Q-Learning for Cognitive Radios

Neil Hosey
Dept. of Computer Science
NUI Maynooth
Maynooth, Co. Kildare
Email: nhosey@cs.nuim.ie

Susan Bergin
Dept. of Computer Science,
NUI Maynooth,
Maynooth, Co. Kildare

Irene Macaluso
CTVR,
Trinity College,
Dublin 2

Diarmuid O'Donohue
Dept. of Computer Science,
NUI Maynooth,
Maynooth, Co. Kildare

Abstract—**Machine Learning approaches such as Reinforcement Learning (RL) can be used to solve problems such as spectrum sensing and channel allocation in the cognitive radio domain. These approaches have been applied to other similiar domains such as in mobile telephone networks and have shown much greater performance than the static channel allocation schemes used.**

The objective of this research is to use an RL technique known as Q-Learning to provide a possible solution for allocating channels in a wireless network containing independent cognitive nodes. Q-Learning is an attractive algorithm for such a problem because of the low computational demands per iteration. Many of the current proposed techniques suggest using a negotiation policy between two nodes to decide on which channel each may use, however a considerable problem with this is the overhead involved in the negotiation involved between the nodes. This paper suggests an approach where each node acts as an individual independant node, with virtually no collaboration with the other nodes.

Results have shown that using such a technique gives fast convergence on an optimal solution when correct rates are chosen. It has also shown that the algorithm is very scalable, in that as the network grows, the state-action space does not grow sufficiently to cause major memory or computational demands.

I. Introduction

Research in the area of cognitive radio (CR) has broadened significantly in the past number of years since it was first presented by Mitola in 1999 [1]. It is now recognised as being an essencial replacement for the current regulation of the electromagnetic spectrum where vast bands of usable spectra is being underutilised. One such example of this was in the USA where the Spectrum Policy Task Force [2] had found that for a particular period on a police broadcasting channel, the typical channel occupancy was less than 15% , while the peak usage was nearly 85%. This has led to much research into the area of opportunistically accessing underutilised spectrum where no primary user is currently active.

The goal of this research is to not focus on primary and secondary users in a frequency domain, but rather to allow each cognitive node to learn by its own mistakes. In this case, the agent considers the environment to include all other cognitive nodes, and any other transmitters working on the same channels as being part of the environment. This will ensure an even distribution of channels not only for the cognitive nodes, but also for any other type of wireless communication device. It is hoped that future work will look at primary and secondary users working in the same environment using a Q-Learning approach. Much of the research in this area has focused on cooperative sensing where nodes within a cognitive radio network share information about the environment. There are several levels on which this can happen [3], which at the very least have a need for a control channel to pass information between nodes. This alone can lead to massive overheads on a network as the number of nodes and the amount of data shared increases. The proposed solution to this is individual sensing where each node is a single cognitive entity having the ability to acquire information about the environment or network without the help of other nodes in its vicinity.

This is achieved using a reinforcement learning algorithm known as Q-Learning whereby the agent goes through a phase of *learning* before it can converge on an optimal solution for channel allocation. In this learning phase, the node makes decisions on what channels to select pseudo-randomly, the outcome of taking these actions will weigh strongly on what decisions are made later on. Once a node has finished learning, it can then make decisions on what it has learned. The ability for a node to be able to preempt whether a channel is going to be in use before accessing it allows it to optimise bandwidth usage for itself and any other nodes that may be accessing the same channel.

The rest of this paper is organised as follows. First, an overview of reinforcement learning is presented, along with an in-depth look at Q-Learning and how it is applicable to this domain. Details on how Q-Learning has been applied in channel selection in cognitive networks are provided. Results and outcomes of simulations performed are then given, followed by current and future work. Finally, conclusions from this work and possible future work in this area are provided.

II. Reinforcement Learning

A. Application of Reinforcement Learning

Reinforcement Learning is a machine learning technique whereby an agent interacts with an environment in the hope of achieving a goal. This interaction occurs on a continual basis with the hope of the agent being able to learn to function in an optimal fashion within that environment. The way in which the agent interacts with the environment is through a series of actions that can be performed. These actions can have

positive or negative outcomes which can, over time, be used to determine how best to work in the current environment. At each point in time, an agent can be in a particular state, with the ability to choose an action based on what it has learned in previous iterations.

The overall goal is to find an optimal policy that maps each state to an action an agent should take in those states [5]. Figure 1 shows how the agent interacts with the environment and uses this to determine its next state.

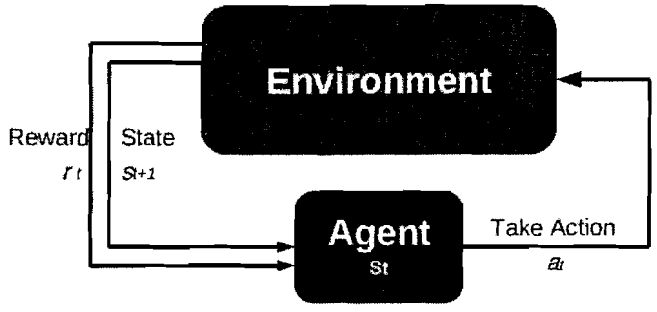

Fig. 1. Agent-Environment Relationship

To represent this formally, we assume the agent receives the next state from the environment as shown in figure 1 at each iteration of the algorithm, $s_t \in S$, where S is the set of possible states and $t = 0, 1, 2, ...$ for each individual discrete timestep or iteration. On receiving this state, an action is chosen, $a_t \in A(s_t)$ where $A(s_t)$ is the set of possible actions that can be taken in state s_t. On taking this action, the agent observes the result, and receives a reward r_{t+1} where $r_{t+1} \in \Re$. After taking this action, the agent has now moved into a new state, s_{t+1}. At each iteration of the algorithm, a policy is created that maps the action taken, a_t to the state s_t and this policy is denoted by $\pi_t(s_t, a_t)$. The way in which this mapping occurs is dependent on each RL algorithm and is usually based on one of a number of action selection strategies which will be described in the next section.

B. Q Learning Algorithm

Q-Learning is an RL off-policy temporal-difference learning algorithm introduced by Watkins in 1989. The algorithm works by learning an action-value function that gives an expected utility of taking an action in a particular state and following that policy thereafter [6]. The environment in which the agents exist can be modelled as a Markov Decision Process. The agent-environment shown in Figure 1 consists of a number of steps:

- Agent examines state $x_t \in X$
- Action $a_t \in A$ is taken based on x_t
- A transition occurs as a result of action a_t being taken, and a new state $x_t + 1$ is taken into account. A reward is generated based on this transition, r_t.
- The reward, r_t, that is returned is then stored or learned for that state action pair, and the above process is repeated.

The goal of repeating this process is for the agent to find an optimal policy $\pi^* \in \pi$ for each state $x_t \in X$ in a recursive manner. The fact that the Q-Learning algorithm can converge on π^* without having any prior knowledge of the environment makes it very suitable for cognitive radio channel selection because of the unpredicability of other nodes and the electromagnetic spectrum.

The algorithm can be described as a simple value iteration update as shown below:

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(s_t, a_t) \times [r_t + \gamma \times max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \quad (1)$$

where $\alpha(s_t, a_t)$ is the learning rate where $0 < \alpha \leq 1$ and which represents to what extent newly acquired information will be taken into account. A learning rate of 1 will mean that only the most recent rewards will be taken into account whereas a learning rate of 0 will mean the agent will learn nothing, and any current reward will be discarded. The discount factor, γ, $0 < \gamma \leq 1$, decides how important future rewards are for the agent. Another thing that makes the Q-Learning algorithm suitable for this type of problem is because it has been shown [7] that Q-Learning will converge with a probability of 1 as long as each state action pair is visited infinately as the learning rate approaches zero. The way in which the Q function in Equation 1 is implemented can be shown in the pseudocode:

```
For each episode, while s is not terminal
  Sense environment and state s
  For each iteration:
    Choose a from s using certain policy
    Take action a, observe output, r, s_{t+1}
    Update Q value for state-action pair (eqn. 1)
    s ← s_{t+1}
  Loop
Loop
```

The electromagnetic spectrum environment that the agent is working in is very unpredictable, making it suitable to use an off-policy RL algorithm such as Q-Learning so as to allow a period of random exploration before following the target policy of the agent. The policy used in selecting which action to take is dependent on the type of policy used. The simplest example is to select the action with the greatest reward for that state, although this may not always lead to an optimal solution as it would lead to a totally greedy policy that would not explore parts of the state space that would not appear to be advantageous but could lead to an optimal solution in the future.

ε-Greedy is an example of a strategy that overcomes this problem. It does so by only selecting the best action $1-\varepsilon$ of the time and another action is chosen randomly selected for the remainder of the time, ε. The value of ε is in the range $0 < \varepsilon \leq 1$. The higher the value, the more random exploration

will occur. A similiar strategy known as ε-decreasing strategy is what is used in this experiment. The main difference between this and ε-Greedy is that ε decreases over a period of time so that the agent goes through a period of random exploration or learning before becoming totally exploitative.

III. Algorithm Implementation

To explain how Q-Learning was used for this problem, an examination of how each state, action, and reward is structured is provided. First, the available spectrum was broken up into a number of channels which could be used for communication. The number of channels available, C, that was used in this experiment was 4, but can change depending on available spectrum. This amount of channels was chosen to provide a simple case, although any number of channels could be chosen as this implementation does not suffer from scalability problems.

The state was defined as a 2-dimensional structure

$$s_t(tr_t, if_t) \tag{2}$$

where the 1st dimension, tr_t, represents the number of channels that a node is currently transmitting on at that time, $0 < tr_t \leq C$ and, the 2nd dimension, if_t, represents the number of channels that an agent attempted to transmit on but were in use at that time, $0 \leq if_t \leq tr_t$. The number of interfering channels is based on the number of channels in use, either by other nodes or through interference that are within range of the node. Through sensing the environment, a binary 2 dimensional vector, $if_t(c)$ is populated as per equation 2.

$$if_t(c_t) = \begin{cases} 1 & \text{if channel } c \text{ at time } t \text{ is in use.} \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

where $c_t = 1, 2, ..., C$. This vector is scanned at each iteration upon taking an action to move to the next state. As the network may be spread over a large area, and would not be a fully-connected network as a result of wireless restrictions on distance, a connectivity vector, V is used to store what cognitive nodes are within range. This is only needed for simulations as a real world implementation would only be able to sense nodes within the wireless transceivers maximum distance. There are 3 possible actions that can be taken by an agent at a particular timestep:

- *I* - do nothing.
- *II*- acquire a channel.
- *III* - drop a channel.

Action I will not acquire or drop any channels, action II will acquire a channel for transmission, and action III will drop a channel that it already has in use. The channel that is selected for drop or acquire is completely random in this implementation, but future work may allow an agent to learn which channels are good and which are bad.

There is an immediate reward or punishment received for taking action a_t while in state s_t. Although there are many posible ways in which to calculate a reward in such an instance, it is important to ensure that the agent doesn't act in a greedy manner, by acquiring as many channels as possible leaving other nodes in the network, or surrounding networks starved of bandwidth. The proposed function shown in equation 4 ensures that the number of active channels is proportionately greater than the number of intefering channels. The weighted interfering channels forces a punishment to any agent that acquires a hnigh proportion of channels in an environment where channel usage is high.

$$r(s_t, a_t) = tr_t - (tr_t \times if_t) - 1 \tag{4}$$

This function ensures the channels are evenly distributed between all the nodes of the network, and also for other devices operating in the same environment. A simple example would be where an agent is transmitting on 2 channels and had also attempted to transmit and failed on another channel after action 1 or *acquire channel* was chosen, the state the agent would be in is $s_t(2, 1)$. So the reward calculated based on equation 4 is

$$r(s(2, 1), 1) = 2 - (2 \times 1) - 1 = -1 \tag{5}$$

In this case, it is a small punishment that the agent receives for this state-action pair. It shows that because the agent is transmitting on 2 channels, but interfering on one of them channels, the reward is negative.

As mentioned in the previous section, the policy by which the actions are chosen for Q-Learning is based on a particular strategy, and in this case a hybrid on ε-Greedy known as the ε-Decreasing strategy. The idea of this is, as explained in the previous section is to allow the agent to go through a period of random learning or exploration before exploiting what it has learned.

Many of the simulations that we have carried out have used a value for ε of 0.8. An example of how action selection occurs is as follows. In the first iteration of the algorithm, there is a 80% probability that a random action will be chosen (exploration) and a 20% probability that the best (or max Q value) action will be chosen (exploitation). As the algorithm progresses, this value decreases to allow the agent to slowly transform into an agent that selects the best action based on what it has learned rather than randomly hopping through the state space, thus exploiting the information it has gathered. As ε approaches zero, the agent should eventually converge on a stable, non-greedy state that uses an appropriate amount of channels without causing interference to other nodes in the environment. It then converges on a fixed state where the agent is transfering over a fixed number of channels until the environment changes enough to warrent a re-learning. These changes could be due to other nodes leaving or joining the network or other outside interference.

As this value reduces overtime and effectively controls whether the agent is in a *learning mode* or not, it should be possible to adjust this value during the running of the algorithm. There are numerous reasons why we would want

to do this but the most important is that the electromagnetic spectrum is an ever changing environment, and as long as it is slow changing, the agent can re-enter the learning phase when the environment has changed enough as to make what it has learned redundent. This forces the agent to re-learn so that it can effectively operate in the altered environment again. This can be an ongoing process where the agent goes in and out of a learning phase whenever some metric that measures environmental change reaches some threshold. The function used in decreasing ε is the same as the one used in decreasing α, which is explained below.

The convergence of a discrete algorithm to an optimal policy is vital for this algorithm, and Watkins and Dayan have proved that Q-Learning does converge [7] as long as a number of conditions hold. The learning rate, α_t, where $0 \leq \alpha \leq 1$ decreases at each iteration. We looked at 2 ways of decreasing α_t, the first being to simply decrease it by a fixed value each time as shown in Equation 6.

$$\alpha_t = \alpha_t - (\alpha_t \div EstNumI) \tag{6}$$

where $EstNumI$ is the estimated number of iterations needed for the algorithm to converge. Another method for decreasing α suggested by Watkins was to decrease it based on the number of times a particular state-action pair, $\alpha(s,a)$ is visited.

$$\alpha(s,a) = \frac{1}{n(s,a)} \tag{7}$$

where $n(s,a)$ is the number of times that state action pair has been visited. This *alpha* value will give 1, $\frac{1}{2}, \frac{1}{3}, ..$ at each visit to a particular state-action pair.

Eventually the algorithm needs to converge on a particular state which tells the agent the optimal amount of channels it can transmit on without causing interference. This is achieved by allowing the policy to continue choosing actions until action 0, *'do nothing'* is chosen for a fixed number of iterations, meaning that the algorithm has reached a stage where it will stay in the same state forever.

Finally, the greatest advantage of using this particular implementation is the small amount of memory and computational requirements needed. The state-action space would be considered substantially smaller than many other problems that use Q-Learning. In a 4 channel network there is a maximum of 16 possible states, with 3 possible actions making a total state-action memory space of 48. Assuming the use of floating point numbers, the memory space used storing these Q values is only 192 bytes.

IV. Results and Findings

A. Simulation

This work was based on a simple 4-node network as shown in figure 2 for simulation purposes.

We simulated an environment where there was 4 channels available for transmission, although the number of channels could be altered for each individual node to simulate a real world environment where there may be other radio operating in the same environment. As we wanted to model this as accurately as we could, we used nodes that could only transmit on 1 channel and in some cases 2 to ensure that the algorithm, for each agent or node would converge on an optimal solution each time in different scenarios. The way in which interference was represented in the simulated environment was using a 2 dimensional vector. An example of one is shown in the table below.

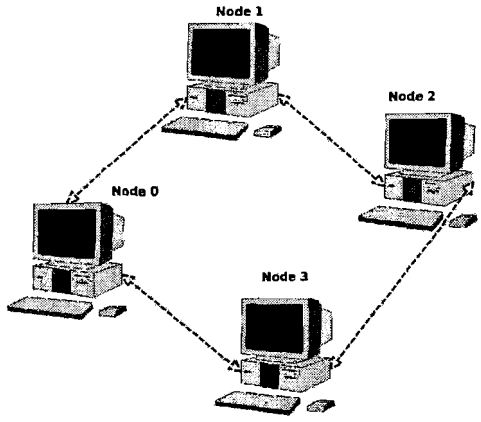

Fig. 2. 4-Node Network

	N0	N1	N3	N2
N0	0	1	1	0
N1	1	0	0	1
N3	1	0	0	1
N2	0	1	1	0

Table 1. Sample interference vector over all nodes where 0 represents if two nodes are not in interferable range and 1 if they are in range.

Each node, as mentioned above has an interference vector which holds details on which nodes are within range that could cause interference upon acquiring a channel. The connecting lines in Figure 2 represent 2 nodes being within interfering range. For example, node 0 is within interfering range of nodes 1 and 3. For the purpose of these simulations, this table was used for determining interferable nodes within the network. In a real world case, each node would need to determine which nodes are within range themselves. We implemented this in Java, as this was the first authors main language. Although this sufficed, future work will include implementing this on a number of SDR's, which would require a C implementation.

B. Results

The simulations carried out were mainly focused on different rates and different decreasing factors for ϵ, γ and λ. We also looked at how the overall interference throughout the network reduced as the agents neared convergence. Finally, we looked at using different forms of action selection strategies.

1) Experiments and final states: As discussed above, for each independent node, it will eventually settle on a state which would hopefully maximise spectrum usage without causing any interference with other nodes. Simple 4-node experiments have shown that this is the case. A sample of some of the experiments is shown in Table 2 and have shown good results. In each case, the output shows that the algorithm has converged on an optimal solution that uses the maximum amount of channels possible for each node without causing interference.

	Node0	Node1	Node2	Node3	Output
1	4	4	4	4	2,2,2,2
2	4	1	4	4	2,1,2,2
3	4	1	1	4	3,1,1,3
4	4	4	1	1	2,2,1,1
5	4	2	3	4	2,2,2,2
6	4	1	4	1	2,1,2,1

Table 2. Output of results of channel usage

The values shown in Table 2 for each experiment represent the number of channels available for transmission for each node. The output is the number of channels the algorithm converged to for each node respectively. In each of these cases, the number of channels is the maximum amount possible without causing interference with other nodes. The channels have also been divided equally without any communication or passing of information.

/subsubsectionϵ-reduction The speed at which the algorithm converges on a solution depends alot on how ϵ is reduced. The faster this value is reduced, the quicker the algorithm moves into the exploitation phase. Results have shown that the faster ϵ is reduced the less chance it has to explore the state-action space in full and thus usually results in the algorithm converging on a bad solution. Figure 3 shows sampled results of how the interference of a single node transitions during the course of the learning of the environment every 1000 iterations.

It can be seen that for a large amount of the iterations at the beginning of the algorithm, the node is interfering on all four channels available, but as it begins to transition into the exploitation phase, this number slowly digresses and eventually does not interfere on any channel. It has been explained that one of the rates at which ϵ decreases is based on the function:

$$\epsilon_t = \epsilon_t - (\epsilon_t \div EstNumI) \tag{8}$$

In the case above, a value of 100,000 has been set for $EstNumI$, meaning that at each iteration, it will be decreased by $\epsilon_t \div EstNumI$. If $EstNumI$ is decreased substantially, thus causing a faster decrease in ϵ, the algorithm will act in a much more erratic manner and will fail to converge on a fair solution. Figure 4 illustrates an example where $EstNumI$ is set to 15000. It shows how in comparison to Figure 3, the algorithm fails to reduce the number of channels it is interfering on, and although it converges much quicker than a higher $EstNumI$, each node will not acquire a fair amount of channels, with the one below acquiring all 4 channels, causing interference on the network with some nodes, whilst causing other nodes to not acquire any.

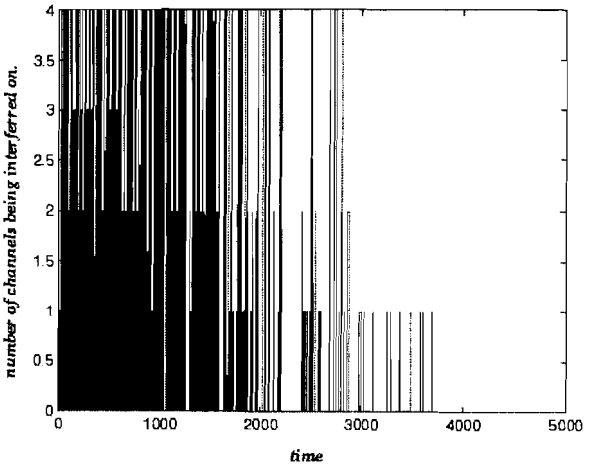

Fig. 3. Single Cognitive Node Interference Pattern over the course of learning environment in a 4 node 4 channel network.

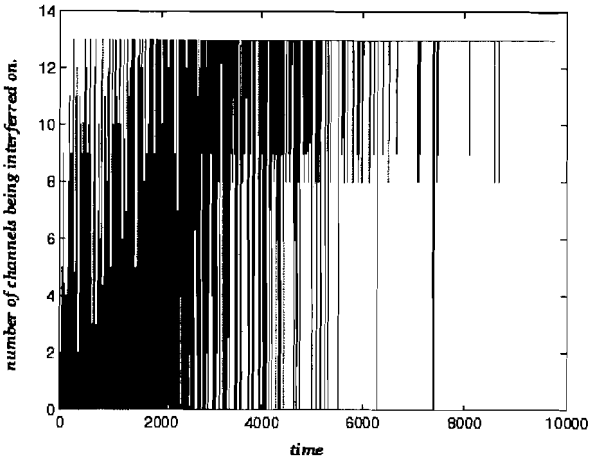

Fig. 4. Single Cognitive Node Interference Pattern where the algorithm does not converge on a good solution.

Therefore the selection of $EstNumI$, must be large enough to allow an agent to traverse the state-action space enough before reaching an exploitative phase.

2) ϵ-Decreasing vs. ϵ-Greedy: The action selection strategy which appeared to work best was ϵ-Decreasing. Many of the experiments carried out using ϵ-Greedy resulted in the algorithm taking much more iterations to converge. While in comparison to ϵ-Decreasing, which converged very quickly as long as robust values were chosen for the different rates. ϵ-Decreasing allowed for greater exploration at the beginning, and greater exploitation towards convergence. This suited the needs of the experiments as the goal was to find an optimal

solution for channel allocation.

3) γ-Selection: How much of the future rewards the algorithm takes into account during the exploration phase very much depends on the initial selection of the discount factor, γ. It has already been shown that Q-learning will converge with a probability of 1, but what what it converges to may not be *good* for what the algorithm hopes to achieve. It has been noted that the smaller γ is, the less the probability that the algorithm could converge on a good solution, and usually resulted in a number of the nodes acting in a greedy fashion while other nodes not being able to acquire any channels. A lower bound on the learning rate was discovered, and if it was set below this bound, these problems would occur. As long as γ is greater than the lower bound specified in Equation 8, the algorithm will strive to find a long term goal as opposed to only focusing on current goals.

$$\gamma \geq 0.5 \tag{9}$$

It was also noted that any value over this threshold made very little difference in the both the number of the iterations needed to converge and the solution that the algorithm converged to. What is different between λ and the other variables is that it is fixed. It does not decrease or increase throughout the course of the algorithm. This threshold is based on the fact that γ is a discount factor for future awards. As this approaches zero, the less the algorithm considers future rewards of importance. This makes each individual agent work in a greedy fashion and only consider current rewards.

4) α-Selection (initial): The learning rate, α as discussed earlier determines how much the algorithm takes into account what it learns at each iteration over what it has previously learned. In comparrison to other uses of Q-Learning, the importance of *alpha* is quite low. It has been noted, through multiple experiments, that as long as there is robust selection for ϵ and γ, it does not matter what initial value is selected for α as long as it follows the basic criteria of being decreased appropriately through time (iterations) and in the range $0 \leq \alpha \leq 1$.

V. Future Work

This research is still in the early stages, and current work is focusing on implementing such an algorithm on a number of Software Defined Radios (SDR) to develop a working example of how Q-Learning could be used in solving this problem. The Maynooth Adaptable Radio System (MARS) has been under development at NUI Maynooth's electronic engineering department since 2004 [8].

Current software demonstrations allow for transmission of images using the IRiS software architecture developed at the CTVR, Trinity College, Dublin and a large number of waveforms using a MARS demonstration application from a transmitter to a receiver. Implementing the Q-Learning algorithm on the MARS boards will give us a working example of a machine learning algorithm to this problem in a real world environment, although there are a number of challenges to overcome first. Currently the transmitter and receiver boards run off separate machines, and since cognitive radios need to be fully duplex for both scanning and receiving, there is a need to have both boards running on the same machine. Current work will focus on determining if the kernel will be able to recognise both boards running concurrently.

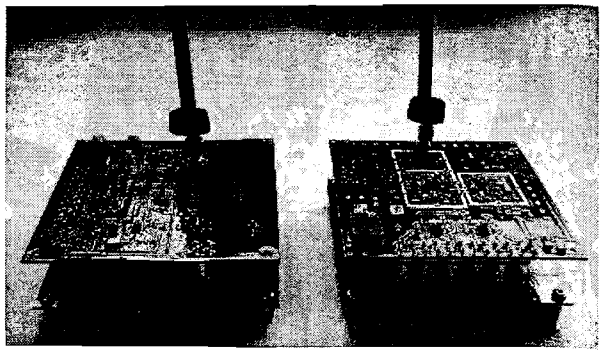

Fig. 5. MARS Receiver and Transceiver Boards

There are also a number of improvements to be made to the current algorithm. For example individual channels can be included so that agents will be able to differentiate between good channels and bad channels as opposed to whether a number of channels to use is good or not. This would vastly improve the performance of the algorithm in terms of avoiding interference.

Although this is a completely independent learning algorithm with no co-operation with other nodes, it may be worth exploring what advantages some limited communication may have between nodes. One such example could be passing information about *bad channels* between nodes based on some threshold of the Q-Values. It has been mentioned that there is no communication between nodes, but in the real-time implementation, there may need to be a small low bandwidth control channel for communicating which channel a node is going to transmit on to another node.

It may also be worth exploring how well a centralised approach would work using the Q-Learning algorithm by using a master-slave setup in a network. This would involve one fat node doing much of the computation and using a control channel to transmit channel usage information back and forth.

VI. Conclusion

In this paper, we have presented a simple RL technique for channel assignment in a network of independent cognitive nodes. This is achieved using a self-learning scheme based on a TD learning algorithm known as Q-Learning using a 2 dimensional state in an unknown environment. Q-Learnings suitability to this is has been shown as it can take in unknown situations and act upon them using its own experiences.

Simulations carried out on a 4 node network with 4 channels have shown good results in fair non-greedy channel assignment, so much so as to pursue implementing this

on a number of Software Defined Radios in a real time environment with the future goal being to use this as a benchmark to measure other machine learning algorithms abilities to perform this task.

The most significant advantage of this implementation is how small the memory, bandwidth and computational requirements are in comparison to many other cognitive radio channel assignment schemes.

VII. Acknowledgements

This work has been carried out with the support of Science Foundation Ireland (SFI) through the Centre for Telecommunication Value Chain Research (CTVR) and the Institute of Microelectronics and Wireless Systems at the National University of Ireland, Maynooth.

References

[1] J. Mitola and G. Q. Maguire, *Cognitive Radio:Making Software Radios More Personal*, IEEE Personal Communcations 1999

[2] FCC *Spectrum Policy Task Force Report*, ET Docket No. 02 - 135, 2002.

[3] S. M. Mishra, A. Sahai and R. W. Brodersen, *Cooperative Sensing among Cognitive Radios*, In Proc. of the IEEE International Conference on Communication(ICC), pp. 1658 - 1663, 2006.

[4] N. Lilith and K. Dogancy, *Dynamic Channel Allocation for Mobile Cellular Traffic using Reduced-State Reinforcement Learning*, WCNC 2004 pp. 2195-2200.

[5] L. Kaelbling, M. L. Littman and A. W. Moore, *Reinforcement Learning: A Survey*, Journal of Artificial Intelligence Research 4-237-285, 1996.

[6] C. J. Watkins, *Learning from Delayed Rewards*, Ph.D Thesis, Cambridge, 1989.

[7] C. J. Watkins, *Q Learning*, Machine Learning , Volume 8 pp279-292, 1992.

[8] R. Farrell *Software-Defined Radio Demonstrators: An Example and Future Trend*, Centre for Telecommunications Value Chain Research, Institute of Microelectronics and Wireless Systems, National University of Ireland Maynooth, Maynooth, Co. Kildare, Ireland