# Environment Robust Handover Optimization in Wireless Networks for Mobile Users

**Alperen Duru**
aduru@utexas.edu
Department of Electrical and Computer Engineering
The University of Texas at Austin

**Rosemary Lach**
rlach@utexas.edu
Department of Electrical and Computer Engineering
The University of Texas at Austin

**Project Github:** https://github.com/rosemarylach/RL-Project-LB

**Project Video:** https://youtu.be/cBMOCS_c808

## 1 Introduction

In a wireless communications system, each user equipment (UE) must be assigned to a base station (BS) for service. In a static system, each UE will ideally connect to the BS which will provide it with the highest signal-to-interference plus noise ratio (SINR). As UEs move, the individual SINRs and therefore the achievable total data throughput to the users might degrade if the UE-BS associations are not managed adaptively. UE-BS association changes can be handled by handovers of UE to a different BS that can supply a higher SINR. However, performing a handover itself takes time in which the UE is not being served, decreasing the average rate in time. This requires a balance between the number of handovers and the achievable rate for each UE-BS association, which affects the total system throughput. Reinforcement Learning (RL) algorithms can help solve the UE-BS associations that maximize the total throughput for mobile UEs. This project aims to extend the research paper (1), focusing on load balancing and handover optimization using deep reinforcement learning, to improve overall performance in simple antenna environments and allow for robustness across more diverse and complex environments.

### 1.1 Related Work

There are many opportunities for reinforcement learning to be used in the field of wireless communications, as many network systems can be modeled as a Markov Decision Process (MDP), where entities need to make decisions to optimize some network performance parameters. Deep reinforcement learning has been of particular interest in multiple problems such as traffic routing and resource allocation (2).

This project is primarily centered around the fundamental ideas presented in (1), a paper that focuses on load balancing and handover optimization using deep reinforcement learning. Gupta sets up a sequential decision-making problem that attempts to maximize the sum-log-rate as its reward by adaptively finding UE-BS associations with parameter $x_{ij}[t]$ for UE i and BS j. The state space consists of UE locations and velocities (implicitly represented through sets of reference signal received power (RSRP) values between each BS-UE pair), and loads on each BS. This project will use this same formulation and state space while attempting to make a more robust system model that has better performance and can be adapted to new environments. New 3GPP standards as outlined in (3) will be also used for this dataset, as opposed to the version published in 2017 that was originally used. However, no standards that are relevant to this project have changed since then.

In the literature, similar states and rewarding mechanisms taking into account the handover timings exist with different learning approaches to a similar problem. Authors of (4) discuss the mobile UAV and mobile users association for a mobile edge computing scenario and solve the learning problem by using the twin-delayed deep deterministic policy gradient (TD3). However, they only have 1 moving BS terminal and they are optimizing for the UAV trajectory only. Authors of (5) use deep neural networks to do the learning asynchronously at the UE side by clustering some UEs for faster and better learning. Multi-armed bandits and contextual bandits approaches are tested in (6) and (7), respectively, where (7) also includes the beamforming aspects of the system. Finally, (8) proposes a deep Q-learning approach to optimize for energy efficiency, where we have a different reward metric of sum-log-rate. Our work is going to distinguish itself from the literature by building on top of (1) to improve overall performance through reward shaping and robustness to changing environments through multi-agent learning.

## 2 System Model

The system model is defined similar to (1) being replicated as a first step; we model a downlink system of an urban macrocell (UMa) environment with multiple BS and UE, where the UEs are able to move along rectangular trajectories, simulating the motions in the streets. Each base station is equipped with $k$ multiple frequency bands using the same sectors, with $K$ frequency bands in total. Each base station has 3 sectors covering their region for the UEs traveling around them. For the analysis notation, different frequency bands and sectors of the same physical BS is assumed to be acting as if they were other BSs co-located at the same point in space. Fig. **??** is a representative high level system model of 2 physically different located BSs and a single UE connection. Here, the different sectors and frequency bands are also counted as different BSs that are co-located.

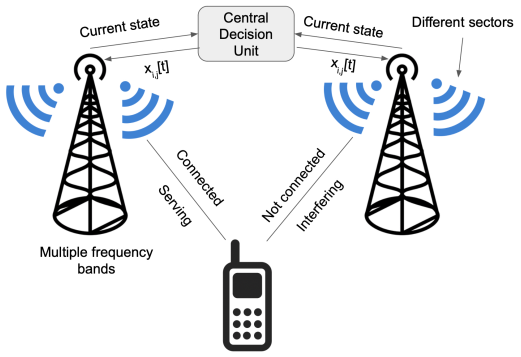


Figure 1: High Level System Model

The connected (or associated) BS serves the UE while the other BSs act as an interference. For RSRP signals for each UE $i$ and BS $j$ in the set $\mathcal{B}$, the received SINR value at time $t$ is computed as

$$\mathrm{SINR}_{ij}[t] = \frac{\mathrm{RSRP}_{ij}[t]}{\sigma^2 W_j + \sum_{j' \in \mathcal{B}, j' \neq j} \mathrm{RSRP}_{ij'}[t]},$$

where $\sigma^2$ is the noise spectral density and $W_j$ is the bandwidth of the BS $j$. Each BS will probably be serving multiple UEs at a given time, sharing the bandwidth. The rate with load taken into account can computed to be

$$c_{ij}[t] = \frac{W_j}{\ell_j[t]} \log_2\left(1 + \mathrm{SINR}_{ij}[t]\right).$$

The $\ell_j[t]$ stands for the current number of UEs being served by that BS, effectively sharing the bandwidth $W_j$. Finally, the effective rate a UE experiences within a time slot is also dependent on whether there was a handover or not for that particular UE in that time frame. Handovers also take time and therefore, it decreases the effective rate achieved by the UE. The effective rate then can be formulated as

$$r_{ij}[t] = \left(1 - \frac{T_{\mathrm{HO},i}[t]}{T_s}\right) c_{ij}[t],$$

where $T_{\mathrm{HO},i}[t]$ is the handover time and $T_s$ is the slot time.

## 3 Problem Formulation

The problem formulation follows a similar approach as in (1): to maximize the sum utility of the users by first defining it as an optimization problem, then mapping it into a a Markov decision process (MDP), with additional reward shaping mechanisms and different state spaces to observe their effects on the rate and handover distributions. The optimization problem becomes intractable due to its combinatorial nature, very large search space for an optimal utility, and its dependence on the environment. Within the radio environment, each handover decision is subject to a possible failure, causing time loss and an effective rate for the customer. This process can be reformulated as an MDP with a reasonable set of states, actions, and rewards. The agent then needs to learn the link failure probability and the wireless network environment by observing the states to estimate the next state's action values, increasing the users' rate performance versus the number of handovers.

### 3.1 UE-BS Association Optimization Formulation

The utility function to be optimized can be selected to be the log(.) utility function, which is commonly used in wireless communications for its diminishing returns property and fairness metric. However, there might be other utility functions that can generate better achievable rate per customer versus the number of handovers. The log(.) utility is used to formulate the optimization problem for its wide use in wireless communications for its tradeoff between high rate and fairness, and the other utility functions are discussed later in the report. The agent is a central controller with access to all RSRP values between each UE and BS. The decision task for the central controller is to decide at each time slot for the optimal UE-BS association, maximizing the total utility. The optimization problem can be formulated as

$$\begin{aligned}
&\max_{\{x_{ij}[t]\}} \sum_{t=1}^{T}\sum_{k=1}^{K}\sum_{j\in\mathcal{B}_k}\sum_{i\in\mathcal{U}} x_{ij}[t]\log\left(r_{ij}[t]\right)\\
&\text{s.t.}\\
&r_{ij}[t] = \left(1-\frac{T_{\text{HO},i}[t]}{T_s}\right)\frac{W_j}{\ell_j[t]}\log_2\left(1+\text{SINR}_{ij}[t]\right)\\
&\ell_j[t] = \sum_{u\in\mathcal{U}} x_{uj}[t] \quad \forall j\in\mathcal{B}\\
&\sum_{k=1}^{K}\sum_{j\in\mathcal{B}_k} x_{ij}[t] = 1 \quad \forall i\in\mathcal{U}\\
&x_{ij}[t]\in\{0,1\} \quad \forall i\in\mathcal{U},\forall j\in\mathcal{B},
\end{aligned}$$

where $x_{i,j}[t]$ is the binary decision of whether UE $i$ is associated to the BS $j$ at time slot $t$, with a value of 1 meaning the corresponding association, $r_{i,j}[t]$ is defined as the effective rate achieved by the current selection, $T_{HO,i}[t]$ is the handover duration, $T_s$ is the slot duration, $l_j[t]$ is the current load on BS $j$, $\mathcal{U}$ and $\mathcal{B}$ are the sets of users and base stations, respectively.

### 3.2 Markov Decision Process Formulation

The UE-BS association problem has characteristics of a typical Markov Decision Process (MDP) where the central controller acts as an agent to decide on the UE-BS associations at each time step. The goal is to have tradeoff between the throughput versus the number of handovers via maximizing the total utility by selecting the optimal UE-BS associations, which appears to require an optimal policy that can be generated by an RL algorithm.

### 3.2.1 Sequential Nature

User movement and user-base station associations affect the RSRP values between each UE and BS and therefore, affect the SINR and the total reward. Based on this reward, the RL agent learns to update its policy and generate a new UE-BS association decision rule. The agent learns about the link failure probability and based on the current SINR mesaurements for each UE-BS association, decides to take an action that increases its reward. Having a behavior of learning the link failure probability and enforcing specific users to have more handovers implies the agent's ability to balance the link failure risk with the individual user's behavior such as their motion, reflecting a long term reward maximization. Such an association action not only affects the immediate reward but also affects the states (SINR values) and future rewards (throughputs) by balancing the link failure risk and actions, which makes it a sequential problem.

### 3.2.2 State Space

The state space for the paper being replicated in this project consists of the current UE-BS SINR measurements of size $|\mathcal{U}| \times |\mathcal{B}|$ and the current BS load of size $|\mathcal{B}| \times 1$, denoted by $\mathbf{\Gamma}[t]$ and $\boldsymbol{\ell}[t]$, respectively. This yields an increasingly large matrix with the number of customers. Each UE-BS association SINR is a real number, necessiating the use of either the function approximation methods or deep learning methods that can handle such state spaces. The state space is then given by

$$S[t] = (\mathbf{\Gamma}[t], \boldsymbol{\ell}[t]).$$

The UE locations and velocities are only partially observable through the received SINR and load values. The agent can infer some information from the current SINR measurements coming from both BSs to decide on the best action.

### 3.2.3 Action Space

The action space consists of the UE-BS associations at the next time step as a matrix of dimensions with the number of UEs and BSs, $|\mathcal{U}| \times |\mathcal{B}|$. The action is a discrete action to be decided by $x_{i,j}$ as defined in the optimization formulation, or can be selected to be a vector of $|\mathcal{U}| \times 1$ dimension, showing the index of each BS selection at each dimension.

### 3.2.4 Reward Function

The reward function is defined as the sum-log-rate capturing the opportunistic nature and fairness tradeoff between different users. This throughput measurement favors rewarding users with low data rates and saturates for large data rate users. The reward is affected by the long time it takes to handover. During the handover time, there is no data transmission, leading to a decreased average rate. The reward can be defined as

$$R[t+1] = \sum_{j \in \mathcal{B}} \sum_{i \in \mathcal{U}} x_{ij}[t] \log \left( r_{ij}[t] \right),$$

however, the reward can also be defined as a Gaussian shape or a combination of the Gaussian and linear functions, as will be discussed throughout the report. For the Gaussian scenario, the reward function can be given as

$$R[t+1] = \sum_{j \in \mathcal{B}} \sum_{i \in \mathcal{U}} A x_{ij}[t] \mathcal{N}(r_{ij}[t]; \mu, \sigma^2),$$

where A is a scaling parameter, $\mathcal{N}$ is the Gaussian distribution, $\mu$ and $\sigma^2$ are the mean and variance of the Gaussian distribution. For this rewarding scheme, there are three parameters to tune for a good behavior, which can be helpful in achieving a more controlled tradeoff between the achievable rate versus the number of handovers.

## 4 Dataset Generation

To generate the dataset, we use QuaDRiGa, a simulation software that generates channel matrices via ray tracing in a custom environment (9). This is the same tool used in (1), however, not all the environment parameters are given in (1) and therefore, it is not very easy to replicate the results. Therefore, the environment setup is unique to this project. This process of generating a dataset from scratch, that was compatible for a given gym environment was a particularly useful process beyond what was taught in class, as there are no existing datasets that represent the environment we wanted to model. Creating a custom dataset also provided flexibility to alter the environment, using different UE trajectories and antenna setups. For static BSs, a set of RSRPs can be generated from a QuaDRiGa layout defined in MATLAB, and then the individual SINR can be computed for each UE-BS association as done in (1). For the replication of the work in (1), we can produce a single episode of moving UEs' RSRPs and can learn a policy by changing the current UE-BS associations, similar to the planning introduced in (10). However, to prevent overfitting to a particular set of channel realizations, we generated various channels for the learning algorithm to generalize its learning better.

### 4.1 Environment Layout

The simulated wireless environment includes 12 unique BSs. This consists of two different sites, positioned at the coordinates (-100, 0) and (100, 0) m with a height of 25 m. Each site supports three sectors and two frequency bands at 2500 MHz and 700 MHz. Additionally, we created 13 rectangular drive routes around the BS sites, as depicted in Fig.2, which represents the scenario of a planned city with a regular road network (11). This base environment setup is the same for all channel realizations. However, different realizations of this environment are represented by randomly initializing start position along one of the 13 tracks, velocity, and direction for 120 different UEs. UE antennas are modeled as vertically polarized omni-antennas. QuaDRiGa then uses this environment to generate channel matrices for each BS-UE pair at 100 ms time intervals. As the UEs move around their respective tracks, channel matrices are generated for 900 timesteps, which serve as the data for one episode in the RL problem. These channel matrices are then used to create csv files that can be processed by the gym environment provided by Gupta to generate episodes.

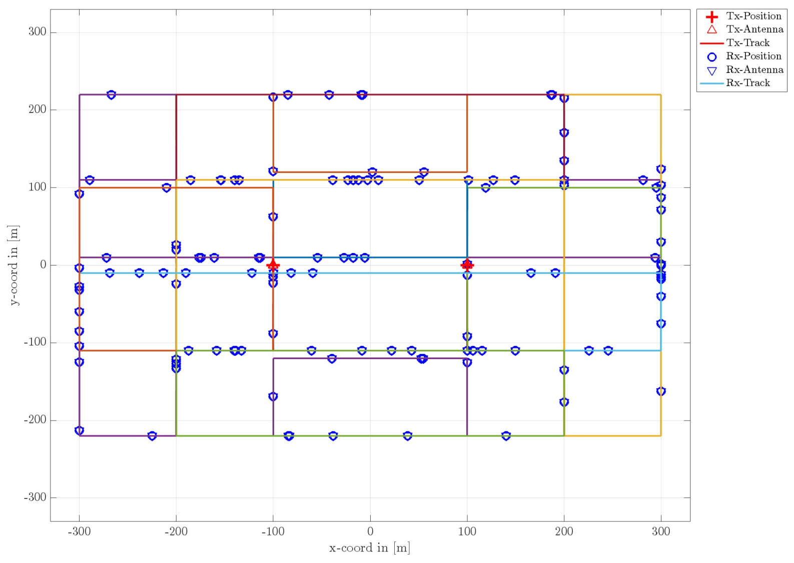


Figure 2: Wireless environment and UE drive routes

### 4.2 Environment Differences From Literature

There are two primary differences between the dataset used in (1), and the dataset generated for this project. Most obviously, there are more track options that UEs can be assigned to; 6 additional routes are added, primarily ones that are longer in the x dimension. This is meant to create a more complex and realistic environment where a greater number of handovers are expected. Additionally, this project aims to evaluate performance across multiple antenna environments, so two different antenna configurations are used. The first environment consists of a 10x1 antenna configuration on the transmitter, which has 10 elements in elevation and 1 in azimuth. This is likely a similar setup as proposed in (1). The second environment consists of a 2x2 antenna on the transmitters, containing 2 elements in both elevation and azimuth. This results in higher selectivity along the azimuth, meaning that we would predict more handovers to be required, and for the learning algorithm to have to learn more information in order to achieve good performance. In both cases, the antennas are 3GPP model antennas that are tilted 12 degrees downward, which is standard for modeling an Urban Macrocell environment representing a city. A common dataset of 30 different channel realizations (based on different randomized starting states) is used for all experiments involving the 10x1 environment, and a dataset of 26 channel realizations is used for experiments involving the 2x2 environment.

## 5 Learning Policies for Solving The RL Problem

Similar to (1) being replicated, a deep Q-network (DQN) can solve this UE-BS association problem under the high dimensional continuous state space. To replicate the results, a DQN is implemented in this project by also tuning the parameters and also a brief guideline for tuning the DQN is also included. However, using the entire state space of size $|\mathcal{U}| \times |\mathcal{B}|$ and individual loads per each BS in the learning can slow down the learning procedure and may not be plausible sometimes. For some scenarios with challenging environments, it may be harder for the agent to learn within the episode time and available dataset. For example, when the link failure probability is itself changing and the antenna selectivity is high along the serving UE direction, it may be harder for the agent to learn the optimal policy as there are more things to learn.

Lowering down the state space to cover only a user at a time by partitioning the state space into $|\mathcal{U}|$ different state spaces can be very useful in learning. When the dataset is limited and the environment is challenging to learn from, such a simplification can help the agent learn a policy that performs worse than optimizing all the users jointly, however, it can learn a better policy in the short run and can be more robust to the challenging conditions. For the policy exploration, an $\epsilon$-greedy method with a decaying $\epsilon$ is considered in this project. The decaying $\epsilon$ also means that there needs to be enough exploration before the decay ends and therefore, learning fast can be important in such challenging environments. However, simply increasing the learning rate to achieve a faster learning can damage the learning stability. Hence, lowering down the state dimensionality can help with the faster learning requirement in challenging environments without increasing the learning rate. For such a high dimensional state, the function approximation methods described in (10) cannot help as the feature weight vector length increases with the number of dimensions, and even $\mathcal{O}(N^2)$ in some scenarios.

### 5.1 Deep Q-Network

DQN estimates the state-action value function $\mathcal{Q}$ leveraging a neural network and then uses the estimated $\mathcal{Q}$ function to learn in a Q-learning setting, hence the name DQN. DQN can be helpful to solve such a high dimensional problem as it is able to perform well with the raw data as it is used in (1), and can work with a high dimensional and continuous state spaces. For the wireless network system introduced in this project with multiple UEs, a DQN architecture can learn from the raw SINR values and the current load to achieve a high level performance. DQN can also translate the different UEs raw data to learn a policy for UEs with different trajectories. DQN also leverages an experience replay, where the experience so far is recorded in memory to sample at each time step to update with a Q-learning scheme as mini batches of updates between the time steps. This idea is similar to the planning discussed in (10) but with sampling from the experience so far. The $\mathcal{Q}$ function can be given as

$$\mathcal{Q}^{\pi}(S[t], \mathbf{a}[t]) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^{k} R[t+k+1] \mid S[t], \mathbf{a}[t]\right],$$

where $\gamma$ is the discount factor and $\mathbf{a}[t]$ is the action vector. The Bellman optimality condition is satisfied and the optimal policy is achieved (10) when

$$\begin{aligned}\mathcal{Q}^{*}(S, \mathbf{a}) &= \mathbb{E}\left[R_{t+1} + \gamma \max_{a'} \mathcal{Q}^{*}\left(S_{t+1}, a'\right) \mid S[t], \mathbf{a}_{t} = \mathbf{a}\right] \\ &= \sum_{S', r} p\left(S', r \mid S, \mathbf{a}\right)\left[r + \gamma \max_{\mathbf{a}'} \mathcal{Q}^{*}\left(S', \mathbf{a}'\right)\right].\end{aligned}$$

To support the sequential characteristics of the agent so that it can learn for the different UE trajectories and act accordingly, a long short term memory (LSTM) is included in the system. From the observability standpoint, such a memory should allow the agent to estimate the individual UE trajectories, given that the BS locations are known to the controller. Hence, it becomes possible to learn for specific UE trajectories and react accordingly, differentiating it from the simple combinatorial optimization problem. For this purpose and to reproduce the results of (1), this project uses the same DQN structure with tuned parameters, as can seen in Fig. 3.

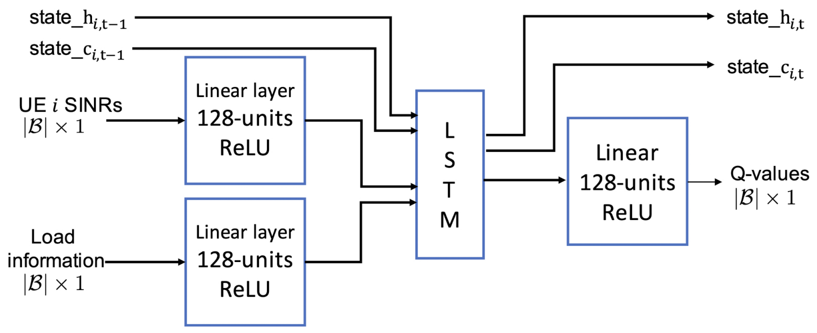

state_h$_{i,t-1}$
state_c$_{i,t-1}$
UE $i$ SINRs $|\mathcal{B}| \times 1$
Linear layer 128-units ReLU
Load information $|\mathcal{B}| \times 1$
Linear layer 128-units ReLU
L S T M
state_h$_{i,t}$
state_c$_{i,t}$
Linear 128-units ReLU
Q-values $|\mathcal{B}| \times 1$

Figure 3: DQN Block Diagram

Each UE SINR measurement and BS load are fed into the system together with the previous state and rate with the load data. These are passed through the LSTM to generate estimations for the next state and $\mathcal{Q}$ function so that the estimation of the next time step's parameters are connected to the previous states and therefore, the system can estimate the individual UE SINR changes.

### 5.2 Function Approximation

One of the initial goals of this project was to evaluate the performance of function approximation methods such as tile coding, using the same environment as the DQN in order to compare performance of these different methods. However, as mentioned above, the dimensionality of our state space is $|\mathcal{U}| \times |\mathcal{B}|$, which in practice comes to the intractably high value of 1440. Most function approximation methods cannot handle a state space this large, which we quickly realized after a failed attempt at extending standard one-hot tile coding as described in (10) to this dimension. Although the code could theoretically extended itself to arbitrary dimensionality, the matrix operations required to actually compute updates to the one-hot state representation were infeasible from a computational and memory perspective. Additionally, even if we managed to compute this representation, the matrix would be so sparse that there would simply not be enough data from past exploration to accurately estimate values of new states. The only function approximation methods in the literature that could handle a state space of this size, still involved the use of a deep network somewhere in the RL pipeline, such as (12), which effectively uses a deep network to create a lower dimensional tile coded vector. Instead of continuing on this function approximation path, that would have essentially just required another deep network to train, similar to the DQN that was already being used, we decided to focus on ways to lower the dimensionality of the state space used to train the DQN to improve effeciency and performance, particularly in the 2x2 environment.

## 5.3 Independent Agents

In the original DQN, SINR measurements between all UE-BS pairs are used to represent a state, which can be difficult to learn in more challenging environments such as the more selective 2x2 antenna environment, even for a deep network. In this scenario, it would be beneficial for the Q network to be able to train more frequently, and with lower dimensional states. However, in order to maintain the realistic nature of this model, these lower dimensional states should only contain features that the central controller would have access to, meaning that they still had to consist of RSRP values between UE-BS pairs, or values directly computable from this information. The method proposed here that achieves both of these desired constraints is based on the multi-agent learning strategy proposed in (13), which uses action-dependent features to locally evaluate possible actions for a given agent.

For this formulation, the central controller treats each UE as a separate agent that acts independently and without knowledge of what actions the other agents are taking. While it is not entirely true that taking the optimal action for each user will be optimal for the system as a whole, it is expected that this will be a relatively safe approximation because in any given timestep, few handovers are expected to occur. The Q function and overall network architecture remain the same as in section 5.1, however the state space, action space, and training process are updated.

### 5.3.1 Independent Agent State Space

The new state space for the independent agent case includes SINR measurements to each BS from a single UE of size $|\mathcal{B}| \times 1$ and the current BS load, also of size $|\mathcal{B}| \times 1$, denoted by $\mathbf{\Gamma}'[t]$ and $\boldsymbol{\ell}'[t]$, respectively. This state representation does not grow with the number of customers since it is only concerned with describing the state from a single UE. The updated state space is then given by

$$S'[t] = (\mathbf{\Gamma}'[t], \boldsymbol{\ell}'[t]).$$

### 5.3.2 Independent Agent Action Space

The action space consists of the BS association at the next time step for the given UE. This is represented as a single integer in (1, 12) indicating the index of the selected BS.

### 5.3.3 Updates to Training Process

In the original DQN, for each timestep, the Q function is sampled, an action selected, and a reward generated along with the next state representation. In this version, the Q function is sampled for every timestep and for every UE. Actions for each UE are independently selected before returning any rewards. The next state for each UE is then generated based on this action. After all UEs have selected their actions, the same reward, representing the sum-log-rate of the entire system after this timestep (or shaped version of this reward) is provided to all agents. In addition to the smaller state and action spaces, this has the benefit of the DQN experiencing more states for the same number of timesteps, proportional to the number of users in the system.

## 6 Results and Discussion

Not all the parameters were given in (1), affecting its reproducibility (14) in the case of differences in environments and datasets . For this project's reproducability, all the relevant parameters are included in Table 1. One can find the remaining parameters in the dataset generation code for the reproducibility. However, some of the parameters are changed for the different environments and learning algorithms. These are removed from the table as they are changing for each application but they are given within each relevant subsection. Note that due to the use of experience replay, even a seemingly small number of episodes such as 20, can actually still provide enough information for the agent to learn effectively, because the number of steps that agent sees within an episode increases with each episode since there is more experience stored in memory. While using more episodes does yield better peak performance, we found that an episode length of 20 yielded good results, while balancing the runtime of the code so that we could run multiple experiments multiple times each, within the timeframe of this project. However, the number of training episodes were increased up to 360 whenever a significant change in the policy was observed within the training episodes.

| | |
|---|---|
| Batch Size | 16 |
| Learning Rate | $1 \times 10^{-3}$ |
| Discount Rate $\gamma$ | 0.99 |
| Optimizer | Adam |
| Number of Episodes | 20 |
| $\epsilon$ Start | 0.1 |
| $\epsilon$ End | 0.001 |
| $\epsilon$ Decay Rate | 0.995 |

Table 1: RL Parameters

### 6.1 Baseline Algorithms as Max SINR and Max RSRP

The baseline algorithms used are fairly simple and greedy. They can be useful in different environments, with some potential drawbacks on the number of handovers. The Max SINR chooses the maximum UE-BS SINR and Max RSRP chooses the associations with the maximum UE-BS RSRP. These algorithms do not account for the handover duration or the link failure probability, possibly showing the good sides of leveraging a learning algorithm for such purposes. It is therefore expected to see improvements in the rate and handover tradeoff performance in the learning algorithms, as discussed below.

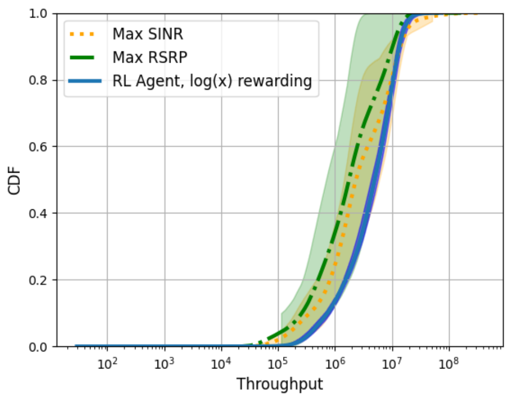
### 6.2 10x1 Antenna Environment, Different Rewards

For the 10-by-1 antenna configuration in the BSs, the antenna selectivity is fairly low, meaning that it is possible to have an acceptable performance with a smaller number of handovers, compared to a more selective antenna configuration. First, the log(.) sum utility was analyzed and then, the reward was shaped to have a similar performance with smaller number of handovers.

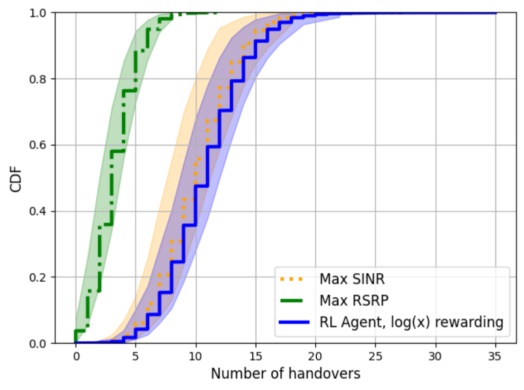
#### 6.2.1 log(x) Reward Function

As it is widely used in communication networks and was also used in (1), this rewarding scheme was analyzed first. The rate and handover CDFs are plotted in Fig. 4 with 50 different realizations over the mean, showing how variable the different algorithms are.

From Fig. 4, it appears to be the case that the RL agent is achieving the best possible results that can be achieved by the baseline algorithms, with much less variance. However, in order to achieve such a persistent rate CDF, the RL agent is using more handover than the other algorithms. It appears to be using visibly more number of handovers compared to the max RSRP baseline and



(a) Rate CDF



(b) Handover CDF

Figure 4: Rate and Handover CDFs for log(.) Reward Function, 10-by-1 Antenna Environment

slightly more number of handovers compared to the max SINR baseline. This is different from the literature findings, where the RL agent was performing similarly in the rate CDF, however, with a number of handovers between the max RSRP and max SINR baselines.

The difference in the handover CDF with the literature could have been caused by the fact that we generated our own dataset with our own parameters after some point. Therefore, the changes in the environment, such as the antenna shape, could have resulted in the difference. Within simulations, it was observed that when different trajectories are also added into the system, the behavior stayed similar, the DQN was able to generalize the similar behavior into a different set of trajectories.

### 6.2.2 Gaussian Reward Function

After observing the rate CDF shapes, different reward functions were tested to shape the rate CDF and possibly the handover CDF accordingly. For that reason, the Gaussian rewarding scheme was introduced with the individual gain, mean, and variance. The mean was adjusted to reward the agent more for their action in the correct direction. The potential problem with the log(.) type rewarding is that even though it has a diminishing returns in terms of the rates, it is also favoring the UE-BS scenarios with larger data rates, possibly causing more handovers than needed. To overcome this issue, a Gaussian reward shape was generated to reward the middle rates more to have a less number of handovers. Fig. 5 depicts the Gaussian reward function tuned to have the desired smaller number of handovers with reward function gain, mean, and variance tuned to be 1000, $10^7$, and $3 * 10^6$

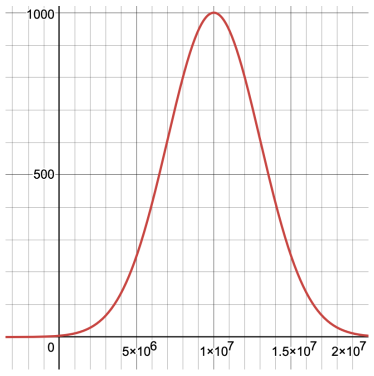
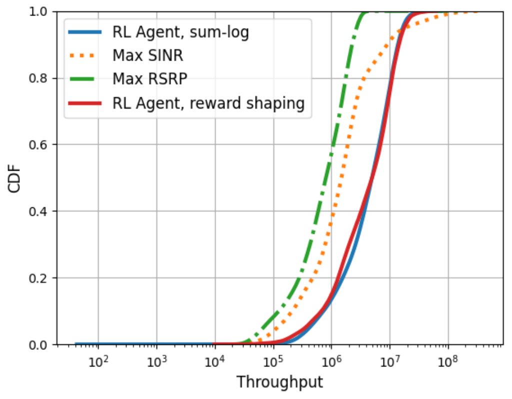
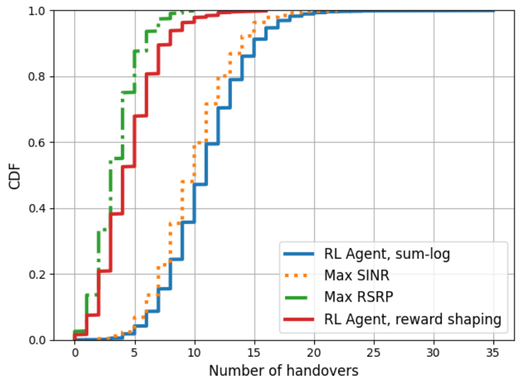


Figure 5: Gaussian Reward Function

With the Gaussian rewarding, one must beware that the tail of the Gaussian distribution becomes flatter as the argument of the function gets farther away from the Gaussian's mean. This effect can be seen in Fig. 6 as the reward not being able to push the rate CDF to the right for the portions where it coincides with the Gaussian reward function with a flat tail. However, we see a useful behavior in the number of handovers, which is closer to the max RSRP scenario now. The RL agent is now able to perform similar to the log(.) type reward case with a notably smaller number of handovers.



(a) Rate CDF Realization



(b) Handover CDF Realization

Figure 6: Gaussian Reward Function and A Sample Realization of CDFs

As a final step in the reward shaping, the Gaussian's flat tail problem was fixed by generating a piecewise defined function for the flat portions causing the trouble. The flat portions of the Gaussian were made to be linear so that the UEs with lower rates are pushed to have a larger rate meanwhile the agent is not pushing towards getting the highest rate possible, possibly reducing the number of handovers. The Gaussian-linear piecewise reward function in Fig. 7 achieves these requirements for a linear piece with slope $10^{-4}$ for rates smalles than 1 Mbps.

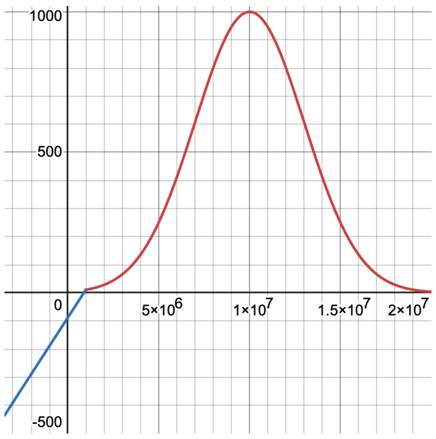
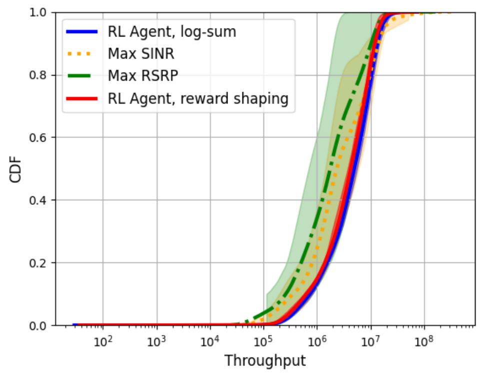
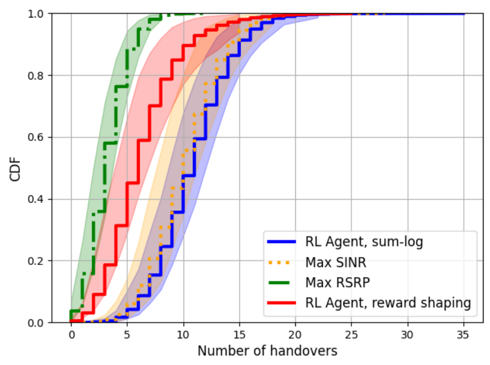


Figure 7: Gaussian-linear Reward Function

When the reward-shaped version is run with 20 different inputs, the outputs in Fig. 8 show that the rate CDF is still performing very similar to the log(.) type rewarding, with a notably smaller number of handovers. Such decreased number of handovers could be very useful when the environment's link failure probability is higher.



(a) Rate CDF Realization



(b) Handover CDF Realization

Figure 8: Gaussian-linear Reward Function's Rate and Handover CDFs

### 6.2.3 Tested Reward Functions

Before settling down with the Gaussian-linear combination, multiple different reward functions were tested, many of which failed to converge or yielded bad rate CDF performances. A drawback of the Gaussian rewarding scheme is that it requires a knowledge about where to put the mean and how wide the variance should be. Various simulations were run to determine that if the Gaussian mean is unrealistically selected to be a very high or very low rate, the system performs poorly. This appears to happen due to the geometry of the problem. Given a UE location and current state, that UE can only be served a specific set of rate values. It can either be something high or something very low.

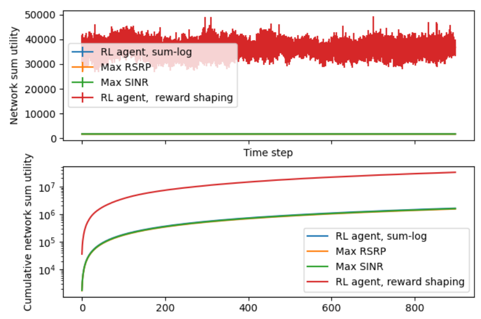
The linear rewarding scheme for the rate does not converge at all. Moreover, the triangle function shape to be a representative shape of the Gaussian was observed to perform poorly in the convergence to a reasonable policy. When the variance of the Gaussian scheme was selected to be very small, the agent does not know how to develop its policy when the current state of the UE happens to be on the tail of the Gaussian, which is very close to 0 and the reward barely changes with the actions. Hence, even though the Gaussian-linear case appears to be performing the best, it requires a knowledge about where to expect the rates and possibly the geometry of the environment.

### 6.2.4 Reward Alignments

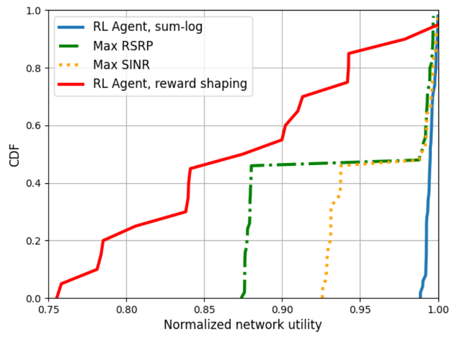
After the training, the resultant rewards during an episode was observed to follow a similar pattern for the different rewarding schemes, increasing until the end of the episode in time. However, comparing the reward alignment for the different rewarding schemes are not very straightforward as each reward function is valuing different rates differently. Especially for the Gaussian shaped reward function, some of the rewards are praised much more than the others. Therefore, the network sum utility for different reward functions in Fig. 9a only show that both rewards keep increasing in time.

The reward alignments can be compared by checking how many of the realizations are in line with the reward shaping. Fig. 9b presents the sum utility CDFs for the different methods followed, which are generated from 20 different realizations. It can be seen that the log(.) type rewarding schemes are behaving in line with the paper being replicated **??**, with max SINR and max RSRP methods acting differently due to the small number of realizations. It can be observed that the reward shaping method is achieving the worst in terms of the network utility CDF. This might have been caused by a small Gaussian reward function variance so that the different rewards are rewarded notably different between the different runs. Hence, it is indeed an expected behavior and might be an important metric while tuning a better reward function with the Gaussian rewarding scheme. One can generate another Gaussian that alignes better with what the agent is capable of, which may result in a better and faster alignment with the reward function. For the large rates, the log(.)

reward does not change its value a lot, generating a tighter gaps between the plots and making it harder to compare.



(a) Network Sum Utility vs. Time



(b) Normalized Network Utility CDF

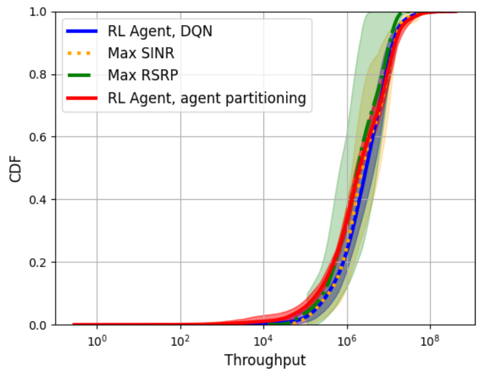
Figure 9: Reward Alignment Comparisons of log(.) and Gaussian-linear Reward Shaping

### 6.3 Standard DQN vs. Independent Agents in 10x1 and 2x2 Environments

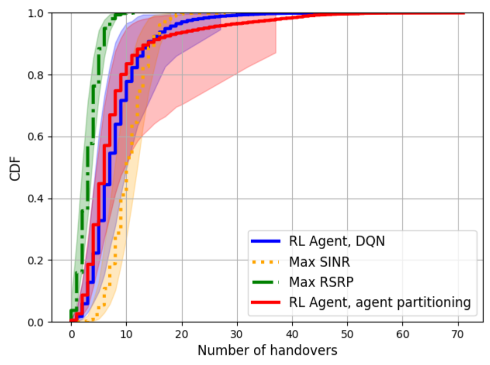
The previous section is able to show improved performance compared to the literature using reward shaping in the relatively easy to solve 10x1 antenna environment. This section explores using the independent agent paradigm described in section 5.3 to approach this same environment with limited performance drops, and more importantly to apply it to the more difficult 2x2 antenna environment, which the standard DQN cannot solve on its own without modifications.

#### 6.3.1 10x1 Antenna Environment

The same environment used in the original DQN with the 10x1 antenna is tested using the independent agent paradigm. Results shown in Fig. 10 indicate that the performance using this paradigm is slightly degraded when compared to the model that encapsulates the entire system within its state space. This makes sense because the independence assumption is a bit of an oversimplification, so it is not possible to reach peak performance that a full state representation does with manual tuning. However, the fact that it still performs relatively well in this environment may still make this paradigm a feasible option in this environment, considering the environmental robustness benefits outlined in the next section



(a) Rate CDF Realization

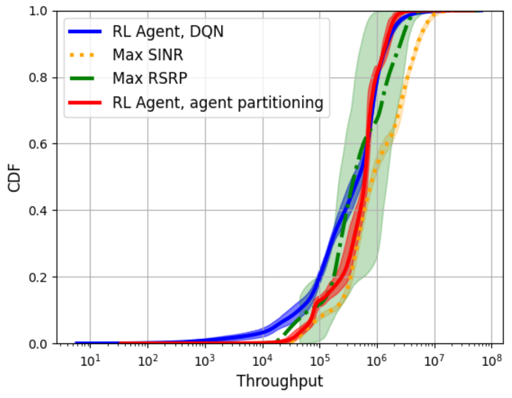


(b) Handover CDF Realization

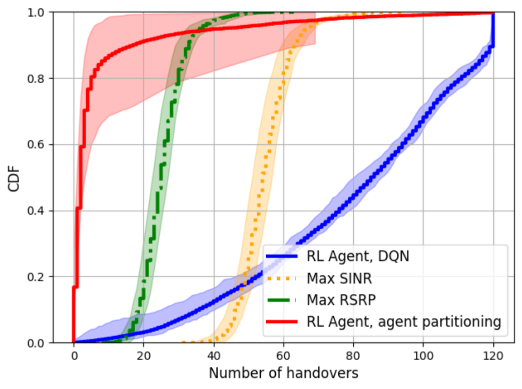
Figure 10: Rate and Handover CDFs for 10x1 Environment Using Independent Agents

### 6.3.2 2x2 Antenna Environment

The primary benefit for the independent agent paradigm became apparent in the 2x2 antenna environment. As this environment is more difficult to learn, it required the DQN parameters to be separately tuned and tested manually, or for the algorithm to run for a greater number of episodes to achieve acceptable performance. However, since the independent agent paradigm employs more frequent sampling of the Q network, this version is able to learn more quickly, and using the same training setup as the 10x1 environment, is able to achieve significantly improved results. This is beneficial because it allows for transfering models between environments without having to retune parameters, and only having to retrain the same model in the new environment. As seen in Fig. 11, the performance of the standard DQN agent (using parameters that worked in the 10x1 environment) is degraded significantly for low-rate users in the 2x2 environment. Additionally, an unhealthy number of handovers are used, indicating that the model likely did not converge within the given number of episodes. Alternatively, when the agents are partitioned by user, the low-rate user performance is improved, and the number of handovers reduces significantly, even compared to the Max SINR and Max RSRP baselines.



(a) Rate CDF Realization

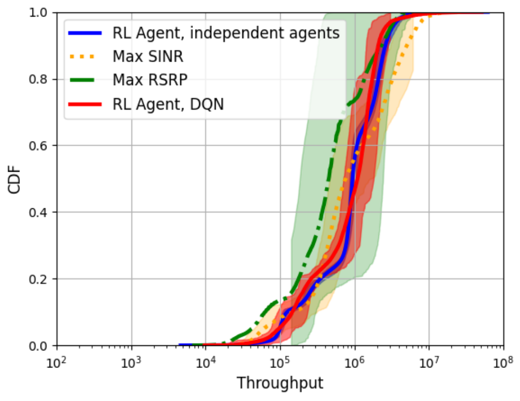


(b) Handover CDF Realization

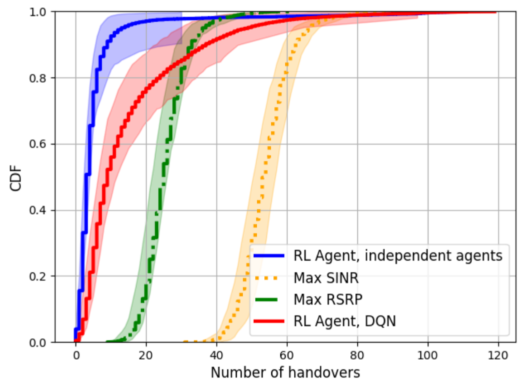
Figure 11: Rate and Handover CDFs for 2x2 Environment Using Standard DQN and Independent Agents

We also evaluate the performance of both the standard DQN and independent agent paradigm with more training episodes, in order to provide a more fair comparison to a DQN that does converge. For this training setup, instead of training for 20 episodes, the standard DQN runs for 360 training episodes, while the DQN in the independent agent paradigm runs for 200 episodes. Here, we see that both paradigms are able to perform relatively well compared to the Max SINR and Max RSRP baselines. Even though we initially expected that with additional training time, the standard DQN would begin to outperform the independent agent version, this is only the case in certain data rate regimes. Additionally, the independent agent version is able to maintain lower throughput variance, and achieve this similar overall performance with fewer handovers. This indicates that the 2x2 environment is even more difficult to solve than initially anticipated, and that the independent agent implementation is likely the better solution for this scenario. Another thing to note about the throughput CDF in the 2x2 environment is that all methods seem to produce a less smooth curve. This is likely due to the geometry of the system, where the selectivity would have us observe very high or very low throughput values much more frequently than moderate throughput values. With less observations in this moderate range, it becomes more difficult to arrive at a smoothed CDF.

In the 2x2 environment, we see that using the independent agent paradigm provides performance improvements to the standard DQN, and does not provide significant performance reduction in the 10x1 environment. However, even in the 10x1 environment, the ability to adapt more easily to new



(a) Rate CDF Realization



(b) Handover CDF Realization

Figure 12: Rate and Handover CDFs for 2x2 Environment Using Standard DQN and Independent Agents (Long Training Period)

environments with less training time is a benefit that may be worth the tradeoff in some scenarios. Additionally, the significantly reduced dimensionality of the state space could potentially lend itself to faster non deep learning methods, which as described in section 5.2, are computationally infeasible in the original paradigm.

### 6.3.3 Network Sum Utility Comparisons

Different from the reward shaping scenario, comparing the different learning algorithms is much easier as they were both using the same reward function log(.). The reward shaping was not seen to be necessary since the number of handovers were fairly small compared to the baseline algorithms. It was observed by Fig. 13 that even with the lowered state dimensionality, the agent can learn to utilize the reward function to a similar extent, while preserving the similar performance metrics on the rate and handover CDFs. Both metrics utilize the reward function to a better extent and the Max RSRP baseline's weird behavior is due to the small number of realizations.

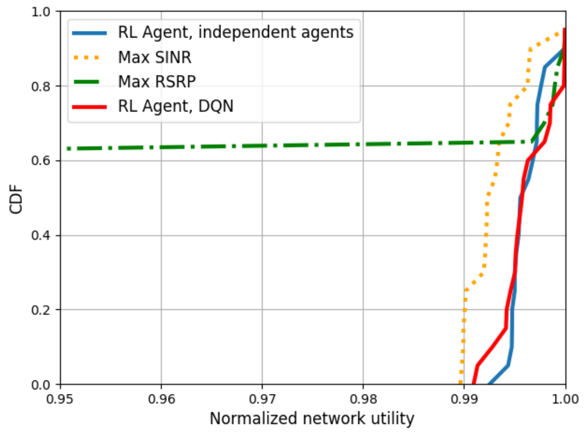


Figure 13: Network Sum Utility CDF of DQN and Independent Agents Scenarios

## 7 Conclusion and Future Directions

This project expanded work in (1), which explores optimizing handovers within a wireless network in a limited and relatively simple environment, to have better performance in a simple environment and robustness across a more difficult environment. We generated our own dataset representing a slightly more complex and realistic city environment, and modeled two antenna configuration environments. The easy to learn 10x1 antenna configuration, as was likely used in the original paper, is optimized even further to use a piecewise linear and Gaussian reward shaping function to achieve the similar performance with a smaller number of handovers. Such a smaller number of handovers could be very useful in environments where the link failure probability is higher. Additionally, as the DQN outlined in (1) is not transferable to the more selective and difficult to learn 2x2 antenna configuration, we develop a new paradigm that treats individual UE actions as independent, in order to learn a handover policy in this environment more quickly. Since reinforcement learning algorithms can be quite fragile, especially in complex scenarios such as modeling many users in a wireless network, having these more robust paradigms that also do not scale in dimensionality with number of users, are very beneficial to have.

Future work related to this study would be to test the new independent learning paradigm for handover optimization in a wider variety of environments to truly capture the level of robustness relative to DQN using the more complex and fragile state space. There could be another setting with multiple collaborative agents for each UE to learn about the different movement behaviors quicker. Additionally, it could be interesting to see how much efficiency could be improved by trying to take advantage of the lower dimensional state space using simpler non-deep learning or function approximation methods that were initially deemed infeasible with the original state space. Finally, one can also consider moving BSs, such as drones serving UEs, as an extension where the action space also contains finding the optimal drone trajectory, increasing the complexity of the system.

## References

[1] Manan Gupta, Ryan M. Dreifuerst, Ali Yazdan, Po-Han Huang, Sanjay Kasturia, and Jeffrey G. Andrews, "Load balancing and handover optimization in multi-band networks using deep reinforcement learning," in *2021 IEEE Global Communications Conference (GLOBECOM)*. 2021, p. 1–6, IEEE Press.

[2] Nguyen Cong Luong, Dinh Thai Hoang, Shimin Gong, Dusit Niyato, Ping Wang, Ying-Chang Liang, and Dong In Kim, "Applications of deep reinforcement learning in communications and networking: A survey," *Commun. Surveys Tuts.*, vol. 21, no. 4, pp. 3133–3174, oct 2019.

[3] "Study on channel model for frequencies from 0.5 to 100 ghz (release 17)," Tech. Rep. 38.901, Jan 2024.

[4] Sangwon Hwang, Juseong Park, Hoon Lee, Mintae Kim, and Inkyu Lee, "Deep reinforcement learning approach for uav-assisted mobile edge computing networks," in *GLOBECOM 2022 - 2022 IEEE Global Communications Conference*, 2022, pp. 3839–3844.

[5] Zhi Wang, Lihua Li, Yue Xu, Hui Tian, and Shuguang Cui, "Handover control in wireless systems via asynchronous multiuser deep reinforcement learning," *IEEE Internet of Things Journal*, vol. 5, no. 6, pp. 4296–4307, 2018.

[6] Yao Zhao and Xiliang Luo, "Handover mitigation in dense hetnets via bandit arm elimination," in *2019 IEEE Global Communications Conference (GLOBECOM)*, 2019, pp. 1–6.

[7] Vijaya Yajnanarayana, Henrik Ryden, and Laszlo Hevizi, "5g handover using reinforcement learning," in *2020 IEEE 3rd 5G World Forum (5GWF)*. sep 2020, IEEE.

[8] Yujae Song, Sung Hoon Lim, and Sang-Woon Jeon, "Distributed online handover decisions for energy efficiency in dense hetnets," in *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, 2020, pp. 1–6.

[9] Stephan Jaeckel, Leszek Raschkowski, Kai Borner, and Lars Thiele, “Quadriga: A 3-d multi-cell channel model with time evolution for enabling virtual field trials,” *IEEE Transactions on Antennas and Propagation*, vol. 62, pp. 3242–3256, 2014.

[10] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, Massachusetts: MIT Press Ltd, Cambridge, MA, 2018.

[11] Hang Zheng, Wei Li, Lei Tian, Chongpeng Xu, Fusheng Huang, and Jianhua Zhang, “Path loss models for urban macro cell scenario at 3.35, 4.9 and 5.4 ghz,” in *2015 IEEE 26th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, 2015, pp. 2229–2233.

[12] Yangchen Pan, “Deep tile coder: an efficient sparse representation learning approach with applications in reinforcement learning,” *CoRR*, vol. abs/1911.08068, 2019.

[13] Peter Stone and Manuela Veloso, “Team-partitioned, opaque-transition reinforcement learning,” in *RoboCup-98: Robot Soccer World Cup II*, Minoru Asada and Hiroaki Kitano, Eds., vol. 1604 of *Lecture Notes in Artificial Intelligence*, pp. 261–72. Springer Verlag, Berlin, 1999.

[14] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger, “Deep reinforcement learning that matters,” in *Proceedings of the AAAI conference on artificial intelligence*, 2018, vol. 32.

## A Appendix I: Notes On Implementation

The starting point for the implementation of this project consisted of the training and test code used in (1). This included the initlal DQN training script, a testing script, a set of utility functions containing common communications computations, and a gym environment that could be defined using csv files created in QuaDRiGa. Our additional functionality is added to this starting code, primarily with modifications and edits to the gym environment for results in section 6.2, and to the training script for results in section 6.3. The independent agents scenario was developed on top of the DQN scripts. We also updated the testing code to have clearer reproducibility, in order to produce CDFs with variance bounds. Lastly, the MATLAB code used to generate the QuaDRiGa layout for both environments was developed entirely from scratch.