UNICO I+D Project
Grant No. TSI-0630002021-138

6G-SORUS

SORUS-RIS-A2.3-E3

“Final version of the RIS Simulator”

Financiado por la Unión Europea NextGenerationEU
GOBIERNO DE ESPAÑA MINISTERIO DE ASUNTOS ECONÓMICOS Y TRANSFORMACIÓN DIGITAL
SECRETARÍA DE ESTADO DE TELECOMUNICACIONES E INFRAESTRUCTURAS DIGITALES
Plan de Recuperación, Transformación y Resiliencia
UNICO I+D

Document properties

Document number	SORUS-RIS-A2.3-E2
Document title	Final version of the RIS Simulator
Document responsible	Andra Lutu
Document editor	Andra Lutu
Editorial team	Mohammadsina Beyraghi, Paul Almasan, Jose Suarez-Varela, Stefanos Bakirtzis, Giovanni Geraci, Andra Lutu (TID)
Target dissemination level	Private
Status of the document	Final
Version	3.0
Delivery date	18-06-2025
Actual delivery date	09-06-2025

Production properties

Reviewers	Paul Almasan, Jose Suarez-Varela (TID)

Disclaimer

This document has been produced in the context of the SORUS-RIS Project. The research leading to these results has received funding from the Spanish Ministry of Economic Affairs and Digital Transformation and the European Union-NextGenerationEU through the UNICO 5G I+D programme.

Contents

List of Figures

List of Acronyms

3D	Three-Dimensional
4G	Fourth Generation (mobile network)
5G	Fifth Generation (mobile network)
6G	Sixth Generation (mobile network)
AI	Artificial Intelligence
API	Application Programming Interface
BIRCH	Balanced Iterative Reducing and Clustering using Hierarchies
BS	Base Station
CDF	Cumulative Distribution Function
dB	Decibel
DPE	Digital Propagation Environment
EM	Electromagnetic
GHz	Gigahertz
GNN	Graph Neural Network
GPU	Graphics Processing Unit
MAE	Mean Absolute Error
ML	Machine Learning
MNO	Mobile Network Operator

Resumen Ejecutivo

El proyecto SORUS-RIS ofrece un marco avanzado basado en datos para el despliegue automatizado de Superficies Inteligentes Reconfigurables (RIS) en redes celulares, con un fuerte enfoque en escenarios de Industria 4.0 y en la integración de inteligencia artificial (IA) y aprendizaje automático (ML) para la orquestación y optimización. El proyecto aborda el desafío de garantizar una conectividad inalámbrica robusta en entornos urbanos densos—donde la atenuación de la señal, los bloqueos y la propagación con múltiples trayectorias suelen generar zonas sin cobertura—mediante el uso de RIS para reflejar inteligentemente las señales hacia áreas desatendidas. Este enfoque proporciona una alternativa eficiente en términos de energía frente a soluciones tradicionales, como el despliegue de estaciones base adicionales.

El núcleo de la contribución del simulador RIS combina el trazado de rayos específico del sitio (utilizando el motor de código abierto Sionna), el agrupamiento de usuarios para identificar regiones con mala cobertura, y algoritmos heurísticos para determinar la colocación óptima de los RIS, su orientación, configuración de cambio de fase y la formación de haces en las estaciones base. Se proponen dos estrategias de despliegue complementarias: una basada en reflexiones especulares y otra en rayos dispersos, ambas validadas en una red urbana realista con múltiples celdas móviles que abarca las capas de radio 4G, 5G y 6G. La calibración de materiales, basada en mediciones empíricas, garantiza la coherencia física del motor de trazado de rayos.

Una contribución paralela es la introducción de GraphWave, un modelo de propagación basado en redes neuronales de grafos inspirado en el trazado de rayos. GraphWave procesa un entorno de propagación digitalizado para construir una nube de puntos y extraer una representación equivalente en forma de grafo, con el fin de inferir magnitudes relacionadas con la propagación por radio, como la potencia de señal recibida (RSS) en entornos tridimensionales (3D). Este modelo demuestra su capacidad para replicar datos sintéticos con alta precisión y menor tiempo de inferencia, superando además a los modelos convencionales cuando se entrena con mediciones del mundo real.

Los resultados del proyecto muestran que el despliegue de RIS puede mejorar sustancialmente la cobertura, especialmente en regiones con alta densidad de cortes de señal. Sin embargo, lograr mejoras a gran escala requiere desplegar un número significativo de unidades RIS, lo que pone de manifiesto una compensación entre rendimiento y coste. El marco propuesto ha sido validado tanto con datos sintéticos como reales, y se publicará como código abierto para fomentar la reproducibilidad y la investigación futura.

Executive Summary

The SORUS-RIS project delivers an advanced, data-driven framework for the automated deployment of Reconfigurable Intelligent Surfaces (RIS) in cellular networks, with a strong focus on Industry 4.0 scenarios and the integration of artificial intelligence (AI) and machine learning (ML) for orchestration and optimization. The project addresses the challenge of ensuring robust wireless connectivity in dense urban environments—where signal attenuation, blockage, and multipath propagation often result in coverage holes—by leveraging RIS to intelligently reflect signals toward underserved areas. This approach provides an energy-efficient alternative to traditional solutions, such as deploying additional base stations.

The core of the RIS simulator contribution combines site-specific ray tracing (using the open-source Sionna engine), user clustering to identify regions of poor coverage, and ray-based heuristics to determine optimal RIS placement, orientation, phase-shift configuration, and base station beamforming. Two complementary deployment strategies are proposed: one based on specular reflections and another on scattering rays, both validated in a realistic multi-cell urban network across 4G, 5G, and 6G radio layers. Material calibration, grounded in empirical measurements, ensures the physical consistency of the ray tracing engine.

A parallel contribution is the introduction of GraphWave, a neural graph-driven propagation model inspired by ray tracing. GraphWave processes a digitized propagation environment to construct a point cloud and extract an equivalent graph representation, applying neural message passing to infer radio-related quantities such as received signal strength (RSS) in 3D environments. This model demonstrates the ability to replicate synthetic data with high accuracy and reduced inference time, and also outperforms conventional models when trained on real-world measurements.

The project's results show that RIS deployment can substantially improve coverage, especially in high-density outage regions. However, achieving large-scale gains requires deploying a significant number of RIS units, highlighting a tradeoff between performance and cost. The proposed framework is validated with both synthetic and real-world data, and will be made available as open-source to support reproducibility and further research.

1. Introduction

In this report, we build upon the results we previously presented in Deliverable SORUS-RIS-A2.3-E2 and present our final solution for simulations in the radio access network of a commercial operator. For this, we explored two main directions, which we detail next.

Data-Driven Deployment of Reconfigurable Intelligent Surfaces in Cellular Networks (Section 2): Reconfigurable intelligent surfaces (RIS) [1] have been proposed to enhance wireless cellular coverage, especially in dense urban environments where signal attenuation and blockage are more severe. However, realizing RIS gains in practice remains challenging due to the complexity of deployment in real-world scenarios. Key questions regarding RIS placement, orientation, and configuration—as well as the number and size of deployed units—require careful evaluation under site-specific conditions.

In this section, we present a fully automated, data-driven framework for large-scale RIS deployment in cellular networks. Our framework integrates Sionna ray tracing [2], user clustering, and ray-based heuristics to determine RIS placement, orientation, phase-shift configuration, and base station beamforming. We selected Sionna based on the exploration we previously reported in Deliverable SORUS-RIS-A2.3-E2. Though we explored other tools (namely, Wireless InSite), we opted for the open-source tool that would improve the reproducibility of our work.

We propose two complementary strategies to identify candidate deployment locations: one based on specular reflections, and another on scattering rays. We validate our framework in a realistic multi-cell deployment modelled on a real-world urban network and evaluated across 4G, 5G, and 6G radio layers. We further carry out a material calibration procedure based on empirical signal measurements to ensure physical consistency of the ray tracing engine. Our results show that RIS can substantially improve coverage, particularly when deployed in high-density outage regions. However, achieving large-scale gains may require hundreds of RIS units per square km, highlighting the trade-off between performance and deployment cost. We open-source our implementation to support reproducibility and future evaluation of the feasibility of RIS deployment in practical environments.

A Ray-Tracing-Inspired Neural Radio Propagation Model (Section 3): Artificial intelligence-driven radio propagation models provide agile and robust solutions for mobile network operators in their effort to ensure the optimal performance of the wireless ecosystem and support its efficient expansion. In this section, we introduce **GraphWave**, a neural graph-driven propagation solver hinging on the governing principles of ray tracing. Specifically, our model leverages a digitized version of the propagation environment to build a point cloud and extract an equivalent graph representation of the radio environment. Consequently, applying neural message passing over the equivalent graph, allows us to accurately infer radio-related quantities, e.g., received signal strength, in a three-dimensional environment.

We demonstrate that our model can learn to replicate synthetic radio data while achieving lower inference time and can also learn from real-world data, outperforming conventional radio propagation models in accuracy.

If our contribution on data-driven deployment of RIS in cellular network complements the previous results we shared in Deliverable SORUS-RIS-A2.3-E2 (Section 5), our work on GraphWave (A Ray-Tracing-Inspired Neural Radio Propagation Model) brings our exploration on a new direction, namely on the shift from conventional empirical or deterministic models to AI-powered solutions that exhibit enhanced accuracy, computational efficiency and lower cost. This additional ambitious exploration was possible due to the 6-months extension granted in the framework of the SORUS-RIS project, and it presents a state-of-the-art deep learning model that could replace the use of tools such as Sionna.

In the remainder of this document, we will discuss our main results corresponding to the two above-mentioned directions. We note that we have currently submitted these works as academic research papers to top venues in the area.

2. Data-Driven Deployment of Reconfigurable Intelligent Surfaces in Cellular Networks

In this section, we address the challenge of ensuring consistent wireless connectivity in dense urban environments, where signal attenuation, blockage by buildings, and complex multipath propagation lead to coverage holes and unreliable service—especially at higher frequencies used in 5G and prospective 6G networks. Reconfigurable Intelligent Surfaces (RIS) have emerged as a promising technology to enhance coverage by intelligently reflecting signals toward underserved areas, offering an energy-efficient alternative to deploying additional base stations. However, realizing practical gains from RIS remains difficult due to the combinatorial complexity of deployment: there are many possible locations, orientations, and configurations for RIS units, and their optimal arrangement depends on site-specific propagation conditions.

Existing research often oversimplifies the problem, focusing on single-user or single-cell scenarios with idealized or stochastic channel models that ignore real-world building layouts and material properties. Moreover, most studies do not consider the practical constraints of urban environments or the need for scalable, automated deployment strategies.

The key questions we answered with this work include: Where should RIS units be placed for maximum impact? How should they be oriented and configured? How many RIS units are needed, and what size should they be? The current document provides actionable answers to these questions, supporting mobile network operators in making informed decisions about RIS deployment.

2.1 Methodology

We propose a fully automated, data-driven framework for large-scale RIS deployment in cellular networks. Our approach integrates three main components: (1) site-specific ray tracing based on the open-source Sionna engine [2], (2) user clustering to identify regions of poor coverage (outage UEs), and (3) ray-based heuristics to determine optimal RIS placement, orientation, phase-shift configuration, and base station beamforming.

Network and Ray Tracing Model: We model a realistic urban network, comprising 12 base stations (BSs) with three sectors each, spanning a 1340 m x 1390 m area. The simulation environment is constructed using OpenStreetMap [3] data, with accurate building geometries and calibrated material properties. Ray tracing is used to compute the received signal power (RSRP) for user equipment (UEs) distributed on a grid, with propagation effects including reflections, diffraction, and, optionally, scattering.

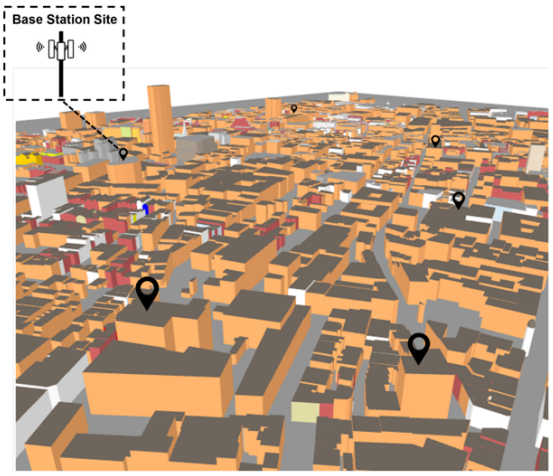
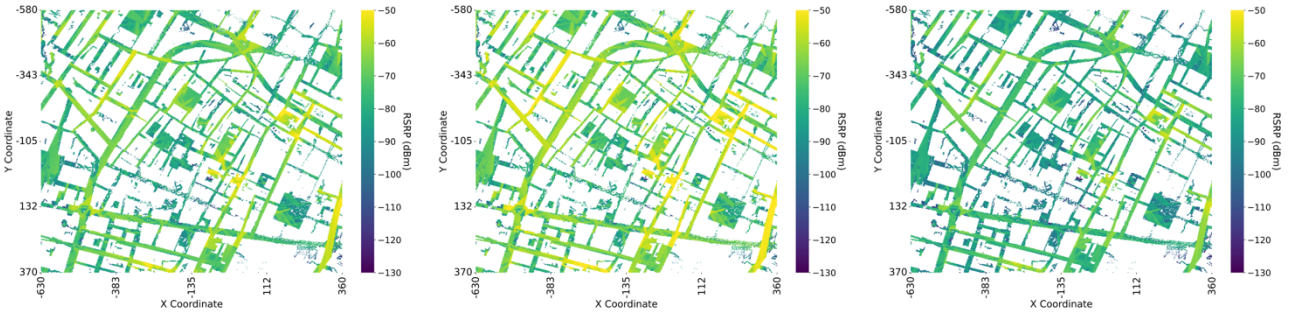


FIGURE 1. 3D VISUALIZATION OF THE AREA UNDER CONSIDERATION, PRODUCED WITH OPENSTREETMAP. BLACK PINS INDICATE BS SITES.

Figure 1 visualizes the urban area under consideration, with base station locations marked, illustrating the complexity and scale of the scenario for which RIS deployment is being optimized.



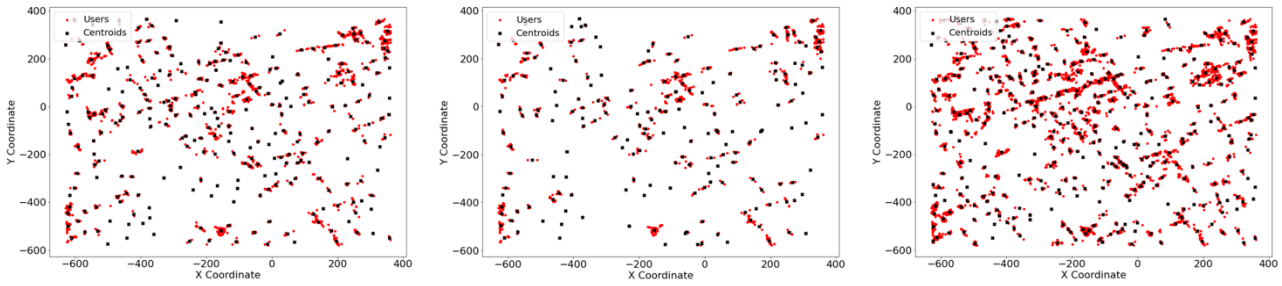
(a) 4G deployment at 2 GHz (b) 5G deployment at 3.5 GHz (c) 6G deployment at 10 GHz

FIGURE 2. RSRP HEATMAPS ACROSS THE CONSIDERED URBAN AREA FOR 4G, 5G, AND 6G WITHOUT RIS.

Figure 2 illustrates RSRP heatmaps for the urban area across 4G, 5G, and 6G networks, highlighting regions of poor coverage that motivate RIS deployment.

Outage Detection and Clustering: UEs with RSRP below -100 dBm are classified as outage UEs [6]. These are clustered using the BIRCH algorithm [5] to identify spatially coherent regions of poor coverage, enabling each RIS to serve a cluster rather than a single UE. This reduces the number of required RIS units and makes deployment scalable.

Figure 3 shows the clustering of outage UEs and the resulting cluster centroids, demonstrating the spatial distribution and grouping of users for RIS targeting.



(a) 4G deployment at 2 GHz (b) 5G deployment at 3.5 GHz (c) 6G deployment at 10 GHz

FIGURE 3. SCATTER PLOT OF OUTAGE UES AND THEIR CLUSTERING FOR THE THREE RADIO DEPLOYMENTS.

RIS Candidate Location Selection: Two complementary strategies are used to identify candidate RIS locations:

- *Reflection-based*: Uses the reflection points of the strongest ray (or all received rays) at the cluster centroid to determine the RIS placement. The RIS is oriented and configured to maximize RSRP for the cluster. The procedure of Strongest Ray algorithm is detailed in **Annex 1**.
- *Scattering-based*: Considers scattering points from single-bounce scattering rays reaching the cluster centroid. RIS is placed at the geometrically valid point closest to the centroid.

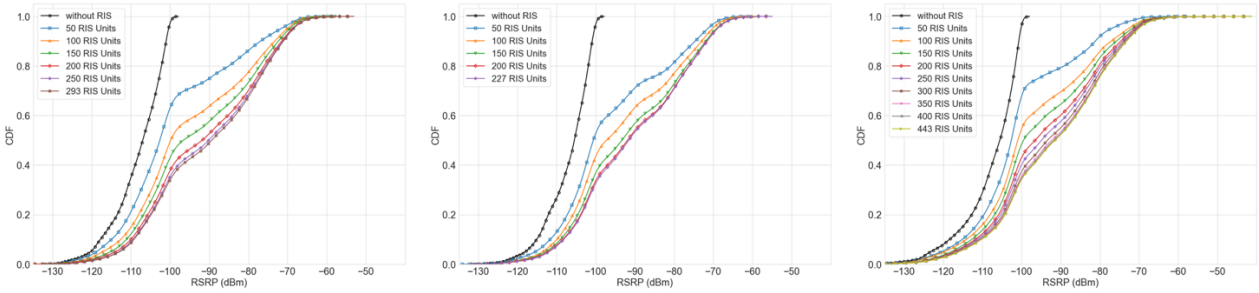
RIS Phase Shift Configuration and BS Beamforming: Once a candidate location is selected, the RIS is oriented to align with the incident and reflected ray directions. The spatial modulation function is computed using a physically consistent reradiation model, and BS beamforming is optimized to maximize RSRP at the RIS location. The procedure of RIS phase shift configuration and BS beamforming are detailed in **Annex 1**.

Fallback Mechanisms: UEs not covered by initial RIS deployments are re-clustered with tighter thresholds or reassigned to existing RIS units based on line-of-sight feasibility.

2.2 Main Results

The framework is evaluated across 4G (2 GHz), 5G (3.5 GHz), and 6G (10 GHz) deployments, using both calibrated and uncalibrated ray tracing models. The main findings are as follows:

Coverage Improvement: RIS deployment significantly enhances coverage in high-density outage regions. For example, in the 4G deployment, initial RIS placement brings over 50% of outage UEs into coverage, with re-clustering and re-association raising the total to about 64%. Similar gains are observed for 5G and 6G, although higher frequencies require more RIS units due to increased path loss and weaker diffraction.



(a) 4G deployment at 2 GHz (b) 5G deployment at 3.5 GHz (c) 6G deployment at 10 GHz

FIGURE 4. RSRP ENHANCEMENT VS. NUMBER OF DEPLOYED RIS IN THE THREE DEPLOYMENT SCENARIOS.

RIS Deployment Density: Achieving substantial coverage gains requires deploying many RIS units—up to several hundred per square kilometer. Prioritizing larger clusters can reduce the number of RIS units needed while still recovering a significant fraction of outage UEs. Figure 4 shows the relationship between the number of deployed RIS units and the RSRS improvement; the figure shows the improvements as the number of RIS increases. At 2 GHz, deploying RISs in the top 200 clusters recovers over 60% of outage UEs, while 100 RISs recover more than 50%. At 3.5 GHz, 150 RISs recover nearly 50% of UEs, while at 10 GHz, 250 RISs address outage for over 60%.

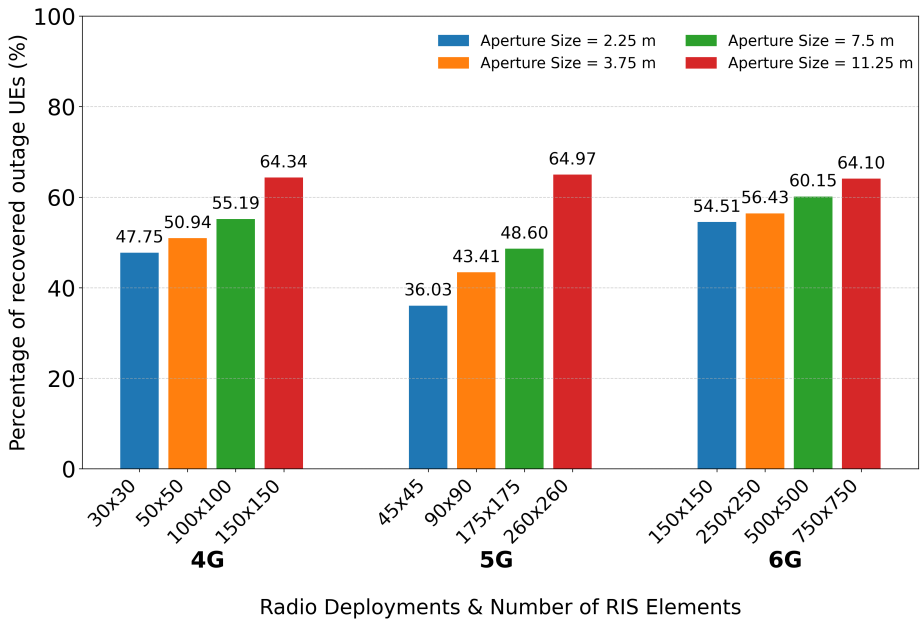


FIGURE 4. PERCENTAGE OF OUTAGE UES RECOVERED WITH DIFFERENT RIS APERTURE SIZES IN VARIOUS RADIO DEPLOYMENTS.

Impact of RIS Aperture: We also investigate the deployment of RIS units with varying aperture sizes, leading to different numbers of RIS elements. The element spacing is always half a wavelength. The assessment is carried out for the highest number of RIS units, namely 293, 227, and 443 in the 4G, 5G, and 6G radio deployments, respectively.

Larger RIS apertures (more elements) yield better coverage improvements but increase hardware complexity and cost. Figure 4 demonstrates the percentage of outage UEs recovered for different RIS aperture sizes and radio technologies, showing that gains plateau as aperture size increases.

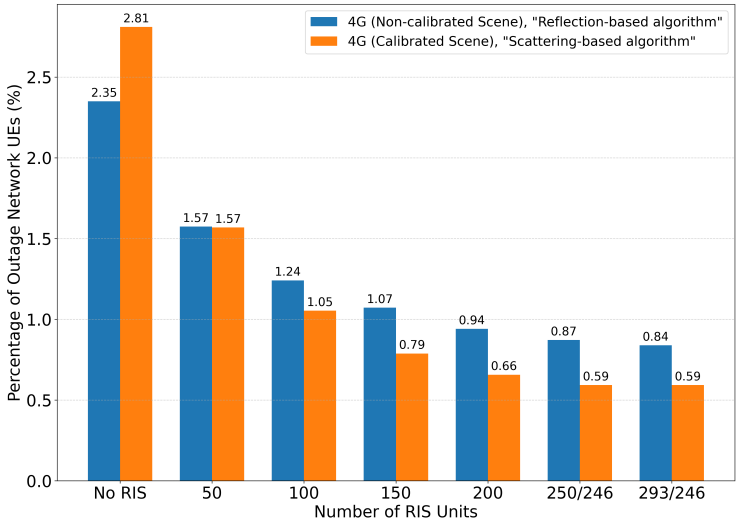


FIGURE 5. PERCENTAGE OF OUTAGE UES AS A FUNCTION OF THE NUMBER OF DEPLOYED RIS UNITS FOR A 4G DEPLOYMENT AT 2 GHZ: NON-CALIBRATED RAY TRACING WITH REFLECTION-BASED ALGORITHM (BLUE) VS. CALIBRATED RAY TRACING WITH SCATTERING-BASED ALGORITHM (ORANGE).

Scattering-Based Deployment vs. Reflection-Based Algorithm: Calibrating material properties based on empirical measurements improves the accuracy of ray tracing predictions – and our results confirm that material calibration significantly improves predictive accuracy. We compare the outcome of the scattering-based placement algorithm, applied in the calibrated scene, with that of the reflection-based algorithm, applied in the non-calibrated scene.

Figure 5 illustrates the percentage of outage UEs recovered as a function of the number of deployed RIS units, comparing reflection-based and scattering-based strategies. To compare both algorithms, we consider the 4G radio deployment (at 2.0GHz) and use RIS with 150×150 elements spaced by λ/2, corresponding to a 11.24m × 11.24m aperture.

For the scattering-based algorithm, we first cluster outage UEs using the BIRCH algorithm, with a clustering threshold of T = 15. Candidate RIS locations are identified based on single bounce scattering rays reaching the centroid, while RSRP is computed using specular reflection and diffraction only, since scattering rays do not contribute practically to received power in simulation. From all candidate locations, we select the one with the minimum 3D distance to the centroid tile. After optimizing the beamforming at the associated BS and the RIS phase configuration, we evaluate the RSRP at the centroid tile. If the RSRP at the centroid improves with the deployed RIS, the unit is configured and evaluated across the cluster UEs. A RIS is retained if more than 60% of UEs benefit relative to the RSRP baseline (without RIS deployment). Otherwise, re-clustering is performed with a tighter threshold (T = 10) to generate smaller, more localized clusters. The algorithm is reapplied to these new groups, and any remaining uncovered UEs are reassigned via the RIS re-association method described in Section III-F. The scattering-based approach recovers approximately 70.2% of the outage UEs after the initial clustering. Re-clustering increases this percentage to nearly 73.2%, and RIS re-association raises the total to 79.6%.

3. From Waves to Graphs: A Ray-Tracing-Inspired Neural Radio Propagation Model

In this section, our primary objective is to overcome the limitations of traditional radio propagation models, which often rely on empirical or deterministic approaches. These conventional methods are either too simplistic, as in empirical models, or computationally intensive and inflexible, as in deterministic ray-tracing simulations. By contrast, we present here the results of our work on gauging the benefits of an AI-driven solution that can accurately infer radio-related quantities—such as received signal strength (RSS) or path loss—in complex, real-world 3D environments. We design our framework to learn from both synthetic and real-world data, bridging the gap between simulation and reality, and offering significant improvements in both accuracy and computational efficiency.

3.1 Methodology and System Design

Our methodology integrates three main components: the digital propagation environment (DPE), the graph representation, and the neural message-passing model.

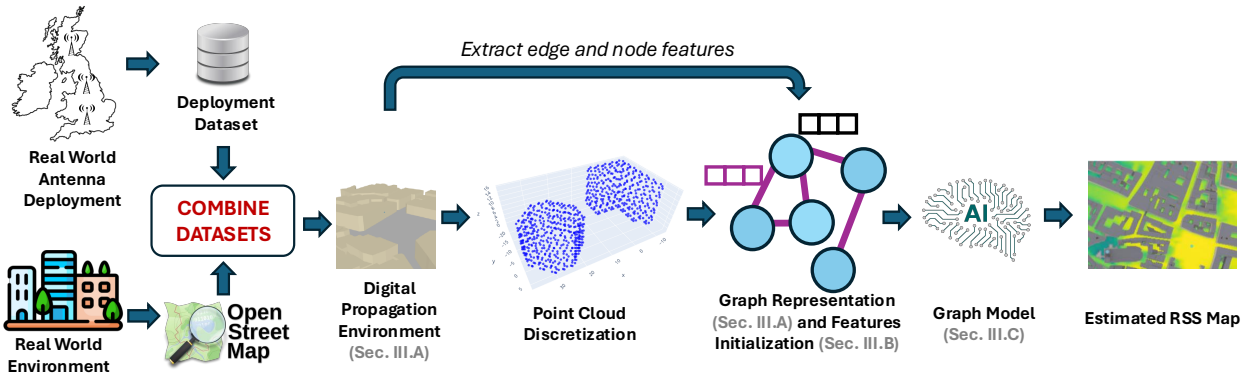


FIGURE 6. SYSTEM ARCHITECTURE OVERVIEW OF THE PROPOSED METHOD, USING VARIOUS DATA MODALITIES TO CONSTRUCT A DPE, EXTRACT AN EQUIVALENT GRAPH REPRESENTATION, AND APPLY A NEURAL TRANSFORMATION ON IT TO INFER RSS.

Figure 6 provides a schematic overview of the GraphWave system architecture, illustrating the sequential steps from constructing a digital propagation environment (DPE) that captures the 3D scene with buildings, transmitters, and receivers, to extracting a graph representation where these elements become interconnected nodes and edges, and finally applying a neural transformation to infer received signal strength (RSS) at each receiver, thereby mimicking the physical principles of ray tracing within a unified graph-driven framework.

Digital Propagation Environment (DPE): We begin by constructing a digital replica of the real-world radio environment. This DPE includes all relevant physical objects (e.g., buildings), a transmitting antenna, and a set of receivers. Each building object is represented as a collection of triangular meshes in 3D space, capturing the geometry and spatial relationships of the scene. The transmitter is defined by its orientation, configuration parameters (e.g., polarization, number of antennas, radiation pattern, and operating frequency), and the receivers are positioned at arbitrary locations within the DPE.

Graph Representation: We convert the DPE into a point cloud, where each point corresponds to a node in a graph. Edges are established between nodes to simulate the possible paths that electromagnetic (EM) waves can take, mimicking the ray-tracing process. Specifically, we connect the transmitter node to all building nodes, link all building nodes to the receiver node (to represent first-order reflections), and add a direct link from the transmitter to the receiver (to account for line-of-sight propagation).

Node and Edge Features: Each node and edge in the graph are assigned a set of features that encode both physical and network-related properties. For nodes, these include the node type (transmitter, building, or receiver), orientation angles relative to the ground and antenna, and height. For edges, features include the edge type, distance between nodes, phase shift (real and imaginary parts), free space path loss, angles relative to the ground and antenna, and radiation pattern values where applicable. These features enable the GNN to learn the complex interactions governing radio propagation.

Neural Message-Passing Model: We implement a GNN that performs message passing over the constructed graph [4]. The GNN processes node and edge features through multiple message-passing layers, updating hidden states based on information from neighboring nodes and edges. The final readout layer estimates the target radio quantity (e.g., RSS) at each receiver. The model is trained to minimize the mean absolute error (MAE) between predicted and ground truth values, using either synthetic data from ray-tracing simulations or real-world measurements.

In **Annex 2** we provide the implementation details of the three main components of GraphWave: the digital propagation environment (DPE), the graph representation, and the neural message-passing model.

3.2 Evaluation Results

Proof-of-Concept with Ray-Tracing Data: We first evaluated GraphWave using synthetic data generated by a state-of-the-art ray-tracing simulator (Sionna RT). We constructed DPEs for various antenna configurations and simulated the

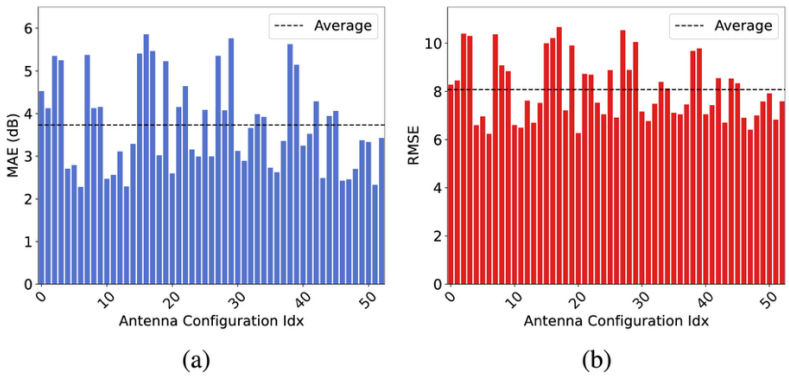


FIGURE 7. (A) MAE SAMPLE DISTRIBUTION. (B) RMSE SAMPLE DISTRIBUTION. THE X-AXIS INDICATES THE ANTENNA CONFIGURATION INDEX FROM THE 53 DIFFERENT ANTENNA CONFIGURATIONS USED FOR TESTING.

RSS for a set of receivers. GraphWave was trained on a subset of these configurations and tested on unseen ones. Our results demonstrated that GraphWave can accurately replicate the output of the ray-tracer, achieving an MAE of 3.73 dB (with a range of errors from 2.28 to 5.86 dB) across different antenna configurations. The inference time for GraphWave was, on average, 1.03 seconds, compared to 1.86 seconds for Sionna RT. This gap, though small due to the DPE size, highlights the potential for GNN-based methods to deliver fast and accurate predictions, especially as the complexity of the simulation increases.

Figure 7 demonstrates robust performance across different antenna configurations by showing the distribution of mean absolute error (MAE) and root mean squared error (RMSE) between GraphWave's predictions and ground truth ray-tracing results.

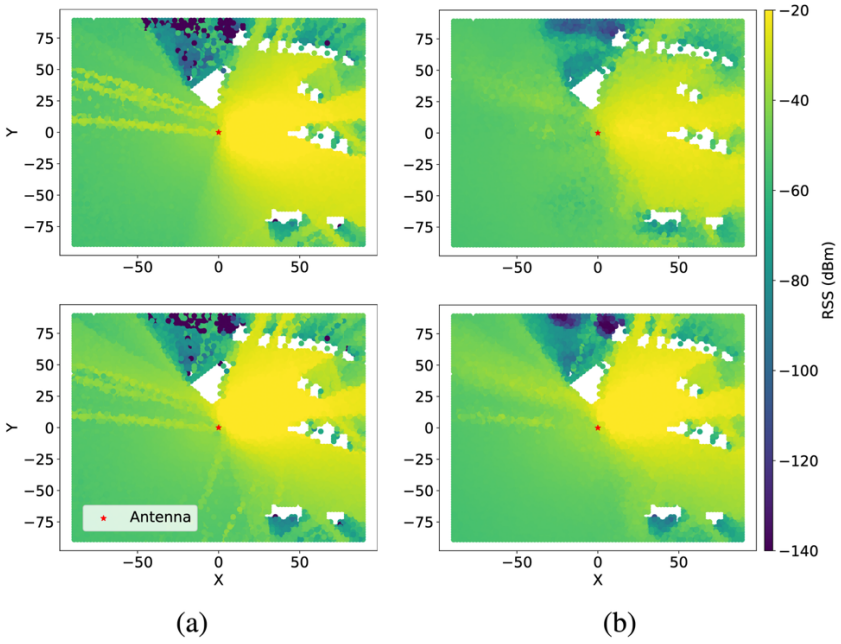


FIGURE 8. (A) GROUND TRUTH RSS. (B) RSS PREDICTED BY GRAPHWAVE.

Additionally, Figure 8 visually compares the ground truth RSS with GraphWave's predictions for scenarios with the highest and lowest MAE, highlighting the model's ability to accurately estimate RSS in most receiver positions, thereby mimicking the physical principles of ray tracing within a unified graph-driven framework.

High-Fidelity RSS Inference with Real-World Data: To validate the practical utility of GraphWave, we trained the model on real-world, crowdsourced measurements from a major mobile network operator in the UK. We selected measurements from two cities, constructed the corresponding DPEs, and evaluated the model's performance on held-out test sets. GraphWave outperformed both the Sionna ray-tracer and an empirical Ericsson channel model.

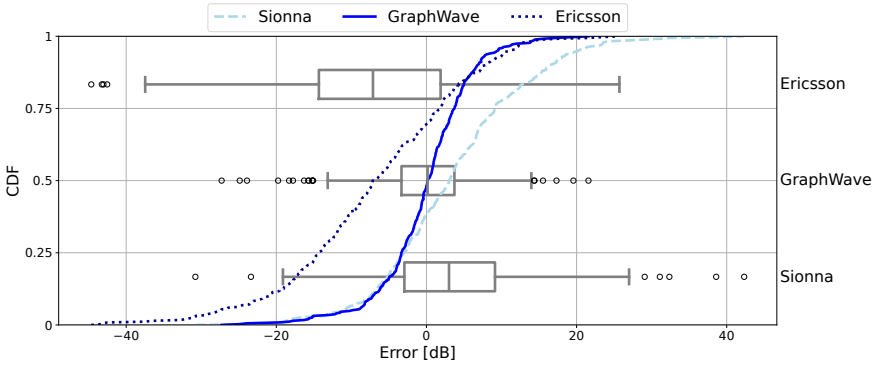


FIGURE 9. RESULTS WITH REAL-WORLD MEASUREMENTS ON CITY A.

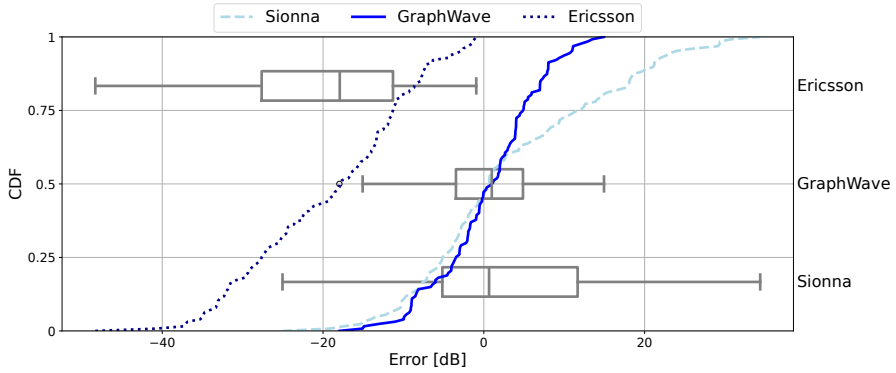


FIGURE 10. RESULTS WITH REAL-WORLD MEASUREMENTS ON CITY B.

Figure 9 and Figure 10 present the comparative results for real-world measurements in two different cities (City A and City B, respectively), showing both the cumulative distribution function (CDF) of prediction errors and boxplots that highlight the accuracy and bias of GraphWave compared to conventional models such as Sionna and the Ericsson channel model. The figures demonstrate that GraphWave's predictions are centered around zero error for most samples, outperforming the baselines.

Specifically, for over 90% of the samples, the error between GraphWave's predictions and the real-world measurements was within ±10 dB. Moreover, GraphWave's predictions were more centered around zero error, indicating less bias compared to the baselines, which tended to underestimate or overestimate the RSS.

4. Summary and Conclusions

In this report, we presented our work towards achieving the main goals of the SORUS-RIS project: to design, implement, and validate an advanced, data-driven framework for the automated deployment of Reconfigurable Intelligent Surfaces (RIS) in cellular networks, with a particular focus on Industry 4.0 scenarios and the integration of AI/ML for orchestration and optimization.

Through a comprehensive approach that combines site-specific ray tracing, user clustering, and ray-based heuristics, we have demonstrated that RIS deployment can substantially improve wireless coverage in dense urban environments—especially in high-density outage regions where conventional solutions fall short. Our results show that, by leveraging data-driven strategies for RIS placement and configuration, it is possible to recover a significant fraction of outage user equipment (UEs) and enhance overall network performance across 4G, 5G, and 6G layers. However, achieving large-scale gains requires deploying a large number of RIS units, highlighting the trade-off between performance and cost. The proposed data-driven framework, validated with both synthetic and real-world data, provides a scalable and automated approach to RIS deployment, supporting network operators in optimizing coverage and infrastructure investment.

In parallel, the development and validation of the GraphWave neural radio propagation model represent a significant advancement in the field, attempting to answer whether state-of-the-art AI models compare with ray tracing tools such as Sionna. By accurately inferring key radio quantities in complex 3D environments using both synthetic and real-world data, GraphWave outperforms conventional empirical and deterministic models in terms of accuracy, computational efficiency, and adaptability. This AI-driven approach not only bridges the gap between simulation and real-world deployment but also lays the foundation for future agile and cost-effective network planning tools.
However, we acknowledge challenges related to the availability of accurate geometric and material information for real-world environments, as well as scalability limitations due to GPU memory constraints. Despite these challenges, GraphWave represents a significant step forward in the development of physics-inspired, AI-driven radio propagation tools, paving the way for more efficient and reliable network planning and optimization.
We note that GraphWave can further be extended to answer "what-if" questions on the optimal placement of RIS in production networks, thus building agile frameworks for MNOs towards network planning.

Both the RIS deployment framework and the GraphWave model have been rigorously validated using synthetic simulations and real-world measurements, confirming their effectiveness and practical relevance. The open sourcing of our implementation will further support reproducibility and foster ongoing research in the community.

The academic papers corresponding to the two technical sections we presented in this report are currently under submission in venues relevant within the research community. Upon publications, we commit to also openly publish the code associated to the work we presented in Section

Annex 1. Code and methodology consideration for Data-Driven Deployment of Reconfigurable Intelligent Surfaces in Cellular Networks

Strongest Ray Selection Algorithm

For each cluster, the strongest incoming ray at the centroid is identified, and candidate RIS deployments are evaluated along the corresponding reflection points. The RIS is configured with an optimized phase profile, and the resulting RSRP is computed at the centroid. The configuration that yields the maximum improvement over the baseline is selected. A cluster is deemed RIS-effective if at least 60% of its users exhibit RSRP enhancement. This threshold was determined empirically to balance optimization performance and computational efficiency. The pseudo code for this algorithm is provided in the figure below.

Algorithm 1 Strongest Ray Selection.

Input : Centroid of each cluster, UEs $\mathcal{U}_k$
Output: Optimized RIS location for each cluster
for *each centroid* **do**
  Identify the set of received rays;
  Determine strongest ray;
  Extract reflection locations $\{L_k\}$
  Initialize $P^{\text{RIS}}_{\max,k} = 0$
  for $k = 1$ *to* 4 *in* $\{L_k\}$ **do**
    Deploy RIS at L_k
    Run BS beamforming optimization
    Run RIS phase shift configuration
    Compute P^{RIS}_k
    if $P^{RIS}_k > P^{RIS}_{\max,k}$ **then**
      Update $P^{\text{RIS}}_{\max,k} \leftarrow P^{\text{RIS}}_k$ Update $L_{\text{RIS}} \leftarrow L_k$
    end
  end
  if $P^{RIS}_{\max,k} \leq P_k$ **then**
    Execute All-ray Selection Algorithm **Continue**
  end
  Deploy RIS at the optimal location L_{RIS}
  for *each user* $u \in \mathcal{U}_k$ **do**
    Align RIS phase shift toward each u
    Compute P^{RIS}_u
  end
  if $\frac{1}{|\mathcal{U}_k|} \sum_{u \in \mathcal{U}_k} 1(P^{RIS}_u > P_u) > 0.6$ **then**
    Classify as a RIS-effective cluster
  end
  else
    Execute All-ray Selection Algorithm
  end
end

RIS phase shift configuration

Given a BS location, a UE location, and a RIS location, the spatial modulation function is computed using the spatially consistent reradiation model. This function defines the phase gradient needed over the RIS surface to redirect the wave from the BS to the UE.

Algorithm 2 RIS Phase Configuration

Input : $L_{\text{tx}}, L_{\text{rx}}$, Reflection locations $\{L_k\}$
Output: Optimized $\boldsymbol{\Gamma}(x', y')$
for $k = 1$ *to* 4 **do**
  if L_k *exists* **then**
    Deploy RIS at L_k
    if $k = 1$ **then**
      $\hat{K}_i \leftarrow$ from L_{tx} to L_k
    else
      $\hat{K}_i \leftarrow$ from $L_{\text{k-1}}$ to L_k
    end
    $\hat{K}_r \leftarrow$ from L_{k} to L_{rx}
    Configuring RIS Phase shift $\leftarrow$ Eq. (5)
    RIS elements amplitude $\boldsymbol{A}_m(x', y') = 1$
    Derive RIS reflection coefficient matrix $\boldsymbol{\Gamma}(x', y')$
  end
end

BS Beamforming

The serving BS must select the beam that maximizes the RSRP at the RIS location. This procedure is iterated over the available DFT beams, and the one with the highest directional gain is selected.

Algorithm 2 BS Beamforming Optimization

Input : L_{RIS}, Cell t_{serving}, Number of beams M
Output: Optimal Beam Index m^*
Deploy RIS at L_{RIS}
Set initial beam index $m^* \leftarrow [0, 0]$
for $m \in \mathcal{M}$ **do**
  Compute P_m^{RIS}
  if $P_m^{RIS} > P_{m^*}^{RIS}$ **then**
    Update $m^* \leftarrow m$
  end
end
Select optimal beam index m^* for RIS deployment

Annex 2. Code and methodology considerations for GraphWave

In this section we provide implementation details of the three main components of GraphWave: the digital propagation environment (DPE), the graph representation, and the neural message-passing model.

Digital propagation environment

We build a digital propagation environment per antenna in our dataset. The code snapshot below indicates the steps we take to iterate over each antenna from our dataset and we extract the 3D geometric scene, the number of buildings and the ground plane size. Each loop iteration results in one digital environment directory per antenna, ready to be simulated using Sionna or converted to a point cloud.

```
# Iterate over each unique tuple and filter the df dataframe to obtain the rows with the same cell_id and csr
for index, row in unique_tuples.iterrows():
    cell_id = row['cell_id']
    csr = row['csr']
    # Check if the cell_id and csr combination has already been visited
    subset_cell_id_csr = df.loc[(df['cell_id'] == cell_id) & (df['csr'] == csr)]
    azimuth = subset_cell_id_csr['azimuth'].values[0]
    power = subset_cell_id_csr['power'].values[0]
    elec_tilt = subset_cell_id_csr['elec_tilt'].values[0]
    tilt = subset_cell_id_csr['tilt'].values[0]
    ant_height = subset_cell_id_csr['ant_height'].values[0]
    latitude = subset_cell_id_csr['latitude'].values[0]
    longitude = subset_cell_id_csr['longitude'].values[0]

    # Obtain latitude and longitude from subset_cell_id_csr
    latitude = subset_cell_id_csr['latitude'].values[0]
    longitude = subset_cell_id_csr['longitude'].values[0]

    # Compute the bounding box of the centroid of the tile
    max_lat, min_lat, max_lon, min_lon = get_bounding_box(latitude, longitude, args.dist)

    ground_material = "itu_concrete" # "itu_very_dry_ground"
    buildings_material = "itu_concrete"

    try:
        # Download the dae file corresponding to the surrounding box of the centroid coordinates
        number_buildings, ground_plane_size_x, ground_plane_size_y = download_dae_files_and_export_2_mitsuba(max_la
    except:
        list_cells_with_errors.append(cell_dir)
        os.system("rm -rf "+cell_dir)
        continue
```

Once the digital environment is downloaded, the next step is to discretize the scene and convert it to a point cloud. This is done following the steps below. The main for loop iterates over all the faces of all the geometries in the 3D scene. Then, each face is triangulated and for each triangle we create a point cloud that intersects with the triangle plane. Finally, in the point_cloud array we store the coordinates of all the points for the current scene. This process is repeated for all scenes, a necessary pre-computation step before building the graph.

```
# Code below is to create a grid of points and add them to the point cloud
# Divide the face into triangles (even for quads/ngons)
for loop in bmesh.ops.triangulate(bm, faces=[face])['faces']:
    # Get the vertices of the triangle
    v0, v1, v2 = [v.co for v in loop.verts]

    # Generate a 2D grid of points within the triangle's bounding box
    points = grid_points_in_triangle(v0, v1, v2, spacing)

    # # Generate randomly distributed points within the triangle
    # points = random_distribution_in_triangle(v0, v1, v2, num_points_per_side)

    # Add the points and their associated normal to the point cloud as an array
    for point in points:
        point_cloud.append([point.x, point.y, point.z, normal.x, normal.y, normal.z])
```

Graph representation

Once the point cloud is created for the dataset, we proceed to build the graph representation of the scene. The snapshot below shows the implementation details of the main class GraphScene, used to convert each scene, together with the receivers, antenna and buildings, to a graph representation. The key functions to build the graph are add_antenna_node, add_building_node, add_coverage_node, connect_antenna_with_building_nodes, connect_antenna_with_coverage_node and connect_building_nodes_with_coverage_node. The first three are used to add the antenna; the building point clouds, and the receiver to the graph representation. The latter ones are used to connect the nodes of the graph between them.

```
21  class GraphScene:
22      def __init__(self, antenna_info):…
48
49      def visualize_antenna_radio_pattern(self, scene_dir, antenna_info, train_test_dir_suffix):…
185
186     def add_antenna_node(self, row, antenna_info, scene_dir, train_test_dir_suffix):…
203
204     def add_building_node(self, row):…
218
219     def add_coverage_node(self, row):…
234
235     def store_antenna_pattern_intersection(self, node_type, src, dst, intersec, train_test_dir_suffix, mitsuba_dataset_dir, dire):…
303
304     def connect_antenna_with_building_nodes(self, train_test_dir_suffix, mitsuba_dataset_dir, dire, antenna_info, dict_fspl_distance):
379
380     def connect_antenna_with_coverage_node(self, train_test_dir_suffix, dict_fspl_distance):…
458
459     def connect_building_nodes_with_coverage_node(self, antenna_info, dict_fspl_distance):…
503
504     def delete_node(self, node):…
510
511     def fill_edges_states(self):…
518
519     def print_building_vertices(self, train_test_dir, dire_dist_suffic, df_znorm, dist_param):…
593
594     def check_edge_in_file(self, origin_pos, dest_pos):…
606
607     def connect_building_vertices(self, mitsuba_dataset_dir, dire, num_building_nodes, dist_param):…
644
645     def add_building_vertices_to_graph(self, train_test_dir, mitsuba_dataset_dir, dire, antenna_info, dist_param, dire_dist_suffic):…
```

Once the graph is built, the next step is to extract the indices of the adjacency matrix, store them to a file, initialize the edge features and store the entire graph structure to a file. Note that the node features are initialized during the creation of the graph. The code below indicates the steps to perform.

```
# Get the adjacency matrix of the graph
adjacency_matrix = nx.adjacency_matrix(network_topology.Graph)
# Obtain pairs of nodes that are connected
first, second = np.nonzero(adjacency_matrix)
network_topology.first = first
network_topology.second = second

# Store first and second in a file
np.save(directory+"first_nodes_"+mode+".npy", network_topology.first)
np.save(directory+"second_nodes_"+mode+".npy", network_topology.second)

# Fill the edges states
network_topology.fill_edges_states()

feature = serialize_node_tuple_list(network_topology.nodes_states)
serialize_edge_tuple_list(network_topology.edges_states, feature)

example_proto = tf.train.Example(features=tf.train.Features(feature=feature))
writer.write(example_proto.SerializeToString())
```

Neural message-passing model

The last step is to implement the GNN that will process the graph created in the step before. We implement a GNN that performs message passing over the constructed graph [4]. In the snapshot below we show the code implementation of our GNN using Tensorflow. Note that the function process_single_sample is applied for each sample within the batch and the for loop corresponds to the message passing step [4]. After the message passing, we aggregated the hidden states of all the nodes and apply a readout function to obtain the final RSS for a given receiver.

```
@tf.function
def call(self, inputs):
    first = inputs['first_tensor']
    second = inputs['second_tensor']
    node_state = inputs['concat_node_features']
    edge_state = inputs['edge_features']

    num_nodes = tf.shape(node_state)[1]

    def process_single_sample(inputs):
        first_sample, second_sample, node_state_sample, edge_state_sample = inputs

        for hh in range(self.hparams['T']):
            mainNodes = tf.gather(node_state_sample, first_sample, batch_dims=0)
            neighNodes = tf.gather(node_state_sample, second_sample, batch_dims=0)

            nodesConcat = tf.concat([mainNodes, neighNodes, edge_state_sample], axis=-1)

            outputs = self.Message(nodesConcat)

            nodes_inputs = tf.math.unsorted_segment_sum(data=outputs, segment_ids=second_sample, num_segments=num_nodes)

            outputs, node_state_sample = self.Update(nodes_inputs, [node_state_sample])
            node_state_sample = node_state_sample[0]

        # The code below corresponds to summing the node states and apply readout on the sum
        node_state_sum = tf.reduce_sum(node_state_sample, axis=0)
        # Reshape to (1, 5)
        reshaped_tensor = tf.expand_dims(node_state_sum, axis=0)
        # Normalize the final node state
        reshaped_tensor = self.norm(reshaped_tensor)

        r = self.Readout([reshaped_tensor])
        return r

    # Apply the process_single_sample function to each element in the batch
    results = tf.vectorized_map(process_single_sample, (first, second, node_state, edge_state))
    return results
```

References

[1] Liu, Y., Liu, X., Mu, X., Hou, T., Xu, J., Di Renzo, M., & Al-Dhahir, N. (2021). Reconfigurable intelligent surfaces: Principles and opportunities. *IEEE communications surveys & tutorials*, *23*(3), 1546-1577.

[2] Hoydis, J., Aït Aoudia, F., Cammerer, S., Nimier-David, M., Binder, N., Marcus, G., & Keller, A. (2023, December). Sionna RT: Differentiable ray tracing for radio propagation modeling. In *2023 IEEE Globecom Workshops (GC Wkshps)* (pp. 317-321). IEEE.

[3] OpenStreetMap contributors, Planet dump retrieved from https://planet.osm.org, https://www.openstreetmap.org, 2017.

[4] J. Gilmer et al., "Neural message passing for quantum chemistry," in International conference on machine learning, PMLR, 2017, pp. 1263–1272.

[5] Zhang, Tian, Raghu Ramakrishnan, and Miron Livny. "BIRCH: A new data clustering algorithm and its applications." *Data mining and knowledge discovery* 1 (1997): 141-182.

[6] Teltonika Networks Wiki, "RSRP and RSRQ." [Online]. Available: https://wiki.teltonika-networks.com/view/RSRP_and_RSRQ