Mission-Critical Communications Planning Over Contested RF Spectrum with Deep Reinforcement Learning Aided Artificial Intelligence

*Milidu N. Jayaweera**

La Cueva High School, Albuquerque, NM

ABSTRACT

Mission-critical communications (MCC) refer to those that support operations involving high risk to human life and property. As radio frequency (RF) spectrum becomes highly contested, ensuring mission-success with MCC requires intelligent planning policies. This project develops a novel game-theoretic model for MCC and a Deep Q-Network (DQN) implemented Deep Reinforcement Learning (DRL) based Mission-Critical Communications Protocol (MCCP) to learn to complete a mission within given resource-constraints against an adversary. An example critical mission is defined as two radios exchanging messages within a given time-constraint over a two-way communication link in the presence of a jammer. Mission-planning requires radios to learn when and how to switch directions vs. channels in response to the behavior of the adversarial jammer as well as wireless channel anomalies. Through extensive-form sequential-game modeling, the problem was shown to be too complex to solve analytically and beyond traditional reinforcement-learning due to uncountable state-space. Results on an actual wireless network showed that the DQN-implemented DRL could achieve mission-success with as high as 0.9 probability. A new DRL algorithm called Deep Policy Hill Climbing was developed that outperformed the original DQN-DRL algorithm. Beyond MCC, this framework can be applied to a wide-variety of planning under uncertainty problems that arise, for instance, in disease control, refugee crises, disaster relief and resource-allocation in management.

KEYWORDS: Deep Policy Hill Climbing, Deep Q-Networks, Deep Reinforcement Learning, Game Theory, Mission-critical Communications, Mission-critical Communications Protocols, Nash Equilibrium, Sequential Games, Subgame Perfect Equilibrium

INTRODUCTION

From Isaac Asimov's *Bicentennial Man* to the *Terminator* franchise, developing *Artificial Intelligence* (AI) capable of learning to make decisions by itself has long been a human fascination. Formal scientific and engineering approaches to developing AI are commonly known as *machine learning* (ML). A seldom considered novel application of machine learning and AI is in *mission planning*.

Many modern military, humanitarian, and scientific missions critically depend on *over-the-air* (OTA) wireless communications between agents/nodes. A failure in the communications network can be catastrophic, costly, and even life-threatening. Figure 1 shows an example in

which a rocket launch is managed remotely by two locations. Such distributed control of critical missions is becoming increasingly desirable to reduce the chances of human error and to defeat deliberate sabotage by rogue actors.

This gives rise to the importance of mission-critical communications (MCC) planning since mission-success now crucially depends on the communications system. *Mission-critical communications* refers to reliable communications between nodes that will lead to mission-success in operations involving high risk to human life and property, especially when there are deliberate and adversarial *jamming signals* (1). Most existing work on MCC focuses on the reliability of communications infrastructure (2). As RF spectrum becomes highly contested, however, ensuring success of missions involving distributed agents as in Figure 1 requires such infrastructure to be supported by *mission-critical communications protocols* (MCCP) that implement intelligent planning policies. Both design of *mission-critical communications planning* techniques and their implementation as MCCPs are problems that have not yet received sufficient attention despite their growing importance.

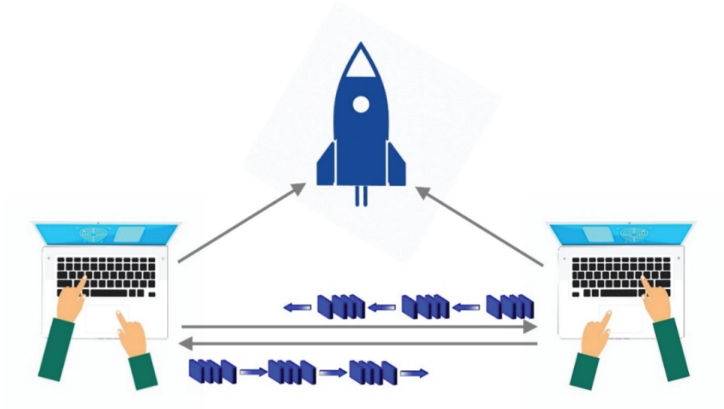


Figure 1. Mission-Critical Communications Planning for Distributed Control of a Rocket Launch.

Mission-critical communications planning is one example from the general problem of *planning-under-uncertainty* (3) encountered in different fields ranging from how to assign resources in a manufacturing plant to how to respond to an adversary in a war-like conflict. Mathematically, a broad class of planning problems can be modeled as a *Markov Decision Process* (MDP) (4). In an MDP, an agent takes *action* a_t in an environment characterized by a *state* s_t and receives a *reward* r_t. Agent's goal of mission-success can be modeled as maximizing an accumulated discounted-reward. Agent's actions effect the environment so that the next state is not only a function of the current state but also of the agent's selected action. Thus, the next state of the environment s_{t+1} is determined according to the transition probabilities $P(s_{t+1} \mid s_t, a_t)$ where $P(s_{t+1} \mid s_t, a_t)$ is the probability of state s_{t+1} given action a_t was executed in current state s_t.

If all state transition probabilities and rewards were known exactly, the so-called *Bellman optimality equations* characterize the optimal planning policy which can be solved using *Dynamic Programming* (4). When these are unknown, *reinforcement learning* (RL) is a popular machine learning paradigm for obtaining optimal planning policies (5).

The MCC planning, however, is more difficult than classic MDPs for three reasons: First, there are adversaries that attempt to hinder the mission by deliberately jamming signals. This turns the problem into a *stochastic game* (6). Second, state transition probabilities are not only unknown but also can be time-varying due to the nature of the wireless links (7). Third, the state of a *mission-critical communications game* may take values on an uncountable set rendering the traditional RL techniques not applicable.

The project objective is to develop a new approach to solve the MCC planning problem by leveraging the recent advances in *Deep Reinforcement Learning* (DRL). The project defines the MCC problem to be OTA sharing of mission-critical information residing at two distributed agents within a given time constraint. The MCC planning problem is to select at each decision epoch who should transmit information packets and on which channel, as shown in Figure 2.

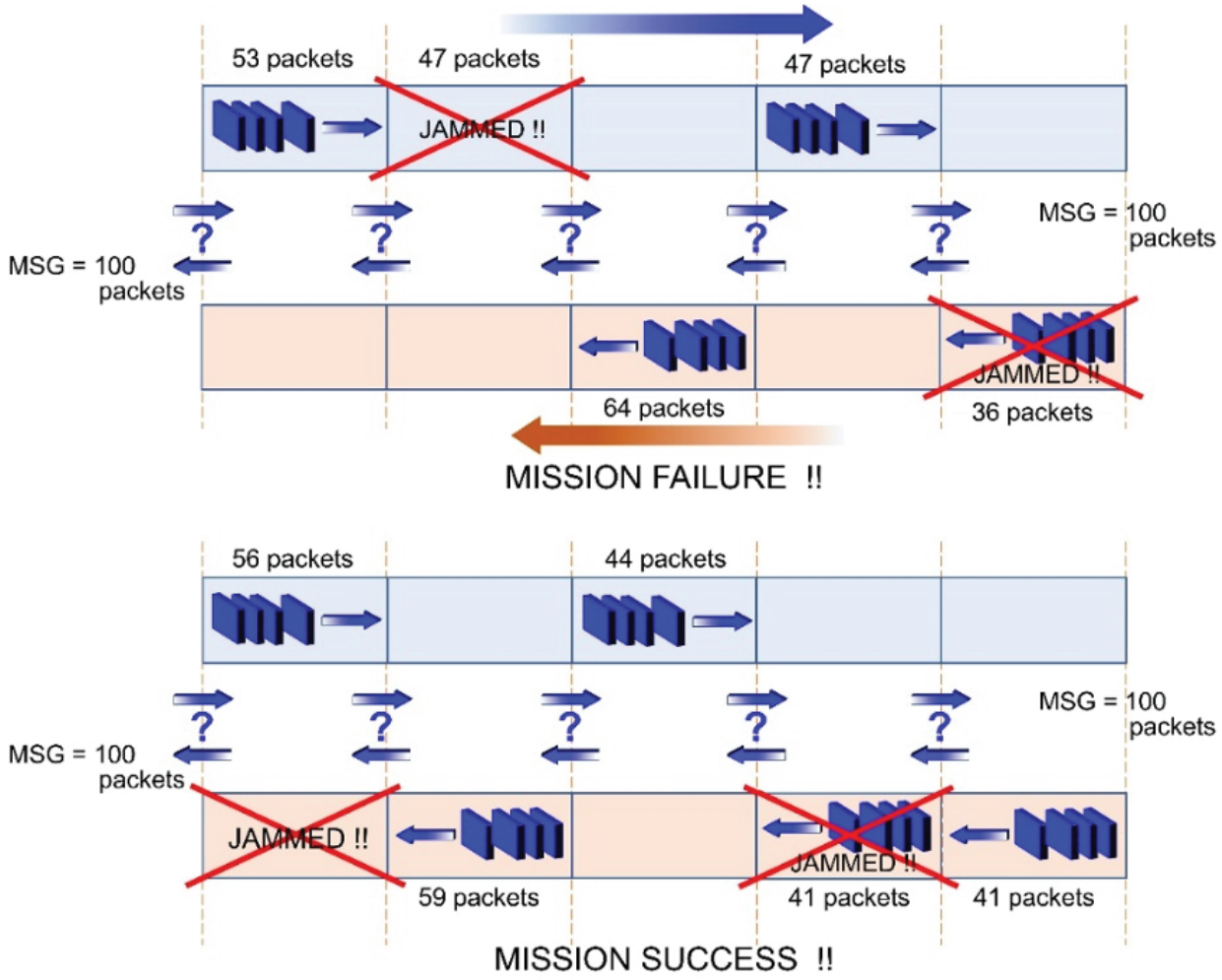


Figure 2. Dependence of Mission-Success and Failure on a Mission-Critical Communications System.

We develop a general game-theoretic model of this MCC planning problem that lends itself for the application of DRL. Then, we design and implement a *Deep Q-Network* (DQN) based DRL algorithm to learn effective planning policies that lead to mission-success with

high probability. The project also proposes a new DQN-based DRL algorithm, termed *Deep Policy Hill Climbing (DPHC)* which is shown to beat the original DQN-DRL algorithm when applied to the MCC planning problem (8). These algorithms are used to develop a MCCP for a two-node network made of Universal Software Radio Peripheral (USRP) Software-defined Radios (SDRs). Finally, the MCCP is implemented on this real wireless network and the effectiveness of the DQN-based DRL MCC planning is shown for exchanging OTA mission-critical information in the presence of a jammer and wireless channel anomalies.

THEORETICAL BASIS

First, we develop a mathematical model for the MCC planning problem using game-theory. Consider two nodes, A and B, that need to exchange their private, unique messages with each other within a given time constraint of N decision-epochs for a mission to be successful. The messages of nodes A and B are made of N_A and N_B and packets, respectively. There are M/2 frequency channels in each direction. At each decision-epoch, nodes make a decision on who should transmit using which channel. Since channels are unidirectional, the choice of channel also implies which node will transmit. This is a finite-horizon sequential decision problem in which N decisions are to be made so that N_A and N_B packets can be exchanged in the two directions. To develop a formal approach to solving this mission planning problem, we use game theory.

The MCC planning problem is an extensive-form game made of N sequential actions. Since two nodes are on one side against either the nature or a jammer, or both, the two nodes are represented by a single "Radio" player. The other player is the nature or the jammer. Figure 3

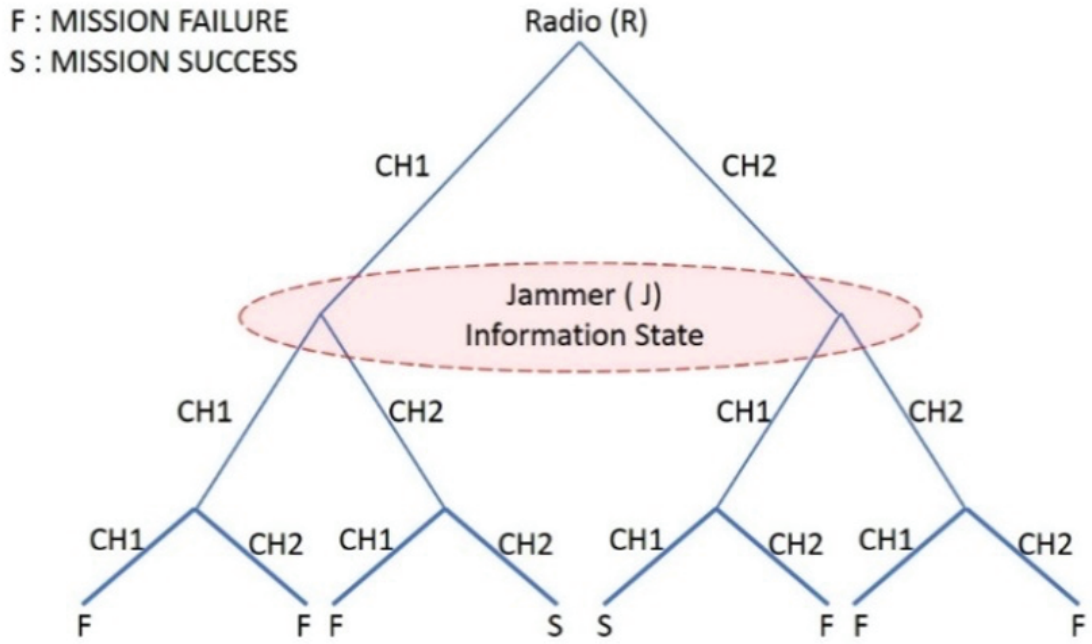


Figure 3. Extensive-form Game Representation of the Mission-Critical Communications Planning Problem with Two Decision Epochs.

shows the extensive-form game representation of a mission consisting $N = 2$ epochs with each link having one frequency channel (CH 1 from A to B and CH 2 from B to A).

The game played at each decision epoch can be characterized by a *state* that represents the progress of nodes towards completing the mission. Hence, this is a *stochastic*, or a *Markov*, game. At each epoch of the game/mission, a channel is selected with the goal of ensuring exchange of messages of each other before the N decision epochs are over. Since what matters is the mission success or failure at the end of the N periods, we define the reward for selecting action a when in state s at time t as

$$r_t(s,a) = \begin{cases} 1 & if\ L_A^t = 0\ \&L_B^t = 0 \\ -1 & if\ (L_A^t \neq 0\ or\ L_B^t \neq 0)\ \&\ T_R^t = 0 \\ 0 & otherwise \end{cases} \quad (1)$$

Where $L_A{}^t$, $L_B{}^t$ and $T_R{}^t$ denote the remaining packets at node A, remaining packets at node B and the remaining time before the mission runs out of time respectively.

A suitable formulation is to seek a channel selection policy over the mission-duration (MD) in order to maximize the following *discounted-reward*:

$$R = \sum_{t=1}^{N} \gamma^t r_t \quad (2)$$

Where $0 \leq \gamma \leq 1$ is a discount-factor. The most well-known solution concept for games is the Nash Equilibrium (NE). However, in extensive-form games the NE proves inadequate since some NE action sequences may not look credible when players are half-way through the mission, although they might have looked credible at the outset. Hence, when solving extensive-form games, the most common goal instead is to find a so-called *subgame perfect equilibrium (SPE)* which is an NE of the game with the additional requirement that any restriction of the strategy is a NE for the corresponding restriction of the game for all sub-games of the game (6).

In the MCC planning game, the number of policies to consider can be very large and even uncountable. For example, observe from Figure 3 that the number of action paths increases exponentially with mission-duration N and the channels on each link, making analytically solving for the desired equilibria in real-time impractical. For example, finding a solution along the game-theoretic criterion of sub-game perfect equilibrium can be computationally prohibitive. Adding an adversary (i.e., a jammer) further increases the game's complexity.

As a practical solution that can be applicable to a wide range of situations, this project proposes to train a suitable *artificial neural network* (ANN) through DRL to learn an optimal MCC plan. Such neural networks are called the DQN's (8).

EXPERIMENTAL METHODS

Deep Q-Network DRL Solution for Mission-Critical Communications Planning Problem: Since the radio's decisions must be a function of the amount of message packets remaining at each node ($L_A{}^t$ and $L_B{}^t$) as well as the remaining time from the mission-duration ($T_R{}^t$), we define the state s_t of the game at time t as the following 4-tuple:

$$s_t = (L_A^t, L_B^t, a_{t-1}, T_R^t)^T \tag{3}$$

Where a_{t-1} denotes the channel selected at the previous decision-epoch. An experience-tuple is defined as $x = (s, a, r, s')$ if executing action $a_t = a$ when in state st = s results in system transitioning to the new state $s_{t+1} = s'$ while observing a reward $r_t = r$.

Shallow RL uses a *Q-table* to learn a discretized version of the optimal *state-action value function,* denoted as $Q(s_t,a)$, from which an optimal policy can be obtained (5). If the input state s_t takes a continuum of values, as in the MCC planning problem, using a Q-table to learn the state-action value function is not possible since there is an infinite number of states. Therefore, this project constructed an ANN to learn the state-action value function. The idea is to train the ANN sufficiently so that when the game state $s_t = (L_A{}^t, L_B{}^t, a_{t-1}, T_R{}^t)^T$ is input to the ANN, it can output the corresponding $Q(s_t,a)$ estimates from which an optimal policy can be obtained.

The DQN Algorithm, developed by the Google DeepMind in 2015, is a clever extension of the classic *Q-Learning algorithm* to handle very large (or even uncountable) state-action spaces (8). It uses a *Deep Neural Network* (DNN) as a function estimator to approximate the state-action value function $Q(s_t,a)$ and succeeds in learning the *optimal state-action value function* $Q^*(s, a)$ thanks to two innovative concepts: *experience-replay* and the *target network* (8). The experience replay avoids the temporal correlations of experiences used for training which can lead to non-convergence of ANN weights. The use of a so-called target Q-network removes the problem of having to learn a moving target. Figure 4 shows the DQN-DRL algorithm for MCC planning developed in this project. We denote this as the DQN Algorithm #1.

At each decision-epoch, the DQN Algorithm #1 either picks the action with the highest estimated $Q(s,a)$ value according to the DQN output, or randomly with an appropriate *exploration rate* ϵ. After executing the chosen action, nodes observe a reward r depending on whether the nodes succeeded in completing the mission or not and the resulting next state s'. The *experience tuple* $x=(s, a, r, s')$ is added to *the experience replay memory* and then a random *minibatch* of L experience tuples is drawn from the replay memory. These L experience tuples are used to update the weights of the DQN assuming that the desired output is the one that is predicted by the target Q-network. As is the case with all ANNs, DQN Algorithm #1 uses a variation of the so-called *back propagation algorithm* to update the weights (9).

Mission-Critical Communications Planning in a Real Wireless Network made of Software-defined Radios: The DQN-based DRL of MCC planning was fully-implemented as a protocol on an MCC network formed of real wireless transceiver hardware. We used two USRP SDRs as two radio nodes (A and B). Each node has a unique message to convey to the other node for

the mission to be a success. Two communication links were then defined going in opposite directions: A → B and B →A. Two channels each assigned to each link: Channels 1 and 2 for sending messages from A to B and Channels 3 and 4 for sending those from B to A.

Initialize $DQN\ Q(.)$ weights, set target $DQN\ Q_T = Q$

For mission episode $m = 1: M$

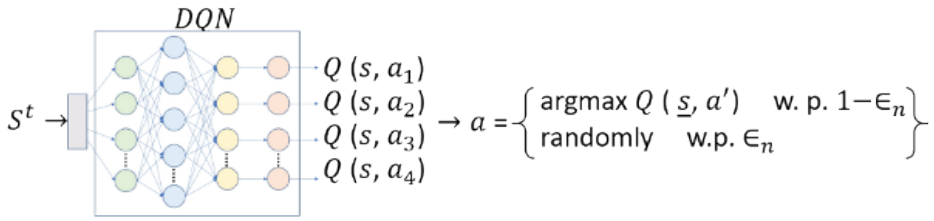
$\epsilon_n \leftarrow \epsilon_n - \frac{(\epsilon_0 - \epsilon_f)}{n}$

If mod (episode, T) == 0

Set $Q_T(.) = Q(.)$

For $n = 1: N$ mission duration

1. When In state $S^t = [\ L_A^t, L_B^t, a^{t-1}, T_R^t\ ]$, select action a such that :

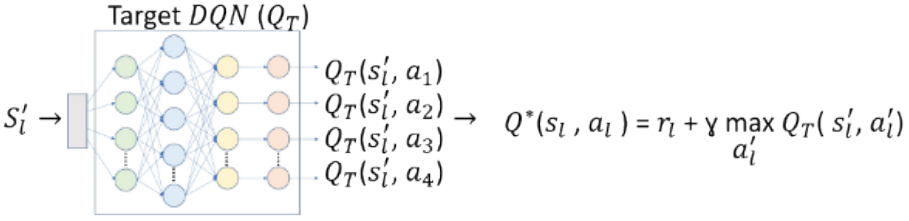


2. Execute action a, and observe next state $s^{t+1} = s'$ and reward $r^t = r$.

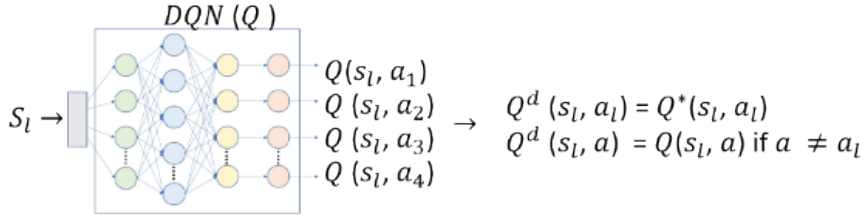
3. Add the experience tuple $e = (s, a, r, s')$ to the experience replay memory.

4. Select a random minibatch of experience $\mathcal{E}$ of size L from experience replay memory.

4a. For each $\underline{e_l} = (s_l\ , a_l, s_l', a_l) \in \mathcal{E}$



4b.



4c. $err(a) = Q_d(s_l, a\ ) - Q\ (s_l, a\ )$ for $\forall a$.

4d. Update DQN weights to minimize err vector.

end.

end.

Figure 4. DQN Algorithm #1: Deep Q-Network Deep Reinforcement Learning (DQN-DRL) Algorithm.

The total time available to complete the mission, called the mission-duration, is N decision epochs. Within each decision epoch of duration T_d, a node can transmit a random number of packets. In terms of decision epoch index k, for $k = 1,\ldots,N$, the state s_k of the game at decision epoch k is $s_k = (L_A^k, L_B^k, a_{k-1}, T_R^k)^T$.

The implementation of DQN-based MCC planning on USRPs required the development of an MCCP protocol for two USRPs to establish and maintain a wireless link. This MCCP was implemented in LabVIEW and integrated in to the USRP hardware over an Ethernet port. Each USRP was connected to its own computer terminal running LabVIEW software. For networking simplicity, we implemented the DQN-DRL only at the node A USRP and made Node B execute actions learned at node A and provided to it over a wireless feedback channel by A.

For this scheme to work, the USRP A which maintains the DQN needs to be able to determine if and when a node's receiving channel is jammed so that it can decide whether to transmit or receive and which channel to switch to. To construct the state s_k at epoch k, it must also know how much message is remaining on B's side (L_B^k), as well as its own (L_A^k).

Obviously, Node A always knows the value of L_B^k. It can also determine if its own reception is jammed by observing how many packets it receives within a decision epoch and comparing that with a threshold. For Node A to learn the value of L_A^k, Node B must send Node A acknowledgments for packets it receives. For this also we used the same feedback channel: In each decision epoch, node B feedback the number of packets received from node A so far when it successfully decodes a packet. We assume that the feedback channel operates on a completely separate frequency from the four communications channels used for actual message packets and is sufficiently secure so that the action decisions and acknowledgment messages cannot be jammed.

Figure 5 shows the timing flow-graph of the designed MCCP that was implemented on a wireless network made of two USRP SDRs. In the example in Figure 5, the action sequence selected by the DQN agent is channels 1, 3, 4, and 2.

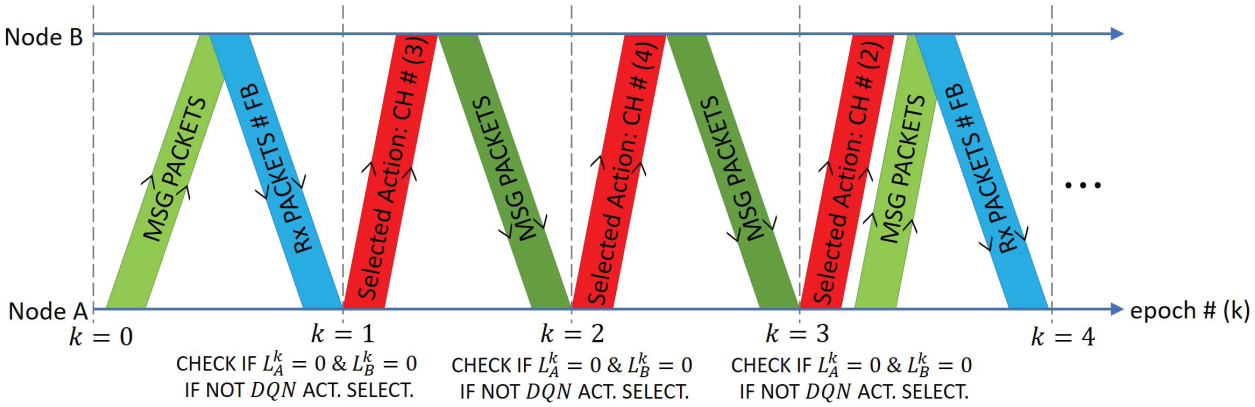


Figure 5. Designed and Implemented DQN Planning-based Mission-Critical Communications Protocol.

Statistical Modeling of Over-the-Air Mission-Critical Communications for Offline Training of Deep Q-Network: Directly constructing a DQN in LabVIEW and training on USRP hardware in real-time is a time-consuming task since an actual communications mission may last 10 to 20 minutes. Hence, a DQN was first designed, trained, and tested offline by simulating the entire system made of radios and jammers in MATLAB and then fine-tuned by updating weights during real-time MCC. For this approach to be successful, the MATLAB simulation needs to be as close to the real communications system as possible.

The critical quantity that needs to agree with the actual wireless network is the number of packets successfully communicated OTA during each decision epoch on links from nodes A to B and B to A, denoted respectively by random variables X_A and X_B. We model X_A and X_B as *normal random variables* (10).

A normal random variable is fully characterized by its mean and the variance (10). If $X_{A,i}$ and $X_{B,i}$, for $i = 1, \ldots, n$, denote a collection of measurements of X_A and X_B, the means and the variances of packet throughput per decision epoch on each direction can be estimated as follows:

$$\mu_A = \frac{1}{n}\sum_{i=0}^{n} X_{A,i} \quad \text{and} \quad \mu_B = \frac{1}{n}\sum_{i=0}^{n} X_{B,i} \tag{4}$$

$$\sigma_A^2 = \frac{1}{n}\sum_{i=0}^{n}\left(X_{A,i} - \mu_A\right)^2 \quad \text{and} \quad \sigma_B^2 = \frac{1}{n}\sum_{i=0}^{n}\left(X_{B,i} - \mu_B\right)^2$$

where μ_A and μ_B are the means of X_A and X_B, respectively and σ_A^2 and σ_B^2 are the variances of X_A and X_B, respectively. Then during each decision epoch of a simulated mission, we draw X_A according to the following normal pdf (similarly, for X_B) (10):

$$p_{X_A}(x) = \frac{1}{\sqrt{2\pi}\sigma_A} e^{-\frac{1}{2\sigma_A}(x-\mu_A)^2} \tag{5}$$

Adversarial Jammer Design and Implementation: The jammer is a separate radio capable of transmitting on any channel in either link. The developed sequential jammer jams channels one at a time in a pre-defined order. It jams a channel for a certain duration and changes to the next channel. The initial channel of this jammer is drawn uniformly randomly. Hence, even with a pre-defined sweeping pattern, this jammer can pose a significant challenge to an MCC link due to the randomness observed in the jamming sequence in different mission trials. This was implemented as a protocol on LabVIEW and integrated on to a third USRP SDR.

A Brand-New DQN-DRL Algorithm: Deep Policy-hill Climbing (DPHC) Algorithm: A drawback of the *Q-Learning algorithm* (11) used in the DQN Algorithm #1 is that it can only learn *pure strategies*. A pure strategy assigns 100% probability to a single action. However, a mixed strategy assigns probabilities to several different actions. This ensures the possibility of learning an NE since mixed strategy NE are guaranteed to be present while pure strategy NE may not always exist (6). Moreover, traditional Q-Learning can suffer from possible slow convergence in multi-agent games (12). The so-called Policy Hill Climbing (PHC) algorithm is a formal approach to learn mixed strategies in multi-agent games (13).

For the first time, this project developed a Deep PHC algorithm that accelerates the policy learning through a variable learning rate. In Figure 6 we show the newly developed *Deep Policy-hill Climbing* (DPHC) DRL algorithm for MCC planning that we have denoted as the DQN Algorithm #2. As can be seen from Figure 6, the DPHC algorithm operates similar to the DQN

Initialize $DQN\ Q(.)$ weights, set target $DQN\ Q_T = Q$

For mission episode m= 1 : M

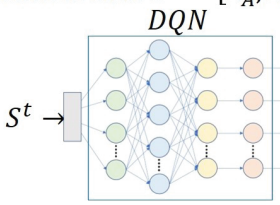
$\epsilon_n \leftarrow \epsilon_n - \frac{(\epsilon_0 - \epsilon_f)}{n}$

If mod (episode, T) = = 0

Set $Q_T(.) = Q(.)$

For $n = 1: N$ mission duration

1.When in state $S^t = [L_A^t, L_B^t, a^{t-1}, L_R^t]$, select action a such that :



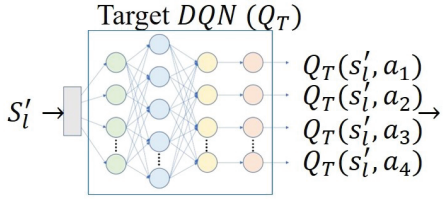
$$\rightarrow a = \begin{cases} \text{Argmax } Q(s, a') & \text{w.p. } 1-\epsilon_n \\ \text{randomly} & \text{w.p. } \epsilon_n \end{cases}$$

2. Execute action a, and observe next state $s^{t+1} = s'$and reward $r^t = r$.

3. And the experience tuple $e = (s, a, r, s')$ to the experience replay memory.

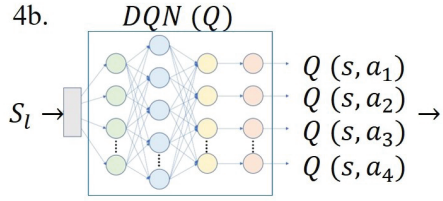
4. Select a random minibatch of experience $\mathcal{E}$ of the size L from the experience replay memory.

4a. For each $e_l = (s_l, a_l, s_l', a_l) \in \mathcal{E}$



$$\rightarrow Q^*(s_l, a_l) = r_l + \gamma \max_{a_l'} Q_T(s_l', a_l')$$

4b.



If, $a_l = \arg\max_{a'} Q(s_l, a')$

$Q^d(s_l, a_l) = Q^*(s_l, a_l) + \delta$

$Q^d(s_l, a) = Q(s_l, a_l) - \frac{\delta}{(M-1)}$ if $a \neq a_l$

Otherwise

$Q^d(s_l, a_l) = Q^*(s_l, a_l)$

$Q^d(s_l, a) = Q(s_l, a_l)$ if $a \neq a_l$

4c. $err(a) = Q_d(s_l, a) - Q(s_l, a)$ for $\forall a$.

4d. Update DQN weights to minimize eff vector.

end.

end.

Figure 6. DQN Algorithm #2: Deep Policy-hill Climbing (DPHC) Deep Reinforcement Learning Algorithm.

but for a key difference in updating the weights in step (4d): we reinforce those action choices that are the best for that state according to the current DQN by amplifying the desired $Q(s,a)$ for those actions while de-emphasizing those for the remaining actions.

RESULTS

Since the primary objective of mission-planning is achieving mission-success, our main performance metric is the *mission-success probability* (MSP) defined as:

$$mission\ success\ probability\ (MSP) = \frac{\#\ of\ missions\ completed\ successfully}{total\ \#\ of\ missions} \tag{6}$$

To show that the DQN is an effective approach for solving the MCC planning problem, performance of the DQN was evaluated in five cases.

In the first case based entirely off MATLAB simulations, we simulated a general MCC planning problem in which the message at each node is made of five packets and during each decision epoch a single packet can be transmitted. We used an ANN made of three hidden layers with 24 neurons each. Both input and output layers are made of four neurons each corresponding to the length of the state vector and the number of channels. We denote this as the 4x24x24x24x4 net.

Figure 7 shows the MSP during and after training for mission-durations *N=10, 12* or *14* epochs in the presence of a sequential jammer. Since initial channel of this jammer is drawn uniformly randomly, even with a pre-defined sweeping pattern, it poses a significant challenge to the mission-success. Figure 7 not only shows the effectiveness of DQN in learning good planning policies, but also shows how difficult the learning problem is as the mission-duration tightens.

Figure 8 shows the effectiveness of the MCC planning with a DQN-DRL (after training over 10K missions) when compared to random decisions. Clearly, DQN-DRL results in MCC planning policies that are significantly better performing than those with random decisions. We quantify this by defining the *Percentage Mission-Success Improvement* (PMSI) relative to the random planning as below:

$$Percentage\ Mission\ Success\ Improvement\ (MSI) = \frac{MSP_{DQN} - MSP_{RP}}{MSP_{RP}} \times 100\% \tag{7}$$

where MSP_{DQN} and MSP_{RP} refer, respectively, to MSP with DQN based DRL and with random planning.

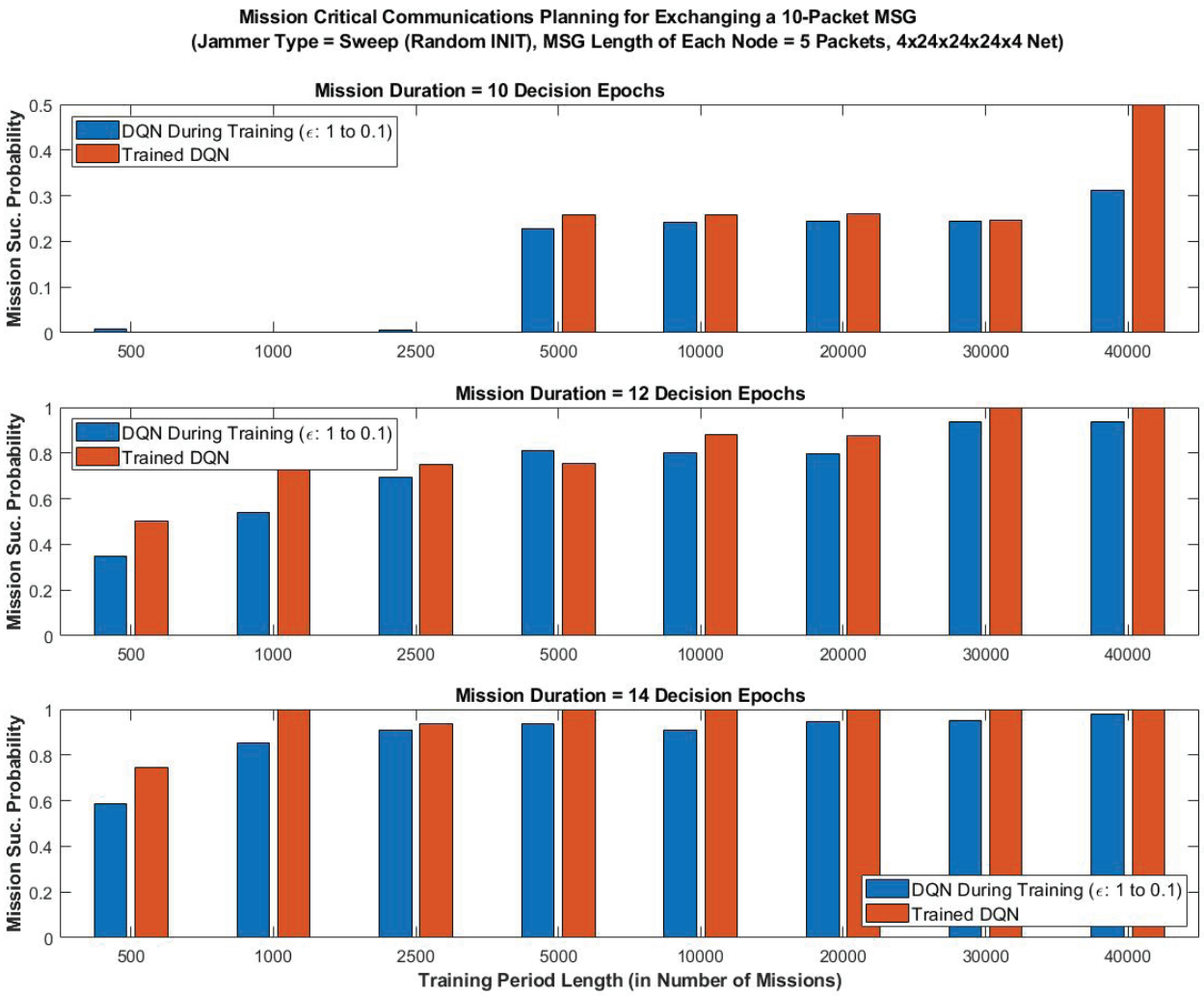


Figure 7. Mission-Success Probability of DQN-based Mission-Critical Communications Planning during and after Training in the Presence of a Sequential Jammer. (TOP) Planning Over 10 Decision Epochs, (MIDDLE) Planning Over 12 Decision Epochs, (BOTTOM) Planning Over 14 Decision Epochs.

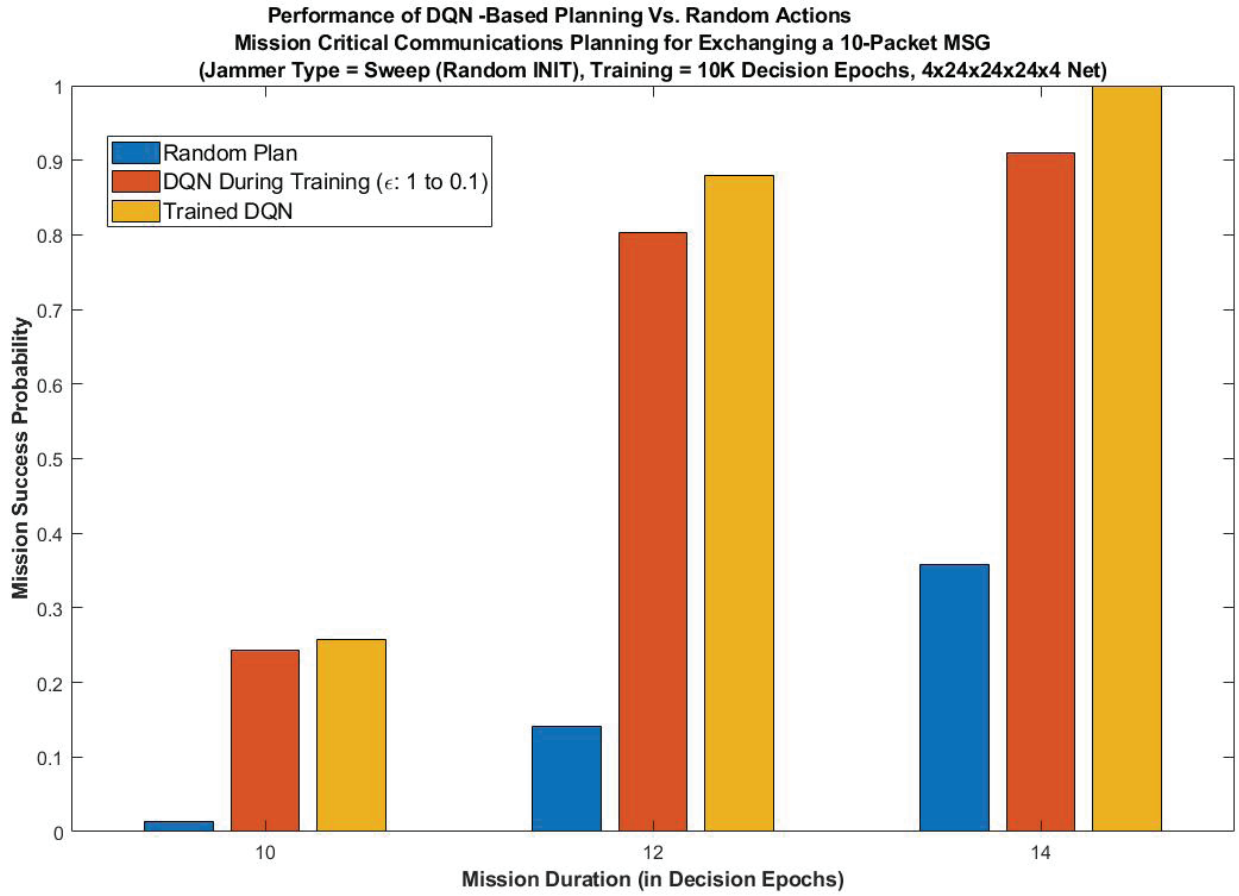


Figure 8. Performance Improvement of DQN-based Mission-Critical Communications Planning during and after Training Compared to Random Planning Lengths in the Presence of a Sequential Jammer (Different Mission-Durations).

Table 1 shows the computed PMSI improvements after training corresponding to the results shown in Figure 8. The improvements are more pronounced for shorter mission-durations since there is more room for improvement due to the smaller MSPs.

Mission-Duration (decision epochs)	Percentage Mission-Success Improvement (PMSI)
10	1739%
12	520%
14	179%

Table 1. Percentage Mission-Success Improvement with DQN-DRL Planning Relative to Random-decision Planning.

Next, we offline trained a DQN suitable for the hardware-implemented MCC system built with USRPs controlled by LabVIEW. The four channel frequencies were 2.04 GHz, 2.08 GHz, 2.12 GHz and 2.16 GHz. Both nodes used QPSK modulation at 50K symbols/second. The feedback channel operated at frequency 2.38GHz using the same modulation. We assigned each node a message made of 100 packets. Each packet contained a critical information that was packed in to 5888 bits. Before transmission 1380 guard bits and 1104 synchronization bits were added so that each transmit packet was about 8372 total bits.

To pre-train a DQN in offline simulations, we estimated the means and variances of random variables X_A and X_B corresponding to the number of successful packet transmissions from node A to B and B to A, respectively. For this, we collected the number of packets transferred on each link during a decision epoch as shown in Table 2.

Link	Ep. 1	Ep. 2	Ep. 3	Ep. 4	Ep. 5	Ep. 6	Ep. 7	Ep. 8	Ep. 9	Ep. 10
A to B ($X_{A,i}$)	72	71	67	78	81	87	90	82	89	75
B to A ($X_{B,i}$)	84	87	70	89	84	79	72	77	80	86

Table 2. Number of Successful Packet Transmissions During a Decision Epoch.

From these measurements we computed the means and the variances of packet throughput per decision epoch on each direction as follows:

$$\mu_A = \frac{1}{n}\sum_{i=0}^{n} X_{A,i} = 79.200 \quad \text{and} \quad \mu_B = \frac{1}{n}\sum_{i=0}^{n} X_{B,i} = 80.800$$
$$\sigma_A^2 = \frac{1}{n}\sum_{i=0}^{n}\left(X_{A,i} - \mu_A\right)^2 = 63.505 \quad \text{and} \quad \sigma_B^2 = \frac{1}{n}\sum_{i=0}^{n}\left(X_{B,i} - \mu_B\right)^2 = 60.628 \tag{8}$$

Figure 9 shows the MSP with the above model for the packet transmission distribution on each link for mission-durations of $N = 4, 6$ and 8 decision epochs.

Figure 10 shows the MSP of the DQN-DRL mission-critical communications planning compared to random planning while Table 3 shows the PMSI corresponding to the results shown in Figure 10.

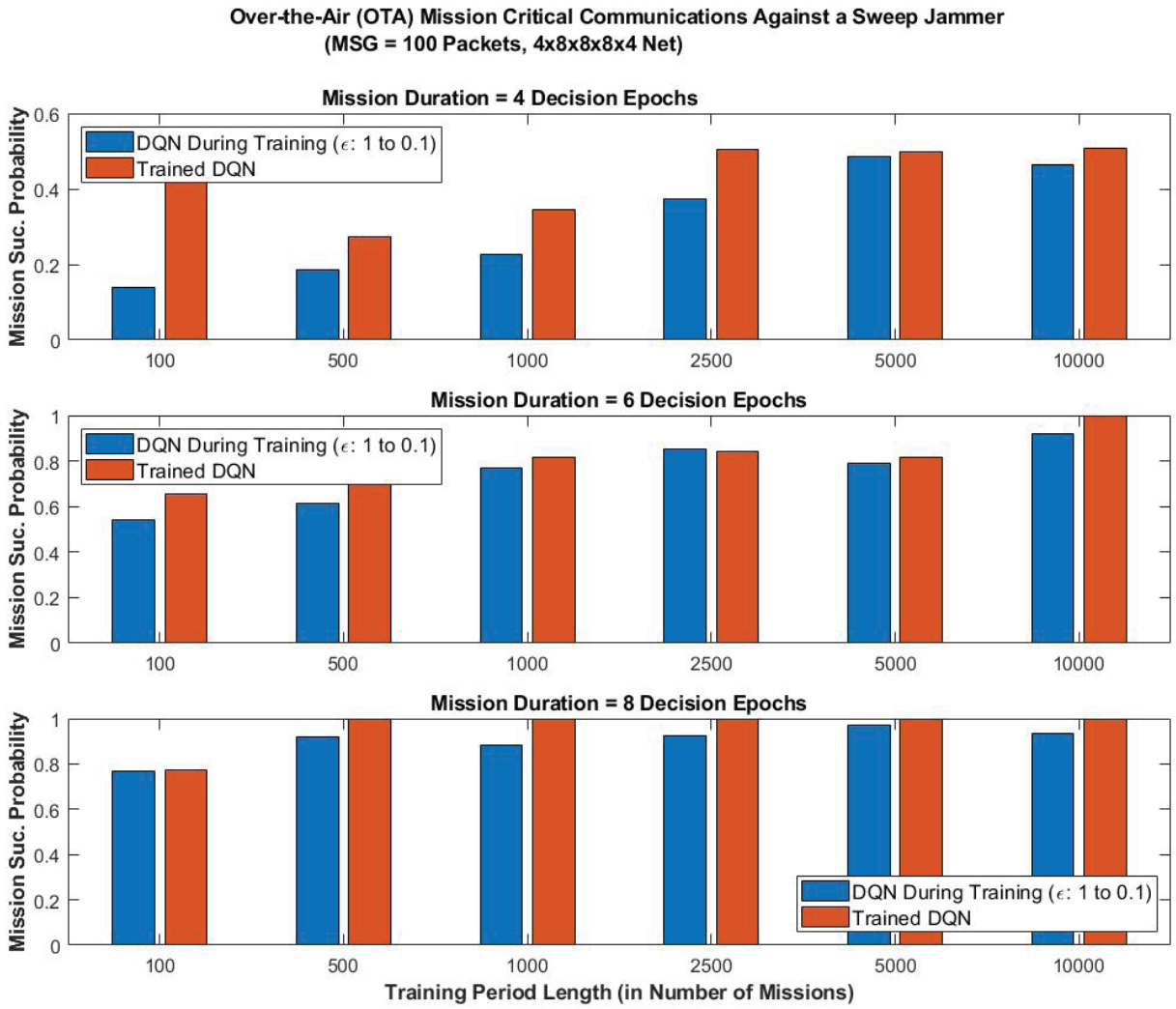


Figure 9. Mission-Success Probability of DQN-based Mission-Critical Communications Planning in a Wireless Network during and after Training in the Presence of a Sequential Jammer. (TOP) Planning Over 4 Decision Epochs, (MIDDLE) Planning Over 6 Decision Epochs, (BOTTOM) Planning Over 8 Decision Epochs.

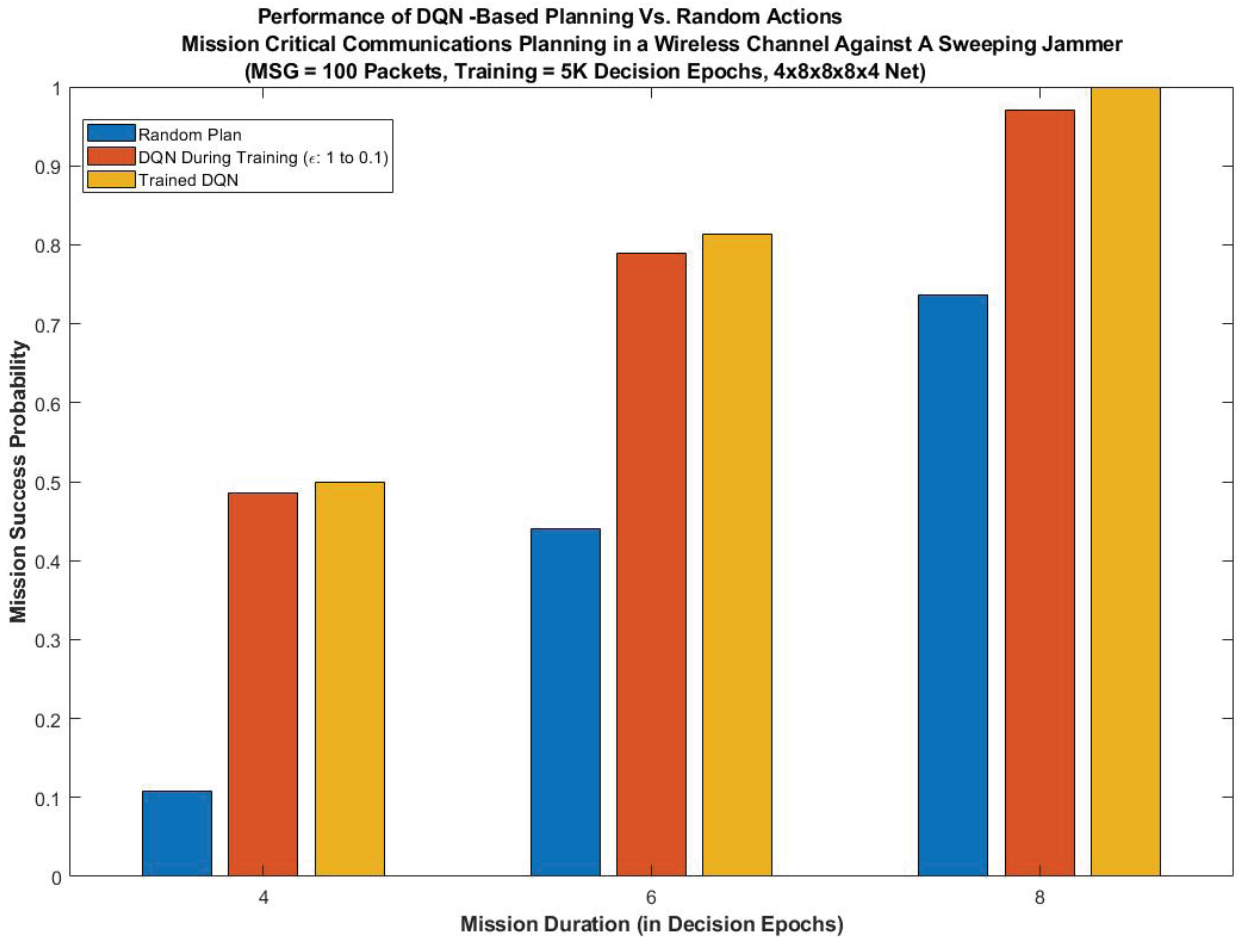


Figure 10. Performance Improvement of DQN-based Mission-Critical Communications Planning in a Wireless Network during and after Training Compared to Random Planning Lengths in the Presence of a Sequential Jammer (Different Planning Lengths).

Table 3 shows that the trained DQN-based MCC planning achieves significantly better mission-success improvements over random planning policies. Again, with tighter mission-durations, the improvements are more pronounced since MSPs are relatively low.

Mission-Duration (decision epochs)	Percentage Mission-Success Improvement (PMSI)
4	359%
6	85%
8	36%

Table 3. Percentage Mission-Success Improvement with DQN-DRL Planning Relative to Random-decision Planning in a Wireless Network Simulated with Estimated Throughput Parameters.

The third set of experiments evaluated the DQN-based MCC planning performance with OTA mission-critical information exchanges in an actual wireless network made of USRP SDRs. The DQN made of the 4x8x8x8x4 net trained for a mission-duration of *N=6* epochs over a *20K* training period was integrated into the node A's LabVIEW program to implement the MCCP protocol. The actual OTA MCC testing of the two-node link in the presence of

a sequential jammer was performed at the Communications Lab at the University of New Mexico. Jammer was designed to sequentially jam channels starting from a random channel with a channel change every 3 epochs.

Figure 11 shows the remaining packets at node A (Left Y-axis) and node B (right Y-axis) at each decision epoch as the mission evolves. Each decision epoch was about 150 seconds so that the duration of a single mission made of six decision epochs was about 15 minutes. Figure 11 summarizes results obtained over 10 mission trials. As can be observed from Figure 11, only 1 out of the 10 missions ended in failure showing the power of DQN-based DRL to learn effective mission planning policies in dynamic channels with wireless propagation anomalies and adversarial jammers.

We denote by random variable T the number of decision epochs for completing the mission when the mission was successfully completed. Table 4 shows observed values of T for the above 10 mission trials.

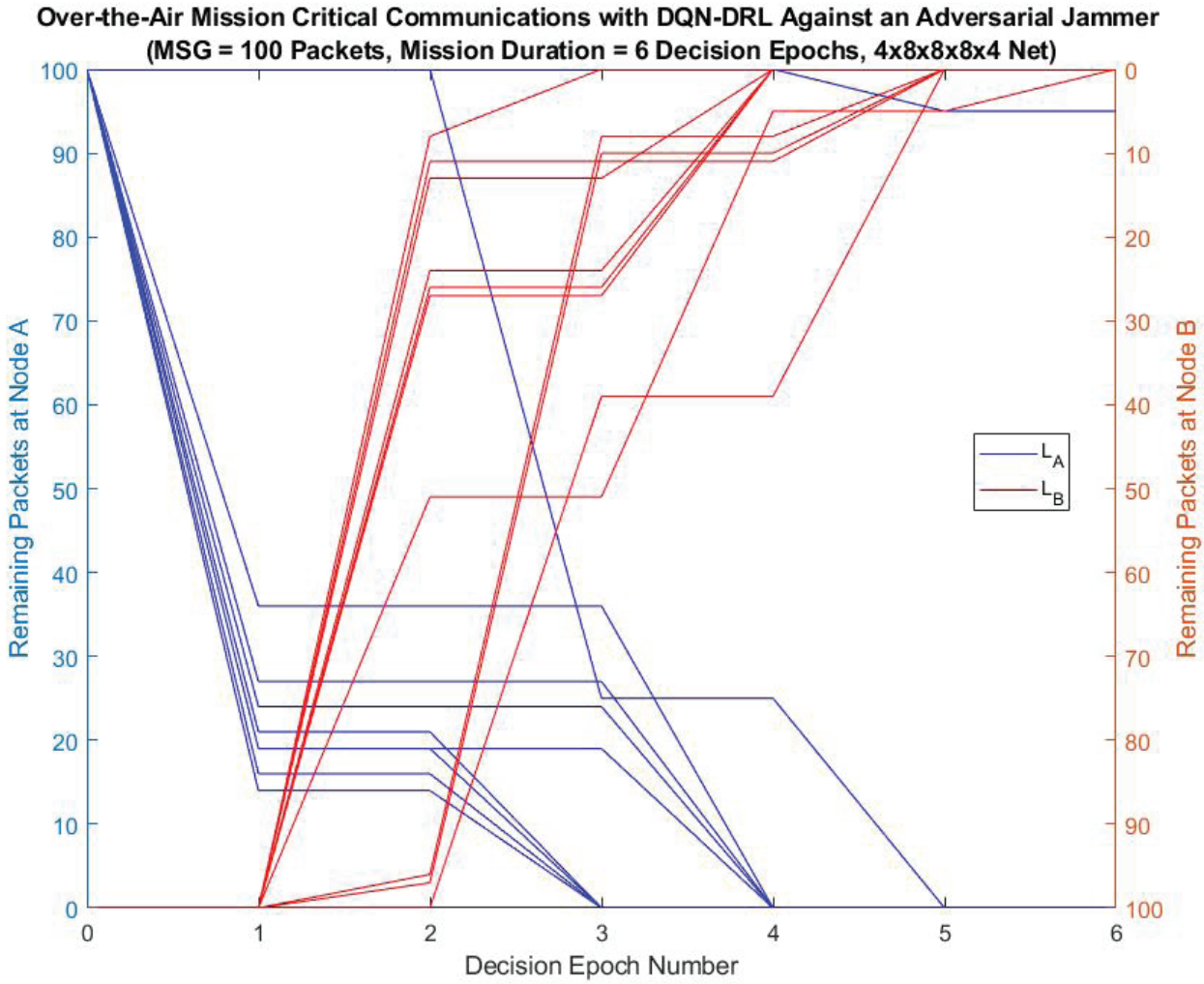


Figure 11. Critical Message Exchange Performance of DQN-based Mission-Critical Communications Planning in an Actual Wireless Network in the Presence of a Sequential Jammer.

Mission #	1	2	3	4	5	6	7	8	9	10
Epochs for mission completion (T)	4	6	5	N/A	4	5	4	4	5	5

Table 4. Number of Decision Epochs to Complete Each Successful Over-the-Air Mission.

From this data we compute the average duration for mission completion, denoted by the symbol T with a line over it, when the mission is completed:

$$\bar{T} = \frac{1}{K}\sum_{i=1,T_i\leq N}^{n} T_i = 4.67 \text{ decision epochs} \quad (9)$$

Where $K \leq N$ is the number of missions that was completed successfully. The spread of T around this average can be characterized by its standard deviation computed as:

$$standard\ deviation\ of\ T = \sqrt{\frac{1}{K}\sum_{i=0,T_i\leq N}^{n}\left(T_{,i} - \bar{T}\right)^2} = 0.7071 \quad (10)$$

Fourth case was to evaluate the performance of MCC planning in wireless networks with DQN-based DRL for larger critical message lengths of 250 and 500 packets. Tables 5 and 6 summarize the performance averaged over 10 trials for different mission-durations.

MD = 8 decision epochs			MD = 10 decision epochs			MD = 12 decision epochs		
MSP_{DQN}	MSP_{RP}	PMSI	MSP_{DQN}	MSP_{RP}	PMSI	MSP_{DQN}	MSP_{RP}	PMSI
0.51875	0.06698	674%	0.83853	0.27745	202%	0.88875	0.52955	68%

Table 5. Averaged Performance of MCC Planning with DQN-DRL in Simulated Wireless Networks with Estimated Realistic Throughput Parameters. Message Size = 250 Packets.

MD = 16 decision epochs			MD = 18 decision epochs			MD = 20 decision epochs		
MSP_{DQN}	MSP_{RP}	PMSI	MSP_{DQN}	MSP_{RP}	PMSI	MSP_{DQN}	MSP_{RP}	PMSI
0.52749	0.06768	679%	0.64726	0.19251	236%	0.83078	0.36954	125%

Table 6. Averaged Performance of MCC Planning with DQN-DRL in Simulated Wireless Networks with Estimated Realistic Throughput Parameters. Message Size = 500 Packets.

Tables 5 and 6 affirm the effectiveness of DQN-based MCC planning in ensuring mission-success with drastically higher likelihoods compared to what is achievable with random planning.

Finally, we evaluated the performance of our new DPHC algorithm compared to the Google DeepMind's original DQN algorithm. Figure 12 shows the MSP achieved under DQN and DPHC algorithms during training and testing.

According to Figure 12, the DQN outperforms DPHC during training, while during testing, DPHC-DRL outperforms the DQN. The reason is that during training, we force our DPHC to select actions according to a mixed policy whereas original DQN is allowed to follow the pure strategy suggested by the DQN. This makes the DPHC to select inferior actions that lead to mission-failure. However, these failures allow it to learn a more robust decision policy. During testing, on the other hand, both DQN and DPHC are allowed to select actions according to the greedy pure strategy derived from the predicted Q(s,a). Now, the DPHC's more robust policy is capable of planning MCC more successfully while original DQN displays somewhat erratic performance. With 10K training, the new DPHC achieves a PMSI of 29% relative to the original DQN.

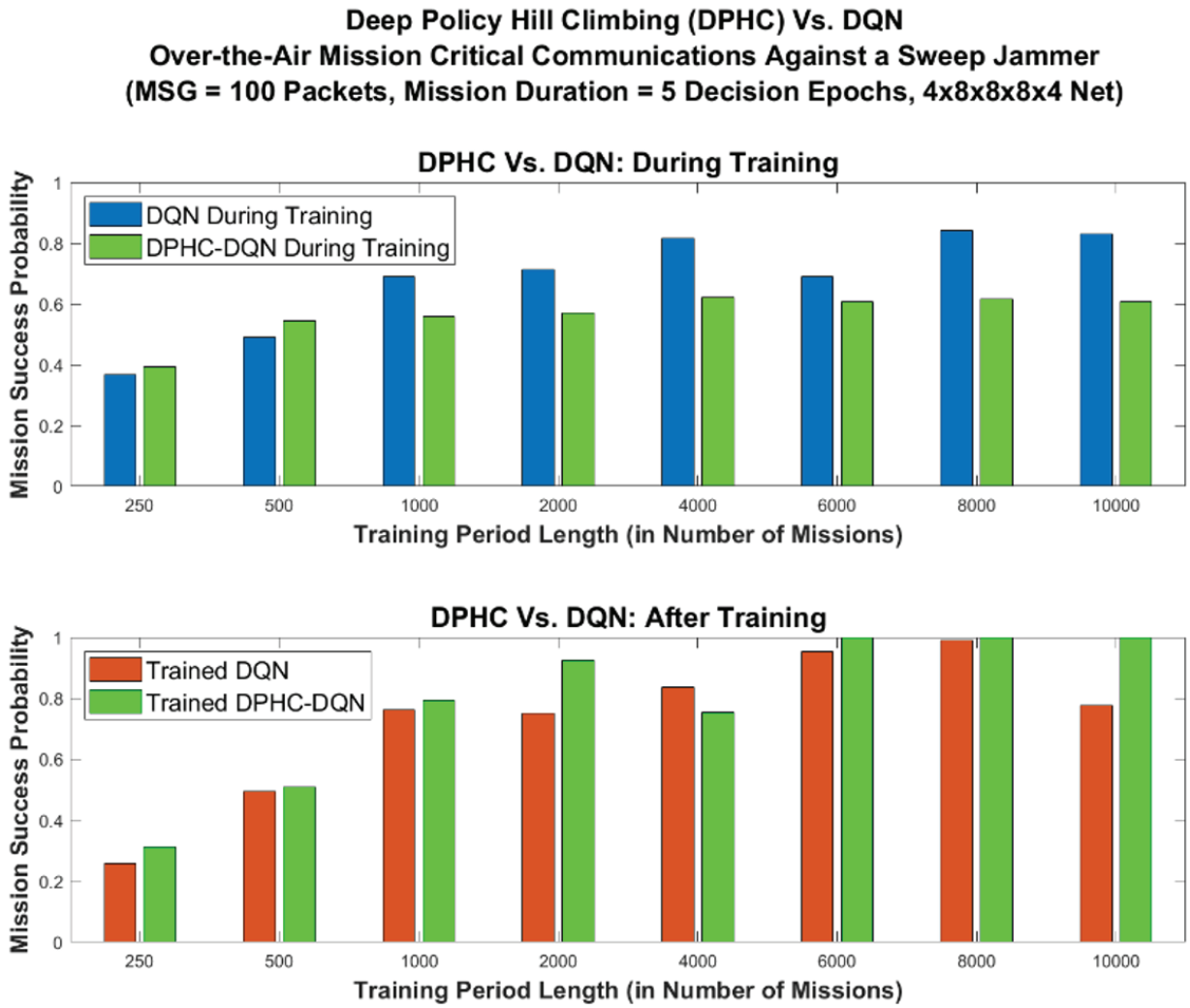


Figure 12. Comparison of Mission-Success Probability of DQN-based and the Newly Designed DPHC-based Mission-Critical Communications Planning (Over 6 Decision Epochs) in the Presence of a Sequential Jammer. (TOP) during Training. (BOTTOM) after Training.

DISCUSSION

The first experiment case established the potential of the proposed approach in a simplified setting without the added complications of wireless channel anomalies, node synchronizations, computational/processing delays, and USRP hardware imperfections. Results in Figures 7 and 8 verified that indeed MCC planning with DQN-based DRL can lead to significant performance improvements compared to random policies. For example, for mission-durations of 14 epochs, the DQN's MSP of 0.9998 is a 179% improvement over the random planning. This case also served to offer guidelines on how to select the overall structure of the ANN as well as parameter values such as learning rate and training period length.

The second case used a faithful simulation of wireless networks with estimated throughput parameters to train a DQN that has a good chance of performing well once integrated into the actual hardware. The key to the success of this approach was good statistical modeling and

estimation of the key parameters that affect the MCC planning. The 90% MSP achieved in the third case, once a DQN trained with this model was integrated in to the real USRP hardware based wireless network, proved that our statistical modeling and estimation approach of the associated parameters are indeed reasonable.

As seen in Figure 10, the DQN does considerably better in all tested mission-durations compared to random planning. Tables 5 and 6 showed that these conclusions hold even with larger critical information messages of 250 and 500 packets. In all cases, the reported performance metrics were evaluated by averaging over 10 trials to reduce the effects of statistical variations.

Finally, we compared the performance of the MCCP based on the developed DPHC Algorithm to that based on the original DQN algorithm. As Figure 12 illustrates, our DPHC proved on-par or better than the DQN after the training.

Looking ahead towards future work, since the DQN must learn to complete a mission as quickly as possible, a new reward system can be designed to assign higher rewards whenever a mission is completed with a smaller number of decision epochs, while a fixed negative reward is assigned for those failed missions as before. For example, if the nodes managed to finish 2Td time intervals before the mission duration, a total of +2 would be added to the episode reward rather than the meager +1. This incentivizes the DQN to complete a mission faster. Another future work is to analyze the effectiveness of the developed DQN-based DRL policies against other types of jammers such as Markov and smart jammers.

CONCLUSIONS

The contributions of this project are two-fold: First, the project developed a novel mathematical model for mission-critical operations and showed through MATLAB simulations that a DQN can learn how to solve the associated planning problem. Second, it explored an example scenario of exchanging OTA messages in a real wireless network formed of USRP SDRs within a time constraint. In addition, a new DPHC algorithm was developed that outperformed the Google DeepMind's DQN algorithm significantly. Hence, the project not only serves as an important contribution to the field of MCC planning but also shows the relevance of the developed DQN-based DRL approach to a wide range of planning under uncertainty problems. These include disease control, food/crop management, refugee crises, troop distribution, and stock optimization to name a few.

Ultimately, this project demonstrates that DQN-based DRL can be an effective solution for mission planning problems. Since currently most mission-critical operations are being carried out manually making them prone to human error that can lead to disastrous outcomes, this project serves to make the case for adapting AI techniques to remove or reduce the risk of human errors in high stake missions.

AUTHOR INFORMATION

Corresponding Author

*Milidu Jayaweera, milidutkd@gmail.com

ACKNOWLEDGMENT

I would like to thank Dr. Sudharman Jayaweera for getting me access to the UNM Communications Laboratory and Dr. Mohamed A. Aref for aiding in laboratory experiments. Special thanks to my science teacher Mrs. Lena Eddings for her time filling out the numerous forms to ensure my participation in the contests of my choosing.

ABBREVIATIONS

AI, Artificial Intelligence; ANN, Artificial Neural Network; DNN, Deep Neural Network; DPHC, Deep Policy Hill Climbing; DQN, Deep Q-Network; DRL, Deep Reinforcement Learning; MCC, Mission-critical Communications; MCCP, Mission-critical Communications Protocol; MD, mission-duration; MDP. Markov Decision Process; ML, Machine Learning; MSP, Mission-success Probability; NE, Nash Equilibrium; OTA, over-the-air; PMSI, Percentage Mission-success Improvement; RL, Reinforcement Learning; SDR, Software-defined Radio; SPE, Subgame Perfect Equilibrium; USRP, Universal Software Radio Peripheral.

REFERENCES

1. Grover, K.; Lim, A.; Yang, Q. Jamming and Anti-jamming Techniques in Wireless Networks: A Survey. *International Journal of Ad Hoc and Ubiquitous Computing.* **2014**, 17(4), 197-215.

2. Understanding the Difference Between Mission-Critical and Business-Critical Communications; https://usatcorp.com/understanding-the-difference-between-mission-critical-and-business-critical-communications/

3. Blythe, J. An overview of planning under uncertainty. *AI Magazine*. **1999**. 20(2), 37-54.

4. Puterman, M. L. *Markov decision processes: Discrete stochastic dynamic programming*. John Wiley & Sons: Hoboken, NJ, 2010.

5. Sutton, R. S.; Barto, A. G. *Reinforcement learning: an introduction*. The MIT Press: Cambridge, MA, 2012.

6. Leyton-Brown, K.; Shoham, Y. *Essentials of Game Theory*. Morgan & Claypool Publishers, 2008.

7. Rappaport, T. S . *Wireless Communications: Principles and Practices*. Prentice-Hall, 2002.

8. Mnih V.; et al. Human-level control through deep reinforcement learning. *Nature*. **2015**, 518(7540).

9. Duda, R. O.; Hart, P. E.; Stork, D. G. *Pattern classification*. Wiley-Interscience, 2000.

10. Leon-Garcia, A. *Probability, Statistics, and Random Processes for Electrical Engineering*. Pearson Prentice Hall: Upper Saddle River, NJ, 2008.

11. Watkins, C.; Dayan, P. Q-learning. *Machine Learning*. **1992**, 8(3), 279-292.

12. Schwartz, H. M. *Multi-agent Machine Learning*. Wiley: New Jersey, USA. 2014.

13. Bowling, M.; Veloso, M. Multiagent learning using a variable learning rate. *Artificial Intelligence***. 2002**, 136(2), 215-250.