# Handover Optimisation in 5G Network

Gowtham Chandrasekaran, Satya Pranavi Manthena
Department of Computer Science, San José State University, San Jose, California 95192, USA
Email: gowtham.chandrasekaran@sjsu.edu, satyapranavi.manthena@sjsu.edu

***Abstract*—In the world of pure 5G networks, the gNodeBs that were previously called antennas will be placed very close to each other due to their smaller ranges. This makes the whole deployment dense. For a dense network like this, there will be a lot of reassigning of ongoing sessions from one gNodeB to another to provide connectivity continuously. This is known as a handover in wireless cellular systems. Handovers decrease performance and consume a lot of energy. In this paper, we are aiming to optimize the number of handovers by using reinforcement learning. We are creating a 3x6 grid world environment with multiple antennas close to each other. The agent moves around randomly and learns about all the available antennas at each state. We are using the famous Q-learning method of reinforcement learning to construct a Q table after performing 5000 iterations. The antenna with the highest Q value should be chosen to optimize handovers.**

***Index Terms*—Handover, Q-learning, gNodeB, Reinforcement Learning, User Equipment.**

## I. Introduction

The latest advancement in the wireless cellular field is the inception of the Fifth-generation(5G) networks. It has drastically increased the speed and the number of simultaneously connected devices. In cellular wireless systems, mobility is achieved through the handover (HO) mechanism. Handover is the process of reassigning an ongoing session from one gNodeB to a neighbouring available gNodeB. It is depicted in Fig. 1. Since 4G networks have comparatively a larger signal radius, the handover from one node to another happens intermittently. Unlike the Fourth-generation network, 5G wireless signals are transmitted through a large number of closely located gNodeBs. So, after the deployment of pure standalone 5G networks, there will be a possibility of frequent inter gNodeB handovers (shifting from one gNodeB to another).
The current method to reduce handovers is a non-standalone deployment which has a mixture of 4G and 5G nodes with 4G nodes in the majority. Once a pure 5G stand-alone environment is deployed, the handovers are bound to happen more frequently which causes data connectivity problems since, during a handover, the user equipment loses connectivity from one gNodeB and gets connected to the neighboring gNodeB. This being a real-time problem, we would not have a training data set to make the agent learn. Instead, the agent has to interact with the environment to collect information. Therefore, we use reinforcement learning techniques to optimize the number of handovers which in turn maintains hassle-free connectivity with lesser number of overall handovers. Handovers also cause the ping pong effect. It is the handover happening to and fro between two gNodeBs frequently when the user equipment is present in the range of two gNodeBs. This problem is also addressed by optimising the handovers using reinforcement learning.

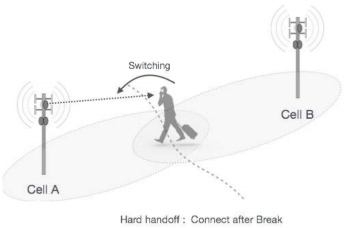


Fig. 1. Illustration of Handover

## II. Formal Problem and Evaluation Criteria

Reinforcement Learning system primarily consists of an environment, agent, state, action, reward. Fig. 2. shows the framework of an RL system. To formulate the problem of providing a stable connection with minimum disruption of connectivity we have considered a two dimensional grid world environment which acts as the real world path in which the agent takes a random walk between any two points in the grid. The grid world consists of multiple gNodeB's such that every coordinate point in the grid will have access to signals from at least one gNodeB. The state space of our system would be the coordinate points of the grid world, Example : (0,0), (2,0), etc. Since we consider a random walk policy the action space would be right, left, up, down. Action space = [r, l, u, d]. Our aim is to minimize connection disruption which is achieved by minimizing handovers. The agent learns to avoid handovers by following the reward function. The reward would be -1 whenever a handover takes place, else it would be +1 whenever there is no handover (the agent is connected to the same antenna). Minimum number of handovers gives us the maximum reward.

```
if <change of antenna>:
    reward = −1   (Handover)
    handover = 1
else:
    reward = 1   (No Handover)
    handover = 0
```

We maintain a variable that stores the count of handovers at each episode and keep monitoring it for the reduction of handovers.

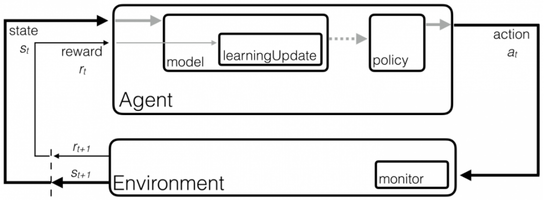


Fig. 2. RL Framework

## III. Methodology

The main idea is that the agent walks through the grid world moving from point A to point B. There are multiple closely spaced gNodeBs along the way the agent is traveling. In a pure 5g network, the gNodeBs have to be placed close to each other because of their small range. The agent performs a random walk and sees a lot of available gNodeBs to connect to. It has to connect to the best available gNodeB and keep moving towards the goal. Agents can then distill the results from planning into a learned policy. To achieve our goal we should find the optimal action-selection policy, since our agent performs a random walk, we have to apply the optimal action-selection policy on the antenna which the agent has to connect to reduce handovers.

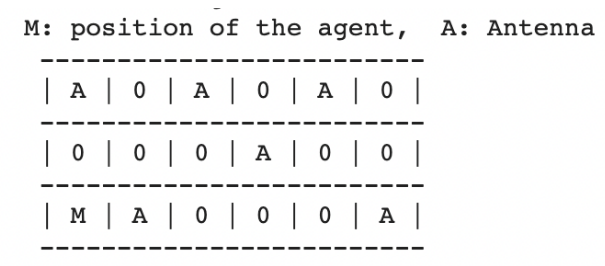
We use Q-Learning to optimize our antenna selection. Q-Learning is a value-based reinforcement learning algorithm that is used to find the optimal action-selection policy using a Q function. The value function Q should be maximized as much as possible. The Q table assists us in determining the appropriate course of action for each state. It aids in maximizing the expected payoff by allowing you to choose the best of all available actions. The expected future payoff of that action at that state is returned by Q(state, action). This function can be approximated using Q-Learning, which uses the Bellman equation to iteratively update Q(s, a). We begin by investigating the environment and updating the Q-Table. The agent will begin to exploit the surroundings and take better actions once the Q-Table is ready. The Epsilon greedy strategy concept comes into play next to explore and exploit the environment. In the beginning, the epsilon rates will be higher. The agent will explore the environment and randomly choose actions. This occurs logically since the agent does not know anything about the environment. As the agent explores the environment, the epsilon rate decreases and the agent starts to exploit the environment. One of the advantages of Q-Learning is that it can compare the expected utility of various actions without the need for a model of the environment. Reinforcement Learning is a method of problem-solving in which the agent learns without the assistance of a tutor. In a reinforcement environment, the agent's main signal for learning from his actions is the so-called reward, a number that tells him if his last action was good (or) not. Q-Learning is a type of Reinforcement Learning algorithm that doesn't require a model of the environment and may be utilized in real-time.

**Algorithm 1** Q-Learning

```
Q($s$,$a$) initiliazed arbitrarily
for each episode $i$ do
  State $s$ initialized
  for each step $j$ do
    action $a$ from state $s$ using policy derived by Q
    action $a$ taken
    reward $r$ and state $s'$ observed
    $Q$(s,a) $= Q$(s,a) $+ \alpha[r + \gamma max_{a'} Q$(s',a')-$Q$(s,a)$]$
    $s$ = $s'$
  end for
end for=0
```

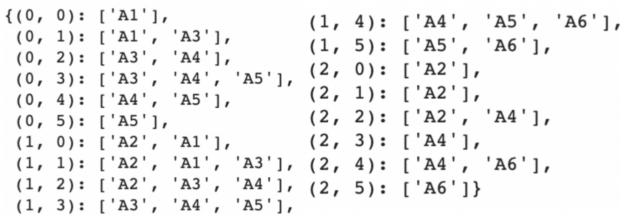
## IV. Evaluation Settings

We would need a real-world environment with 5G antennas and mobile phones to implement our challenge. To simulate such a scenario, we constructed a 2D grid world with a 3 x 6 grid size that can be understood as a real-world path, and a cell phone serves as our RL agent. Multiple antennas are positioned across the path in this 3 × 6 grid, ensuring that the agent has access to at least one antenna no matter where the agent is as seen in Fig. 3. We determine the signal availability for each coordinate in the grid first, and we assume that our antenna's signal intensity is 1 in all directions, including the diagonals as seen in Fig. 4. Because we're using a random walk strategy, our agent starts at coordinate (2,0) and can do any action from our predetermined action space. The agent must choose an antenna to connect to from its signal availability vector in addition to picking an action after each step. When the agent takes a step, our goal is to optimize the antenna selection. As previously said, we will use Q-Learning to achieve this goal, which means there will be a Q-table that will be updated after each step using the Bellman equation based on the action taken by the agent during its exploration phase. We keep track of the epsilon value that degrades as the exploration progresses to leverage our antenna selection. We set epsilon to 0.82 and set the number of steps done by an agent in one episode before resetting to the start state to 150. If the value of np.random.rand() is less than 0.82, the agent chooses an antenna at random and goes forward; otherwise, the agent chooses the best antenna at that particular state. In this case, the Antenna with the highest Q-value for that coordinate is chosen using the signal availability vector and Q table. If the antenna we choose is the same as the one we already have connected to, there will be no handover between the gNodeBs, and the reward will be +1; otherwise, the reward for a handover will be -1. We created a negative reward for handover to encourage the agent to learn and seek the highest possible reward by going to the destination with the fewest possible handovers. Following the previously established Q-learning method, the agent first picks an action, after which an antenna is selected depending on available antennas at that state. The Q-table updates at every episode. We are keeping track of the accumulated rewards at each episode. We also define a Handovers counter to determine whether or not the amount of handovers is being optimized. To ensure that the agent learns successfully, we now run the process for 5000 episodes with 150 steps every episode. Because this is a random walk experiment, there is no stable visualization to show that the agent has reduced the number of handovers. Instead, we evaluate our algorithm based on the Q-table values. If the values of the Q-table are updated suitably, it means that this policy can be applied to any fixed path and the agent performs with fewer handovers than before.



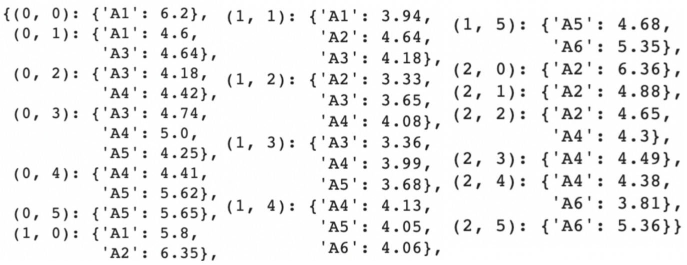
```
M: position of the agent,  A: Antenna
 -------------------------
 | A | 0 | A | 0 | A | 0 |
 -------------------------
 | 0 | 0 | 0 | A | 0 | 0 |
 -------------------------
 | M | A | 0 | 0 | 0 | A |
 -------------------------
```

Fig. 3. 3x6 Grid World- Experiment settings



```
{(0, 0): ['A1'],                  (1, 4): ['A4', 'A5', 'A6'],
 (0, 1): ['A1', 'A3'],            (1, 5): ['A5', 'A6'],
 (0, 2): ['A3', 'A4'],            (2, 0): ['A2'],
 (0, 3): ['A3', 'A4', 'A5'],      (2, 1): ['A2'],
 (0, 4): ['A4', 'A5'],            (2, 2): ['A2', 'A4'],
 (0, 5): ['A5'],                  (2, 3): ['A4'],
 (1, 0): ['A2', 'A1'],            (2, 4): ['A4', 'A6'],
 (1, 1): ['A2', 'A1', 'A3'],      (2, 5): ['A6']}
 (1, 2): ['A2', 'A3', 'A4'],
 (1, 3): ['A3', 'A4', 'A5'],
```

Fig. 4. Signal Availability vector for the grid world environment



```
{(0, 0): {'A1': 6.2},  (1, 1): {'A1': 3.94,  (1, 5): {'A5': 4.68,
 (0, 1): {'A1': 4.6,            'A2': 4.64,            'A6': 5.35},
          'A3': 4.64},          'A3': 4.18},  (2, 0): {'A2': 6.36},
 (0, 2): {'A3': 4.18,  (1, 2): {'A2': 3.33,  (2, 1): {'A2': 4.88},
          'A4': 4.42},          'A3': 3.65,  (2, 2): {'A2': 4.65,
 (0, 3): {'A3': 4.74,           'A4': 4.08},          'A4': 4.3},
          'A4': 5.0,   (1, 3): {'A3': 3.36,  (2, 3): {'A4': 4.49},
          'A5': 4.25},          'A4': 3.99,  (2, 4): {'A4': 4.38,
 (0, 4): {'A4': 4.41,           'A5': 3.68},          'A6': 3.81},
          'A5': 5.62}, (1, 4): {'A4': 4.13,  (2, 5): {'A6': 5.36}}
 (0, 5): {'A5': 5.65},          'A5': 4.05,
 (1, 0): {'A1': 5.8,            'A6': 4.06},
          'A2': 6.35},
```

Fig. 5. Q-table after 5000 episodes

## V. Evaluation Results

As explained in the previous sections, we built a Q-table having the epsilon value as 0.82, gamma as 0.95, and the learning rate as 0.1. We performed the epsilon decay after every episode to decay exploration of antennas. We set the maximum steps taken per episode as 150. We observed the Q values of the antennas after 5000 episodes. We were able to see a decrease in the number of handovers towards the end of the iterations which showed that our model worked fine as expected. The final Q-table shown in Fig. 5. that we got after 5000 episodes shows the Q values at each available state on the grid. Whenever the mobile phone reaches a particular state on the grid, the mobile phone can refer to this Q table and select the antenna with the highest value. This can be repeated until the mobile phone has reached its final destination in the grid. We can also observe different values of handovers at different episodes. The handovers are random as each episode considers a random walk policy.

## VI. Conclusion and Discussion

In this paper, we employed Q-Learning of reinforcement learning to minimize the number of handovers in a dense 5G network. We created a prototype of the dense 5G network by creating a 3x6 grid world environment with multiple gNodeBs on different states. We showed how the agent started from the initial state and performed random walks in all possible directions to learn about different states in the grid world environment and reduce the number of overall handovers in the dense 5G network. We showed how the final Q-table can be utilized to select the best antenna at a particular state of the grid world. The results also indicate how the final Q-table can be used to minimize the ping pong effect caused by a gNodeB pair.

The number of episodes can be increased and also the epsilon value for choosing antennas can also be modified to make the algorithm more robust. DynaQ learning can be applied instead of Q-Learning to make the model learn in fewer iterations. This model can be deployed in the 5G enabled mobile phones of users to learn about the neighboring gNodeBs. The problem of the selection of link beams in a 5G network can also be addressed using the proposed model.

## VII. Related Works

There only exists a very few researches that use reinforcement learning to solve the problem of frequent handovers. [1] considers multi-arm bandit and takes in Reference Signal Received Power (RSRP) values as parameters for the handover process. They have utilized the advantages of Q-learning to optimize handovers. While on the other hand in [2] , the researchers have utilized supervised learning to optimize the handovers. They claim that offline learning is much better than the online way of learning the model. For this reason, they focus their work on the supervised ML, in which the training data is collected and an offline model is trained, after which an online prediction is done using the trained model. [3] focuses on the handovers that take place in LTE for railways (LTE-R). LTE for Railways (LTE-R) is a next-gen communications network dedicated to railway services, enabling high-speed wireless voice and data communications inside trains, from the train to the ground, and from train to train. They have utilized reinforcement learning to solve this problem. [4] uses Machine Learning to combat the latency challenges in mmWave MIMO systems.

# References

[1] V. Yajnanarayana, H. Rydén, and L. Hévizi, "5g handover using reinforcement learning," in *2020 IEEE 3rd 5G World Forum (5GWF)*, pp. 349–354, 2020.

[2] A. Masri, T. Veijalainen, H. Martikainen, S. Mwanje, J. Ali-Tolppa, and M. Kajó, "Machine-learning-based predictive handover," in *2021 IFIP/IEEE International Symposium on Integrated Network Management (IM)*, pp. 648–652, 2021.

[3] X. Cai, C. Wu, J. Sheng, J. Zhang, and Y. Wang, "A parameter optimization method for lte-r handover based on reinforcement learning," in *2020 International Wireless Communications and Mobile Computing (IWCMC)*, pp. 1216–1221, 2020.

[4] A. Alkhateeb, I. Beltagy, and S. Alex, "Machine learning for reliable mmwave systems: Blockage prediction and proactive handoff," in *2018 IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, pp. 1055–1059, 2018.