus consider the one-hop interference as contention (managed by the sending of RTS/CTS), which is not possible to be eliminated. It remains to mitigate the maximum interference at two hops. It is important to remark that this interference model is used only to create the interference graph to perform the channel assignment. Moreover, this interference model is not used for data communication. The consideration of data communication at the channel assignment procedure is left, however, for future study. We assume, as an input for the distribution, a binary interference model (two nodes either completely interfere or do not interfere with each other) where all the links have the same priority to avoid the interference. It is worth noting that this may be seen as the worst case to be considered, i.e., all links generate interference on the neighbors. In a more realistic scenario, nodes may have an independent time-varying amount of traffic to be sent, and do not generate interference all the time.

The links that interfere with each other are represented using a conflict graph, whose vertices correspond to links of the network graph and the edges to the possible interference if the links were using the same channel.

Figure 1 illustrates the correspondence between a network topology, referred to as a network graph in Figure 1a, and a conflict graph in Figure 1b respecting the adopted interference model. Note that in Figure 1b, the links 2-3, 3-4, and 3-6 are not marked as interfering with each other in the conflict graph because we consider each node as having only one radio interface for data communication. More specifically, a node can only communicate with another single node at the same time, otherwise a collision will take place. We consider the MAC layer will be responsible for managing collisions in both the considered radio interfaces.

D. Problem statement

For small networks that do not show significant variations (due to node mobility or change in the channel availability), a suitable approach for channel assignment is to elect a node as the central decision entity. This central node receives the information about all the other nodes, uses a channel allocation function to obtain the minimum possible interference, and finally sends the determined channel assignment to all the other nodes. This, however, is unsuitable for networks with varying operation conditions.

In the case of CRNs, the main and the first issue to care about is to not to interfere with the communication of the PUs [15]. This condition suggests that the set of available channels is time variant according to the behavior of PUs. As a consequence, the communication network formed by SUs varies according to the behavior of the PUs. Therefore, the use of a distributed approach is suitable as messages are only exchanged by nodes affected by the change in the network. The use of such an approach derives benefit of the localized nature of interference.

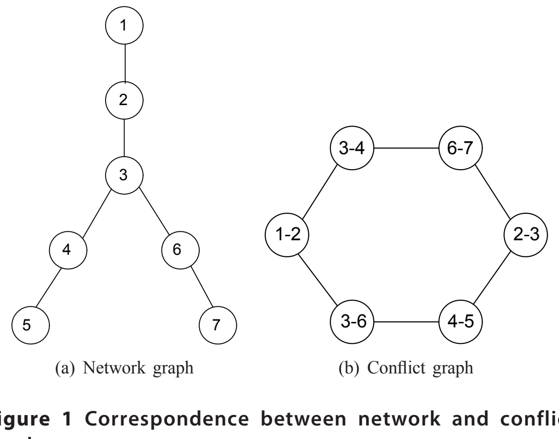

Figure 1 Correspondence between network and conflict graphs.

Once the channel sensing and characterization process end, SUs have to allocate a channel for communication to each link they have with neighbors. More specifically, the channel assignment problem corresponds to the selection of a single channel for each link (i.e., for each vertex of the conflict graph) from those available at the intersection between the ranked channel lists of the two nodes involved at the given link. This selection should be done so that two interfering links (i.e., vertices that have a common edge in the conflict graph) do not make use of the same channel.

Since a uniform traffic is assumed on all the links, the total network interference (I_{total}) is defined as the number of link pairs that interfere with each other, i.e., that they have the same channel assigned and are connected by an edge in the conflict graph. To analyze the efficiency of a channel assignment algorithm, we evaluate the removed interference after the assignment (I_{removed}) with respect to the interference when only one channel is available (i.e., the maximum interference $I_{\max}$), according to the following equation:

$$I_{\text{removed}}(\%) = \frac{I_{\max} - I_{\text{total}}}{I_{\max}}. \tag{1}$$

To obtain an efficient channel assignment, the aim then becomes *to maximize the removed interference (I_{removed})*, which results in an increase of the total network capacity [16].

3. The proposed ZAP algorithm

In this section, we present the ZAP algorithm for dynamic channel assignment in CRNs. By operating in a distributed way and above the MAC layer, ZAP requires only local processing on each node and the exchange of messages between nearby nodes.

According to the CR model presented in Section 2-A, the considered node has a radio interface permanently tuned on a CCC. Thus, when executed by each node, ZAP messages are exclusively exchanged at the CCC and never interferes with the node's data communication. Once the stopping stage of the channel assignment is reached, nodes accordingly tune their radio interfaces permanently used for data communication, to the channels dictated by ZAP. In this way, the proposed ZAP protocol aims at mitigating interferences by providing a suitable channel distribution among neighbor nodes.

ZAP operates through the exchange of two types of messages: `Hello` and `Interaction` messages. `Hello` messages are used by SUs to discover the two-hop neighbors (required at the construction of the communication and conflict graphs) and contains two lists. The first list contains the IDs of the source node and its neighbors: We consider that each node is assigned to an ID, which uniquely identifies it in the network (e.g., node's MAC address). The second list contains the channels available for each of those nodes. On the other hand, `Interaction` messages are used by each SU to propose a channel assignment to link with neighbors and to inform them about the SU priority in the assignment (further details in Section 3-C). In this way, channel assignments are first performed locally by each SU according to some criteria and then sent to its neighbors. Assignments proposed by high-priority SUs are accepted by neighbors upon the reception of their `Interaction` messages.

Note that `Hello` and `Interaction` messages are only exchanged through the CCC channel and in parallel to ongoing data communication occurring in the previously assigned channels. Hence, no additional data delay is imposed by the Interactions of the ZAP approach. In addition, the messages are only exchanged by neighbors affected by the change in the network, and thus, ZAP incorporates a distributed behavior and derives benefit of the localized nature of interference. It is worth noting that no partial solution for the channel distribution problem is allowed, thus avoiding any ping-pong effect. In fact, CR nodes shortly interrupt their data communication and perform frequency channel changes only after the ZAP algorithm reaches its stopping stage, i.e., after few SU Interactions (as discussed in Section 3-E). The definition of a stopping stage avoids the problem of convergence.

In short, the idea behind the ZAP protocol is thus to find a sub-optimal solution for the channel distribution problem, which allows stabilizing the channel assignment mechanism, simplifying the tackled problem as well as limiting processing and time costs, while providing performance results close to the centralized approach. To achieve this, ZAP operations based on a four-state machine shown in Figure 2 are detailed in the following.

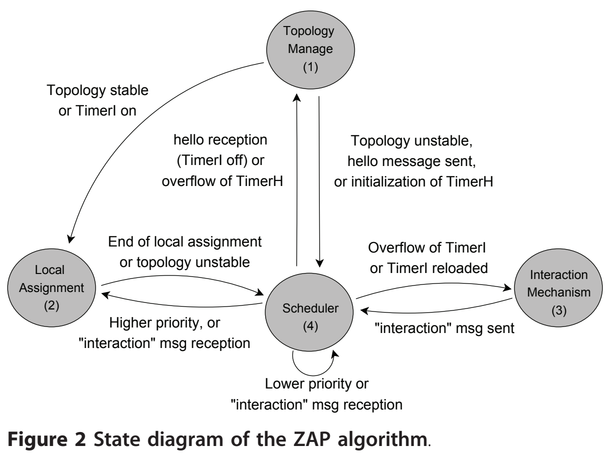
Topology Manage (1)
Topology stable or TimerI on
hello reception (TimerI off) or overflow of TimerH
Topology unstable, hello message sent, or initialization of TimerH
Local Assignment (2)
End of local assignment or topology unstable
Overflow of TimerI or TimerI reloaded
Interaction Mechanism (3)
Scheduler (4)
Higher priority, or "interaction" msg reception
"interaction" msg sent
Lower priority or "interaction" msg reception

Figure 2 State diagram of the ZAP algorithm.

A. State 1: topology manager

The Topology Manager is responsible for keeping the topology information up-to-date–the network and conflict graphs (NetGraph and ConflictGraph, respectively)–as well as the lists of links (LinkList) and available channels (ChannelList). As long as a stable scenario is not reached, periodic `Hello` messages are exchanged between neighboring nodes and the algorithm switches between states 1 and 4 (cf. Figure 2). When the stability condition is reached, the algorithm moves to state 2 and ceases the exchange of `Hello` messages. State 1 will only be active again in the case of changes either in the channel availability or in the network topology.

The time interval between two consecutive `Hello` messages is given by

$$\text{TimerH} = \frac{T_\text{h}}{2} + \text{random}\left[0, \frac{T_\text{h}}{2}\right], \qquad (2)$$

where T_h is the maximum desired value for the time interval between two consecutive `Hello` messages and random$[x, y]$ denotes a uniformly distributed value in the interval $[x, y]$. This avoids the synchronization of message transmissions and thereby minimizes the number of collisions. Still, we consider that the MAC layer is responsible for dealing with collisions.

Upon the arrival in state 1, the algorithm checks the event that triggered this state change. If it was caused by the reception of a `Hello` message from a neighboring node, then the network graph is updated. Otherwise, if the change was occasioned by the overflow of timer TimerH, then information stability is verified. If the network graph has remained the same between two consecutive message transmissions, then it is assumed that the information is stable. Then, TimerH is initialized, the conflict graph and list of links are built, and the algorithm flow jumps to state 2. Otherwise (i.e., unstable information), TimerH is reset and state 4 is activated.

B. State 2: local assignment

The local assignment is responsible for computing a preliminary channel assignment based on local knowledge of the node. Once this assignment is determined, the algorithm switches to state 4 and only returns to state 2 when an `Interaction` message is received from a node with higher priority decision.

Algorithm 1 details the procedure for Local Assignment. At the beginning, it creates four lists: *L*, containing the links of LinkList yet to be assigned; *C*, containing the lists of available channels to each link of *L* (obtained from ChannelList); InterferentList, empty to store interfering links; and AssignedList, containing the links already assigned of LinkList. As long as there are still links to be assigned in *L*, they are selected to be assigned according to the following criteria, applied sequentially:

1) the most restrictive link (i.e., the link with the lowest number of available channels);
2) the link with the highest probability of interference (i.e., largest number of edges in the conflict graph). It is worth noting that through this criterion, ZAP can operate with conflict graphs generated through any interference model, only needing the interference classification among links;
3) the link with the largest sum of degrees of its nodes (the degree of a node is the number of links of LinkList in which the node is part of);
4) the lowest link ID (e.g., the lowest MAC address), i.e., the one on top of *L*.

We tested several sets of criteria and selected the previously described order, which presented the best overall results. In the criterion sequence, a next criterion is applied only if there is an indecision between two or more links when applying the previous criterion.

The selected link resulting from the criteria execution is stored in the variable link and it is removed from the list *L*. If there is no available channel for link, it is marked as interferer and included in InterferentList for later assignment. A new link is then chosen respecting the above criteria. Otherwise, if at least one channel is available, the best channel among the ones available for link is stored in the variable *ch*. The classification of channels is done by the spectrum characterization module, as mentioned in Section 2-A. We assume that the channel list is sorted from the worst to the best channel, so that the best channel has the highest index. As the next step, the channel *ch* is assigned to link in the list AssignedList and *ch* is then removed from C in positions indexes of which correspond to links in *L*, which are connected to link in the conflict graph.

```
Algorithm 1: Local assignment–State 2
 Input: LinkList, ConflictGraph
 Output: AssignedList
 L $\Re$ links of LinkList yet to be assigned;
 AssignedList $\Re$ links already assigned from LinkList;
 while $L \neq \emptyset$ do
     select a link according to the criteria and store it in link;
     remove link from L;
     if no channel in C is available for link then
         insert link in InterferentList;
         continue;
     end
     ch $\Re$ best channel in C available for link;
     assign ch to link in AssignedList;
```

```
        foreach $l \in L$ do
            if $l$ is neighbor of link in ConflictGraph then
                remove ch from C in the position that corresponds to $l$;
            end
        end
    end
    foreach link $\in$ InterferentList do
        $c \Re$ channel of ChannelList available for link with the lowest occurrence among the neighbors of link in ConflictGraph;
        assign $c$ to link in AssignedList;
    end
```

The process is repeated until *L* has no more links to be assigned. At this moment, the links that were included in InterferentList will be assigned in the same order in which they were added to the list. Again, the variable link is used to store the link being processed. As the possible channels for link may have been assigned to another link, the originally available channels for link must be sought in ChannelList. Among these channels, the one which generates minimal interference with the already assigned links is selected, based on the ConflictGraph and the AssignedList. At the end, this channel is assigned to link in AssignedList.

C. State 3: Interaction mechanism

The Interaction mechanism is responsible for merging the channel assignments proposed by different nodes, based on the degree of knowledge of the network as a whole that each node possesses. While the stopping criterion is not reached, the nodes exchange `Interaction` messages at regular intervals, and the algorithm switches between states 3 and 4 (cf. Figure 2). A vector with three priority levels is used: (i) to express the degree of local network knowledge of a node and (ii) to define the order in which nodes first decide the channel assignment. Such a vector is represented by the parameters $[x, y, z]$ (as shown in the example of Figure 3), where

- x describes the total number of 1- and 2-hop links known by the node;
- y describes the number of 1-hop links known by the node;
- z describes the lowest node ID (chosen to ensure deterministic execution).

Figure 3 shows an example of the priority vectors of the nodes in a network graph. For this network, the descending order of knowledge degree (priority) of the nodes is: 3, 2, 4, 6, 1, 5, 7. Therefore, node 3 will first decide the assignment of links 2-3, 3-4, and 4-5; node 2 will decide the assignment of the link 1-2; node 4, the assignment of 4-5; node 6, the assignment of 6-7; and, finally, the remaining nodes will just accept the assignments already proposed by other nodes due to their lower priority.

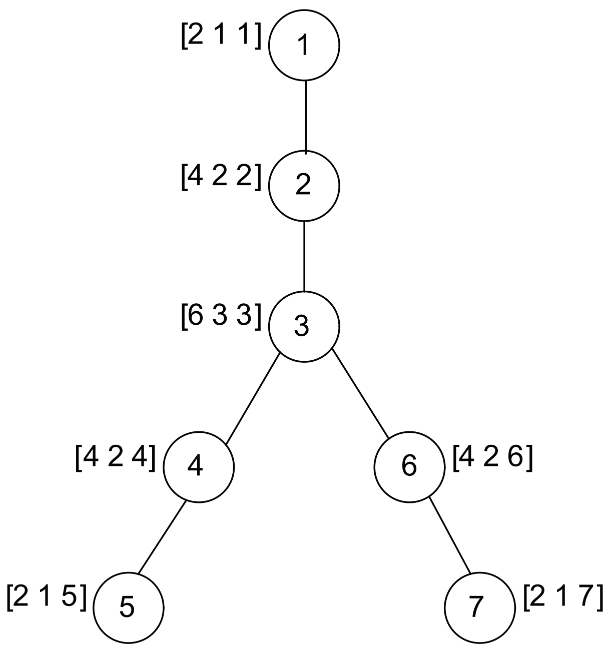

Figure 3 Network graph illustrating the priority vectors of each node.

The `Interaction` message created by a node consists of its priority vector and a list with its assignment for the 1-hop links. When the `Interaction` message is sent, the information it contains is no longer just a preview and becomes the assignment used for data communication. This assignment will only be modified in two situations: (i) when sending a new `Interaction` message; or (ii) when receiving an `Interaction` message coming from a node with higher degree of knowledge (priority). In the second case, the node does not have the permission to modify the received link assignment that was contained in the message and recomputes its channel assignment for the remaining links accordingly.

Similar to the case of exchanging `Hello` messages, again in order to hinder message collisions, the time interval between two consecutive `Interaction` messages is given by

$$\text{TimerI} = \frac{T_i}{2} + \text{random}\left[0, \frac{T_i}{2}\right], \tag{3}$$

where T_i is the maximum desired value for the time interval between two consecutive `Interaction` messages.

D. State 4: scheduler

The Scheduler is responsible for responding to the stimuli internal and external to the node (overflow of timers and received messages, respectively), and for blocking the process in the case of the prolonged absence of these stimuli. By identifying the received stimulus, state 4 causes the execution flow to deviate to any of the other states and, therefore, is called the Scheduler.

The time when an overflow of TimerH occurs is the moment to send a new `Hello` message, and the execution flow jumps to state 1. The flow also shifts to state 1 when receiving a `Hello` message and in this case, TimerI must be disabled to interrupt Interactions, because there were topology changes. If there is an overflow of TimerI, a new `Interaction` message should be sent and the flow jumps to state 3 after resetting TimerI. Finally, when an `Interaction` message is received, the Scheduler compares the priorities of both the message and the receiving node. If the message has higher priority, state 2 is called to recompute the assignment. Otherwise, the message is ignored and state 4 is maintained.

E. Complexity analysis of the ZAP algorithm

The complexity of the ZAP algorithm is basically related to the local assignment part (state 2). The other states do not require a large processing capacity, and therefore are not considered in the complexity analysis. As we can see in Algorithm 1, there are two nested loops with $|L|$ iterations each. Therefore, the local algorithmic complexity of ZAP is $O(|L|^2)$, where $|L|$ is the number of links known by the node. Thus, there are no scalability constraints from the point of view of the algorithm complexity as it depends only on the local node density, thus being *independent* of the total number of nodes.

In terms of exchanged messages, only the states 1 and 3 have an influence. The Topology Manager requires the exchange of an average of three messages until the nodes obtain full knowledge of their two-hop neighborhoods. The Interaction Mechanism dictates the exchange of a finite number of messages, corresponding to the established stopping criterion. More specifically, only six Interactions are required for the stopping criterion to be reached, regardless of the size of the network or the local node density (cf. Section 4).

4. Performance evaluation

In order to evaluate the ZAP performance, two other channel assignment methods are implemented for comparison purposes: the Centralized Tabu-Based Algorithm (CTBA) [6] and a random channel attribution method (RANDOM). We evaluate the performance of channel assignment by analyzing the percentage of removed interference achieved by the three considered strategies: CTBA, ZAP, and RANDOM. CTBA is centralized and presents minimum interference results, thus it is used as a reference for the upper bound on channel assignment performance. The performance evaluation only considers the first phase of CTBA because the second concerns the representation of communications restrictions [6], and these restrictions are not necessary to channel assignment. The RANDOM was designed to choose one channel for each link by using a uniform random function to select an available channel. RANDOM assignment incurs minimal cost and was used as a lower bound on the channel assignment performance in terms of percentage of removed interference. To simulate losses, the memoryless Gilbert-Elliot (GE) model [17,18] is used, as it is able to accurately approximate the behavior of a fading radio channel in burst applications. Parameters were adjusted so that the GE model yields 5% average packet loss and loss bursts of five packets in average.

The behavior of PUs was modeled considering an average inactive time at least ten times larger than the time needed for the algorithm to reach the stopping criterion. This behavior model does not impact ZAP results, since ZAP is only applied on channels previously sensed and characterized as unoccupied by the sensing function. Nevertheless, the effect caused by a variation in the number of available channels before ZAP arrives at the stopping criterion corresponds to the restart of the ZAP algorithm. The restart impact is irrelevant when considering this as an exception event. Therefore, the neighbors and lists of available channels were considered stable during all the simulated scenarios. Under these conditions, the removed interference of the ZAP algorithm was evaluated in four different scenarios: (i) varying number of available channels; (ii) varying average link density; (iii) varying number of network nodes; and (iv) varying stopping criterion. For each scenario, we generated 1000 random topologies to have, in the simulation results, a 95% confidence interval that is less than 1% around the mean.

Figure 4a presents results for random topologies with 100 nodes, mean network link density set to 5 with the number of available channels varying from 2 to 10. We considered the stopping criterion fixed at six Interactions–this value is selected from experiments shown in Figure 4d and discussed later in this section. We observe that the number of available channels directly impacts the efficiency of the three considered methods. As expected, the performance in terms of removed interference increases with the number of available channels. ZAP arrives near the upper bound achieved by CTBA, even when under 5% of losses, and presents better results as compared to RANDOM. ZAP performance tends to level off for a number of available channels larger than 8. This is an interesting outcome, as it suggests

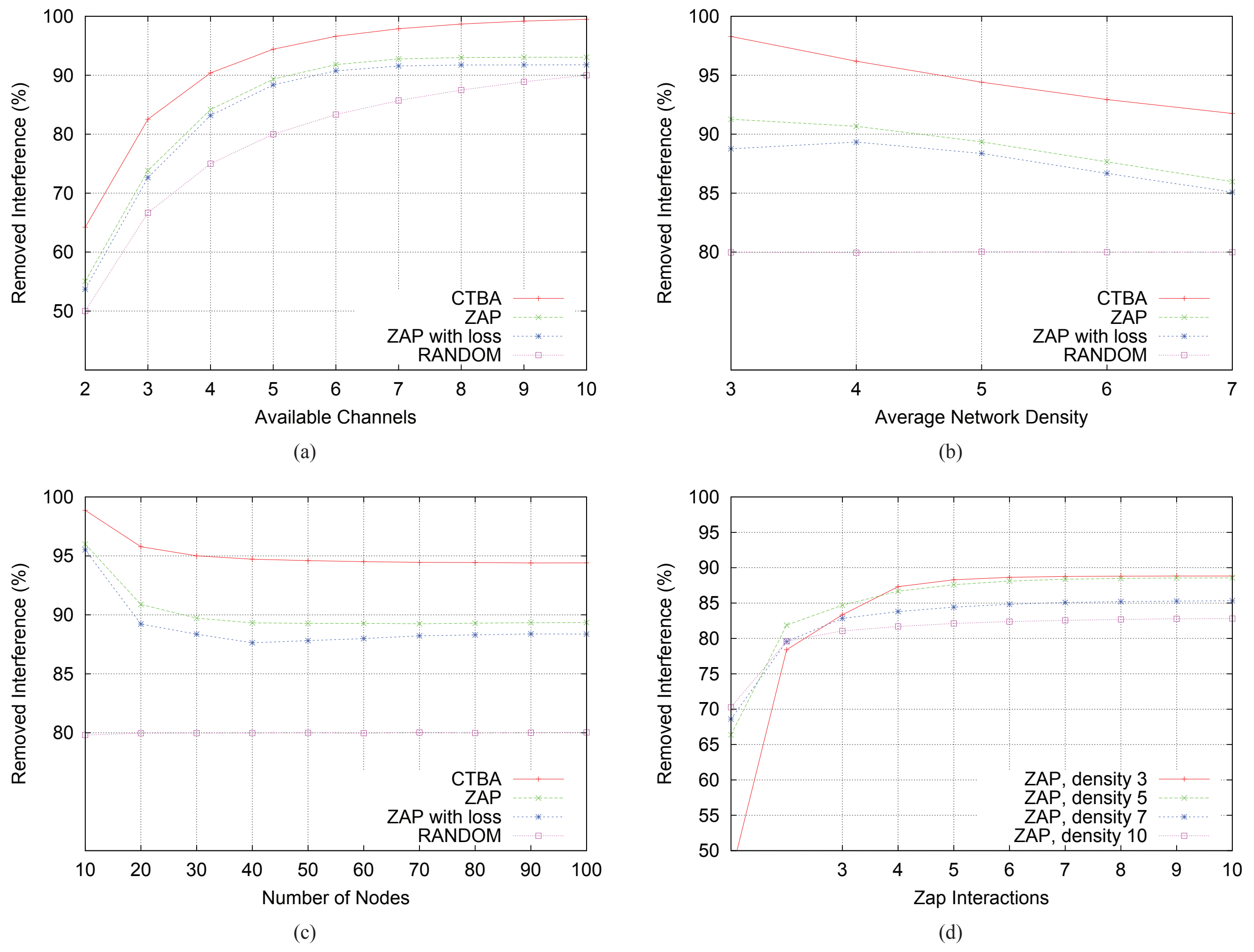

Figure 4 **Impact on removed interference considering different metrics**.

that the sensing period may be limited to only search for up to eight available channels without significant performance loss. Such a procedure would alleviate the load of the sensing function and the amount of demanded resources for the system. Note, however, that this limit on available channels to be identified is dependent on the local link density, but, for an average link density between 3 and 7, it can still be considered enough for ZAP to get results near the upper bound one. RANDOM assignment shows results whose pattern matches the function $1 - \frac{1}{c}$, where c is the number of available channels.

Figure 4b shows results for networks with 100 nodes, but the network link density mean is varied between 3 and 7. The scenario starts with 100 nodes in the considered area. The area is then reduced, increasing the number of neighbors per node. The number of available channels was fixed at 5 and, similar to the previous scenario, the stopping criterion was fixed at six Interactions. As shown in Figure 4b, the RANDOM performance is independent of network link density. In fact, even with increasing density, due to a fixed number of nodes and channels, the percentage of interfering links stay constant and equals to 20% of the total number of links, as discussed before. On the other hand, ZAP and CTBA present a decrease in their performances in denser topologies. This behavior is a consequence of a fixed number of available channels (e.g., 5) combined with an increasing number of neighbors (i.e., links), leading to an increasing level of interference.

Figure 4c presents the network expansion effect, with the number of nodes varying between 10 and 100. To understand this effect, the average network link density is considered constant and equal to 5. The area has the same node density (i.e., if the number of nodes is increased, the area also will be increased to maintain a constant average network link density). Similar to the previous result, the RANDOM performance is independent of the number of nodes, remaining dependent only on the number of channels. CTBA and ZAP have a decrease in performance between 10 and 40 nodes. This is because smaller topologies have higher probability of having some nodes with a high knowledge of the network (i.e., dense nodes). Such knowledge helps us to decrease the interfering links during the ZAP execution. Nevertheless, for more than 40 nodes, these methods do not have a significant variation in the interference reduction. In fact, large networks also tend to be more spread, which results in an increasing number of nodes with similar high densities. This helps us to converge the results.

Although having the upper bound results, CTBA is centralized and, consequently, has an exponential communication cost proportional to the number of network nodes. In contrast, ZAP does not have any scalability constraints (cf. Section 3-E), since it works in a decentralized way, exchanging messages only with two-hop neighbors. Moreover, a centralized algorithm needs a routing protocol or at least a flooding mechanism to send and receive control messages of the complete network. Therefore, CTBA suffers from all the well-known scalability constraints of a centralized method, whereas ZAP achieves relatively close performance without similar scalability constraints.

Figure 4d shows results for scenarios with 5% of message losses, using different random topologies with 100 nodes, 5 available channels, a varying number of network link densities, and a varying number of ZAP Interactions (i.e., ZAP's stopping criterion). It is worth noting that the number of channels was selected from the results observed in Figure 4a, in which the five available channels have shown good performances in terms of removed interference for all the strategies. We then vary the number of Interactions to reach the stopping criterion in the ZAP algorithm for different link densities. We remark that after six Interactions, performance levels off, without any significant gain for further Interactions. These results suggest that a suitable stopping criterion is six Interactions independent of link density. The same performance level was observed in simulations with network sizes up to 1000 nodes, though this testing considered only 100 topologies to reduce simulation time (massively increased by the number of total links). For the sake of clarity on the graph, we only show results obtained for the 100-node networks.

It is worth noting that, if less Interactions were used (e.g., three Interactions in Figure 4d, causing a reduction from 88 to 85% of removed interference), the side effect would be the increase of contention to be dealt by the MAC layer, since nodes will be operating in interfering channels. Nevertheless, *no error preventing the data communication among nodes will occur*. Data communication would be shortly interrupted to assign nodes to the new channels only once such three Interactions were performed, after such three Interactions and communication would continue through a non-optimized channel distribution. We have, however, shown that ZAP, although being a localized and distributed approach, provides results close to the optimized solution given by the centralized TABU approach (cf. Figure 4a,b,c).

5. Related work

The channel assignment problem has been largely investigated in the literature related to multi-radio wireless ad hoc and mesh networks [19-22]. For instance, the solutions proposed in [20,21] are centralized approaches, and aim to limit interference while preserving connectivity [21] or while considering nodes traffic [20]. More complete approaches study the channel assignment problem in conjunction with the routing problem [19,22]. Although presenting interesting solutions, they disregard the challenges imposed by CR networks. In fact, the challenges in performing channel assignment in CRNs arose from different issues compared with ad hoc and meshed networks, even though some channel assignment solutions can be found in the literature for these latter type of networks. In particular, when multi-channels are considered in mesh or ad hoc networks, they are usually well-known channels and are *a priori* defined. Instead, nodes in CRNs make use of the underutilized portions (i.e., white holes) of the licensed frequency spectrum as new communication opportunities. Such communication opportunities are, however, highly time-variable and are usually possible only for short periods of time. In addition, no well-known or *a priori*-defined set of channels can be considered. Finally, primary nodes have higher priorities in the communication process, requiring a constant verification of their presence apart from secondary nodes. All these factors add a lot of dynamism and, consequently, new challenges to the channel assignment problem in CRNs.

The channel assignment problem for CRNs has been previously addressed by centralized [6] and distributed [7,8] approaches. Li and Zekavat [7] present different methods to channel assignment using the CR and cluster techniques. The proposed methods–five in total–mitigate the need of a central controller and reduce the overhead of the CRNs. This is achieved by clustering the CR nodes and electing a cluster-head to drive the channel assignment on each cluster forming a hierarchical structure. One of these methods (the fourth one) is comparable to ZAP as it proposes a channel assignment based on the interference level. Nevertheless, this method proposes a random choice of the cluster-head, and the channel assignment is done using an ascendant order of the interference level. In ZAP, in contrast, the choice of the channels is based on a modified interference model to take into account the two-hops neighborhood (Section 2-C) and uses only local knowledge (neighborhood) in a flat distributed way.

Huang et al. [15] analyze throughput performance bounds in CRNs. Their aim is to mainly maintain the protection of the PUs without any degradation of the throughput performance of the CR nodes. The impact of the interference is not considered in the throughput performance of both the PUs and the CR nodes.

Shiang and van der Schaar [8] investigate the management problem of multiuser resources in CRNs for delay-sensitive applications. They propose a distributed algorithm based on local information through the adoption of a multiagent learning concept (i.e., adaptive fictitious play) that utilizes the available interference information. In fact, the proposed channel distribution is based on the learning of the behavior of PUs and should be repeated for each changing in such behavior. Hence, there exist a mandatory information exchange and a special cost to the learning phase. Moreover, the performance of their proposed study is dependent on the low variability of applications and network conditions.

ZAP analyzes the interference models to multihop wireless networks and proposes a distributed proposal for the channel assignment problem in a CRNs. Our proposal mitigates the interferences among the two-hop simultaneous transmissions. ZAP thus achieves an efficient tradeoff compared with centralized and random strategies, in terms of balancing optimal channel assignment and low communication overhead.

6. Conclusion

In this article, we have proposed the ZAP algorithm to assign channels in a distributed manner in CR networks. The main contribution of our proposal is the ZAP capability of achieving an efficient channel assignment in a fully distributed manner only using the local knowledge (neighborhood) of each node, thus incurring a low message overhead. In this way, ZAP offers an efficient tradeoff in terms of an optimal channel assignment offered by a centralized solutions and a reduced message overhead to achieve this assignment. The results suggest that only after six Interactions, ZAP achieved 99% of the performance reachable if we had infinite Interactions (independently of the network size and density), proving the scalability of the proposal. Further, the ZAP algorithm guarantees a distributed optimization of the network capacity by reducing the number of interferences.

This proposal opens also possibilities for new future works. We plan to investigate the following issues: (i) evaluation of new parameters to the priority vector; and (ii) include routing and QoS metrics–weights–in links, prioritizing the interference mitigation over the higher weights links.

Abbreviations

CCC: common control channel; CR: cognitive radio; CRNs: cognitive radio networks; CTBA: centralized Tabu-Based Algorithm; GE: Gilbert-Elliot; Pus: primary users; RANDOM: random channel attribution method; SUs: secondary users.

Acknowledgements

This study has been partially supported by CNPq, FAPERJ, and Fundação Araucária (Brazilian agencies).

Author details

[1]Pontifical Catholic University of Paraná (PUC-PR), Brazil [2]Federal Technological University of Paraná (UTFPR), Brazil [3]INRIA, France [4]National Laboratory for Scientific Computing (LNCC), Brazil

Competing interests

The authors declare that they have no competing interests.

Received: 11 November 2010 Accepted: 4 July 2011
Published: 4 July 2011

References

1. S Haykin, Cognitive radio: brain-empowered wireless communications. IEEE J Sel Areas Commu. **32**(2), 201–220 (2005)
2. K Hicham, M Naceur, F Serge, Multihop cognitive radio networks: to route or not to route. IEEE Netw. **23**(4), 20–25 (2009)
3. Y Zhao, S Mao, JO Neel, JH Reed, Performance evaluation of cognitive radios: metrics, utility functions, and methodology. Proc IEEE. **97**(4), 642–659 (2009)
4. J Mitola III, GQ Maguire-Junior, Cognitive radio: making software radios more personal. IEEE Personal Commun. **6**(4), 13–18 (1999). doi:10.1109/98.788210
5. W Cheng, X Cheng, T Znati, X Lu, Z Lu, The complexity of channel scheduling in multi-radio multi-channel wireless networks, in *INFOCOM (IEEE, 2009)*, 1512–1520 (2009)
6. AP Subramanian, H Gupta, SR Das, J Cao, Minimum interference channel assignment in multiradio wireless mesh networks. IEEE Trans Mobile Comput. **7**(12), 1459–1473 (2008)
7. Li X, SA Zekavat, Distributed channel assignment in cognitive radio networks, in *Proc of the 2009 International Conference on Wireless Communications and Mobile Computing: Connecting the World Wirelessly*, Leipzig, Germany (2009)
8. H-P Shiang, M van der Schaar, Distributed resource management in multihop cognitive radio networks for delay-sensitive transmission. IEEE Trans Veh Technol. **58**(2), 941–953 (2009)
9. IF Akyildiz, W-Y Lee, KR Chowdhury, CRAHNs: cognitive radio ad hoc networks. Ad Hoc Netw. **7**(5), 810–836 (2009). doi:10.1016/j.adhoc.2009.01.001
10. IEEE 802.11 Standard–2007. IEEE P802.11, C1–1184 (2007)
11. ME Sahin, H Arslan, System design for cognitive radio communications, in *Proc Int Conf on Cognitive Radio Oriented Wireless Networks and Commun (CrownCom 2006)*, Mykonos Island, Greece (2006)
12. J Zhao, H Zheng, G-H Yang, Distributed coordination in dynamic spectrum allocation networks IEEE DySPAN (2005)
13. I Akyildiz, W-Y Lee, MC Vuran, M Shantidev, NeXt generation/dynamic spectrum access/cognitive radio wireless networks: a survey. Elsevier Comput Netw. **50**(13), 2127–2159 (2006). doi:10.1016/j.comnet.2006.05.001
14. J Padhye, S Agarwal, VN Padmanabhan, L Qiu, A Rao, B Zill, Estimation of link interference in static multi-hop wireless networks, in *Proc of the 5th ACM SIGCOMM conference on Internet Measurement (IMC)*, Berkeley, CA (2005)
15. S Huang, X Liu, Z Ding, Optimal transmission strategies for dynamic spectrum access in cognitive radio networks. IEEE Trans Mobile Comput. **8**(12), 1636–1648 (2009)
16. P Gupta, PR Kumar, The capacity of wireless networks. IEEE Trans Inf Theory. **46**(2), 388–404 (2000). doi:10.1109/18.825799

17. EO Elliott, Estimates of error rates for codes on burst-noise channels. Bell Syst Tech J. **42**, 1977–1997 (1993)
18. EN Gilbert, Capacity of a burst-noise channel. Bell Syst Tech J. **39**, 1252–1265 (1960)
19. A Raniwala, T Chiueh, Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network. in *IEEE Infocom* 2223–2234 (2005)
20. KN Ramachandran, EM Belding, KC Almeroth, MM Buddhikot, Interference-aware channel assignment in multi-radio wireless mesh networks, in *IEEE Infocom*, 1–12 (2006)
21. J Tang, G Xue, W Zhang, Interference-aware topology control and qos routing in multi-channel wireless mesh networks, in *ACM MobiHoc*, 68–77 (2005)
22. S Avallone, IF Akyildiz, G Ventre, A channel and rate assignment algorithm and a layer-2.5 forwarding paradigm for multi-radio wireless mesh networks. IEEE/ACM Trans Netw. **17**(1), 267–280 (2009)

doi:10.1186/1687-1499-2011-27
Cite this article as: Junior *et al.*: ZAP: a distributed channel assignment algorithm for cognitive radio networks. *EURASIP Journal on Wireless Communications and Networking* 2011 **2011**:27.