Seleção dinâmica de Canais de Controle em Rádios Cognitivos utilizando *Reinforcement Learning* *

Raphael M. Guedes[1] **e José Ferreira de Rezende**[1]

[1]Grupo de Teleinformática e Automação (GTA) – COPPE
Universidade Federal do Rio de Janeiro (UFRJ)

{raphael,rezende}@gta.ufrj.br

Resumo. *Os rádios cognitivos têm despontado como uma tecnologia promissora no compartilhamento de espectro entre usuários secundários (SUs). Como as dificuldades encaradas por esta solução são diversas, muitas propostas partem da premissa de que existe um canal, denominado canal de controle, no qual os SUs podem trocar mensagens de coordenação quando necessitarem. No entanto, em decorrência de uma série de fatores, a existência de um único canal de controle, disponível para todos os SUs, pode ser pouco provável. Como solução, pode-se utilizar uma abordagem multicanal, onde salta-se de canal em canal com o intuito de encontrar outros SUs para a troca de informações. Desta forma, deve existir uma sequência de troca de canais, na qual a maior parte de SUs possam se comunicar. Este trabalho discute o uso de Reinforcement Learning como uma possível ferramenta para descobrir esta sequência de canais.*

Abstract. *Cognitive radios have emerged as a promising technology in spectrum sharing among secondary users (SUs). Due to difficulties faced by this solution, many proposals consider the existence of a channel, called control channel. Through this channel SUs can exchange coordination messages. However, due to a number of factors, it is unlikely that a single control channel is simultaneously available to all SUs. One solution is to use a multi-channel approach, where nodes can hop from channel to channel in order to find other users to exchange information. Thus, there must be a channel sequence, which most of users can communicate. This paper discusses the use of Reinforcement Learning as a possible tool for discovering those channel sequences.*

1. Introdução

Um rádio cognitivo (RC) [Mitola 1999], conforme definido em [Haykin 2005], é um sistema de comunicação sem fio inteligente, capaz de utilizar metodologias de aprendizagem e compreensão para adaptar seus parâmetros operacionais à dinâmica do ambiente ao seu redor [Galindo-Serrano et al. 2010]. Como aplicação mais comum, pode-se citar a exploração por espectros de frequências. Assim, espectros que estejam temporariamente disponíveis podem ser oportunisticamente utilizados, permitindo desta maneira um uso mais eficiente deste recurso.

*Este trabalho recebeu recursos do CNPq, CAPES, FAPERJ, FINEP e RNP.

Em um contexto multiusuário, este conceito é muitas vezes referido como DSAN - *Dynamic Spectrum Access Networks* [Akyildiz et al. 2006], onde dispositivos não autorizados ou usuários secundários (SU - *Secondary Users*) são possibilitados a utilizar o espectro atribuído a um usuário primário (PU - *Primary Users*) licenciado, quando este espectro não estiver ocupado. A estes períodos de espectros temporariamente vagos dá-se o nome de "espaços em branco" (*whitespaces*). O uso oportunista destes "espaços em branco" pelos SUs, está condicionado à uma tolerância de baixo nível de interferência sobre os nós PUs [Ren Y. and P. 2010].

Pesquisas em RCs abrangem amplas áreas, incluindo: análise de espectro, estimação/seleção de canais, compartilhamento de espectro, controle de acesso ao meio e roteamento [Chen et al. 2007]. Uma característica comum existente em grande parte das abordagens, independentemente das soluções propostas e dificuldades tratadas, é a consideração de uso de um canal livre e exclusivo denominado canal de controle. Pelo canal de controle transitam informações, trocadas entre SUs, necessárias para o bom funcionamento da rede, como indicativos sobre compartilhamento e gerenciamento de frequências. Entretanto, a existência de um canal de controle único disponível para todos os SUs de uma região pode ser pouco provável. Isto ocorre devido aos efeitos causados pela atividade heterogênea dos nós PUs [Chen et al. 2007].

Os RCs quando utilizados numa topologia em malha chamada de Redes em Malha Cognitiva (CWM - *Cognitive Wireless Mesh* [Tao Chen and Katz 2009]), sofrem ainda mais com o problema do canal de controle. Este tipo de rede tem por definição ser autogerenciável, logo o acesso dinâmico ao espectro (DSA - *Dynamic Spectrum Acesss*) é possibilitado pela troca de informações através do canal de controle. Consequentemente, mensagens perdidas no canal de controle podem prejudicar severamente a rede, pois um nó SU ao perder alguma mensagem de controle, pode ficar sem a indicação de qual canal será utilizado em alguma comunicação futura.

Soluções para CWM promovem o agrupamento dos nós em *clusters*, o que facilita o gerenciamento e aumenta a probabilidade de que exista um canal livre para controle entre os nós. Porém, existem situações que em função da disposição dos PUs e suas atividades, nem todos os componentes do *cluster* possuem canais livres em comum. Desta forma, ao invés de se empregar um único canal como de controle, podemos utilizar vários canais em sequência para a transmissão das mensagens de coordenação. Assim, aumentam-se as chances de comunicação entre os nós. Ideia semelhante pode ser encontrada na solução chamada de *Channel Hopping* [Tang and Garcia-Luna-Aceves 1999], onde os nós em uma rede *Ad Hoc* trabalham com uma sequência de chaveamento de canais em suas transmissões, porém o objetivo é atacar o problema de terminais escondidos devido o uso de múltiplos canais.

Logo, propomos utilizar estas duas soluções em conjunto, a fim de contornar o uso exclusivo de um único canal como de controle a todos os nós SUs componentes de um *cluster*. No entanto, a ideia do *Channel Hopping* quando aplicada ao ambiente de RCs, deve possuir algumas ressalvas quanto a sequência de transmissão utilizada, pois a solução originalmente não necessita diferenciar os nós nem preocupa-se com interferências sobre as transmissões de nós PUs. Surge uma dificuldade que é descobrir uma sequência de chaveamento de canais, em que os nós recebam mensagens de controle e não interfiram nos usuários licenciados que estejam ao seu alcance. Além disso, em de-

corrência da atividade de PUs, a sequência de canais pode se alterar no decorrer do tempo, com isso os nós SUs devem "aprender" quando a sequência, até então utilizada, não é mais adequada.

Para a questão da escolha da sequência, propomos o uso de uma máquina de aprendizagem, chamada *Reinforcement Learning* (RL). Desta maneira, a máquina de aprendizagem será encarregada de descobrir uma sequência de chaveamento de canais que leve a um melhor desempenho. Além disso o RL, por característica, possui uma complexidade computacional baixa o que o torna atrativo para o uso embarcado em RCs.

Com este intuito, o atual trabalho está organizado da seguinte forma: a Seção 2 discute alguns dos trabalhos em RCs relacionados ao problema do canal de controle. A Seção 3 apresenta em maiores detalhes a máquina de aprendizagem RL. A modelagem do problema em conjunto com suas considerações são objetos de estudo da Seção 4. Em seguida, a Seção 5 traz os cenários de testes e os resultados das simulações realizadas, assim como uma discussão a respeito. E, por fim, a Seção 6 apresenta as conclusões e trabalhos futuros.

2. Trabalhos Relacionados

Nos últimos anos a pesquisa na área de rádios cognitivos se intensificou, justamente por tratar-se de uma tecnologia promissora, vista como uma alternativa ao problema da disponibilidade de espectro de frequências entre usuários não licenciados. Além disso, é uma área com diversos desafios que podem ser tratados separadamente. Logo, o número de publicações e problemas abordados são vastos, por exemplo: análise de espectro, seleção de canais, compartilhamento de espectro etc.

Como consideração, em grande parte destes trabalhos, os nós possuem um canal exclusivo por onde trafegam as mensagens de controle trocadas entre os usuários SUs, e desta forma a rede pode funcionar. Porém, existem alguns problemas devido este tipo de comportamento. Como os canais que os nós SUs utilizam são compartilhados com os nós PUs, existe uma dificuldade de se encontrar um canal disponível que seja global, e funcione como canal de controle. Ou seja, a atividade e o número de PUs influenciam na disponibilidade de canais para os nós SUs. Com o intuito de tratar destes problemas, surgiram algumas propostas que em geral são divididas conforme a quantidade de interfaces por RC.

Existe um método chamado de Canal de Controle Comum (*CCC - Common Control Channel*), que procura por um canal que esteja livre momentaneamente para a sinalização de todos os nós. O caso típico é o uso de um MMAC (*Multi-Channel Medium-Access Control*) [So and Vaidya 2004], que define um canal de controle padrão, onde todos os nós devem periodicamente chavear para se sincronizarem em uma janela de tempo predeterminada. O canal de transmissão de dados é negociado nesta janela. Esta abordagem é mais comum em caso de múltiplas interfaces, onde existe a utilização contínua de uma interface exclusiva para controle e assim, não há necessidade de chaveamento para um canal padrão. Este método, possui a problemática de nem sempre encontrar um canal livre que possa ser utilizado como de controle.

Outra possibilidade, mais aplicada a topologias CWN, sugere que os nós se organizem em *clusters*, pois as chances de se possuir um canal que seja comum entre os componentes do *cluster* é maior que um único canal global para toda rede

[Tao Chen and Katz 2009]. Porém, mesmo neste caso, pode acontecer de não existir um único canal de controle que possa ser compartilhado entre os nós componentes.

Uma outra ideia, que pode ser aproveitada em RCs é a *Channel Hopping*. Neste método cada nó pode trocar de canais de acordo com um determinado padrão, e assim cada nó tem a chance de se encontrar com outros nós periodicamente em diferentes canais. Como exemplo, pode-se citar o HRMA (*Hop-Reservation Multiple Access*) utilizado como solução em uma rede *Ad Hoc* para atacar o problema de terminais escondidos. Entretanto, quando aplicado a RCs, temos dificuldades relacionadas a definição da sequência de saltos de canais, visto que esta deve se preocupar com a interferência causada sobre as comunicações dos nós primários. Salvo tal dificuldade, esta abordagem apresenta uma maior flexibilidade de adaptação quando comparada as outras soluções, pois não se prende a apenas um canal, sendo desta forma mais robusta que as soluções anteriores.

Em uma outra solução chamada *Home Channel*, um canal predefinido é atribuído a cada nó (seu *Home Channel*) e os nós retornam imediatamente para este canal quando ociosos, para o recebimento de mensagens de controle [Tao Chen and Katz 2009]. Uma forma de atribuir este canal antecipadamente é associar sua escolha ao endereço MAC de cada nó [PalChaudhuri et al. 2003]. Esta é uma solução simples que se aplica bem quando a atividade de PUs não é intensa. Caso a atividade de PUs seja heterogênea, pode acontecer do *Home Channel* ser um canal que impossibilite o nó de receber mensagens de controle.

Existem outras formas de solução que utilizam ferramentas de aprendizagem. Uma ferramenta que vem sendo utilizada para solucionar problemas em RCs é a máquina de aprendizagem do tipo RL. O uso de RL em RCs vem crescendo devido a sua baixa complexidade computacional e consequentemente maior viabilidade de ser embarcada. Como possível utilização, pode-se citar a seleção dinâmica de canais [Yau et al. 2010b, Yau et al. 2009], controle de congestionamento, escalonamento, entre outras encontradas em [Yau et al. 2010a].

Neste trabalho, a proposta é utilizar em conjunto a solução *Channel Hopping* adaptada para RCs e a ferramenta RL para encontrar uma sequência de chaveamento de canais de controle, em uma topologia de organização em *cluster*, típica de redes CWM. Na Seção 3 apresentaremos a ferramenta RL e após, na Seção 4, mostraremos como modelamos o problema.

3. Máquina de Aprendizagem *Reinforcement Learning*

Na área da ciência da computação, *Reinforcement Learning* é um tipo de máquina de aprendizagem que preocupa-se com a maneira com que um agente realiza ações em um ambiente, de modo que alguma métrica, referida como recompensa, seja maximizada. Resumidamente, em RL temos agentes que, ao inspecionarem um ambiente, realizam alguma ação que se reflete em uma métrica, também coletada do ambiente, chamada de recompensa ou custo. A partir do valor da recompensa coletada, o agente percebe se realizou uma boa ação ou não.

O RL adota uma abordagem simples, cuja a complexidade envolvida na modelagem do meio pode ser minimizada [Yau et al. 2010a]. Em contrapartida a esta facilidade de adaptação e baixa complexidade computacional, esta ferramenta pode possuir o inconveniente de ter uma convergência lenta.

Esta máquina de aprendizagem possui diversos algoritmos como SARSA (*State-Action-Reward-State-Action*), ACLA (*Actor Critic Learning Automaton*) [Wiering and van Hasselt], *Q-learning* [Sutton and Barto 1998] etc. A principal diferença destes algoritmos se dá na forma em que as ações são contabilizadas em recompensa, com pesos dados as ações, ao histórico de ações e/ou expectativa de recompensas futuras. Apresentaremos em mais detalhes o algoritmo do tipo *Q-learning*.

Q-learning é um algoritmo *on-line* do tipo RL que busca determinar uma política/decisão ótima, sem uma modelagem detalhada do ambiente. Este algoritmo possui alguns parâmetros apresentados a seguir:

- **Épocas de decisão** $\rightarrow$ denotadas por $t \in T$, $T = \{1, 2, ...\}$. Indicam quando o algoritmo entra em execução;
- **Estados** $\rightarrow s \in S$. São próprios da modelagem de cada problema;
- **Ações** $\rightarrow a \in A$. Representam as decisões que podem ser realizadas;
- **Recompensas** $\rightarrow r_t(s_t, a_t)$. Relativas a um par estado-ação, é o indicativo do quão bom ou não foi a **ação** realizada;
- ***Q-value*** $\rightarrow Q_t(s_t, a_t)$. Indica o valor que é atualizado através da coleta de **recompensas**;
- ***Q-table*** $\rightarrow$ Tabela onde os **estados** são as linhas e as colunas são as **ações**, e onde os ***Q-value's*** são armazenados. Possui $S \times A$ entradas.

Cada agente mantém um *Q-value* $Q_t(s_t, a_t)$ para todas as ações possíveis em sua *Q-table*. O *Q-value* estima o nível de recompensa para o par estado-ação, assim mudanças nos *Q-value's* levam a mudanças nas ações dos agentes.

A cada Época de decisão t, o agente observa seu estado atual (linha) e escolhe uma ação (coluna) em sua *Q-table*. Posteriormente a execução da ação, o agente recebe uma recompensa r_t relativa a esta ação a_t selecionada.

Com o valor da recompensa, o agente atualiza seu respectivo *Q-value* no tempo $t + 1$ conforme:

$$Q_{t+1}(s, a) \leftarrow (1 - \alpha) \times Q_t(s_t, a_t) + \alpha r_{t+1}(s_t, a_t) \quad (1)$$

Na Equação 1, α é a taxa de aprendizagem; logo, maiores valores de α indicam maior peso a valores recentes de recompensa. Esta equação de atualização foi a mesma utilizada em [Yau et al. 2010a].

O *Q-learning* inicia com toda a *Q-table* zerada e a cada instante o agente seleciona uma ação baseada em uma estratégia de exploração. Uma estratégia comumente usada é a ε-*greedy* [Sutton and Barto 1998], que seleciona uma ação gananciosa, $\arg max_a Q(s, a)$, com alta probabilidade e, ocasionalmente, com uma pequena probabilidade seleciona uma ação uniforme e aleatória. Isso tenta garantir que todas as ações e seus efeitos sejam experimentados [Kok and Vlassis 2006].

Na seção seguinte, apresentaremos como utilizamos o RL para tratar do problema da seleção do canal de controle em uma topologia CWM.

4. Modelagem do problema

Conforme apresentado, a proposta é adaptar a solução de *Channel Hopping* para RCs, em conjunto com o RL em uma topologia do tipo CWM, onde a tarefa de encontrar a

sequência de saltos de canais será desempenhada pelo RL.

Primeiro apresentaremos como a solução *Channel Hopping* foi aplicada a RCs e depois como o modelo de RL foi elaborado.

4.1. *Channel Hopping* em RCs

Na modelagem, consideramos três tipos de nós componentes, a saber: nós PUs, nós SUs comuns e nó SU de controle. A modelagem será aplicada exclusivamente aos nós SUs e, conforme temos diferentes SUs, seus comportamentos também serão distintos.

Todos os nós possuem apenas uma interface de rede; desta maneira, os nós SUs devem alternar seu tipo de funcionamento. Por um intervalo de tempo o nó deve se preocupar com as mensagens de controle e, em outros instantes pode trocar dados com outro SU. Assim, definimos dois períodos, um chamado de período de controle e outro de período de utilização. Este procedimento é necessário devido os SUs não poderem utilizar permanentemente um canal escolhido, pois suas transmissões podem interferir em algum nó PU que até então estava fora de atividade, seja por questões espaciais ou temporais. Durante o período de controle é que o método entrará em execução. Na Figura 1 apresentamos o comportamento do nó SU de controle.

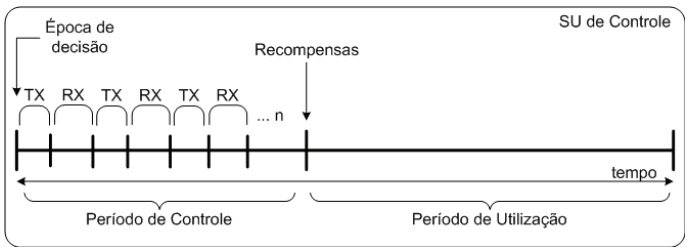


Figura 1. Método de funcionamento.

A transmissão da mensagem de controle é realizada pelo SU de controle, enquanto os nós SUs comuns confirmam com *ack* o recebimento desta mensagem. No início do período de controle (Época de decisão), a primeira atividade do nó é recorrer a algum mecanismo para decidir qual será seu canal de controle. Na parte TX do período de controle é que o nó irá realmente transmitir pelo canal escolhido. Na parte RX, o nó SU de controle reservará um tempo para o recebimento de *acks* de seus vizinhos. Este recebimento deve ser sincronizado, com um *slot* de tempo reservado para cada nó SU vizinho do nó SU de controle, justamente para se evitar uma colisão de *acks*. O comportamento dos nós SU comuns é o inverso. Na Época de decisão eles também recorrem ao mecanismo de decisão, porém para determinar em qual canal eles irão "ouvir". Depois, no tempo reservado, os SUs comuns passam para a etapa em que cada um irá transmitir seu *ack* no instante apropriado. Este procedimento se repete a cada conjunto TX-RX da Figura 1. Assim, durante a Época de decisão, o nó SU de controle terá uma sequência de canais em que ele irá transmitir, enquanto os SUs comuns terão uma sequência de canais onde eles irão "ouvir" pela mensagem de controle.

Este esquema pode ser repetido até que uma sequência seja estabelecida e, depois disso, a mensagem de controle possa ser transmitida adotando esta sequência. Como consequência o tamanho da sequência indicará a quantidade de conjuntos TX-RX.

Antes dos nós iniciarem todo este funcionamento é necessário que exista uma coordenação entre eles, justamente para definir qual dentre os nós será o SU de controle. Existem diversas possibilidades de se chegar a tal configuração. Como primeiro passo, deve ser feita uma descoberta de vizinhança justamente para cada nó ter a indicação de quem são seus vizinhos. A dificuldade é que não existe um canal exclusivo de controle em que os nós possam se comunicar afim de obterem tal informação.

Como solução, cada nó pode varrer o meio e, ao primeiro canal livre de atividade de PUs, o nó opte por transmitir uma mensagem como um *hello*, informando sua identificação, ou opte por apenas ouvir o meio. Todos os nós realizam semelhante procedimento; caso nenhum nó receba *hello* algum, pode ter ocorrido de todos terem optado por ouvir. Logo, repete-se o procedimento até que algum nó escolha transmitir. Ao primeiro recebimento de algum *hello*, este nó receptor responde com um *ack* em oportunidade breve. No *ack*, o nó informa quantos vizinhos ele tem conhecimento até o momento. Assim, cada nó verifica se existe algum outro nó com maior quantidade de vizinhos que ele próprio. Desta forma, o nó mais indicado a ser o SU de controle é o que possuir maior número de vizinhos. Assim que um nó verificar que é o que possui mais vizinhos, ele notificará que será o SU de controle. Neste procedimento podem ocorrer colisões de mensagens pois nenhum nó está sincronizado; logo, a eleição do líder pode ser demorada.

Como o nó SU de controle sabe a quantidade de vizinhos que possui, ele pode construir um esquema de transmissão e recepção do tipo TDMA (*Time Division Multiple Access*), e informar quando cada nó terá a oportunidade de TX e RX. Esta ordem pode ser atribuída pelo número de identificação de cada nó. Para o bom funcionamento é preciso uma sincronização, assim a cada mensagem trocada entre os nós, já no período de utilização do mecanismo de decisão, pode-se incluir informações de *clock* que auxiliem no sincronismo. Existem outras maneiras encontradas em livros de grafos como [Berge 1982]. Por exemplo, pode-se inciar por um algoritmo de cobertura mínima (*Minimal Spanning Tree*) seguido por um outro de eleição de líder, que opte pelo nó que tenha o grau mais elevado.

Conjuntamente com este método de acesso, propomos um mecanismo de decisão do tipo RL para determinar qual canal utilizar. A forma como foi implementado este mecanismo de decisão será apresentada na subseção a seguir.

4.2. Algoritmo do RL

A modelagem do problema consistiu em o que definimos como estados, ações, recompensas e como atualizamos os valores da *Q-table*. Realizamos duas modelagens chamadas RL-Ciclo e RL-Seq, que serão apresentadas a seguir. Nelas explicaremos o que foi definido como ações, estados e recompensas.

Em ambas modelagens, a recompensa para o nó SU de controle foi a quantidade de *acks* recebida, enquanto para os nós SU comuns representou o recebimento da mensagem de controle. A atualização da *Q-table* seguiu a Equação 1 apresentada anteriormente e ambos nós SUs utilizaram o Algoritmo 1 a cada Época de decisão.

Conforme o Algoritmo 1, no início, todos os pares estado-ação da *Q-table* foram completados com zero. Realizada esta fase de inicialização de variáveis, entramos na fase de aprendizado. Nesta fase, em decorrência do valor de ε e a sorteado, o nó pôde optar por exploração ou não. Quando optou por exploração, uma ação aleatória foi selecionada

dentre as possíveis no estado atual. Quando não realizou exploração, o nó no estado atual buscou pelo maior valor até então preenchido de *Q-value*, e assim selecionou esta ação. Quando todos os campos eram zero, o nó escolheu uma ação aleatoriamente. O procedimento de selecionar exploração ou não, e depois o maior valor até então obtido, é devido a utilização da estratégia ε-*greedy*.

Após, executou-se a ação, recolheu-se a recompensa adquirida e atualizou-se o valor da *Q-table*. Ao fim adicionamos um teste para agilizar a convergência, quando o *Q-value* obtido foi menor que um limiar δ, passamos o valor de *Q-value* para zero. Este procedimento de aprendizagem foi repetido durante todo período de funcionamento. A seguir apresentaremos como funcionam o RL-Ciclo e o RL-Seq.

Inicialização das variáveis;
begin
  $t = 0$
  for *para cada* $s \in S$, $a \in A$ **do**
    Inicializo ***Q-value*** para todas as **ações** a_t de cada **estado** s_t
end
Aprendizado:
repeat
  Gero um número aleatório a entre 0 e 1
  if *(*$a < \varepsilon$*)* **then**
    Seleciono uma **ação** a_t aleatoriamente $\rightarrow$ Exploração
  else
    Escolho **ação** a_t que possua o maior ***Q-value*** para o **estado** atual
    ou
    caso todas as **ações** sejam zero, seleciono alguma aleatoriamente
  Executo **ação** a_t
  Verifico **recompensa** $r(s_t, a_t)$ ganha pela **ação**
  Atualizo entrada na ***Q-table*** conforme:
  $Q_{t+1}(s,a) \leftarrow (1-\alpha) \times Q_t(s_t, a_t) + \alpha r_{t+1}(s_t, a_t)$
    if *(*$Q_{t+1}(s,a) < \delta$*)* **then** $Q_{t+1}(s,a) \leftarrow 0$
  e atualizo **estado** atual $s_t = s_{t+1}$
until *durante período de funcionamento* ;

Algoritmo 1: *Q-learning* com estratégia ε-*greedy*.

4.3. Modelagem RL-Ciclo

De forma intuitiva, a primeira maneira de modelagem que podemos seguir é o estado representar uma sequência de canais. Porém, se modelarmos desta forma, o número de combinações pode ser grande em decorrência do tamanho estabelecido da sequência e o número de canais a se utilizar. Por exemplo, para uma sequência de tamanho a com b canais disponíveis, teríamos b^a possibilidades, o que pode tornar a convergência do algoritmo lenta.

Assim, modelamos cada estado como um canal, e não como uma sequência de canais. Por exemplo: para uma sequência de tamanho 3 com 4 canais disponíveis, como um possível funcionamento para o nó SU de controle, temos:

1. Nó está no canal 1, olha sua *Q-table*, passa para o canal 2, TX no canal 2;
2. Nó está no canal 2, olha sua *Q-table*, passa para o canal 3, TX no canal 3;
3. Nó está no canal 3, olha sua *Q-table*, passa para o canal 4, TX no canal 4.

Assim, sua sequência de transmissão foi: 2, 3 e 4. Após a realização de toda a sequência é que o nó irá contabilizar o quão proveitoso foi a transmissão nestes canais; porém a recompensa não é devida a sequência escolhida, e sim pela decisão de cada canal, pois a atualização se dá no par estado-ação. Por exemplo, para a escolha do primeiro canal da sequência do exemplo anterior: estava no canal 1, passou para o 2, TX no canal 2 e recebeu 3 *acks* por esta ação. Para o par estado-ação (canal 1 e TX no canal 2), a entrada na *Q-table* será atualizada ao equivalente de 3 *acks*. Desta forma, a sequência é decidida a cada passo pela consulta à *Q-table*. Ao fim, contabiliza-se o quão boa foi esta escolha. Assim, diminuímos o número de sequências testadas e a convergência tende a ser mais rápida. Se o nó escolhesse uma sequência de início e contabilizasse as recompensas devido esta sequência, ele teria que varrer uma quantidade grande de combinações. Como o último canal escolhido equivale ao próximo estado em que o nó estará, nesta modelagem obtemos um ciclo de chaveamento de canais, e não propriamente uma sequência.

4.4. Modelagem RL-Seq

Na modelagem RL-Seq, o estado equivale ao ordenamento na sequência, e a ação equivale a qual canal utilizar. Assim, a primeira linha da *Q-table* refere-se a primeira posição da sequência, a segunda linha indica a segunda posição da sequência etc. As colunas representam os canais que podem ser usados. Em qualquer linha (ordenamento) pode-se selecionar qualquer canal. No funcionamento recorre-se sempre inicialmente a primeira linha, e escolhe-se um canal que será o primeiro canal da sequência de transmissão. Depois, passa-se obrigatoriamente a segunda linha, novamente elege-se um canal que representará o canal da segunda posição da sequência, e assim por diante, até que toda a sequência seja preenchida. Quando atingimos a última linha da tabela retornamos a primeira.

Em ambas as modelagens, quando o nó SU de controle seleciona um canal para transmitir e, no instante de enviar sua mensagem verifica que existe algum PU em atividade neste canal, o nó perde sua chance de transmitir e marca o respectivo *Q-value* com zero, independente do valor prévio, isto é para refletir uma escolha que não foi adequada.

Na próxima seção, apresentaremos os cenários utilizados para testar os dois mecanismos propostos na escolha da ordem de chaveamento de canais.

5. Ambiente de Simulação

Com a finalidade de avaliar o método de funcionamento proposto, aliado aos mecanismos de RL-Ciclo e RL-Seq, desenvolvemos um simulador que usa os recursos básicos do NS-2, que é um simulador a eventos discretos. A proposta foi avaliada em dois cenários, um conforme a Figura 2 e outro onde a disposição dos nós PUs foi aleatória.

Para o cenário da Figura 2 temos 3 nós PUs, 4 nós SUs e 1 nó SU de controle. Neste cenário, a influência dos nós PUs deve-se a questões temporais, visto que seus posicionamentos não se alteram. Nele o nó SU de controle está posicionado dentro do alcance de comunicação dos outros nós SUs, e todos os nós SUs são interferidos por algum PU. Assim, neste cenário, pode ocorrer o caso em que o nó SU de controle envia alguma mensagem de controle, que é recebida por algum SU, porém este nó SU receptor pode

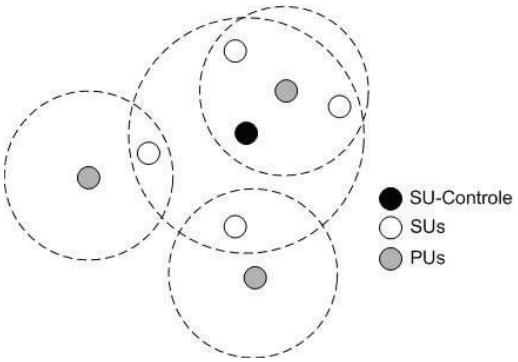


Figura 2. Cenário de simulação.

estar impossibilitado de responder com um *ack* naquele canal. Este é um caso de falso negativo para o nó SU de controle, enquanto que para o nó SU comum, o recebimento é visto como recompensa. Consideramos que as mensagens de controle podem ser recebidas mesmo nesta situação devido a questões de captura do meio. A seguir apresentaremos os resultados obtidos nas simulações.

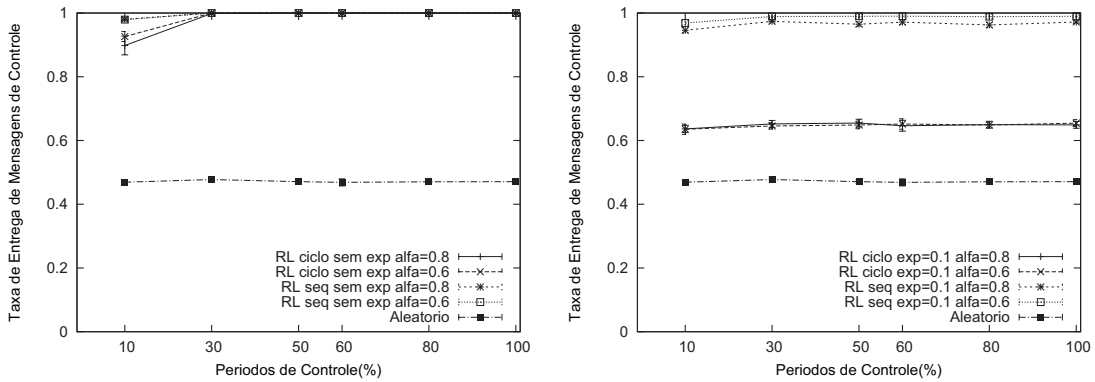
5.1. Resultados da Simulação

Todos os resultados são médias de 30 experimentos e os gráficos possuem intervalo de confiança de 95%. Cada experimento possui 1000 períodos de controle. O eixo x indica a porcentagem de períodos de controle até o instante da coleta de informação. A taxa de entrega contabiliza quantos nós receberam alguma mensagem no período de controle, e cada valor no gráfico representa a média da taxa de entrega entre o ponto atual e o ponto anterior, ou seja, os valores de taxa de entrega não foram acumulados.

Realizamos quatro experimentos de teste, a saber: no primeiro a atividade dos nós PUs foi fixa, ou seja, uma vez definido os canais dos nós PUs eles não se alteraram durante o período de simulação. No segundo, a atividade do nós PUs seguiu uma distribuição exponencial, com um parâmetro λ_1 que indicou o tempo médio que o nó pôde permanecer em atividade e um λ_2 que foi relativo ao tempo médio de desligamento. No terceiro caso, quando a simulação atingiu 30% dos períodos de controle, a configuração de canais dos nós PUs modificou-se aleatoriamente. Desta forma, pretendíamos verificar o tempo de recuperação dos mecanismos RL. Em um quarto experimento, alteramos a quantidade de PUs e seus posicionamentos foram aleatórios, justamente para testarmos condições em que a comunicação poderia ser impossibilitada devido a atividade de PUs. Comparamos o uso de RL-Ciclo, RL-Seq e o caso aleatório. Nas simulações não contabilizamos perdas de mensagens devido a colisões e problemas no meio de comunicação.

A Figura 3 apresenta os resultados com uma configuração onde os nós possuíam uma sequência de tamanho 3 e podiam utilizar 3 canais diferentes. Nesta configuração cada PU iniciou com um canal diferente dos outros PUs e possuiu o comportamento de estar em atividade ou não no canal.

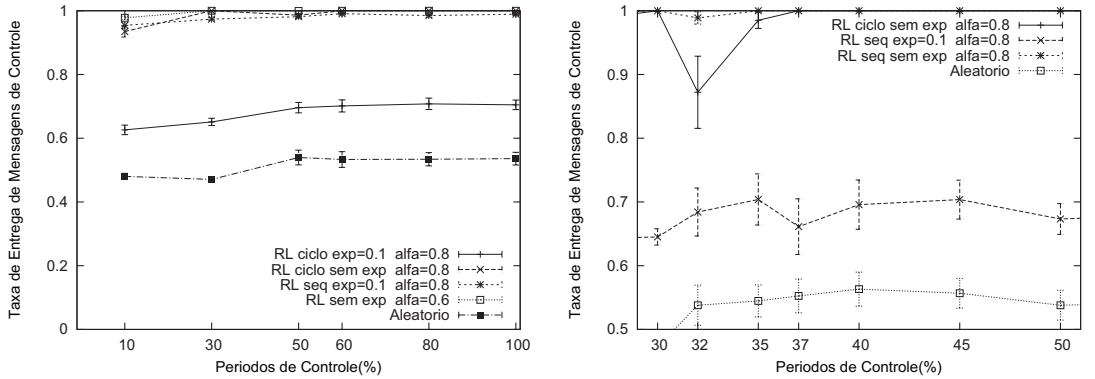
Simulamos os mecanismos com valores de α sendo 0.6 ou 0.8, e examinamos o caso do nó explorar com 10% de probabilidade. O comportamento dos nós PUs, entre estar ativo ou não, seguiu uma variável aleatória exponencial. Neste teste os nós não trocavam de canal, apenas suspendiam sua atividade no canal que estavam utilizando e



(a) Sem exploração. (b) Com exploração.

Figura 3. Nós PUs com atividade *On-Off*.

depois retornavam a reutilizá-lo. Usamos $\lambda_2 > \lambda_1$ assim o tempo médio que o nó permaneceu em atividade foi maior que o tempo médio que o nó esteve desligado. Verificamos uma convergência rápida nos dois mecanismos RLs propostos. Com 30% dos períodos de controle os nós já convergiram para a sequência de chaveamento de canais. No caso com exploração, Figura 3 (b), em ambos modelos a taxa de entrega decresceu, porém no caso do RL-Ciclo a perda foi bem mais significativa. Os mecanismos, ao perceberem um canal livre, que antes estava em uso, passaram a utilizá-lo, e quando o nó PU retornou a sua atividade, este canal teve de ser liberado. Logo, os nós alternaram neste comportamento, por isso a taxa de entrega diminuiu. O RL-Ciclo é o mais prejudicado, pois ao explorar ele passa a tentar não só um novo canal mas um novo ciclo e isto dificulta sua recuperação. Também foram realizados testes alterando a porcentagem de exploração em 20% e, nestes casos, o comportamento dos gráficos foi similar ao caso de 10%. Embora não apresentado, os gráficos com atividade de PUs fixa obtiveram comportamento semelhante.



(a) Período geral de simulação. (b) Período de recuperação.

Figura 4. PUs trocam de canais em 30% da simulação.

A Figura 4 apresenta o caso em que, decorridos 30% dos períodos de controle, os nós PUs trocam aleatoriamente de canais. Neste período, os nós já possuem valores na *Q-table* que indicam os saltos de canais a seguir. Porém, com a mudança de canais pelos

PUs, pode-se tornar necessária a busca por uma nova sequência adequada. No gráfico da Figura 4 (a), temos a visão geral do período de simulação. Na Figura 4 (b), analisamos apenas um trecho do período com a finalidade de mostrar o quanto em média os mecanismos levaram para se recuperar em número de períodos de controle. Podemos notar que o RL-Seq recupera-se rápido, cerca de 5 períodos de controle, enquanto o RL-Ciclo demora um pouco mais, cerca de 7 períodos. No caso da exploração o mais prejudicado foi o RL-Ciclo, cuja razão já foi apresentada nas explicações dos resultados da Figura 3(b).

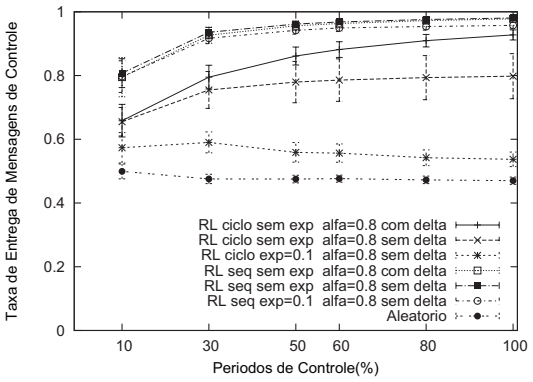
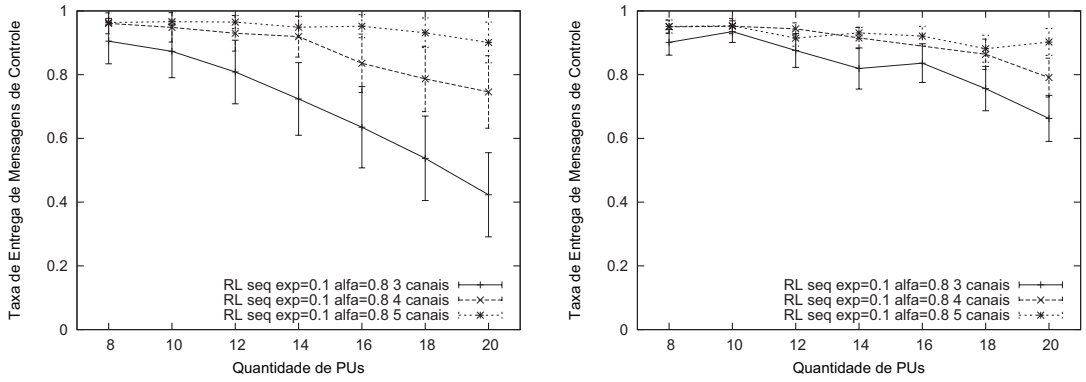


Figura 5. Tempo de convergência medido em períodos de controle.

Para o gráfico da Figura 5 alteramos o número de períodos de controle de 1000 para 100, assim o gráfico equivale aos instantes de 0 até 10% dos gráficos anteriores. Testamos também a influência do valor de δ na convergência dos métodos. Na legenda, quando indicamos o método "com delta", nos referimos ao caso onde se o valor estava abaixo de δ a entrada da *Q-table* era atualizada para zero. Enquanto que nas legendas "sem delta", o valor decresceu normalmente, a partir do funcionamento da equação de atualização. Pôde-se perceber que o modelo mais influenciado pelo uso do δ foi o RL-Ciclo, pois desta forma um ciclo escolhido anteriormente, que não era um dos melhores, foi abandonado mais rapidamente.



(a) Atividade de PUs fixa. (b) Atividade de PUs *On-Off*.

Figura 6. Taxa de entrega com aumento do nós PUs.

A Figura 6 apresenta os resultados obtidos no cenário onde o posicionamento dos nós PUs foi aleatório. Este cenário possuiu uma área quadrada de 150 metros de

lado, com cada nó tendo alcance de transmissão de 45 metros. Neste teste aumentamos a quantidade de experimentos de 30 para 60 rodadas. No cálculo da taxa de entrega final foi descontado um período inicial de 40% de períodos de controle. No eixo x alteramos a quantidade de PUs por teste; assim, pode-se perceber que conforme aumentamos a quantidade de PUs a taxa de entrega decai nos dois gráficos, justamente porque existem configurações em que o nó SU de controle fica impossibilitado de transmitir mensagens de controle. Conforme aumentamos a quantidade de canais, cresce a possibilidade do nó SU de controle conseguir algum canal disponível para transmitir aos nós SUs comuns, daí uma taxa de entrega maior. Na Figura 6 (b), a taxa de entrega foi maior pois as oportunidades de algum canal ficar livre devido a suspensão da atividade de algum nó PU foi maior. A seguir, apresentaremos as conclusões e trabalhos futuros.

6. Conclusões

Os rádios cognitivos são uma solução ao compartilhamento de espectro de frequências entre usuários não licenciados. Diversas são as áreas de pesquisa e dificuldades tratadas neste tema e, em muitas abordagens, considera-se o uso de um canal de comunicação exclusivo entre os nós não licenciados, chamado canal de controle. Pelo canal de controle os nós secundários podem se comunicar e dar prosseguimento ao bom funcionamento da rede. O que acontece é que devido a diversos fatores nem sempre existe um canal livre que sirva para todos os nós secundários se comunicarem. Uma abordagem da literatura, chamada *Channel Hopping*, propõe para uma rede *Ad Hoc* que os nós trabalhem com a possibilidade de saltar de canal em canal para assim se encontrarem com outros nós e, desta forma, trocarem mensagens. Esta abordagem pode ser utilizada em rádios Cognitivos para a transmissão de mensagens de controle, porém exite a dificuldade de definir uma ordem de saltos em que mais nós possam trocar mensagens de controle sem causar interferência nos nós licenciados.

Neste trabalho, discutimos o uso de uma ferramenta computacional chamada *Reinforcement Learning* (RL) em conjunto a solução de *Channel Hopping* adaptada para rádios cognitivos, onde a tarefa de encontrar a sequência de saltos foi atribuída ao RL. Com esta finalidade, modelamos dois tipos de mecanismos chamados, RL-Ciclo e RL-Seq. Pelos resultados obtidos pode-se constatar que com o emprego dos mecanismos de RL propostos, em principal o RL-Seq, os nós conseguiram descobrir sequências de saltos de canais que forneceram um desempenho satisfatório para a taxa de entrega de mensagens de controle, mesmo em ambientes com maior atividade de nós licenciados.

Como trabalhos futuros pretende-se avaliar o impacto do número de usuários primários e secundários com a relação a quantidade de canais da sequência. Pode-se também expandir o funcionamento com mais de um nó de controle e acrescentar a troca de mensagens entre entre estes nós.

Referências

Akyildiz, I. F., Lee, W.-Y., Vuran, M. C., and Mohanty, S. (2006). Next generation/dynamic spectrum access/cognitive radio wireless networks: a survey. *Computer Networks: The Int. J. of Comp. and Telecom. Networking*, 50:2127–2159.

Berge, C. (1982). *The Theory of Graphs and its Applications*. Greenwood Press Reprint; New edition edition.

Chen, T., Zhang, H., Maggio, G. M., and Chlamtac, I. (2007). CogMesh: A Cluster-based Cognitive Radio Network. In *IEEE)*.

Galindo-Serrano, A., Giupponi, L., Blasco, P., and Dohler, M. (2010). Learning from Experts in Cognitive Radio Networks: The Docitive Paradigm. In *5th International Conference on Cognitive Radio Oriented Wireless Networks and Communications, (CROWNCOM 2010), Cannes (France)*.

Haykin, S. (2005). Cognitive radio: brain-empowered wireless communications. *IEEE J. Select. Areas Commun*, 23:201–220.

Kok, J. R. and Vlassis, N. (2006). Collaborative multiagent reinforcement learning by payoff propagation. *J. Mach. Learn. Res.*, 7:1789–1828.

Mitola, J. (1999). *Cognitive radio: Model-based competence for software radios*. PhD thesis, Dept. of Teleinformatics, KTH.

PalChaudhuri, S., Kumar, R., and Saha, A. K. (2003). A MAC protocol for Multi frequency Physical layer. Technical report, Rice University, Houston ,TX.

Ren Y., D. P. and P., K. (2010). Analysis and Implementation of Reinforcement Learning on a GNU Radio Cognitive Radio Platform]. In *5th International Conference on Cognitive Radio Oriented Wireless Networks and Communications, (CROWNCOM 2010), Cannes (France)*.

So, J. and Vaidya, N. (2004). Multi-channel mac for ad hoc networks: Handling multi-channel hidden terminals using a single transceiver. *MobiHoc 2004, Tokyo, Japan.*, 50:222–233.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. A Bradford Book The MIT Press Cambridge, Massachusetts London, England.

Tang, C. and Garcia-Luna-Aceves, J. (1999). Hop-reservation multiple access (hrma) for ad hoc networks. *IEEE INFOCOM 1999, New York, USA*.

Tao Chen, H. Z. and Katz, M. D. (2009). Cloud Networking Formation in CogMesh Environment. Technical report, Zhejiang University, China. VTT, Finland.

Wiering, M. A. and van Hasselt, H. Ensemble Algorithms in Reinforcement Learning. Technical report, University of Groningen and Utrecht University, Netherlands.

Yau, K.-L. A., Komisarczuk, P., and Teal, P. D. (2009). A Context-aware and Intelligent Dynamic Channel Selection Scheme for Cognitive Radio Networks. In *4th International Conference on CROWNCOM 2009*.

Yau, K.-L. A., Komisarczuk, P., and Teal, P. D. (2010a). Applications of Reinforcement Learning to Cognitive Radio Networks . In *Communications Workshops (ICC), 2010 IEEE International Conference*.

Yau, K.-L. A., Komisarczuk, P., and Teal, P. D. (2010b). Enhancing Network Performance in Distributed Cognitive Radio Networks using Single-Agent and Multi-Agent Reinforcement Learning. In *55th Annual IEEE Conference on Local Computer Networks, (LCN 2010), Denver,Colorado.*