CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF ELECTRICAL ENGINEERING

DEPARTMENT OF TELECOMMUNICATION ENGINEERING

DIPLOMA THESIS

MACHINE LEARNING FOR HANDOVER IN MOBILE NETWORKS

STROJOVÉ UČENÍ PRO ŘÍZENÍ HANDOVERU V MOBILNÍCH SÍTÍCH

Author: Bc. Petr Škába

Study Programme: Electronics and Communications

Specialization: Mobile Communications

Supervisor: doc. Ing. Zdeněk Bečvář, Ph.D.

Prague 2023

Thesis Supervisor

doc. Ing. Zdeněk Bečvář, Ph.D.

Department of Telecommunication Engineering

Faculty of Electrical Engineering

Czech Technical University in Prague

Technická 2

160 00 Prague 6

Czech Republic

MASTER'S THESIS ASSIGNMENT

I. Personal and study details

Student's name: **Škába Petr** Personal ID number: **474252**

Faculty / Institute: **Faculty of Electrical Engineering**

Department / Institute: **Department of Telecommunications Engineering**

Study program: **Electronics and Communications**

Specialisation: **Mobile Communications**

II. Master's thesis details

Master's thesis title in English:

Machine Learning for Handover in Mobile Networks

Master's thesis title in Czech:

Strojové učení pro řízení handoveru v mobilních sítích

Guidelines:

Study possibility to exploit machine learning for control and management of handovers in mobile networks. Develop an algorithm that learns and classifies statistics of the communication channel changes over time via machine learning and provides this knowledge to the algorithm that learns setting of the handover control-related parameters. Develop also a framework for cooperation of both machine learning algorithms to adjust their own decisions with respect to the decision of the other one in order to maximize quality of service of the users. Evaluate performance of the proposed solution via simulations and compare it with related works.

Bibliography / sources:

[1] A. Madelkhanova, Z. Becvar and T. Spyropoulos, "Optimization of Cell Individual Offset for Handover of Flying Base Stations and Users," IEEE Transactions on Wireless Communications, early access, 2022.
[2] A. Madelkhanova, Z. Becvar and T. Spyropoulos, "Q-Learning-based Setting of Cell Individual Offset for Handover of Flying Base Stations," IEEE Vehicular Technology Conference (IEEE VTC2022-Spring), 2022.
[3] M. Najla, Z. Becvar, P. Mach and D. Gesbert, "Predicting Device-to-Device Channels from Cellular Channel Measurements: A Learning Approach," IEEE Transactions on Wireless Communications, vol. 19, no. 11, 2020.
[4] S. Su, T. Chih and S. Wu, "A novel handover process for mobility load balancing in LTE heterogeneous networks," IEEE ICPS, 2019.

Name and workplace of master's thesis supervisor:

doc. Ing. Zdeněk Bečvář, Ph.D. Department of Telecommunications Engineering FEE

Name and workplace of second master's thesis supervisor or consultant:

Date of master's thesis assignment: **12.01.2023** Deadline for master's thesis submission: **26.05.2023**

Assignment valid until: **22.09.2024**

doc. Ing. Zdeněk Bečvář, Ph.D. Supervisor's signature	Head of department's signature	prof. Mgr. Petr Páta, Ph.D. Dean's signature

III. Assignment receipt

The student acknowledges that the master's thesis is an individual work. The student must produce his thesis without the assistance of others, with the exception of provided consultations. Within the master's thesis, the author must state the names of consultants and include a list of references.

. Date of assignment receipt	Student's signature

Declaration

I hereby declare I have written this diploma thesis independently and quoted all the sources of information used in accordance with methodological instructions on ethical principles for writing an academic thesis. Moreover, I state that this thesis has neither been submitted nor accepted for any other degree.

In Prague, May 2023 ....

Bc. Petr Škába

ABSTRACT

This diploma thesis presents a novel approach for optimizing handovers in mobile networks by combining reinforcement learning-based Cell Individual Offset (CIO) optimization and deep learning-based channel quality prediction for Flying Base Stations (FlyBS) to User Equipment (UE) channels. The proposed solution is part of a two-step workflow, where the FlyBS-UE channels are predicted based on the known Ground Base Station (GBS) to UE channels, followed by a reinforcement learning agent setting CIO values for the network base stations based on the immediate network state, including the predicted channel qualities. This diploma thesis also discusses the reinforcement learning-based reward function and its relationship to handovers, highlighting the overall trend of the function. The function value increases with the number of FlyBSs, indicating potential network quality improvements. The performance of the proposed solution is compared with state-of-the-art approaches. In terms of network capacity, the proposed solution achieves superior results, outperforming other state-of-the-art approaches. Additionally, the proposed solution effectively reduces the number of all handovers as well as ping-pong handovers compared to other approaches. This diploma thesis demonstrates the potential of combining reinforcement learning and deep learning techniques for optimizing handovers in next generation mobile networks with reduced signaling overhead for channel quality acquisition.

Keywords: mobile network, handover, reinforcement learning, deep learning, flying base station, cell individual offset

ABSTRAKT

Tato diplomová práce představuje nový přístup k optimalizaci handoveru v mobilních sítích pomocí kombinace optimalizace parametru Cell Individual Offset (CIO), založené na zpětnovazebním učení a předpovědi kvality kanálu, založené na hlubokém učení, pro kanály mezi létajícími základnovými stanicemi (FlyBS) a uživatelským vybavením (UE). Navrhované řešení je součástí dvoufázového procesu, kde jsou kanály FlyBS-UE predikovány na základě známých kanálů mezi pozemními základnovými stanicemi (GBS) a UE, po nichž následuje zpětnovazební učení nastavující hodnoty CIO pro základnové stanice mobilní sítě na základě okamžitého stavu této sítě, včetně predikovaných kvalit kanálu. Diplomová práce také pojednává o funkci odměny zpětnovazebního učení a jejímu vztahu k handoverům, přičemž zdůrazňuje celkový trend této funkce. Hodnota funkce roste s počtem FlyBS, což naznačuje možné zlepšení kvality sítě. Výsledky navrženého řešení jsou porovnány s nejmodernějšími přístupy. Z hlediska kapacity sítě dosahuje navrhované řešení vynikajících výsledků a překonává ostatní nejmodernější přístupy. Kromě toho, v porovnání s ostatními přístupy, navrhované řešení účinně snižuje počet všech handoverů, včetně ping-pong handoverů. Tato diplomová práce demonstruje potenciál kombinace zpětnovazebního a hlubokého učení pro optimalizaci handoverů v sítích nové generace s redukcí signalizace při zjišťování kvality kanálu.

Klíčová slova: mobilní síť, handover, zpětnovazební učení, hluboké učení, létající základnová stanice, cell individual offset

TABLE OF CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

TABLE OF EQUATIONS

TABLE OF ALGORITHMS

TABLE OF ABBREVIATIONS

BS .... *Base Station*

CIO .... *Cell Individual Offset*

DNN .... *Deep Neural Network*

FlyBS .... *Flying Base Station*

FSL .... *Free Space Loss*

GBS .... *Ground Base Station*

HPR .... *Handover Ping-Pong Ratio*

LSTM .... *Long Short-Term Memory*

R .... *Reward (function)*

RMSE .... *Root Mean Square Error*

SINR .... *Signal-To-Interference-Plus-Noise Ration*

TD .... *Temporal Difference*

TTT .... *Time-To-Trigger*

UAV .... *Unmanned Aerial Vehicle*

UE .... *User Equipment*

1 INTRODUCTION

New generations of mobile networks tries to build upon the older generation networks to improve performance. With the introduction of next generation of mobile networks, this is achieved via the implementation of *flying base stations* (FlyBS) carried by drones, or more generally speaking, *unmanned aerial vehicles* (UAV) [1]. The FlyBSs, connected via wireless link to the conventional *ground base station* (GBS), allow to extend the network coverage [2], boost quality of service or even provide mobile services altogether in a certain area, unreachable by the conventional GBSs [3], for example due to the natural disaster [4]. However, the introduction of FlyBSs to the mobile network poses new challenges of its own for modern technology to solve. To name a few, we need to solve the problem of optimal positioning of FlyBSs [5], find a solution to optimizing their trajectory [6] and also associating *User Equipment* (UE) with FlyBSs [7]. If FlyBS serves as a transparent relay, the problem complicates even more, as UEs have no way of measuring the FlyBS to UE channel quality on their own. UEs then must rely on other methods of learning the channel quality, e.g., channel prediction from data already available to the network [1].

As said in [8], another challenge in next generation of mobile networks with FlyBSs is to provide a seamless mobility for the UEs connected to the FlyBSs. This means that to work properly as a *base station* (BS)[1], transparent or not, FlyBS perform handovers. The FlyBS mobility is not unlike the mobility of UEs in a conventional mobile network, which is also managed by the handover procedure [9]. In conventional mobile networks, handover is initiated when a potential neighboring GBS provides a higher channel quality than the current serving GBS. However, to avoid frequent ping-pong handovers, more parameters than just channel quality are needed. In the next generation mobile networks, with FlyBSs included, the same principle can be utilized. The same rules and parameters as in conventional handover procedure can be applied to handovers performed by the FlyBSs. With this principle in mind, we can apply a unified solution to both the conventional UE mobility and FlyBS mobility. However, with the addition of FlyBSs into the network, the overall number of BSs increase substantially. Even if we presume only few FlyBSs per single GBS, the amount of computation required by traditional methods and even some more complex machine learning-based methods, grows quickly [10]. Thus, there is a need for a more refined solution, which is applicable to both the conventional handovers and FlyBSs handovers and is not so strongly dependent on the number of actors in the mobile network.

To solve both the FlyBS to UE channel prediction and handover decision optimization described above, this diploma thesis focuses on the optimization of *Cell Individual Offset* (CIO), present with every BS, while simultaneously predicting the channel quality for every in the FlyBSs in the network. This is achieved via the newly proposed solution, which works by combining two different machine learning approaches into a single unified framework. The two combined machine learning approaches are the reinforcement learning-based CIO setting proposed by the work of A. Madelkhanova et al., 2022 [8] and deep learning-based FlyBS to UE channel prediction proposed by M. Najla et al., 2020 [1]. However, instead of two separate solutions working independently, the approach proposed in this diploma thesis takes advantage of both solutions at once and incorporates them into a single framework. This framework is usable in any mobile network model regardless of the used scenario.

[1] Throughout the work, the abbreviation BS is used as an umbrella term for both GBS and FlyBS, when the distinction is not important for the context or used solution.

The remainder of this diploma thesis is organized as follows. Section 2 presents the communication model of the mobile network used in this diploma thesis. Section 3 formulates the problems, which are then addressed by the approach presented in this diploma thesis. Section 4 proposes the solutions to the problems formulated in Section 3. Section 5 presents a way to evaluate the performance of the proposed solution against other state-of-the-art approaches. Section 6 presents the performance evaluation data of the proposed solution via the simulations results. Finally, Section 7 concludes this diploma thesis and gives directions for future improvements and the following works. Sections are followed by Appendix A, which lists all digital media included with this diploma thesis, meaning custom-made functions and simulation scripts.

2 SYSTEM MODEL

In this section, the communication model of the network is presented, designed in a fashion similar to the work of A. Madelkhanova et al., 2022 [8]. This is then followed by the description of the channels between BSs, both flying and ground and UEs. Lastly, the adopted handover procedure is defined with its trigger conditions for both the UEs and FlyBSs.

2.1 Network Model

In the presented network model, there are N UEs deployed in the area covered with G GBS and F FlyBSs. This makes the total number of base stations equal to B = G + F. Specific numbers of UEs, GBSs and FlyBSs are listed in the later sections.

The UEs move over time to simulate user movement while maintaining a connection to either GBS or FlyBS. The position of the FlyBSs is influenced by the UEs connected to them. Each FlyBS is deployed in the *center of gravity* of their connected UEs. However, the proposed approach is independent of the movement and position of both UEs and FlyBSs and can be applied in all possible scenarios. Specifics of the used UE movement model are listed in the later sections.

For each UE, the *required communication capacity* is defined. For additional accuracy of the proposed network model the required capacity is set differently for each UE. To always maintain a good connection, and subsequently, a good channel capacity, the UEs and the FlyBSs perform conventional handovers and FlyBS handover, respectively, to satisfy the required UE capacity. The presence of handovers in the proposed model means that the connections of UEs to the BSs, either flying or ground, or FlyBSs to the GBSs change over time. To this end, we define a binary matrix $\boldsymbol{\beta}$ which indicates if the n-th UE is connected to the b-th BS ($\beta_{n,b} = 1$) or not ($\beta_{n,b} = 0$). Same matrix is also defined for the FlyBSs, to indicate which FlyBS is connected to which GBS.

To remove an unnecessary variability from the network model, the UEs are always connected to the network, meaning that there is no possibility of UEs disconnecting from the network due to the distance, line of sight or an overall poor channel quality. Similarly, every FlyBS always maintain a connection to GBS.

2.2 Channel Model

For our model only the downlink communication is considered either from GBS or FlyBS. For the sake of simplicity, the communication channel between each BS and UE or GBS and FlyBS, is defined by parameter *Free Space Loss* (FSL) [dB], which is defined as:

$$\mathrm{FSL} = \left(\frac{4\pi d f}{c}\right)^2, \tag{1}$$

where d [m] is distance, f [Hz] is frequency and c [m/s] is the speed of light.

Another channel parameter is *Signal-to-Interference-Plus-Noise Ratio* (SINR) [-] defined as:

$$\mathrm{SINR} = \frac{S}{I + N}, \tag{2}$$

where S [dB] is the received signal level, I [dB] is an interference, representing the signals from other UEs or BSs in the same band[2], and N [dB] is the background noise. SINR is measured from GBS to UE,

[2] In this work, we presume that all UEs and BSs share the same band, meaning the interference I is from all other UEs and BSs in the simulation.

FlyBS to UE and GBS to FlyBS, where the measured channel is degraded by interference from other communications and the background noise.

Channel parameter SINR is then used to obtain the *channel capacity* C [bits/s], which is defined as:

$$C = B \log_2(1 + SINR), \tag{3}$$

where B [Hz] is the channel bandwidth and $SINR$ [-] is the signal-to-interference-plus-noise ratio defined by Equation *(2)*.

Based on the required communication capacity the *bandwidth* B_n for the n-th UE is allocated at the start of the simulation to satisfy the UEs requirements. All subsequent bandwidth allocations and releases are managed by the handover procedure. All UEs, connected to the b-th BS, impose *total load* ρ_b on b-th BS. However, the proposed solution is not dependent on the way the bandwidth is allocated or the way the channel gain and capacity are calculated.

2.3 Handover Procedure

Handover between the serving BS and the target BS is triggered according to the common 3GPP handover event A3 [8]. This means that handover is performed if the following is satisfied at least for the period of *Time-To-Trigger* (TTT): the signal strength to the target BS plus target BS CIO is higher than the signal strength to the serving BS plus serving BS CIO plus handover hysteresis. In this case the signal strength is defined as signal power plus channel gain in decibels. To put all this in an equation, we define the condition for handover in mobile networks as:

$$\text{Handover Condition (valid for TTT): } \mathrm{P_t} + \mathrm{G_t} + \mathrm{CIO_t} > \mathrm{P_s} + \mathrm{G_s} + \mathrm{CIO_s} + \mathrm{Hys}, \tag{4}$$

where $\mathrm{P_t}$ [dB] and $\mathrm{P_s}$ [dB] is the received signal strength at the target and serving BSs, respectively, $\mathrm{G_t}$ [dB] and $\mathrm{G_s}$ [dB] is the channel gain to the target and serving BS, respectively and $\mathrm{CIO_t}$ [dB] and $\mathrm{CIO_s}$ [dB] is the cell individual offset of the target and serving BS, respectively. The last term Hys [dB] is the hysteresis value to prevent quick back-and-forth, i.e., ping-pong handovers. The handover itself is triggered when the condition *(4)* is true for the whole duration of the TTT value.

With the introduction of FlyBSs into the next generation network, we define a new type of handover, namely the FlyBS handover. Unlike the conventional handover, this new type of handover is done by FlyBS between GBSs. Same as the UEs in conventional mobile networks, FlyBSs also measure the channel quality from all the neighboring GBSs. This channel quality is then reported periodically to the GBSs in the same way the UEs report their own channel quality to their neighboring GBSs. The handover procedure for the FlyBSs is triggered according to the same A3 3GPP event as the conventional UE handovers.

3 PROBLEM FORMULATION

In this section, the main problem is defined, which is addressed by this diploma thesis. Main objective is divided into two categories, each solved by a different machine learning approach. The first part is the CIO optimization addressed via the reinforcement learning. The second part is the problem of the channel quality prediction.

3.1 CIO Optimization

The primary objective of this diploma thesis is to optimize handover decision in the mobile networks for both UEs and FlyBSs, which serve the UEs. This is done via the CIO optimization. The CIO setting has a direct impact on the handover decision, as stated above. Handovers and their number have an impact on the sum capacity of the network. By moving the UE via handover to a less loaded BS, the overall stability of the network is increased as it prevents a potential problem with peak loads and/or insufficient bandwidth on particularly high-traffic BSs.

However, capacity alone is not the only critical parameter for CIO optimization as a pure maximization of capacity can lead to an excessive number of redundant handovers. Redundant handovers increase the signaling overhead and consequently the energy consumption of both the network and communicating UEs. The increase of the signaling overhead can even lead to a decrease in communication capacity. Thus, the number of handovers should be controlled to find the right balance between the cost of network operation and sum capacity.

In other words, we optimize the handover decision by setting a CIO value to each BS and then we observe the goal metrics like number of handovers and sum capacity. The goal is to decrease the number of handovers, more specifically to avoid quick ping-pong handovers while increasing the sum capacity.

3.2 Channel Prediction

To fulfill the primary objective, we first need to find a way to work with FlyBS. In conventional mobile networks, the UEs periodically report their channel quality to the corresponding serving GBS to perform handovers, thus fulfilling their capacity requirements. With the addition of FlyBSs as transparent relays, we not only need to know the channel quality from GBSs to UEs but also from FlyBSs to UEs.

One way of approaching this problem would be to utilize the same concept as in conventional mobile networks with only GBSs and make UEs periodically measure the channel quality to their neighboring FlyBSs as well as GBSs. However, this would add unnecessary signaling messages to the network and thus increase the energy consumption of UEs while lowering the capacity. Also, such channel measurements would be impossible with FlyBSs acting as transparent relays.

Another, more suitable approach, would be to use already known information, such as the channel quality between the GBSs and the UEs, which UEs already periodically measure and use these data to train a deep neural network to predict channel quality between FlyBSs and UEs. The channel prediction approach generates no additional signaling messages. All the work is done on the network side, so it poses no additional energy cost for the UEs. As the FlyBSs are in the center of gravity of its multiple connected UEs, it is presumed that the predictions are possible with only a little prediction error. The prediction accuracy for a FlyBS to UE channel quality should also increase with the number of UEs connected to the network.

However, we still need to consider the possible prediction error, which might render the proposed approach unsuitable for use in real mobile networks. The prediction error, in this diploma thesis defined as *Root Mean Square Error* (RMSE) [dB], should remain as low as possible. RMSE is defined as:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}|A_i - F_i|^2}\,, \tag{5}$$

where n [-] is number of realizations, A_i [dB] is the measured channel quality and F_i [dB] is the predicted channel quality.

However, due to the time constraints the influence of certain parameters on the prediction error is not fully explored in this diploma thesis. The considered parameters are length of the data sets, the number of data sets, number of training epochs, batch size or learning rate[3]. The optimizing of the prediction error remains a candidate for future works.

[3] The best RMSE values achieved in this work during the DNN training were between 1.5 to 1 dB after 750 to 1000 epochs with batch size equal to 20 training samples and learning rate set to 0.001.

4 PROPOSED SOLUTION

In this section a solution to the problems described in the previous section is proposed. First, a way to predict the FlyBS to UE channel quality is utilized to even be able to implement the proposed solution itself [11]. Next, the handover decision is improved by optimizing the CIO parameter of each BS in the network [8].

4.1 Channel Quality Prediction based on Deep Learning

As stated in the previous chapter, we need a way to predict the channel quality between FlyBSs and UEs based on the channel quality from GBSs to UEs. This task is not a matter of simple prediction from past data, but a real-time prediction of desired data from other, unrelated set of data. To achieve this, we utilize a part of *Deep Learning* called *Deep Neural Network* (DNN). Unlike other, more simple forms of neural networks, DNN is equipped with a higher number of different and sometimes specialized layers. This makes DNN more suited for channel quality prediction. To achieve an accurate prediction, we first need to prepare a neural network and training data.

The proposed neural network is created by combining several types of layers, namely sequence input layer, which accepts whole sequences of data, like measured channel quality over time. The input layer is then followed by hidden layers. The hidden layers are a combination of three *Long Short-Term Memory* (LSTM) [12] layers and three *Fully Connected* [13] layers that alternate with each other. The last layer is an output regression layer to make the final prediction sequences. The combination and number of these layers is selected based on the trial-error approach during the DNN design phase.

To train the DNN in a supervised manner [14], training data sets from simulations are needed. In contrast with a real-life mobile network, in simulations, desired output data can easily be obtained, meaning channel quality from FlyBSs to UEs. The FlyBSs to UEs channel quality data as *Targets*, together with the UEs to GBSs channel quality data as *Predictors* create a training data set for the used model. Theoretically, some training data could be obtained from a real mobile network, like UEs to GBSs channel quality. However, the main problem would be to get real FlyBSs to UEs channel quality data to train the proposed DNN. For that end, a real-life experiment with one, or preferably, multiple FlyBSs and UEs, which would measure the channel quality, would have to be orchestrated. This would deviate from the main goal of this diploma thesis, which is to prove that the proposed solution even works. For that end, the simulation data are suitable. However, such an experiment is a potential candidate for future works.

The sub-goal is to train the neural network to output predicted channel quality from FlyBSs to UEs based on the input, which is channel quality from neighboring GBSs to UEs, which is known in real-life mobile networks. Quality of such training is then measured in RMSE, which represents the difference between predicted channel quality and the input data set.

Deep neural network trained in such manner can be deployed in a real mobile network, meaning without the prior knowledge of channel quality between FlyBSs and UEs.

4.2 CIO Adjustment based on Reinforcement Learning

The problem defined in the previous section is overly complex. It stems from the fact that the environment of the mobile network is highly randomized. This is caused by the movement of both the UEs and FlyBSs, which follow their connected UEs. Random movement patterns can be hard to predict. However, to be able to optimize the handover decision, we need to be able to work with such a random environment at the real time without prior knowledge of the UEs movement or their potential future position. Prior information would certainly make the work much easier but in a real-life mobile network, such information is often unavailable, mostly due to privacy concerns or simply because of the sheer impracticality of such measurements. Thus, we need to deploy an algorithm that is able to work in real time without any prior knowledge or data which leads us to the use of *Reinforcement Learning*.

Unlike other machine learning approaches, which require prior knowledge of environment, reinforcement learning can make decisions based on immediate data from the environment without prior extensive training [15]. In this diploma thesis, the reinforcement learning is used to set the CIO value of individual base stations, both flying and ground, based on the immediate information from the network such as the sum capacity or in this case a custom-defined *Reward* (R) function.

Reinforcement learning itself has several variants. Some are based on Q-learning while others are focused on deep learning, such as the *Actor-Critic* Reinforcement Learning approach. In the work [8] it is proven, that for the CIO setting optimization the Actor-Critic, i.e. deep reinforcement learning approach is superior to the Q-learning-based reinforcement learning and hence, the Actor-Critic approach was chosen, which utilizes two deep neural networks working in tandem, working as a *reinforcement learning agent*.

4.2.1 Reinforcement Learning Agent

A distinctive feature of the Actor-Critic approach is the presence of two deep neural networks, one serving as the Actor and the other as Critic. As the name suggests, actor network acts on the environment itself, like setting the CIO parameter, chosen from a predefined set of CIO values. The critic network evaluates the action, like checking whether the new CIO parameter increased or decreased defined reward function. This is represented as *Temporal Difference* (TD) error, which serves as feedback for the Actor-Critic deep neural networks. Next action is then taken based upon the increase or decrease in the TD error. Both neural networks also update their internal weights based on this error. The goal of this approach is to set the optimal CIO of all the base stations to maximize the defined R function, which serves as a representative of more common parameters like the sum capacity of the network or a number of handovers and handover cost.

4.2.2 Reward Function

The custom-defined R function , taken from the work [8] with only a slight change, represents the objective of the problem formulated above. The R function is a combination of several key parameters like sum capacity, number of handovers and their cost or the BS load. In terms of equation, it is defined as:

$$\mathrm{R} = \frac{\sum_{n=1}^{N_f} c_n}{N_f c_{req,n}} - \left(\sum_{i=1}^{n_h} \rho_i \frac{\rho_{t,i}}{\rho_{s,i}} + n_h \mu_u \right) + const., \quad (6)$$

where N_f [-] is the number of UEs connected to the FlyBS, c_n [bits/s] is a channel capacity for the n-th UE and $c_{req,n}$ [bits/s] is the required capacity of the n-th UE. The term ρ_i [-] corresponds to the load, which is implied by the UE performing handover. The terms $\rho_{t,i}$ [-] and $\rho_{s,i}$ [-] correspond to the load of the target and serving BSs, respectively. Lastly, to prevent unnecessary handovers, the μ_u [bits] corresponds to the handover cost and n_h [-] to the number of UEs performing handover in the same time slot. When compared with the R function in [8], the R function in this diploma thesis has a constant added at the end. This constant is set to a value 1. This was done via trial-error during the model implementation to improve the R function stability and human readability. The possible negative values of the R function were causing errors with the used MATLAB Reinforcement Learning Toolbox [16]. Value 1 was added to keep the R function always in non-negative values without changing the original R function too much.

4.2.3 Reinforcement Learning Environment

The agent defined above, meaning the actor and critic deep neural networks, operates on the custom-defined *environment*. The reinforcement learning environment is a set of Objects and Actions which emulate the workings of a mobile network, working as a mobile network model.

In the model there are two types of Objects: UEs and BSs, which are divided into FlyBSs and GBSs. The Actions represent different interactions within the network like UE movement, handovers or measurements of different signals and parameters.

4.3 Naming Convention

Throughout this diploma thesis, words model and environment are mentioned frequently. However, for clear understanding of the proposed solution, it is necessary to establish a firm naming convention. This is due to the fact that in this diploma thesis, the defined network model, meaning the mobile network itself, serves two different purposes, depending on its position in the proposed solution workflow.

4.3.1 Model

The first use case for the network model is to obtain simulation data, which are needed for additional steps, for example GBSs to UEs or FlyBSs to UEs simulated channel quality data. The measured channel quality data obtained from the model are used to train the prepared DNN. In this case the network model is referred to simply as *model.*

4.3.2 Environment

The second use case for the network model is to serve as an action space for the reinforcement learning agent. In this case, the agent is changing the network parameters, namely the CIO of each base station, to optimize the handover decision process. More specifically, the task of the reinforcement learning agent is to select a CIO value for each base station, both ground and flying, from a predefined set of discrete CIO values. In this step of the workflow, the network model is referred to as an *environment*, as per the official MATLAB naming convention set by the Reinforcement Learning Toolbox [16]. Unlike the *model,* the output of the *environment* are not channel quality data but performance data like capacity, number of handovers or the R function.

4.3.3 Key Differences

The key differences in the inner working of the *model* and the *environment* are narrowed down to two key aspects, the reinforcement learning usage and the deep learning usage.

4.3.3.1 Reinforcement Learning

The general difference between *model* and *environment* in terms of the reinforcement learning is that in the *model*, the CIO values are selected randomly from a predefined set of CIO values and remain fixed for the duration of the simulation. On the other hand, the *environment* has the CIO values set by a reinforcement learning agent. In each time step, the CIO values are chosen from the predefined set by the agent based on the observations received from the *environment* in real time

4.3.3.2 Deep Learning

From the perspective of the DNN channel prediction, the *model* channel qualities, namely GBS to UE, FlyBS to UE and GBS to FlyBS, are measured and not predicted. In other words, the DNN is not part of the *model* in any way. This contrasts with the *environment*, where the FlyBS to UE channel is specifically predicted, not measured, by the integrated DNN. Other channels, namely the GBS to UE and GBS to FlyBS are still measured as in the *model.*

Other inner workings of the *model* and *environment* like movement or handover trigger procedure remains the same., i.e., are not influenced by the changes in the agent or DNN.

4.4 Workflow of the proposed solution

As already mentioned in the introduction, this diploma thesis builds on the works of A. Madelkhanova et al., 2022 [8] and M. Najla et al., 2020 [1] by combining two different machine learning approaches into a single new solution. The connection of the two approaches is done via the network model. As mentioned in the previous subsection, the network model, as described by the Algorithm 1, is modified by the proposed solution into the *environment*, which represents the connection of the two machine learning approaches. However, the *environment* itself is part of the entire solution workflow. Before defining the proposed solution, the solution workflow is described. The entire solution workflow is in Algorithm 2 and Figure 2.

The first major step in the workflow, aside from defining the necessary input variables, is at the line 5 of the Algorithm 2. A more detailed description of the first step, i.e., the *model*, is in Algorithm 1.

When omitting basic procedures like UE position update, which has no influence over the solution, the first major lines in Algorithm 1 are lines 6 and 7. These lines depict the way the channel quality is obtained. The channel quality data are calculated, i.e., measured using only the data available internally to the *model*. After the channel quality is measured, the condition for the handover is checked at line 8. If the conditions are satisfied, the handover is performed. At the end of the algorithm, every necessary output variable is obtained like capacity or the number of handovers. The output variables are then returned back to the workflow as training data for the DNN. The training data are separated into the GBS to UE channel quality data sets, serving as predictors and FlyBS to UE channel quality data sets, serving as targets.

Another major step in the workflow is at the line 9 of the Algorithm 2, which represents the supervised training of the DNN network. The prediction data sets are fed as an input to the DNN. The output of the DNN is then compared to the target data, with the difference between the DNN output and target data measured as RMSE.

Algorithm 1 High level description of the model[4]

1	input variables $Vars_{in}$ = *CIO*
2	counters *Count* = counters for HOs[5], FlyBS HOs, HO_{pp}, other variables
3	**for** number of time slots *ts* **do**
4	**for** number of UEs *i* **do**
5	**for** number of GBSs, FlyBSs *g, f* **do**
6	channel quality G_{GBS_UE} = **calculate**(*FSL, i, g*)
7	channel quality G_{FlyBS_UE} = **calculate**(*FSL, i, f*)
8	**if** A3 event handover condition = **true**
9	**perform_handover**(*CIO*)
10	*Count* = number of UE HOs, FlyBS HOs, HO_{pp}, other variables
11	*SINR*, capacity *C* = **calculate**(*G, Count*)
12	output variables $Vars_{out}$ = *Count, C*
13	**return** $Vars_{out}$ as training data

[4] For simplicity, listed algorithm follows Python convention, ending the *loops* and *if-statements* with indents.

[5] HO stands for Handover, HO_{pp} stands for ping-pong handover

Algorithm 2 High level description of the solution workflow

1	input variables $Vars_{in}$ = number of GBS, FlyBS, CIO set, observations, steps
2	model M = **initialize_model**($Vars_{in}$)
3	**for** number of observations o **do**
4	**for** number of steps n **do**
5	training data D_{train} (o) = **run_model**(M, n), see Algorithm 2
6	targets T, predictors P = **separate_data**(D_{train})
7	empty DNN_{empty} = **initialize_dnn**(*layers*)
8	**for** number of observations o **do**
9	trained $DNN_{trained}$ = **train_dnn**(T,P, DNN_{empty})
10	RL environment $RLE = M + DNN_{trained}$
11	RL A-C[6] agent RLA_{empty} = **initialize_agent**(RLE)
12	**for** number of steps n **do**
13	trained $RLA_{trained}$ = **train_dnn**(RLE, RLA_{empty})
14	**for** number of simulations s **do**
15	**reset_environment**
16	for number of steps n **do**
17	CIO = **select_cio**($RLA_{trained}$, CIO_set, $Vars_{out}(n\text{-}1)$)
18	output variables $Vars_{out}$ = **run_environment**(RLE, $RLA_{trained}$, CIO, n), see **Error! Reference source not found.**
19	**return** $Vars_{out}$ as performance data

4.4.1 Proposed Solution

The line 10 of the Algorithm 2 is the beginning of the proposed solution. Line 10 describes the creation of the *environment* discussed in the previous subsection. After performing generic, although necessary [16] reinforcement learning-based steps[7] in the workflow at line 11 and line 13 of the Algorithm 2, the last part of the proposed solution is at line 17 and line 18 of the Algorithm 2. At line 17 of the Algorithm 2, the defined *environment* is subject to the Actions taken by the reinforcement learning agent, which sets the CIO values for the *environment*. The Action is based on the *environment* output variable from the previous time step called State, as per the official MATLAB naming convention for the Reinforcement Learning Toolbox [16]. In the proposed solution, the State variable is equal to the custom-defined R function, see Figure 2. This whole process is described in detail in Algorithm 3.

The Algorithm 3 works in the same way as the Algorithm 1, except for the lines 7, 8 and 13. The difference in line 7 is the way the channel quality is obtained. As described in previous sections, the channel quality data for the FlyBS to UE channel is predicted and not measured. The predicted channel quality is multiplied by a *Reward Difference* parameter, which is obtained at line 13. The *Reward Difference* represents the change in R function value obtained in the current time step and the previous time step. After the DNN-based prediction is done, the DNN update its internal weights based on the

[6] A-C stands for Actor-Critic

[7] These steps include creation and training of the A-C DNN-based RL Agent in a similar fashion to the training of the channel prediction DNN

predicted channel quality. The DNN weights update helps to maintain the necessary prediction accuracy, i.e., the low prediction error.

For better illustration of the proposed solution, Figure 1 depicts a high-level overview of the core of the proposed solution. More detailed diagram with all variables included in the solution is in Figure 2.

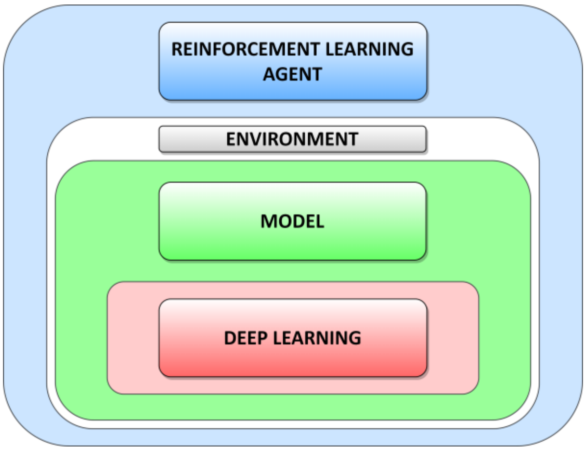


Figure 1 High-level diagram of the proposed solution

Algorithm 3 High level description of the environment

input variables = *RLE* (contains *CIO* and *DNN*), *ts* as time step from **Error! Reference source not found.**
counters *Count* = counters for HOs, FlyBS HOs, HO_{pp}, other variables
for number of time slots *ts* **do**
    for number of UEs *i* **do**
        for number of GBSs, FlyBSs *g, f* **do**
            channel quality G_{GBS_UE} = **calculate**(*FSL, i, g*)
            channel quality G_{FlyBS_UE} = (1 + *dR*) × **predict**(*DNN, i, f*)
            $DNN_{updated}$ = **update_network_weights**(*DNN*)
            if A3 event handover condition = **true**
                perform_handover(*CIO*)
            Count = number of UE HOs, FlyBS HOs, HO_{pp}, other variables
            SINR, capacity *C*, reward *R* = **calculate**(*G, Count*)
            reward difference *dR* = 1 – (*R*(*ts*-1)/*R*(*ts*))
output variables $Vars_{out}$ = *Count, C, R,* $DNN_{updated}$
return $Vars_{out}$ as performance data

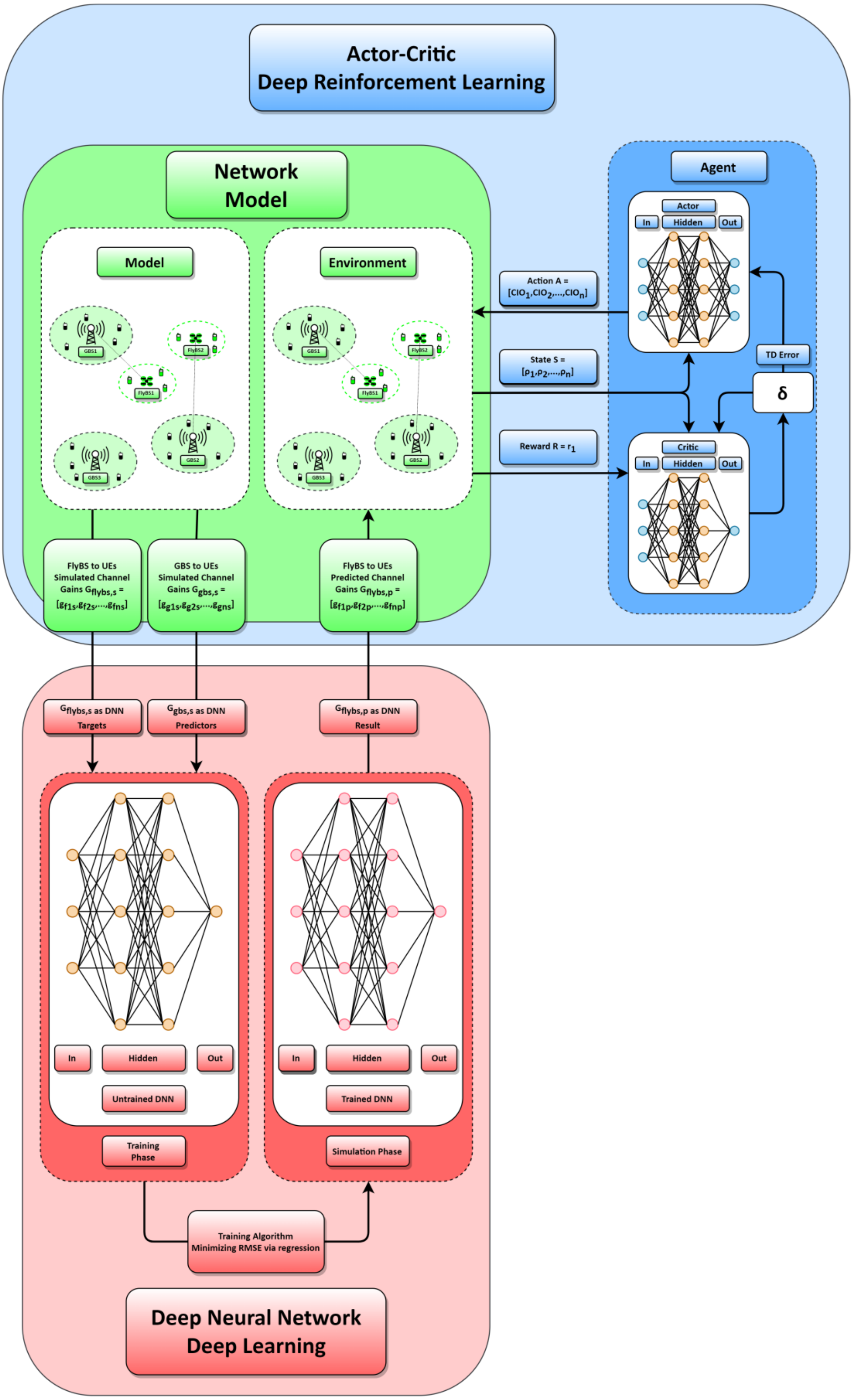


Figure 2 Diagram of the proposed solution workflow. For clarity, each distinct part of the solution is shown in a different color.

5 PERFORMANCE EVALUATION

In this section, the simulation parameters are defined. Some of these parameters are then used as the performance metrics for the evaluation of the simulations. First, all the parameters are divided into three categories. Each of the categories is tied to a different element within the simulation: BSs, UEs and handover procedure. After simulation parameters, important metrics for the performance evaluation are defined. Lastly, different simulation scenarios are presented, which will be compared with one another.

5.1 Simulation Parameters

To obtain enough data for the performance evaluation we run 25 different simulations, each running for 1000 steps, with 1 step being equal to 0.01 seconds. Some simulation parameters are randomized across simulations, like the starting position of each UE or their speed, while another set of parameters remains the same across all simulations. These constant parameters are mostly GBS-related like transmission power or bandwidth. Every important simulation parameter is listed in the Table 1.

5.1.1 Environment

We consider the simulation area of 4 km^2 (2 × 2 km). Within this area, three conventional GBSs are deployed at random positions. The average distance between each GBS is 500 meters. In addition to the GBSs, there are F FlyBSs. The position of each FlyBS is determined as the *center of gravity* of all UEs connected to the FlyBS. However, the positioning of these FlyBSs is indifferent to the proposed solution. This approach was chosen solely for its simplicity, meaning that the movement of both the UEs, and subsequently their serving FlyBSs, is unimportant for the simulations overall. The main purpose of this diploma thesis is to propose a new handover optimizing solution, not to assess the movement patterns of all involved objects.

5.1.2 User Equipment

The number of UEs in a simulation is defined as 45 UEs per GBS and 15 UEs per FlyBS. These UEs start the simulation uniformly distributed in the simulation area, connected to the closest base station. The UEs move in a custom-defined randomized manner. The position of each UE is updated every time step. The position update $\Delta_{position}$ is calculated as:

$$\Delta_{position} = Direction \times Speed \times \sin(Angle) + Random\,Value, \tag{7}$$

where $Direction$ [-] is the value from set {-1, 1} denoting whether the UE is going in the direction the axis or in reverse , $Speed$ [m/s] is a value from set {1.5, 3, 4.5}, which denotes the speed of the UE. The set of speed values is loosely based on different types of urban transportation, meaning pedestrians walking, running, or riding e.g., a bicycle. Each UE retains their speed for the entire duration of the simulation. Parameter $Angle$ [rad] has a value from range $\langle 0, 2\pi \rangle$, which is iterated over the course of the simulation. Parameter $Random\,Value$ [-] has value from range $\langle 0, 5 \rangle$, which pseudo-randomizes the movement of each UE. $\Delta_{position}$ is calculated for both the X and Y axis of the simulation area. The proposed movement model is based on movement model generally present in 2D video game simulations [17].

5.1.3 Handover

The handover is triggered according to A3 event as defined by 3GPP standards with parameter TTT set to 0.05 s and Hysteresis being equal to 3 dB. Parameter CIO is chosen from the set {-6, -3, 0, 3, 6} [dB] [8]. The handover trigger condition remains the same for both the conventional handover and the FlyBS handover.

5.2 Performance Metrics

The following performance metrics are defined to evaluate the performance of the proposed solution.

Main metric is a mean capacity of all UEs. The capacity is obtained from the maximum channel capacity defined by Equation *(3)*. In this diploma thesis we presume no losses due to the signaling overhead, transmission errors or protocol inefficiencies, thus the *loss coefficient* L $\left[\frac{\%}{100}\right]$ is set to 0.

Another performance metric is the number of handovers, both UE-based handovers and FlyBS-based handovers, performed over the course of all the simulations and time steps. However, the number of handovers itself is not the only indicator of network performance.

Another performance indicator is the number of ping-pong handovers. Ping-pong handovers are defined as frequent handovers from one BS to another and then back to the original BS in a short amount of time. Ping-pong handovers are best to be avoided as every handover generates a certain amount of signaling messages, which lowers overall capacity and increase energy costs for the involved UEs. Handover is labeled as ping-pong handover if the UE connects to the target BS and then back to its original serving BS in less than a critical time, labeled as *minimum time-of-stay*. To be able to compare the number of ping-pong handovers between different simulation scenarios, we define the *handover ping-pong ratio* (HPR) $\left[\frac{\%}{100}\right]$, which is defined as:

$$HPR = \frac{N_{PP}}{N_{total}}, \quad (8)$$

where N_{PP} [-] is the number of ping-pong handovers and N_{total} [-] is the number of all handover.

All this is followed by the R function. While not applicable for approaches, which do not take advantage of the reinforcement learning, it can serve as a miscellaneous metric, which shows the relation between number of handovers and capacity.

Table 1 Simulation Parameters

Parameter	Value
Simulation Area	2 × 2 km
Carrier Frequency	1800 MHz
Tx Power of GBS/FlyBS	23/15 dBm
Bandwidth of GBS/FlyBS	100 MHz
Number of UEs[8]	165
Hysteresis	3 dB
TTT	0.05 s
Time Step	0.01 s
CIO Set	{-6, -3, 0, 3, 6} dB

[8] This number of UEs is valid for 3 GBS and 2 FlyBS. The total number of UEs is calculated as 45 per GBS and 15 per FlyBS.

5.3 Simulation Scenarios

In this diploma thesis, the proposed handover decision optimizing algorithm is compared with other state-of-the-art approaches in different simulation scenarios.

5.3.1 Fixed CIO

This approach, based on the work of M. Najla et al., 2020 [1] is different from the proposed solution by setting the CIO to fixed integer value, thus circumventing the reinforcement learning all together. This applies to both ground and flying base stations. The CIO value is selected randomly from the set of CIO values at the start of each simulation and retained for the remainder of simulation. With the addition of FlyBSs to the mobile network, we still need to utilize the deep learning algorithms and DNN. As described in the previous sections, this is for the purpose of predicting the channel quality between flying and ground base stations based on the channel quality between GBSs and UEs. This approach should yield lower capacity values than the proposed solution as the reinforcement learning is not present and CIO values are fixed.

5.3.2 No Machine Learning

This approach, based loosely on the network model presented in the work of T. Sap, 2022 [4] is different from the proposed solution by the absence of machine learning altogether. At the start of each simulation the CIO parameter of all the BSs is set to a fixed integer value selected randomly from a predefined set of CIO values. The CIO value remains the same for the duration of the simulation, as in the previous *Fixed CIO* approach. However, in this approach the deep learning and DNN are absent. The channel quality between FlyBSs and UEs is measured, not predicted, by the UEs in the same manner as the channel quality between GBSs and UEs in a conventional network. This approach should, theoretically, yield higher capacity than the proposed solution, but as described in the earlier chapter, it is compensated by a higher energy cost for the UEs, thus making it a less desirable option.

5.3.3 Association

The *Association* approach replaces the conventional handover decision with a pure association process, with respect to the base station load. Every UE is connected to GBS with the highest channel quality, either directly or through FlyBS. If the target GBS is at its maximum load, UE connects to the GBS with the second highest channel quality and so on. This approach uses no reinforcement learning to optimize the BS CIO values, as the association uses no CIO values during handover. To use the deep learning to predict the FlyBS to UE channel quality also has no real advantages, as this approach is mainly demonstrative. Thus, the FlyBS to UE channel quality in this approach is measured and not predicted. This approach serves mainly as proof that the defined network model works correctly. This scenario sets the upper limit to the capacity values of all the state-of-the-art approaches as it eliminates the variance brought to the simulations by the handover decision process. However, its use in real mobile networks is highly improbable as the number of handovers while using this approach should increase dramatically. Due to the increased energy costs and signaling, the increased number of handovers in the mobile network is undesirable.

5.3.4 Other possible approaches

Another possible approach to consider would be approach presented by A. Madelkhanova et al., 2022 [8] with measured channel quality from FlyBSs to UEs and CIO parameter set by the Actor-Critic reinforcement learning. However, this comparison seemed unnecessary. The only relevant difference between the approach proposed in this diploma thesis and the approach from [8] is the FlyBS to UE channel quality. As the channel quality in [8] is fully measured, thus not affected by prediction error, it is bound to perform better in terms of capacity than the approach proposed in this diploma thesis. On the other hand, the solution proposed in this diploma thesis should theoretically perform better in terms of energy cost. This is due to the fact that the FlyBS to UE channel quality is predicted and thus pose no additional energy costs to the UEs.

However, as mentioned in the previous sections, both the DNN training optimization and UE energy consumption are beyond the scope of this diploma thesis. With this in mind, the comparison between proposed solution and the approach proposed in [8] make little sense. It remains a candidate for future works, where energy costs or DNN training optimization might be considered.

Main differences between all considered approaches are highlighted in the Table 2

5.4 Used Equipment

Simulations were done in the MATLAB R2022a Computing Environment by MathWorks with MATLAB Parallel Computing Toolbox [18] enabled. Simulations were done on the Personal Computer equipped with the CPU Intel(R) Core(TM) i7-7700HQ CPU @ 2.80GHz, GPU NVIDIA GeForce GTX 1050 Ti and 16 GB of RAM.

Table 2 State-Of-The-Art Approaches

Approach	Channel Quality	CIO Setting
Proposed Solution	Predicted by DNN	Set by RL Agent
Fixed CIO	Predicted by DNN	Fixed, selected randomly
No Machine Learning	Measured	Fixed, selected randomly
Association	Measured	N/A

6 SIMULATION RESULTS

In this section, the results of the simulations of the proposed solution are discussed. The proposed solution is also compared to other state-of-the-art approaches, while discussing the impact it has on the network and UEs. The section is divided into several subsections based on used metric i.e., capacity, handovers and R function.

6.1 Capacity

One of the main metrics to consider in mobile networks is the average capacity. Figure 3 depicts the main performance metric, the average capacity across all simulations and connected UEs with the number of FlyBSs as a parameter. The proposed solution is then compared with other mentioned approaches.

As we predicted in the previous section, the proposed solution is not the best in terms of the average capacity. However, it is important to note that the only better performing approach, namely the *Association* approach serves only as an upper bound to all simulation results. Despite that, the proposed solution is on average only 5 % below the *Association* approach and works the best out of all competitive state-of-the-art approaches.

The *No Machine Learning* approach, while useable in real networks, with only an average decrease of 3.9 % compared to the proposed solution, is not deemed ideal, because it imposes higher energy consumption to the UEs in the network. However, the measuring of energy costs is not in the scope of this diploma thesis, but it might be discussed in the next works. In addition to the higher energy costs, another obstacle in the use of this approach in real networks is the usage of FlyBS as a transparent relay. When the FlyBS works as a transparent relay, measuring the channel quality is impossible, as the FlyBS is not visible to the UE.

Lastly, the proposed solution is also better than the *Fixed CIO* approach. The average capacity of the proposed solution has increased by 9.2 % when compared to the *Fixed CIO* approach. This result is to be expected, as the *Fixed CIO* approach lacks the CIO optimization the proposed solution achieves via the reinforcement learning.

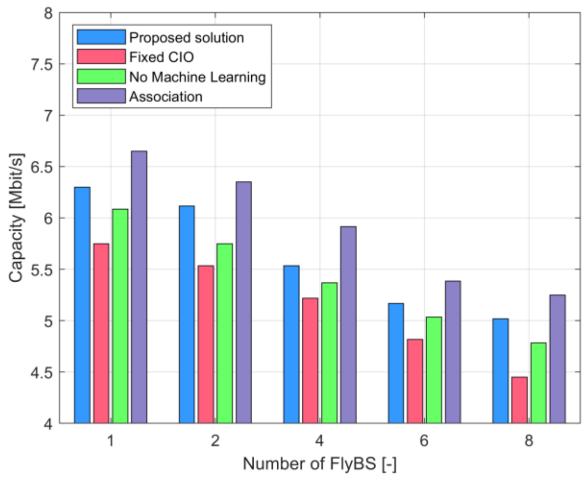


Figure 3 Impact of number of flying base stations on an average capacity. Capacity is averaged across all simulations and connected UEs. In this figure we compare different approaches.

6.2 Ping-Pong Handovers

When dealing with the handover optimization, the number of handovers itself can be misleading. When we deal with CIO optimization, the especially important performance metric is the number of ping-pong handovers or better yet the HPR ratio, which is more comparable among the different approaches. Figure 4 depicts the HPR ratio with the number of FlyBSs as a parameter.

The proposed solution boasts the lowest value of HPR of all the compared approaches. This shows that the proposed solution is truly working as intended as it lowers the number of redundant handovers when compared to other simulated approaches.

On the other hand, the *No Machine Learning* approach has on average the highest HPR ratio, more than double that of the proposed solution, with *Fixed CIO* approach being the second highest and *Association* approach the third. This is to be expected as all three approaches use fixed CIO values and lesser degree of machine learning-based optimization than the proposed solution.

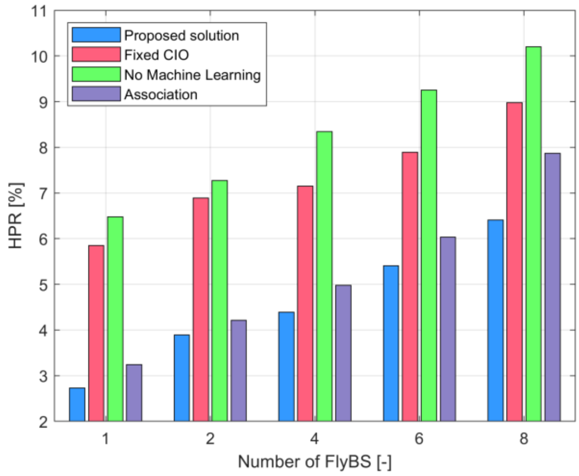


Figure 4 Impact of number of flying base stations on HPR. The value of the HPR is averaged across all simulations and connected UEs. In this figure we compare different approaches.

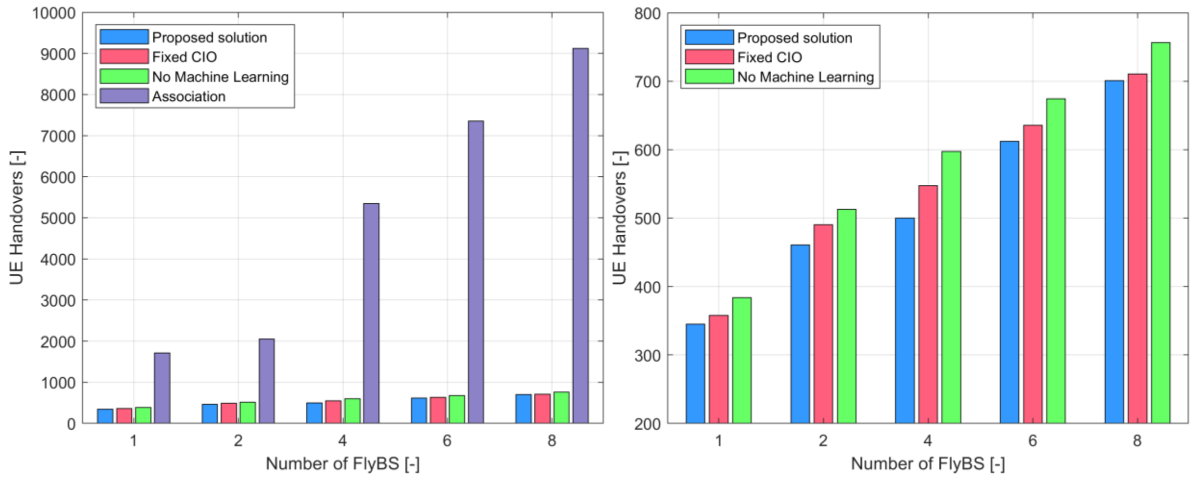
6.3 UE Handovers

Another network performance metric to consider is the number of handovers, in this case UE handovers. Although not as important as HPR ratio, the number of handovers is useful as well, as lower number of handovers means less energy spent on the UE side. These are the conventional handovers as we know them from older generations of mobile networks. However, it is important to note that this metric includes the UEs which did not perform handover on their own but rather were handed over to other GBS by the FlyBS handovers. Figure 5 depicts the number of UE handovers with the number of FlyBSs as a parameter.

As we expected, the *Association* approach has by far the highest number of handovers. The increase in handovers is in the hundreds of per cent compared to the proposed solution. However, as mentioned in the previous subsection, this approach is not really applicable in the real mobile network.

As for the other approaches, their values are comparable with one another. The proposed solution comes on top with an average 8 % decrease in the number of handovers when compared to the *Fixed CIO* and *No Machine Learning* approaches. This is to be expected, as the proposed solution is designed, among other things, to prevent redundant handovers.

Another thing to notice is the overall increase in the number of handovers across all approaches. However, this is an expected behavior. As the number of FlyBS increases, the chance for a FlyBS handover increases as well, which is projected in the overall increase of the number of handovers.



(a) *(b)*

Figure 5 Impact of number of flying base stations on a number of handovers. Handovers are averaged across all simulations and connected UEs. In this figure we compare different approaches. (a) number of handovers across all approaches, (b) number of handovers without Association approach for more detail.

6.4 FlyBS Handovers

With the introduction of FlyBSs into the mobile network we also need to consider the number of FlyBS handovers for the performance evaluation. These handovers are performed by the FlyBSs between their serving GBS and target GBS, while carrying all their connected UEs with them to a new GBS. Figure 6 depicts the number of FlyBS handovers with the number of FlyBSs as a parameter.

Here a similar pattern to that of the conventional UE handovers is depicted. The *Association* approach has on average the highest number of FlyBS handovers of all the considered approaches. However, unlike the conventional UE handovers, the increase here is not in the hundreds of per cent. The increase over the proposed solution is 92 %. Once again it is important to note that the *Association* approach is not applicable in real networks due to the extreme number of handovers and the energy costs and signaling that comes with it.

As for the other approaches, the average number of FlyBS handovers is comparable to one another with the proposed approach having on average the lowest number of FlyBS handovers and *No Machine Learning* the highest number of FlyBS handover with the average increase of 14 % over the proposed approach.

The general trend in the number of FlyBS handovers is similar to that of conventional UE handovers. The overall number of FlyBS handovers is increasing with the number of FlyBSs. With the increased number of FlyBSs, the chance for FlyBS handover has also increased, so this trend is to be expected.

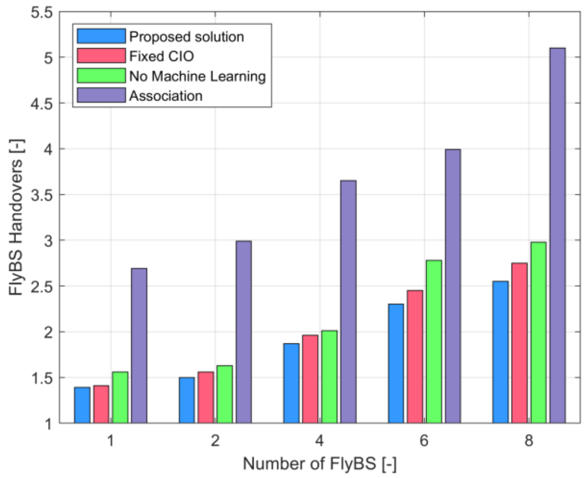


Figure 6 Impact of number of flying base stations on a number of FlyBS handovers. FlyBS handovers are averaged across all simulations and connected UEs. In this figure we compare different approaches.

6.5 Reward Function

When compared to other performance metrics like capacity or HPR, the R function is not as important, but it helps to complete the whole picture. Another way to look at the R function is an inversion to the number of handovers. The way the R function is defined, its value lowers with the increase in the number of handovers. The values of the R function are depicted in Figure 7.

The lowest value of the R function corresponds to the *Association* approach as it has by far the highest number of handovers. The highest values of the R function correspond to the *No Machine Learning* approach. The proposed solution shares similar values with the *Fixed CIO* approach. However, it is worth noting that the R function is truly used only when reinforcement learning is present in the simulation scenario. This means that apart from the proposed solution, the R function is only illustrative.

For the proposed solution, *No Machine Learning* approach and *Fixed CIO* approach the value of R function increases steadily with the number of FlyBSs. While the average capacity decreases with the number of FlyBSs, and subsequently the number of UEs, the number of handovers, on the other hand, increases. The increase in the values of the R function might indicate that the network quality is actually increasing with the number of FlyBSs, despite the decrease in average network capacity. In other words, the handover-capacity ratio is increasing, as an analogy to the price-performance ratio. This further illustrates the disadvantage of the *Association* approach as it drops quickly with the increasing number of FlyBSs.

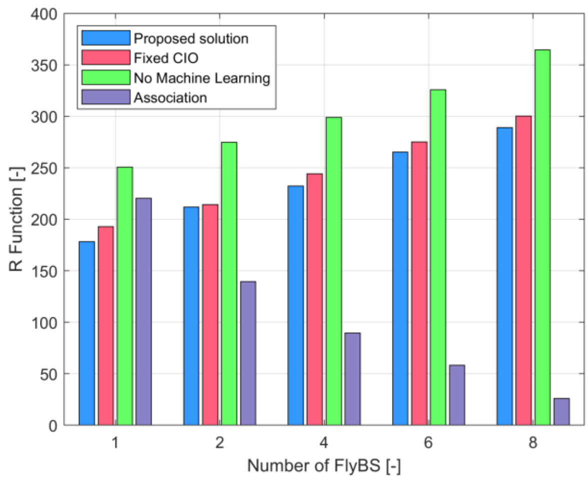


Figure 7 Impact of number of flying base stations on an R function. The value of the R function is averaged across all simulations and connected UEs. In this figure we compare different approaches.

7 CONCLUSION

In this diploma thesis, a novel approach has been proposed, which combines the reinforcement learning-based CIO optimization for management of handovers and deep learning-based channel quality prediction for FlyBS to UE channels. The solution is part of an entire proposed workflow, which encompasses two distinct steps. The first step of the solution is predicting the FlyBS to UE channels based on the information already known to the network, i.e., the GBS to UE channel. After that, a reinforcement learning agent sets CIO values of the network BSs based on the immediate state of the network, including the FlyBSs with predicted channel qualities.

The proposed solution has been compared with several state-of-the-art approaches. In terms of capacity, the proposed solution fared very well, being the best of all the state-of-the-art approaches. The average network capacity when compared with other approaches has been increased by up to 10 %, while also lowering the raw number of handovers by up to 15 %. The number of ping-pong handovers has even decreased multiple times.

The only approach with higher average network capacity than the proposed solution is the upper-bound *Association* approach. However, this upper-bound approach is plagued by several critical disadvantages, which limits its application to real networks due to an extreme number of handovers and consequent energy costs.

When the R function is taken into consideration for performance, the proposed solution is the second worst in terms of raw numbers. However, as other compared approaches are not reinforcement learning-enabled, the comparison of R function raw numbers carries no useful information. What is more important is the overall trend of the R function, which grows with the number of FlyBSs. The growing trend might indicate an increase in the quality of the network.

The diploma thesis can be further enhanced towards a consideration of more parameters, which might influence the results of the proposed solution, like handover cost or energy cost. Another future work might consider a more complex terrain, like an urban scenario with different density of city infrastructure. Future enhancements to this diploma thesis might also focus more on a single element of the proposed solution, like deep learning predictions and the associated prediction accuracy, which might be influenced by the number and size of the training data sets.

APPENDIX A

A.1 Custom MATLAB Scripts

Script 1 *a_MAIN.m*

Main script for the simulations presented in this thesis. Selection of different approaches is done via script variables or in the global_variables.m function.

Script 2 *b_PLOTTER.m*

Script for plotting, i.e., presenting all simulated scenarios.

A.2 Custom MATLAB Functions

Function 1 *data_for_dnn.m*

This function prepares data sets for the training of the DNN network.

Function 2 *dnn_learning.m*

This function creates the DNN network and trains it using the data from data_for_dnn.m function.

Function 3 *global_variables.m*

This function is the main control function for the solution workflow. It sets all variables needed by every script and function in this thesis, like number of UEs, BSs or UE and BS parameters.

Function 4 *reinfocement_learning.m*

This function creates the RL environment and agent. It also trains the Actor-Critic networks of the reinforcement learning agent by the MATLAB native function train.m.

Function 5 *reset_model.m*

This function resets the defined model, i.e., sets all needed variables and matrices to default values.

Function 6 *step_model.m*

This function serves as the main function for iterating the model, i.e., it defines the model, movement, channel or handover procedure.

Function 7 *step_model_loop.m*

This function serves as the main loop for non-reinforcement learning approaches. The reinforcement learning-enabled approaches use the step_model.m iterated over by the MATLAB native sim.m function.

A.3 Others

File 1 *README.pdf*

README file containing all necessary information for running and controlling the simulation functions and scripts.

A.4 Additional Information

All the digital media are contained in a ZIP repository, which is included with this thesis.

Custom scripts included in this thesis were created in the version R2022a of the MathWorks MATLAB environment. Other used scripts are also compatible with this version.

REFERENCES

[1] M. Najla, Z. Becvar, P. Mach and D. Gesbert, "Integrating UAVs as Transparent Relays into Mobile Networks: A Deep Learning Approach," in *2020 IEEE 31st Annual International Symposium on Personal, Indoor and Mobile Radio Communications*, London, UK, 2020.

[2] Y. Zeng, R. Zhang and T. J. Lim, "Wireless communications with unmanned aerial vehicles: opportunities and challenges," *IEEE Communications Magazine,* vol. 54, no. 5, pp. 36-42, 2016.

[3] X. Li, H. Yao, J. Wang, X. Xu, C. Jiang and L. Hanzo, "A Near-Optimal UAV-Aided Radio Coverage Strategy for Dense Urban Areas," *IEEE Transactions on Vehicular Technology,* vol. 68, no. 9, pp. 9098-9109, 2019.

[4] T. Sap, "Deployment of Flying Base Stations in Emergency Situations," *Diploma Thesis,* pp. 1-73, 2022.

[5] Y. Chen, W. Feng and G. Zheng, "Optimum placement of UAV as relays," *IEEE Communications Letters,* vol. 22, no. 2, pp. 248-251, 2018.

[6] S. Zeng, H. Zhang, K. Bian and L. Song, "UAV Relaying: Power Allocation and Trajectory Optimization Using Decode-and-Forward Protocol," in *IEEE ICC Workshops*, 2018.

[7] M. Najla, Z. Becvar, P. Mach and D. Gesbert, "Positioning and Association Rules for Transparent Flying Relay Stations," *IEEE Wireless Communications Letters,* vol. 10, no. 6, pp. 1276-1280, 2021.

[8] A. Madelkhanova, Z. Becvar and T. Spyropoulos, "Optimization of Cell Individual Offset for Handover of Flying Base Stations and Users," *IEEE Transactions on Wireless Communications,* 2022.

[9] 3GPP, "Specification #: 23.009 Handover procedures," 3GPP, 1999.

[10] A. Madelkhanova, Z. Becvar and T. Spyropoulos, "Q-Learning-based Setting of Cell Individual Offset for Handover of Flying Base Stations," in *IEEE Vehicular Technology Conference (IEEE VTC2022 Spring)*, 2022.

[11] M. Najla, Z. Becvar, P. Mach and D. Gesbert, "Predicting Device-to-Device Channels from Cellular Channel Measurements: A Learning Approach," *IEEE Transactions on Wireless Communications,* vol. 19, no. 11, 2020.

[12] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural Computation,* vol. 9, no. 8, pp. 1735-1780, 1997.

[13] B. Ramsundar and B. R. Zadeh, TensorFlow for Deep Learning, O'Reilly Media, Inc.

[14] O. Fontana, "Understanding the difference between supervised and reinforcement learning for deep neural networks," Neal Analytics/Fractal, 07 02 2023. [Online]. Available: https://nealanalytics.com/blog/understanding-the-difference-between-supervised-and-reinforcement-learning-for-deep-neural-networks/. [Accessed 24 05 2023].

[15] S. Bhatt, "towardsdatascience.com," Towards Data Science, 19 03 2018. [Online]. Available: https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292. [Accessed 21 05 2023].

[16] MathWorks, "MATLAB Reinforcement Learning," MathWorks, 07 2019. [Online]. Available: https://www.mathworks.com/products/reinforcement-learning.html. [Accessed 21 05 2023].

[17] P. Viktorin, "naucse.python.cz," PyLadies CZ, 2019. [Online]. Available: https://naucse.python.cz/2019/advanced-en-prague/advanced-en/asteroids/. [Accessed 21 05 2023].

[18] S.-L. Su, T.-H. Chih and S.-B. Wu, "A novel handover process for mobility load balancing in LTE heterogeneous networks," in *IEEE ICPS*, 2019.

[19] MathWorks, "MATLAB Parallel Computing Toolbox," MathWorks, 2019. [Online]. Available: https://www.mathworks.com/products/parallel-computing.html. [Accessed 24 05 2023].