**POLITECNICO DI TORINO**

Corso di Laurea Magistrale in Ingegneria Meccatronica

Tesi di Laurea Magistrale

# Optimization of User–LEO Satellite Assignments


**Relatore**
Prof. Alberto Tarable

**Candidato**
Ali Zein Khalifeh

firma del relatore

firma del candidato

Anno Accademico 2024–2025

# Abstract

Low Earth Orbit (LEO) satellite networks, such as Starlink SpaceX (2020), are becoming an important solution to provide global Internet access due to their low latency and wide coverage. However, the fast movement of LEO satellites makes it difficult to maintain stable connections, as users must frequently switch from one satellite to another.

This thesis focuses on improving the way users are assigned to satellites in such a fast-changing environment. We propose a method where each user connects to two satellites at the same time, a primary and a secondary, to ensure smoother transitions during handovers. To determine the best user-satellite pairings at every moment, we use the Munkres algorithm (Hungarian), considering satellite load, link quality, and connection stability.

Our simulations use realistic assumptions, including satellite movement, visibility, and signal fading, following the ITU-R P.681 model International Telecommunication Union (2017). We explore both an **ideal lower bound** based on instantaneous channel state information (CSI) and a predictive handover strategy based on averaged channel profiles. These correspond to analyzing instant and average signal quality, respectively. Key metrics such as signal-to-noise ratio (SNR), data rate, and service continuity are used to assess performance.

The results show that having two connections per user greatly improves stability and data throughput, especially when channel conditions change rapidly. We also analyze how different transmission power levels affect performance and examine the trade-offs between complexity and system gains. This approach builds on recent work on soft handover strategies for LEO networks Ben Salem et al. (2025); Feng et al. (2020), offering a scalable way to improve handovers in satellite Internet systems.

# List of Figures

# List of Tables

# Contents

# Chapter 1

# Introduction

In recent years, the vision of delivering fast and reliable Internet access via satellite has become increasingly feasible with the emergence of low-earth orbit (LEO) satellite constellations. These satellites orbit the Earth at low altitudes, typically between 500 and 2,000 kilometers, and traverse the sky at high speeds. Their proximity to Earth enables stronger signals and significantly lower communication delays compared to traditional satellite systems, making LEO networks ideal for latency-sensitive applications such as video conferencing, online gaming, and real-time data services. However, despite these advantages, LEO systems present unique challenges, particularly in maintaining stable connections during the rapid and frequent transitions between satellites.

To contextualize the role of LEO satellites, it is helpful to briefly contrast them with other orbital categories used in satellite communications. Geostationary Earth Orbit (GEO) satellites operate at approximately 36,000 kilometers above the Earth's surface. At this altitude, they complete one orbit per day, synchronized with the Earth's rotation, thus appearing stationary from the ground. This allows GEO satellites to provide continuous coverage to specific regions using fixed-ground antennas. Although this stability and broad coverage footprint are advantageous for applications such as broadcasting and long-distance communication, the significant signal propagation delay, often around 250 milliseconds one-way—limits their effectiveness in real-time scenarios.

Medium-Earth Orbit (MEO) satellites operate between GEO and LEO, typically at altitudes ranging from 8,000 to 20,000 kilometers. MEO satellites are commonly used for global navigation systems such as GPS, Galileo, and GLONASS. They offer a trade-off between latency and coverage, with moderate propagation delays and longer satellite visibility windows compared to LEO systems. However, they still require tracking mechanisms and occasional handovers as a result of their motion relative to Earth.

LEO satellites, on the contrary, orbit at much lower altitudes and complete a full revolution in approximately 90 to 120 minutes. Consequently, they can serve a ground user for only a few minutes before moving out of range. Maintaining continuous service thus necessitates frequent handovers between satellites. If these handovers are not executed seamlessly, users may experience service degradation, interruptions, or complete link loss.

Unlike GEO systems with fixed ground antennas, LEO networks require ground stations

or user terminals to dynamically track satellites and manage transitions. This introduces considerable complexity into the network architecture. Moreover, unlike terrestrial systems, where users typically move while the infrastructure remains stationary, LEO networks involve highly mobile nodes, creating an inverse scenario that demands new handover and assignment strategies.

## 1.1 Motivation and Problem Statement

The global demand for high-speed, low-latency connectivity continues to grow, particularly in underserved or remote regions where terrestrial infrastructure is sparse or nonexistent. LEO satellite systems, such as the Starlink constellation SpaceX (2020), are uniquely positioned to bridge this digital divide. However, realizing their full potential critically depends on addressing the handover challenge.

Unlike terrestrial cellular networks, where base stations are stationary and handovers occur relatively infrequently, LEO satellites move rapidly across the sky, leading to frequent and unavoidable user-to-satellite reassignments. This creates a fundamentally different and more demanding handover scenario Juan et al. (2022); Kodheli et al. (2021).

Moreover, link quality in satellite networks is subject to rapid fluctuations due to obstructed line-of-sight paths, multipath propagation, and varying atmospheric conditions. These impairments have been extensively documented in propagation models and satellite communication studies International Telecommunication Union (2017); Jung et al. (2022); Maral et al. (2020), and they significantly complicate the decision process for robust handover management.

Ensuring seamless and efficient assignments under such time-varying conditions, while respecting the limited capacity of each satellite, requires fast and scalable algorithms. Prior research has shown that the design of efficient assignment strategies is essential to guarantee both continuity and fairness in LEO systems Miao et al. (2019); Liu et al. (2024). Developing such strategies is therefore a non-trivial problem, with significant implications for network performance, user experience, and overall scalability.

## 1.2 Objective of the Thesis

This thesis aims to develop and evaluate an optimization framework for user-to-satellite assignment in LEO constellations. A dual-handover strategy is proposed, in which each user maintains two simultaneous satellite links, a primary and a secondary, to enhance robustness during transitions. The assignment problem is solved at each time step using the Munkres algorithm (Hungarian) Kuhn (1955); Munkres (1957), subject to constraints such as satellite capacity, quality of service (QoS) thresholds, and penalties for frequent handovers.

The overarching objective is to maximize the network sum rate through an efficient and fair allocation of users to satellites, while simultaneously ensuring service continuity and

mitigating the negative impact of excessive handovers. This framework therefore balances throughput maximization with robustness and user experience, addressing the key challenges posed by the dynamic nature of LEO satellite networks.

## 1.3 Methodology Overview

The proposed framework incorporates realistic system modeling, including:

- Satellite motion and user visibility simulation based on a dense LEO constellation.
- Elevation angle-based channel modeling using the ITU-R P.681 model International Telecommunication Union (2017).
- Maximization of the key performance indicators such as SNR and achievable data rate per link.
- Assignment based on Munkres-based cost-optimized algorithm.

The effectiveness of the algorithms designed is evaluated through extensive Monte Carlo simulations, which account for the randomness of user locations, dynamic constellation geometry, and channel variations. Performance is analyzed under varying transmission powers, channel conditions, and user distributions. Key metrics include average SNR, throughput, handover frequency, and link continuity.

## 1.4 Thesis Organization

This thesis is organized as follows.

- **Chapter 2** provides a basic background on satellite communications and categorizes orbital regimes.
- **Chapter 3** reviews existing handover strategies in LEO satellite networks.
- **Chapter 4** presents the proposed optimization framework, detailing the dual assignment model and the matching algorithm.
- **Chapter 5** introduces the simulation setup, defines key performance metrics, presents the results, and discusses their implications in depth.
- **Chapter 6** concludes the thesis and outlines potential directions for future research.

# Chapter 2

# Satellite Communication Systems

Satellite communication systems rely on orbiting spacecraft networks to establish links between geographically distant points on Earth. Depending on their orbital altitude and dynamics, satellites are classified into three main categories. Geostationary Earth Orbit (GEO), Medium Earth Orbit (MEO), and Low Earth Orbit (LEO). Each regime presents distinct advantages and challenges that shape the performance and design of satellite-based services in various applications.

## 2.1 Classification of Satellite Orbits

### 2.1.1 Geostationary Earth Orbit (GEO)

GEO satellites are positioned approximately $\sim 35{,}786\,\text{km}$ above the Earth's equator. At this altitude, they complete one orbit every 24 hours, matching Earth's rotation and appearing stationary to a ground observer. This stationary footprint enables GEO satellites to provide continuous coverage over a fixed geographic region, making them ideal for applications such as television broadcasting, weather monitoring, and long-distance communication.

The principal advantage of **GEO systems** lies in their **persistent coverage and broad footprint**: each geostationary satellite can view nearly one-third of the Earth's surface, meaning just a **small number of satellites**—often three, spaced appropriately—can provide **near-global service** (Maral et al., 2020). However, the **high altitude** of GEO orbits ($\sim 35{,}786\,\text{km}$) leads to **substantial propagation delays**—on the order of **250 ms one-way**—which are far greater than those experienced in terrestrial networks (Smith & Lee, 2023). This high latency, coupled with increased **free-space path loss (FSPL)**, makes GEO communication systems **suboptimal for latency-sensitive applications** such as video conferencing or interactive gaming (CNR IRIS Research, 2021).

### 2.1.2 Medium Earth Orbit (MEO)

Medium Earth Orbit (MEO) satellites operate at altitudes between Low Earth Orbit (LEO) and Geostationary Orbit (GEO), typically ranging from about 2,000 to 20,000 km. Their orbital periods vary from a few hours up to approximately 12 hours, depending on altitude, with navigation systems such as GPS, Galileo, and GLONASS commonly deployed in these orbits (Vallado, 2013; Kaplan & Hegarty, 2005). MEO offers an attractive trade-off between coverage and latency: compared to GEO satellites, they provide significantly lower round-trip delays (down to 125 ms vs. ∼600 ms), while still maintaining longer visibility durations (2–8 hours) than LEO satellites (Maral et al., 2020; Kodheli et al., 2021; CNR IRIS Research, 2021; Smith & Lee, 2023). Although they are not geostationary, their slower relative motion compared to LEO reduces the frequency of handovers, which simplifies ground terminal operations (Hofmann-Wellenhof et al., 2007).

### 2.1.3 Low Earth Orbit (LEO)

LEO satellites orbit at altitudes between 500 and 2,000 kilometers and travel at speeds of approximately 7.8 km/s. They complete a full orbit in 90 to 120 minutes, offering low-latency communication links with round-trip times as low as 25–40 milliseconds. This latency is comparable to fiber-optic terrestrial networks (Juan et al., 2022; SpaceX, 2020).

However, due to their high velocity and limited visibility duration, LEO satellites necessitate frequent handovers to maintain continuous service. Ground terminals must constantly track moving satellites and switch links accordingly. Despite these complexities, the lower path loss and minimal delay make LEO networks highly attractive for modern broadband services (Juan et al., 2022; Miao et al., 2019; SpaceX, 2020).

In recent years, the deployment of large-scale LEO constellations such as Starlink, OneWeb, and Kuiper has fundamentally transformed the landscape of satellite communications. These commercial systems plan to operate thousands of satellites at altitudes between 500 km and 1,200 km, providing global broadband coverage with unprecedented capacity and low latency SpaceX (2020); Juan et al. (2022); Abdu et al. (2023). The dynamic topology of these mega-constellations introduces new challenges for network management, particularly in terms of mobility support, frequent handovers, and radio resource allocation Liu et al. (2024).

These considerations motivate the choice of a realistic, large-scale LEO constellation as the basis for simulation and performance evaluation in this thesis.

### 2.1.4 Why GEO Satellites Appear Fixed and LEO/MEO Satellites Move

The apparent motion of a satellite as seen from the ground depends fundamentally on its orbital period relative to the Earth's rotation.

**Geostationary Earth Orbit (GEO) satellites** are positioned at an altitude of approximately 35,786 km above the equator. At this altitude, the orbital period of a satellite matches the Earth's rotational period (24 hours). This means that as the Earth rotates, the GEO satellite completes one orbit in the same time and at the same angular velocity. As a result, the GEO satellite appears stationary to a ground observer, remaining fixed above a specific longitude. This unique property allows for continuous coverage of a particular region on Earth's surface (Vallado, 2013; Maral et al., 2020).

In contrast, **Low Earth Orbit (LEO)** and **Medium Earth Orbit (MEO)** satellites are placed at much lower altitudes (typically 500–2,000 km for LEO and 2,000–20,000 km for MEO). At these altitudes, the gravitational pull is stronger, requiring satellites to travel at higher orbital velocities to maintain a stable orbit. According to Kepler's third law, the square of the orbital period is proportional to the cube of the orbit's semi-major axis, which implies that satellites closer to Earth must orbit faster (Wertz & Larson, 1999; Vallado, 2013).

$$T^2 = \frac{4\pi^2}{\mu} a^3, \tag{2.1}$$

where $T$ is the orbital period, $a$ is the semi-major axis of the orbit, and $\mu = GM$ is Earth's standard gravitational parameter (Wertz & Larson, 1999; Vallado, 2013). This relation, known as Kepler's third law, shows that the orbital period increases with the distance from Earth. As a result, satellites in lower orbits complete revolutions much faster: approximately 90–120 minutes for LEO and 2–12 hours for MEO. Since their orbital periods are much shorter than the Earth's rotational period (24 hours), these satellites appear to move rapidly across the sky from the perspective of a ground-based observer.

It is **physically impossible** for LEO and MEO satellites to remain fixed over a single point on the Earth's surface. To do so, a satellite would need to maintain a 24-hour orbital period at a much lower altitude than GEO. However, at those altitudes, the gravitational force is much stronger, and the satellite would have to move too slowly to sustain a stable orbit. As a result, it would inevitably fall back to Earth. Only at the specific altitude of GEO does the balance between gravitational force and orbital velocity allow for a geostationary orbit. According to the European Space Agency (ESA), only at a precise altitude of approximately 35,786 km does a satellite's orbital period match Earth's rotation—making a geostationary orbit possible. At lower altitudes such as LEO or MEO, satellites travel much faster and cannot remain fixed above a point on Earth European Space Agency (ESA) (n.d.).

Table 2.1: Comparison of Satellite Orbits and Apparent Motion Relative to the Earth

| **Orbit Type** | **Altitude (km)** | **Orbital Period** | **Appears Fixed?** |
|---|---|---|---|
| GEO | ∼35,786 | 24 hours | Yes |
| MEO | 2,000–20,000 | 2–12 hours | No |
| LEO | 500–2,000 | 1.5–2 hours | No |

These physical constraints explain why only GEO satellites can appear stationary from

the ground, while all other satellites exhibit apparent motion and necessitate frequent handover mechanisms, which is a central focus of this thesis.

**Orbital Velocity and Gravitational Balance:**

A satellite remains in a stable orbit not by counteracting gravity, but by continuously falling toward Earth while moving forward fast enough that the surface curves away beneath it. This delicate balance between gravitational pull and tangential velocity results in a stable orbital path. For circular orbits, this condition is achieved when the gravitational force equals the required centripetal force:

$$\frac{GMm}{r^2} = \frac{mv^2}{r}, \tag{2.2}$$

where $G$ is the gravitational constant, $M$ is Earth's mass, $m$ is the satellite's mass, $r$ is the orbital radius, and $v$ is the satellite's orbital speed. Solving for $v$ gives:

$$v = \sqrt{\frac{GM}{r}}. \tag{2.3}$$

This shows that satellites in lower orbits must travel at higher velocities to stay in balance. For instance, LEO satellites at 500–2,000 km altitude orbit at speeds around 7.8 km/s, while GEO satellites, much farther out, require only about 3.1 km/s. This difference explains why LEO satellites have shorter orbital periods and appear to move rapidly across the sky Wertz & Larson (1999).

**Origin of Tangential Velocity:**

The necessary tangential velocity is imparted by the launch vehicle during orbital insertion. After reaching the desired altitude, the rocket performs a horizontal burn to accelerate the satellite to the required orbital speed. This tangential motion ensures that, as gravity pulls the satellite inward, it continuously falls around the Earth rather than directly back to the surface. In this sense, an orbit represents a state of perpetual free-fall, where the forward velocity of the satellite precisely matches the curvature of the planet. If the velocity is insufficient, the satellite will spiral back to Earth, while an excessive velocity would cause it to escape Earth's gravitational influence Vallado (2013); Wertz & Larson (1999).

## 2.2 LEO Megaconstellations

Several LEO megaconstellations have been proposed and deployed to meet the increasing demand for global broadband connectivity. Among them, the most advanced is the **Starlink constellation**, developed by SpaceX, which envisions more than **4,000 satellites** distributed across multiple orbital shells, with altitudes ranging from **340 to 570 km** and inclinations between **53° and 97.6°** (SpaceX, 2020). Starlink aims to provide high-throughput, low-latency connectivity to users worldwide.

**OneWeb** is another major initiative, targeting near-global coverage with an initial deployment of **648 satellites** in polar orbits at an altitude of around **1,200 km**. Similarly,

**Amazon's Project Kuiper** has received regulatory approval for deploying up to **3,236 satellites** in LEO to provide broadband access, with altitudes between **590 and 630 km**.

Beyond these broadband-oriented constellations, the **Iridium network** represents one of the earliest operational LEO systems, consisting of **66 active satellites** in near-polar orbits at an altitude of **780 km**, primarily designed for voice and data services with global coverage.

These constellations illustrate the diversity of design choices in terms of altitude, inclination, and scale, highlighting the trade-offs between latency, coverage, capacity, and cost. They also provide the real-world motivation for studying efficient user-to-satellite assignment and handover strategies in highly dynamic multi-satellite environments (Juan et al., 2022; Kodheli et al., 2021).

## 2.3 Towards 6G: The Role of Non-Terrestrial Networks

Looking beyond current satellite deployments, the future of global connectivity is increasingly defined by the convergence of terrestrial and non-terrestrial networks. Sixth-generation (6G) wireless systems are expected to move beyond the boundaries of traditional ground-based infrastructure by integrating spaceborne, aerial, and terrestrial components into a seamless communication fabric. In this emerging multi-layer architecture, low Earth orbit (LEO) satellites will work alongside high-altitude platforms and unmanned aerial vehicles, complementing terrestrial networks to deliver reliable, high-capacity connectivity on a global scale Rago et al. (2024).

This integration is not a mere extension of coverage, but a fundamental transformation in how communication networks operate. By combining these diverse technologies, 6G aims to support a wide range of use cases, including broadband access in both urban and remote areas, uninterrupted connectivity for transportation across land, sea, and air, massive Internet of Things (IoT) deployments, and robust communication during emergencies or natural disasters when terrestrial networks may be compromised. Realizing this vision, however, introduces significant challenges in network management, especially in the areas of handover coordination, dynamic resource allocation, latency control, and security across heterogeneous platforms Kodheli et al. (2021).

The evolution toward a truly global and resilient network, where "coverage everywhere, for everything, all the time" becomes a reality, places renewed emphasis on efficient handover mechanisms and intelligent user–satellite assignment strategies. These challenges, central to the 6G paradigm, provide the broader context and motivation for the research presented in this thesis.

### 2.3.1 Geometric Description of Satellite Links

In satellite communication, the geometry of the user–satellite link plays a central role. Two parameters are particularly important: the *elevation angle* $\theta$, which is the angle between the user's horizon and the satellite, and the *slant distance* $d$, which is the straight-line distance between the user terminal and the satellite. For a satellite at orbital altitude $h$ and an Earth radius $R_e$, the slant distance can be expressed as Vallado (2013); Wertz & Larson (1999):

$$d = \sqrt{(R_e + h)^2 - (R_e \cos\theta)^2} - R_e \sin\theta. \tag{2.4}$$

The elevation angle directly impacts link quality: low elevation angles correspond to longer slant distances, higher path losses, and an increased probability of obstruction Maral et al. (2020).

### 2.3.2 Channel Modeling

The received signal power depends on the free-space path loss (FSPL), given by Maral et al. (2020):

$$L_{fs} = \left(\frac{4\pi df}{c}\right)^2, \tag{2.5}$$

where $f$ is the carrier frequency and $c$ is the speed of light. To capture realistic fading effects, this thesis adopts the ITU-R P.681 land mobile satellite model International Telecommunication Union (2017). It describes a two-state channel:

- **Line-of-sight (LoS) state**: dominated by a direct component subject to log-normal shadowing.
- **Non-line-of-sight (NLoS) state**: characterized by severe shadowing and multi-path scattering, typically modeled as Rayleigh fading.

The channel alternates between LoS and NLoS with exponentially distributed state durations Juan et al. (2022); Miao et al. (2019). Doppler effects, due to satellite motion at $\sim 7.8$ km/s, introduce additional frequency shifts and spreads that must be tracked to maintain synchronization Kodheli et al. (2021).

### 2.3.3 Network Architecture

Modern LEO constellations rely on a multi-tiered network architecture. Each satellite connects to the core network via a feeder link to a terrestrial gateway and simultaneously serves multiple users within its coverage footprint, known as a *cell*. Satellites may employ either Earth-fixed cells (static beams projected onto fixed regions of the surface) or Earth-moving cells (beams that move along with the satellite's trajectory). Multi-cell satellites with frequency reuse further increase system capacity Maral et al. (2020); SpaceX (2020).

### 2.3.4 Handover Fundamentals

A *handover* is the process by which an ongoing user connection is transferred from one satellite or beam to another to preserve service continuity. In terrestrial systems, handovers are usually triggered by user mobility. In contrast, in LEO networks the satellites themselves move rapidly across the sky, leading to frequent handovers even for stationary users Juan et al. (2022).

The handover process generally includes three stages:

1. **Measurement**: the terminal monitors link quality indicators such as received power, SINR, or Doppler shift.
2. **Decision**: based on predefined thresholds or optimization criteria, the network decides when to trigger the handover.
3. **Execution**: the connection is re-established with the new serving satellite or beam while minimizing service disruption.

Although conceptually simple, this process in LEO systems is complicated by short satellite visibility windows, frequent beam transitions, and limited on-board resources Jung et al. (2022).

## 2.4 The Munkres Algorithm

The Munkres algorithm, also known as the Hungarian method Kuhn (1955); Munkres (1957), is a combinatorial optimization technique designed to solve the classical assignment problem in polynomial time and is the foundation on which our assignment of the user-satellite is based. Given a cost matrix that models the cost of assigning each agent to a task, the algorithm systematically transforms the matrix through a series of reductions, coverings, and adjustments of rows and columns. At each step, it seeks to uncover a set of zero-cost assignments that minimizes the total cost (or equivalently, maximizes the total reward).

The method is guaranteed to find the global optimum with computational complexity $\mathcal{O}(n^3)$, where $n$ is the number of agents or tasks. Its efficiency and optimality have made it a standard tool in various domains, including operations research, scheduling, image processing, and, more recently, user–satellite association problems in non-terrestrial networks.

# Chapter 3

# Existing Handover Approaches in LEO Networks

The rapid expansion of Low Earth Orbit (LEO) satellite constellations has introduced new challenges for maintaining seamless connectivity, particularly due to the frequent and rapid handovers necessitated by satellites moving at high velocities Rago et al. (2024); Jung et al. (2022).

Efficient user-to-satellite assignment is therefore a cornerstone for maintaining service quality and network stability. However, traditional fixed or greedy assignment schemes often struggle to adapt to dynamic link qualities and fluctuating satellite availability. This has motivated the exploration of more sophisticated, mathematically grounded approaches in recent literature.

A diverse range of strategies has emerged, spanning algorithmic methods such as maximum weight matching Feng et al. (2020), multi-attribute decision making Miao et al. (2019), and auction-based mechanisms Jung et al. (2022), as well as physical-layer solutions including soft handover via inter-satellite link (ISL) relaying Ben Salem et al. (2025) and multi-antenna (MIMO) techniques Feng et al. (2020).

More recently, the increasing complexity of LEO network dynamics has spurred the adoption of machine learning-based algorithms Liu et al. (2024); Abdu et al. (2023), particularly deep reinforcement learning methods that predict and optimize handover decisions under time-varying channel conditions. Each of these approaches entails trade-offs in terms of signaling overhead, service continuity, computational complexity, and resilience to real-world impairments.

In the following sections, we review several prominent strategies, highlighting their operating principles, mathematical models, strengths, and limitations.

## 3.1 Maximum Weight Matching Strategy with MIMO Support

A prominent example is the work by Feng et al. Feng et al. (2020), which models the assignment problem as a bipartite graph matching task. Here, one set of nodes represents gateway stations (or users), and the other set corresponds to visible satellites. Edges are drawn between each user and each satellite within view, with edge weights reflecting the quality of the potential communication link. Specifically, the weight is determined by the channel coefficient magnitude $|H_{ijk}|$, which encapsulates the effects of satellite position, elevation angle, and other propagation factors.

What sets this approach apart is its integration of Multiple-Input Multiple-Output (MIMO) technology. By leveraging MIMO, the system can support multiple concurrent streams per link, substantially boosting the achievable data rate and enhancing link reliability, a critical advantage given the variable and often challenging LEO channel conditions. The matching algorithm accounts for these MIMO capabilities by adapting the edge weights to reflect multi-antenna gains.

The Kuhn-Munkres (Hungarian) algorithm is employed to solve for the maximum weight matching, efficiently pairing users with satellites to maximize the sum quality of all active links. Importantly, Feng et al. Feng et al. (2020) also propose algorithmic enhancements to ensure all users can be assigned, even when the number of visible satellites is limited, a common situation during certain orbital configurations.

This strategy demonstrates significant improvements in both load balancing and overall network throughput, as validated through simulation studies. However, it relies on the availability of accurate channel state information and necessitates frequent updates as the topology evolves, which can present computational and signaling challenges in large-scale deployments. Still, compared to simpler heuristics, the maximum weight matching framework offers a compelling balance between optimality and scalability, particularly when extended with physical-layer features like MIMO.

Recent research continues to expand on this foundation, exploring alternative assignment strategies such as auction-based methods Jung et al. (2022) reinforcement learning Liu et al. (2024), and multi-attribute decision algorithms Miao et al. (2019), each aiming to further enhance robustness and real-world applicability in LEO satellite networks.

## 3.2 Soft Handover via Inter-Satellite Link Relaying

A fundamentally different approach to improving handover robustness in LEO networks leverages the concept of soft handover through inter-satellite link (ISL) relaying. As explored by Ben Salem et al. Ben Salem et al. (2025), this strategy enables a ground user to establish and maintain simultaneous uplink connections with both the currently serving satellite and the upcoming target satellite during the critical handover interval.

In this scenario, one of the satellites, typically the one about to lose coverage, acts as

a relay, forwarding the received user signal to the other satellite using either amplify-and-forward (AF) or decode-and-forward (DF) protocols over the ISL. This overlapping connectivity period allows for a seamless transition, significantly reducing the risk of packet loss or service interruption that is characteristic of conventional hard handover schemes.

To capture the practical performance of this approach, Ben Salem et al. Ben Salem et al. (2025) employ system models grounded in the 3GPP channel standard and incorporate elevation-angle-based link quality analysis. Their simulation results highlight a substantial reduction in block error rate (BLER) when adopting soft handover with ISL relaying, particularly under realistic fading and mobility conditions.

Amplify-and-Forward (AF) SNR Model

$$\gamma_{\text{AF}} = \frac{\gamma_{us} \cdot \gamma_{ss'}}{\gamma_{us} + \gamma_{ss'} + 1}$$

Here, $\gamma_{us}$ is the signal-to-noise ratio (SNR) of the user-to-satellite link, and $\gamma_{ss'}$ is the SNR of the inter-satellite link. This model captures the performance degradation from noise accumulation in the AF relaying process.

However, this increased robustness does not come without cost. The implementation of soft handover requires reliable and sufficiently strong ISLs, whose alignment and synchronization can be especially challenging at high frequencies (e.g., optical ISLs). The study also finds that the potential gains from more complex relaying techniques, such as decode-and-forward, may not always justify their increased hardware and processing requirements, as amplify-and-forward often offers a favorable trade-off between performance and complexity in practical settings.

Despite these challenges, the soft handover paradigm represents a compelling solution for future LEO constellations, especially as ISL technology matures. Its ability to improve link continuity, minimize error rates, and support stringent quality-of-service demands makes it an important area of ongoing research and a valuable complement to algorithmic handover strategies.

## 3.3 Multi-Attribute Decision Handover Schemes

Another line of research tackles the handover problem by considering multiple link attributes at once, rather than relying on a single criterion such as signal strength. Miao et al. Miao et al. (2019) propose a multi-attribute decision handover scheme for LEO mobile satellite networks, where the handover decision takes into account several factors simultaneously, specifically, the received signal strength, the remaining service time (i.e., how long the satellite will stay in view), and the number of idle channels available on each satellite.

In this approach, each user evaluates all candidate satellites using a weighted algorithm that balances these attributes. This comprehensive view helps reduce unnecessary han-

dovers, minimizes channel switching, and maintains stronger, more stable connections.

**Weighted Multi-Attribute Decision Score**

$$\text{Score}_{ij} = w_1 \cdot S_{ij} + w_2 \cdot T_{ij} + w_3 \cdot C_{ij}$$

Here, $S_{ij}$ denotes the signal strength, $T_{ij}$ the expected coverage duration, and $C_{ij}$ the number of idle channels for satellite $j$ from user $i$'s perspective, with $w_1$, $w_2$, and $w_3$ representing tunable positive weights.

Simulation results show that this scheme achieves fewer handovers and improved link quality compared to single-metric or traditional methods. The method is especially practical because it does not require complex optimization, yet it adapts well to the highly dynamic conditions of LEO constellations.

## 3.4 Auction-Based and Game-Theoretic Approaches

Distributed decision-making for handover is also addressed using auction-based and game-theoretic methods. Almathami et al. Jung et al. (2022) introduce an auction-based handover mechanism where each user terminal (UT) initiates an auction among nearby LEO satellites. Satellites evaluate their own suitability (based on factors like signal strength and service time) and submit bids. The user then selects the winner according to the Myerson auction principle, which ensures the process is trustworthy and discourages selfish behavior.

**Auction Utility Function**

$$U_{ij} = \alpha Q_{ij} - \beta L_{ij}$$

Where $Q_{ij}$ is the quality of link and $L_{ij}$ is the estimated load of satellite $j$, and $\alpha$, $\beta$ are positive parameters to balance user preference.

This distributed, utility-maximizing approach is designed to avoid the bottlenecks and scalability issues of centralized schemes and is particularly useful in large, rapidly changing constellations. Experiments show that this method balances network load and reduces unnecessary handover attempts, while still maintaining robust connectivity. By leveraging principles from economics, the strategy offers a fresh perspective on handover management, making it both scalable and practical for real-world deployment.

## 3.5 Reinforcement Learning-Based Handover

Given the unpredictable and dynamic nature of LEO satellite channels, several researchers have turned to machine learning to optimize handover strategies over time. Liu et al. Liu et al. (2024) propose a multi-agent deep reinforcement learning (DRL) approach, where each

user or agent independently learns the best time and target for handover by continuously interacting with the network environment.

Their method models the handover scenario using a three-state Markov process to capture changing channel conditions and leverages DRL to maximize long-term network performance, such as throughput and service continuity. Both centralized and distributed versions of the algorithm are explored: in the distributed case, each user relies only on local information, making the approach scalable to large constellations.

DRL Reward Function

$$r_t = \lambda_1 Q_{\text{link}}(t) - \lambda_2 C_{\text{handover}}(t)$$

Where $Q_{\text{link}}(t)$ captures the link quality and $C_{\text{handover}}(t)$ models the associated handover cost, with $\lambda_1$ and $\lambda_2$ representing positive policy trade-off weights.

Simulation results demonstrate that these intelligent agents can outperform traditional, static handover algorithms by quickly adapting to time-varying link conditions and balancing satellite loads. The main challenge, as noted in the study, is the computational complexity and the need for training data, but the results point toward a promising direction for future LEO network management.

## 3.6 Comparative Summary

The handover strategies presented in this chapter illustrate the broad spectrum of solutions developed to address the unique mobility challenges of LEO satellite networks. On one end are algorithmic scheduling approaches, such as graph-based maximum weight matching (MWM) with MIMO support, which focus on optimizing assignments for load balancing and resource efficiency. On the other end are physical, layer techniques like soft handover with inter-satellite link (ISL) relaying, which prioritize link robustness and seamless service continuity.

In addition, recent advances have introduced multi-attribute decision-making, auction-based game-theoretic schemes, and reinforcement learning-based algorithms, each offering a different balance between implementation complexity, adaptability, and network performance. Table 3.1 summarizes the main attributes of several major handover strategies discussed.

Table 3.1: Comparison of major handover strategies for LEO satellite networks

| **Strategy** | **Key Mechanism** | **Advantages** | **Limitations** |
|---|---|---|---|
| Maximum Weight Matching (MWM) with MIMO | Assignment as bipartite matching; weights based on link quality and MIMO channel | Load balancing; supports multi-user/multi-link; scalable | Requires frequent updates; assumes accurate channel state; possible link instability |
| Soft Handover with Inter-Satellite Link (ISL) Relaying | Dual connectivity; ISL relaying (AF/DF); overlapping coverage from serving and target satellites | Reduces block error rate and interruptions; seamless transitions | Requires strong ISL; added hardware complexity; sensitive to ISL misalignment |
| Multi-Attribute Decision Making | Weighted evaluation of signal strength, service time, and channel availability | Reduced handover frequency; improved stability; no complex optimization needed | Attribute weighting may require tuning; may not guarantee global optimum |
| Auction-Based / Game-Theoretic | Distributed auction among satellites; user selects winner based on utility | Trustworthy; scalable; avoids centralized bottlenecks | May require additional signaling; performance depends on auction rules |
| Reinforcement Learning-Based | Agents learn handover policy via deep RL, adapting to environment | Adaptive to dynamic conditions; potential for global performance gains | Requires training; computationally intensive; may need online learning |
| Hard Handover (Conventional) | Instant switch from serving to target satellite; single connectivity | Simplicity; minimal hardware | High risk of service interruption; packet loss during transition |

This comparison highlights the trade-offs between algorithmic complexity, adaptability, robustness, and performance across different handover design philosophies for LEO satellite networks.

In summary, while existing handover solutions have made substantial progress in coping with the rapid dynamics of LEO constellations, many approaches still rely on simplifying assumptions or incur significant implementation overhead. These limitations motivate the search for scalable, real-time optimization frameworks that can combine the strengths of both algorithmic scheduling and signal-level robustness. The next chapter presents our proposed framework, which aims to bridge this gap and deliver reliable, efficient handover management for future satellite internet systems.

# Chapter 4

# Proposed Optimization Framework

This chapter introduces the optimization framework developed to address the user–satellite assignment problem in Low Earth Orbit (LEO) networks. The goal is to design an assignment mechanism that maximizes overall link quality while explicitly accounting for the cost of frequent handovers, satellite capacity limitations, and the possibility of dual connectivity.

A central feature of this formulation is the inclusion of a *handover penalty*, which captures the negative impact of excessive satellite switching on service continuity and signaling overhead. By embedding this penalty directly into the optimization objective, the framework strikes a balance between maximizing throughput and minimizing disruptions due to handovers.

**Assignment strategies.** Within this framework, we evaluate two distinct assignment strategies:

- **Simultaneous Handover Assignment (SiHA)**: both primary and secondary satellite connections for a user are reassigned *at the same time step*, ensuring coordinated handovers but potentially exposing the user to simultaneous disruptions.
- **Staggered Handover Assignment (StHA)**: users also maintain dual connectivity, but handovers for the primary and secondary links are deliberately *offset in time*, reducing the chance of both links failing simultaneously at the cost of more complex scheduling.

These two strategies are studied under two assumptions of the channel state information (CSI): (i) *ideal mode*, where the algorithm has perfect 10 s look-ahead knowledge of channel conditions, and (ii) *forecast mode*, where only statistical channel averages are available based on elevation. The comparative evaluation of SiHA and StHA under these two CSI regimes forms the core of this thesis.

In the following sections, we formally define the system model, introduce the optimization variables, formulate the objective function with the handover penalty, and describe the algorithmic approach adopted to solve the problem.

## 4.1 Optimization Model

We consider a set of users $\mathcal{U} = \{1, 2, \ldots, U\}$ and a set of visible satellites $\mathcal{S} = \{1, 2, \ldots, S\}$ at a given time slot $t$. Each user $u \in \mathcal{U}$ may be connected to one or more satellites, subject to system constraints.

### 4.1.1 Assignment Variables

We define the binary assignment variable:

$$x_{u,s}(t) = \begin{cases} 1, & \text{if user } u \text{ is connected to satellite } s \text{ at time } t, \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

### 4.1.2 Objective Function with Handover Penalty

The instantaneous utility of connecting user $u$ to satellite $s$ at time $t$ is modeled by its achievable rate $R_{u,s}(t)$, which depends on the link SNR:

$$R_{u,s}(t) = B \log_2 \left(1 + \mathrm{SNR}_{u,s}(t)\right), \tag{4.2}$$

where $B$ is the allocated bandwidth.

To discourage excessive switching between satellites, we introduce a *handover penalty* term. Let $x_{u,s}(t-1)$ be the assignment at the previous time slot. Then the handover indicator is:

$$h_{u,s}(t) = \max\{0,\, x_{u,s}(t) - x_{u,s}(t-1)\}, \tag{4.3}$$

which equals 1 if user $u$ initiates a new connection with satellite $s$ at time $t$, and 0 otherwise.

The global optimization problem is therefore:

$$\max_{x_{u,s}(t)} \quad \sum_{u\in\mathcal{U}} \sum_{s\in\mathcal{S}} \Big( R_{u,s}(t) \cdot x_{u,s}(t) \Big) - \lambda \sum_{u\in\mathcal{U}} \sum_{s\in\mathcal{S}} h_{u,s}(t), \tag{4.4}$$

where $\lambda > 0$ is a tunable weight that balances throughput maximization against the cost of handovers.

### 4.1.3 Constraints

The optimization is subject to the following constraints:

1. **Satellite capacity:**

$$\sum_{u\in\mathcal{U}} x_{u,s}(t) \le G, \quad \forall s \in \mathcal{S}, \tag{4.5}$$

where $G$ is the maximum number of users that each satellite can simultaneously serve.

2. **User connectivity:**
$$\sum_{s \in \mathcal{S}} x_{u,s}(t) \leq K, \quad \forall u \in \mathcal{U}, \tag{4.6}$$
where $K = 2$ allows for dual connectivity.

3. **Binary assignments:**
$$x_{u,s}(t) \in \{0, 1\}, \quad \forall u \in \mathcal{U},\ s \in \mathcal{S}. \tag{4.7}$$

This formulation yields a combinatorial optimization problem that can be solved using assignment algorithms such as the Hungarian (Munkres) method, suitably adapted for capacity and handover penalty constraints.

## 4.2 Satellite Constellation and User Distribution

The satellite constellation considered in this thesis is modeled after recent LEO mega-constellations (e.g., Starlink SpaceX (2020)), which deploy hundreds or even thousands of satellites in coordinated orbital planes to provide near-global coverage. In line with standard mission analysis practices Wertz & Larson (1999), satellites are distributed across $N_{\text{orb}}$ orbital planes, each at a nominal altitude $h_0$ and inclination $i_K$.

The user population is composed of $N_{\text{usr}}$ terminals randomly distributed within the service region. The service area is modeled as a *spherical rectangle* on the Earth's surface, delimited by latitude bounds $[\varphi_{\min}, \varphi_{\max}]$ and longitude bounds $[\lambda_{\min}, \lambda_{\max}]$. Each user is identified by a pair of geographic coordinates $(\varphi_u, \lambda_u)$, sampled uniformly within these limits. This modeling choice captures both the diversity and the unpredictability of real-world user locations while ensuring reproducibility of the simulation setup.

## 4.3 Satellite Visibility and Geometry

At any given time slot, the set of satellites visible to a particular user is determined by computing the elevation angle and slant distance between each satellite and the user's position. Only satellites whose elevation exceeds a minimum operational threshold (typically 20° or higher) are considered candidates for assignment. This is consistent with established practice in the satellite communications literature Wertz & Larson (1999); SpaceX (2020).

## 4.4 Radio Channel Modeling

Reliable user connectivity in a LEO system depends not only on satellite geometry, but also on the vagaries of the propagation channel. To ensure realism, we employ the ITU-R P.681-10 recommendation International Telecommunication Union (2017) to model both the large-scale (shadowing and path loss) and small-scale (multipath fading) effects experienced by the satellite-to-ground links. This widely-adopted channel model enables simulation of link quality under a variety of environmental and operational conditions, and it has been used in multiple recent works addressing LEO handover and assignment optimization Juan et al. (2022); Ben Salem et al. (2025).

The instantaneous received SNR for a user $u$ from satellite $m$ at time $t$ is therefore calculated as:

$$\mathrm{SNR}_{u,m}(t) = \frac{P_s G_{\mathrm{SAT}} G_{\mathrm{UE}} L(d_{m,u}(t), f_c) |h_{u,m}(t)|^2}{N_0}$$

where $P_s$ is the satellite transmit power, $G_{\mathrm{SAT}}$ and $G_{\mathrm{UE}}$ are antenna gains, $L(\cdot)$ represents path loss (including atmospheric and shadowing losses), $|h_{u,m}(t)|^2$ models shadowing and fading, and $N_0$ denotes the noise power Ben Salem et al. (2025); Liu et al. (2024).

## 4.5 Capacity and Assignment Constraints

In practice, each satellite can only support a finite number of concurrent user connections, limited by its onboard processing power, available bandwidth, and antenna resources SpaceX (2020); Juan et al. (2022). This limitation is captured in Eq. 4.5, where $G$ denotes the *maximum number of concurrent users that can be served by a single satellite*. In other words, at any given time a satellite cannot be assigned to more than $G$ users simultaneously in the optimization model.

On the user side, terminals are allowed up to two simultaneous satellite connections (a dual-assignment approach), as expressed in Eq. 4.6 with $K = 2$. This strategy is increasingly recognized as an effective way to mitigate the high handover frequency inherent to fast-moving LEO constellations Ben Salem et al. (2025); Abdu et al. (2023).

# Chapter 5

# Performance Assessment of the Proposed Assignment Algorithm

The ability of a satellite network to maintain reliable and high-quality user connectivity depends on much more than clever algorithms; it is shaped by the unpredictable realities of orbital motion, channel fading, and resource contention. In this chapter, we present a detailed simulation-based evaluation of the proposed *simultaneous handover user-satellite assignment framework*. The aim is not only to quantify its performance, but also to demonstrate how it manages the unique challenges posed by LEO mega-constellations.

## 5.1 Simulation Environment and Methodology

The network consists of 72 orbital planes, each containing 22 satellites, yielding a total of $72 \times 22 = 1584$ satellites. All satellites are deployed at an altitude of 550 km and an inclination of 53°. This configuration ensures a dense and persistent satellite presence over the user region, while still capturing the dynamic topology characteristic of LEO constellations.

Ground users, fifty in total, are randomly distributed within a $5° \times 5°$ region in mid-latitudes, each remaining stationary during a simulation run. Although static, this setup offers a challenging testbed, as each user's view of the sky is constantly changing due to satellite movement.

Channel modeling is grounded in the ITU-R P.681-10 recommendation International Telecommunication Union (2017), a standard that captures both the average signal attenuation caused by atmospheric and terrain shadowing, and the rapid multipath fluctuations that characterize real-world propagation. The channel coefficients for each user-satellite pair are updated at every simulation step, with a sampling time of 10 s, based on the instantaneous elevation angle and a suburban fading profile.

All relevant simulation parameters are summarized in Table 5.1. These values, including carrier frequency, bandwidth, antenna gains, and the suburban propagation environment,

are chosen to reflect current industry practice SpaceX (2020); Wertz & Larson (1999).

Table 5.1: Simulation parameters for the channel and system model.

| **Parameter** | **Value** |
|---|---|
| Carrier frequency $f_c$ | 2 GHz |
| System bandwidth | 5 MHz |
| Satellite antenna gain $G_{\mathrm{SAT}}$ | 50 dBi |
| User antenna gain $G_{\mathrm{UE}}$ | 0 dBi |
| Channel model | ITU-R P.681-10 (LMS) |
| Propagation environment | Suburban |
| Sampling interval (time step) | 10 s |

All relevant simulation parameters are summarized in Table 5.1. In addition, Table 5.2 lists the constellation topology and assignment-related parameters.

Table 5.2: Topology, region, and constraint parameters used in simulation.

| **Parameter** | **Value** |
|---|---|
| Orbital planes $N_{\mathrm{orb}}$ | 72 |
| Satellites per plane | 22 |
| Total satellites $N_{\mathrm{sat}}$ | 1 584 |
| Altitude $h_0$ | 550 km |
| Inclination $i_K$ | 53° |
| Users $N_{\mathrm{usr}}$ | 50 |
| Service area bounds | $[\varphi_{\min}, \varphi_{\max}] \times [\lambda_{\min}, \lambda_{\max}]$ |
| Window size | $5° \times 5°$ (mid-latitudes) |
| Elevation threshold $\theta_{\min}$ | 25° |
| Per-satellite capacity $G$ (Eq. 4.5) | 4 users |
| Max user links $K$ (Eq. 4.6) | 2 |
| Time horizon | 361 slots @ 10 s (1 hour) |
| Transmit power $P_s$ | varied; baseline 1 W |

The simulation operates on a time-slotted basis, with 361 steps at 10-second intervals, spanning one hour of system evolution. At each slot, the system computes which satellites are visible (requiring at least 25° elevation), updates the channel state, and applies the assignment algorithm. Each satellite can serve up to four users at a time, enforcing a practical resource constraint.

It is important to note that the radio channel varies on a much faster time scale than the 10 s sampling interval adopted here. For instance, at a pedestrian speed of 5 km/h, a user moves nearly 14 m in 10 s, which is sufficient to alter the local scattering environment and thereby change both shadowing and fading effects. The simulation therefore approximates these rapid variations by updating the channel coefficients once every slot, using a new realization consistent with the underlying fading distribution.

Recognizing that perfect channel knowledge is rarely available in reality, two assignment modes are evaluated:

- *Ideal mode*: the algorithm is assumed to know not only the instantaneous channel coefficients but also their exact values in the next time slot (a 10 s look-ahead). This represents an upper performance bound, where scheduling decisions are made with perfect foresight.
- *Forecast mode*: assignments are made based on average channel values per elevation bin, while actual performance is evaluated using the instantaneous realizations. This captures the more realistic case of imperfect prediction.

This distinction is essential for gauging the robustness of the assignment framework under channel uncertainty, a recurring theme in recent LEO literature Liu et al. (2024); Juan et al. (2022).

## 5.2 Performance Evaluation Criteria

Assessing a handover framework requires a multi-dimensional view of performance. Beyond raw throughput, we must consider fairness, continuity, and resilience. Here, several metrics are emphasized:

- *Average user throughput and SNR*: How much useful data (and signal quality) does each user receive on average?
- *Distributional fairness*: How evenly is performance spread among users, avoiding the pitfall where a few users suffer disproportionately?
- *Temporal robustness*: Can the system maintain performance as satellites come in and out of view and as the radio environment fluctuates?
- *Impact of channel uncertainty*: Does the framework deliver under both perfect and imperfect channel state information?
- *Redundancy benefit*: How much does maintaining two simultaneous satellite links per user (dual-assignment handover) improve performance and reliability compared to a conventional single-link approach?

Throughout the analysis, these questions guide the interpretation of results, always connecting to the practical goals of the system and the lessons highlighted in the foundational works Ben Salem et al. (2025).

## 5.3 Results for the Simultaneous Handover Assignment (SiHA) Algorithm

This section evaluates the performance of the *Simultaneous Handover Assignment (SiHA)* algorithm, where each user is connected to two satellites at each time step, as outlined

in Chapter 4.5. The results are presented for both the *ideal mode* (perfect CSI) and the *forecast mode* (predicted CSI).

### 5.3.1 Impact of Transmit Power on Aggregate Performance

Figures 5.1 and 5.2 illustrate average user throughput and average user SNR, respectively, as functions of satellite transmit power $P_s$. The curves appear closely matched, showing consistent, almost linear growth in performance with increasing $P_s$, for both ideal and forecast modes.

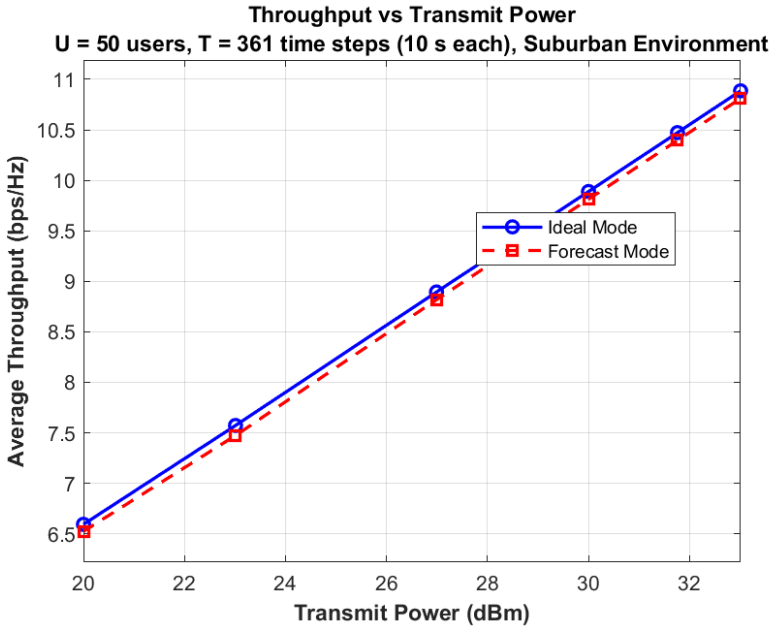


Figure 5.1: Average user throughput versus transmit power in ideal and forecast modes.

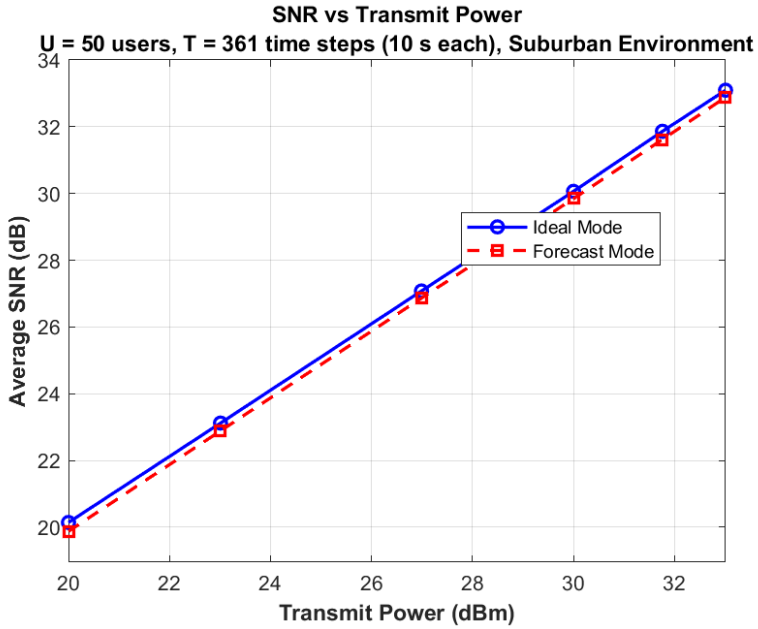


Figure 5.2: Average user SNR versus transmit power in ideal and forecast modes.

At first glance, these results suggest that the forecast-based strategy nearly matches the performance of an ideal strategy. However, such aggregate metrics can mask important temporal dynamics and variations between users. Therefore, deeper insights require a time-domain and per-user analysis, as presented next.

### 5.3.2 System Behavior Over Time

To understand the temporal evolution of network performance, Figures 5.3 and 5.4 plot the total system throughput and the total SNR at each time step. Unlike the smoothed averages, these time series plots expose clear and persistent performance gaps between the ideal and forecast modes.

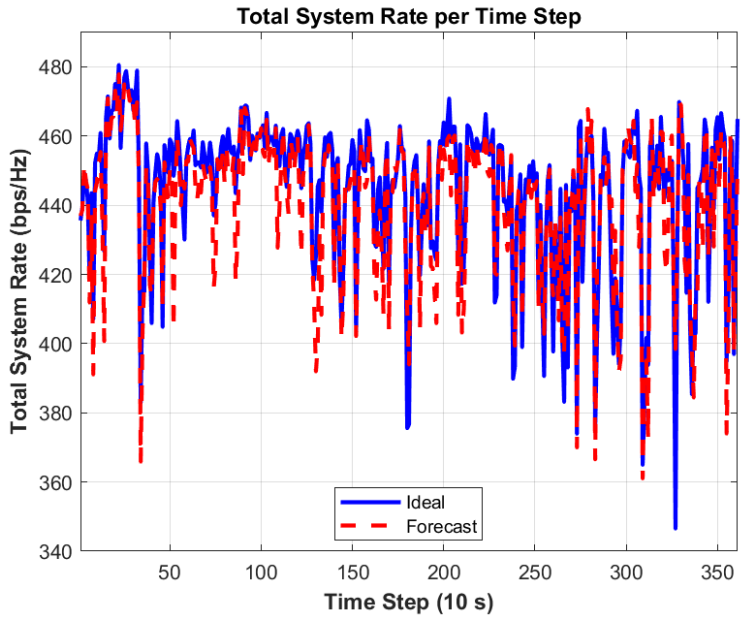


Figure 5.3: Total system throughput versus time under ideal and forecast CSI for a fixed transmit power of $P_s = 1$ W.

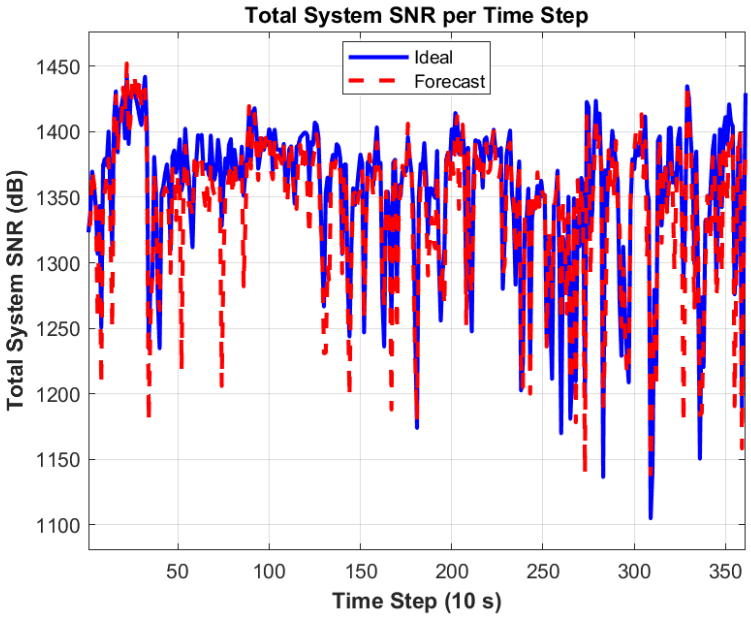


Figure 5.4: Total system SNR versus time under ideal and forecast CSI for a fixed transmit power of $P_s = 1$ W.

These plots highlight non-negligible fluctuations and consistent underperformance in forecast mode. Despite similar average values, forecast-based decisions accumulate errors over time, leading to higher performance variance and more frequent degradations.

### 5.3.3 User-Level Service Continuity

Figures 5.5 and 5.6 show the rate and evolution of the SNR with time for a representative user. Although the forecast mode tracks the ideal behavior reasonably well, it suffers from sharper and more frequent dips, especially during handovers or deep fades.

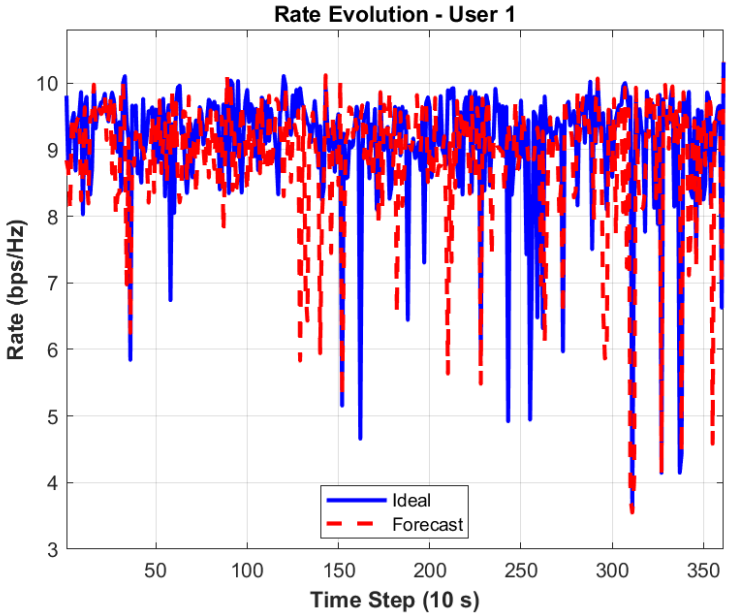


Figure 5.5: Rate evolution over time for a representative user under a fixed transmit power of $P_s = 1$ W.

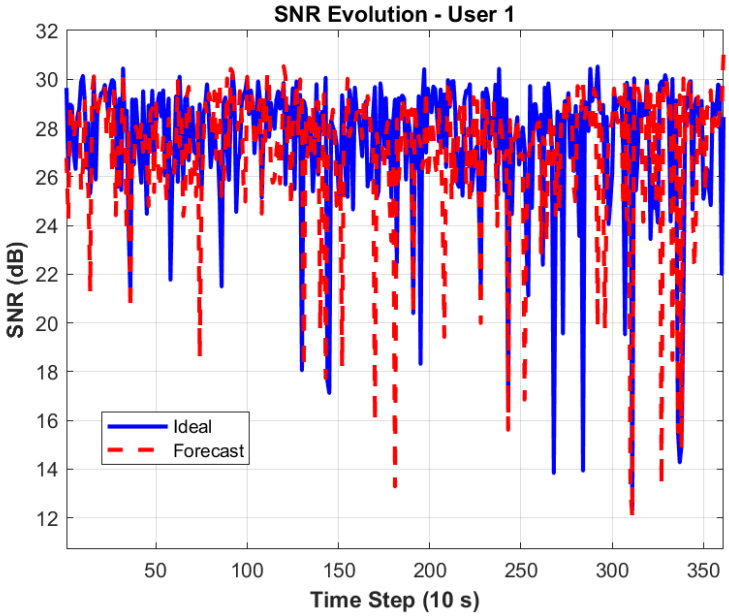


Figure 5.6: SNR evolution over time for the same user under a fixed transmit power of $P_s = 1$ W.

These dips align with handover instances or simultaneous degradation of both assigned links. However, the dual link strategy ensures resilience, as one link often compensates when the other deteriorates, validating the concept of "soft" handover robustness, consistent with the results in Ben Salem et al. (2025).

### 5.3.4 Contribution of Dual Assignments

Figures 5.7 and 5.8 decompose the total system rate and the SNR into contributions from the primary and secondary assignments. In particular, the second (redundant) link is not passive; it actively increases system capacity and helps bridge performance dips.

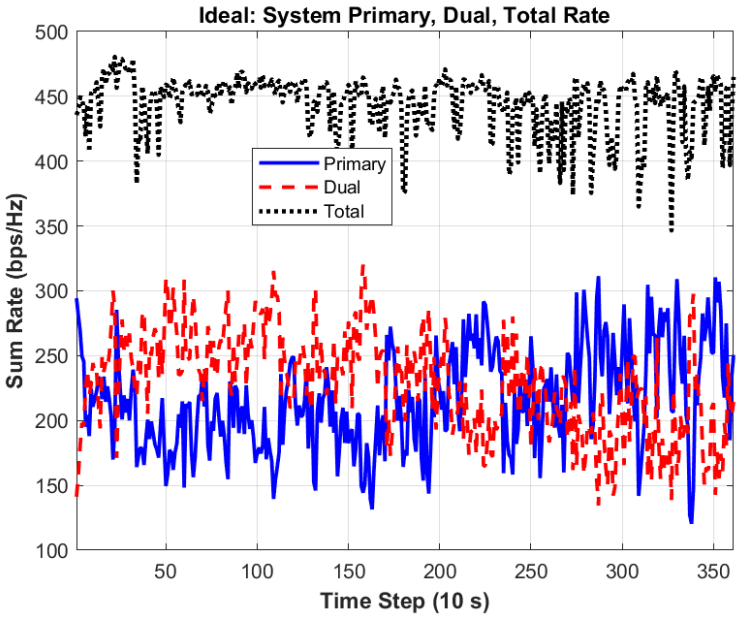


Figure 5.7: Decomposition of system sum rate in ideal mode for a fixed transmit power of $P_s = 1$ W.

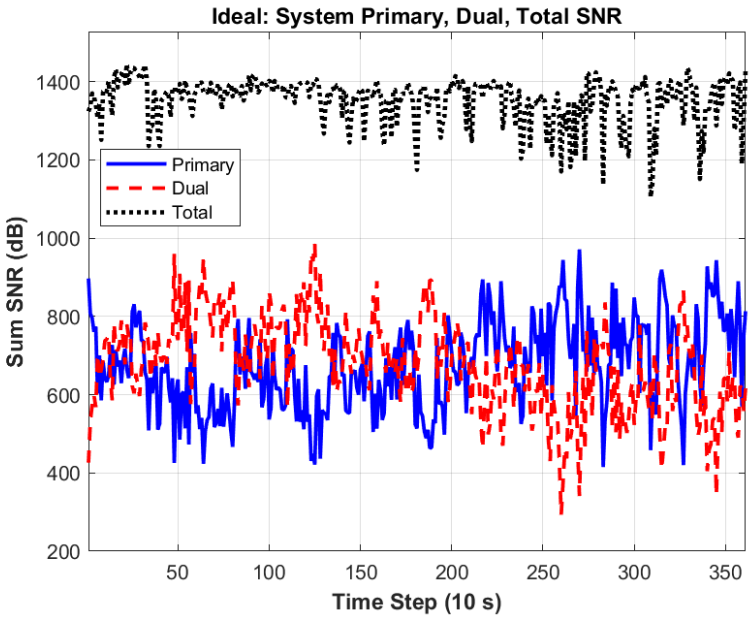


Figure 5.8: Decomposition of system sum SNR in ideal mode for a fixed transmit power of $P_s = 1$ W.

This confirms that the dual-assignment structure is not only a fallback but a primary contributor to enhanced performance, especially under non-stationary conditions like satellite mobility and dynamic fading.

### 5.3.5 Fairness Across Users

Lastly, Figures 5.9 and 5.10 present the cumulative distribution functions (CDFs) of the average user rate and the SNR. The distributions are narrow and steep, indicating strong fairness and tight clustering across the user population, with no significant outliers.

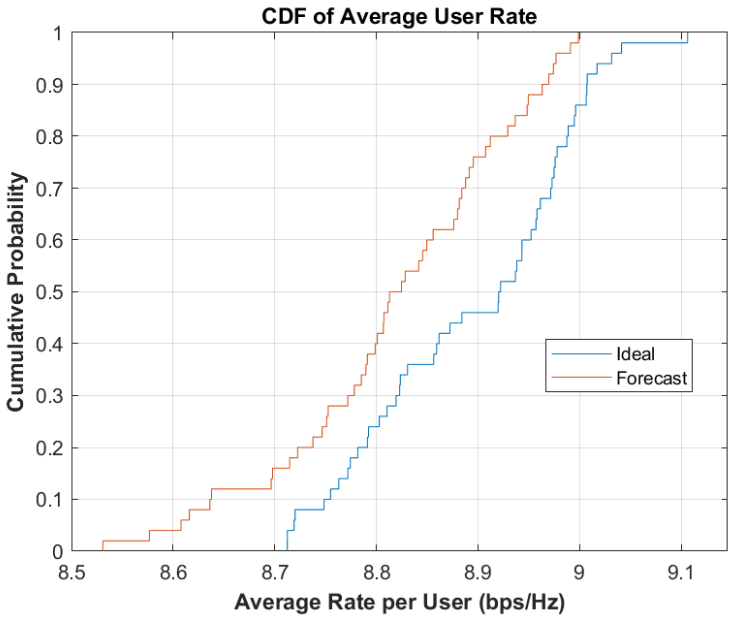


Figure 5.9: CDF of average user rate under a fixed transmit power of $P_s = 1$ W.

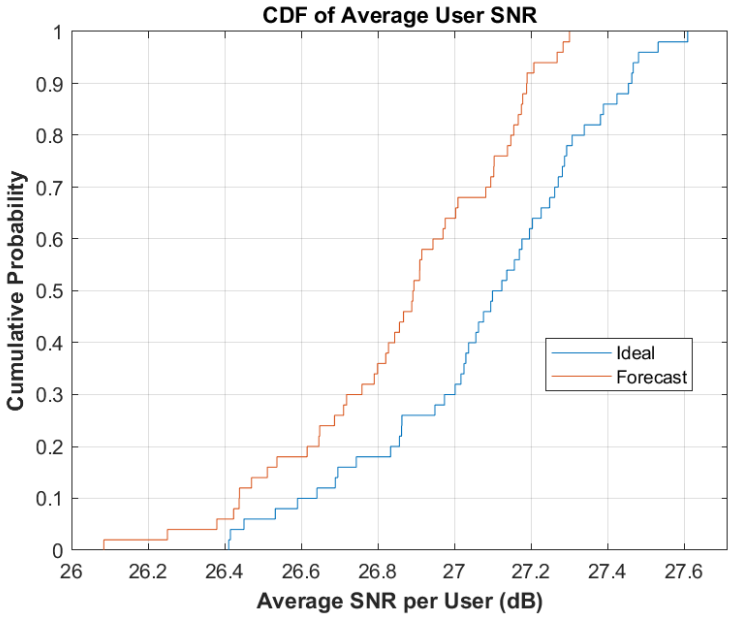


Figure 5.10: CDF of average user SNR under a fixed transmit power of $P_s = 1$ W.

Compared to typical greedy schemes, the simultaneous handover mechanism ensures nearly uniform performance levels. The forecast mode shows a slightly wider spread but still avoids significant service inequality, which is essential for large-scale broadband LEO deployments.

## 5.4 Results for Algorithm 2: Staggered Handover Assignment (StHA)

Having established the strengths and subtleties of simultaneous handover, we now turn to a different approach: the *staggered assignment* algorithm. In this method, each user is still granted two satellite connections at any time, but the assignments are deliberately staggered, meaning that handovers for the first and second links do not occur simultaneously. The idea is to reduce the risk of both connections being disrupted at once, potentially smoothing user experience.

### 5.4.1 Scaling of Throughput and SNR with Transmit Power

Figures 5.11 and 5.12 show how the average user throughput and SNR scale with increasing satellite transmit power in the staggered assignment scenario, for both the ideal and forecast modes.

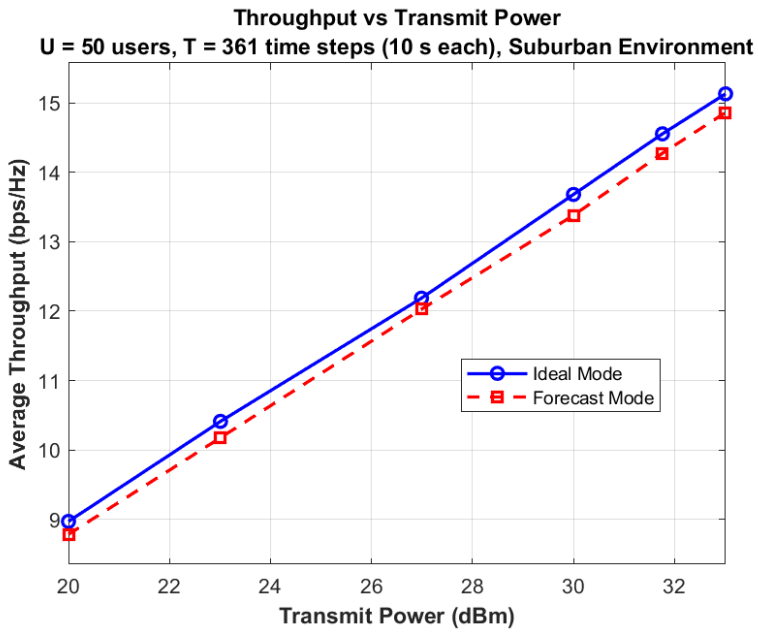


Figure 5.11: Average user throughput versus transmit power for ideal and forecast assignment modes (staggered). Results are shown for a fixed transmit power of $P_s = 1$ W.

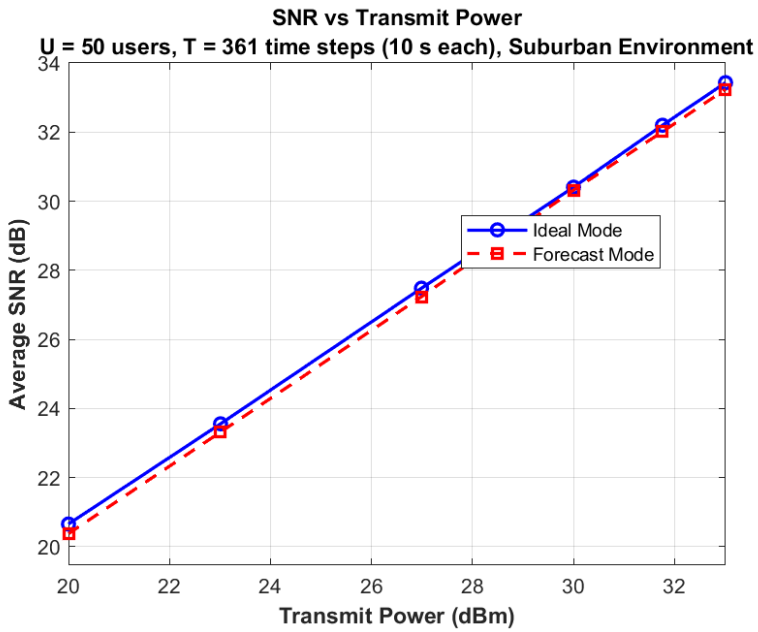


Figure 5.12: Average user SNR versus transmit power for ideal and forecast modes (staggered). Results are shown for a fixed transmit power of $P_s = 1$ W.

At first glance, these results look almost reassuring: as transmit power rises, the average throughput and SNR both improve steadily, and the gap between ideal and forecast modes remains modest. For a reader focused on mean values, it might seem as though the network is performing admirably regardless of the underlying assignment method.

But this is exactly where the danger lies. By averaging over all users and all time steps, the system can hide rare but severe disruptions, outages or deep fades that, while infrequent, can completely break the experience for some users. The apparent smoothness in these curves belies significant, and sometimes dramatic, variation beneath the surface.

### 5.4.2 System Performance Over Time

To unmask the real impact of staggered assignment, we must look at how total system performance fluctuates over time, as depicted in Figures 5.13 and 5.14.

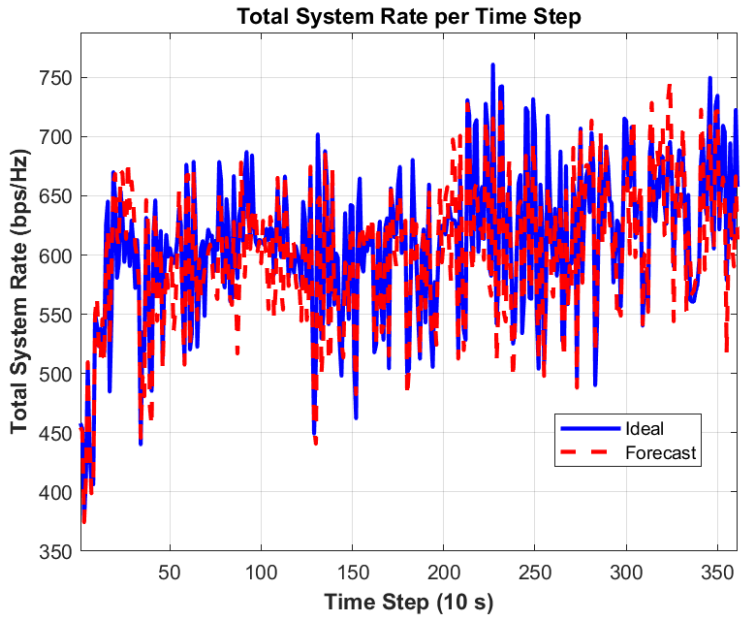


Figure 5.13: Total system throughput per time step in ideal and forecast modes (staggered), for a fixed transmit power of $P_s = 1$ W.

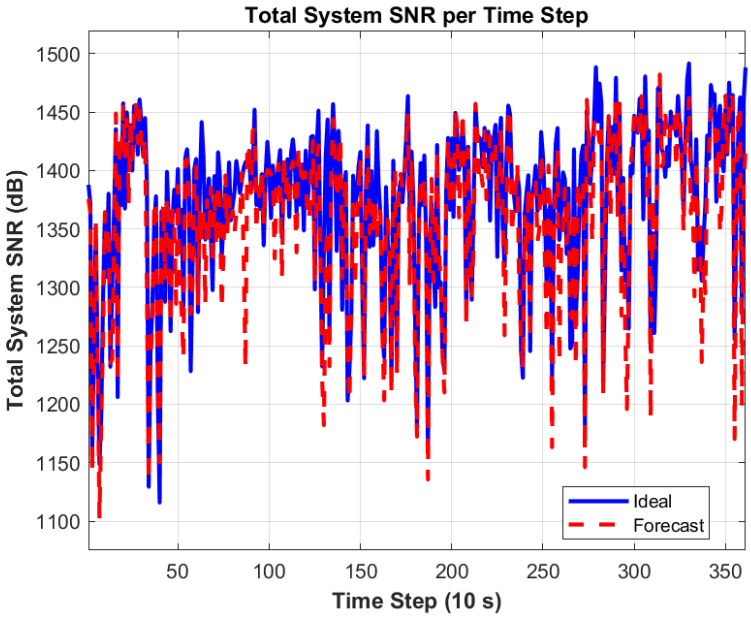


Figure 5.14: Total system SNR per time step in ideal and forecast modes (staggered), for a fixed transmit power of $P_s = 1$ W.

Here, the illusion of stability starts to break down. Unlike in the simultaneous handover case, there are periods, sometimes short, sometimes persistent, where the system experiences pronounced dips. These fluctuations hint at moments when many users are forced into less favorable links, or when staggered handovers fail to shield users from simultaneous signal degradation. The time-series view reveals a much "spikier" network life, especially in the forecast mode.

### 5.4.3 User-Level Continuity and Vulnerability

The real story emerges at the individual user level. Figures 5.15 and 5.16 illustrate the rate and SNR for a representative user over the simulation window.

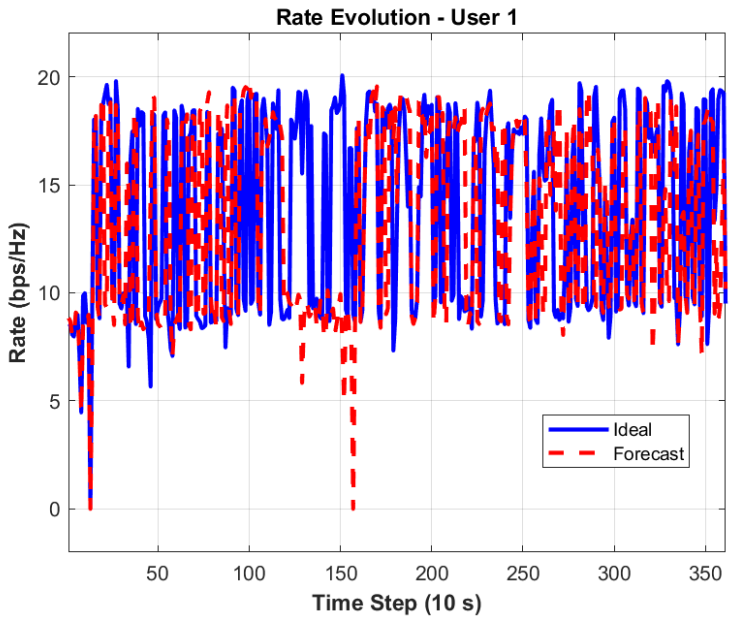


Figure 5.15: Rate evolution for a representative user in staggered assignment, for a fixed transmit power of $P_s = 1$ W.

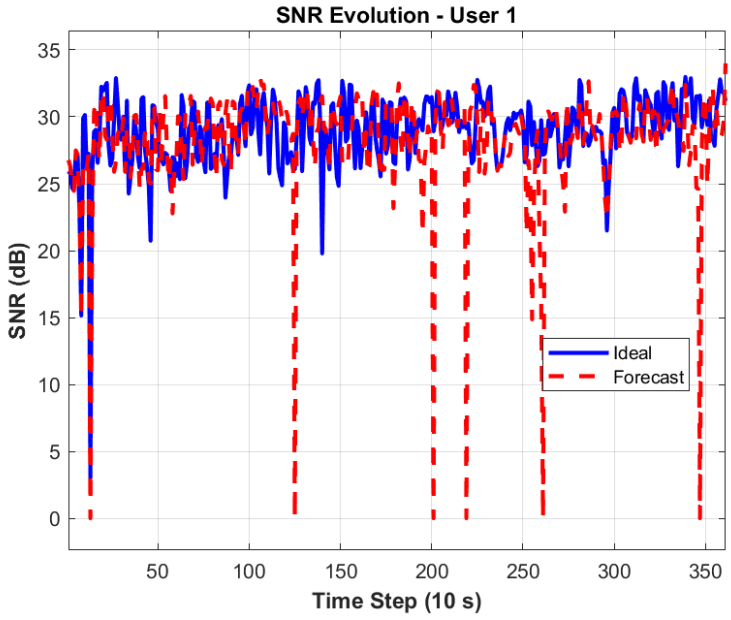


Figure 5.16: SNR evolution for the same user in staggered assignment, for a fixed transmit power of $P_s = 1$ W.

It is now impossible to ignore the story that the average concealed: there are long stretches

where this user's performance is not only unstable but can drop to alarmingly low levels, especially during certain handover transitions or periods of weak satellite visibility. The handover is less "soft," and users can experience outages or degraded quality for tens of seconds at a time, enough to be disruptive for latency, sensitive applications. This is the price of complexity: what looks good on average can feel very different for real people using the network.

### 5.4.4 Contribution of First and Second Assignments

Figures 5.17 and 5.18 break down the contributions from the first and second links, as well as their combined effect, in the ideal mode.

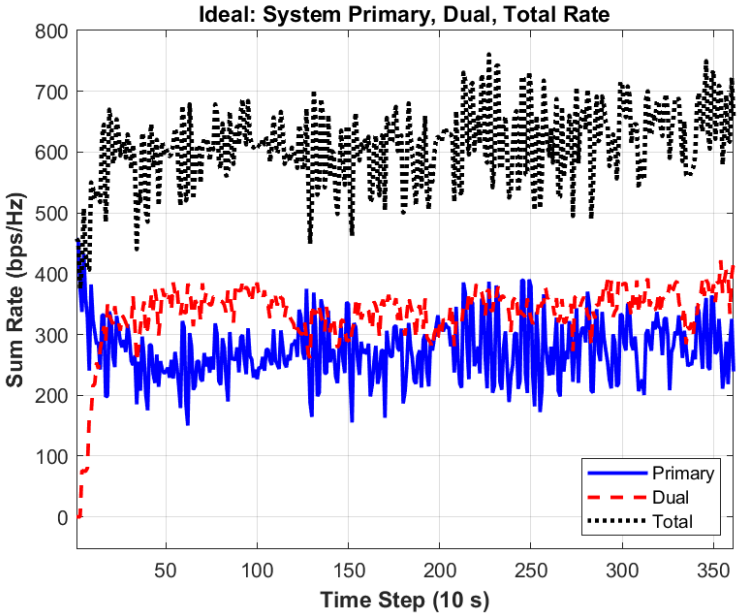


Figure 5.17: System sum rate over time in the ideal mode, separated into first, second, and total assignments (staggered), for a fixed transmit power of $P_s = 1$ W.

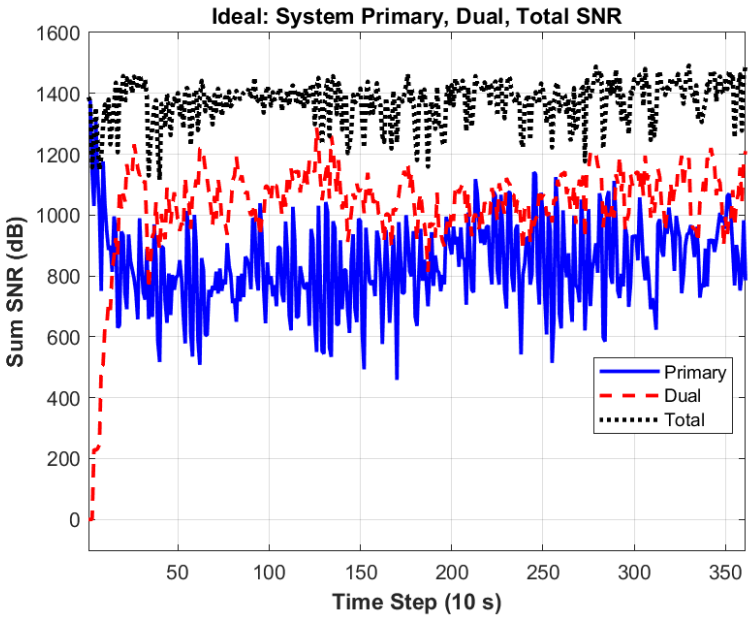


Figure 5.18: System sum SNR in the ideal mode, with both assignments shown separately and combined (staggered), for a fixed transmit power of $P_s = 1$ W.

We observe here a system that depends on both assignments, similar to the simultaneous case, where both links contribute consistently to the overall performance. The key difference, is that the secondary-link configuration exhibits fewer fluctuations, with reduced dips and jumps. In contrast, the primary link alone experiences more pronounced variations, resulting in greater instability.

### 5.4.5 Fairness and Distributional Impact

Finally, the CDFs in Figures 5.19 and 5.20 expose the cost in fairness. Although most users still enjoy decent average rates and SNRs, the distribution is wider and the left tail (the 'unlucky' users and moments) has shifted downward. In other words, more users are forced into lower quality experiences, at least part of the time.

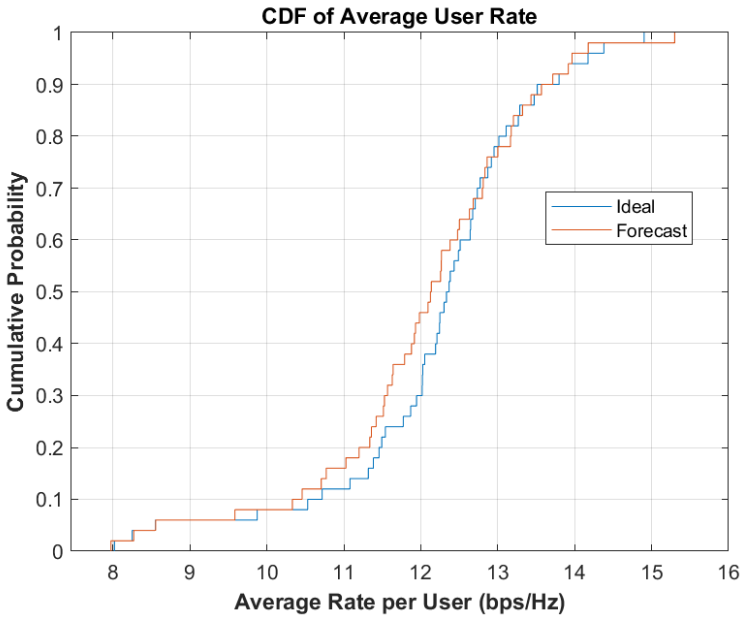


Figure 5.19: CDF of average user rate across the user population (staggered), for a fixed transmit power of $P_s = 1$ W.

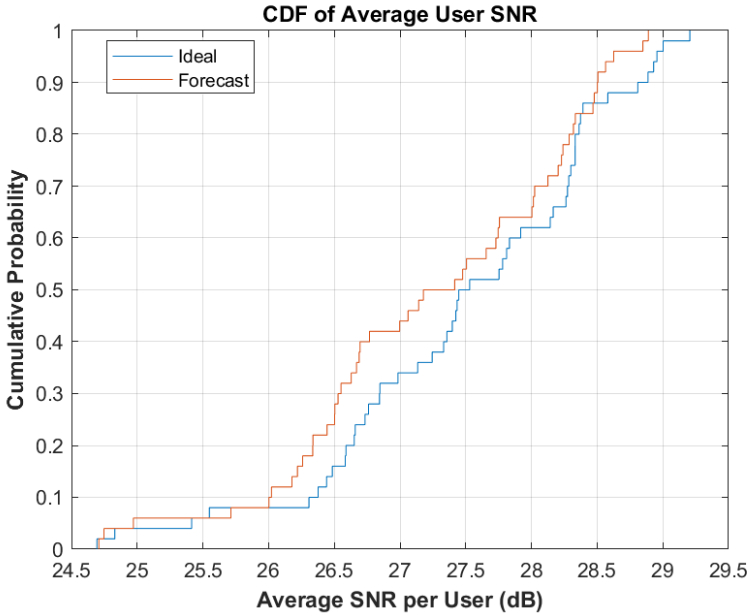


Figure 5.20: CDF of average user SNR across the user population (staggered), for a fixed transmit power of $P_s = 1$ W.

This demonstrates why "average" metrics can be dangerously misleading for systems where reliability and continuity are key. The staggered assignment, while theoretically robust, introduces vulnerabilities that become visible only through detailed simulation and per-user analysis.

In summary, the staggered assignment strategy provides an instructive contrast: it shows that what works on paper, or on average, does not always translate to a smooth or fair

experience in real networks. True robustness in LEO handover demands that we look beyond the mean and focus on the outliers, the worst cases, and the hidden valleys that users may fall into, even if only briefly.

## 5.5 Comparison of Algorithm Performance

A direct comparison between Algorithm 1 (Simultaneous Handover Assignment) and Algorithm 2 (Staggered Handover Assignment) across the evaluated parameters reveals a clear divergence in both aggregate and user-level performance characteristics. In terms of **average throughput** and **average SNR**, both algorithms achieve closely aligned results under ideal channel state information (CSI) and, importantly, the forecast-based CSI mode. Figures 5.1, 5.2, 5.11, and 5.12 collectively show that as satellite transmit power increases, mean system throughput and SNR rise almost linearly for both strategies, with forecast curves closely shadowing ideal ones. This indicates that, in a purely aggregate sense, predictive decision making is effective for both handover schedules, with minimal loss relative to perfect knowledge. However, these similarities in average metrics mask substantial differences in temporal stability and distributional fairness, as exposed by subsequent analyzes. Although Algorithm 1's forecast mode exhibits a small but persistent shortfall relative to ideal, these deviations are minor and stable compared to the sharp fluctuations seen in Algorithm 2.

When we examine the behavior of the time domain , the differences become more pronounced. In the simultaneous assignment case (Figures 5.3 and 5.4), the forecast mode operation exhibits only modest deviations from the ideal curve, with performance dips generally contained recovered quickly. This reflects the inherent resilience of maintaining two concurrent satellite connections, ensuring that degradations on one link are often compensated by the other. In contrast, the time series of the staggered assignment (Figures 5.13 and 5.14) reveals deeper and more frequent fluctuations, particularly in forecast mode, where the 'realistic stability' seen in the averages breaks down. In several observed intervals, the staggered schedule fails to prevent clusters of users from being simultaneously on suboptimal links, producing more dramatic drops in total system throughput and SNR. This suggests that the staggered approach, while conceptually aimed at smoothing network transitions, can in practice amplify transient instabilities when prediction errors or unfavorable link conditions occur in sequence.

The divergence is even more evident at the **individual user level**. For a representative user in the simultaneous case (Figures 5.5 and 5.6), the forecast mode operation closely follows the ideal baseline, with occasional sharper dips during handovers or deep fades, but with both links rarely degrading simultaneously for extended periods. This confirms that the dual link structure here is not merely a backup but an active contributor to sustained performance, consistent with established 'soft handover' robustness principles. In contrast, the representative user results of the staggered mode (Figures 5.15 and 5.16) show long periods of unstable or low performance, in some cases lasting tens of seconds, particularly when the offset handover timing leaves the user momentarily dependent on a single weak link or when sequential link degradation occurs. This reduction in continuity undermines one of the primary goals of dual-assignment: seamless service during satellite

transitions.

Fairness analysis, using cumulative distribution functions (CDFs) of average per-user throughput and SNR (Figures 5.9, 5.10 versus Figures 5.19 and 5.20), further highlights the contrast. Simultaneous assignment produces a narrow, symmetric distribution, with nearly all users clustered around similar performance levels. This indicates a high degree of fairness, avoiding the large disparities common in greedy or non-optimized schemes. However, the staggered assignment yields a visibly wider distribution, with its left tail - representing the worst-served users - shifted downward. Although many users still experience acceptable service, a larger minority suffer significantly reduced average throughput and SNR, reflecting the vulnerabilities identified in the time series and user-level analyses. In particular, these fairness degradations are not apparent from average performance curves alone, underscoring the importance of detailed distributional metrics in assessing network reliability.

In summary, while both algorithms demonstrate strong average performance and a close match between forecast and ideal CSI in aggregate metrics, simultaneous assignment consistently delivers superior stability, fairness, and worst-case user experience. The staggered assignment, despite its theoretical appeal for smoothing transitions, introduces specific vulnerabilities, particularly in maintaining continuity for individual users and in preventing synchronized degradations, that become visible only through fine-grained temporal and distributional analysis. These findings suggest that, for scenarios where user experience consistency and fairness are paramount, the simultaneous dual-assignment strategy offers a more robust and reliable solution.

## 5.6 Impact of Propagation Environment on System Performance

Up to this point, the evaluation of the system has been framed primarily in terms of transmit power, temporal dynamics, and user-level continuity. While these dimensions are critical, they only tell part of the story. A satellite system does not operate in a vacuum, and the wireless channel itself is profoundly shaped by the surrounding environment. The difference between a receiver located in an open rural plain, another in a suburban town, and a third deep in an urban canyon can be as dramatic as the difference between a clear sky and a storm. Multi-path fading, shadowing from buildings and vegetation, and intermittent line-of-sight conditions all conspire to alter the effective quality of the link. For this reason, a complete performance assessment must go beyond power scaling and explore how the system behaves under diverse propagation conditions.

The methodology adopted here follows the ITU-R P.681-10 land-mobile satellite (LMS) channel model International Telecommunication Union (2017), which prescribes distinct fading profiles for typical environments such as urban, suburban, village, and rural wooded. By holding the constellation geometry and user locations fixed, and varying only the fading environment, we are able to isolate the algorithm's sensitivity to channel variability. This approach mirrors real-world deployment scenarios, where the same constellation must serve a heterogeneous mix of users scattered across cities, towns, and sparsely populated

regions. Understanding whether an algorithm that excels in one environment falters in another is therefore central to establishing its robustness.

In what follows, the analysis begins with the **suburban environment**, chosen as a natural starting point. Suburban conditions represent a middle ground; more demanding than the near ideal open and rural cases, but less hostile than dense urban terrain. It is also the default profile often assumed in system-level studies of LEO broadband constellations, making it a useful baseline for comparison. The results obtained in this setting are then contrasted with those in urban, rural, and open environments, allowing us to build a layered understanding of how the system navigates the spectrum of propagation challenges.

### Suburban Environment

*Note on reproducibility.* The results shown for the suburban environment are generated from a fresh MATLAB run using the same system parameters as in the main evaluation. To ensure consistency across runs, the random number generator was initialized with `rng("default")`, so that only stochastic channel and user realizations differ, while all system parameters remain unchanged. This explains the minor numerical differences compared to earlier figures, without altering the qualitative conclusions.

We begin with the suburban case, which serves as a balanced benchmark between the near-ideal rural or open settings and the more challenging urban scenarios. Suburban conditions typically involve moderate shadowing from buildings and vegetation, as well as occasional multi-path effects, making them a realistic representation of many broadband user deployments.

Figures 5.21 and 5.22 depict the per-user average rate and SNR for both ideal and forecast modes at $P_s = 1$ W. The results are remarkably consistent across the user population, with little variance between users. Importantly, the forecast mode tracks the ideal mode closely, showing only a slight downward shift. Quantitatively, the overall average user rate is 9.889 bps/Hz in the ideal mode and 9.813 bps/Hz in the forecast mode, while the corresponding average SNRs are 30.06 dB and 29.85 dB. These marginal differences highlight the ability of the forecast-based assignment to approximate ideal performance even under imperfect channel knowledge.

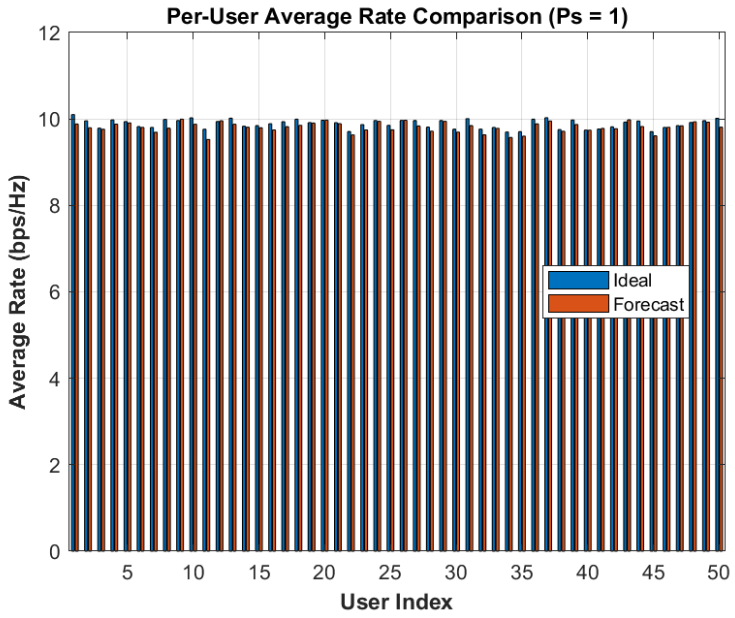


Figure 5.21: Per-user average rate comparison in suburban environment.

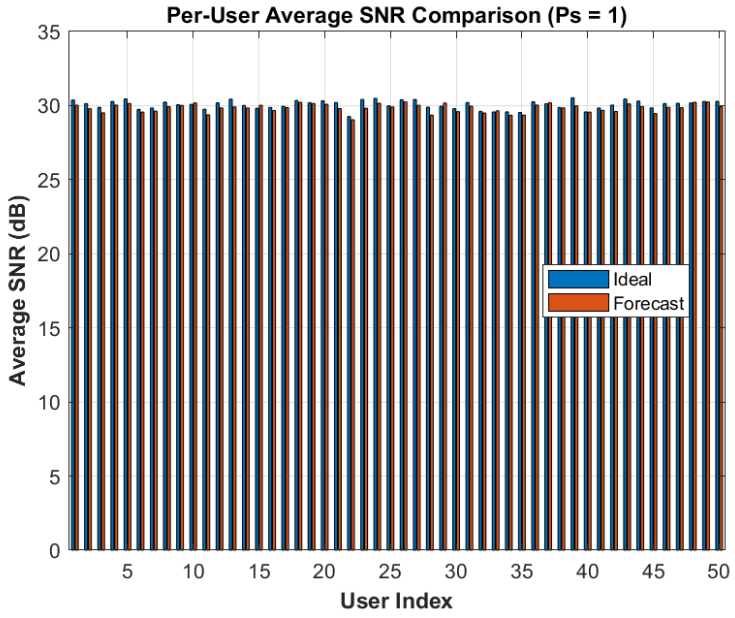


Figure 5.22: Per-user average SNR comparison in suburban environment.

A complementary perspective is provided by the cumulative distribution functions in Figures 5.23 and 5.24. These reveal that, although the forecast curves are consistently shifted left relative to the ideal, the spread remains tight, indicating strong fairness across users. The suburban channel introduces some variability due to multipath and intermittent shadowing, yet the dual-assignment strategy ensures that no user suffers catastrophic drops in quality. The forecast mode does produce a slightly wider dispersion in both rate and SNR, but the degradation is minor compared to the overall system performance.

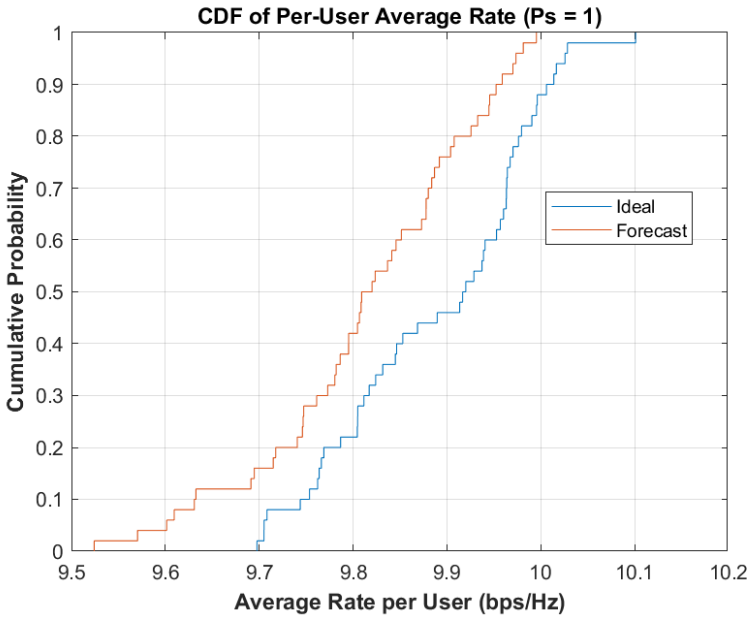


Figure 5.23: CDF of per-user average rate in suburban environment.

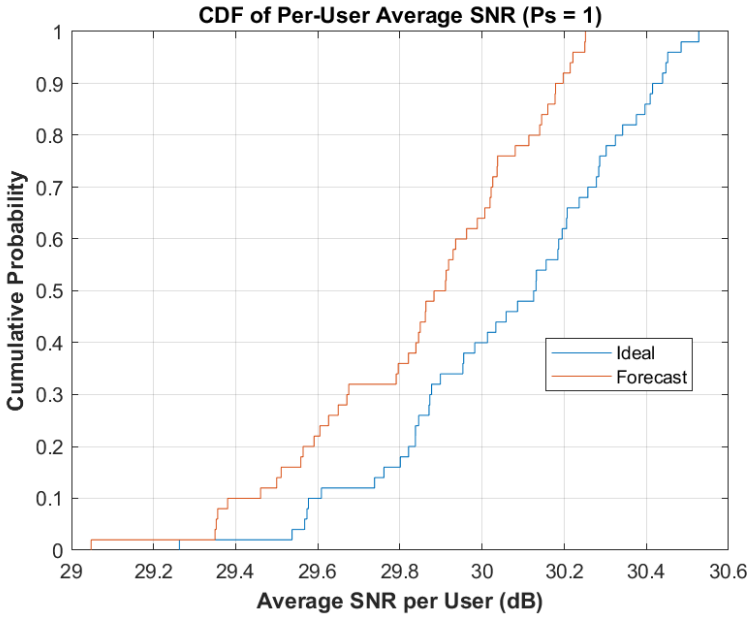


Figure 5.24: CDF of per-user average SNR in suburban environment.

Taken together, these results confirm that suburban conditions do not significantly erode the robustness of the system, it maintains nearly uniform service quality, with minimal gaps between ideal and forecast operation. This reinforces the role of suburban channels as a practical baseline for comparison: sufficiently challenging to expose the impact of fading and prediction errors, yet not so harsh as to obscure the benefits of dual-link handover.

### Urban Environment

The urban environment represents a significantly harsher propagation scenario compared to suburban conditions. Tall buildings, dense infrastructure, and frequent blockages introduce stronger shadowing and deeper multipath fading, resulting in a channel where line-of-sight availability is often intermittent. For LEO satellite systems, this setting is particularly challenging, as users may frequently transition between good and poor visibility conditions, testing the resilience of the handover algorithm.

Figures 5.25 and 5.26 illustrate the per-user average rate and SNR for the urban case under both ideal and forecast modes. The overall averages highlight a noticeable degradation compared to the suburban scenario: the mean throughput decreases to 8.76 bps/Hz (ideal) and 8.61 bps/Hz (forecast), while the mean SNR drops to 27.36 dB (ideal) and 26.95 dB (forecast). These reductions, while modest in absolute terms, underline the fact that denser environments systematically erode link quality. The forecast mode continues to closely track the ideal case, though the gap between the two is now slightly wider than in suburban settings, reflecting the increased difficulty of prediction in rapidly varying channels.

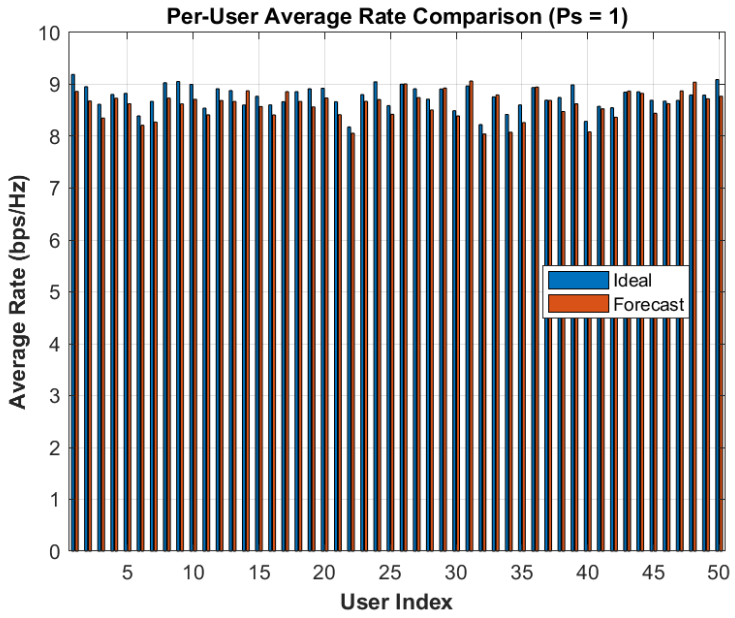


Figure 5.25: Per-user average rate comparison in urban environment.

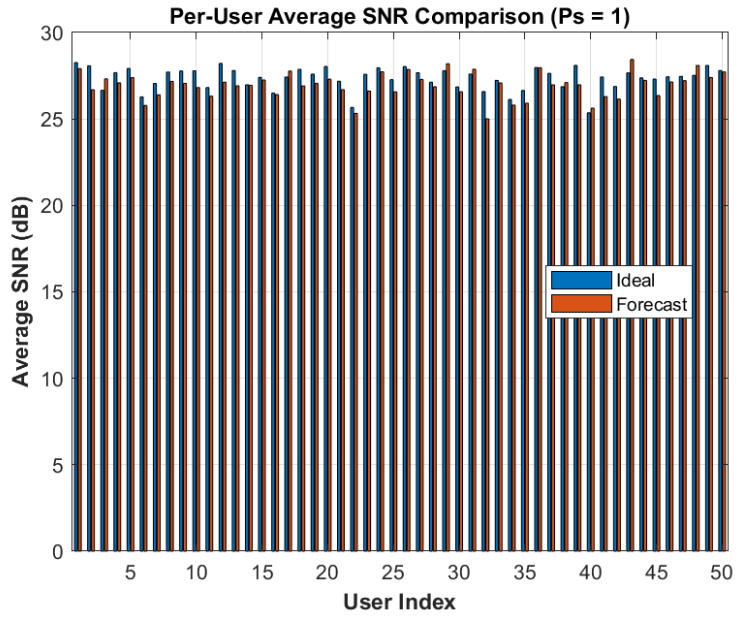


Figure 5.26: Per-user average SNR comparison in urban environment.

The distributional perspective provided by the CDFs in Figures 5.27 and 5.28 offers deeper insights. The forecast mode is consistently shifted left relative to the ideal, revealing that users are more exposed to weaker performance in this environment. In particular, the SNR CDF shows a broader spread, indicating that some users experience considerably lower average quality than others, even when dual assignments are employed. This widened distribution highlights the unevenness of service in dense urban deployments, where prediction errors can compound with environmental blockages to create larger disparities across users.

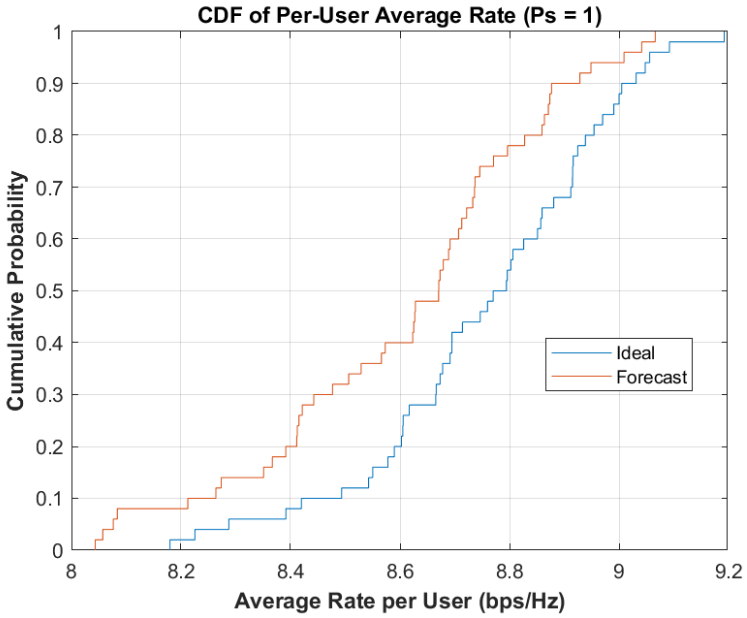


Figure 5.27: CDF of per-user average rate in urban environment.

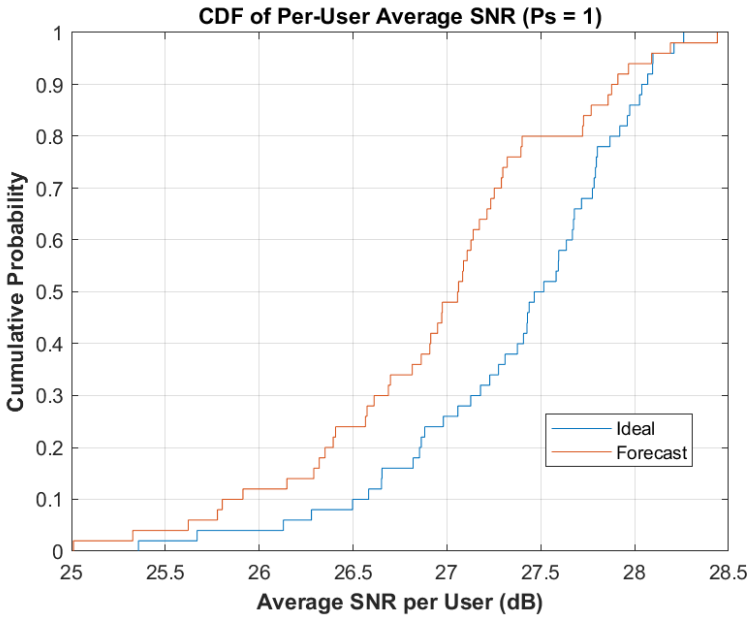


Figure 5.28: CDF of per-user average SNR in urban environment.

Despite these challenges, the system demonstrates commendable robustness. The dual assignment strategy prevents catastrophic drops in service quality, even when users face severe blockages or shadowing. Although the forecast mode performs slightly worse than the ideal case, the degradation remains modest and bounded, suggesting that the framework retains practical utility in urban settings. However, the results make clear that user fairness and consistency are more vulnerable here, and that urban deployments may demand additional enhancements, such as improved channel prediction or hybrid integration with terrestrial infrastructure, to guarantee uniformly high-quality service.

### Village Environment

The village profile sits between suburban and rural wooded conditions: buildings are sparser than in cities, yet shadowing and intermittent line-of-sight still shape the channel in ways that are more erratic than in open terrain. With the constellation geometry and user locations fixed, only the fading process was shifted to the *village* The LMS parameters produce a clear and instructive change in how the system behaves.

At a glance, the aggregate numbers already tell a story. The overall average rate drops to 8.58 bps/Hz in the ideal mode and 8.54 bps/Hz in the forecast mode, while the corresponding average SNRs are 21.18 dB and 21.02 dB, respectively. Compared to the suburban baseline and the urban case, the trend is monotonic: as the environment becomes less benign, both SNR and throughput decline. However, the gap between ideal and forecast remains modest (about 0.04 bps/Hz in rate and 0.16 dB in SNR on average), suggesting that the elevation-binned prediction still provides a serviceable proxy for instantaneous channel quality in this regime.

The per-user bar plots in Figures 5.29 and 5.30 expose the texture behind those averages.

The rates are broadly clustered between 7.5 and 9.5 bps/Hz, with a visible but small advantage for ideal allocation almost everywhere. SNRs, by contrast, exhibit a wider spread, dipping for some users into the high teens and low twenties. This widening is consistent with more pronounced slow shadowing and longer excursions of the direct component, which the algorithm cannot fully neutralize even with dual assignments. Still, the "two-hands-on-the-rope" nature of simultaneous handover is evident: users who experience a rougher first link tend not to collapse entirely, because the second link cushions the fall.

Distributional views reinforce that picture. In the rate CDF (Figure 5.31), both curves shift left relative to suburban/urban, and the forecast curve tracks the ideal one closely, separated by only a few tenths of a bit per second per hertz across most quantiles. The SNR CDF (Figure 5.32) shows the sharper environmental penalty: the median settles a couple of decibels lower than in urban, and the left tail stretches toward 14–16 dB. Despite this, the CDFs remain fairly steep, which implies that the system preserves a good degree of fairness—performance is degraded, but degraded for *everyone* in a controlled way rather than creating a class of chronically underserved users.

Two observations are worth underlining for design intuition. First, the small and steady gap between ideal and forecast modes indicates that most of the village-induced loss is *structural* (caused by the environment) rather than *algorithmic* (caused by prediction). The elevation-binned forecast captures enough of the average channel behavior to keep assignment choices near-optimal most of the time. Second, the simultaneous dual-link structure continues to pay off: even as SNR variance grows, the aggregate rate remains resilient because the algorithm exploits diversity in space and time, smoothing over deep fades that would otherwise drag the system down.

In short, village conditions compress the SNR budget more aggressively than suburban or urban, but the system remains stable and equitable. The penalty is real and visible in the leftward shift of the CDFs and the wider SNR bars, yet the forecast-controlled scheduler stays close to the ideal envelope, and the dual-assignment design continues to convert fragile instantaneous channels into a serviceable, human-grade experience.

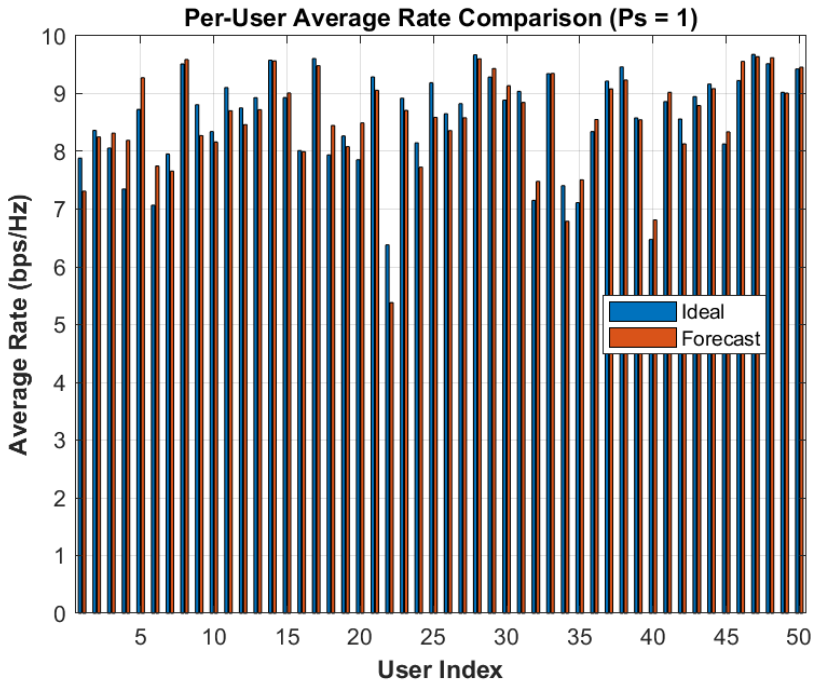


Figure 5.29: Per-user average rate in the village environment (Algorithm 1, $P_s = 1$).

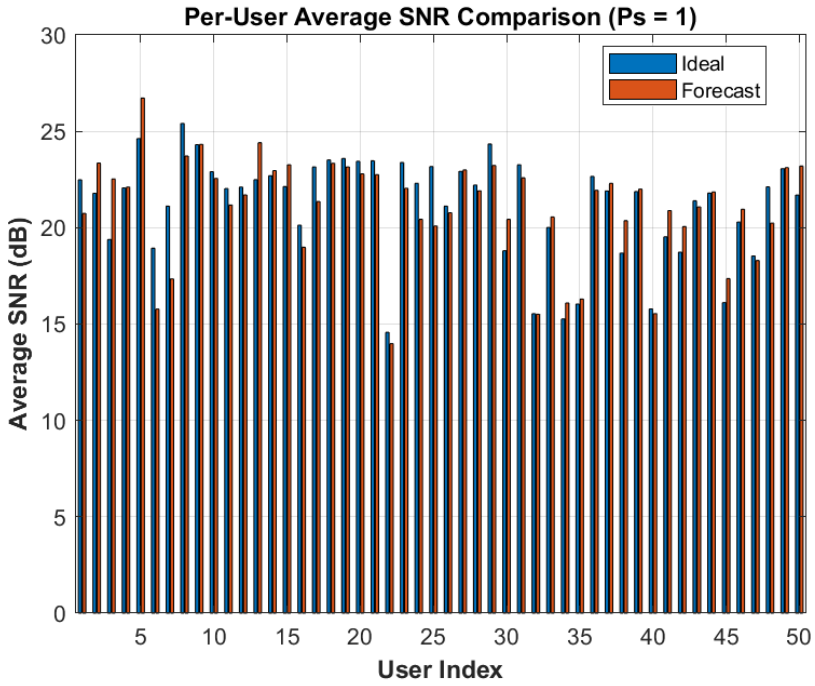


Figure 5.30: Per-user average SNR in the village environment (Algorithm 1, $P_s = 1$).

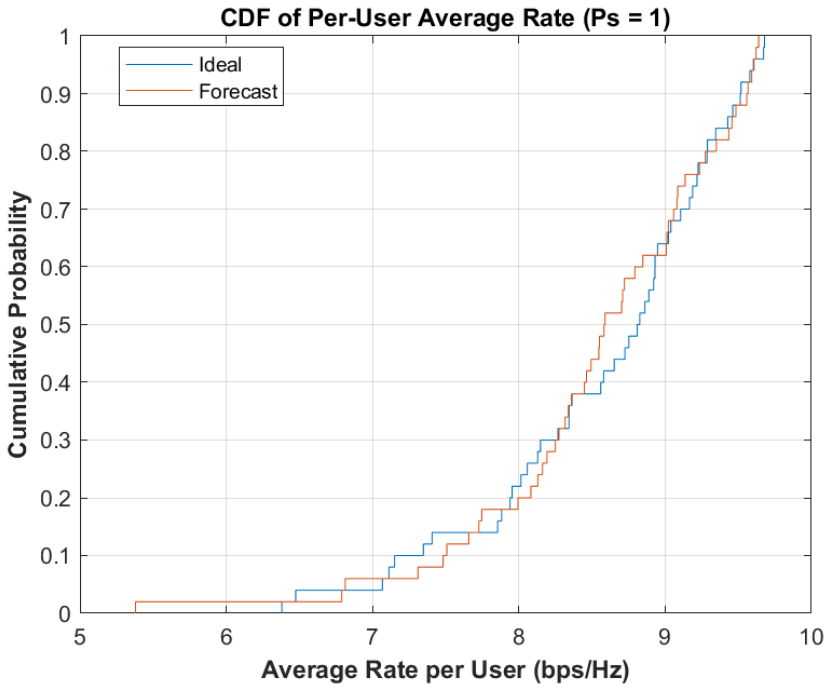


Figure 5.31: CDF of per-user average rate in the village environment (ideal vs. forecast).

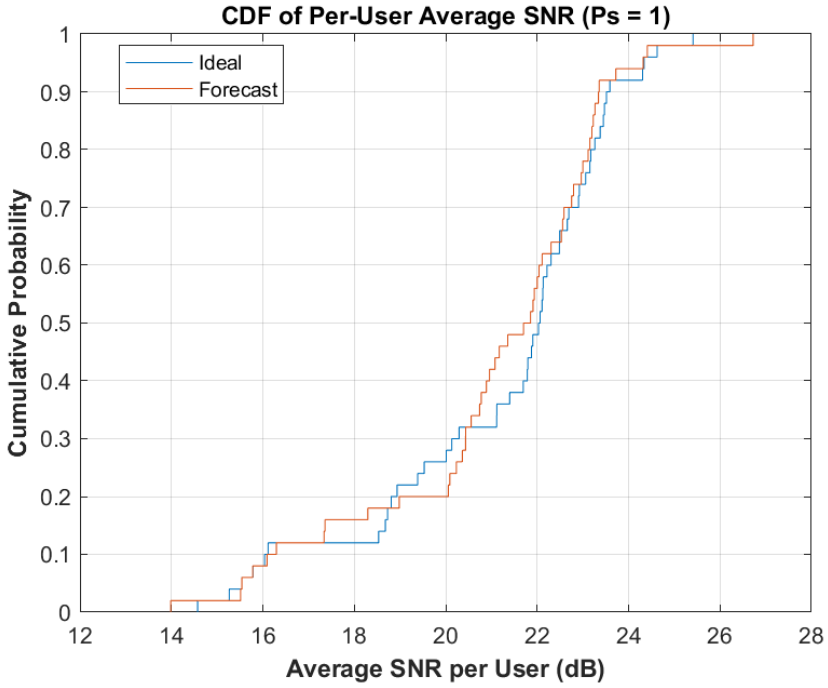


Figure 5.32: CDF of per-user average SNR in the village environment (ideal vs. forecast).

### Rural Wooded Environment

The rural wooded profile represents conditions where foliage and terrain dominate the channel, introducing additional attenuation and shadowing compared to open or lightly

obstructed areas. Unlike the urban and suburban cases, where buildings and infrastructure create sharp blockages, the rural wooded environment produces more gradual but persistent fading due to trees and uneven terrain. This makes it a useful test case for assessing the resilience of the system in sparsely populated, vegetation-heavy regions.

Quantitatively, the overall average throughput is reduced to 9.60 bps/Hz in the ideal mode and 9.42 bps/Hz in the forecast mode, while the corresponding SNR averages are 29.47 dB and 29.13 dB, respectively. These values represent a clear penalty relative to the suburban baseline, though the performance remains stronger than in the village case, reflecting that while the foliage introduces sustained attenuation, it does not degrade the channel as severely as deep shadowing in the village. Nevertheless, the gap between ideal and forecast operation remains modest: about 0.18 bps/Hz in rate and 0.34 dB in SNR. This suggests that the elevation-binned forecast retains predictive value even in environments with longer correlation times and deeper slow fades.

Figures 5.33 and 5.34 provide the per-user perspective. Here, the uniformity of rate performance is striking: nearly all users achieve between 9.0 and 9.8 bps/Hz, with only small deviations between ideal and forecast allocations. By contrast, the SNR results show a somewhat wider spread, with users clustering mostly between 28 and 30 dB. This indicates that while foliage attenuation reduces the SNR margin, the simultaneous assignment strategy successfully translates that into consistently high rates across the user set. In other words, the algorithm converts fragile SNRs into robust throughput outcomes by leveraging dual-link diversity.

The CDFs in Figures 5.35 and 5.36 further reinforce these conclusions. The forecast distributions are consistently shifted left relative to the ideal, but the separation remains small and the curves stay steep, signaling strong fairness across the population. The SNR CDF, in particular, highlights the environmental cost: the median lies below 29.5 dB, and the left tail approaches 28 dB. Yet even at these lower margins, the simultaneous handover framework prevents severe service inequality, ensuring that no users are pushed into extremely poor conditions.

Two insights stand out. First, most of the performance penalty here is structural, arising from the propagation environment rather than the forecasting scheme. The close tracking between forecast and ideal confirms that prediction errors are not the dominant factor. Second, the dual-link structure once again demonstrates its resilience: by combining partially independent fades, the algorithm cushions users against the deeper excursions of wooded channels, yielding rates that remain well above 9 bps/Hz for virtually all users.

In summary, rural wooded conditions reduce SNR and throughput compared to suburban, but still outperform urban and village profiles. This places rural wooded in the middle of the spectrum: not as strong as suburban, but significantly better than the harsher cases.

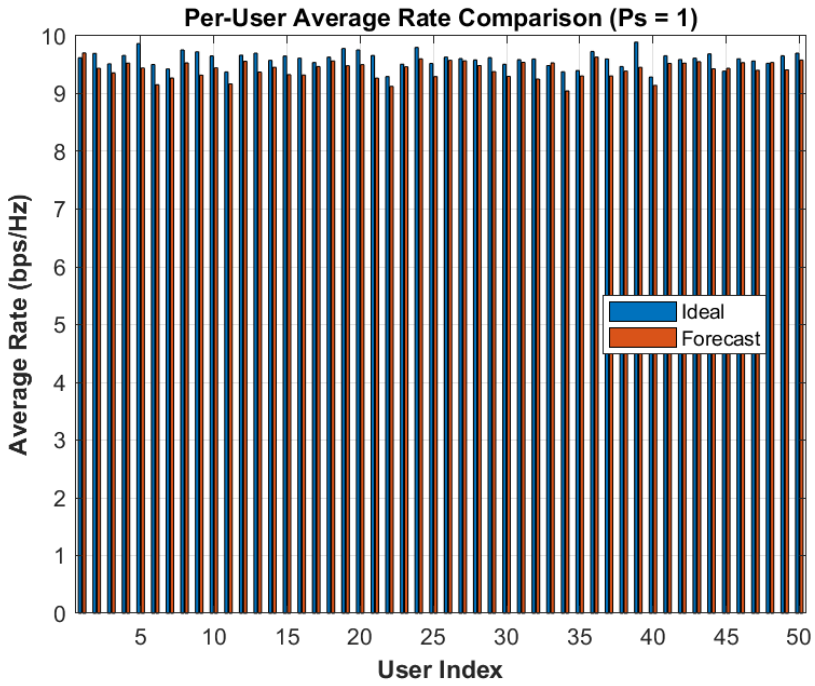


Figure 5.33: Per-user average rate in the rural wooded environment (Algorithm 1, $P_s = 1$).

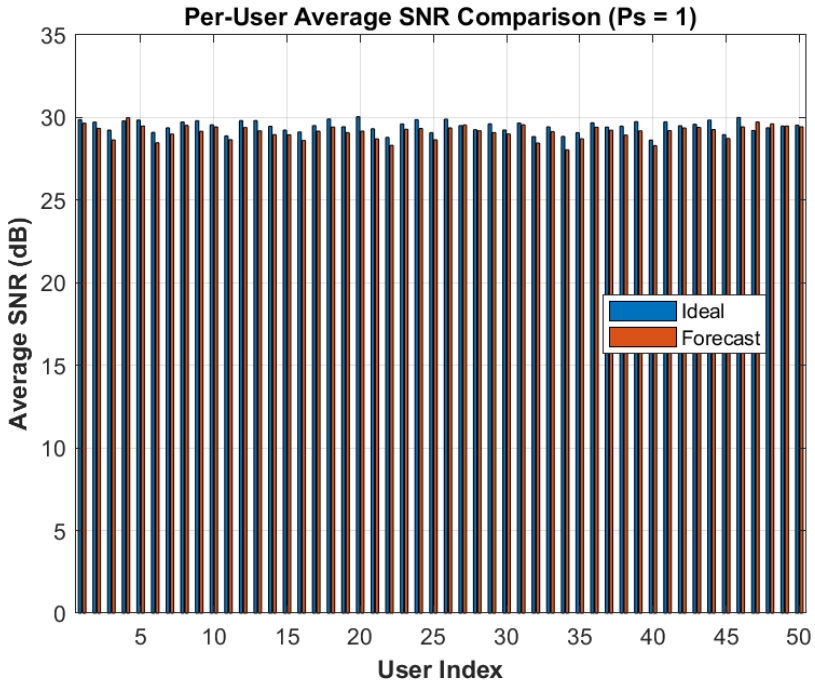


Figure 5.34: Per-user average SNR in the rural wooded environment (Algorithm 1, $P_s = 1$).

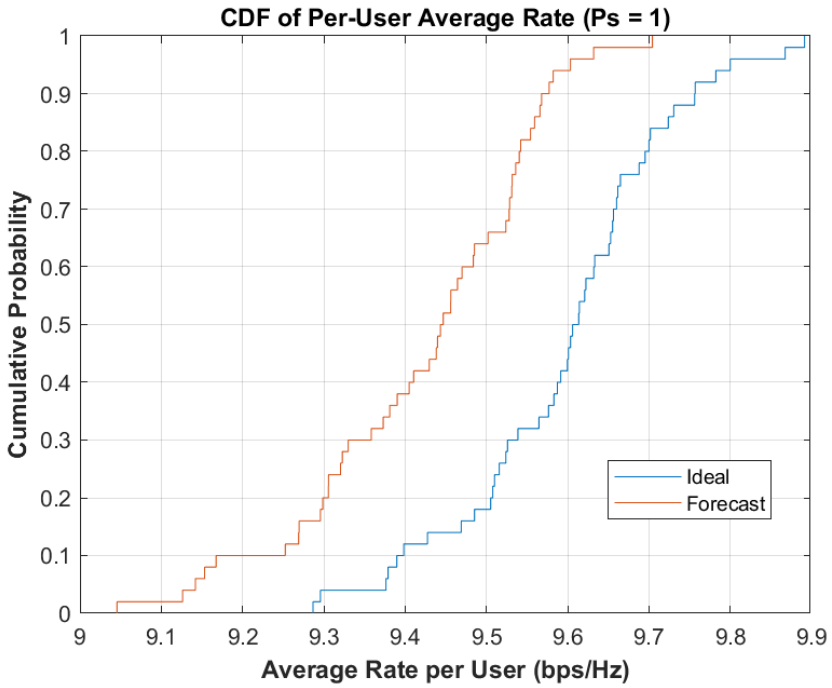


Figure 5.35: CDF of per-user average rate in the rural wooded environment (ideal vs. forecast).

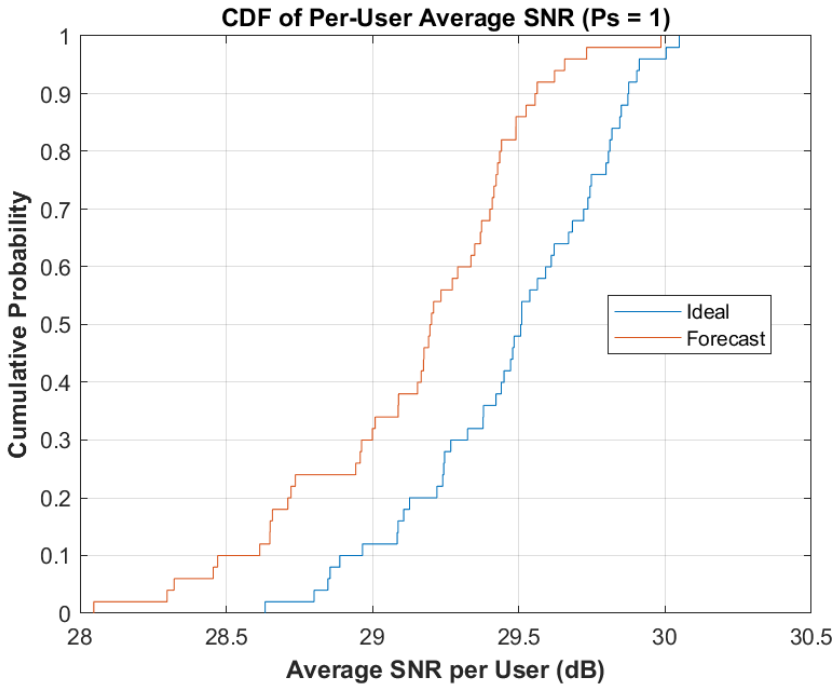


Figure 5.36: CDF of per-user average SNR in the rural wooded environment (ideal vs. forecast).

### 5.6.1 Overall Summary and Insights

The performance of the system across different propagation environments is summarized in Table 5.3. The table compares average per-user rate and SNR values under both ideal and forecast modes, and also reports the forecast penalty ($\Delta$). This consolidated view highlights the algorithm's robustness under varying conditions.

Table 5.3: Comparison of Algorithm 1 performance across environments ($P_s = 1$ W).

| **Metric** | **Suburban** | **Urban** | **Village** | **Rural Wooded** | **Average** |
|---|---|---|---|---|---|
| **Rate (bps/Hz)** | | | | | |
| Ideal | 9.889 | 8.760 | 8.580 | 9.600 | 9.207 |
| Forecast | 9.813 | 8.610 | 8.540 | 9.420 | 9.096 |
| $\Delta$ (Ideal−Forecast) | 0.076 | 0.150 | 0.040 | 0.180 | 0.112 |
| **SNR (dB)** | | | | | |
| Ideal | 30.06 | 27.36 | 21.18 | 29.47 | 27.02 |
| Forecast | 29.85 | 26.95 | 21.02 | 29.13 | 26.74 |
| $\Delta$ (Ideal−Forecast) | 0.21 | 0.41 | 0.16 | 0.34 | 0.28 |

$\Delta$ denotes the forecast penalty (Ideal − Forecast). A smaller $\Delta$ indicates tighter tracking of the ideal mode.

The results show three main trends:

- **Minimal forecast penalty:** In all environments, the gap between ideal and forecast performance is negligible, with throughput losses below 0.2 bps/Hz and SNR differences under 0.5 dB. This demonstrates that elevation-binned forecasting provides a reliable approximation of instantaneous channel quality.
- **Environmental sensitivity:** The main driver of performance variation is the environment itself rather than prediction error. Suburban and rural wooded conditions yield near-ideal performance (around 9.8 bps/Hz and 30 dB), while harsher village and urban settings reduce average throughput to $\sim$8.6 bps/Hz and SNR to as low as 21 dB.
- **Fairness and resilience:** Despite degradations in harsh environments, the system ensures equitable service quality. The dual-assignment strategy cushions users against deep fades, preventing catastrophic service drops and keeping distributions compact across the user population.

In summary, the system achieves near-ideal performance under imperfect knowledge, maintains robustness across heterogeneous environments, and preserves fairness among users. These properties make it a strong candidate for practical LEO broadband deployments, where both variability in channel conditions and imperfect prediction must be addressed simultaneously.

# Chapter 6

# Conclusion and Future Work

The design and optimization of user-to-satellite assignments in Low Earth Orbit (LEO) satellite networks constitute one of the most pressing challenges in modern communications engineering. This thesis has addressed this multifaceted problem through a comprehensive framework that merges theoretical modeling, algorithmic innovation, and empirical validation. At its core, this work reconciles the dynamic, transient nature of LEO constellations with the growing quality-of-service (QoS) expectations of modern applications, ranging from ultra-reliable low-latency communication (URLLC) to the vast connectivity demands of the Internet of Things (IoT).

LEO systems promise transformative communication capabilities due to their proximity to Earth, enabling high-throughput and low-latency links. However, these advantages are counterbalanced by the challenges introduced by rapid orbital motion, frequent handover events, and volatile link quality. In response, this thesis developed a dual-connectivity assignment strategy enriched by real-time link quality evaluation, satellite capacity constraints, and a globally optimal bipartite matching framework solved via the Kuhn-Munkres (Hungarian) algorithm Feng et al. (2020). This flexible design ensures robustness and scalability, aligning with the operational realities of mega-constellations like Starlink SpaceX (2020).

Empirical simulations, grounded in the ITU-R P.681-10 propagation standard International Telecommunication Union (2017), validated the performance of this framework under diverse conditions, showing substantial improvements in system throughput, user fairness, and handover continuity compared to baseline heuristics. Importantly, the graph-based formulation and modular weighting mechanism allow the integration of diverse decision metrics, such as signal strength, elevation angle, and predicted coverage duration, without compromising computational tractability.

Beyond this, the thesis explored recent innovations in handover enhancement. Notably, soft handover via Inter-Satellite Link (ISL) relaying Ben Salem et al. (2025) was shown to significantly reduce block error rates by maintaining dual connections during satellite transitions. Additionally, reinforcement learning approaches Liu et al. (2024) were analyzed for their potential to offer adaptive, decentralized decision-making in dynamic and large-scale networks.

By addressing key bottlenecks, ranging from satellite visibility constraints to fairness-aware user allocation, this thesis contributes a practically viable and theoretically grounded methodology. It aligns with the design trends emerging in next-generation 6G networks, where the integration of terrestrial and non-terrestrial components will require intelligent, agile handover management strategies Rago et al. (2024); Juan et al. (2022).

## 6.1 Limitations and Future Work

While the proposed framework demonstrates strong performance and adaptability, several limitations warrant discussion and offer fertile ground for future investigation.

### 6.1.1 Model Limitations

First, the assumption of perfect, instantaneous knowledge of satellite ephemeris and channel state information (CSI) simplifies the problem but diverges from real-world constraints. Delays in CSI feedback and ephemeris inaccuracies may degrade decision accuracy. Future implementations could integrate predictive modeling tools such as Kalman filters or neural predictors to estimate and smooth such information streams in real time.

Second, the optimization is conducted in a fully centralized, slot-based manner. While this design ensures optimality at each time step, it poses challenges for real-time scalability across thousands of users and hundreds of satellites. Implementing decentralized or hierarchical variants of the algorithm, potentially using onboard satellite computing or edge processing nodes, could significantly improve efficiency and fault tolerance.

Third, the current system model focuses on link quality and capacity-aware assignment, but does not incorporate parameters such as energy consumption, queueing latency, or user velocity. Incorporating these variables could allow for multi-objective optimization and better reflect real-world operating conditions, especially in heterogeneous user populations.

Moreover, although ITU-R P.681-10 channel models account for most large-scale fading effects, they do not cover all atmospheric phenomena. Effects such as rain attenuation, ionospheric scintillation, and Doppler shift may significantly impact performance in specific environments or frequency bands (e.g., Ka/Ku bands), and should be included in more comprehensive models.

### 6.1.2 Directions for Future Research

Building upon this work, several research avenues emerge:

- **Hybrid connectivity modeling:** Integration with terrestrial 6G networks and aerial platforms (e.g., UAVs and HAPS) could enable multi-layer handover strategies

that further reduce service interruptions Rago et al. (2024).

- **Learning-based optimization:** Deep reinforcement learning agents could be embedded into user terminals or satellites to make predictive, context-aware assignment decisions without centralized coordination Liu et al. (2024).
- **Dynamic topology control:** Beyond user assignment, controlling satellite transmission beams, resource blocks, and ISL topologies dynamically may further optimize end-to-end latency and system utility Wang et al. (2024).
- **Federated and edge learning:** To manage signaling overhead, federated learning frameworks could allow each satellite or user terminal to train models locally and share parameters with minimal bandwidth cost, improving adaptability while preserving privacy Zhai et al. (2024).

In summary, while the current framework establishes a strong foundation for optimizing user-to-satellite assignments in LEO constellations, realizing its full potential in operational deployments will require further innovation across modeling, computation, and integration with broader communication ecosystems.

# Appendix A

# Appendix

This appendix contains the main MATLAB functions and scripts used for the simulation and evaluation of the proposed user–satellite assignment algorithms. The code is organized into three groups:

- **Common utilities:** channel modeling, satellite visibility, weight matrix construction, and the Hungarian (Munkres) algorithm.
- **Algorithm 1 (Simultaneous Handover Assignment, SiHA):** assignment and simulation functions specific to Algorithm 1.
- **Algorithm 2 (Staggered Handover Assignment, StHA):** assignment and simulation functions specific to Algorithm 2.

## A.1 Common Utilities

### A.1.1 build_weight_matrices.m

```
function [WeightMatrix_SNR_ideal, WeightMatrix_Rate_ideal,
    WeightMatrix_SNR_forecast, WeightMatrix_Rate_forecast] =
    build_weight_matrices_AT(Ps, channel_trace_map, elevation_quantized_all,
    distance_ts_all, allowed_elevs, f_c)

U = size(elevation_quantized_all, 3);
M = size(elevation_quantized_all, 1);
T = size(elevation_quantized_all, 2);

distance_m_all = distance_ts_all * 1e3;
GGU_dBi = 0; GSAT_dBi = 50;
GGU = 10^(GGU_dBi / 10); GSAT = 10^(GSAT_dBi / 10);
B = 5e6; N0_dBmHz = -174;
N0 = 10^((N0_dBmHz - 30)/10) * B;
c = 3e8;
```

```
13
14  WeightMatrix_SNR_ideal = zeros(U, M, T);
15  WeightMatrix_Rate_ideal = zeros(U, M, T);
16  WeightMatrix_SNR_forecast = zeros(U, M, T);
17  WeightMatrix_Rate_forecast = zeros(U, M, T);
18
19  for u = 1:U
20      for m = 1:M
21          elev_row = elevation_quantized_all(m, :, u);
22          dist_row = distance_m_all(m, :, u);
23
24          for k = 1:length(allowed_elevs)
25              elev_val = allowed_elevs(k);
26              h2_trace = channel_trace_map(elev_val);
27
28              ind_elev_vec = find(elev_row == elev_val);
29              d = dist_row(ind_elev_vec);
30              L = (c ./ (4 * pi * d * f_c)).^2;
31              rho = Ps * GGU * GSAT * L / N0;
32
33              segLen = 1500;                      % samples per slot you want to
    average
34              Lh     = numel(h2_trace);
35
36              for ind = 1:length(ind_elev_vec)
37                  t = ind_elev_vec(ind);
38
39                  % start index for this time slot
40                  base = (t-1)*segLen;
41
42                  % circular indices to avoid out-of-bounds regardless of Lh
43                  idx  = mod(base : base+segLen-1, Lh) + 1;
44
45                  % 1×segLen window
46                  h_t = h2_trace(idx);
47
48                  snr_inst = rho(ind) .* h_t;
49                  WeightMatrix_SNR_ideal(u, m, t)  = 10 * log10(mean(snr_inst));
50                  WeightMatrix_Rate_ideal(u, m, t) = mean(log2(1 + snr_inst));
51              end
52
53
54              h_t = mean(h2_trace); % average channel power for forecast
55              snr = rho .* h_t;
56              WeightMatrix_SNR_forecast(u, m, ind_elev_vec) = 10 * log10(snr);
57              WeightMatrix_Rate_forecast(u, m, ind_elev_vec) = log2(1 + snr);
58          end
59      end
60  end
```

### A.1.2 coef_time_series_long_pedestrian.m

```
function [y, state_duration_vec, MA_vec, SigmaA_vec, MP_vec, L_trans_vec,
    average_power] = coef_time_series_long_pedestrian(f_c, envi, elev_deg, Lch)

% Generates a long sequence of channel samples from the land mobile
% satellite channel, according to the model described in ITU-R P.681-10,
% Sect. 6.

% INPUTS
% f_c:      carrier frequency [Hz]
% envi:     environment ('Urban', 'Suburban', 'Village', 'Rural wooded',
%   'Residential'
% elev_deg: quantized elevation [deg] (20, 30, 45, 60 or 70)

% OUTPUTS
% y: row vector containining the complex channel coefficients, sampled
%   every Ls metres (see below) for a total plath length of Lmax metres (see
    below)
% state_duration_vec: row vector of the sequence of state durations [m]
% MA_vec: row vector of the sequence of values of Loo parameter M_A
% SigmaA_vec: row vector of the sequence of values of Loo parameter Sigma_A
% MP_vec: row vector of the sequence of values of Loo parameter MP
% L_trans_vec: row vector of the sequence of transition lengths [m]

if(f_c >= 1.5e9 && f_c <= 3e9)
    switch envi
        case 'Urban'
            if(elev_deg == 20)
                mu = [2.0042, 3.689]; sigma = [1.2049, 0.9796];
                dur_min = [3.9889, 10.3114];
                mu_MA = [-3.3681, -18.1771]; sigma_MA = [3.3226, 3.2672];
                h1 = [0.1739, 1.1411]; h2 = [-11.5966, 4.0581];
                g1 = [0.0036, -0.2502]; g2 = [1.3230, -1.2528];
                Lcorr = [0.9680, 0.9680];
                f1 = 0.0870; f2 = 2.8469;
                pBmin = 0.1; pBmax = 0.9;
            elseif(elev_deg == 30)
                mu = [2.7332, 2.7582]; sigma = [1.1030, 1.2210];
                dur_min = [7.3174, 5.7276];
                mu_MA = [-2.3773, -17.4276]; sigma_MA = [2.1222, 3.9532];
                h1 = [0.0941, 0.9175]; h2 = [-13.1679, -0.8009];
                g1 = [-0.2811, -0.1484]; g2 = [0.9323, 0.5910];
                Lcorr = [1.4731, 1.4731];
                f1 = 0.1378; f2 = 3.3733;
                pBmin = 0.1; pBmax = 0.9;
            elseif(elev_deg == 45)
                mu = [3.0639, 2.9108];
                sigma = [1.6980, 1.2602];
                dur_min = [10.0, 6.0];
                mu_MA = [-1.8225, -15.4844];
```

```
                sigma_MA = [1.1317, 3.3245];
                h1 = [-0.0481, 0.9434];
                h2 = [-14.7450, -1.7555];
                g1 = [-0.4643, -0.0798];
                g2 = [0.3334, 2.8101];
                Lcorr = [1.7910, 1.7910];
                f1 = 0.0744;
                f2 = 2.1423;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 60)
                mu = [2.8135, 2.0211];
                sigma = [1.5962,  0.6568];
                dur_min = [10.0, 1.9126];
                mu_MA = [-1.5872, -14.1435];
                sigma_MA = [1.2446, 3.2706 ];
                h1 = [-0.5168, 0.6975];
                h2 = [-17.4060, -7.5383];
                g1 = [-0.1953, 0.0422];
                g2 = [0.5353, 3.2030];
                Lcorr = [1.7977, 1.7977];
                f1 = -0.1285;
                f2 = 5.4991;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 70)
                mu = [4.2919, 2.1012];
                sigma = [2.4703, 1.0341];
                dur_min = [118.3312, 4.8569];
                mu_MA = [-1.8434, -12.9383];
                sigma_MA = [0.5370, 1.7588];
                h1 = [-4.7301, 2.5318];
                h2 = [-26.5687, 16.8468];
                g1 = [0.5192, 0.3768];
                g2 = [1.9583, 8.4377];
                Lcorr = [2.0963, 2.0963];
                f1 = -0.0826;
                f2 = 2.8824;
                pBmin = 0.1;
                pBmax = 0.9;
            else
                error('Elevation not supported!')
            end
        case 'Suburban'
            if(elev_deg == 20)
                mu = [2.2201 2.2657];
                sigma = [1.2767 1.3812];
                dur_min = [2.2914 2.5585];
                mu_MA = [-2.7191 -13.8808];
                sigma_MA = [1.3840 2.5830];
                h1 = [-0.3037 1.0136];
```

```
            h2 = [-13.0719 0.5158];
            g1 = [-0.1254 -0.1441];
            g2 = [0.7894 0.7757];
            Lcorr = [0.9290 0.9290];
            f1 = 0.2904;
            f2 = 1.0324;
            pBmin = 0.1;
            pBmax = 0.9;
        elseif(elev_deg == 30)
            mu = [3.0138, 2.4521]; sigma = [1.4161, 0.7637];
            dur_min = [8.3214, 5.9087];
            mu_MA = [-0.7018, -11.9823]; sigma_MA = [1.2107, 3.4728];
            h1 = [-0.6543, 0.6200]; h2 = [-14.6457, -7.5485];
            g1 = [-0.1333, -0.1644]; g2 = [0.8992, 0.2762];
            Lcorr = [1.7135, 1.7135];
            f1 = 0.1091; f2 = 3.3000;
            pBmin = 0.1; pBmax = 0.9;
        elseif(elev_deg == 45)
            mu = [4.5857, 2.2414]; sigma = [1.3918, 0.7884];
            dur_min = [126.8375, 4.3132];
            mu_MA = [-1.1496, -10.3806]; sigma_MA = [1.0369, 2.3543];
            h1 = [0.2148, 0.0344]; h2 = [-17.8462, -14.2087];
            g1 = [0.0729, 0.0662]; g2 = [1.0303, 3.5043];
            Lcorr = [3.2293, 3.2293];
            f1 = 0.5766; f2 = 0.7163;
            pBmin = 0.1; pBmax = 0.9;
        elseif(elev_deg == 60)
            mu = [3.4124, 1.9922];
            sigma = [1.4331, 0.7132];
            dur_min = [19.5431, 3.1213];
            mu_MA = [-0.7811, -12.1436];
            sigma_MA = [0.7979, 3.1798];
            h1 = [-2.1102, 0.4372];
            h2 = [-19.7954, -8.3651];
            g1 = [-0.2284, -0.2903];
            g2 = [0.2796, -0.6001];
            Lcorr = [2.0215, 2.0215];
            f1 = -0.4097;
            f2 = 8.7440;
            pBmin = 0.1;
            pBmax = 0.9;
        elseif(elev_deg == 70)
            mu = [4.2919, 2.1012];
            sigma = [2.4703, 1.0341];
            dur_min = [118.3312, 4.8569];
            mu_MA = [-1.8434, -12.9383];
            sigma_MA = [0.5370, 1.7588];
            h1 = [-4.7301, 2.5318];
            h2 = [-26.5687, 16.8468];
            g1 = [0.5192, 0.3768];
            g2 = [1.9583, 8.4377];
```

```
                Lcorr = [2.0963, 2.0963];
                f1 = -0.0826;
                f2 = 2.8824;
                pBmin = 0.1;
                pBmax = 0.9;
            else
                error('Elevation not supported!')
            end
        case 'Village'
            if(elev_deg == 20)
                mu = [2.7663 2.2328];
                sigma = [1.1211 1.3788];
                dur_min = [6.5373 2.8174];
                mu_MA = [-2.5017 -15.2300];
                sigma_MA = [2.3059 5.0919];
                h1 = [0.0238 0.9971];
                h2 = [-11.4824 0.8970];
                g1 = [-0.2735 -0.0568];
                g2 = [1.3898 1.9253];
                Lcorr = [0.8574 0.8574];
                f1 = 0.0644;
                f2 = 2.6740;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 30)
                mu = [2.4246 1.8980];
                sigma = [1.3025 1.0505];
                dur_min = [5.4326 2.4696];
                h1 = [-2.2284 -15.1583];
                h2 = [1.4984 4.0987];
                g1 = [-0.3431 0.9614];
                g2 = [-14.0798 0.3719];
                mu_MA = [-0.2215 -0.0961];
                sigma_MA = [1.0077 1.3123];
                Lcorr = [0.8264 0.8264];
                f1 = -0.0576;
                f2 = 3.3977;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 45)
                mu = [2.8402 1.8509];
                sigma = [1.4563 0.8736];
                dur_min = [10.4906 2.6515];
                mu_MA = [-1.2871 -12.6718];
                sigma_MA = [0.6346 3.1722];
                h1 = [-0.0222 0.8329];
                h2 = [-16.7316 -3.9947];
                g1 = [-0.3905 -0.0980];
                g2 = [0.4880 1.3381];
                Lcorr = [1.4256 1.4256];
                f1 = -0.0493;
```

```
                f2 = 5.3952;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 60)
                mu = [3.7630 1.7192];
                sigma = [1.2854 1.1420];
                dur_min = [17.6726 2.5981];
                mu_MA = [-0.5364 -9.5399];
                sigma_MA = [0.6115 2.0732];
                h1 = [-0.1418 -0.4454];
                h2 = [-17.8032 -16.8201];
                g1 = [-0.2120 0.0609];
                g2 = [0.7819 2.5925];
                Lcorr = [0.8830 0.8830];
                f1 = -0.8818;
                f2 = 10.1610;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 70)
                mu = [4.0717 1.5673];
                sigma = [1.2475 0.5948];
                dur_min = [30.8829 2.1609];
                mu_MA = [-0.3340 -8.3686];
                sigma_MA = [0.6279 2.5603];
                h1 = [-1.6253 0.1788];
                h2 = [-19.7558 -9.5153];
                g1 = [-0.4438 -0.0779];
                g2 = [0.6355 1.1209];
                Lcorr = [1.5633 1.5633];
                f1 = -0.3483;
                f2 = 5.1244;
                pBmin = 0.1;
                pBmax = 0.9;
            else
                error('Elevation not supported!')
            end
        case 'Rural wooded'
            if(elev_deg == 20)
                mu = [2.1597 1.9587];
                sigma = [1.3766 1.5465];
                dur_min = [2.0744 1.3934];
                mu_MA = [-0.8065 -10.6615];
                sigma_MA = [1.5635 2.6170];
                h1 = [-0.9170 0.8440];
                h2 = [-12.1228 -1.4804];
                g1 = [-0.0348 -0.1069];
                g2 = [0.9571 1.6141];
                Lcorr = [0.8845 0.8845];
                f1 = 0.0550;
                f2 = 2.6383;
                pBmin = 0.1;
```

```
                pBmax = 0.9;
            elseif(elev_deg == 30)
                mu = [2.5579 2.3791];
                sigma = [1.2444 1.1778];
                dur_min = [3.5947 2.2800];
                mu_MA = [-1.3214 -10.4240];
                sigma_MA = [1.6645 2.4446];
                h1 = [-1.0445 0.6278];
                h2 = [-14.3176 -4.8146];
                g1 = [-0.1656 -0.0451];
                g2 = [0.7180 2.2327];
                Lcorr = [1.0942 1.0942];
                f1 = 0.0256;
                f2 = 3.8527;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 45)
                mu = [3.1803 2.5382];
                sigma = [1.3427 1.1291];
                dur_min = [6.7673 3.3683];
                mu_MA = [-0.9902 -10.2891];
                sigma_MA = [1.0348 2.3090];
                h1 = [-0.4235 0.3386];
                h2 = [-16.8380 -9.7118];
                g1 = [-0.1095 -0.0460];
                g2 = [0.6893 2.1310];
                Lcorr = [2.3956 2.3956];
                f1 = 0.2803;
                f2 = 4.0004;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 60)
                mu = [2.9322, 2.1955]; sigma = [1.3234, 1.1115];
                dur_min = [5.7209, 1.6512];
                mu_MA = [-0.6153, -9.9595]; sigma_MA = [1.1723, 2.2188];
                h1 = [-1.4024, 0.2666]; h2 = [-16.9664, -9.0046];
                g1 = [-0.2516, -0.0907]; g2 = [0.5353, 1.4730];
                Lcorr = [1.7586, 1.7586];
                f1 = 0.1099; f2 = 4.2183;
                pBmin = 0.1; pBmax = 0.9;
            elseif(elev_deg == 70)
                mu = [3.8768 1.8445];
                sigma = [1.4738 0.8874];
                dur_min = [16.0855 2.9629];
                mu_MA = [-0.7818 -6.7769];
                sigma_MA = [0.7044 2.1339];
                h1 = [-2.9566 -0.3723];
                h2 = [-20.0326 -14.9638];
                g1 = [-0.2874 -0.1822];
                g2 = [0.4050 0.1163];
                Lcorr = [1.6546 1.6546];
```

```
                f1 = -0.3914;
                f2 = 6.6931;
                pBmin = 0.1;
                pBmax = 0.9;
            else
                error('Elevation not supported!')
            end
        case 'Residential'
            if(elev_deg == 20)
                mu = [2.5818 1.7136];
                sigma = [1.7310 1.1421];
                dur_min = [9.2291 1.6385];
                mu_MA = [-0.8449 -10.8315];
                sigma_MA = [1.3050 2.2642];
                h1 = [-0.3977 0.8589];
                h2 = [-12.3714 -2.4054];
                g1 = [0.0984 -0.1804];
                g2 = [1.3138 0.8553];
                Lcorr = [1.1578 1.1578];
                f1 = 0.0994;
                f2 = 2.4200;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 30)
                mu = [3.2810 1.8414];
                sigma = [1.4200 0.9697];
                dur_min = [14.4825 2.7681];
                mu_MA = [-1.3799 -11.1669];
                sigma_MA = [1.0010 2.4724];
                h1 = [-0.8893 -0.1030];
                h2 = [-16.4615 -13.7102];
                g1 = [-0.2432 -0.1025];
                g2 = [0.6519 1.7671];
                Lcorr = [1.9053 1.9053];
                f1 = 0.0196;
                f2 = 3.9374;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 60)
                mu = [3.255 3.277];
                sigma = [1.287 1.260];
                dur_min = [6.47 7.81];
                mu_MA = [0 -2.32];
                sigma_MA = [0.30 2.06];
                h1 = [-2.024 -1.496];
                h2 = [-19.454 -22.894];
                g1 = [0.273 -0.361];
                g2 = [0.403 -0.119];
                Lcorr = [3.84 3.84];
                f1 = -1.591;
                f2 = 12.274;
```

```
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 70)
                mu = [4.3291 3.4534];
                sigma = [0.7249 0.9763];
                dur_min = [27.3637 8.9481];
                mu_MA = [-0.1625 -1.6084];
                sigma_MA = [0.3249 0.5817];
                h1 = [0.6321 -0.3976];
                h2 = [-21.5594 -22.7905];
                g1 = [0.1764 -0.0796];
                g2 = [0.4135 0.1939];
                Lcorr = [1.6854 1.6854];
                f1 = 3.0127;
                f2 = 6.2345;
                pBmin = 0.1;
                pBmax = 0.9;
            else
                error('Elevation not supported for this environment!')
            end
        otherwise
            error('Environment not supported!')
    end
elseif(f_c > 3e9 && f_c <= 5e9)
    switch envi
        case 'Urban'
            if(elev_deg == 20)
                mu = [2.5467 3.6890];
                sigma = [1.0431 0.9796];
                dur_min = [5.2610 10.3114];
                mu_MA = [-2.7844 -19.4022];
                sigma_MA = [2.6841 3.2428];
                h1 = [0.1757 0.9638];
                h2 = [-12.9417 -0.9382];
                g1 = [-0.2044 0.0537];
                g2 = [1.5866 4.5670];
                Lcorr = [1.4243 1.4243];
                f1 = 0.1073;
                f2 = 1.9199;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 30)
                mu = [2.0158 2.2627];
                sigma = [1.2348 1.4901];
                dur_min = [4.5491 2.0749];
                mu_MA = [-3.7749 -17.9098];
                sigma_MA = [2.2381 2.9828];
                h1 = [-0.1564 0.8250];
                h2 = [-15.1531 -2.5833];
                g1 = [-0.0343 -0.0741];
                g2 = [1.0602 2.1406];
```

```
                Lcorr = [0.8999 0.8999];
                f1 = 0.2707;
                f2 = -0.0287;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 45)
                mu = [2.3005 2.6314];
                sigma = [1.6960 1.1210];
                dur_min = [10.0 6.0];
                mu_MA = [-1.4466 -15.3926];
                sigma_MA = [1.1472 3.2527];
                h1 = [0.1550 0.9509];
                h2 = [-13.6861 -1.2462];
                g1 = [0.1666 0.0363];
                g2 = [1.2558 4.4356];
                Lcorr = [1.6424 1.6424];
                f1 = 0.2517;
                f2 = -0.3512;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 60)
                mu = [2.4546 1.8892];
                sigma = [1.9595 0.8982];
                dur_min = [10.0 1.9126];
                mu_MA = [-1.6655 -14.4922];
                sigma_MA = [0.8244 3.4941];
                h1 = [-0.4887 0.4501];
                h2 = [-17.2505 -9.6935];
                g1 = [-0.3373 0.1202];
                g2 = [0.3285 4.8329];
                Lcorr = [2.3036 2.3036];
                f1 = 0.0025;
                f2 = 1.4949;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 70)
                mu = [2.8354 1.5170];
                sigma = [2.4631 1.1057];
                dur_min = [67.5721 3.6673];
                mu_MA = [-1.0455 -14.2294];
                sigma_MA = [0.2934 5.4444];
                h1 = [-3.0973 0.0908];
                h2 = [-20.7862 -15.8022];
                g1 = [0.0808 0.0065];
                g2 = [0.8952 3.1520];
                Lcorr = [2.2062 2.2062];
                f1 = 0.0755;
                f2 = 2.1426;
                pBmin = 0.1;
                pBmax = 0.9;
            else
```

```
                error('Elevation not supported!')
            end
        case 'Suburban'
            if(elev_deg == 20)
                mu = [2.8194 2.5873];
                sigma = [1.6507 1.3919];
                dur_min = [11.1083 4.4393];
                mu_MA = [-4.8136 -17.0970];
                sigma_MA = [1.9133 2.9350];
                h1 = [-0.4500 0.8991];
                h2 = [-17.9227 -2.4082];
                g1 = [-0.1763 0.0582];
                g2 = [0.8244 4.0347];
                Lcorr = [1.2571 1.2571];
                f1 = 0.0727;
                f2 = 2.8177;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 30)
                mu = [2.9226 2.7375];
                sigma = [1.3840 0.6890];
                dur_min = [6.7899 7.7356];
                mu_MA = [-1.9611 -15.3022];
                sigma_MA = [1.8460 2.9379];
                h1 = [0.2329 0.5146];
                h2 = [-15.0063 -8.9987];
                g1 = [0.0334 0.0880];
                g2 = [1.3323 4.4692];
                Lcorr = [1.6156 1.6156];
                f1 = 0.1281;
                f2 = 2.3949;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 45)
                mu = [4.3019 2.3715];
                sigma = [0.8530 1.3435];
                dur_min = [36.1277 9.5511];
                mu_MA = [-1.2730 -5.6373];
                sigma_MA = [0.9286 2.9302];
                h1 = [0.2050 -0.7188];
                h2 = [-17.5670 -21.0513];
                g1 = [0.0074 -0.2896];
                g2 = [0.7490 -0.3951];
                Lcorr = [1.1191 1.1191];
                f1 = -0.9586;
                f2 = 10.8084;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 60)
                mu = [2.8958 1.9128];
                sigma = [1.7061 0.6869];
```

```
                dur_min = [13.9133 2.9398];
                mu_MA = [-1.1987 -13.1811];
                sigma_MA = [1.0492 2.6228];
                h1 = [-1.6501 0.6911];
                h2 = [-18.9375 -6.0721];
                g1 = [-0.1369 0.0598];
                g2 = [0.4477 3.7220];
                Lcorr = [3.0619 3.0619];
                f1 = -0.0419;
                f2 = 5.8920;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 70)
                mu = [4.1684 1.4778];
                sigma = [1.0766 0.7033];
                dur_min = [42.0185 1.8473];
                mu_MA = [0.1600 -10.2225];
                sigma_MA = [0.5082 1.8417];
                h1 = [-3.4369 0.3934];
                h2 = [-18.1632 -9.6284];
                g1 = [-1.1144 -0.1331];
                g2 = [0.9703 0.7223];
                Lcorr = [2.5817 2.5817];
                f1 = -0.1129;
                f2 = 4.0555;
                pBmin = 0.1;
                pBmax = 0.9;
            else
                error('Elevation not supported!')
            end
        case 'Village'
            if(elev_deg == 20)
                mu = [2.0262 1.9451];
                sigma = [1.2355 1.4293];
                dur_min = [2.2401 1.9624];
                mu_MA = [-3.1324 -16.5697];
                sigma_MA = [1.8929 4.0368];
                h1 = [-0.4368 1.0921];
                h2 = [-15.1009 1.6440];
                g1 = [-0.0423 -0.0325];
                g2 = [1.2532 2.4452];
                Lcorr = [0.8380 0.8380];
                f1 = 0.0590;
                f2 = 1.5623;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 30)
                mu = [2.4504 1.7813];
                sigma = [1.1061 1.2802];
                dur_min = [2.3941 2.1484];
                mu_MA = [-1.8384 -15.4143];
```

```
                sigma_MA = [1.7960 4.5579];
                h1 = [-0.5582 0.8549];
                h2 = [-14.4416 -2.2415];
                g1 = [-0.4545 -0.0761];
                g2 = [0.8188 1.6768];
                Lcorr = [0.9268 0.9268];
                f1 = -0.0330;
                f2 = 2.7056;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 45)
                mu = [2.2910 1.2738];
                sigma = [1.4229 1.1539];
                dur_min = [2.8605 0.7797];
                mu_MA = [-0.0018 -12.1063];
                sigma_MA = [1.1193 2.9814];
                h1 = [-1.2023 0.6537];
                h2 = [-14.0732 -4.5948];
                g1 = [-0.1033 -0.0815];
                g2 = [0.9299 1.6693];
                Lcorr = [0.9288 0.9288];
                f1 = 0.0002;
                f2 = 1.9694;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 60)
                mu = [3.0956 1.0920];
                sigma = [1.3725 1.2080];
                dur_min = [8.1516 0.7934];
                mu_MA = [-0.5220 -12.1817];
                sigma_MA = [1.0950 3.3604];
                h1 = [0.0831 1.1006];
                h2 = [-16.8546 0.5381];
                g1 = [0.0411 -0.0098];
                g2 = [1.1482 2.4287];
                Lcorr = [1.2251 1.2251];
                f1 = -0.0530;
                f2 = 2.7165;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 70)
                mu = [3.9982 1.4165];
                sigma = [1.3320 0.4685];
                dur_min = [28.3220 2.5168];
                mu_MA = [-1.3403 -11.9560];
                sigma_MA = [0.7793 1.5654];
                h1 = [-0.4861 0.5663];
                h2 = [-19.5316 -6.8615];
                g1 = [-0.2356 -0.2903];
                g2 = [0.7178 -1.2715];
                Lcorr = [1.4378 1.4378];
```

```
                f1 = -0.0983;
                f2 = 3.9005;
                pBmin = 0.1;
                pBmax = 0.9;
            else
                error('Elevation not supported!')
            end
        case 'Rural wooded'
            if(elev_deg == 20)
                mu = [2.0294 2.0290];
                sigma = [1.4280 1.5493];
                dur_min = [1.7836 1.5269];
                mu_MA = [-3.2536 -14.3363];
                sigma_MA = [1.6159 2.7753];
                h1 = [-0.5718 0.8186];
                h2 = [-16.1382 -2.9963];
                g1 = [-0.0805 -0.0822];
                g2 = [0.9430 1.7660];
                Lcorr = [1.0863 1.0863];
                f1 = 0.1263;
                f2 = 1.4478;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 30)
                mu = [2.1218 2.2051];
                sigma = [1.4895 1.5741];
                dur_min = [2.4539 2.1289];
                mu_MA = [-1.5431 -12.8884];
                sigma_MA = [1.8811 3.0097];
                h1 = [-0.7288 0.6635];
                h2 = [-14.1626 -4.6034];
                g1 = [-0.1241 -0.0634];
                g2 = [0.9482 2.3898];
                Lcorr = [1.3253 1.3253];
                f1 = 0.0849;
                f2 = 1.6324;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 45)
                mu = [3.1803 2.4017];
                sigma = [1.3427 1.1315];
                dur_min = [6.7673 3.5668];
                mu_MA = [0.0428 -11.3173];
                sigma_MA = [1.6768 2.7467];
                h1 = [-0.9948 0.2929];
                h2 = [-14.4265 -9.7910];
                g1 = [-0.1377 -0.0387];
                g2 = [1.0077 2.6194];
                Lcorr = [2.0419 2.0419];
                f1 = 0.1894;
                f2 = 2.1378;
```

```
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 60)
                mu = [2.4961 2.2113];
                sigma = [1.4379 1.1254];
                dur_min = [3.7229 1.9001];
                mu_MA = [-1.0828 -12.3044];
                sigma_MA = [1.0022 2.3641];
                h1 = [-1.2973 0.5456];
                h2 = [-16.6791 -6.4660];
                g1 = [-0.1187 -0.0443];
                g2 = [0.6254 2.3029];
                Lcorr = [1.9038 1.9038];
                f1 = 0.1624;
                f2 = 1.8417;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 70)
                mu = [2.8382 2.1470];
                sigma = [1.3804 1.0038];
                dur_min = [6.8051 1.9195];
                mu_MA = [-0.8923 -11.5722];
                sigma_MA = [0.9455 2.3437];
                h1 = [-1.3425 0.3459];
                h2 = [-17.5636 -9.5399];
                g1 = [-0.1210 -0.0275];
                g2 = [0.6444 2.6238];
                Lcorr = [2.1466 2.1466];
                f1 = 0.0593;
                f2 = 2.8854;
                pBmin = 0.1;
                pBmax = 0.9;
            else
                error('Elevation not supported!')
            end
        case 'Residential'
            if(elev_deg == 20)
                mu = [2.9050 2.1969];
                sigma = [1.7236 0.9865];
                dur_min = [10.7373 2.2901];
                mu_MA = [-1.4426 -14.4036];
                sigma_MA = [1.2989 3.0396];
                h1 = [0.4875 0.5813];
                h2 = [-13.5981 -6.9790];
                g1 = [0.1343 -0.0911];
                g2 = [1.8247 2.1475];
                Lcorr = [1.2788 1.2788];
                f1 = 0.2334;
                f2 = 0.7612;
                pBmin = 0.1;
                pBmax = 0.9;
```

```
            elseif(elev_deg == 30)
                mu = [2.7334 1.8403];
                sigma = [1.6971 0.9268];
                dur_min = [10.2996 1.8073];
                mu_MA = [-0.9996 -12.9855];
                sigma_MA = [1.0752 2.8149];
                h1 = [0.3407 0.3553];
                h2 = [-14.8465 -9.9284];
                g1 = [-0.0413 0.0501];
                g2 = [1.2006 3.8667];
                Lcorr = [1.7072 1.7072];
                f1 = 0.0443;
                f2 = 2.2591;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 60)
                mu = [3.4044 2.5534];
                sigma = [1.3980 1.7143];
                dur_min = [10.4862 4.7289];
                mu_MA = [0.4640 -2.3787];
                sigma_MA = [0.7060 0.8123];
                h1 = [0.3710 -2.3834];
                h2 = [-19.6032 -24.6987];
                g1 = [0.0332 0.0172];
                g2 = [0.5053 0.7237];
                Lcorr = [1.8017 1.8017];
                f1 = 3.1149;
                f2 = 3.5721;
                pBmin = 0.1;
                pBmax = 0.9;
            elseif(elev_deg == 70)
                mu = [2.9223 2.5188];
                sigma = [1.0267 1.3166];
                dur_min = [7.3764 7.2801];
                mu_MA = [-0.1628 -2.3703];
                sigma_MA = [0.5104 1.5998];
                h1 = [0.1590 -1.0228];
                h2 = [-20.4767 -22.4769];
                g1 = [0.1137 -0.0986];
                g2 = [0.4579 0.2879];
                Lcorr = [1.3531 1.3531];
                f1 = -0.0538;
                f2 = 5.1204;
                pBmin = 0.1;
                pBmax = 0.9;
            else
                error('Elevation not supported for this environment!')
            end
        otherwise
            error('Environment not supported!')
    end
```

```
762 else
763     error('Carrier frequency not supported!')
764 end
765
766 %rng(555)
767
768 % Computation of averages
769 state_mean_duration = exp(mu + sigma.^2/2) .* (1 - erf((log(dur_min) - (mu +
    sigma.^2)) ./ sigma / sqrt(2))) ./ (1 - erf((log(dur_min) - mu) ./ sigma /
    sqrt(2)));
770 MA_min = [mu_MA(1) - 1.645*sigma_MA(1), mu_MA(2) + sqrt(2)*sigma_MA(2)*erfinv
    (2*pBmin-1)];
771 MA_max = [mu_MA(1) + 1.645*sigma_MA(1), mu_MA(2) + sqrt(2)*sigma_MA(2)*erfinv
    (2*pBmax-1)];
772 trans_mean_duration = f1 * (mu_MA(1) - mu_MA(2) - sigma_MA(2)^2 * (normpdf(
    MA_min(2),mu_MA(2),sigma_MA(2)) - normpdf(MA_max(2),mu_MA(2),sigma_MA(2)))
    ...
773     / (normcdf(MA_max(2),mu_MA(2),sigma_MA(2)) - normcdf(MA_min(2),mu_MA(2),
    sigma_MA(2)))) + f2;
774
775 state_distr = state_mean_duration + trans_mean_duration;
776 state_distr = state_distr / sum(state_distr);
777
778 K1 = log(10)/10;
779 b0 = (K1 * g2).^2 / 2;
780 b1 = K1^2 * g1 .* g2 + K1;
781 b2 = (K1 * g1).^2 / 2;
782 v1 = 1 - 2 * sigma_MA.^2 .* b2;
783 mu1 = (mu_MA + sigma_MA.^2 .* b1) ./ v1;
784 sigma1 = sigma_MA ./ sqrt(v1);
785 v2 = (mu_MA.^2 - 2 * sigma_MA.^2 .* b0) ./ v1;
786 average_power_per_state1 = exp((mu1.^2 - v2) ./ (2 * sigma1.^2)) ./ sqrt(v1) .*
     (normcdf(MA_max,mu1,sigma1) - normcdf(MA_min,mu1,sigma1)) ...
787     ./ (normcdf(MA_max,mu_MA,sigma_MA) - normcdf(MA_min,mu_MA,sigma_MA));
788
789 b0 = K1 * h2;
790 b1 = K1 * h1;
791 mu1 = mu_MA + sigma_MA.^2 .* b1;
792 v2 = mu_MA.^2 - 2 * sigma_MA.^2 .* b0;
793 average_power_per_state2 = exp((mu1.^2 - v2) ./ (2 * sigma_MA.^2)) .* (normcdf(
    MA_max,mu1,sigma_MA) - normcdf(MA_min,mu1,sigma_MA)) ...
794     ./ (normcdf(MA_max,mu_MA,sigma_MA) - normcdf(MA_min,mu_MA,sigma_MA));
795
796 average_power_per_state = average_power_per_state1 + average_power_per_state2;
797
798 average_power = sum(state_distr .* average_power_per_state);
799
800
801
802 Ts = 0.02/3;                        % Sampling time [s]
803 vm = 5000 / 3600;                  % Mobile speed (pedestrian) [m/s]
```

```
Ls = vm * Ts;                    % Sampling distance [m]
Lmax = Lch;%1e5;                      % Path length [m]
Nsamples = round(Lmax / Ls);     % Total number of samples
c = physconst('LightSpeed');     % Speed of light [m/s]
fD = f_c * vm / c;               % Doppler shift [Hz]
Lacc = 0;
curr_sample = 0;
state_ind = 1;
state = 1;                       % The initial state is 1 (Good) - Bad is 2

Jakes_impulse_response = Jakes_IR(fD, 1/Ts);

y = zeros(1,Nsamples);

state_duration_vec = zeros(1,2500);
MA_vec = zeros(1,2500);
SigmaA_vec = zeros(1,2500);
MP_vec = zeros(1,2500);
L_trans_vec = zeros(1,2500);
while(Lacc < Lmax)

    % Generation of state duration (length)
    curr_state_duration_m = lognrnd(mu(state), sigma(state));
    while(curr_state_duration_m < dur_min(state))
        curr_state_duration_m = lognrnd(mu(state), sigma(state));
    end
    state_duration_vec(state_ind) = curr_state_duration_m;
    Lacc = Lacc + curr_state_duration_m;

    % Generation of parameters of Loo distribution
    app_var = normrnd(mu_MA(state), sigma_MA(state));
    while(app_var < MA_min(state) || app_var > MA_max(state))
        app_var = normrnd(mu_MA(state), sigma_MA(state));
    end
    MA_vec(state_ind) = app_var;

    SigmaA_vec(state_ind) = g1(state) * MA_vec(state_ind) + g2(state);
    MP_vec(state_ind) = h1(state) * MA_vec(state_ind) + h2(state);

    % Computation of the transition length
    if(state_ind > 1)
        curr_trans_length_m = f1 * abs(MA_vec(state_ind)-MA_vec(state_ind-1)) +
     f2;
        L_trans_vec(state_ind) = curr_trans_length_m;
    else
        curr_trans_length_m = 0;
    end
    Lacc = Lacc + curr_trans_length_m;

    curr_trans_length_samples = round(curr_trans_length_m / Ls);
    curr_state_duration_samples = round(curr_state_duration_m / Ls);
```

```
854
855     rho_S = exp(-Ls / Lcorr(state));
856
857     % Generation of channel samples in the transition
858     if(curr_trans_length_m > 0)
859         % Loo parameters in the transition are linearly interpolated
860         MA_trans = ((1:curr_trans_length_samples) * MA_vec(state_ind) + (
        curr_trans_length_samples:-1:1) * MA_vec(state_ind-1)) / (
        curr_trans_length_samples+1);
861         SigmaA_trans = ((1:curr_trans_length_samples) * SigmaA_vec(state_ind) +
         (curr_trans_length_samples:-1:1) * SigmaA_vec(state_ind-1)) / (
        curr_trans_length_samples+1);
862         MP_trans = ((1:curr_trans_length_samples) * MP_vec(state_ind) + (
        curr_trans_length_samples:-1:1) * MP_vec(state_ind-1)) / (
        curr_trans_length_samples+1);
863
864         % Direct component
865         A_vec = normrnd(0, SigmaA_trans);
866         A_vec = filter(sqrt(1-rho_S^2), [1,-rho_S], A_vec, (1-sqrt(1-rho_S^2))/
        rho_S*A_vec(1)) + MA_trans;
867         alpha_vec = 10.^(A_vec/20);
868
869         % Scattered (diffused) component
870         diff_vec = normrnd(0, 1, 1, curr_trans_length_samples) + 1i * normrnd
        (0, 1, 1, curr_trans_length_samples);
871         %diff_vec = Jakes(diff_vec, fD, 1/Ts);
872         diff_vec = conv(diff_vec, Jakes_impulse_response,'same');
873         sigmad = sqrt(0.5 * 10.^(MP_trans/10));
874         diff_vec = sigmad .* diff_vec;
875
876         % Superposition of direct and diffused
877         alpha_vec = alpha_vec + diff_vec;
878
879         y(curr_sample + (1:curr_trans_length_samples)) = alpha_vec;
880         curr_sample = curr_sample + curr_trans_length_samples;
881     end
882
883     % Generation of channel samples in the state
884     % Direct component
885     A_vec = normrnd(0, SigmaA_vec(state_ind), 1, curr_state_duration_samples);
886     A_vec = filter(sqrt(1-rho_S^2), [1,-rho_S], A_vec, (1-sqrt(1-rho_S^2))/
        rho_S*A_vec(1)) + MA_vec(state_ind);
887     alpha_vec = 10.^(A_vec/20);
888
889     % Scattered (diffused) component
890     diff_vec = normrnd(0, 1, 1, curr_state_duration_samples) + 1i * normrnd(0,
        1, 1, curr_state_duration_samples);
891     %diff_vec = Jakes(diff_vec, fD, 1/Ts);
892     diff_vec = conv(diff_vec, Jakes_impulse_response,'same');
893     sigmad = sqrt(0.5 * 10^(MP_vec(state_ind)/10));
894     diff_vec = sigmad * diff_vec;
```

```
895
896     % Superposition of direct and diffused
897     alpha_vec = alpha_vec + diff_vec;
898
899     y(curr_sample + (1:curr_state_duration_samples)) = alpha_vec;
900     curr_sample = curr_sample + curr_state_duration_samples;
901
902     % Definition of the new state
903     state_ind = state_ind + 1;
904     state = 3-state;
905 end
906
907 state_duration_vec = state_duration_vec(1:(state_ind-1));
908 MA_vec = MA_vec(1:(state_ind-1));
909 SigmaA_vec = SigmaA_vec(1:(state_ind-1));
910 MP_vec = MP_vec(1:(state_ind-1));
911 L_trans_vec = L_trans_vec(1:(state_ind-1));
```

### A.1.3 generate_passage_singleorbit.m

```
1  function [elevation_ts, distance_ts, T0] = generate_passage_singleorbit(h0, iK,
       phi0, alpha0, Nsat, lambdaT, etaT, dt, Tmax)
2
3  % INPUT
4  % h0: height of the orbit [km]
5  % iK: orbit inclination [degrees]
6  % phi0: longitude at t=0 of the ascending node of the orbit [degrees]
7  % alpha0: angular position of the 1st satellite at t = 0 [degrees]
8  % Nsat: number of satellites in the orbit
9  % lambdaT: Ground user latitude [degrees]
10 % etaT: Ground user longitude [degrees]
11 % dt: Sampling time [seconds]
12 % Tmax: duration of the time window [seconds]
13
14 % OUTPUT
15 % elevation_ts: matrix that contains in the rows the elevation time series
16 % for all satellites in the orbit [degrees]
17 % distance_ts : matrix that contains in the rows the distance time series
18 % for all satellites in the orbit [km]
19 % T0: orbital period [s]
20
21 mu = 3.986e5; % Gravitational constant [km^3 s^-2]
22 Re = 6371; % Earth radius [km]
23
24 R0 = Re + h0; % Orbit radius [km]
25
26 v0 = sqrt(mu / R0); % Satellite Radial speed [km/s]
27 w0 = v0 / R0; % Satellite Angular speed [rad / s]
28 T0 = 2*pi / w0; % Satellite Orbital period [s]
```

```
wEd = 1 / 24/10; % Earth angular speed [degree/s]

alpha_sep = 2*pi / Nsat;
alpha0_rad = pi / 180 * alpha0;
alpha_0_vec = (0:(Nsat-1)) * alpha_sep + alpha0_rad;
alpha_0_vec = alpha_0_vec(:);

t_vec = 0:dt:Tmax; %Time axis [s]

alpha = w0 * t_vec + alpha_0_vec; % Satellite position in the orbit [rad]
app_vec = sind(iK) * sin(alpha);
theta = acosd(app_vec); % Polar coordinate theta [degrees]
phi_rad = atan2(cosd(iK)*sin(alpha), cos(alpha));  % Polar coordinate phi [
    radiants, wrapped]
% Unwrapping of the polar coordinate phi
phi = 180 / pi * unwrap(phi_rad, [], 2) ; % convert from radiants to degrees
    smoothly with no wraps(errors)

lambdaS = 90 - theta; % Satellite latitude [degrees]
etaS = phi + phi0 - wEd * t_vec; % Satellite longitude [degrees]
etaS = mod(etaS,360);
etaS(etaS>180) = etaS(etaS>180) - 360 ;

% gamma is the central angle between GU and Sub-Satellite Point
gamma = acos(sind(lambdaT) * sind(lambdaS) + cosd(lambdaT) * cosd(lambdaS) .*
    cosd(etaS-etaT));

elevation_ts = atan2d(cos(gamma) - Re / R0, sin(gamma)); % Elevation time
    series [degrees]

distance_ts = sqrt(Re^2 * (sind(elevation_ts)).^2 + h0^2 + ...
    2*h0*Re) - Re * (sind(elevation_ts)); % Distance time series [km]
```

### A.1.4 Jakes_IR.m

```
function impulse_response = Jakes_IR(fD, W)

% input_signal: row vector with the filter input
% fD : Doppler frequency
% W : sampling rate of the input process (typically equal to the signal
% bandwidth =1/Ts*vm

if(W < fD)
    error('The bandwidth must be greater than the doppler frequency')
end

over = round(W/fD);
nsamp = 60;
df = fD * 0.999/nsamp; %frequency interval (avoid singularity)
```

```
delay = nsamp * over;
%dt = 1/delay/df;

f_vec = (0:nsamp) * df;
Tf_1 = 1 ./ sqrt(sqrt(1 - (f_vec / fD).^2));

impulse_response_1 = Tf_1(1) + 2 * cos(2*pi*(0:delay)'* (1:nsamp) / delay) *
    Tf_1(2:end)';
% Windowing
impulse_response_1 = impulse_response_1' .* (1+cos(pi * (0:delay) / delay));
impulse_response = [impulse_response_1(end:-1:2), impulse_response_1];
impulse_response = impulse_response ./ norm(impulse_response);
```

### A.1.5 munkres.m

```
function [assignment,cost] = munkres(costMat)
% MUNKRES   Munkres (Hungarian) Algorithm for Linear Assignment Problem.
%
% [ASSIGN,COST] = munkres(COSTMAT) returns the optimal column indices,
% ASSIGN assigned to each row and the minimum COST based on the assignment
% problem represented by the COSTMAT, where the (i,j)th element represents the
    cost to assign the jth
% job to the ith worker.
%
% Partial assignment: This code can identify a partial assignment is a full
% assignment is not feasible. For a partial assignment, there are some
% zero elements in the returning assignment vector, which indicate
% un-assigned tasks. The cost returned only contains the cost of partially
% assigned tasks.

% This is vectorized implementation of the algorithm. It is the fastest
% among all Matlab implementations of the algorithm.

% Examples
% Example 1: a 5 x 5 example
%{
[assignment,cost] = munkres(magic(5));
disp(assignment); % 3 2 1 5 4
disp(cost); %15
%}
% Example 2: 400 x 400 random data
%{
n=400;
A=rand(n);
tic
[a,b]=munkres(A);
toc                 % about 2 seconds
%}
% Example 3: rectangular assignment with inf costs
```

```
%{
A=rand(10,7);
A(A>0.7)=Inf;
[a,b]=munkres(A);
%}
% Example 4: an example of partial assignment
%{
A = [1 3 Inf; Inf Inf 5; Inf Inf 0.5];
[a,b]=munkres(A)
%}
% a = [1 0 3]
% b = 1.5
% Reference:
% "Munkres' Assignment Algorithm, Modified for Rectangular Matrices",
% http://csclab.murraystate.edu/bob.pilgrim/445/munkres.html

% version 2.3 by Yi Cao at Cranfield University on 11th September 2011

assignment = zeros(1,size(costMat,1));
cost = 0;

validMat = costMat == costMat & costMat < Inf;
bigM = 10^(ceil(log10(sum(costMat(validMat))))+1);
costMat(~validMat) = bigM;

% costMat(costMat~=costMat)=Inf;
% validMat = costMat<Inf;
validCol = any(validMat,1);
validRow = any(validMat,2);

nRows = sum(validRow);
nCols = sum(validCol);
n = max(nRows,nCols);
if ~n
    return
end

maxv=10*max(costMat(validMat));

dMat = zeros(n) + maxv;
dMat(1:nRows,1:nCols) = costMat(validRow,validCol);

%*************************************************
% Munkres' Assignment Algorithm starts here
%*************************************************

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   STEP 1: Subtract the row minimum from each row.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
minR = min(dMat,[],2);
minC = min(bsxfun(@minus, dMat, minR));
```

```
85
86  %*************************************************************************
87  %   STEP 2: Find a zero of dMat. If there are no starred zeros in its
88  %           column or row start the zero. Repeat for each zero
89  %*************************************************************************
90  zP = dMat == bsxfun(@plus, minC, minR);
91
92  starZ = zeros(n,1);
93  while any(zP(:))
94      [r,c]=find(zP,1);
95      starZ(r)=c;
96      zP(r,:)=false;
97      zP(:,c)=false;
98  end
99
100 while 1
101 %*************************************************************************
102 %   STEP 3: Cover each column with a starred zero. If all the columns are
103 %           covered then the matching is maximum
104 %*************************************************************************
105     if all(starZ>0)
106         break
107     end
108     coverColumn = false(1,n);
109     coverColumn(starZ(starZ>0))=true;
110     coverRow = false(n,1);
111     primeZ = zeros(n,1);
112     [rIdx, cIdx] = find(dMat(~coverRow,~coverColumn)==bsxfun(@plus,minR(~
    coverRow),minC(~coverColumn)));
113     while 1
114         %
    *************************************************************************
115         %   STEP 4: Find a noncovered zero and prime it.  If there is no
    starred
116         %           zero in the row containing this primed zero, Go to Step 5.
117         %           Otherwise, cover this row and uncover the column containing
118         %           the starred zero. Continue in this manner until there are
    no
119         %           uncovered zeros left. Save the smallest uncovered value and
120         %           Go to Step 6.
121         %
    *************************************************************************
122         cR = find(~coverRow);
123         cC = find(~coverColumn);
124         rIdx = cR(rIdx);
125         cIdx = cC(cIdx);
126         Step = 6;
127         while ~isempty(cIdx)
128             uZr = rIdx(1);
129             uZc = cIdx(1);
130             primeZ(uZr) = uZc;
```

```
            stz = starZ(uZr);
            if ~stz
                Step = 5;
                break;
            end
            coverRow(uZr) = true;
            coverColumn(stz) = false;
            z = rIdx==uZr;
            rIdx(z) = [];
            cIdx(z) = [];
            cR = find(~coverRow);
            z = dMat(~coverRow,stz) == minR(~coverRow) + minC(stz);
            rIdx = [rIdx(:);cR(z)];
            cIdx = [cIdx(:);stz(ones(sum(z),1))];
        end
        if Step == 6
            %
***************************************************************************
            % STEP 6: Add the minimum uncovered value to every element of each
covered
            %         row, and subtract it from every element of each uncovered
 column.
            %         Return to Step 4 without altering any stars, primes, or
covered lines.
            %
***************************************************************************
            [minval,rIdx,cIdx]=outerplus(dMat(~coverRow,~coverColumn),minR(~
coverRow),minC(~coverColumn));
            minC(~coverColumn) = minC(~coverColumn) + minval;
            minR(coverRow) = minR(coverRow) - minval;
        else
            break
        end
    end
    %***************************************************************************
    % STEP 5:
    %  Construct a series of alternating primed and starred zeros as
    %  follows:
    %  Let Z0 represent the uncovered primed zero found in Step 4.
    %  Let Z1 denote the starred zero in the column of Z0 (if any).
    %  Let Z2 denote the primed zero in the row of Z1 (there will always
    %  be one).  Continue until the series terminates at a primed zero
    %  that has no starred zero in its column.  Unstar each starred
    %  zero of the series, star each primed zero of the series, erase
    %  all primes and uncover every line in the matrix.  Return to Step 3.
    %***************************************************************************
    rowZ1 = find(starZ==uZc);
    starZ(uZr)=uZc;
    while rowZ1>0
        starZ(rowZ1)=0;
        uZc = primeZ(rowZ1);
```

```
        uZr = rowZ1;
        rowZ1 = find(starZ==uZc);
        starZ(uZr)=uZc;
    end
end

% Cost of assignment
rowIdx = find(validRow);
colIdx = find(validCol);
starZ = starZ(1:nRows);
vIdx = starZ <= nCols;
assignment(rowIdx(vIdx)) = colIdx(starZ(vIdx));
pass = assignment(assignment>0);
pass(~diag(validMat(assignment>0,pass))) = 0;
assignment(assignment>0) = pass;
cost = trace(costMat(assignment>0,assignment(assignment>0)));

function [minval,rIdx,cIdx]=outerplus(M,x,y)
ny=size(M,2);
minval=inf;
for c=1:ny
    M(:,c)=M(:,c)-(x+y(c));
    minval = min(minval,min(M(:,c)));
end
[rIdx,cIdx]=find(M==minval);
```

## A.2 Algorithm 1: Simultaneous Handover Assignment (SiHA)

### A.2.1 assign_users_munkres_algo_1.m

```
function [assignments_primary, assignments_dual, cost_primary, cost_dual] =
    assign_users_munkres_algo_1( ...
    W, prev_assign, prev_assign_dual, G, ...
    alpha, beta, threshold, low_quality_threshold, ...
    first_assignment, use_dual)
% assign_users_munkres
% Unified assignment function with dual-handover support and cost breakdown.
%
% Inputs:
%   W                   - (U x M) metric matrix (SNR [dB] or rate)
%   prev_assign         - (U x 1) previous primary assignment
%   prev_assign_dual    - (U x 1) previous dual assignment
%   G                   - max users per satellite
%   alpha               - penalty for switching primary
%   beta                - penalty for switching dual
%   threshold           - minimum link value for repair
```

```
%   low_quality_threshold- quality below which dual reassignment is triggered
%   first_assignment     - true at t = 1 (disables penalties)
%   use_dual             - enable dual-handover step
%
% Outputs:
%   assignments_primary  - (U x 1) initial primary assignment before repair
%   assignments_dual     - (U x 1) repair assignment only (0 if unchanged)
%   cost_primary         - primary assignment cost
%   cost_dual            - dual handover cost

[U, M] = size(W);
assignments_primary = zeros(U, 1);
assignments_dual = zeros(U, 1);
cost_primary = 0;
cost_dual = 0;

%%Step 1: Apply primary handover penalties
W_penalized = W;
if ~first_assignment
    for u = 1:U
        if prev_assign(u) > 0 && prev_assign(u) <= M
            W_penalized(u, prev_assign(u)) = W_penalized(u, prev_assign(u)) -
    alpha;
        end
    end
end

%%Step 2: Filter valid users and satellites
valid_users = any(W_penalized, 2);
valid_sats  = any(W_penalized, 1);
if ~any(valid_users) || ~any(valid_sats)
    return;
end

user_idx_map = find(valid_users);
sat_idx_map  = find(valid_sats);
W_small = W_penalized(valid_users, valid_sats);

%%Step 3: Expand satellite capacity for Munkres
W_expanded = repmat(W_small, 1, G);  % (U' x M'*G)

%%Step 4: Primary assignment using Hungarian algorithm
[raw_assignment, raw_cost] = munkres(-W_expanded);
slot_idx = ceil(raw_assignment / G);
assignments_reduced = zeros(length(user_idx_map), 1);
valid_idx = raw_assignment > 0;
assignments_reduced(valid_idx) = sat_idx_map(slot_idx(valid_idx));
assignments_primary(user_idx_map) = assignments_reduced;

% Record cost from primary assignment
cost_primary = -raw_cost;
```

```
cost_total = cost_primary;

%%Step 5: Dual-handover repair (optional)
if ~use_dual
    return;
end

% Step 5.1: Identify users with poor or no assignment
assigned_metrics = zeros(U, 1);
assigned = assignments_primary > 0;
assigned_metrics(assigned) = W(sub2ind(size(W), find(assigned),
    assignments_primary(assigned)));
repair_users = find(assigned_metrics < low_quality_threshold);
if isempty(repair_users)
    return;
end

% Step 5.2: Find satellites with available slots
sat_load = histcounts(assignments_primary(assignments_primary > 0), 1:M+1);
available_slots = G - sat_load;
available_sats = find(available_slots > 0);
if isempty(available_sats)
    return;
end

% Step 5.3: Build repair matrix
W_repair = W(repair_users, available_sats);
W_repair(W_repair < threshold) = 0;

% Remove users who have no valid satellite options in W_repair
valid_users_dual = any(W_repair, 2);
repair_users = repair_users(valid_users_dual);
W_repair = W_repair(valid_users_dual, :);

% Remove satellites (columns) that have no value left
valid_sats_dual = any(W_repair, 1);
available_sats = available_sats(valid_sats_dual);
W_repair = W_repair(:, valid_sats_dual);

% Re-check size
if isempty(repair_users) || isempty(available_sats)
    return;
end

% Step 5.4: Apply dual-handover penalties
if ~first_assignment
    for i = 1:length(repair_users)
        u = repair_users(i);
        prev_dual = prev_assign_dual(u);
        if prev_dual > 0
            local_idx = find(available_sats == prev_dual);
```

```
            if ~isempty(local_idx)
                W_repair(i, local_idx) = W_repair(i, local_idx) - beta;
            end
        end
    end
end

% Step 5.5: Expand for remaining satellite capacity
sat_repeat = arrayfun(@(s) repmat(s, 1, available_slots(s)), ...
                      available_sats, 'UniformOutput', false);
sat_repeat = [sat_repeat{:}];

W_expanded_repair = [];
for i = 1:length(available_sats)
    col = W_repair(:, i);
    reps = available_slots(available_sats(i));
    W_expanded_repair = [W_expanded_repair, repmat(col, 1, reps)];
end

% Step 5.6: Run Munkres again for repair users
[repair_assignment, cost_dual_raw] = munkres(-W_expanded_repair);
valid_repair = repair_assignment > 0;
repair_slots = repair_assignment(valid_repair);
repair_users_final = repair_users(valid_repair);
repair_sats = sat_repeat(repair_slots);

% Update dual and final assignments
assignments_dual(repair_users_final) = repair_sats;

% Compute cost: consistent with penalized assignment
cost_dual = -cost_dual_raw;

end
```

### A.2.2 simulate_ideal_mode_algo_1.m

```
function [actual_user_SNR, actual_user_rate, actual_user_SNR_primary,
    actual_user_SNR_dual, actual_user_rate_primary, actual_user_rate_dual] =
    ...
    simulate_ideal_mode_algo_1(W_SNR_ideal, W_Rate_ideal,
    elevation_quantized_all, distance_ts_all, channel_trace_map, allowed_elevs,
    f_c, Ps)

U = size(elevation_quantized_all, 3);
T = size(elevation_quantized_all, 2);

G = 4; alpha = 0.2; beta = 0.1;
snr_threshold = -5; rate_threshold = 0.05;
low_quality_threshold_snr = 2;
```

```
low_quality_threshold_rate = 0.5;

actual_user_SNR         = zeros(U, T);
actual_user_rate        = zeros(U, T);
actual_user_SNR_primary = zeros(U, T);
actual_user_SNR_dual    = zeros(U, T);
actual_user_rate_primary= zeros(U, T);
actual_user_rate_dual   = zeros(U, T);

prev_assign_snr        = zeros(U, 1);
prev_assign_snr_dual   = zeros(U, 1);
prev_assign_rate       = zeros(U, 1);
prev_assign_rate_dual  = zeros(U, 1);

for t = 1:T
    W_snr_ideal  = squeeze(W_SNR_ideal(:, :, t));
    W_rate_ideal = squeeze(W_Rate_ideal(:, :, t));

    if ~any(W_snr_ideal(:)) && ~any(W_rate_ideal(:)), continue; end

    [assign_snr_primary, assign_snr_dual, ~, ~] = assign_users_munkres_algo_1(
    W_snr_ideal, ...
        prev_assign_snr, prev_assign_snr_dual, G, alpha, beta, ...
        snr_threshold, low_quality_threshold_snr, t == 1, true);

    [assign_rate_primary, assign_rate_dual, ~, ~] = assign_users_munkres_algo_1
    (W_rate_ideal, ...
        prev_assign_rate, prev_assign_rate_dual, G, alpha, beta, ...
        rate_threshold, low_quality_threshold_rate, t == 1, true);

    prev_assign_snr      = assign_snr_primary;
    prev_assign_snr_dual = assign_snr_dual;
    prev_assign_rate     = assign_rate_primary;
    prev_assign_rate_dual= assign_rate_dual;

    for u = 1:U
        v1 = 0; v2 = 0;
        if assign_snr_primary(u) > 0
            v1 = W_snr_ideal(u, assign_snr_primary(u));
        end
        if assign_snr_dual(u) > 0
            v2 = W_snr_ideal(u, assign_snr_dual(u));
        end
        actual_user_SNR_primary(u, t) = v1;
        actual_user_SNR_dual(u, t)    = v2;

        % Correct SNR Combination: Linear Sum, then dB
        vals = [v1 v2];
        vals_linear = 10.^(vals(~isnan(vals))/10);
        if isempty(vals_linear)
            actual_user_SNR(u, t) = NaN;    % No assignment at all
```

```
        else
            actual_user_SNR(u, t) = 10*log10(sum(vals_linear));
        end

        % Rate: primary and dual assignments
        r1 = 0; r2 = 0;
        if assign_rate_primary(u) > 0
            r1 = W_rate_ideal(u, assign_rate_primary(u));
        end
        if assign_rate_dual(u) > 0
            r2 = W_rate_ideal(u, assign_rate_dual(u));
        end
        actual_user_rate_primary(u, t) = r1;
        actual_user_rate_dual(u, t)    = r2;
        actual_user_rate(u, t)         = r1 + r2;
    end
end
end
```

### A.2.3 simulate_forecast_mode_algo_1.m

```
function [actual_user_SNR, actual_user_rate, ...
          actual_user_SNR_primary, actual_user_SNR_dual, ...
          actual_user_rate_primary, actual_user_rate_dual] = ...
    simulate_forecast_mode_algo_1(W_SNR_forecast, W_Rate_forecast, W_SNR_ideal,
    W_Rate_ideal, ...
                           elevation_quantized_all, distance_ts_all,
   channel_trace_map, allowed_elevs, f_c, Ps)

U = size(elevation_quantized_all, 3);
T = size(elevation_quantized_all, 2);
M = size(elevation_quantized_all, 1);

G = 4; alpha = 0.2; beta = 0.1;
snr_threshold = -5; rate_threshold = 0.05;
low_quality_threshold_snr = 2;
low_quality_threshold_rate = 0.5;

actual_user_SNR         = zeros(U, T);
actual_user_rate        = zeros(U, T);
actual_user_SNR_primary = zeros(U, T);
actual_user_SNR_dual    = zeros(U, T);
actual_user_rate_primary= zeros(U, T);
actual_user_rate_dual   = zeros(U, T);

prev_assign_snr        = zeros(U, 1);
prev_assign_snr_dual   = zeros(U, 1);
prev_assign_rate       = zeros(U, 1);
prev_assign_rate_dual  = zeros(U, 1);
```

```
27
28  for t = 1:T
29      W_snr_forecast = W_SNR_forecast(:, :, t);    % [U x M]
30      W_rate_forecast = W_Rate_forecast(:, :, t);  % [U x M]
31      W_snr_ideal = W_SNR_ideal(:, :, t);          % [U x M]
32      W_rate_ideal = W_Rate_ideal(:, :, t);        % [U x M]
33
34      if ~any(W_snr_forecast(:)) && ~any(W_rate_forecast(:)), continue; end
35
36      [assign_snr_primary, assign_snr_dual, ~, ~] = assign_users_munkres_algo_1(
        W_snr_forecast, ...
37          prev_assign_snr, prev_assign_snr_dual, G, alpha, beta, ...
38          snr_threshold, low_quality_threshold_snr, t == 1, true);
39
40      [assign_rate_primary, assign_rate_dual, ~, ~] = assign_users_munkres_algo_1
        (W_rate_forecast, ...
41          prev_assign_rate, prev_assign_rate_dual, G, alpha, beta, ...
42          rate_threshold, low_quality_threshold_rate, t == 1, true);
43
44      prev_assign_snr      = assign_snr_primary;
45      prev_assign_snr_dual = assign_snr_dual;
46      prev_assign_rate     = assign_rate_primary;
47      prev_assign_rate_dual= assign_rate_dual;
48
49      for u = 1:U
50          v1 = 0; v2 = 0;
51          % SNR Primary
52          if assign_snr_primary(u) > 0
53              v1 = W_snr_ideal(u, assign_snr_primary(u));
54          end
55
56          % SNR Dual
57          if assign_snr_dual(u) > 0
58              v2 = W_snr_ideal(u, assign_snr_dual(u));
59          end
60
61          actual_user_SNR_primary(u, t) = v1;
62          actual_user_SNR_dual(u, t)    = v2;
63
64          vals = [v1 v2];
65          valid = ~isnan(vals) & (vals > 0);
66          if any(valid)
67              actual_user_SNR(u, t) = 10*log10(sum(10.^(vals(valid)/10)));
68          else
69              actual_user_SNR(u, t) = 0;
70          end
71
72          r1 = 0; r2 = 0;
73          % Rate Primary
74          if assign_rate_primary(u) > 0
75              r1 = W_rate_ideal(u, assign_rate_primary(u));
```

```
        end
        % Rate Dual
        if assign_rate_dual(u) > 0
            r2 = W_rate_ideal(u, assign_rate_dual(u));
        end

        actual_user_rate_primary(u, t) = r1;
        actual_user_rate_dual(u, t)    = r2;
        actual_user_rate(u, t)         = r1 + r2;
    end
end
end
```

### A.2.4 siso_algo_1_comparison.m

```
clear all
close all
clc

% --- System Parameters ---
h0 = 550; iK = 53; Nsat = 22; Norbits = 72; U = 50; dt = 10; Tmax = 3600;
deltaPhi = 30; M = Nsat * Norbits; T = Tmax / dt + 1;
user_latitudes = 45 + (50 - 45) * rand(U, 1);
user_longitudes = 55 + (60 - 55) * rand(U, 1);

elevation_ts_all = zeros(M, T, U); distance_ts_all = zeros(M, T, U);
for u = 1:U
    for ind = 1:Norbits
        base_idx = (ind - 1) * Nsat + 1;
        [elev_ts, dist_ts, ~] = generate_passage_singleorbit(h0, iK, ...
            (ind - 1) * deltaPhi, -(ind - 1) * 360 / Nsat / Norbits, ...
            Nsat, user_latitudes(u), user_longitudes(u), dt, Tmax);
        elevation_ts_all(base_idx:base_idx + Nsat - 1, :, u) = elev_ts;
        distance_ts_all(base_idx:base_idx + Nsat - 1, :, u) = dist_ts;
    end
end

% --- Elevation Quantization ---
allowed_elevs = [30, 45, 60, 70]; % not using 20 since the threshold is 25
partition = [25, 37.5, 52.5, 65, inf];
elevation_quantized_all = NaN(size(elevation_ts_all));
for u = 1:U
    quantized_indices = discretize(elevation_ts_all(:, :, u), partition);
    temp = NaN(size(elevation_ts_all(:, :, u)));
    valid_idx = ~isnan(quantized_indices);
    temp(valid_idx) = allowed_elevs(quantized_indices(valid_idx));
    temp(elevation_ts_all(:, :, u) < 5) = NaN;
    elevation_quantized_all(:, :, u) = temp;
end
```

```
35
36 % --- Channel Fading Traces ---
37 f_c = 2e9; envi = 'Suburban'; Tch = 5000;
38 channel_trace_map = containers.Map('KeyType', 'double', 'ValueType', 'any');
39 for k = 1:length(allowed_elevs)
40     elev = allowed_elevs(k);
41     try
42         [y, ~, ~, ~, ~, ~] = coef_time_series_long_pedestrian(f_c, envi, elev,
       Tch);
43         channel_trace_map(elev) = abs(y).^2;  % Only average used
44     catch
45         channel_trace_map(elev) = zeros(1, Tch);
46     end
47 end
48
49 % --- Transmit power values (in Watts) ---
50 Ps_values = [0.1 0.2 0.5 1 1.5 2];
51
52 % Preallocate results for plotting and analysis
53 avg_throughput_ideal    = zeros(length(Ps_values), 1);
54 avg_throughput_forecast = zeros(length(Ps_values), 1);
55 avg_snr_ideal           = zeros(length(Ps_values), 1);
56 avg_snr_forecast        = zeros(length(Ps_values), 1);
57
58 results_ideal    = cell(length(Ps_values), 1);
59 results_forecast = cell(length(Ps_values), 1);
60
61 sum_snr_ideal     = cell(length(Ps_values), 1);
62 sum_snr_forecast  = cell(length(Ps_values), 1);
63 sum_rate_ideal    = cell(length(Ps_values), 1);
64 sum_rate_forecast = cell(length(Ps_values), 1);
65
66 for i = 1:length(Ps_values)
67     Ps = Ps_values(i);
68
69     % --- Building the weight matrices ---
70     [W_SNR_ideal, W_Rate_ideal, W_SNR_forecast, W_Rate_forecast] = ...
71     build_weight_matrices_AT(Ps, channel_trace_map, elevation_quantized_all,
       distance_ts_all, allowed_elevs, f_c);
72
73     % --- Ideal Mode ---
74     [snr_ideal, rate_ideal, snr_ideal_primary, snr_ideal_dual,
       rate_ideal_primary, rate_ideal_dual] = ...
75         simulate_ideal_mode_algo_1(W_SNR_ideal, W_Rate_ideal,
       elevation_quantized_all, distance_ts_all, channel_trace_map, allowed_elevs,
        f_c, Ps);
76     avg_throughput_ideal(i) = mean(rate_ideal(:),'omitnan');
77     avg_snr_ideal(i)        = mean(snr_ideal(:),'omitnan');
78     results_ideal{i} = struct('snr', snr_ideal, 'rate', rate_ideal, ...
79         'snr_primary', snr_ideal_primary, 'snr_dual', snr_ideal_dual, ...
80         'rate_primary', rate_ideal_primary, 'rate_dual', rate_ideal_dual);
```

```
    sum_snr_ideal{i}  = sum(snr_ideal, 1, 'omitnan');   % [1 x T], sum over users
    sum_rate_ideal{i} = sum(rate_ideal, 1, 'omitnan');  % [1 x T], sum over users

    % --- Forecast Mode ---
    [snr_forecast, rate_forecast, snr_forecast_primary, snr_forecast_dual, rate_forecast_primary, rate_forecast_dual] = ...
    simulate_forecast_mode_algo_1(W_SNR_forecast, W_Rate_forecast, W_SNR_ideal, W_Rate_ideal, elevation_quantized_all, distance_ts_all, channel_trace_map, allowed_elevs, f_c, Ps);
    avg_throughput_forecast(i) = mean(rate_forecast(:),'omitnan');
    avg_snr_forecast(i)        = mean(snr_forecast(:),'omitnan');
    results_forecast{i} = struct('snr', snr_forecast, 'rate', rate_forecast, ...
        'snr_primary', snr_forecast_primary, 'snr_dual', snr_forecast_dual, ...
        'rate_primary', rate_forecast_primary, 'rate_dual', rate_forecast_dual);
    sum_snr_forecast{i}  = sum(snr_forecast, 1, 'omitnan');   % [1 x T]
    sum_rate_forecast{i} = sum(rate_forecast, 1, 'omitnan');  % [1 x T]
end

% --- Plotting Section:

output_folder = fullfile(pwd, 'Plots_LEO_Handover_algo_1');
if ~exist(output_folder, 'dir')
    mkdir(output_folder);
end
save_plot = @(name) saveas(gcf, fullfile(output_folder, name));

% Convert Ps to dBm for plotting
Ps_dBm = 10 * log10(Ps_values * 1000);

i_plot = 3; user_idx = 1;
d_ideal = results_ideal{i_plot};
d_forecast = results_forecast{i_plot};

%%1) Average Throughput vs Transmit Power (in dBm)
figure;
plot(Ps_dBm, avg_throughput_ideal, 'b-o', 'LineWidth', 1.6); hold on;
plot(Ps_dBm, avg_throughput_forecast, 'r--s', 'LineWidth', 1.6);
xlabel('Transmit Power (dBm)', 'FontWeight', 'bold');
ylabel('Average Throughput (bps/Hz)', 'FontWeight', 'bold');
title({'Throughput vs Transmit Power', ...
       'U = 50 users, T = 361 time steps (10 s each), Suburban Environment'}, ...
       'FontWeight', 'bold');
legend('Ideal Mode', 'Forecast Mode', 'Location', 'best');
grid on;
axis tight; ylim padded;
save_plot('1_Throughput_vs_Ps_1.png');
```

```
124
125 %%2) Average SNR vs Transmit Power (in dBm)
126 figure;
127 plot(Ps_dBm, avg_snr_ideal, 'b-o', 'LineWidth', 1.6); hold on;
128 plot(Ps_dBm, avg_snr_forecast, 'r--s', 'LineWidth', 1.6);
129 xlabel('Transmit Power (dBm)', 'FontWeight', 'bold');
130 ylabel('Average SNR (dB)', 'FontWeight', 'bold');
131 title({'SNR vs Transmit Power', ...
132        'U = 50 users, T = 361 time steps (10 s each), Suburban Environment'},
    ...
133        'FontWeight', 'bold');
134 legend('Ideal Mode', 'Forecast Mode', 'Location', 'best');
135 grid on;
136 axis tight; ylim padded;
137 save_plot('2_SNR_vs_Ps_1.png');
138
139 %%3) System Sum Rate vs Time
140 figure;
141 plot(sum_rate_ideal{i_plot}, 'b-', 'LineWidth', 2); hold on;
142 plot(sum_rate_forecast{i_plot}, 'r--', 'LineWidth', 2);
143 xlabel('Time Step (10 s)', 'FontWeight', 'bold');
144 ylabel('Total System Rate (bps/Hz)', 'FontWeight', 'bold');
145 legend('Ideal', 'Forecast', 'Location', 'best');
146 title('Total System Rate per Time Step', 'FontWeight', 'bold');
147 grid on;
148 axis tight; ylim padded;
149 save_plot('3_System_Sum_Rate_vs_Time_1.png');
150
151 %%4) System Sum SNR vs Time
152 figure;
153 plot(sum_snr_ideal{i_plot}, 'b-', 'LineWidth', 2); hold on;
154 plot(sum_snr_forecast{i_plot}, 'r--', 'LineWidth', 2);
155 xlabel('Time Step (10 s)', 'FontWeight', 'bold');
156 ylabel('Total System SNR (dB)', 'FontWeight', 'bold');
157 legend('Ideal', 'Forecast', 'Location', 'best');
158 title('Total System SNR per Time Step', 'FontWeight', 'bold');
159 grid on;
160 axis tight; ylim padded;
161 save_plot('4_System_Sum_SNR_vs_Time_1.png');
162
163 %%5) Single-User Rate Evolution
164 figure;
165 plot(d_ideal.rate(user_idx,:), 'b-', 'LineWidth', 2); hold on;
166 plot(d_forecast.rate(user_idx,:), 'r--', 'LineWidth', 2);
167 xlabel('Time Step (10 s)', 'FontWeight', 'bold');
168 ylabel('Rate (bps/Hz)', 'FontWeight', 'bold');
169 legend('Ideal', 'Forecast', 'Location', 'best');
170 title(sprintf('Rate Evolution - User %d', user_idx), 'FontWeight', 'bold');
171 grid on;
172 axis tight; ylim padded;
173 save_plot(sprintf('5_SingleUser_Rate_User%d_1.png', user_idx));
```

```
174
175 %%6) Single-User SNR Evolution
176 figure;
177 plot(d_ideal.snr(user_idx,:), 'b-', 'LineWidth', 2); hold on;
178 plot(d_forecast.snr(user_idx,:), 'r--', 'LineWidth', 2);
179 xlabel('Time Step (10 s)', 'FontWeight', 'bold');
180 ylabel('SNR (dB)', 'FontWeight', 'bold');
181 legend('Ideal', 'Forecast', 'Location', 'best');
182 title(sprintf('SNR Evolution - User %d', user_idx), 'FontWeight', 'bold');
183 grid on;
184 axis tight; ylim padded;
185 save_plot(sprintf('6_SingleUser_SNR_User%d_1.png', user_idx));
186
187 %%7) System Primary vs Dual Sums (Ideal)
188 figure;
189 plot(sum(d_ideal.rate_primary, 1, 'omitnan'), 'b-', 'LineWidth', 1.8); hold on;
190 plot(sum(d_ideal.rate_dual,    1, 'omitnan'), 'r--', 'LineWidth', 1.8);
191 plot(sum(d_ideal.rate,         1, 'omitnan'), 'k:', 'LineWidth', 2.2);
192 xlabel('Time Step (10 s)', 'FontWeight', 'bold');
193 ylabel('Sum Rate (bps/Hz)', 'FontWeight', 'bold');
194 legend('Primary', 'Dual', 'Total', 'Location', 'best');
195 title('Ideal: System Primary, Dual, Total Rate', 'FontWeight', 'bold');
196 grid on;
197 axis tight; ylim padded;
198 save_plot('7_System_Primary_Dual_Total_Rate_Ideal_1.png');
199
200 figure;
201 plot(sum(d_ideal.snr_primary, 1, 'omitnan'), 'b-', 'LineWidth', 1.8); hold on;
202 plot(sum(d_ideal.snr_dual,    1, 'omitnan'), 'r--', 'LineWidth', 1.8);
203 plot(sum(d_ideal.snr,         1, 'omitnan'), 'k:', 'LineWidth', 2.2);
204 xlabel('Time Step (10 s)', 'FontWeight', 'bold');
205 ylabel('Sum SNR (dB)', 'FontWeight', 'bold');
206 legend('Primary', 'Dual', 'Total', 'Location', 'best');
207 title('Ideal: System Primary, Dual, Total SNR', 'FontWeight', 'bold');
208 grid on;
209 axis tight; ylim padded;
210 save_plot('8_System_Primary_Dual_Total_SNR_Ideal_1.png');
211
212 %%8) CDF of Average User Rate
213 user_rate_mean_ideal    = mean(d_ideal.rate,    2, 'omitnan');
214 user_rate_mean_forecast = mean(d_forecast.rate, 2, 'omitnan');
215 valid_ideal             = user_rate_mean_ideal(~isnan(user_rate_mean_ideal));
216 valid_forecast          = user_rate_mean_forecast(~isnan(
    user_rate_mean_forecast));
217 if isempty(valid_ideal) || isempty(valid_forecast)
218     warning('No valid user averages for RATE CDF plot.');
219 else
220     figure;
221     cdfplot(valid_ideal); hold on;
222     cdfplot(valid_forecast);
223     legend('Ideal', 'Forecast', 'Location', 'best');
```

```
    xlabel('Average Rate per User (bps/Hz)', 'FontWeight', 'bold');
    ylabel('Cumulative Probability', 'FontWeight', 'bold');
    title('CDF of Average User Rate', 'FontWeight', 'bold');
    grid on;
    axis tight; xlim padded;
    save_plot('9_CDF_AvgUserRate_1.png');
end

%%9) CDF of Average User SNR
user_snr_mean_ideal    = mean(d_ideal.snr,    2, 'omitnan');
user_snr_mean_forecast = mean(d_forecast.snr, 2, 'omitnan');
valid_ideal_snr        = user_snr_mean_ideal(~isnan(user_snr_mean_ideal));
valid_forecast_snr     = user_snr_mean_forecast(~isnan(user_snr_mean_forecast))
    ;
if isempty(valid_ideal_snr) || isempty(valid_forecast_snr)
    warning('No valid user averages for SNR CDF plot.');
else
    figure;
    cdfplot(valid_ideal_snr); hold on;
    cdfplot(valid_forecast_snr);
    legend('Ideal', 'Forecast', 'Location', 'best');
    xlabel('Average SNR per User (dB)', 'FontWeight', 'bold');
    ylabel('Cumulative Probability', 'FontWeight', 'bold');
    title('CDF of Average User SNR', 'FontWeight', 'bold');
    grid on;
    axis tight; xlim padded;
    save_plot('10_CDF_AvgUserSNR_1.png');
end
```

## A.3 Algorithm 2: Staggered Handover Assignment (StHA)

### A.3.1 assign_users_munkres_algo_2.m

```
function [assignments_primary, assignments_dual, cost_primary, cost_dual] =
    assign_users_munkres_new( ...
    W, prev_assign, prev_assign_dual, G, ...
    alpha, beta, threshold, low_quality_threshold, ...
    first_assignment, use_dual)
% assign_users_munkres (fully modular, capacity enforced for both primary and
    dual)
%
% Inputs:
%   W                   - (U x M) metric matrix (SNR [dB] or rate)
%   prev_assign         - (U x 1) previous primary assignment
%   prev_assign_dual    - (U x 1) previous dual assignment
%   G                   - max users per satellite
```

```
%   alpha, beta          - switching penalties
%   threshold, low_quality_threshold - for dual repair
%   first_assignment     - true at t=1 (no penalty)
%   use_dual             - false: do primary only, true: do dual only
%
% Outputs:
%   assignments_primary  - (U x 1) new primary (if primary step)
%   assignments_dual     - (U x 1) new dual (if dual step)
%   cost_primary         - primary assignment cost
%   cost_dual            - dual handover cost

[U, M] = size(W);
assignments_primary = prev_assign;     % Default: unchanged
assignments_dual   = prev_assign_dual; % Default: unchanged
cost_primary = 0;
cost_dual = 0;

if ~use_dual
    % ======================
    % === Primary only! ====
    % ======================
    W_penalized = W;
    if ~first_assignment
        for u = 1:U
            if prev_assign(u) > 0 && prev_assign(u) <= M
                W_penalized(u, prev_assign(u)) = W_penalized(u, prev_assign(u))
    - alpha;
            end
        end
    end

    % --- Enforce satellite capacity (count both primary and dual assignments)
    ---
    all_assigned = [prev_assign(:); prev_assign_dual(:)];
    sat_load = histcounts(all_assigned(all_assigned > 0), 1:M+1);
    available_slots = G - sat_load;
    for m = 1:M
        if available_slots(m) <= 0
            W_penalized(:, m) = 0; % Block assignments to full satellites
        end
    end

    valid_users = any(W_penalized, 2);
    valid_sats  = any(W_penalized, 1);
    if ~any(valid_users) || ~any(valid_sats)
        assignments_primary = zeros(U, 1);
        return;
    end

    user_idx_map = find(valid_users);
    sat_idx_map  = find(valid_sats);
```

```
    W_small = W_penalized(valid_users, valid_sats);

    % Expand satellite capacity for Munkres (based on actual available slots)
    W_expanded = [];
    sat_idx_expanded = [];
    for i = 1:length(sat_idx_map)
        m = sat_idx_map(i);
        n_slots = available_slots(m);
        if n_slots > 0
            W_expanded = [W_expanded, repmat(W_small(:,i), 1, n_slots)];
            sat_idx_expanded = [sat_idx_expanded, repmat(m, 1, n_slots)];
        end
    end

    % Primary assignment using Hungarian algorithm
    [raw_assignment, raw_cost] = munkres(-W_expanded);

    assignments_reduced = zeros(length(user_idx_map), 1);
    valid_idx = raw_assignment > 0;
    assignments_reduced(valid_idx) = sat_idx_expanded(raw_assignment(valid_idx)
    );

    assignments_primary = zeros(U, 1);
    assignments_primary(user_idx_map) = assignments_reduced;
    cost_primary = -raw_cost;

    % Dual remains as input (unchanged)
    assignments_dual = prev_assign_dual;
    cost_dual = 0;
    return;
end

% ======================
% ==== Dual only! ======
% ======================
% Use prev_assign as the current primary assignment

% 1. Identify users needing repair (poor or no assignment)
assigned_metrics = zeros(U, 1);
assigned = prev_assign > 0;
assigned_metrics(assigned) = W(sub2ind(size(W), find(assigned), prev_assign(
    assigned)));
repair_users = find(assigned_metrics < low_quality_threshold);

assignments_primary = prev_assign;      % (carry over)
assignments_dual = prev_assign_dual;   % <-- HOLD previous dual assignments!
cost_primary = 0;                      % not updated in dual mode

if isempty(repair_users)
    return;
end
```

```
110
111
112 % 2. Satellites with available slots (count both primary and dual)
113 all_assigned = [prev_assign(:); prev_assign_dual(:)];
114 sat_load = histcounts(all_assigned(all_assigned > 0), 1:M+1);
115 available_slots = G - sat_load;
116 available_sats = find(available_slots > 0);
117 if isempty(available_sats)
118     return;
119 end
120
121 % 3. Build dual repair matrix
122 W_repair = W(repair_users, available_sats);
123 W_repair(W_repair < threshold) = 0;
124 valid_users_dual = any(W_repair, 2);
125 repair_users = repair_users(valid_users_dual);
126 W_repair = W_repair(valid_users_dual, :);
127 valid_sats_dual = any(W_repair, 1);
128 available_sats = available_sats(valid_sats_dual);
129 W_repair = W_repair(:, valid_sats_dual);
130
131 if isempty(repair_users) || isempty(available_sats)
132     return;
133 end
134
135 % 4. Dual penalties
136 if ~first_assignment
137     for i = 1:length(repair_users)
138         u = repair_users(i);
139         prev_dual = prev_assign_dual(u);
140         if prev_dual > 0
141             local_idx = find(available_sats == prev_dual);
142             if ~isempty(local_idx)
143                 W_repair(i, local_idx) = W_repair(i, local_idx) - beta;
144             end
145         end
146     end
147 end
148
149 % 5. Satellite capacity expansion for dual
150 sat_repeat = arrayfun(@(s) repmat(s, 1, available_slots(s)), ...
151                       available_sats, 'UniformOutput', false);
152 sat_repeat = [sat_repeat{:}];
153 W_expanded_repair = [];
154 for i = 1:length(available_sats)
155     col = W_repair(:, i);
156     reps = available_slots(available_sats(i));
157     W_expanded_repair = [W_expanded_repair, repmat(col, 1, reps)];
158 end
159
160 % 6. Munkres for dual
```

```
[repair_assignment, cost_dual_raw] = munkres(-W_expanded_repair);
valid_repair = repair_assignment > 0;
repair_slots = repair_assignment(valid_repair);
repair_users_final = repair_users(valid_repair);
repair_sats = sat_repeat(repair_slots);

assignments_dual(repair_users_final) = repair_sats;
cost_dual = -cost_dual_raw;

end
```

### A.3.2 simulate_ideal_mode_algo_2.m

```
function [actual_user_SNR, actual_user_rate, actual_user_SNR_primary,
    actual_user_SNR_dual, actual_user_rate_primary, actual_user_rate_dual] =
    ...
     simulate_ideal_mode_algo_2(W_SNR_ideal, W_Rate_ideal,
    elevation_quantized_all, distance_ts_all, channel_trace_map, allowed_elevs,
     f_c, Ps)

U = size(elevation_quantized_all, 3);
T = size(elevation_quantized_all, 2);

G = 4; alpha = 0.2; beta = 0.1;
snr_threshold = -5; rate_threshold = 0.05;
low_quality_threshold_snr = 2;
low_quality_threshold_rate = 0.5;

actual_user_SNR         = zeros(U, T);
actual_user_rate        = zeros(U, T);
actual_user_SNR_primary = zeros(U, T);
actual_user_SNR_dual    = zeros(U, T);
actual_user_rate_primary= zeros(U, T);
actual_user_rate_dual   = zeros(U, T);

prev_assign_snr        = zeros(U, 1);
prev_assign_snr_dual   = zeros(U, 1);
prev_assign_rate       = zeros(U, 1);
prev_assign_rate_dual  = zeros(U, 1);

primary_assign_times = 1:2:T;   % e.g. t = 1, 11, 21, ...
dual_assign_times    = 2:2:T;   % e.g. t = 4, 14, 24, ...

for t = 1:T
    W_snr_ideal  = squeeze(W_SNR_ideal(:, :, t));
    W_rate_ideal = squeeze(W_Rate_ideal(:, :, t));

    % --- SNR PRIMARY assignment/hold ---
    if ismember(t, primary_assign_times)
```

```
        [assign_snr_primary, ~, ~, ~] = assign_users_munkres_algo_2(W_snr_ideal, ...
            prev_assign_snr, prev_assign_snr_dual, G, alpha, beta, ...
            snr_threshold, low_quality_threshold_snr, t == 1, false);
        prev_assign_snr = assign_snr_primary;
    else
        assign_snr_primary = prev_assign_snr;
    end

    % --- SNR DUAL assignment/hold ---
    if ismember(t, dual_assign_times)
        [~, assign_snr_dual, ~, ~] = assign_users_munkres_algo_2(W_snr_ideal, ...
            prev_assign_snr, prev_assign_snr_dual, G, alpha, beta, ...
            snr_threshold, low_quality_threshold_snr, t == 1, true);
        prev_assign_snr_dual = assign_snr_dual;
    else
        assign_snr_dual = prev_assign_snr_dual;
    end

    % --- Rate PRIMARY assignment/hold ---
    if ismember(t, primary_assign_times)
        [assign_rate_primary, ~, ~, ~] = assign_users_munkres_algo_2(W_rate_ideal, ...
            prev_assign_rate, prev_assign_rate_dual, G, alpha, beta, ...
            rate_threshold, low_quality_threshold_rate, t == 1, false);
        prev_assign_rate = assign_rate_primary;
    else
        assign_rate_primary = prev_assign_rate;
    end

    % --- Rate DUAL assignment/hold ---
    if ismember(t, dual_assign_times)
        [~, assign_rate_dual, ~, ~] = assign_users_munkres_algo_2(W_rate_ideal, ...
            prev_assign_rate, prev_assign_rate_dual, G, alpha, beta, ...
            rate_threshold, low_quality_threshold_rate, t == 1, true);
        prev_assign_rate_dual = assign_rate_dual;
    else
        assign_rate_dual = prev_assign_rate_dual;
    end

    for u = 1:U
        % SNR: primary and dual assignments for this user at this time
        v1 = 0; v2 = 0;
        if assign_snr_primary(u) > 0
            v1 = W_snr_ideal(u, assign_snr_primary(u));
        end
        if assign_snr_dual(u) > 0
            v2 = W_snr_ideal(u, assign_snr_dual(u));
        end
```

```
        actual_user_SNR_primary(u, t) = v1;
        actual_user_SNR_dual(u, t)    = v2;

        % Correct SNR Combination: Linear Sum, then dB
        vals = [v1 v2];
        vals_linear = 10.^(vals(~isnan(vals))/10);
        if isempty(vals_linear)
            actual_user_SNR(u, t) = 0;    % No assignment at all
        else
            actual_user_SNR(u, t) = 10*log10(sum(vals_linear));
        end

        % Rate: primary and dual assignments
        r1 = 0; r2 = 0;
        if assign_rate_primary(u) > 0
            r1 = W_rate_ideal(u, assign_rate_primary(u));
        end
        if assign_rate_dual(u) > 0
            r2 = W_rate_ideal(u, assign_rate_dual(u));
        end
        actual_user_rate_primary(u, t) = r1;
        actual_user_rate_dual(u, t)    = r2;
        actual_user_rate(u, t)         = r1 + r2;
    end
end
end
```

### A.3.3 simulate_forecast_mode_algo_2.m

```
function [actual_user_SNR, actual_user_rate, ...
          actual_user_SNR_primary, actual_user_SNR_dual, ...
          actual_user_rate_primary, actual_user_rate_dual] = ...
    simulate_forecast_mode_algo_2(W_SNR_forecast, W_Rate_forecast, W_SNR_ideal,
    W_Rate_ideal, ...
                                  elevation_quantized_all, distance_ts_all,
   channel_trace_map, allowed_elevs, f_c, Ps)

U = size(elevation_quantized_all, 3);
T = size(elevation_quantized_all, 2);
M = size(elevation_quantized_all, 1);

G = 4; alpha = 0.2; beta = 0.1;
snr_threshold = -5; rate_threshold = 0.05;
low_quality_threshold_snr = 2;
low_quality_threshold_rate = 0.5;

actual_user_SNR         = zeros(U, T);
actual_user_rate        = zeros(U, T);
actual_user_SNR_primary = zeros(U, T);
```

```
actual_user_SNR_dual    = zeros(U, T);
actual_user_rate_primary= zeros(U, T);
actual_user_rate_dual   = zeros(U, T);

prev_assign_snr        = zeros(U, 1);
prev_assign_snr_dual   = zeros(U, 1);
prev_assign_rate       = zeros(U, 1);
prev_assign_rate_dual  = zeros(U, 1);

primary_assign_times = 1:2:T;   % assignment at t = 1, 11, ...
dual_assign_times    = 2:2:T;   % assignment at t = 4, 14, ...

for t = 1:T
    W_snr_forecast = W_SNR_forecast(:, :, t);    % [U x M]
    W_rate_forecast = W_Rate_forecast(:, :, t);  % [U x M]
    W_snr_ideal = W_SNR_ideal(:, :, t);          % [U x M]
    W_rate_ideal = W_Rate_ideal(:, :, t);        % [U x M]

    if ~any(W_snr_forecast(:)) && ~any(W_rate_forecast(:)), continue; end

    % --- SNR PRIMARY assignment/hold ---
    if ismember(t, primary_assign_times)
        [assign_snr_primary, ~, ~, ~] = assign_users_munkres_algo_2(
    W_snr_forecast, ...
            prev_assign_snr, prev_assign_snr_dual, G, alpha, beta, ...
            snr_threshold, low_quality_threshold_snr, t == 1, false);
        prev_assign_snr = assign_snr_primary;
    else
        assign_snr_primary = prev_assign_snr;
    end

    % --- SNR DUAL assignment/hold ---
    if ismember(t, dual_assign_times)
        [~, assign_snr_dual, ~, ~] = assign_users_munkres_algo_2(W_snr_forecast
    , ...
            prev_assign_snr, prev_assign_snr_dual, G, alpha, beta, ...
            snr_threshold, low_quality_threshold_snr, t == 1, true);
        prev_assign_snr_dual = assign_snr_dual;
    else
        assign_snr_dual = prev_assign_snr_dual;
    end

    % --- Rate PRIMARY assignment/hold ---
    if ismember(t, primary_assign_times)
        [assign_rate_primary, ~, ~, ~] = assign_users_munkres_algo_2(
    W_rate_forecast, ...
            prev_assign_rate, prev_assign_rate_dual, G, alpha, beta, ...
            rate_threshold, low_quality_threshold_rate, t == 1, false);
        prev_assign_rate = assign_rate_primary;
    else
        assign_rate_primary = prev_assign_rate;
```

```
    end

    % --- Rate DUAL assignment/hold ---
    if ismember(t, dual_assign_times)
        [~, assign_rate_dual, ~, ~] = assign_users_munkres_algo_2(
    W_rate_forecast, ...
            prev_assign_rate, prev_assign_rate_dual, G, alpha, beta, ...
            rate_threshold, low_quality_threshold_rate, t == 1, true);
        prev_assign_rate_dual = assign_rate_dual;
    else
        assign_rate_dual = prev_assign_rate_dual;
    end

    for u = 1:U
        v1 = 0; v2 = 0;
        % SNR Primary (use ideal matrix for evaluation)
        if assign_snr_primary(u) > 0
            v1 = W_snr_ideal(u, assign_snr_primary(u));
        end
        % SNR Dual
        if assign_snr_dual(u) > 0
            v2 = W_snr_ideal(u, assign_snr_dual(u));
        end

        actual_user_SNR_primary(u, t) = v1;
        actual_user_SNR_dual(u, t)    = v2;

        vals = [v1 v2];
        valid = ~isnan(vals) & (vals > 0);
        if any(valid)
            actual_user_SNR(u, t) = 10*log10(sum(10.^(vals(valid)/10)));
        else
            actual_user_SNR(u, t) = 0;
        end

        r1 = 0; r2 = 0;
        % Rate Primary (use ideal matrix for evaluation)
        if assign_rate_primary(u) > 0
            r1 = W_rate_ideal(u, assign_rate_primary(u));
        end
        % Rate Dual
        if assign_rate_dual(u) > 0
            r2 = W_rate_ideal(u, assign_rate_dual(u));
        end

        actual_user_rate_primary(u, t) = r1;
        actual_user_rate_dual(u, t)    = r2;
        actual_user_rate(u, t)         = r1 + r2;
    end
end
end
```

### A.3.4 siso_algo_2_comparison.m

```
clear all
close all
clc

% --- System Parameters ---
h0 = 550; iK = 53; Nsat = 22; Norbits = 72; U = 50; dt = 10; Tmax = 3600;
deltaPhi = 30; M = Nsat * Norbits; T = Tmax / dt + 1;
user_latitudes = 45 + (50 - 45) * rand(U, 1);
user_longitudes = 55 + (60 - 55) * rand(U, 1);

elevation_ts_all = zeros(M, T, U); distance_ts_all = zeros(M, T, U);
for u = 1:U
    for ind = 1:Norbits
        base_idx = (ind - 1) * Nsat + 1;
        [elev_ts, dist_ts, ~] = generate_passage_singleorbit(h0, iK, ...
            (ind - 1) * deltaPhi, -(ind - 1) * 360 / Nsat / Norbits, ...
            Nsat, user_latitudes(u), user_longitudes(u), dt, Tmax);
        elevation_ts_all(base_idx:base_idx + Nsat - 1, :, u) = elev_ts;
        distance_ts_all(base_idx:base_idx + Nsat - 1, :, u) = dist_ts;
    end
end

% --- Elevation Quantization ---
allowed_elevs = [30, 45, 60, 70];
partition = [25, 37.5, 52.5, 65, inf];
elevation_quantized_all = NaN(size(elevation_ts_all));
for u = 1:U
    quantized_indices = discretize(elevation_ts_all(:, :, u), partition);
    temp = NaN(size(elevation_ts_all(:, :, u)));
    valid_idx = ~isnan(quantized_indices);
    temp(valid_idx) = allowed_elevs(quantized_indices(valid_idx));
    temp(elevation_ts_all(:, :, u) < 5) = NaN;
    elevation_quantized_all(:, :, u) = temp;
end

% --- Channel Fading Traces ---
f_c = 2e9; envi = 'Suburban'; Tch = 5000;
channel_trace_map = containers.Map('KeyType', 'double', 'ValueType', 'any');
for k = 1:length(allowed_elevs)
    elev = allowed_elevs(k);
    try
        [y, ~, ~, ~, ~, ~] = coef_time_series_long_pedestrian(f_c, envi, elev,
    Tch);
        channel_trace_map(elev) = abs(y).^2;  % Only average used
    catch
        channel_trace_map(elev) = zeros(1, Tch);
    end
end

```

```
% --- Transmit power values (in Watts) ---
Ps_values = [0.1 0.2 0.5 1 1.5 2];

% Preallocate results for plotting and analysis
avg_throughput_ideal    = zeros(length(Ps_values), 1);
avg_throughput_forecast = zeros(length(Ps_values), 1);
avg_snr_ideal           = zeros(length(Ps_values), 1);
avg_snr_forecast        = zeros(length(Ps_values), 1);

results_ideal    = cell(length(Ps_values), 1);
results_forecast = cell(length(Ps_values), 1);

sum_snr_ideal     = cell(length(Ps_values), 1);
sum_snr_forecast  = cell(length(Ps_values), 1);
sum_rate_ideal    = cell(length(Ps_values), 1);
sum_rate_forecast = cell(length(Ps_values), 1);

for i = 1:length(Ps_values)
    Ps = Ps_values(i);

    % --- Building the weight matrices ---
    [W_SNR_ideal, W_Rate_ideal, W_SNR_forecast, W_Rate_forecast] = ...
    build_weight_matrices_AT(Ps, channel_trace_map, elevation_quantized_all,
    distance_ts_all, allowed_elevs, f_c);

    % --- Ideal Mode ---
    [snr_ideal, rate_ideal, snr_ideal_primary, snr_ideal_dual,
    rate_ideal_primary, rate_ideal_dual] = ...
        simulate_ideal_mode_algo_2(W_SNR_ideal, W_Rate_ideal,
    elevation_quantized_all, distance_ts_all, channel_trace_map, allowed_elevs,
     f_c, Ps);
    avg_throughput_ideal(i) = mean(rate_ideal(:),'omitnan');
    avg_snr_ideal(i)        = mean(snr_ideal(:),'omitnan');
    results_ideal{i} = struct('snr', snr_ideal, 'rate', rate_ideal, ...
        'snr_primary', snr_ideal_primary, 'snr_dual', snr_ideal_dual, ...
        'rate_primary', rate_ideal_primary, 'rate_dual', rate_ideal_dual);
    sum_snr_ideal{i}  = sum(snr_ideal, 1, 'omitnan');   % [1 x T], sum over
    users
    sum_rate_ideal{i} = sum(rate_ideal, 1, 'omitnan');  % [1 x T], sum over
    users

    % --- Forecast Mode ---
    [snr_forecast, rate_forecast, snr_forecast_primary, snr_forecast_dual,
    rate_forecast_primary, rate_forecast_dual] = ...
    simulate_forecast_mode_algo_2(W_SNR_forecast, W_Rate_forecast, W_SNR_ideal,
     W_Rate_ideal, elevation_quantized_all, distance_ts_all, channel_trace_map,
     allowed_elevs, f_c, Ps);
    avg_throughput_forecast(i) = mean(rate_forecast(:),'omitnan');
    avg_snr_forecast(i)        = mean(snr_forecast(:),'omitnan');
    results_forecast{i} = struct('snr', snr_forecast, 'rate', rate_forecast,
    ...
```

```
        'snr_primary', snr_forecast_primary, 'snr_dual', snr_forecast_dual, ...
        'rate_primary', rate_forecast_primary, 'rate_dual', rate_forecast_dual)
    ;
    sum_snr_forecast{i}  = sum(snr_forecast, 1, 'omitnan');   % [1 x T]
    sum_rate_forecast{i} = sum(rate_forecast, 1, 'omitnan');  % [1 x T]
end

% --- Plotting Section (Algo 2, with consistent styling) ---

output_folder = fullfile(pwd, 'Plots_LEO_Handover_algo_2');
if ~exist(output_folder, 'dir')
    mkdir(output_folder);
end
save_plot = @(name) saveas(gcf, fullfile(output_folder, name));

% Convert transmit powers to dBm
Ps_dBm = 10 * log10(Ps_values * 1000);

i_plot = 3; user_idx = 1;
d_ideal    = results_ideal{i_plot};
d_forecast = results_forecast{i_plot};

%%1) Average Throughput vs Transmit Power (in dBm)
figure;
plot(Ps_dBm, avg_throughput_ideal, 'b-o', 'LineWidth', 1.6); hold on;
plot(Ps_dBm, avg_throughput_forecast, 'r--s', 'LineWidth', 1.6);
xlabel('Transmit Power (dBm)', 'FontWeight', 'bold');
ylabel('Average Throughput (bps/Hz)', 'FontWeight', 'bold');
title({'Throughput vs Transmit Power', ...
       'U = 50 users, T = 361 time steps (10 s each), Suburban Environment'}, ...
       'FontWeight', 'bold');
legend('Ideal Mode', 'Forecast Mode', 'Location', 'best');
grid on;
axis tight; ylim padded;
save_plot('1_Throughput_vs_Ps_2.png');

%%2) Average SNR vs Transmit Power (in dBm)
figure;
plot(Ps_dBm, avg_snr_ideal, 'b-o', 'LineWidth', 1.6); hold on;
plot(Ps_dBm, avg_snr_forecast, 'r--s', 'LineWidth', 1.6);
xlabel('Transmit Power (dBm)', 'FontWeight', 'bold');
ylabel('Average SNR (dB)', 'FontWeight', 'bold');
title({'SNR vs Transmit Power', ...
       'U = 50 users, T = 361 time steps (10 s each), Suburban Environment'}, ...
       'FontWeight', 'bold');
legend('Ideal Mode', 'Forecast Mode', 'Location', 'best');
grid on;
axis tight; ylim padded;
save_plot('2_SNR_vs_Ps_2.png');
```

```
138
139 %%3) System Sum Rate vs Time
140 figure;
141 plot(sum_rate_ideal{i_plot}, 'b-', 'LineWidth', 2); hold on;
142 plot(sum_rate_forecast{i_plot}, 'r--', 'LineWidth', 2);
143 xlabel('Time Step (10 s)', 'FontWeight', 'bold');
144 ylabel('Total System Rate (bps/Hz)', 'FontWeight', 'bold');
145 legend('Ideal', 'Forecast', 'Location', 'best');
146 title('Total System Rate per Time Step', 'FontWeight', 'bold');
147 grid on;
148 axis tight; ylim padded;
149 save_plot('3_System_Sum_Rate_vs_Time_2.png');
150
151 %%4) System Sum SNR vs Time
152 figure;
153 plot(sum_snr_ideal{i_plot}, 'b-', 'LineWidth', 2); hold on;
154 plot(sum_snr_forecast{i_plot}, 'r--', 'LineWidth', 2);
155 xlabel('Time Step (10 s)', 'FontWeight', 'bold');
156 ylabel('Total System SNR (dB)', 'FontWeight', 'bold');
157 legend('Ideal', 'Forecast', 'Location', 'best');
158 title('Total System SNR per Time Step', 'FontWeight', 'bold');
159 grid on;
160 axis tight; ylim padded;
161 save_plot('4_System_Sum_SNR_vs_Time_2.png');
162
163 %%5) Single-User Rate Evolution
164 figure;
165 plot(d_ideal.rate(user_idx,:), 'b-', 'LineWidth', 2); hold on;
166 plot(d_forecast.rate(user_idx,:), 'r--', 'LineWidth', 2);
167 xlabel('Time Step (10 s)', 'FontWeight', 'bold');
168 ylabel('Rate (bps/Hz)', 'FontWeight', 'bold');
169 legend('Ideal', 'Forecast', 'Location', 'best');
170 title(sprintf('Rate Evolution - User %d', user_idx), 'FontWeight', 'bold');
171 grid on;
172 axis tight; ylim padded;
173 save_plot(sprintf('5_SingleUser_Rate_User%d_2.png', user_idx));
174
175 %%6) Single-User SNR Evolution
176 figure;
177 plot(d_ideal.snr(user_idx,:), 'b-', 'LineWidth', 2); hold on;
178 plot(d_forecast.snr(user_idx,:), 'r--', 'LineWidth', 2);
179 xlabel('Time Step (10 s)', 'FontWeight', 'bold');
180 ylabel('SNR (dB)', 'FontWeight', 'bold');
181 legend('Ideal', 'Forecast', 'Location', 'best');
182 title(sprintf('SNR Evolution - User %d', user_idx), 'FontWeight', 'bold');
183 grid on;
184 axis tight; ylim padded;
185 save_plot(sprintf('6_SingleUser_SNR_User%d_2.png', user_idx));
186
187 %%7) System Primary vs Dual Sums (Ideal)
188 figure;
```

```
plot(sum(d_ideal.rate_primary, 1, 'omitnan'), 'b-', 'LineWidth', 1.8); hold on;
plot(sum(d_ideal.rate_dual,    1, 'omitnan'), 'r--', 'LineWidth', 1.8);
plot(sum(d_ideal.rate,         1, 'omitnan'), 'k:', 'LineWidth', 2.2);
xlabel('Time Step (10 s)', 'FontWeight', 'bold');
ylabel('Sum Rate (bps/Hz)', 'FontWeight', 'bold');
legend('Primary', 'Dual', 'Total', 'Location', 'best');
title('Ideal: System Primary, Dual, Total Rate', 'FontWeight', 'bold');
grid on;
axis tight; ylim padded;
save_plot('7_System_Primary_Dual_Total_Rate_Ideal_2.png');

figure;
plot(sum(d_ideal.snr_primary, 1, 'omitnan'), 'b-', 'LineWidth', 1.8); hold on;
plot(sum(d_ideal.snr_dual,    1, 'omitnan'), 'r--', 'LineWidth', 1.8);
plot(sum(d_ideal.snr,         1, 'omitnan'), 'k:', 'LineWidth', 2.2);
xlabel('Time Step (10 s)', 'FontWeight', 'bold');
ylabel('Sum SNR (dB)', 'FontWeight', 'bold');
legend('Primary', 'Dual', 'Total', 'Location', 'best');
title('Ideal: System Primary, Dual, Total SNR', 'FontWeight', 'bold');
grid on;
axis tight; ylim padded;
save_plot('8_System_Primary_Dual_Total_SNR_Ideal_2.png');

%%8) CDF of Average User Rate
user_rate_mean_ideal    = mean(d_ideal.rate,    2, 'omitnan');
user_rate_mean_forecast = mean(d_forecast.rate, 2, 'omitnan');
valid_ideal             = user_rate_mean_ideal(~isnan(user_rate_mean_ideal));
valid_forecast          = user_rate_mean_forecast(~isnan(
   user_rate_mean_forecast));
if isempty(valid_ideal) || isempty(valid_forecast)
    warning('No valid user averages for RATE CDF plot.');
else
    figure;
    cdfplot(valid_ideal); hold on;
    cdfplot(valid_forecast);
    legend('Ideal', 'Forecast', 'Location', 'best');
    xlabel('Average Rate per User (bps/Hz)', 'FontWeight', 'bold');
    ylabel('Cumulative Probability', 'FontWeight', 'bold');
    title('CDF of Average User Rate', 'FontWeight', 'bold');
    grid on;
    axis tight; xlim padded;
    save_plot('9_CDF_AvgUserRate_2.png');
end

%%9) CDF of Average User SNR
user_snr_mean_ideal    = mean(d_ideal.snr,    2, 'omitnan');
user_snr_mean_forecast = mean(d_forecast.snr, 2, 'omitnan');
valid_ideal_snr        = user_snr_mean_ideal(~isnan(user_snr_mean_ideal));
valid_forecast_snr     = user_snr_mean_forecast(~isnan(user_snr_mean_forecast))
    ;
if isempty(valid_ideal_snr) || isempty(valid_forecast_snr)
```

```
    warning('No valid user averages for SNR CDF plot.');
else
    figure;
    cdfplot(valid_ideal_snr); hold on;
    cdfplot(valid_forecast_snr);
    legend('Ideal', 'Forecast', 'Location', 'best');
    xlabel('Average SNR per User (dB)', 'FontWeight', 'bold');
    ylabel('Cumulative Probability', 'FontWeight', 'bold');
    title('CDF of Average User SNR', 'FontWeight', 'bold');
    grid on;
    axis tight; xlim padded;
    save_plot('10_CDF_AvgUserSNR_2.png');
end
```

## A.4 Environment Comparison Script

### A.4.1 siso_algo_environment_comparison.m

The script `siso_algo_environment_comparison.m` was used to compare the performance of the proposed user–satellite assignment algorithms across different propagation environments (e.g., Village, Urban, Suburban, Rural). The random number generator is initialized with a fixed seed (`rng("default")`), ensuring that user positions, satellite geometry, and temporal dynamics remain identical across environments. This guarantees that the only varying factor in the experiments is the propagation model, allowing fair and seed-consistent comparison of per-user average rate and SNR distributions in both ideal and forecast modes.

```
clear all
close all
clc

rng("default");


% --- System Parameters ---
h0 = 550; iK = 53; Nsat = 22; Norbits = 72; U = 50; dt = 10; Tmax = 3600;
deltaPhi = 30; M = Nsat * Norbits; T = Tmax / dt + 1;
user_latitudes = 45 + (50 - 45) * rand(U, 1);
user_longitudes = 55 + (60 - 55) * rand(U, 1);

elevation_ts_all = zeros(M, T, U); distance_ts_all = zeros(M, T, U);
for u = 1:U
    for ind = 1:Norbits
        base_idx = (ind - 1) * Nsat + 1;
        [elev_ts, dist_ts, ~] = generate_passage_singleorbit(h0, iK, ...
            (ind - 1) * deltaPhi, -(ind - 1) * 360 / Nsat / Norbits, ...
            Nsat, user_latitudes(u), user_longitudes(u), dt, Tmax);
        elevation_ts_all(base_idx:base_idx + Nsat - 1, :, u) = elev_ts;
```

```
        distance_ts_all(base_idx:base_idx + Nsat - 1, :, u) = dist_ts;
    end
end

% --- Elevation Quantization ---
allowed_elevs = [30, 45, 60, 70];
partition = [25, 37.5, 52.5, 65, inf];
elevation_quantized_all = NaN(size(elevation_ts_all));
for u = 1:U
    quantized_indices = discretize(elevation_ts_all(:, :, u), partition);
    temp = NaN(size(elevation_ts_all(:, :, u)));
    valid_idx = ~isnan(quantized_indices);
    temp(valid_idx) = allowed_elevs(quantized_indices(valid_idx));
    temp(elevation_ts_all(:, :, u) < 5) = NaN;
    elevation_quantized_all(:, :, u) = temp;
end

% --- Channel Fading Traces ---
f_c = 2e9; envi = 'Village'; Tch = 5000;
channel_trace_map = containers.Map('KeyType', 'double', 'ValueType', 'any');
for k = 1:length(allowed_elevs)
    elev = allowed_elevs(k);
    try
        [y, ~, ~, ~, ~, ~] = coef_time_series_long_pedestrian(f_c, envi, elev,
    Tch);
        channel_trace_map(elev) = abs(y).^2;  % Only average used
    catch
        channel_trace_map(elev) = zeros(1, Tch);
    end
end

% --- Transmit power (fixed) ---
Ps = 1; % Watts

% --- Weight Matrices ---
[W_SNR_ideal, W_Rate_ideal, W_SNR_forecast, W_Rate_forecast] = ...
    build_weight_matrices_AT(Ps, channel_trace_map, elevation_quantized_all,
    distance_ts_all, allowed_elevs, f_c);

% --- Run Ideal Simulation (Algo 1) ---
[snr_ideal, rate_ideal, snr_ideal_primary, snr_ideal_dual, rate_ideal_primary,
    rate_ideal_dual] = ...
    simulate_ideal_mode_algo_1(W_SNR_ideal, W_Rate_ideal,
    elevation_quantized_all, distance_ts_all, channel_trace_map, allowed_elevs,
     f_c, Ps);

% --- Run Forecast Simulation (Algo 1) ---
[snr_forecast, rate_forecast, snr_forecast_primary, snr_forecast_dual,
    rate_forecast_primary, rate_forecast_dual] = ...
    simulate_forecast_mode_algo_1(W_SNR_forecast, W_Rate_forecast, W_SNR_ideal,
     W_Rate_ideal, elevation_quantized_all, distance_ts_all, channel_trace_map,
```

```
    allowed_elevs, f_c, Ps);

% --- Store Results ---
results_ideal = struct('snr', snr_ideal, 'rate', rate_ideal, ...
    'snr_primary', snr_ideal_primary, 'snr_dual', snr_ideal_dual, ...
    'rate_primary', rate_ideal_primary, 'rate_dual', rate_ideal_dual);

results_forecast = struct('snr', snr_forecast, 'rate', rate_forecast, ...
    'snr_primary', snr_forecast_primary, 'snr_dual', snr_forecast_dual, ...
    'rate_primary', rate_forecast_primary, 'rate_dual', rate_forecast_dual);

% --- Save Results Folder ---
output_folder = fullfile(pwd, 'SISO_algo1_Plots_Ps1');
if ~exist(output_folder, 'dir')
    mkdir(output_folder);
end
save_plot = @(name) saveas(gcf, fullfile(output_folder, name));

% --- Per-User Average Metrics ---
user_rate_mean_ideal = mean(rate_ideal, 2, 'omitnan');
user_rate_mean_forecast = mean(rate_forecast, 2, 'omitnan');
user_snr_mean_ideal = mean(snr_ideal, 2, 'omitnan');
user_snr_mean_forecast = mean(snr_forecast, 2, 'omitnan');

% --- Overall Means ---
overall_rate_ideal = mean(user_rate_mean_ideal, 'omitnan');
overall_rate_forecast = mean(user_rate_mean_forecast, 'omitnan');
overall_snr_ideal = mean(user_snr_mean_ideal, 'omitnan');
overall_snr_forecast = mean(user_snr_mean_forecast, 'omitnan');

fprintf('Overall Avg Rate (Ideal):    %.4f bps/Hz\n', overall_rate_ideal);
fprintf('Overall Avg Rate (Forecast): %.4f bps/Hz\n', overall_rate_forecast);
fprintf('Overall Avg SNR  (Ideal):    %.4f dB\n', overall_snr_ideal);
fprintf('Overall Avg SNR  (Forecast): %.4f dB\n', overall_snr_forecast);

% --- Save .mat Results ---
save(fullfile(output_folder, 'results_avg_Ps1_algo1.mat'), ...
    'user_rate_mean_ideal', 'user_rate_mean_forecast', ...
    'user_snr_mean_ideal', 'user_snr_mean_forecast', ...
    'overall_rate_ideal', 'overall_rate_forecast', ...
    'overall_snr_ideal', 'overall_snr_forecast');

% ====== PLOTTING (simple + consistent axes) ======
dt_label = sprintf('Time Step (%d s)', dt);  % For any time-based plots you add
    later
meta_line = sprintf('U = %d users, T = %d time steps (%d s each), %s', U, T, dt
    , envi);

% --- CDF of average rate per user ---
figure;
cdfplot(user_rate_mean_ideal); hold on;
```

```
cdfplot(user_rate_mean_forecast);
legend('Ideal', 'Forecast', 'Location', 'best');
xlabel('Average Rate per User (bps/Hz)', 'FontWeight', 'bold');
ylabel('Cumulative Probability', 'FontWeight', 'bold');
title({'CDF of Per-User Average Rate (Ps = 1)', meta_line}, 'FontWeight', 'bold');
grid on;
axis tight; xlim padded; ylim padded;
save_plot('CDF_User_Avg_Rate_Village.png');

% --- Bar plot of average rate per user ---
figure;
bar([user_rate_mean_ideal, user_rate_mean_forecast]);
xlabel('User Index', 'FontWeight', 'bold');
ylabel('Average Rate (bps/Hz)', 'FontWeight', 'bold');
legend('Ideal', 'Forecast', 'Location', 'best');
title({'Per-User Average Rate Comparison (Ps = 1)', meta_line}, 'FontWeight', 'bold');
grid on;
axis tight; ylim padded;
save_plot('Bar_User_Avg_Rate_Village.png');

% --- CDF of average SNR per user ---
figure;
cdfplot(user_snr_mean_ideal); hold on;
cdfplot(user_snr_mean_forecast);
legend('Ideal', 'Forecast', 'Location', 'best');
xlabel('Average SNR per User (dB)', 'FontWeight', 'bold');
ylabel('Cumulative Probability', 'FontWeight', 'bold');
title({'CDF of Per-User Average SNR (Ps = 1)', meta_line}, 'FontWeight', 'bold');
grid on;
axis tight; xlim padded; ylim padded;
save_plot('CDF_User_Avg_SNR_Village.png');

% --- Bar plot of average SNR per user ---
figure;
bar([user_snr_mean_ideal, user_snr_mean_forecast]);
xlabel('User Index', 'FontWeight', 'bold');
ylabel('Average SNR (dB)', 'FontWeight', 'bold');
legend('Ideal', 'Forecast', 'Location', 'best');
title({'Per-User Average SNR Comparison (Ps = 1)', meta_line}, 'FontWeight', 'bold');
grid on;
axis tight; ylim padded;
save_plot('Bar_User_Avg_SNR_Village.png');
```

# References


Abdu, T. S., Lagunas, E., Ha, V. N., Grotz, J., Kisseleff, S., & Chatzinotas, S. (2023). Demand-aware flexible handover strategy for leo constellation. In *2023 ieee international conference on communications (icc)* (pp. 1–6). doi: 10.1109/ICC45041.2023.10278369

Ben Salem, H., Tarable, A., Nordio, A., & Makki, B. (2025). Uplink soft handover for leo constellations: How strong the inter-satellite link should be. In *Proceedings of the ieee 35th international symposium on personal, indoor and mobile radio communications (pimrc)* (pp. 1–7). doi: 10.1109/PIMRC59610.2024.10817368

CNR IRIS Research. (2021). *Impact of latency in satellite systems on interactive applications.* https://iris.cnr.it/retrieve/d4ca616e-6df2-4b23-bc58-635aae7994c7/prod_160280-doc_126009.pdf. (Accessed: 2025-08-28)

European Space Agency (ESA). (n.d.). *Types of Orbits – GEO vs LEO/MEO.* https://www.esa.int/Enabling_Support/Space_Transportation/Types_of_orbits. (Accessed: 2025-08-29)

Feng, L., Liu, Y., Wu, L., Zhang, Z., & Dang, J. (2020). A satellite handover strategy based on mimo technology in leo satellite networks. *IEEE Communications Letters*, *24*(7), 1505–1509. doi: 10.1109/LCOMM.2020.2988043

Hofmann-Wellenhof, B., Lichtenegger, H., & Wasle, E. (2007). *Gnss – global navigation satellite systems: Gps, glonass, galileo, and more.* Springer.

International Telecommunication Union. (2017). *ITU-R P.681-10: Propagation Data Required for the Design of Earth-space Land Mobile Telecommunication Systems.* Recommendation ITU-R P.681-10. (Available from ITU)

Juan, E., Lauridsen, M., Wigard, J., & Mogensen, P. (2022). Handover solutions for 5g low-earth orbit satellite networks. *IEEE Access*, *10*(1), 93309–93331. doi: 10.1109/ACCESS.2022.3203189

Jung, S., Lee, M.-S., Kim, J., Yun, M.-Y., Kim, J.-H., & Kim, J.-H. (2022). Trustworthy handover in leo satellite mobile networks. *ICT Express*, *8*(3), 432–437. doi: 10.1016/j.icte.2021.10.011

Kaplan, E. D., & Hegarty, C. J. (2005). *Understanding gps: Principles and applications* (2nd ed.). Artech House.

Kodheli, O., Lagunas, E., Maturo, N., Sharma, S., Chatzinotas, S., Kisseleff, S., ... Evans, B. (2021). Satellite communications in the new space era: A survey and future challenges. *IEEE Communications Surveys & Tutorials*, *23*(1), 70–109. doi: 10.1109/COMST.2020.3028247

Kuhn, H. W. (1955). The hungarian method for the assignment problem. *Naval Research Logistics Quarterly*, *2*(1-2), 83–97. doi: 10.1002/nav.3800020109

Liu, H., Wang, Y., Li, P., & Cheng, J. (2024). A multi-agent deep reinforcement learning-based handover scheme for mega-constellation under dynamic propagation conditions. *IEEE Transactions on Wireless Communications*, *23*(10), 13579–13595. doi: 10.1109/TWC.2024.3407358

Maral, G., Bousquet, M., & Sun, Z. (2020). *Satellite communications systems: Systems, techniques and technology* (6th ed.). Wiley.

Miao, F., Zhang, W., Zhang, S., Li, H., Wang, B., Lu, M., & Zhao, Y. (2019). A multi-attribute decision handover scheme for leo mobile satellite networks. *IEEE Transactions on Vehicular Technology*, *68*(2), 2042–2056. doi: 10.1109/TVT.2018.2889768

Munkres, J. (1957). Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, *5*(1), 32–38. doi: 10.1137/0105003

Rago, A., Guidotti, A., Piro, G., Cianca, E., Vanelli-Coralli, A., Morosi, S., ... Grieco, L. A. (2024). Multi-layer ntn architectures toward 6g: The ita-ntn view. *Computer Networks*, *254*, 110725. doi: 10.1016/j.comnet.2024.110725

Smith, J., & Lee, A. (2023). Latency in satellite communications: A comparative analysis. *arXiv preprint arXiv:2303.01633*. Retrieved from https://arxiv.org/abs/2303.01633

SpaceX. (2020). *SpaceX FCC Application for Starlink Gen1.* FCC Application. (Public filing to the US FCC)

Vallado, D. A. (2013). *Fundamentals of astrodynamics and applications* (4th ed.). Microcosm Press and Springer.

Wang, G., Yang, F., Song, J., & Han, Z. (2024). Optimization for dynamic laser inter-satellite link scheduling with routing: A multi-agent deep reinforcement learning approach. *IEEE Transactions on Communications*, *72*(5), 2835–2850. doi: 10.1109/TCOMM.2023.3347775

Wertz, J. R., & Larson, W. J. (1999). *Space mission analysis and design* (3rd ed.). El Segundo, CA, USA: Microcosm Press and Kluwer Academic Publishers.

Zhai, Z., Wu, Q., Yu, S., Li, R., Zhang, F., & Chen, X. (2024). Fedleo: An offloading-assisted decentralized federated learning framework for leo satellite networks. *IEEE Transactions on Mobile Computing*, *23*(5), 5260–5279. doi: 10.1109/TMC.2023.3304988