HANDOVER WITH NETWORK SLICING IN 5G NETWORKS

by

Kübra Sevim

B.S., Computer Engineering, Boğaziçi University, 2017

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering
Boğaziçi University
2021

ACKNOWLEDGEMENTS

I would like start by thanking to my thesis advisor Prof. Tuna Tuğcu for his guidance. His knowledge, experience and support allowed me to widen my perspective in many areas throughout my master's degree.

I would also like to thank the members of my thesis committee, Prof. Cem Ersoy and Prof. Sema Oktuğ, for participating in my thesis defence, and their valuable evaluations.

Lastly, I would like to thank my husband, Oğuzhan Sevim, and my family for their support and love.

ABSTRACT

HANDOVER WITH NETWORK SLICING IN 5G NETWORKS

$5G$ suggests many advantages but these advantages bring some problems to be solved. Network Slicing is one of the key concepts that $5G$ introduces. Slicing enables having multiple isolated virtual networks on top of the same physical infrastructure. Thus, each slice can provide different services with diverse Quality of Service (QoS) requirements. In $5G$, Software Defined Networking (SDN) and Network Function Virtualization (NFV) are critical to support network slicing.

In the literature, several problems of network slicing are studied. Two outstanding areas of focus are admission control and resource allocation. Most of the studies are on the Core Network resources although it is essential to investigate radio resource allocation in order to maintain an end-to-end isolation for slices. While there are considerable contributions around Radio Access Network (RAN) resource allocation, optimizing the throughput of the network is not fully achievable via admission control.

In this thesis, we mainly focus on handover to maintain the usability and high utilization of the radio resources of the networks. When the number of users within one cell suddenly increases up to the limit of the base station, all of the incoming requests may not be handled and most of the users may suffer from not being able to use the offerings of the slices that they demand. We develop an optimization problem to optimize the radio resources and propose an heuristic to reach similar results by leveraging handover in considerably low computing times and with the simulation results we show that our heuristic can present solutions close to the optimal within a short time frame.

ÖZET

5G AĞLARI İÇİN AĞ DİLİMLEME VE AKTARIM

5Gnin ortaya koyduğu birçok avantaj bulunuyor, ayni zamanda bu avantajlar çözülmesi gereken yeni problemleri beraberinde getiriyor. 5G'nin desteklediği temel kavramlardan biri Ağ Dilimlemedir. Dilimleme, aynı fiziksel altyapı üzerinde birden çok izole edilmiş sanal ağa sahip olmayı sağlar. Böylece, her ağ dilimi, çeşitli hizmet kalitesi (QoS) gereksinimlerine göre farklı hizmetler sağlayabilir. 5G'de, Yazılım Tanımlı Ağ İletişimi (SDN) ve Ağ Fonksiyonu Sanallaştırma (NFV), ağ dilimlemeyi desteklemek için kritik öneme sahiptir.

Literatürde, ağ dilimlemenin çeşitli sorunları incelenmiştir ve odaklanan iki önemli alan, kabul kontrolü ve kaynak tahsisidir. Kaynak tahsisi çalışmalarının çoğu çekirdek şebekesi kaynakları üzerinedir, ancak ağ dilimleri için uçtan uca bir izolasyonu sürdürmek için radyo kaynak tahsisini de araştırmak önem arz etmektedir. Radyo Erişim Şebekesi (RAN) kaynak tahsisi alanında önemli katkılar olsa da, ağın verimini optimize etmek, kabul denetimi yoluyla tam olarak gerçekleştirilemez.

Bu tezde, radyo ağ kaynaklarının kullanılabilirliğinde yüksek verimlilik sağlanması için ağırlıklı olarak geçişe (handover) odaklanıyoruz. Bir hücre içerisindeki kullanıcı sayısı baz istasyonu sınırına kadar arttığında, gelen tüm talepler karşılamanmayabilir ve kullanıcılar, talep ettikleri ağ dilimlerini kullanamayabilir. Bu tezde, radyo kaynaklarının kullanımını optimize etmek için bir optimizasyon problemi geliştirip inceliyoruz ve benzer sonuçlara çok daha düşük hesaplama sürelerinde ulaşmak için geçiş tabanlı bir buluşsal yöntem öneriyoruz ve simülasyon sonuçlarıyla, buluşsal yöntemimizin kısa bir zaman çerçevesi içinde optimale yakın çözümler sunabildiğini gösteriyoruz.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

b_s^n	Aggregate data rate for slice s in cell n
c_k^n	Variable that indicates if a user k is in the coverage of cell n
$d_{k,s}$	Variable that indicates if a user k demands to use slice s
k	User index
K	Number of all users in the network
$\mathcal{K}$	Set of all users in the network
n	Cell index
N	Number of all cells in the network
$\mathcal{N}$	Set of all cells
$p_{k,s}$	Variable that indicates if slice s is provided to user k or not
r_s	Required data rate per user for slice s, where $s \in \mathcal{S}$
s	Slice index
S	Number of all slices
$\mathcal{S}$	Set of all slices
x_k^n	Variable that indicates if a user k is connected to cell n or not

LIST OF ACRONYMS/ABBREVIATIONS

$3GPP$	Third Generation Partnership Project
$4G$	Fourth Generation
$5G$	Fifth Generation
AF	Application Function
AKA	Authentication and Key Agreement
AMF	Access and Mobility Function
AUSF	Authentication Server Function
BS	Base Station
CUPS	Control and User Plane Separation
D2D	Device to Device
EPC	Evolved Packet Core
ETSI	The European Telecommunications Standards Institute
GBPS	Gigabits per second
GSM	The Global System for Mobile Communications
IoT	Internet of Things
MM	Mobility Management
MME	Mobility and Management Entity
NAS	Non-Access Stratum
NEF	Network Exposure Function
NRF	Network Function Repository Function
NSSF	Network Slice Selection Function
OFDMA	Orthogonal Frequency Division Multiple Access
PCF	Policy Control Function
PGW	PDN Gateway
PGW-C	PDN Gateway Control Plane Function
PGW-U	PDN Gateway User Plane Function
QoS	Quality of Service
RAN	Radio Access Network

SBA	Service Based Architecture
SGW	Serving Gateway
SGW-C	Serving Gateway Control Plane Function
SGW-U	Serving Gateway User Plane Function
SM	Session Management
SMF	Session Management Function
UDM	Unified Data Management
UE	User Equipment
UPF	User Plane Function

1. INTRODUCTION

In recent decades, mobile data traffic has been increasing exponentially. This growth is mainly due to the accelerating usage of smartphones and the enlarged data capacity for each mobile subscription. From the beginning of 2019 to the first quarter of 2020, the growth in mobile network data traffic is observed to be 56 percent [2]. Recently, the demand for video streaming applications has been rising and this is expected to continue in the upcoming years. Thus, it is predicted that video traffic will be the main cause for data usage and the demand for mobile data will rise by 31 percent annually until 2025 [2]. Besides, recent global events have been changing the user behavior in a way that the data demand is shifting beyond the estimations [3]. Due to the Covid−19 pandemic, which started in 2020, working and studying remotely has been tremendously boosted [4]. These global changes in user behavior have increased the overall data demand. Even though this increase is not observed particularly in the mobile data usage, the importance of being prepared for unexpected major events has become more clear than ever. Therefore, the advantages of Fifth Generation ($5G$), such as its flexibility and capacity promises, will not only enhance today's needs, but will surely play a critical role in these types of unforeseen events.

Similar to the transitions between earlier generations of the European Telecommunications Standards Institute (ETSI), $5G$ technology targets a smooth transition from Fourth Generation ($4G$). In the early stages, $5G$ radio network will be compatible with $4G$ core network [5]. Besides, $5G$ introduces new technologies and approaches not only in the radio network, but also in the core network. One of these introduced technologies is network slicing, which will be the main focus of this thesis.

Network sharing has been in place while two different approaches were considered as network sharing: passive and active. Passive sharing is related to the reuse of elements like cables, coolers, etc. Active sharing includes reuse of antennas, core network, etc. While implementing active sharing, spectrum usage can be arranged

either assigning different spectrum for each network or using one pool of spectrum for the entire network [6]. Third Generation Partnership Project ($3GPP$) specifications for network sharing are mentioned in detail in [7]. Five main scenarios are presented for radio resource usage. In the first scenario, Radio Access Network (RAN) sharing within various operators is described as possible in the level of RAN elements usage but spectrum sharing is not allowed, and different operators have their own dedicated spectrum. The second scenario includes various operators using their licensed frequency bands and together they provide a country wide coverage. In the third scenario, one operator maintains a certain area by providing subscriptions to other operators. The fourth scenario is described as one operator sharing its spectrum with the other operators or as all of the operators in that area using a pool of spectrum by consolidating their individual spectrum. Scenario 5 is about various RANs using the same core network. In the same study, mobile network virtualization is then modeled by applying network virtualization to the $4G$ orthogonal frequency division multiple access (OFDMA) based BSs. Also, the slices are identified as a virtual network that uses the same physical BSs for various virtual networks. The key points to be provided in this network are the isolation of the slices, customization, and utilization of RAN resources.

Dividing the network to meet different types of requirements is an outstanding goal for $5G$ [8]. With the introduction of network slices, one physical infrastructure can be used to provide various services and each network slice resources can be arranged to support different requirements. If this is successfully adjusted, user experience will be advanced and the cost of providing service for users will be reduced significantly.

Network slicing is the concept of having multiple virtual networks that use the same physical infrastructure [9]. The virtual networks are planned to serve different services, applications, devices, or customers with various needs. In case of $5G$, the slices can use the same RAN or different RANs depending on the architecture design. In general, slicing can be used to partition the core network. Network slicing is a critical feature of $5G$ due to the variety of services that $5G$ offers. By using network slicing, different communication needs (e.g., IoT, mobile devices, vehicle communications, and

robots) can be met efficiently. In a $5G$ network that uses slicing, resource allocation can be done by changing the slice widths [10, 11] or by using handover algorithms [12]. For separate slices, latency, throughput, and the usage of bandwidth can be arranged differently by resource allocation. Network slicing implementation can be exemplified as shown in Figure 1.1. In [13], two approaches have been mentioned for the slice selection process. In the initially suggested approach, User Equipment (UE) holds the information about the slices it is requesting and sends that information to the network, which then assigns the corresponding slice. In the second approach, the core network assigns the slices to the UEs according to the service request coming from the UE and the availability of the respective slices. In this thesis, we will apply the second approach. A UE may use more than one slice concurrently by using a single AMF [14].

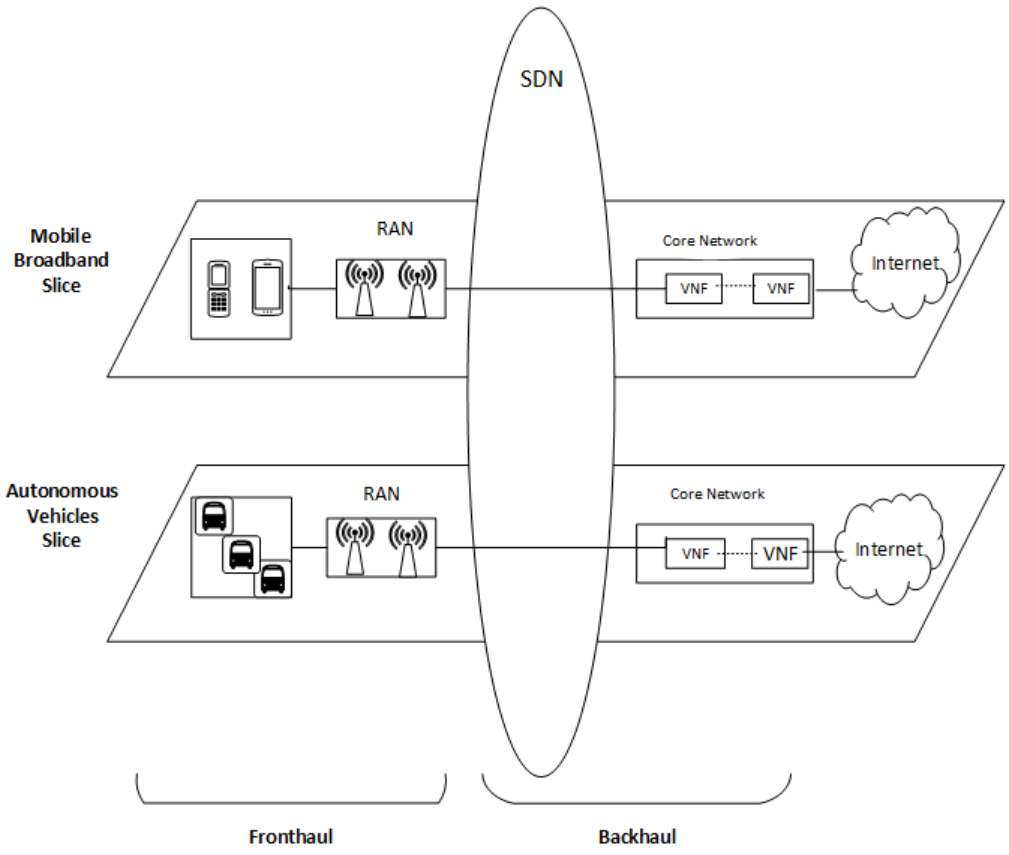

Figure 1.1. Network Slicing Implementation (inspired from [1]).

There are two approaches for providing network slice isolation in the RAN. One is to use suitable mechanisms that allow slices to share the RAN without intercepting each other (e.g. scheduling) while the second approach is to reserve a dedicated spectrum for a slice [8,14]. In this thesis, we apply the second approach to maintain the isolation between slices.

The aim of this thesis is first to define a joint handover/slice allocation problem under an optimization framework for $5G$ networks. This problem is defined to improve RAN resource utilization for the network slices to maximize active slice connections. Then, we try to find pseudo optimal solutions for this problem by proposing heuristic algorithms. By using these heuristics, we aim to decrease the computation time compared to the conventional solution methods for the defined problem. We show that a slight decrease in the optimality results in a high gain for the execution time.

1.1. Basic Information on 5G

$5G$ supports various applications such as Internet of Things (IoT), Device to Device ($D2D$) communications, virtual reality devices, and etc. Supporting high number of connected devices and high data rates is exceeding the limits of $4G$. Therefore, a list of requirements have been defined for $5G$ to provide service for diverse applications. The eight main targets of $5G$ can be listed as follows [15]:

- $1 - 10$ GBPS data rate per user,
- 1 millisecond latency,
- almost 100% availability,
- nearly 100% coverage,
- around 90% energy optimization,
- increased battery life,
- low power consumption,
- support high number of device connections.

These requirements are aimed to be fulfilled by using various methods, which will be discussed later in this chapter. Currently, the allocated bandwidth for wireless transmission is within the range of 300 MHz to 3 GHz [15]. This spectrum range may not be sufficient to meet the increasing demand for more connections and higher data rates. Thus, for $5G$, higher spectrum bands are studied to support this need [16]. Unused $3 - 300$ GHz (millimeter-wave) band is proposed to provide higher data rates [17].

Spectral efficiency in $4G$ is maintained by orthogonal frequency division multiplexing (OFDM) and orthogonal OFDMA. For $5G$, new modulation and multiple access techniques have been investigated to enhance the spectral efficiency. Some of these techniques are filter bank multi-carrier (FBMC), generalized frequency division multiplexing (GFDM), universal filtered multi-carrier (UFMC), and bi-orthogonal frequency division multiplexing (BFDM) [18].

Some applications (e.g., self driving cars and autonomous vehicles) can not tolerate delays higher than 1 ms, $5G$ plays an important role in these new technologies. Massive Multiple Input Multiple Output (MIMO) systems and SDN architecture are among the techniques that are used to support higher data rate and low delay requirements of $5G$ networks. SDN presents the separation of control planes and data planes. Network components are designed to have basic functionalities on traffic forwarding, the control functionality are separated from those devices [19]. In addition, traffic forwarding is based on flow tables [20]. Flow represents the set of packages from one source to one destination. Detached control plane functionality is implemented as SDN controller or Network Operating System (NOS). NOS is a software that aims to provide the environment to program the functionality of network forwarding devices [21]. The communication between the software programs on NOS and data plane allows the network to be programmable.

In order to increase coverage and availability in $5G$ networks, small cells (e.g., femtocells and picocells) will be used together in so called heterogeneous communication networks [22]. These types of networks, where base stations can be self-organized [23], can provide high data rate up to GBPS by decreasing transmitter-receiver distances. Also, the power of heterogeneous $5G$ networks can further be increased by involving the computational power of cloud services. CloudRAN is introduced to manage the operations that are performed in classical RANs so that the antenna will only perform the signal transmission [24].

The new technologies introduced for the core network typically target higher efficiency and effective network management. In this respect, virtualization comes in handy, especially with the improvements in hardware and software for system virtualization [25]. Flexibility, scalability and availability are some of the key factors that $5G$ focuses on by virtualization. NFV enables virtualization of the functions and services that are provided by physical servers [26]. Its target is to implement these physical server functions in software, thus it introduces the opportunity of not being hardware dependent for the network providers. In addition, improvements on hardware and software can be done separately due to separation of functions from hardware. NFV also supports faster network deployment and adjustable installation of network functions which will help in reuse of hardware with different functionalities [27].

1.2. 5G Architecture

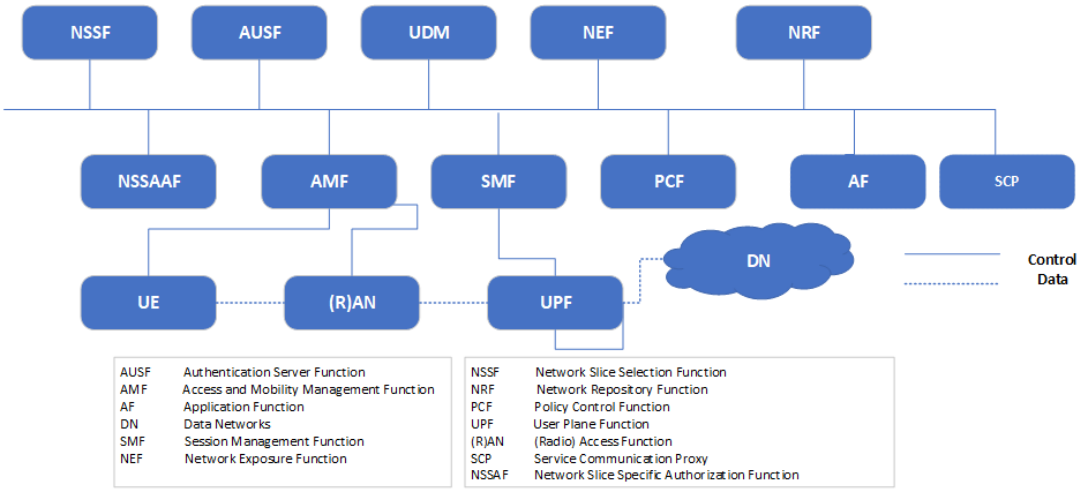

Figure 1.2. 5G System Architecture.

On top of 4G, many improvements have been made on 5G architecture. One of the main differences between two architectures is that 5G is designed with control and user plane separation (CUPS) where there exist distinct service components that serve for different functions [28]. CUPS enables separation of user plane functions (e.g., user data traffic) and control plane functions such as authentication and connection. This separation provides capability of changing the capacity when it is needed without affecting the control plane functionality. By this separation scheme, Serving Gateway (SGW) in 4G is split and reorganized as Serving Gateway Control Plane Function (SGW-C) and Serving Gateway User Plane Function (SGW-U). In addition, PDN Gateway (PGW) is reorganized as PDN Gateway Control Plane Function (PGW-C) and PDN Gateway User Plane Function (PGW-U) [29].

5G service based architecture can be seen in Figure 1.2 [30]. The 5G components in the non-roaming state, are connected as shown in Figure 1.3 [30]. When installed, Service Communication Proxy (SCP) might function as the bridge for network functions and network function services.

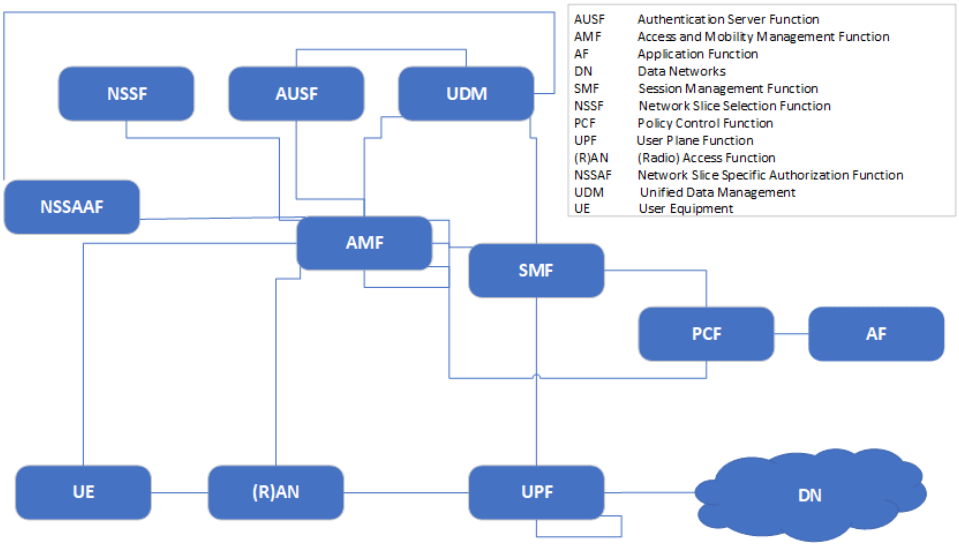

Figure 1.3. 5G Non Roaming Architecture in Reference Point Representation.

The details of the major 5G architecture components can be listed as follows [31]:

- **Access and Mobility Function (AMF):** In 4G, Mobility and Management Entity (MME) in Evolved Packet Core (EPC) is responsible for the core functions in EPC. The MME manages session states, authentication, paging, roaming, and performs other mobility functions. AMF in 5G performs similar operations with MME. AMF is responsible for the following: ending the RAN control plane interface, Non-access stratum (NAS) signaling, signal integrity protection, authentication and authorization of UE, security, registration, connection management, reachability management, and mobility.
- **Session Management Function (SMF):** SMF maintains the functions that are required for session management. The role of SMF in 5G can be listed as follows: assigning IP addresses to UEs, NAS signaling to manage the session, informing the RAN about Quality of Service (QoS) and assigned policies, choosing and managing User Plane Function (UPF) for traffic routing, determining how the policy and charging will be available for UE.
- **User Plane Function (UPF):** SGW-U and PWG-U together build the User Plane Function (UPF). The responsibilities of UPF are: routing and forwarding

the packages, controlling and classifying the packages. In other words, UPF serves as the data plane for $5G$ architecture.

- **Policy Control Function (PCF):** PCF in $5G$ determines the policy rules for network slicing, roaming and mobility management. Policies and charging control functions are provided through PCF.
- **Authentication Server Function (AUSF):** AUSF is responsible for maintaining the processes related to authentication.
- **Application Function (AF):** This component manages the application effect on traffic routing and it cooperates with the PCF for policy checking.
- **Unified Data Management (UDM):** UDM analyzes the data related to UEs. The identification,authorization and subscription of a UE are managed by UDM and this component is also responsible for creating the Authentication and Key Agreement (AKA) credentials.
- **Network Repository Function (NRF):** NRF organizes network functions and service registrations. To organize network functions, it creates a discovery function so that network functions can recognize one another [32].
- **Network Exposure Function (NEF):** NEF acts as a bridge between the core network and outside applications. It enables secure data access for an outside application or entity. In other words, exposing events, services and capabilities are performed by NEF.
- **Network Slice Selection Function (NSSF):** NSSF is responsible for selecting the proper network slice for a UE and directing its traffic towards that slice. In addition, NSSF also decides which AMF will be used for the UE.

1.3. Contribution of the Thesis

The contributions of this thesis can be summarized as follows:

- We define an optimization problem that targets to maximize the total number of slice connections in the $5G$ networks. The main difference between our problem and the majority of the problems defined in the literature [33–35] is that, our

problem considers users' slice demands, required data rates for each slice usage, and the existing capacity of the slices in all BSs. By doing so, we find a good assignment BS for the users and increase the overall slice utilization.

- For the defined mixed-integer nonlinear optimization problem, a linearization method is employed such that the resulting mixed-integer linear problem can be solved by existing solvers within a shorter time frame.
- In order to solve the presented optimization problem, three heuristic methods are developed. One of the heuristics focuses on assigning the users to the emptiest BSs to maximize their slice usages. The remaining two algorithms employ different handover methods to improve the RAN utilization and the number of total slice connections.
- We solve the defined problem with the well known optimization toolbox called Gurobi and compare the solutions with the proposed heuristics. Success of the algorithms is illustrated with simulations. The simulations are carried out under two different network cases. A generic network is considered in the first case while a more specific user density is assumed in the second one. In the performance comparisons, we mainly consider three criteria: runtime, the total number of slice connections, and overall slice utilization. In addition, some simulations regarding the user mobility analysis are performed. In the mobility analysis, we also considered the total number of handovers and the total number of dropped slice connections. We demonstrate that the Intelligent Handover Algorithm is a good method to solve our defined problem.

1.4. Organization of the Thesis

The remaining parts of this study can be summarized as follows: Some of the related studies in the literature is given in Chapter 2. These works are presented by their problem definitions and their solution methods.

In Chapter 3, the mathematical preliminaries and the $5G$ system model is introduced first. Then, the joint handover/slice allocation problem is formulated as an

optimization problem. After some theoretical analysis regarding the convexity of the defined problem, some possible solution methods, including linearization, are discussed. Finally, a widely used optimization toolbox named Gurobi is presented along with its performance on a toy problem.

In Chapter 4, details of the heuristic algorithms are discussed and presented as pseudo-codes. Lastly, time complexity analysis of the proposed algorithms are discussed.

The performance of the proposed algorithms along with the Gurobi toolbox are illustrated by simulation results in Chapter 5.

Finally, some concluding remarks with the possible future research directions are discussed in Chapter 6.

2. RELATED WORK

In the literature, there are not a lot of studies around network slicing resource allocation problem. In one of the recent articles [36], slice selection access problem brought by the handover of UEs is studied. The goal of the defined problem is to maximize the system throughput. To solve the defined problem, a genetic based heuristic algorithm is proposed and its performance is compared with a random assignment algorithm and greedy algorithm. When the UEs in the network are sparse, there is no visible advantage of the proposed heuristic, however as the number of UEs increases, handover success rate and throughput become higher with the genetic algorithm.

In [37], the slice allocation/admission problem is approached by using a slice demand forecasting scheme. By using the Holt-Winter method, which is used for time-series forecasting, the future demands for network slices are predicted. The obtained predictions are then combined with an heuristic method to solve the resulting Knapsack problem that aims to maximize total utilization of the network. The authors show that the advantage of forecasting scheme increases the utilization as the slice demands and network capacity increases.

Authors in [38] formulate the slice allocation problem by defining user-specific objective functions, which are built as the weighted average of different user demands such as latency, user priority, and user/slice bandwidth. For this problem, a slice allocation scheme that integrates inter-slice handover mechanism is proposed. In case of network congestion, the inter-slice handover scheme, which is based on blocking the users from its primary slice candidate, is shown to be an effective solution.

The slice allocation problem is regarded as the maximization of total data rate in [39]. In this study, instead of direct summation of data rates, the objective is defined by the exponents of slice and user priorities. The generated problem is tackled by using heuristic intra-slice and inter-slice allocation schemes.

Another approach for the slice and bandwidth management is given in [40]. In this study, the main objective is defined as the minimization of total used bandwidth. In this regard, a two step approach is used. First, in order to increase the number of admissible UEs in the network, QoS (in terms of data rate and delay) degradation is employed. After the QoS degradation, for the obtain admissible UEs, the slice and bandwidth allocation is performed in an iterative way.

A dynamic and more realistic network model is studied in [41]. It is assumed that each slice in the network is provided by different subset of BSs. By defining the costs of different handover types (e.g., switching slices and BSs), they formalise the handover problem as a reinforcement learning problem of which the objective is to minimize the long-term handover costs. In addition to above study, [42, 43] also exploit the power of reinforcement learning in resource management problem in network slicing. Another example of machine learning usage in the network slicing is given in [44]. In this study, deep neural networks are used for the prediction of slice type from the traffic patterns. Then, by using the output of the neural network, an incoming device is assigned to the proper slice. The studies briefly mentioned in this paragraph constitute good examples of how AI can be a useful tool in network slicing problem.

A two-level resource allocation problem, which includes user admission in the high-level and power control in the low-level, is studied in [45]. For the low-level part of the problem, a dynamic programming approach is used while Lagrangian duality is used for the solution of the upper-level part. Even though combining the problems of two different levels increases the complexity of the problem further, the resulting combined scheme shown to be giving better results compared to the single-level solutions.

The authors in [46] propose two approaches for the problem of data burst grooming, which is the method of aggregating data bursts that are sent to various targets together, in optical networks. The two suggested heuristics aim to minimize network overhead. One of the approaches suggests that for data burst aggregation, no routing overhead is added. The other approach minimizes the total overhead for routing

and padding. Overall, grooming approaches are shown to provide better results as opposed to the no grooming employed cases. The authors in [47] study data grooming in wavelength division multiplexing networks. An optimization problem is presented and a heuristic method based on dynamic programming is proposed to solve the problem. The optimization problem is defined as a minimization problem that targets minimizing the cost of high level network components. It is shown that the optimization problem solution is useful in small network examples and the proposed heuristic method is better for bigger networks.

With all these aforementioned studies in the literature, in this thesis, we have defined slice allocation problem for $5G$ networks as follows: Consider a central process at the base station that manages the slice assignments and handover decisions. For this central unit, we define an optimization problem to maximize the overall slice utilization and increase the total number of slices that each user can get service from. For this problem, the constraints are determined by slice demands of UEs, capacities of slices, and coverage information of UE-BS pairs. We propose heuristics with handover mechanisms to solve our optimization problem in relatively low computing times compared to toolbox solutions. In these heuristics, when the capacity limit of a slice is reached, proposed handover mechanisms enable continuation of service from the slice by reducing the load on the slice with handover of UEs to other BSs.

3. A JOINT HANDOVER/SLICE ALLOCATION PROBLEM FOR 5G NETWORKS

In this chapter, a joint handover/slice allocation problem, based on the load of the network slices, is introduced. We consider a $5G$ network that consists of multiple BSs and UEs. In this network, each BS is assumed to be capable of providing service for all of the slices in the network. The slices are assumed to be allocated on different frequency bands such that they do not interfere with each other. The slice capacities do not change dynamically. In addition, it is assumed that each UE can get service from more than one slice at the same time. Regardless of the number of UEs, each UE is assumed to have an equal and constant data rate for the sake of simplicity. In the nature of the problem that we are handling, we try to maximize the number of slice usage, not the QoS. Also, a UE is assumed to get sufficient service whenever it is connected to a slice. Therefore, the interference within the same slice users is omitted in this thesis.

Since each slice has a limited capacity, incoming UEs cannot get service after the slice limit is reached. In order to improve the RAN resource utilization for the slices, we define an optimization problem where decision variables determine BS connectivity and slice allocations for each UE.

We introduce some mathematical preliminaries and define our network model in Section 3.1. Then, the problem is formulated as an optimization problem in Section 3.2. The convexity analysis, which seeks to show the existence of a unique global optimum, is performed in Section 3.3. Then, the details of mixed-integer nonlinear programming (MINLP) along with a relaxation methods are discussed in Section 3.4. Linearization of our optimization problem is performed in Section 3.5. Lastly, the toolbox solution for our optimization problem is explained in Section 3.6.

3.1. Network Model

The network model is defined along with some mathematical preliminaries. Assume that the identifier of a single UE in the network is denoted by $k \in \mathcal{K} = \{1, 2, \ldots, K\}$, where $\mathcal{K}$ is the set of all UEs existent in the network and the cardinality of $\mathcal{K}$ is given by K. The cells (or BSs) in the network are indexed by $n \in \mathcal{N} = \{1, 2, \ldots, N\}$, where $\mathcal{N}$ denotes the set of all BSs in the network and its cardinality is given by N. Furthermore, a single slice in the network is denoted by $s \in \mathcal{S} = \{1, 2, \ldots, S\}$. Here, $\mathcal{S}$ represents the set of all available slices where S denotes its cardinality. For the sake of simplicity, each BS is assumed to provide service for all S slices, which means the slice set $\mathcal{S}$ is same for all BSs.

The parameters will be defined in two main parts. First, we will introduce c_k^n, $d_{k,s}$, b_s^n, and r_S as the input parameters. Then, we will present the decision variables x_k^n and $p_{k,s}$.

The binary variable c_k^n that denotes if UE k is in the coverage area of BS n is defined as

$$c_k^n = \begin{cases} 1, \text{if UE } k \text{ is in the coverage area of BS } n, \\ 0, \text{otherwise.} \end{cases} \tag{3.1}$$

Here, it is also assumed that the required quality of service is guaranteed for all BS–UE pairs that satisfy $c_k^n = 1$. Therefore, the optimization problem that will be defined in the next section will omit the signal strengths and Signal to Noise Ratios (SNR) between BSs and UEs.

Next, we define variable $d_{k,s}$, which denotes if slice s is demanded by UE k, as follows:

$$d_{k,s} = \begin{cases} 1, \text{if UE } k \text{ demands to use slice } s, \\ 0, \text{otherwise.} \end{cases} \tag{3.2}$$

For any BS n, an upper limit, denoted as b_s^n, for the maximum number of UEs that can use slice s is assumed to exist. If this limit is reached, BS n cannot provide service to any additional UE for slice s. Moreover, we assume that each slice supports a fixed data rate r_s that is independent of UE.

Next, we define x_k^n and $p_{k,s}$ that are the decision variables of the joint handover/slice allocation problem. The binary variable x_k^n denotes whether UE k will get service from BS n or not, and it is defined as follows:

$$x_k^n = \begin{cases} 1, \text{if UE } k \text{ gets service from BS } n, \\ 0, \text{otherwise.} \end{cases} \tag{3.3}$$

By using this definition, we can define the vector that contains the connection information of UE k as $x_k = [x_k^1, x_k^2, \ldots, x_k^N]^T$. Then, the connection information of all UEs in the network can be nested in the vector $\mathbf{x} = [x_1^T, x_2^T, \ldots, x_K^T]^T$. Please note that the decision vector $\mathbf{x}$ constitutes the handover part of the problem.

In 5G networks, the demand of UE k to use a slice s can be rejected [9, 48]. Therefore, we formulate the allocation problem taking that into account by introducing the decision variable $p_{k,s}$ that indicates whether slice s will be provided to UE k or not. $p_{k,s}$ is defined as follows:

$$p_{k,s} = \begin{cases} 1, \text{if UE } k \text{ is provided with slice } s, \\ 0, \text{otherwise.} \end{cases} \tag{3.4}$$

Consequently, complete slice supply information of UE k can be defined by $p_k = [p_{k,1}, p_{k,2}, \ldots, p_{k,S}]^T$, whereas $\mathbf{p} = [p_1^T, p_2^T, \ldots, p_K^T]^T$ contains entire network's slice supply information.

3.2. Problem Formulation

We define the handover/slice allocation problem by using the definitions given in Section 3.1. With the objective of maximizing total number of slice usage, the optimization problem that is on the scope of this thesis is defined as follows:

$$\underset{\mathbf{p},\mathbf{x}}{\text{maximize}} \quad \sum_{s\in S}\sum_{k\in K} p_{k,s}, \tag{3.5a}$$

$$\text{subject to} \quad \sum_{n\in N} x_k^n \leq 1, \quad \forall k \in \mathcal{K}, \tag{3.5b}$$

$$x_k^n \leq c_k^n, \quad \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \tag{3.5c}$$

$$p_{k,s} \leq d_{k,s} \sum_{n\in N} x_k^n, \quad \forall k \in \mathcal{K}, \forall s \in \mathcal{S}, \tag{3.5d}$$

$$r_s \sum_{k\in K} x_k^n p_{k,s} \leq b_s^n, \quad \forall s \in \mathcal{S}, \forall n \in \mathcal{N}, \tag{3.5e}$$

$$p_{k,s} \in \{0,1\}, \quad \forall k \in \mathcal{K}, \forall s \in \mathcal{S}, \tag{3.5f}$$

$$x_k^n \in \{0,1\}, \quad \forall k \in \mathcal{K}, \forall n \in \mathcal{N}. \tag{3.5g}$$

Users can be in the coverage area of more than one BS but they can be connected to one BS at a time. Constraint given in Eq. (3.5b) ensures that a UE can only be connected to at most one BS. Additionally, UEs can connect to BSs that they are covered by. Constraint shown in Eq. (3.5c) makes sure that a UE can be connected to a BS only if it is in the coverage area of that BS. UEs that requests service from more than one slice need to get service from their serving BS. They cannot connect to more than one BS to use different slices. Thus, Eq. (3.5d) ensures that UE k gets service from slice s only if it is connected to a BS and demands for slice s. UEs that use the same slice in the same BS should not violate the slice capacities. By enforcing constraint in Eq. (3.5e), we make sure that the slice usage upper limits for each slice

are not exceeded. Finally, Eq. (3.5f) and Eq. (3.5g) restrict the decision variables to be binary.

3.3. Convexity Analysis

The problem defined in Eq. (3.5) is a Mixed Integer Quadratically Constraint Programming (MIQCP), where the quadratic constraint comes from Eq. (3.5e). Moreover, all decision variables are binary.

One of the most important properties of an optimization problem is its convexity. When the problem is convex, all the local minimum points are guaranteed to be the global minimum [49]. In this case, the solution that is found is unique and is either a global maximum (if the problem is a maximization problem) or global minimum (if the problem is a minimization problem).

In order for an optimization problem to be convex, its objective function and the sets defined by the constraints should be convex. For function $f(\cdot)$ defined on a convex set to be convex, it should satisfy the following inequality for all x, y within the given set and for any $\alpha \in (0, 1)$ [49]:

$$f(\alpha x + (1-\alpha)y) \leq \alpha f(x) + (1-\alpha)f(y). \tag{3.6}$$

Convexity of a set is defined as follows: when a line is drawn between any point a and point b within set $\mathcal{S}$ and all of the points on the line are also within the same set, then this is the convex combination of a and b. If this situation holds for all a and b within the set S, then the set is convex [49]. By this definition, it can be easily shown that the sets defined by constraints given in Eq. (3.5b), Eq. (3.5c), and Eq. (3.5d) are convex. Since the constraint shown in Eq. (3.5e) is a quadratic function of decision variables, its convexity analysis can be done by using the following theorems from [49]:

Theorem 3.1. *Given a function $f(x)$ is convex over set Ω, then Ω is a convex set.*

Theorem 3.2. *Given a function $f : \Omega \to \mathbb{R}$, $f \in C^2$ on $\Omega \subset \mathbb{R}^n$ is convex if and only*

if its Hessian is a positive semidefinite matrix.

Furthermore, a symmetric matrix Q is defined to be positive semidefinite if and only if all of its eigenvalues are greater than or equal to zero. In order to check the convexity of constraint given in Eq. (3.5e), we can write it in vector form as $f(x) = x^{\mathsf{T}}Qx$, and look at the eigenvalues of its Hessian matrix Q, which is symmetric by definition. If Q is neither positive semidefinite nor negative semidefinite, then we can conclude that our problem contains a non–convex constraint set.

In order to prove the non-convexity of Eq. (3.5e), we have shown a contradicting example. Since Eq. (3.5e) is defined as the summation of multiplication of any two variables, it is quadratic. Therefore, constraint given in Eq. (3.5e) can be expressed as $g(z) = \sum x_k^n p_{k,s} = z^{\mathsf{T}}Qz$. If we take $K = 3$, $S = 2$ and $N = 2$, the overall decision vector $\mathbf{z} = [\mathbf{x}^T, \mathbf{p}^T]^T$ can be written as

$$\mathbf{z} = [x_1^1 \;\; x_1^2 \;\; x_1^3 \;\; x_2^1 \;\; x_2^2 \;\; x_2^3 \;\; p_1^1 \;\; p_2^1 \;\; p_3^1 \;\; p_1^2 \;\; p_2^2 \;\; p_3^2]^T. \tag{3.7}$$

Then, the Hessian matrix becomes:

$$Q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{1}{2} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}. \tag{3.8}$$

Since both 0.5 and -0.5 are among the eigenvalues of Q, it is neither positive nor negative semidefinite. In conclusion, our problem is neither convex nor concave.

3.4. Mixed-Integer Nonlinear Programming

Mixed–Integer Nonlinear Programming (MINLP) is one of the most commonly encountered type of optimization problems in communication studies. This type of problems must include at least one integer variable [50], whereas the nonlinearity character arises from one or more nonlinear constraints (or a nonlinear objective function). General form of a MINLP can be given as

$$\begin{aligned} \underset{x,\,y}{\text{minimize}} \quad & f(x,y) \\ \text{subject to} \quad & h(x,y) = 0, \\ & g(x,y) \leq 0, \\ & x \in X \subseteq \mathbb{R}, \\ & y \in Y \subseteq \mathbb{Z}. \end{aligned} \tag{3.9}$$

Due to the nonlinearity, MINLP problems are mostly nonconvex which makes finding the global optimum hard. Also, the analytical solution for these problems does not exist, whereas the numerical solutions are considered as NP-hard [51]. Some of the general methods to solve a convex MINLP can be listed as:

- Generalized Benders Decomposition,
- Branch and Bound,
- Outer Approximation,
- Feasibility Approach,
- Outer Approximation with Equality Relaxation,
- Outer Approximation with Equality Relaxation and Augmented Penalty,
- Generalized Outer Approximation,
- Generalized Cross Decomposition.

When it comes to nonconvex MINLP, above solution methods are not directly applicable. A broad survey given in [52] on nonconvex MINLP outlines the existing methods for solving these types of problems. Some of the exact solution methods for the general nonconvex MINLP are listed as follows:

- Spatial Branch and Bound,
- Branch and Reduce,
- α Branch and Bound,
- Conversion to Mixed Integer Linear Programming (MILP),

As shown in Section 3.3, the problem defined in equation set (3.5) is a nonconvex MIQCP problem with binary decision variables. Our approach to this special subset of nonconvex MINLP problems falls into the category that is shown as the last item of the list above. In the next section, a technique that converts the problem given in equation set (3.5) into MILP will be demonstrated.

3.5. Linearization of Quadratic Constraints

By using a simple yet elegant technique, the quadratic constraint given in Eq. (3.5e) can be converted into a set of linear inequalities [53]. The method can be described as follows: Consider the quadratic constraint of the form rq of two binary variables. By introducing a new variable m along with constraints $m \leq r$, $m \leq q$, and $m \geq r + q - 1$, the rq term can simply be replaced with m. Since the new function and the constraints are linear, we get rid of the quadraticity of the original constraint.

Consequently, by introducing new variables $m_{k,s}^n = x_k^n p_{k,s}$, the constraint given in Eq. (3.5e) can be replaced by the following set of linear inequality constraints:

$$\begin{aligned} r_s \sum_{k\in K} m_{k,s}^n &\le b_s^n, \quad \forall k, \forall s, \forall n \\ m_{k,s}^n &\le x_k^n, \quad \forall k, \forall s, \forall n \\ m_{k,s}^n &\le p_{k,s}, \quad \forall k, \forall s, \forall n \\ m_{k,s}^n &\ge p_{k,s} + x_k^n - 1, \quad \forall k, \forall s, \forall n \end{aligned} \tag{3.10}$$

By replacing Eq. (3.5e) with the new set of constraints given in Eq. (3.10), the new optimization problem can be written as follows:

$$\underset{\mathbf{p},\mathbf{x},\mathbf{m}}{\text{maximize}} \quad \sum_{s\in S}\sum_{k\in K} p_{k,s}, \tag{3.11a}$$

$$\text{subject to} \quad \sum_{n\in N} x_k^n \le 1, \quad \forall k \in \mathcal{K}, \tag{3.11b}$$

$$x_k^n \le c_k^n, \quad \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \tag{3.11c}$$

$$p_{k,s} \le d_{k,s} \sum_{n\in N} x_k^n, \quad \forall k \in \mathcal{K}, \forall s \in \mathcal{S}, \tag{3.11d}$$

$$r_s \sum_{k\in K} m_{k,s}^n \le b_s^n, \quad \forall s \in \mathcal{S}, \forall n \in \mathcal{N}, \tag{3.11e}$$

$$m_{k,s}^n \le x_k^n, \quad \forall k \in \mathcal{K}, \forall s \in \mathcal{S}, \forall n \in \mathcal{N}, \tag{3.11f}$$

$$m_{k,s}^n \le p_{k,s}, \quad \forall k \in \mathcal{K}, \forall s \in \mathcal{S}, \forall n \in \mathcal{N}, \tag{3.11g}$$

$$m_{k,s}^n \ge p_{k,s} + x_k^n - 1, \quad \forall k \in \mathcal{K}, \forall s \in \mathcal{S}, \forall n \in \mathcal{N}, \tag{3.11h}$$

$$p_{k,s} \in \{0,1\}, \quad \forall k \in \mathcal{K}, \forall s \in \mathcal{S}, \tag{3.11i}$$

$$x_k^n \in \{0,1\}, \quad \forall k \in \mathcal{K}, \forall n \in \mathcal{N}, \tag{3.11j}$$

$$m_{k,s}^n \in \{0,1\}, \quad \forall k \in \mathcal{K}, \forall s \in \mathcal{S}, \forall n \in \mathcal{N} \tag{3.11k}$$

We will use the MILP defined in equation set (3.11) in our simulations as it improves the runtime significantly compared to solving the nonconvex MIQCP problem. We have used Gurobi as the solver and implemented the code in Python.

3.6. A Mixed-Integer Programming Solver

In order to provide a benchmark for the algorithm that will be proposed in Chapter 4, the Gurobi Solver is used. Gurobi is a mathematical programming solver that can solve various types of optimization problems including MILP, LP, and MIQCP [54].

In this thesis, Gurobi is used for finding the solution of a MILP problem. In order to solve a Mixed-Integer Programming (MIP), which is the general case that contains both MILP and MIQCP, Gurobi utilizes various methods. Some of these techniques can be listed as follows:

- Branch and bound method that in general works by recursively partitioning the solution space into subspaces [55];
- Different problem reduction methods that are utilized before starting the branch and bound algorithm;
- Cutting planes that eliminate useless part of the solution space;
- Various heuristics.

The general principle of branch and bound methods that is applied by Gurobi can briefly be summarized as follows: Given a MILP, by relaxing integer variables as continuous ones, the problem is converted into a LP. If the solution of LP satisfies the integrality restrictions of the original problem, the solution is terminated. Otherwise, one of the variables that do not satisfy the integral conditions is selected and the decision space is branched into two subspace around that variable. After selecting the suitable subspace that can contain the solution (bounding phase), above steps are repeated until a solution that conforms with the integrality conditions.

Optimization problem solutions with Gurobi provide details about the methods that have been used to solve the problem, run time of the algorithm, root relaxation results, optimal solution and iteration details, etc. In this thesis, we will use the optimal result and algorithm execution time information to compare our proposed algorithm

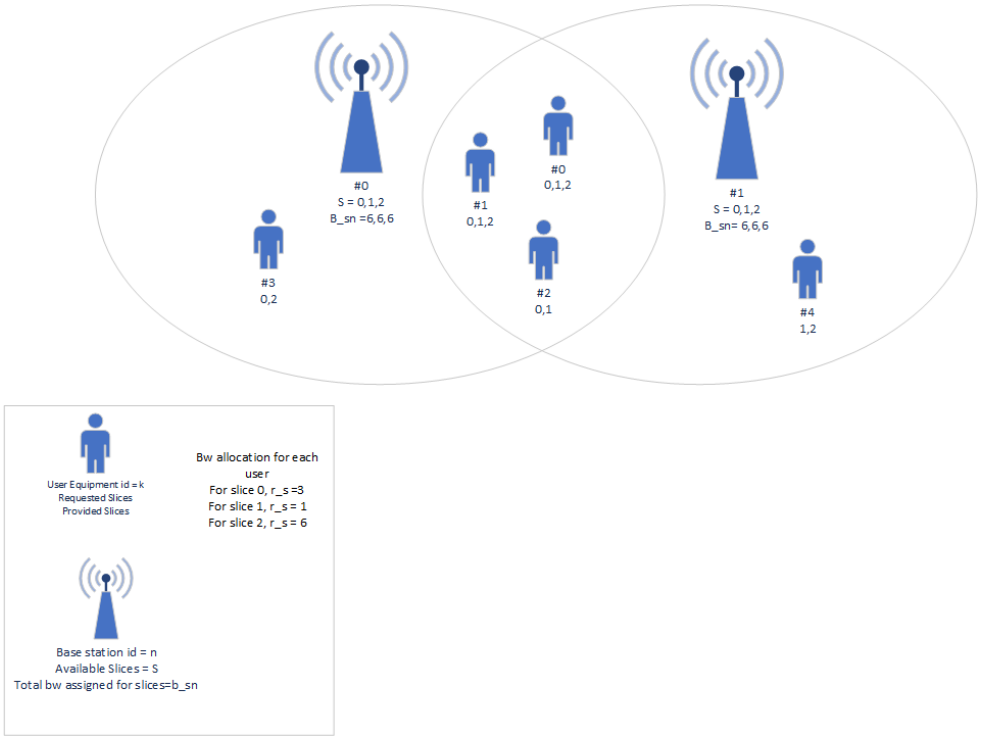

Figure 3.1. Simple Example - Initial State.

with optimization problem solution.

In this section, problem defined in equation set (3.11) has been solved with Gurobi for two small network examples. Later in Chapter 5, comparisons of the simulation results with Gurobi and our heuristic algorithm for a bigger network can be seen.

In the small network example, we consider two BSs and three slices. Capacity for the slices are set as 6 Mbps. For slice #0, #1, and #2, required data rate for each incoming UE is 3, 1, and 6 Mbps, respectively. UEs request service from more than one slice as listed in Table 3.1. Three of the UEs are in the coverage of both BSs and the remaining two UEs are in the coverage of only one BS, as shown in Figure 3.1.

Table 3.1 includes all of the input parameters. Gurobi execution for this problem takes 0.006 seconds, and 10 slice assignments for the UEs have been found as the optimal. Problem solution results in BS and slice assignments where three UEs can

Table 3.1. Small Network Parameters.

Paramater	**Value**
Number of BSs	2
Number of slices	3
Number of UEs	5
Aggregate slice data rate in each BS	6 Mbps
Required data rate for one UE in slice 0	3 Mbps
Required data rate for one UE in slice 1	1 Mbps
Required data rate for one UE in slice 2	6 Mbps
Requested slices by the UE #0	$\#0, \#1, \#2$
Requested slices by the UE #1	$\#0, \#1, \#2$
Requested slices by the UE #2	$\#0, \#1$
Requested slices by the UE #3	$\#0, \#2$
Requested slices by the UE #4	$\#1, \#2$
UEs that are in the coverage area of the BS #0	$\#0, \#1, \#2, \#3$
UEs that are in the coverage area of the BS #1	$\#0, \#1, \#2, \#4$

get service for all of their requested slices, and two of the UEs get service from two out of three slices that they have requested. In this simple network, Gurobi finds one of the optimal assignments and randomly assigns slices to UEs as shown in Figure 3.2.

- UE #0 is connected to BS #0 and getting services from the slices $\#0, \#1$,
- UE #1 is connected to BS #1 and getting services from the slices $\#0, \#1$,
- UE #2 is connected to BS #1 and getting services from the slices $\#0, \#1$,
- UE #3 is connected to BS #0 and getting services from the slices $\#0, \#2$,
- UE #4 is connected to BS #1 and getting services from the slices $\#1, \#2$.

We have run Gurobi for another small network example to see the assignment results where there are UEs that are not in the coverage of any BS. Two BSs are considered with six slices and UE number has been increased to 20. For each UE, the parameter, which demonstrates the BSs that the UE is in the coverage area of,

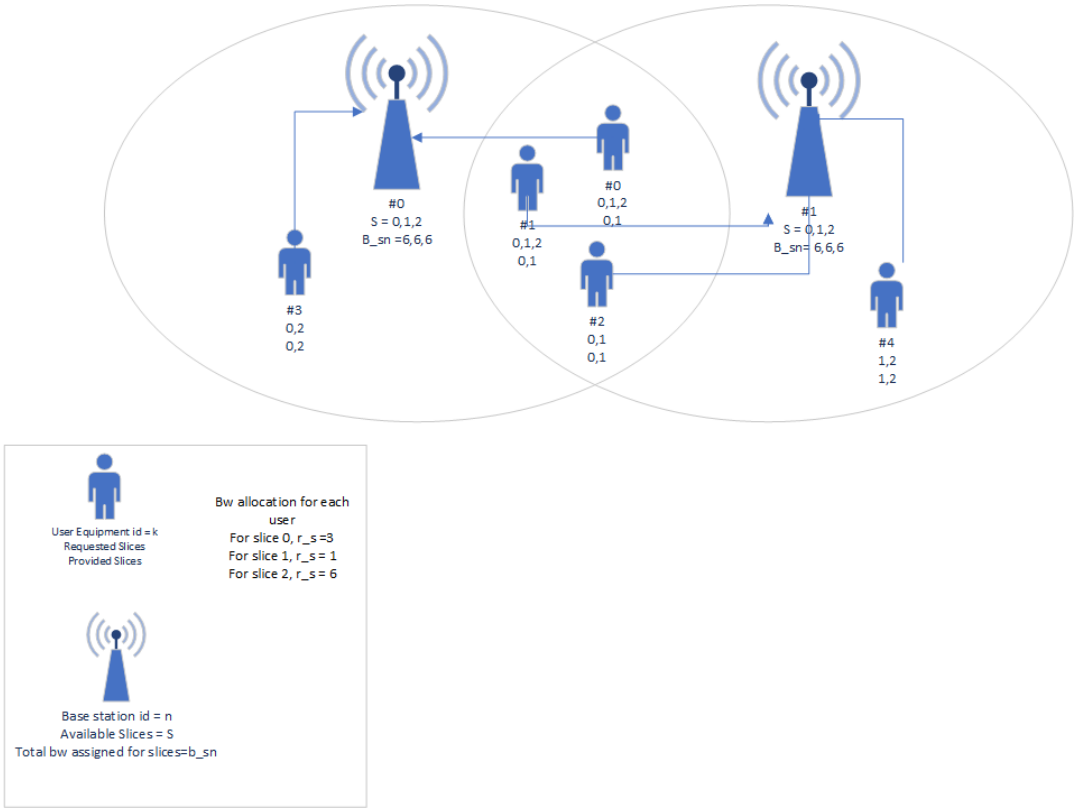

Figure 3.2. Simple Example - Optimal State.

is generated by selecting binary random values from a binomial distribution with the probability of success rate 0.4. Demanded slices for each UE are determined by selecting integer random values with binomial distribution of 0.9 success probability. Maximum capacity for slices on each BS are assigned between two and seven with uniform distribution. Data rate requirement for one UE in each slice is set as a random integer within the slice limit and the value is subjected to uniform distribution.

Half of the UEs are not in the coverage of any BS due to random assignments. The output shows that the UEs that are not in the coverage of any BS are also not connected to any BS and have not received service from their demanded slices at the end state. It is observed that UEs are not prioritized for the slice or BS assignments. Gurobi selects one of the optimal states and assigns slices to the UEs randomly. Furthermore, following observations ensure that the problem constraints are met. Firstly, one UE is

connected to only one BS where it is in the coverage area of, which shows Eq. (3.11b) and Eq. (3.11c) are satisfied. UEs are provided with slices that they request if they are connected to a BS and if the slice capacity is not reached, which are subject to constraints shown in Eq. (3.11d), and Eq. (3.11e) respectively. The BS that a UE is connected to provides the slices to the UE that meets our constraints in Eq. (3.11f), Eq. (3.11g), and Eq. (3.11h).

4. HEURISTIC ALGORITHMS FOR JOINT HANDOVER/SLICE ALLOCATION (JHSA) PROBLEM

In this chapter, we propose three heuristic algorithms for the problem described in Chapter 3. Initially, we define the Simple Algorithm that will also be used for the slice allocations in the other two algorithms. Secondly, we introduce the Greedy Handover Algorithm, which performs handover of UEs to improve the slice utilization without considering communication continuity. Lastly, we introduce the Intelligent Handover Algorithm, which performs handover of UEs where the utilization of the slices improve and UEs' connectivity to the slices are not dropped. All of the proposed algorithms take the following as inputs:

- N: Number of BSs in the network.
- S: Number of all slices in the network.
- K: Number of UEs in the network.
- An $N \times K$ dimensional array, which is composed of c_k^n values. This array contains all coverage area information of entire network.
- A binary array with the size $K \times S$ that consists all of the $d_{k,s}$ values that indicate if UE k demands to use slice s or not.
- An array with the size $S \times N$ that consists all of the b_s^n values that stand for the aggregate data rate for slice s in cell n.
- A list of size S that consists all of the r_s values for the required data rate per user for slice s.

Our proposed heuristics aim to find solutions to the defined problem and our performance measurement is taken as follows: the algorithm execution time and total number of $p_{k,s}$ that indicates if slice s is provided by BS n for UE k or not. Small network examples in Section 3.6 showed that Gurobi selects one of the optimal states and randomly assigns UEs to slices. In order to observe slice utilizations and BS utilizations, we also get the average utilization percentage of each BS in the network and

the average utilization percentage of each slice in the network compare our algorithms and Gurobi in Chapter 5.

Slice allocations are arranged according to the UE slice demands. [8] suggests a similar slice allocation principle. It is stated that slice allocation should be performed in a flexible way by considering the existing states of slices and future demands for the slice usages. Slice utilization is taken into consideration for slice assignments. In [9], the need for a new algorithm that decides whether to accept or reject an incoming slice demand to maximize network's utility is stated, which also supports the approach presented in this thesis.

Rest of the chapter is organized as follows. In Section 4.1, we introduce our first heuristic algorithm, which is used as a base algorithm in this thesis. In Section 4.2, we propose the Greedy Handover Algorithm, which aims to maximize the number of slices that are served. In Section 4.3, we describe the Intelligent Handover Algorithm where the number of slices that are served is maximized with respect to the ongoing connections of the UEs.

4.1. Simple Algorithm

The proposed Simple Algorithm provides an efficient slice assignment process for the incoming UEs. This algorithm increases slice utilizations by assigning the UEs to the BSs with relatively lower load. It targets to distribute the UEs as evenly as possible in the network so that the slice limits are not reached quickly and more UEs can get service.

Slice allocation and BS connection decisions are done centrally using the parameters known to the network. The network model defined in Section 3.1 is used in this algorithm.

Slice utilization and used slice capacity are calculated with every incoming user. For simplicity, a new variable $p_{k,s,n}$, alongside the variables described in Section 3.1, is used in our code to make calculations easier. This variable is used to calculate slice load and utilization. It is a binary variable and its value is assigned in the following manner:

$$p_{k,s,n} = \begin{cases} 1, \text{if UE } k \text{ is provided with slice } s \text{ by cell } n, \\ 0, \text{otherwise.} \end{cases} \tag{4.1}$$

Each UE has a fixed data rate r_s in a slice s. The load of the slice s in BS n can be calculated by multiplying number of users who are currently getting service from slice s in BS n by r_s, which can be formulated as

$$f_{s,n} = \sum_{k \in K} r_s \times p_{k,s,n}, \quad \forall s \in \mathcal{S}, \forall n \in \mathcal{N}. \tag{4.2}$$

Slice utilization that is referred in this thesis is calculated by dividing the total used capacity of the slice by the total slice capacity as follows:

$$u_{s,n} = \frac{f_{s,n}}{b_s^n}, \quad \forall s \in \mathcal{S}, \forall n \in \mathcal{N}. \tag{4.3}$$

Calculating each slice utilization on the BSs allows us to consider assigning the UEs to the least utilized slice, which increases the overall RAN resource utilization in the network.

Simple Algorithm decides BS and slice assignments until the network capacity is reached in the entire network. If the capacity is still available, for every incoming UE k, we check if the UE is in the coverage area of any BS. If UE k is in the coverage area of only BS n, we check the resources on the BS. Since the slices requested by UE k (where $d_{k,s} = 1$) are known by the network, we check utilization of the requested slices on the

target BS. If the requested slice capacity is not full, we check if the remaining capacity can meet the UEs' slice data rate requirement r_s. When Eq. (4.4) is met, the admission of UE to the BS is performed and the slices are provided. Please note that some of the requested slices by the UE k may not be provided to the UE k. Admission in Simple Algorithm is possible when at least one of the requested slices can be provided. If none of the requested slices is provided, UE k will not be admitted to BS n.

If UE k is within the coverage area of more than one BS, the BS selection procedure is executed according to slice utilizations. We check the utilization of the requested slices by UE k (where $d_{k,s} = 1$) at all nearby BSs. For each slice s, BS with the minimum utilization, $u_{s,n}$, gets a higher BS score, and after checking for all of the demanded slices, BS with the highest score is selected. In other words, BS that has the highest number of low utilized slices requested by the UE is selected. Then, for the selected BS, admission checks are performed. We check if the requested slices' unused capacity will meet the UEs' data rate requirement r_s. If Eq. (4.4) is satisfied, UE k is admitted to BS n and receives services from the selected slices where $p_{k,s} = 1$. The Simple Algorithm is implemented as in Figure 4.1.

$$b_s^n >= f_{s,n} + r_s. \tag{4.4}$$

Input: N, S, list of c_k^n, list of $d_{k,s}$, list of b_s^n, list of r_s

Output algorithm execution time, total number of $p_{k,s}$, avg utilization of BSs and slices

Initialize $u_{s,n}, f_{s,n}, p_{k,s,n}, p_{k,s}, x_k^n \Leftarrow 0$, $\forall s \in \mathcal{S}$, $\forall n \in \mathcal{N}$, $\forall k \in \mathcal{K}$

for all $k \in \mathcal{K}$ **do**

Check if the capacity limits are not reached in the network

if $\sum c_k^n = 1$ **then**

get the n where $c_k^n = 1$

for all $s \in \mathcal{S}$ **do**

if $u_{s,n} \neq 1$ and $(f_{s,n} + r_s <= b_s^n)$ **then**

$p_{k,s,n}, p_{k,s}, x_k^n \Leftarrow 1$,

Update $u_{s,n}$ and $f_{s,n}$ by Eq. (4.3), and Eq. (4.2)

end if

end for

end if

if $\sum c_k^n > 1$ **then**

for all n where $c_k^n = 1$ **do**

Compare all $u_{s,n}$ where $d_{k,s} = 1$

Select the assignment BS as the one with the most number of lowest $u_{s,n}$ of the demanded slices by UE k

for all $s \in \mathcal{S}$ **do**

Check if UE can connect to the slected BS

if $u_{s,n} \neq 1$ and $(_{s,n}+r_s <= b_s^n)$ **then**

$p_{k,s,n}, p_{k,s}, x_k^n \Leftarrow 1$,

Update $u_{s,n}$ and $f_{s,n}$ by Eq. (4.3), and Eq. (4.2)

end if

end for

end for

end if

end for

Figure 4.1. Simple Algorithm.

4.2. Greedy Handover Algorithm

Simple Algorithm defined in Section 4.1 lacks flexibility when there is an incoming UE that is in the coverage area of one BS and the requested slice capacities by the incoming UE is full in the BS. Thus, we propose the Greedy Handover Algorithm that introduces handover to improve the slice utilizations and the number of UEs that are getting service from slices.

We propose a heuristic algorithm that simultaneously performs the handover of the UEs and their slice assignments. These handovers and assignments are based on the BS capacities and the slice demands of the UEs. While the number of users increases, without exceeding the capacity limits for each slice, the algorithm performs handovers of the possible UEs to provide the maximum service for network slices that are demanded by the UEs. If a UE is in the intersection area of its current base station and neighboring station, and the load of the slices it demands is lower in the neighboring station, handover can be performed. In summary, this algorithm ensures that the resources of the network are efficiently distributed amongst network users. Thus, the overall utilization is maximized.

Greedy Algorithm applies the same approach as the Simple Algorithm when the incoming UEs are in the coverage area of more than one BS, which is selecting the most lightly loaded BS according to the requested slices. Even though handover approach could improve the utilization here as well, we do not consider it due to computation overhead. We only focus on handover of the UEs when there is a connection request coming from a UE within the coverage area of one BS.

Greedy Handover Algortihm decides BS and slice assignments until the network capacity is reached in the entire network. If the limit is not reached, for every incoming UE k, we check if the UE is in the coverage area of any BS. If the incoming UE is in the coverage area of more than one BS, it will be connected to the BS that can provide services through more slices for the UE. If all BSs can provide the same service for the

UE, the BS selection will be done randomly.

When there is a connection request from UE k_1 that is in the coverage area of BS n_1, we check the resources of BS n_1 for the slices that are requested by the UE, where $d_{k_1,s} = 1$. If one or more of the requested slices cannot provide service for UE k_1 (i.e., $util_{s,n_1} = 1$), we check if handover of one of the UEs that are currently connected to BS n_1 improves the utilization. Handover decision is done as follows: we get the list of UEs that are currently connected to BS n_1. From that list, we get the list of UEs that are in the coverage area of more than one BS. Then, we check the number of slices that these UEs get service from. UE k_2, which is allocating the highest number of slices that UE k_1 requests, is selected for handover check. For example, if UE k_1 requests connection to BS n_1 and requests to use s_1, s_2, s_3 (assuming that s_1 and s_2 are full), we check if handover is possible. Let us assume UE k_2 and UE k_3 are currently connected to BS n_1, and they are in the coverage area of more than one BS. Also, assume UE k_2 is currently getting service from s_1, s_2, and UE k_3 is getting service from s_2. Since UE k_2's handover will empty more slices (s_1, s_2) that are requested by UE k_1, it will be selected for handover check.

Next, we check if UE k_2's handover improves the total utilization. We get the list of BSs that UE k_2 is in the coverage area of. From the list, we check the utilization of slices, where $d_{k_2,s} = 1$, on these BSs. We select the emptiest one as the target BS. Before performing handover, we check the number of slices that UE k_2 can get service from. If the later number of slices is greater than or equal to the sum of the current number of slices that it gets service from and the number of slices that it empties for UE k_1, the handover is performed. After the handover of UE k_2, UE k_1 will be admitted to BS n_1.

Handover can be rejected if it does not improve the overall utilization. Then, UE k_1 can be admitted and get service from the slices that meet the capacity limits as in Eq. (4.4). Similar to Simple Algorithm described in Section 4.1, Greedy Handover Algorithm makes admission possible when one of the requested slices can be provided.

If none of the requested slices is provided, UEs' admission request will be rejected. Implementation of the Greedy Handover Algorithm can be seen in Figure 4.2.

Greedy Handover Algorithm performs better than Simple Algorithm when there is more load on some of the BSs in the network. Handover distributes the load to other BSs to maximize the overall utilization. Overall, more UEs can get service from BSs and slices. Thus, we propose to use Intelligent Handover Algorithm to solve the problem defined in Section 3.2.

Input: N, S, list of c_k^n, list of $d_{k,s}$, list of b_s^n, list of r_s

Output: algorithm exec time, total num of $p_{k,s}$, avg utilization of BSs and slices

Initialize $u_{s,n}, f_{s,n}, p_{k,s,n}, p_{k,s}, x_k^n \Leftarrow 0$, $\forall s \in \mathcal{S}$, $\forall n \in \mathcal{N}$, $\forall k \in \mathcal{K}$

for all $k \in \mathcal{K}$ **do**

Check if the capacity limits are not reached in the network

if $\sum c_k^n = 1$ **then**

get the n where $c_k^n = 1$

if any of the requested slices by UE k is full **then**

UE_h = get the list of UEs that are connected to BS n and in the coverage of more than one BS

for $i \in UE_h$ **do**

select UE k_2 that occupies the most number of requested slices by UE k

end for

covgBS = get the list of BSs that UE k_2 is in the coverage of exclude BS n

compare $u_{s,n}$ in list covgBS, find the emptiest one for $d_{k,s} = 1$

if $\sum p_{k_2,s}current \leq p_{k_2,s}next + p_{k,s}next$ **then**

Perform Handover of UE k_2

end if

end if

for all $s \in \mathcal{S}$ **do**

if $u_{s,n} \neq 1$ and $(f_{s,n} + r_s \leq b_s^n)$ **then**

$p_{k,s,n}, p_{k,s}, x_k^n \Leftarrow 1$,

Update $u_{s,n}$ and $f_{s,n}$ by Eq. (4.3), and Eq. (4.2)

end if

end for

end if

if $\sum c_k^n > 1$ **then**

Execute the algorithm given in Figure 4.1

end if

end for

Figure 4.2. Greedy Handover Algorithm.

4.3. Intelligent Handover Algorithm

We propose Intelligent Handover Algorithm with Network Slicing that introduces an intelligent decision mechanism for the handover of users to the neighboring stations. Simple Algorithm defined in Section 4.1 lacks flexibility of rearranging UE-slice assignments to increase the available capacity when the load increases for some of the BSs in the network. Greedy Handover Algorithm defined in Section 4.2 proposes a solution for slice load assignments, but this algorithm does not consider continuity of the connections. Therefore, after handover, the slices that UEs use on the source BS, may not be provided on the target BS.

The implementation of Intelligent Handover Algorithm is given in Figure 4.3. In order to solve the communication break problem seen in the Greedy Handover Algorithm, we implement additional controls for the handover decision. We use $s_{k,cont}$ vector, which is calculated as follows:

$$s_{k,cont} = \hat{p_k} - p_k, \tag{4.5}$$

where p_k is a $S \times 1$ dimensional vector and defined in Section 3.1. We use $\hat{p_k}$ to store the next p_k vector values that UE k will have on the target BS if handover occurs. We use $s_{k,cont}$ vector with size $S \times 1$, where we observe the differences in the slice usage information of the UE for handover decision. Minimum value in the vector $s_{k,cont}$ becomes zero if UE k can get service from all of the slices that are provided by source BS after handover. Minimum value in the vector $s_{k,cont}$ becomes minus one if UE k can not get service from one or more of the slices that are provided by source BS after handover.

Intelligent Handover and Greedy Handover Algorithms have the same UE and target BS selection mechanism for handover check. Let us assume UE k_1 is requesting admission and UE k_2 is connected to BS n_1 and selected for handover check. If the handover decision is made, UE k_2 will connect to the target BS n_2. Then, handover

decision by the Intelligent Handover Algorithm goes as follows: we first check if UE k_2's handover improves the total slice utilization. We check the number of slices that UE k_2 can get service from the target BS n_2. If the later number of slices is greater than or equal to the sum of the current number of slices that it gets service from and the number of slices that it empties for UE k_1, we check for connection continuity. We calculate $s_{k,cont}$ as in Eq. (4.5) and get the minimum value stored in $s_{k,cont}$. If minimum value is greater than minus one, handover is performed.

Handover can be rejected if it does not improve the overall utilization and if the target BS is not able to provide the slices that are provided by the source BS. Furthermore, similar to Simple Algorithm and Greedy Handover Algorithm, Intelligent Handover Algorithm admits the UEs only if one of the requested slices can be provided. In addition, the controls that are added to the Intelligent Handover Algorithm ensures increase in the overall utilization while continuing the slice service on the target BS after handover.

Input: N, S, list of c_k^n, list of $d_{k,s}$, list of b_s^n, list of r_s

Output: algorithm exec time, total num of $p_{k,s}$, avg utilization of BSs and slices

Initialize $u_{s,n}, f_{s,n}, p_{k,s,n}, p_{k,s}, x_k^n \Leftarrow 0$, $\forall s \in \mathcal{S}$, $\forall n \in \mathcal{N}$, $\forall k \in \mathcal{K}$

for all $k \in \mathcal{K}$ **do**

Check if the capacity limits are not reached in the network

if $\sum c_k^n = 1$ **then**

get the n where $c_k^n = 1$

if any of the requested slices by UE k is full **then**

UE_h = get the list of UEs that are connected to BS n and in the coverage of more than one BS

for $i \in UE_h$ **do**

select UE k_2 that occupies the most number of requested slices by UE k

end for

covgBS = get the list of BSs that UE k_2 is in the coverage of exclude BS n

compare $u_{s,n}$ in list covgBS, find the emptiest one for $d_{k,s} = 1$

if $\sum p_{k_2,s}current \leq p_{k_2,s}next + p_{k,s}next$ and $min(s_{k_2,cont}) > -1$ **then**

Perform Handover of UE k_2

end if

end if

for all $s \in \mathcal{S}$ **do**

if $u_{s,n} \neq 1$ and $(f_{s,n} + r_s \leq b_s^n)$ **then**

$p_{k,s,n}, p_{k,s}, x_k^n \Leftarrow 1$,

Update $u_{s,n}$ and $f_{s,n}$ by Eq. (4.3), and Eq. (4.2)

end if

end for

end if

if $\sum c_k^n > 1$ **then**

Execute the algorithm given in Figure 4.1

end if

end for

Figure 4.3. Intelligent Handover Algorithm with Network Slicing.

4.4. Time Complexity

In [56], time complexity for algorithmic problems is described in detail. Algorithmic problem is stated as a problem that can be solved by a computer which the found solution is explicit. Complexity for this kind of problems is defined as the necessary computation time to solve the optimal algorithm. Computation time depends on the inputs, computer power and properties, computer language that is used to implement the algorithm, and quality of the written code.

We consider a network that has K users, N cells, and S slices. For each user k, Simple Algorithm defined in Section 4.1 checks the number of BSs that cover user k. If the user if in the coverage area of one BS, algorithm checks for the slices that the user demands. In Figure 4.1, this execution is shown between step 2 and 12. The worst case computation time up to this point is $O(K \cdot S)$. For the case of users being in the coverage area of more than one BS, all the nearby BSs' capacity is checked for the demanded slices for each user. Therefore, the computation complexity for the steps between 13 and 26 in Figure 4.1 is $O(K \cdot N \cdot S)$. Overall, the time complexity for the Simple Algorithm is $O(K \cdot N \cdot S)$.

For the Greedy Handover Algorithm and the Intelligent Algorithm, there are additional controls for the handover decision. For each UE k, these algorithms check for handover UEs, and this increases the complexity defined for the Simple Algorithm by $O(K)$. Defined steps $2-23$ in Figure 4.2 and 4.3 have the computation complexity $O(K^2 \cdot N \cdot S)$ in the worst case because the algorithm repeats for each incoming UE and handover UE. The remaining part is the Simple Algorithm execution, which have the time complexity $O(K \cdot N \cdot S)$. As a result, both handover algorithms have the time complexity of $O(K^2 \cdot N \cdot S)$.

5. SIMULATION RESULTS

In this chapter, the performances of the proposed algorithms are compared via simulations as well as against the optimization problem, which is evaluated using Gurobi software. The simulations are conducted under two different scenarios. First, the network models are generated by random but uniformly distributed scenarios. Then, simulations are carried out for a special case where the UE density of the network decreases gradually while moving apart from the designated BSs. The network settings and results for both assumptions are given in Sections 5.1, 5.2, and 5.3. Initially, in Sections 5.1 and 5.2, the algorithms are studied for small networks. The mobility of the users is assumed to be provided by the change of the variable c_k^n, and the number of users in the network increases gradually. Lastly, in Section 5.3, we simulate the mobility of the users with Brownian motion and consider a scenario with more BSs and present the results.

5.1. General Comparison of Proposed Algorithms and Gurobi

The first simulations are carried out to observe the differences between our proposed algorithms and the Gurobi solver in the general network scenarios. The scenarios are generated randomly by using the parameters given in Table 5.1. For each simulation, the table is filled as follows: The number of BSs and slices are respectively fixed to be 7 and 4. The number of UEs is chosen between 50 and 1000 as an integer multiple of 50. ω, which signifies the capacity of a BS spared for each slice, generated randomly between 6 and 18 Mbps. Once this value is generated randomly for the given simulation, same value is used for all BS-slice pairs as the b_s^n parameter. c_k^n values for all n and k, which determine whether UE k is within the coverage area of BS n or not, are modelled with independent Bernoulli random variables that become 1 with the probability of 0.6. Same distribution is used for generating $d_{s,k}$ values as well. Finally, r_s that denotes the required data rate by the users of slice s is generated independently for each slice by using a uniform distribution between 1 and $\omega/3$.

Table 5.1. Simulation Parameters.

Parameter	**Value**
Number of BSs	7
Number of slices	4
Number of UEs	from 50 to 1000 with increment of 50
ω	a uniformly distributed random integer between 6 and 18 (in Mbps)
c_k^n	Bernoulli random variable with $P(c_k^n = 1) = 0.6$
$d_{s,k}$	Bernoulli random variable with $P(d_{s,k} = 1) = 0.6$
b_s^n	ω
r_s	a uniformly distributed random integer between 1 and $\omega/3$ (in Mbps)

The performance of the algorithms are compared in terms of algorithm run times, total number of slice services, and total slice utilizations. For each case, the parameters shown in Table 5.1 are generated repeatedly for 100 times. For instance, when the number of UEs is 250, we carry out 100 simulations such that each simulation has different parameters generated according to Table 5.1. By taking the average of metrics that we are interested in, we tried to eliminate the deviations created by the edge cases.

The simulation results of the general case are given in Figure 5.1, Figure 5.2 and Figure 5.3. We have plotted the figures by limiting the number of UEs to 300, as the network reaches its capacity before UE 300. Figure 5.1 shows that while the number of UEs in the network increases, Gurobi execution time also generally increases. Compared to Gurobi, our algorithm execution time is much lower and stays under 0.5 seconds.

Figure 5.2 shows that our proposed heuristics find considerably close results to the optimization problem solution. Since the network has been created randomly, the benefit provided by our slice-aware heuristics is not apparent under this scenario. Yet, all methods perform quite close to the optimal solution.

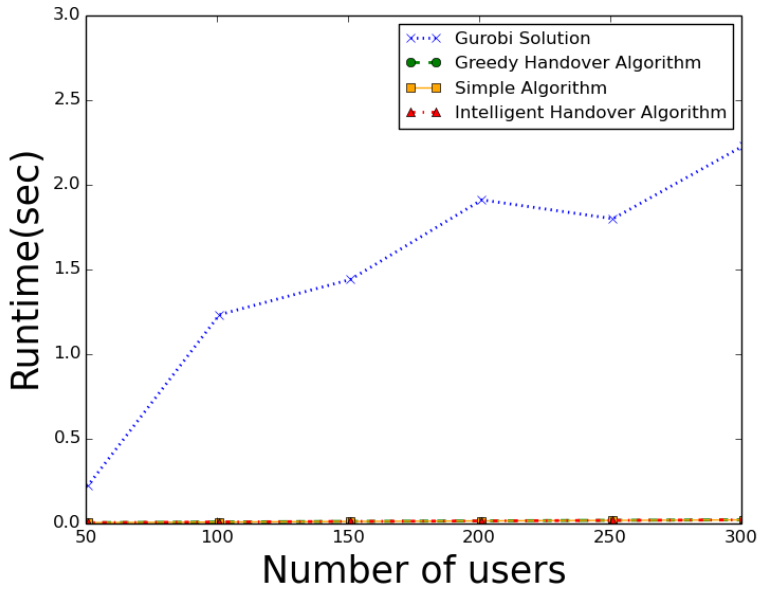

Figure 5.1. Number of users vs. algorithm execution time.

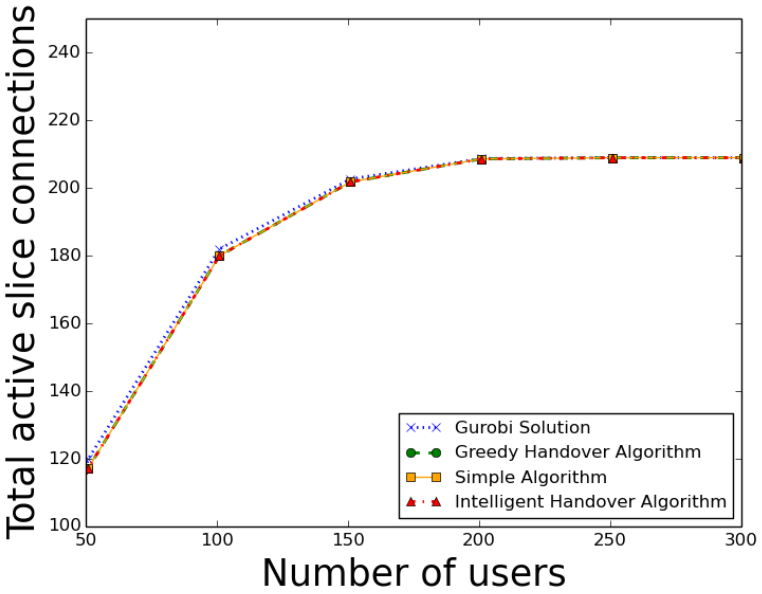

Figure 5.2. Number of users vs. total active slice connections.

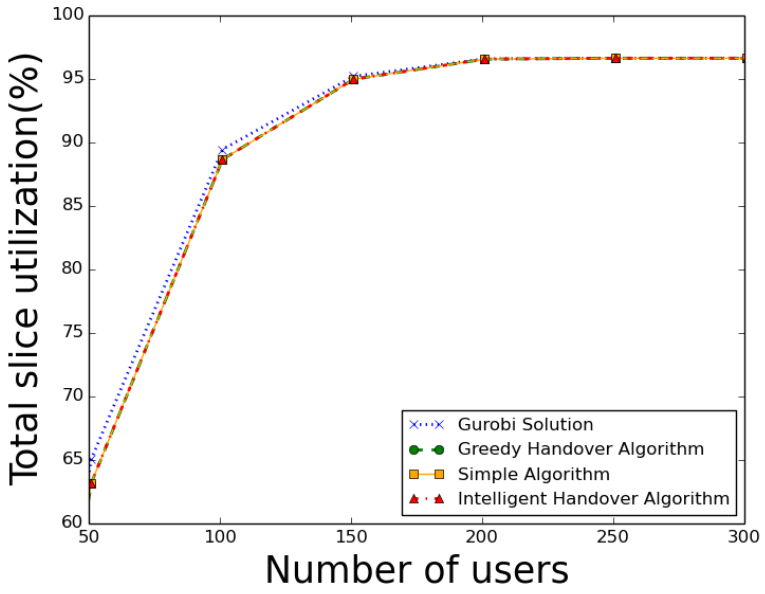

Figure 5.3. Number of users vs. total slice utilization.

Figure 5.3 shows the total slice utilization in the network while the number of users increase. It can be seen that when the number of UEs approaches 200, total network reaches its resource capacity, and the difference between the slice utilizations for different solutions becomes apparent only when the resources are not fully utilized. When the network is considerably empty, most of the UEs can get service from their demanded slices and the slice utilization increase is faster. When the slice utilization hits 90%, some of the UEs' slice requests get rejected by their serving BSs, which results in slower increase in the slice utilization percentage.

Figure 5.3 shows that Gurobi results and the results of the proposed heuristics do not differ considerably if the network is small and the UE increase in the network cause entire network to reach its full capacity. However, we still see a sharper increase in the Gurobi's execution time even for this small network.

5.2. A Special Case: Dense to Sparse Scenario

The uniformly random distribution of the subscribers is not realistic. Furthermore, the idea of proactively balancing the load of slices across cells renders useless when all cells have similar load. Therefore, we have created a realistic scenario where the density of the subscribers is high in a specific area, and it decreases as we move away from the center. The center in this scenario is assumed as located in cell b_1 and b_2. The scenarios are generated by using the parameters given in Table 5.2. For each simulation, a new dense-to-sparse scenario is created with 7 BSs and 4 slices, and the number of UEs varying between 1 and 450 as an integer multiple of 50. For each simulation, ω is picked from a uniform distribution in the range 6 to 19. Then, this value is used to decide b_s^n parameter, which denotes slice s's capacity for each BS n.

b_s^n is generated independently for each slice by using a uniform distribution between 5 and $5 * \omega/3$. r_s that denotes the required data rate by UE k is generated independently for each slice s by using a uniform distribution between 1 and $\omega/3$. c_k^n values for all n and k, which determine whether UE k is within the coverage area of BS n or not, are modelled with independent Bernoulli random variables that become 1 with the probability of 0.9, 0.9, 0.8, 0.7, 0.6, 0.5, and 0.4 for n = 1, 2, 3, 4, 5, 6, 7, respectively. Finally, $d_{s,k}$ values are selected as Bernoulli random variables that become 1 with the probability of 0.6. The simulations are carried out 200 times and the average values are reflected in the graphs.

In Figure 5.4, we see that Gurobi's execution time increases faster compared to our heuristics and since the probability of the users being in the coverage area of more than one BS is decreased, the problem for Gurobi is simplified. So compared to the results in the previous section, we see lower execution times for Gurobi.

Table 5.2. Simulation Parameters.

Parameter	**Value**
Number of BSs	7
Number of slices	4
Number of UEs	from 1 to 451 with increment of 50
ω	a uniformly distributed random integer between 6 and 19 (in Mbps)
c_k^n	Bernoulli random variable with $P(c_k^1 = 1) = 0.9$, $P(c_k^2 = 1) = 0.9$, $P(c_k^3 = 1) = 0.8$, $P(c_k^4 = 1) = 0.7$, $P(c_k^5 = 1) = 0.6$, $P(c_k^6 = 1) = 0.5$, $P(c_k^7 = 1) = 0.4$
$d_{s,k}$	Bernoulli random variable with $P(d_{s,k} = 1) = 0.6$
b_s^n	a uniformly distributed random integer between 5 and $5 * \omega/3$ (in Mbps)
r_s	a uniformly distributed random integer between 1 and $\omega/3$ (in Mbps)

In Figure 5.5 and 5.6, we see that the Greedy Handover Algorithm finds the same results as Gurobi. The Intelligent Handover Algorithm finds very close solutions to the optimal. Both handover algorithms are better at utilizing the RAN resources of the slices. However, the Simple Algorithm is approximately 10% lower.

The difference between our heuristics is not visible when the network is relatively empty since all heuristics assign the users to the BS that they can get service from more number of slices. However, when the network is close to reaching its full capacity, the difference between our heuristics becomes apparent. As there can be a need for some BSs to improve slice utilization due to the increased load, handover allows improving the overall slice utilization. While the network is about to reach its full capacity, the difference between our heuristics decrease since there is not much room for improvement.

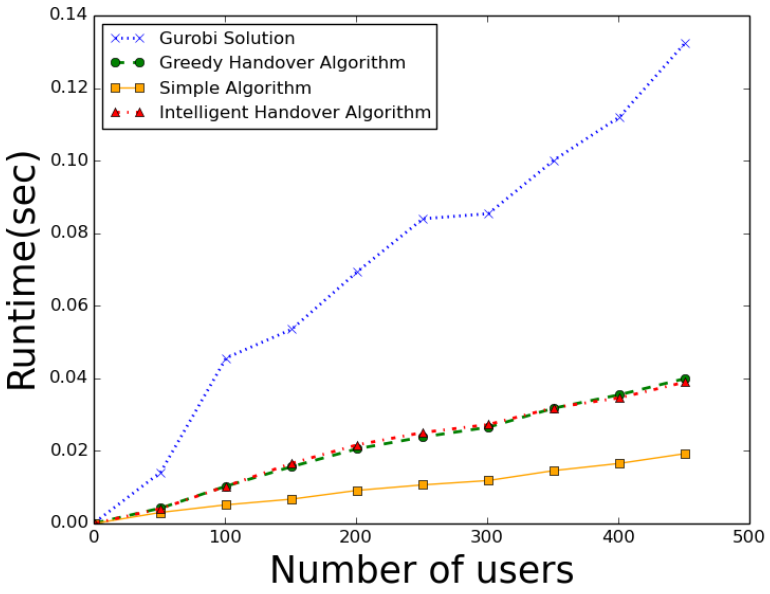

Figure 5.4. Number of users vs. algorithm execution time.

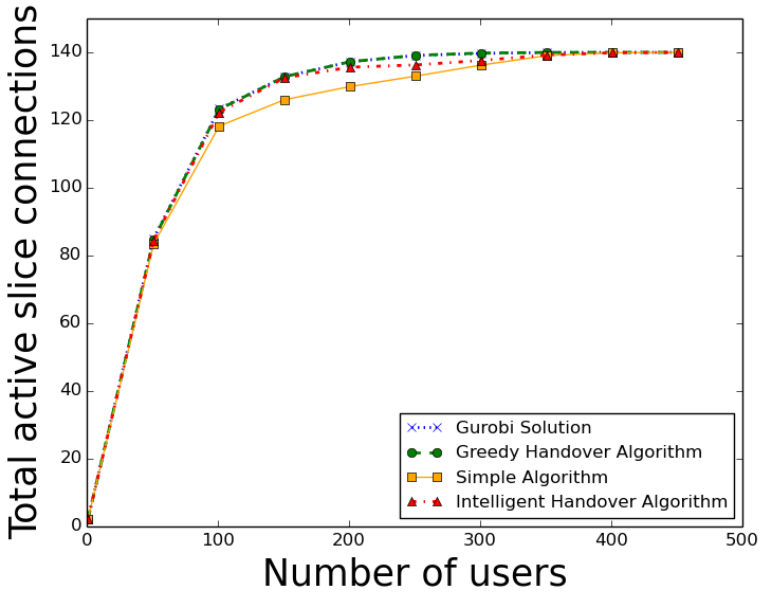

Figure 5.5. Number of users vs. total active slice connections.

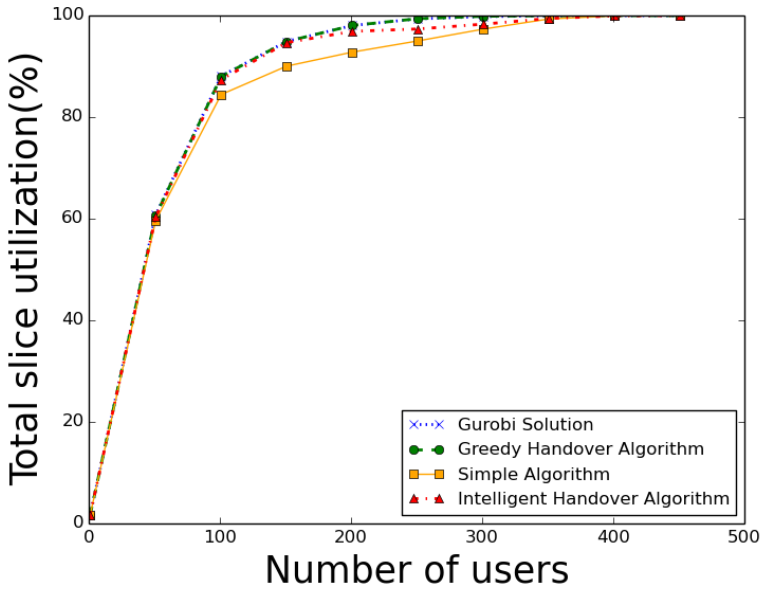

Figure 5.6. Number of users vs. total slice utilization.

5.3. Mobility Analysis

In this section, we analyse user mobility and add more criteria to review the performances of our algorithms and Gurobi. We compare the performances of the algorithms according to algorithm execution time, the number of active slice connections in the network, overall slice utilization, total number of handovers, and total number of dropped slice connections.

5.3.1. Homogeneous Scenario

Mobility analysis is carried out to observe the performance differences between our proposed algorithms and Gurobi solution when mobility of the users is considered. Network is created according to Table 5.3. In the mobility simulations, we have modelled a network with 12 BSs. We consider a scenario as shown in Figure 5.7. In this model, the distance between the neighboring BSs is taken as 150 meters where

the coverage range of BSs is taken as 100 meters. In Figure 5.7, the centers of BSs are shown with dark points whereas their coverage areas are denoted by dashed lines. Even though in real life, the range of macro BS can go up to kilometers, for the sake of simplicity, we consider only densely deployed small cells. However, the proposed solutions are independent of the cell radius. Slice capacities on the center BSs, b_1 and b_9, are taken as the double of their neighboring BSs since we consider these areas as gathering locations. However, in this simulation, there is no higher load on these BSs, and the users are uniformly distributed.

Table 5.3. Simulation Parameters.

Parameter	**Value**
Number of BSs	12
Number of slices	4
Number of UEs	4000
Radius of the BSs	100 (in meters)
Distance between neighboring BSs	150 (in meters)
ω	a uniformly distributed random integer between 6 and 19 (in Mbps)
c_k^n	Generated according to UE locations
$d_{s,k}$	1
b_s^n	a uniformly distributed random integer between 300 and $100 * \omega$ (in Mbps)
r_s	a uniformly distributed random integer between 1 and $\omega/3$ (in Mbps)

The simulations for the mobility analysis are done with 4000 UEs through 10 trials. In each of these trials, the initial position of UEs are taken randomly and the number of users in the network is considered to be the same. Initially, UEs are distributed randomly in the area of the BSs, where x coordinates of the UEs are selected within the range of 0 and 500, and y coordinates selected between 0 and $200 + 300\sqrt{3}$.

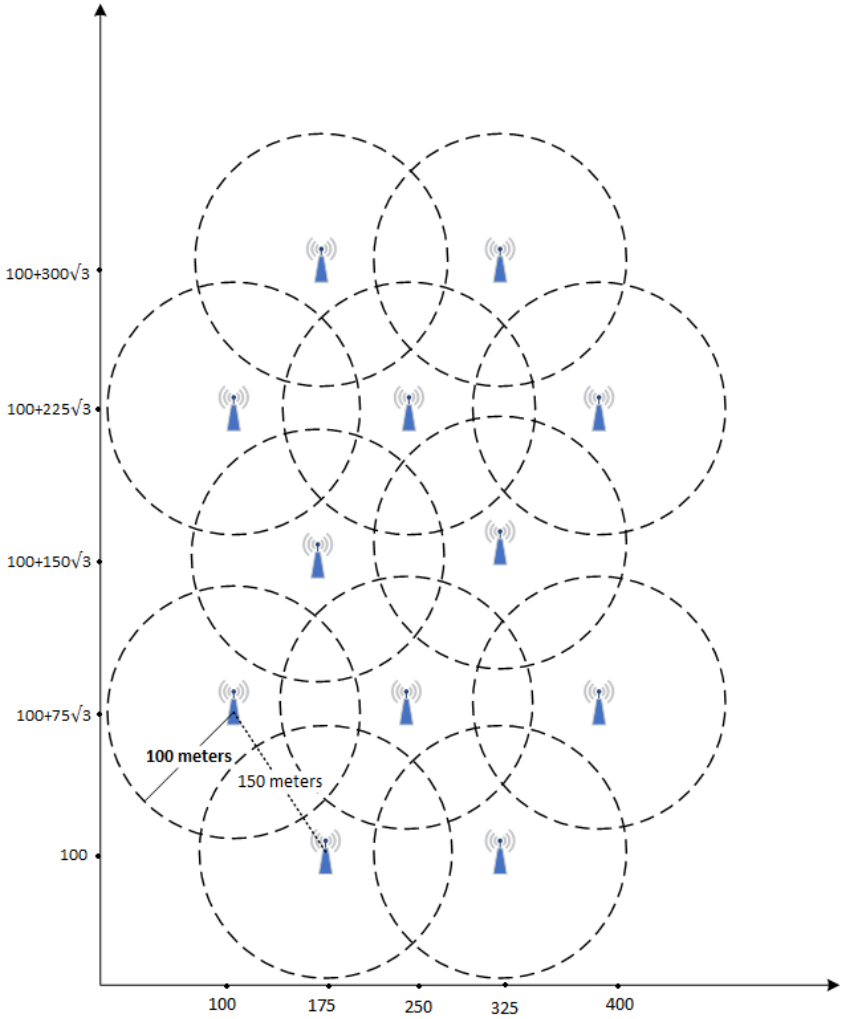

Figure 5.7. Homogeneous Scenario.

User movement is modelled as Brownian motion throughout the simulation and the movement area of the users is considered as in an unlimited space with the speed changing between $0 - 1.25$ meters per second. Simulations continue for 300 seconds, and Gurobi and heuristic algorithms are run every two seconds. We need all algorithms to present their solutions under two seconds, therefore this requirement is applied as a time limit for Gurobi and the heuristics. In addition, we also checked Gurobi without applying any time limit to observe the deviations of the algorithms from the optimal.

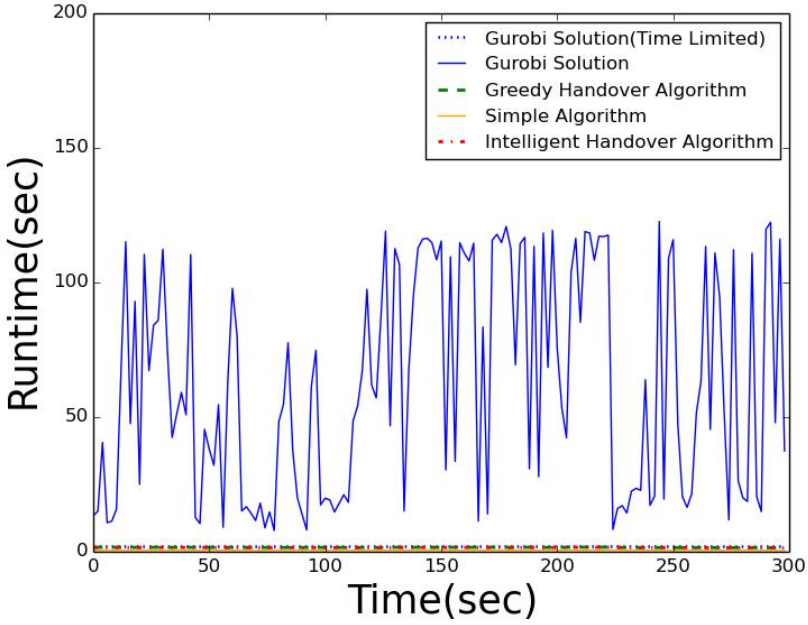

Figure 5.8. Run times of the algorithms.

In Figure 5.8, the execution time difference between our heuristics and Gurobi is shown. As the Simple Algorithm defined in Section 4.1 does not contain any additional handover methods to improve slice utilizations, execution time is considerably low. However, the Greedy Algorithm and the Intelligent Handover Algorithm, defined in Section 4.2 and 4.3 respectively, include additional handover mechanisms; therefore, their execution times are slightly higher than that of the Simple Algorithm. Yet, each heuristic provides a good improvement compared to Gurobi solution. Gurobi time limited solution also provides a good improvement in terms of runtime compared to Gurobi execution without any time limit.

As we run each algorithm in every two seconds, Gurobi is not able to provide an optimum solution within two seconds. While we wait for Gurobi to provide a solution in the initial 2-second window, the second 2-second window arrives without having an optimal solution from Gurobi. The locations of UEs may change without prior assignment if Gurobi is used for our problem.

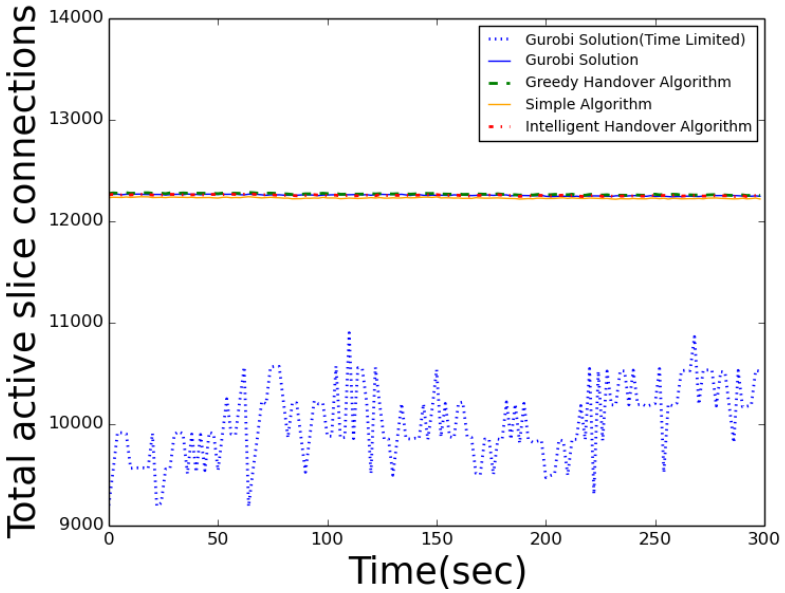

Figure 5.9. Total active slice connections.

In Figure 5.9, we show the number of total active slice connections achieved by executing each algorithm. Gurobi aims to find the maximum number of slice connections that can be achieved by UE admission decisions. Thus, it can be seen that Gurobi solution without any time limit is the highest among all. Our heuristics find relatively close solutions to Gurobi where Greedy Handover and Intelligent Handover algorithms find the closest solutions. However, Gurobi solution with two seconds time limit gives the worst solution. We show that efficiency of our heuristic algorithms is better in terms of providing timely and better assignment decisions.

In Figure 5.10, optimality gap for Gurobi time limited solution is shown. When the time limit of two seconds is applied, Gurobi is unable to provide the optimal solution. The solutions that are found by Gurobi are close to the optimal solution by 20% to 40%.

In Figure 5.11, overall slice utilization in the network is shown. Gurobi achieves the most slice utilization among others when there is no time limit. However, the difference with the solution found by heuristics is around 1% and Simple Algorithm

solution is the lowest among the heuristics. Gurobi achieves the lowest slice utilization when a time limit of two seconds is applied. We show that the proposed heuristic algorithms achieve better slice utilization under two seconds and Gurobi provides worse slice utilization than our algorithms under two seconds.

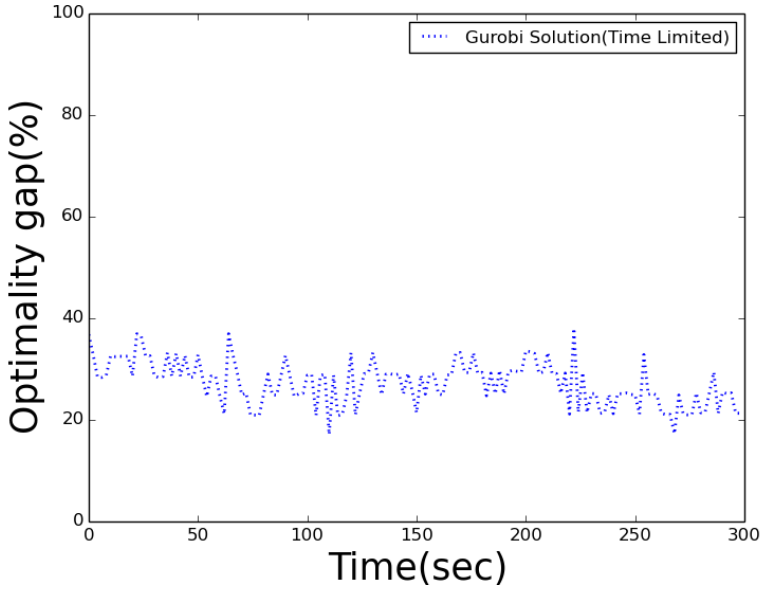

Figure 5.10. Optimality Gap.

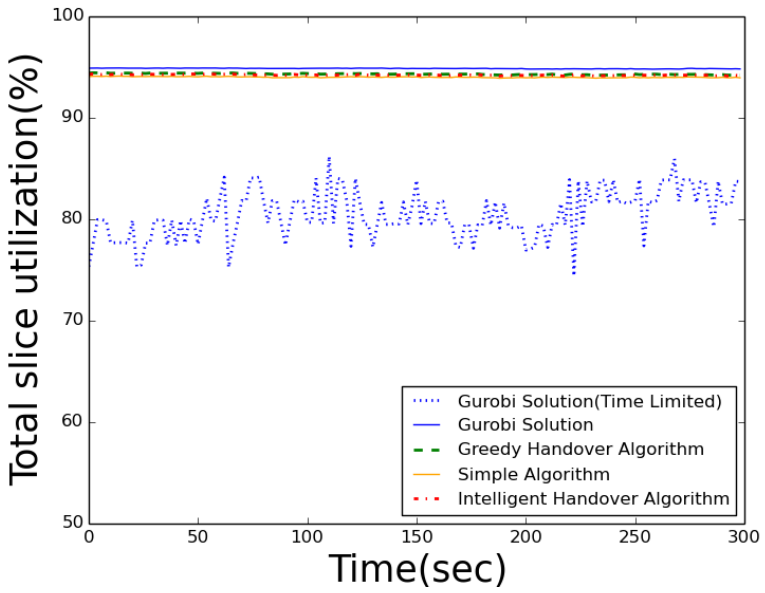

Figure 5.11. Overall slice utilization.

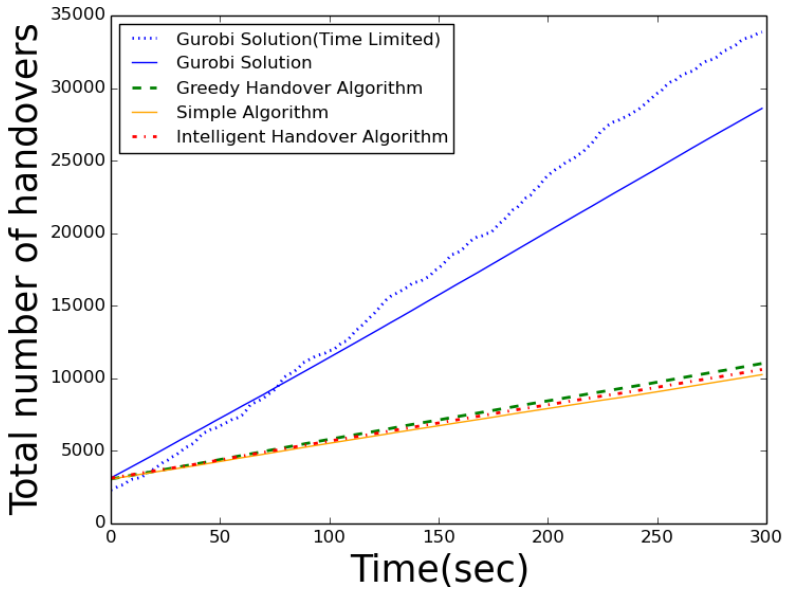

Figure 5.12. Total number of handovers.

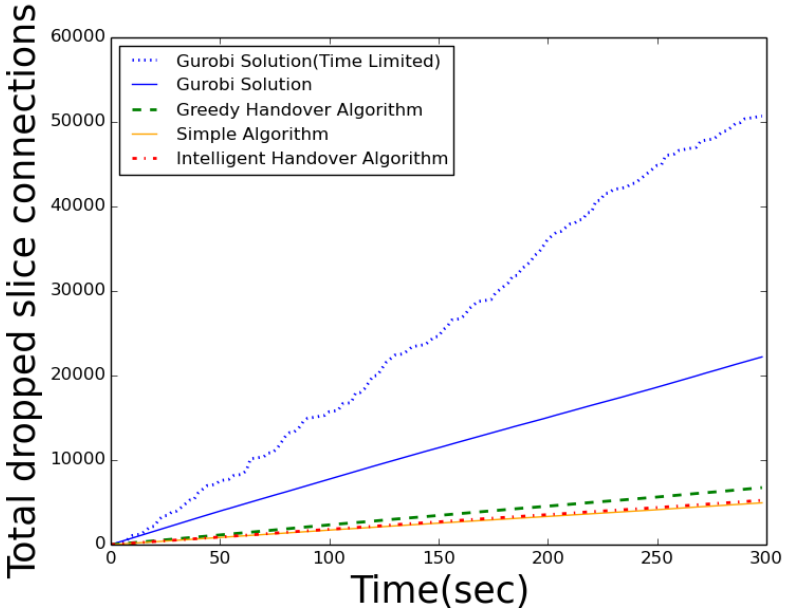

Figure 5.13. Total number of dropped slice connections.

In Figure 5.12, total number of handovers for 300 seconds iteration is shown. Handover count is considered as the sum of the following: the number of UEs entering the coverage area of any of the 12 BSs and being admitted to the respective BS, the

number of UEs that are admitted from one BS to another in our network, and the number of UEs leaving the coverage area of the 12 BSs in the network. It can be seen that Gurobi performs a lot more handovers compared to our heuristics as the simulation time increases. Gurobi time limited solution assigns fewer users initially, therefore, at the beginning of the iterations, Gurobi time limited solution performs fewer handovers compared to no time limit applied Gurobi solution. The difference between proposed heuristics and Gurobi solutions becomes apparent with the continuous run of the algorithms. Overall, our heuristics perform fewer handovers. The Simple Algorithm performs handover due to the mobility of the users whereas the Greedy Handover and Intelligent Handover algorithms perform additional handovers to increase the overall slice utilization.

In Figure 5.13, we show the total number of dropped slice connections due to handover. Slice connection drop means users are no longer able to get service from the slices that they were getting service in the previous assignment. This can be due to UEs not being connected to any BS in the network after handover, or it can be due to the BS that they are connected to after handover cannot provide service for the previously used slices. The graph shows that Gurobi causes increasing slice connection drops in both cases, a situation considered very annoying in telecommunications. Gurobi time limited solution causes more connection drops as it deviates more from the optimal, the slice assignments change frequently. On the other hand, our heuristics cause fewer connection drops. The Simple Algorithm shows the slice connection drops due to only UE mobility. The Greedy Handover Algorithm does not take into account the continuation of slice connections to make handover decisions. However, Intelligent Handover Algorithm considers the continuation of active slice connections after handover. Thus, we see fewer connection drops with the Intelligent algorithm compared to the Greedy algorithm. In other words, Intelligent Handover Algorithm performs better than the Greedy Handover Algorithm in terms of not disturbing UEs' ongoing slice connections.

In the homogeneous scenario simulations, we have shown that Gurobi is better at slice utilization but it is costly to use for our problem. On the other hand, our heuristics

find close solutions to the optimal in a short period of time. In addition, the total number of handovers and the total number of dropped slice connections assessments show that the Intelligent Handover Algorithm is a better alternative compared to our other heuristics and Gurobi.

Table 5.4. Simulation Parameters.

Parameter	**Value**
Number of BSs	12
Number of slices	4
Number of UEs	1000
Radius of the BSs	100 (in meters)
Neighboring BS distance	150 (in meters)
ω	a uniformly distributed random integer between 6 and 19 (in Mbps)
c_k^n	Generated according to UE locations
$d_{s,k}$	Bernoulli random variable with probability $P(d_{s,k} = 1) = 0.6$
$d_{1,k}$	1
b_s^n	a uniformly distributed random integer between 60 and $20 * \omega$ (in Mbps)
r_s	a uniformly distributed random integer between 1 and $\omega/3$ (in Mbps)

5.3.2. Dense Center Scenario

This simulation is carried out to observe the performance differences between our heuristics when the density of the users on the BSs is not equal. Network is created according to Table 5.4. We have modelled a network with 12 BSs and considered a scenario as shown in Figure 5.14. In Figure 5.14, the areas that are circled with red show the crowded areas. The distance between the neighboring BSs is taken as 150 meters where the coverage range of BSs is taken as 100 meters. Slice capacities in the

center BSs, BS b_1 and b_9, where the red circles are drawn, are taken as the double of their neighboring BSs since we consider these areas as gathering locations. In addition, we assume that there is an increase in demand for the slice s_1.

The simulations for the mobility analysis are done with 1000 UEs in 15 trials. The initial position of UEs are taken randomly. 60% of the UEs are distributed randomly in the network and the remaining 40% of the UEs are distributed uniformly in the red circled areas, to the coverage areas of BS b_1 and b_9. User movement is modelled as Brownian motion throughout the simulation and the movement area of the users is considered as in an unlimited space with the speed changing between $0 - 1.25$ meters per second. Simulation continues for 300 seconds and the heuristic algorithms are run every two seconds.

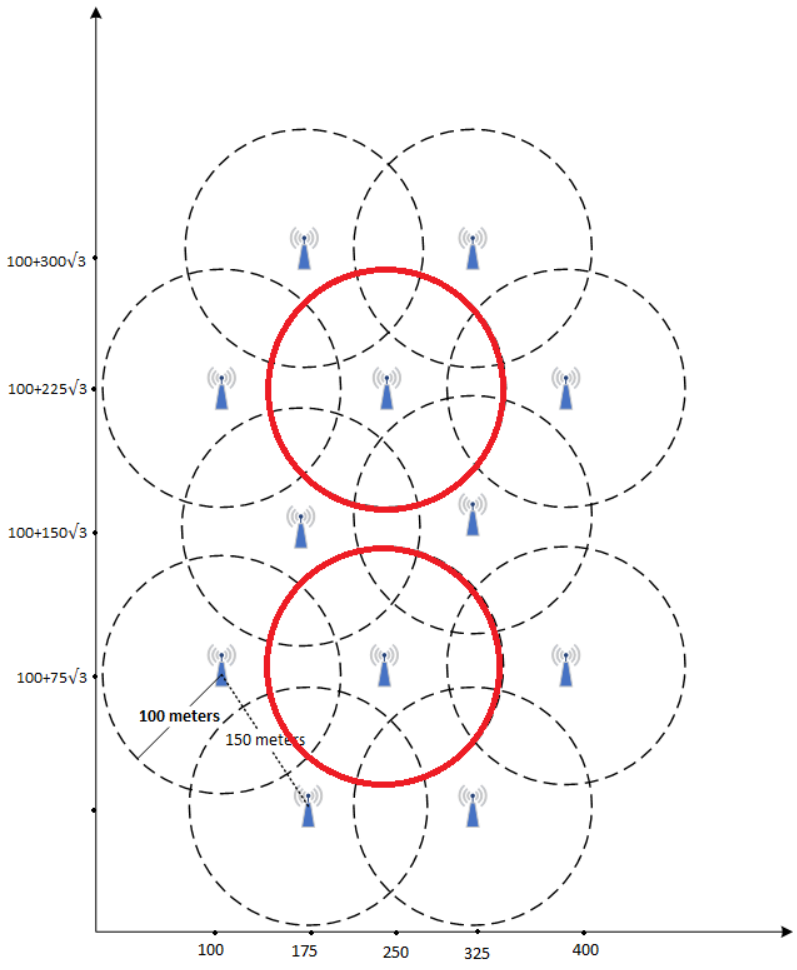

Figure 5.14. Dense Center Scenario.

In Figure 5.15, execution time of the heuristics is presented. Since the Greedy Handover Algorithm and Intelligent Handover Algorithm include additional checks and handover decision to improve slice utilization, their execution times are marginally higher.

In Figure 5.16, it can be seen that the Greedy Algorithm is the best among the heuristics to achieve high number of slice connections in the network. It is followed by the Intelligent Handover Algorithm. The gap between the two handover algorithms represent the difference between the handover decision methods that are applied. Intelligent Handover Algorithm preserves users' connection continuity and does not make solely greedy decisions.

It can be seen in Figure 5.17 that the higher slice utilization can be achieved by the handover algorithms when the number of UEs in some BSs and slices is higher than the others. With Simple Algorithm, overall utilization is lower, which is not a desirable solution for our problem.

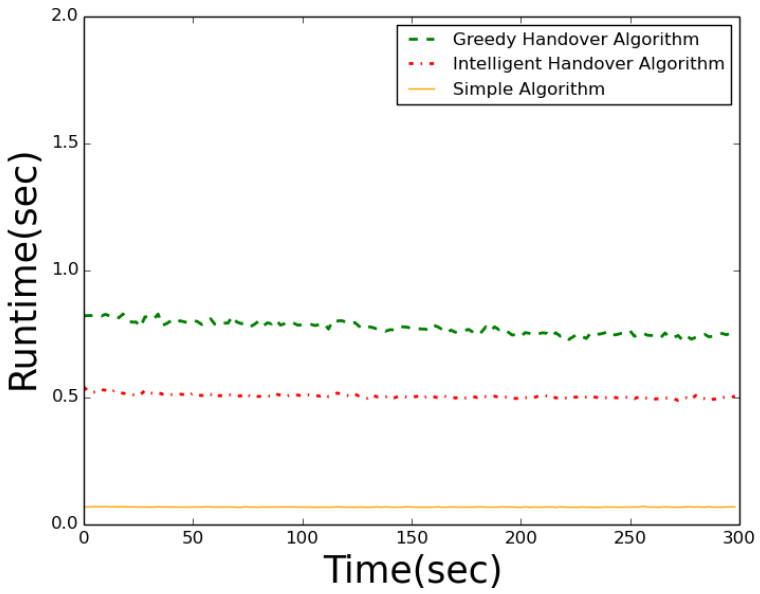

Figure 5.15. Run times of the algorithms.

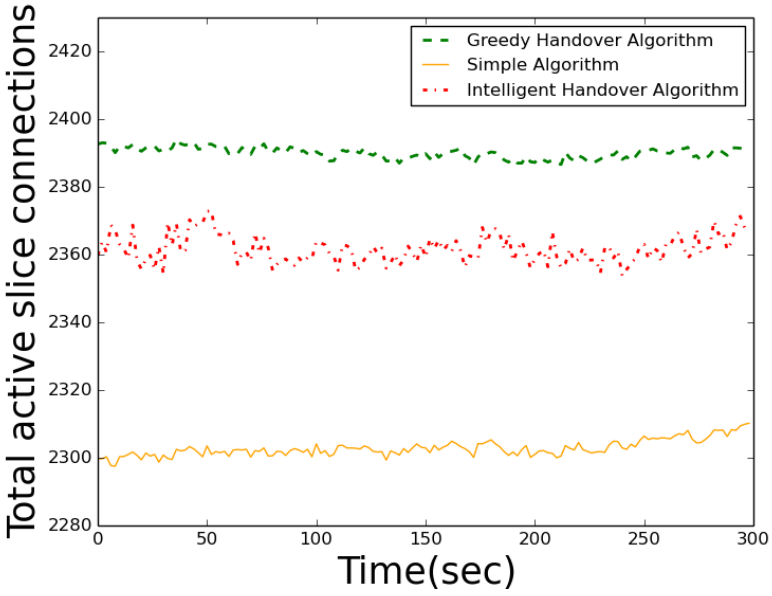

Figure 5.16. Total active slice connections.

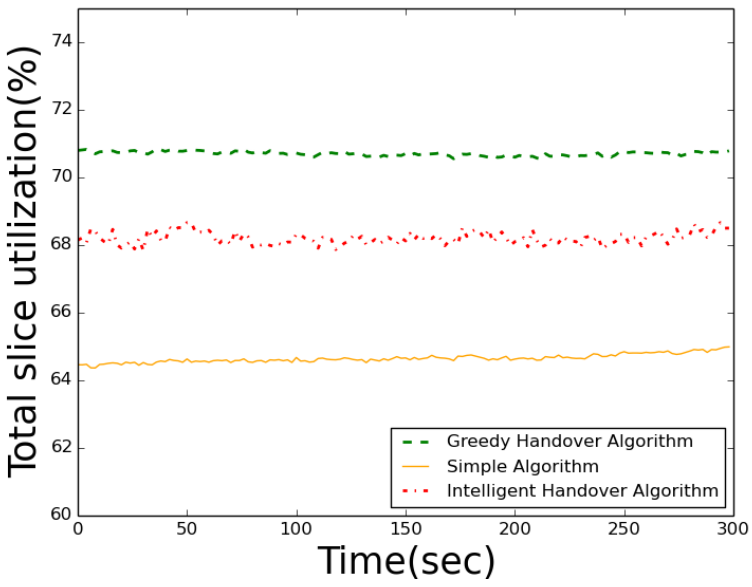

Figure 5.17. Overall slice utilization.

In Figure 5.18, we see that the number of handovers performed by the Greedy Algorithm is highest amongst all and is increasing faster as the simulation time gets higher. As we see in Figure 5.19, Greedy Algorithm causes many connection drops.

These make increasing the slice utilization very costly for some of the users in the network. Their connection can be frequently interrupted by the Greedy Handover Algorithm.

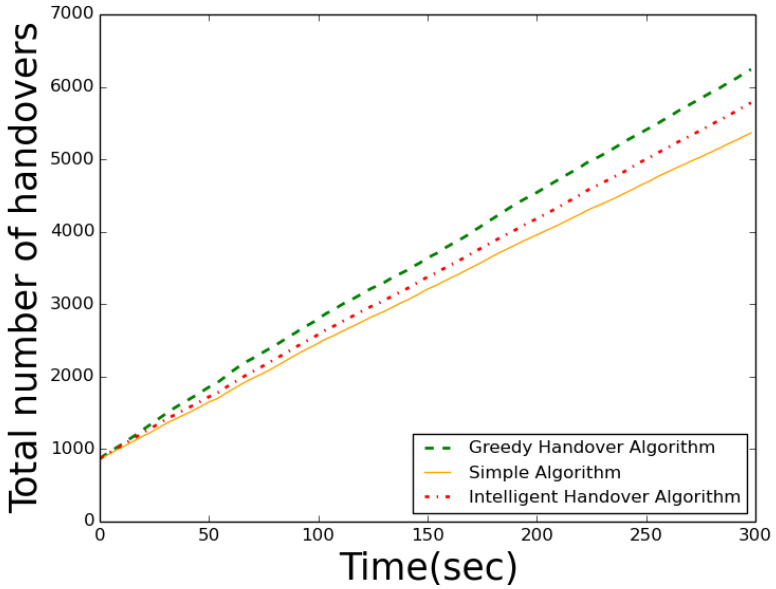

Figure 5.18. Total number of handovers.

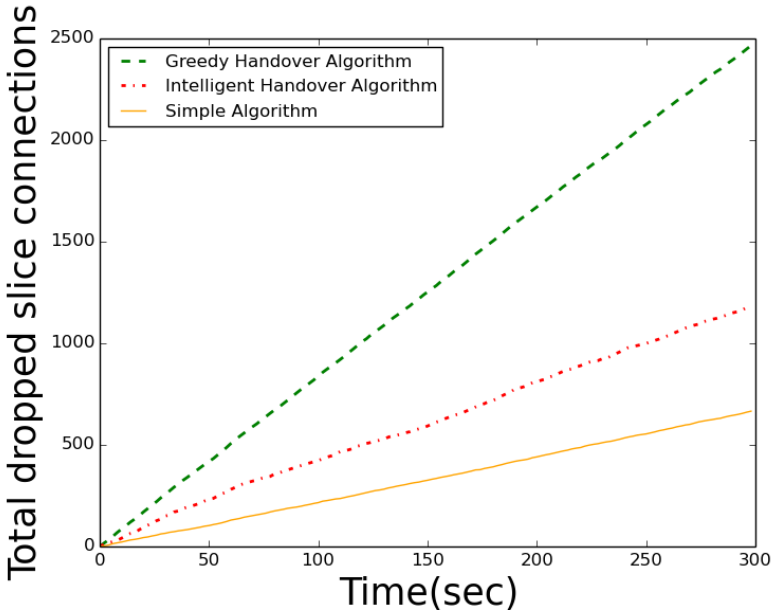

Figure 5.19. Total number of dropped slice connections.

6. CONCLUSION

In this thesis, we focused on handover to maintain the usability and high utilization of the radio resources in $5G$ networks that employ multiple slices. When there is a high demand for slice usage in crowded areas, users may not be admitted due to capacity limitations or may be provided with partial number of the slices that they request. We have defined an optimization problem to maximize the utilization of radio resources of network slices. We have used a solver for the defined optimization problem and simulated the results. We have proposed heuristic algorithms to reach similar results as the solver, but in considerably lower computing times.

Three heuristic algorithms have been proposed, namely: the Simple Algorithm, the Greedy Handover Algorithm, and the Intelligent Handover Algorithm. The Simple Algorithm admits users and allocates slices by comparing slice capacities but it lacks flexibility when the user concentration increases. The Greedy Handover Algorithm introduces handover to maximize the utilization of slices in the network without considering the existing slice connectivity of the users. As a result, after handover is performed, users may suffer from not being able to continue their communication for some slices. We have proposed the Intelligent Handover Algorithm, where handover is used to maximize the utilization of radio resources and slice connectivity of the users is taken into consideration for handover decisions to make sure that user-slice connectivity is not dropped. We have shown with the simulation results that our heuristics can present solutions close to the optimal within a short time frame.

For the sake of simplicity, we have considered the value of 'r' to be constant for each slice. Therefore, all UEs produce the same load for a specific slice and fairness is not an issue. Yet, we believe that this is a constraint that needs to be relaxed and will eventually create a fairness issue. As a future work, different UEs using the same slice may generate different offered load and fairness-aware algorithms may be designed for this purpose.

REFERENCES

1. Dharmadhikari, O., *Network Slicing: Building Next-Generation Wireless Networks*, 2018, `www.cablelabs.com/network-slicing-building-next-generation-wireless-networks`, accessed in March 2019.

2. Ericsson, *Ericsson Mobility Report*, June 2020, `www.ericsson.com/49da93/assets/local/mobility-report/documents/2020/june2020-ericsson-mobility-report.pdf`, accessed on Oct. 1, 2020.

3. Lutu, A., D. Perino, M. Bagnulo, E. Frias-Martinez and J. Khangosstar, "A Characterization of the COVID-19 Pandemic Impact on a Mobile Network Operator Traffic", *Proceedings of the ACM Internet Measurement Conference*, 2020.

4. Favale, T., F. Soro, M. Trevisan, I. Drago and M. Mellia, "Campus Traffic and E-Learning During COVID-19 Pandemic", *Computer Networks*, Vol. 176, pp. 1–9, 2020.

5. Akyildiz, I. F., S. Nie, S.-C. Lin and M. Chandrasekaran, "5G Roadmap: 10 Key Enabling Technologies", *Computer Networks*, Vol. 106, pp. 17–48, 2016.

6. Samdanis, K., X. Costa-Perez and V. Sciancalepore, "From Network Sharing to Multi-Tenancy: The 5G Network Slice Broker", *IEEE Communications Magazine*, Vol. 54, No. 7, pp. 32–39, 2016.

7. Costa-Pérez, X., J. Swetina, T. Guo, R. Mahindra and S. Rangarajan, "Radio Access Network Virtualization for Future Mobile Carrier Networks", *IEEE Communications Magazine*, Vol. 51, No. 7, pp. 27–35, 2013.

8. Foukas, X., G. Patounas, A. Elmokashfi and M. K. Marina, "Network Slicing in 5G: Survey and Challenges", *IEEE Communications Magazine*, Vol. 55, No. 5, pp.

94–100, 2017.

9. Rost, P., C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega *et al.*, "Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks", *IEEE Communications Magazine*, Vol. 55, No. 5, pp. 72–79, 2017.

10. Sattar, D. and A. Matrawy, "DSAF: Dynamic Slice Allocation Framework for 5G Core Network", *CoRR*, Vol. abs/1905.03873, 2019, `http://arxiv.org/abs/1905.03873`.

11. Halabian, H., "Optimal Distributed Resource Allocation in 5G Virtualized Networks", *IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*, pp. 28–35, 2019.

12. Zhang, H., N. Liu, X. Chu, K. Long, A.-H. Aghvami and V. C. Leung, "Network Slicing Based 5G and Future Mobile Networks: Mobility, Resource Management, and Challenges", *IEEE Communications Magazine*, Vol. 55, No. 8, pp. 138–145, 2017.

13. Yoo, T., "Network Slicing Architecture for 5G Network", *International Conference on Information and Communication Technology Convergence (ICTC)*, pp. 1010–1014, 2016.

14. Kaloxylos, A., "A Survey and an Analysis of Network Slicing in 5G Networks", *IEEE Communications Standards Magazine*, Vol. 2, No. 1, pp. 60–65, 2018.

15. Agiwal, M., A. Roy and N. Saxena, "Next Generation 5G Wireless Networks: A Comprehensive Survey", *IEEE Communications Surveys & Tutorials*, Vol. 18, No. 3, pp. 1617–1655, 2016.

16. MacCartney, G. R. and T. S. Rappaport, "Millimeter-Wave Base Station Diversity for 5G Coordinated Multipoint (CoMP) Applications", *IEEE Transactions on*

Wireless Communications, Vol. 18, No. 7, pp. 3395–3410, 2019.

17. Prabu, R. T., M. Benisha, V. T. Bai and V. Yokesh, "Millimeter Wave for 5G Mobile Communication Application", *2nd International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*, pp. 236–240, 2016.

18. Mitra, R. N. and D. P. Agrawal, "5G Mobile Technology: A Survey", *ICT Express*, Vol. 1, No. 3, pp. 132–137, 2015.

19. Yousaf, F. Z., M. Bredel, S. Schaller and F. Schneider, "NFV and SDN—Key Technology Enablers for 5G Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 35, No. 11, pp. 2468–2478, 2017.

20. Nguyen, T.-T., C. Bonnet and J. Harri, "SDN-Based Distributed Mobility Management for 5G Networks", *2016 IEEE Wireless Communications and Networking Conference*, pp. 1–7, 2016.

21. Kreutz, D., F. M. Ramos, P. E. Verissimo, C. E. Rothenberg, S. Azodolmolky and S. Uhlig, "Software-Defined Networking: A Comprehensive Survey", *Proceedings of the IEEE*, Vol. 103, No. 1, pp. 14–76, 2014.

22. An, J., K. Yang, J. Wu, N. Ye, S. Guo and Z. Liao, "Achieving Sustainable Ultra-Dense Heterogeneous Networks for 5G", *IEEE Communications Magazine*, Vol. 55, No. 12, pp. 84–90, 2017.

23. Sevim, O., H. Y. Öksüz and M. Akar, "Joint Frequency and Power Control for Self–Organizing OFDMA Femtocell Networks", *IEEE Transactions on Vehicular Technology*, Vol. 69, No. 5, pp. 5089–5101, 2020.

24. Bashir, A. K., R. Arul, S. Basheer, G. Raja, R. Jayaraman and N. M. F. Qureshi, "An Optimal Multitier Resource Allocation of Cloud RAN in 5G Using Machine Learning", *Transactions on Emerging Telecommunications Technologies*, Vol. 30,

No. 8, pp. 1–20, 2019.

25. Kitindi, E. J., S. Fu, Y. Jia, A. Kabir and Y. Wang, "Wireless Network Virtualization with SDN and C-RAN for 5G Networks: Requirements, Opportunities, and Challenges", *IEEE Access*, Vol. 5, pp. 19099–19115, 2017.

26. Barakabitze, A. A., A. Ahmad, R. Mijumbi and A. Hines, "5G Network Slicing Using SDN and NFV: A Survey of Taxonomy, Architectures and Future Challenges", *Computer Networks*, Vol. 167, pp. 1–40, 2020.

27. Mijumbi, R., J. Serrat, J.-L. Gorricho, N. Bouten, F. De Turck and R. Boutaba, "Network Function Virtualization: State-of-the-Art and Research Challenges", *IEEE Communications Surveys & Tutorials*, Vol. 18, No. 1, pp. 236–262, 2015.

28. Ishii, H., Y. Kishiyama and H. Takahashi, "A Novel Architecture for LTE-B: C-plane/U-plane Split and Phantom Cell Concept", *2012 IEEE Globecom Workshops*, pp. 624–630, 2012.

29. Ksentini, A., M. Bagaa and T. Taleb, "On Using SDN in 5G: The Controller Placement Problem", *2016 IEEE Global Communications Conference (GLOBECOM)*, pp. 1–6, 2016.

30. ETSI, *System Architecture for the 5G System*, July 2020, `https://www.etsi.org/deliver/etsi_ts/123500_123599/123501/16.05.00_60/ts_123501v160500p.pdf`, accessed on Sep. 10, 2020.

31. Kekki, S., W. Featherstone, Y. Fang, P. Kuure, A. Li, A. Ranjan, D. Purkayastha, F. Jiangping, D. Frydman, G. Verin *et al.*, "MEC in 5G Networks", *ETSI White Paper*, Vol. 28, pp. 1–28, 2018.

32. Brown, G., *Service-Based Architecture for 5G Core Networks*, 2017, `www.huawei.com/en/press-events/news/2017/11/HeavyReading-WhitePaper-5G-Core-Network`, accessed on Oct. 1, 2020.

33. Kammoun, A., N. Tabbane, G. Diaz, A. Dandoush and N. Achir, "End-to-End Efficient Heuristic Algorithm for 5G Network Slicing", *2018 IEEE 32nd International Conference on Advanced Information Networking and Applications (AINA)*, pp. 386–392, 2018.

34. Wang, G., G. Feng, T. Q. Quek, S. Qin, R. Wen and W. Tan, "Reconfiguration in Network Slicing—Optimizing the Profit and Performance", *IEEE Transactions on Network and Service Management*, Vol. 16, No. 2, pp. 591–605, 2019.

35. Qiang, L., J. Li and C. Touati, "A User Centered Multi-Objective Handoff Scheme for Hybrid 5G Environments", *IEEE Transactions on Emerging Topics in Computing*, Vol. 5, No. 3, pp. 380–390, 2016.

36. Lu, Y., X. Chen, R. Xi and Y. Chen, "An Access Selection Mechanism in 5G Network Slicing", *IEEE International Conference on Smart Internet of Things (SmartIoT)*, pp. 72–78, 2020.

37. Sciancalepore, V., K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia and A. Banchs, "Mobile Traffic Forecasting for Maximizing 5G Network Slicing Resource Utilization", *INFOCOM-IEEE International Conference on Computer Communications*, pp. 1–9, 2017.

38. Perveen, A., M. Patwary and A. Aneiba, "Dynamically Reconfigurable Slice Allocation and Admission Control within 5G Wireless Networks", *IEEE 89th Vehicular Technology Conference*, pp. 1–7, 2019.

39. Jiang, M., M. Condoluci and T. Mahmoodi, "Network Slicing Management & Prioritization in 5G Mobile Systems", *22th European Wireless Conference*, pp. 1–6, VDE, 2016.

40. Sun, Y., G. Feng, L. Zhang, M. Yan, S. Qin and M. A. Imran, "User Access Control and Bandwidth Allocation for Slice-Based 5G-and-Beyond Radio Access

Networks", *IEEE International Conference on Communications (ICC)*, pp. 1–6, 2019.

41. Sun, Y., W. Jiang, G. Feng, P. V. Klaine, L. Zhang, M. A. Imran and Y.-C. Liang, "Efficient Handover Mechanism for Radio Access Network Slicing by Exploiting Distributed Learning", *IEEE Transactions on Network and Service Management*, Vol. 17, No. 4, pp. 2620–2633, 2020.

42. Li, R., Z. Zhao, Q. Sun, I. Chih-Lin, C. Yang, X. Chen, M. Zhao and H. Zhang, "Deep Reinforcement Learning for Resource Management in Network Slicing", *IEEE Access*, Vol. 6, pp. 74429–74441, 2018.

43. Wang, H., Y. Wu, G. Min, J. Xu and P. Tang, "Data-Driven Dynamic Resource Scheduling for Network Slicing: A Deep Reinforcement Learning Approach", *Information Sciences*, Vol. 498, pp. 106–116, 2019.

44. Thantharate, A., R. Paropkari, V. Walunj and C. Beard, "Deepslice: A Deep Learning Approach Towards an Efficient and Reliable Network Slicing in 5G Networks", *IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 0762–0767, 2019.

45. Lee, Y. L., J. Loo, T. C. Chuah and L.-C. Wang, "Dynamic Network Slicing for Multitenant Heterogeneous Cloud Radio Access Networks", *IEEE Transactions on Wireless Communications*, Vol. 17, No. 4, pp. 2146–2161, 2018.

46. Farahmand, F., Q. Zhang and J. P. Jue, "Dynamic Traffic Grooming in Optical Burst-Switched Networks", *Journal of Lightwave Technology*, Vol. 23, No. 10, pp. 3167–3177, 2005.

47. Ul-Mustafa, R. and A. E. Kamal, "Many-to-One Traffic Grooming with Aggregation in WDM Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 24, No. 8, pp. 68–81, 2006.

48. Li, Q. C., G. Wu, A. Papathanassiou and M. Udayan, "An End-to-End Network Slicing Framework for 5G Wireless Communication Systems", *CoRR*, Vol. abs/1608.00572, 2016, `http://arxiv.org/abs/1608.00572`.

49. Chong, E. K. and S. H. Zak, *An Introduction to Optimization*, John Wiley & Sons, 2004.

50. Pochet, Y. and L. A. Wolsey, *Production Planning by Mixed Integer Programming*, Springer Science & Business Media, 2006.

51. Floudas, C. A., *Nonlinear and Mixed-Integer Optimization: Fundamentals and Applications*, Oxford University Press, 1995.

52. Burer, S. and A. N. Letchford, "Non-Convex Mixed-Integer Nonlinear Programming: A Survey", *Surveys in Operations Research and Management Science*, Vol. 17, No. 2, pp. 97–106, 2012.

53. Glover, F. and E. Woolsey, "Converting the 0-1 Polynomial Programming Problem to a 0-1 Linear Program", *Operations Research*, Vol. 22, No. 1, pp. 180–182, 1974.

54. Gurobi Optimization, L., *Gurobi Optimizer Reference Manual*, 2021, `https://www.gurobi.com`.

55. Mitten, L., "Branch-and-Bound Methods: General Formulation and Properties", *Operations Research*, Vol. 18, No. 1, pp. 24–34, 1970.

56. Wegener, I., *Complexity Theory: Exploring the Limits of Efficient Algorithms*, Springer Science & Business Media, 2005.