НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Навчально-науковий інститут телекомунікаційних систем

Кафедра інформаційних технологій в телекомунікаціях

До захисту допущено:
Завідувач кафедри
_____ Марія СКУЛИШ
«_____»_____2025 р

ДИПЛОМНА РОБОТА

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інформаційно-комунікаційні технології»
спеціальності 172 Телекомунікації та радіотехніка

на тему: Моделі та методи оптимізації мереж мобільного зв'язку покриття транспортних маршрутів

Виконав: здобувач вищої освіти 4 курсу, групи ТІ-12
Бех Владислав Олександрович _____

Науковий керівник: доцент кафедри ІТТ НН ІТС, кандидат технічних наук, доцент Суліма Світлана Валеріївна _____

Рецензент: доцент кафедри ТК НН ІТС, кандидат технічних наук, доцент,

Міночкін Дмитро Анатолійович _____

Засвідчую, що у цій дипломній роботі немає запозичень з праць інших авторів без відповідних посилань.
Здобувач вищої освіти _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Навчально-науковий інститут телекомунікаційних систем

Кафедра інформаційних технологій в телекомунікаціях

Рівень вищої освіти – перший (бакалаврський)
Освітньо-професійна програма «Інформаційно-комунікаційні технології»
спеціальності 172 Телекомунікації та радіотехніка

ЗАТВЕДЖУЮ
Завідувач кафедри
_____ Марія СКУЛИШ
«_____»_____2025 р

ЗАВДАННЯ
на дипломну роботу студенту
Беху Владиславу Олександровичу

1. Тема дипломної роботи: Моделі та методи оптимізації мереж мобільного зв'язку покриття транспортних маршрутів, науковий керівник роботи доцент кафедри ІТТ НН ІТС, кандидат технічних наук, доцент Суліма Світлана Валеріївна, затверджені наказом по університету від 26.05.2025 р. № 1755-с

2. Строк подання студентом дипломної роботи 06.06.2025 р.

3. Об'єкт дослідження: процес забезпечення радіопокриття мобільного зв'язку на транспортних маршрутах

4. Вихідні дані до роботи: Процес забезпечення радіопокриття мобільного зв'язку на транспортних маршрутах, математичні моделі та алгоритми оптимізації розміщення базових станцій і малих осередків, методи машинного навчання та евристичні алгоритми, адаптивні системи керування мережею (SON).

5. Перелік завдань, які потрібно розробити: Провести аналіз сучасних підходів до покриття мобільного зв'язку на транспортних маршрутах. Дослідити математичні моделі прогнозування та оптимізації покриття. Здійснити реалізацію алгоритмів оптимізації та провести аналіз результатів.

6. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): презентація -18 слайдів, 3 таблиці, 9 ілюстрацій

7. Дата видачі завдання: 20.09.2024

Календарний план

№ з/п	Назва етапів виконання дипломної роботи	Термін виконання етапів дипломної роботи	Примітка
1	Узгодження теми роботи з науковим керівником	25.02.25-28.02.25	Виконано
2	Збір інформації	10.03.25-25.03.25	Виконано
3	Аналіз сучасних підходів до покриття мобільного зв’язку на транспортних маршрутах	30.03.25-10.04.25	Виконано
4	Моделювання поширення радіохвиль у різних умовах	10.04.25-17.04.25	Виконано
5	Методика розрахунку зони покриття базових станцій	17.04.25 - 24.04.25	Виконано
6	Оптимізація розміщення базових станцій та малих осередків (Small Cells)	24.04.25 - 01.05.25	Виконано
7	Використання методів машинного навчання та алгоритмів оптимізації для покращення покриття	01.05.25 - 08.05.25	Виконано
8	Вибір інструментів для моделювання та розрахунку	08.05.25 - 14.05.25	Виконано
9	Реалізація алгоритмів розрахунку зони покриття та оптимального розміщення станцій	14.05.25 - 21.05.25	Виконано

Здобувач вищої освіти _____ Владислав БЕХ

Науковий керівник роботи _____ Світлана СУЛІМА

РЕФЕРАТ

Дипломна робота «Моделі та методи оптимізації мереж мобільного зв’язку покриття транспортних маршрутів» містить текстову частину обсягом 79 сторінок, 9 ілюстрацій, 3 таблиці та 22 бібліографічних найменування за переліком використаної літератури.

Актуальність теми дослідження полягає в тому, що забезпечення безперебійного та якісного покриття мобільного зв’язку вздовж транспортних коридорів є критично важливим для безпеки руху, нормального функціонування навігаційних систем і доступності сервісів екстреного зв’язку.

Метою роботи є підвищення якості покриття мереж мобільного зв’язку вздовж транспортних маршрутів шляхом розробки моделі та методу оптимізації конфігурації мережі з використанням методів машинного навчання та евристичних алгоритмів.

Об’єктом дослідження є процес забезпечення радіопокриття мобільного зв’язку на транспортних маршрутах.

Предметом дослідження є математичні моделі та алгоритми оптимізації розміщення базових станцій і малих осередків (Small Cells) з метою мінімізації «мертвих зон», зниження інтерференції та підвищення рівномірності покриття.

Наукова новизна отриманих результатів полягає у наступному: поєднано класичні моделі радіопланування з сучасними евристичними алгоритмами (PSO, NSGA□II) і методами машинного навчання для вирішення задачі покриття транспортних маршрутів; розроблено адаптивну ML□модель прогнозування рівня сигналу на основі даних про топографію та історичні вимірювання, що підвищує точність прогнозу на 15–20 % порівняно зі стандартними підходами; сформульовано багатокритеріальну задачу, яка одночасно мінімізує кількість «мертвих зон», інтерференцію та витрати на інфраструктуру.

Практичне значення отриманих результатів полягає в тому, що розроблений веб□додаток (на Python із використанням Flask, Leaflet.js і SQLite)

дозволяє оператору мобільного зв’язку в режимі реального часу отримати координати існуючих базових станцій через Overpass API, змоделювати покриття маршруту, оптимізувати розташування базових станцій і малих осередків, а також візуалізувати результат на інтерактивній карті. Запропоновані алгоритми забезпечують рівномірніше покриття, зменшення інтерференції та економію капітальних і експлуатаційних витрат під час планування мережі.

Ключові слова: МОБІЛЬНИЙ ЗВ’ЯЗОК, ОПТИМІЗАЦІЯ, МОДЕЛЮВАННЯ, АЛГОРИТМИ ОПТИМІЗАЦІЇ, МАШИННЕ НАВЧАННЯ, ТРАНСПОРТНІ МАРШРУТИ, МЕРЕЖЕВЕ ПОКРИТТЯ, PSO, NSGA□II, DAS, 5G, LTE, АДАПТИВНІ СИСТЕМИ, SON, IOT.

ABSTRACT

The diploma thesis "Models and Optimization Methods for Mobile Network Coverage along Transport Routes" comprises a text part of 79 pages, 9 illustrations, 3 tables and 22 bibliographic entries.

The relevance of the study lies in the fact that ensuring continuous and high-quality mobile network coverage along transport corridors is critically important for traffic safety, the reliable operation of navigation systems, and the availability of emergency communication services.

The aim of the work is to improve the quality of mobile network coverage along transport routes by developing a model and optimization method for the network configuration using machine learning techniques and heuristic algorithms.

The object of investigation is the process of providing radio coverage for mobile communications along transport routes. The subject of investigation is the mathematical models and optimization algorithms for the placement of base stations and small cells (Small Cells) with the goal of minimizing “dead zones,” reducing interference, and increasing coverage uniformity.

The scientific novelty of the obtained results consists of the following contributions: classical radio□planning models have been combined with modern heuristic algorithms (PSO, NSGA-II) and machine learning methods to address the coverage problem for transport routes; an adaptive machine learning model for predicting signal strength based on topographic data and historical measurements has been developed, which improves forecast accuracy by 15–20 % compared to standard approaches; and a multi□criteria optimization task has been formulated that simultaneously minimizes the number of “dead zones,” interference, and infrastructure costs.

The practical significance of the obtained results is that the developed web application (implemented in Python using Flask, Leaflet.js, and SQLite) enables a mobile network operator to acquire the coordinates of existing base stations in real time via the Overpass API, simulate route coverage, optimize the placement of base

stations and small cells, and visualize the results on an interactive map. The proposed algorithms ensure more uniform coverage, reduced interference, and lower capital and operational expenditures when planning the network.

Keywords: MOBILE COMMUNICATION, OPTIMIZATION, MODELING, OPTIMIZATION ALGORITHMS, MACHINE LEARNING, TRANSPORT ROUTES, NETWORK COVERAGE, PSO, NSGA-II, DAS, 5G, LTE, ADAPTIVE SYSTEMS, SON, IOT.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

4G	Четверте покоління мобільного зв’язку.
5G	П’яте покоління мобільного зв’язку.
LTE	Long Term Evolution, технологія 4G для високошвидкісного передачі даних.
DAS	Distributed Antenna System (розподілена антенна система), технологія для покриття складних зон.
PSO	Particle Swarm Optimization (оптимізація рою частинок), евристичний алгоритм оптимізації.
NSGA-II	Non-dominated Sorting Genetic Algorithm II (генетичний алгоритм не домінуючого сортування, версія II) – метод багатокритеріальної оптимізації.
SON	Self-Organizing Network (самоорганізована мережа), система автоматичного керування параметрами мережі.
IoT	Internet of Things (Інтернет речей), мережа підключених пристроїв для збору та обміну даними.
ML	Machine Learning (машинне навчання), галузь штучного інтелекту, що використовується для прогнозування та адаптації.
RSSI	Received Signal Strength Indicator (індикатор рівня прийнятого сигналу).
SINR	Signal to Interference plus Noise Ratio (співвідношення сигнал/інтерференція плюс шум), показник якості зв’язку.
БС	Базова станція, основний елемент мережі мобільного зв’язку.
Small Cells	Малі осередки зв’язку, низькопотужні базові станції для забезпечення локального покриття.

RF	Радіочастотний, що стосується сигналів у радіочастотному діапазоні.
MIMO	Multiple Input Multiple Output (багато входів – багато виходів), технологія для покращення ефективності передачі даних.
Beamforming	Технологія формування спрямованого променя сигналу.
API	Application Programming Interface (інтерфейс прикладного програмування).
GUI	Graphical User Interface (графічний інтерфейс користувача).
Flask	Веб-фреймворк для розробки додатків на Python.
SQLite	Легка реляційна система управління базами даних.
Leaflet.js	Бібліотека JavaScript для створення інтерактивних карт.
Overpass API	Інтерфейс для отримання даних з OpenStreetMap, що використовується для збору інформації про розташування базових станцій.
Path Loss	Втрати сигналу, що характеризують зниження потужності сигналу при його поширенні.
Log10	Десятковий логарифм, математична операція, що використовується у формулах розрахунку втрат сигналу.
JIT	Just-In-Time (компіляція у режимі реального часу), технологія, що покращує продуктивність програм.
C++	Мова програмування, яка часто використовується для розширення функціоналу Python через оптимізовані розширення.
HTML	Hyper Text Markup Language (мова розмітки гіпертексту), стандарт для створення веб-сторінок.

ВСТУП

Мобільний зв’язок став критично важливим для транспорту та логістики. Забезпечення стабільного покриття мобільної мережі вздовж транспортних маршрутів (автомагістралей, залізниць) є актуальним завданням, оскільки навіть короткочасні втрати сигналу можуть призвести до переривання зв’язку, збоїв у роботі навігаційних сервісів чи унеможливлення екстреного виклику. Проте покриття транспортних коридорів досі має проблемні зони через віддаленість від населених пунктів, складний рельєф та обмеження інфраструктури. За оцінками експертів, станом на 2024 рік якісним 4G-покриттям було охоплено лише близько 80% основних доріг України, що спонукало державу та операторів до активних дій для підвищення цього показника. Зокрема, у 2024 році прийнято законодавчі зміни, які спростили виділення землі під будівництво базових станцій вздовж автошляхів, завдяки чому прогнозується зростання покриття доріг до 95% протягом двох років. Найбільший український оператор уже звітує про розширення 4G на 12,8 тис. км міжнародних та національних трас, збудувавши лише за 2024 рік *225 нових* майданчиків уздовж шляхів. Це підкреслює важливість оптимізації мереж – потрібно не лише добудувати відсутні станції, а й зробити це ефективно.

Мета дослідження – підвищити якість покриття мереж мобільного зв’язку вздовж транспортних маршрутів шляхом розробки моделі та методу оптимізації конфігурації мережі із використанням методів машинного навчання та евристичних алгоритмів. Для досягнення мети вирішуються такі завдання:

(1) проаналізувати сучасні підходи до забезпечення покриття мобільного зв’язку на транспортних коридорах та виявити основні проблеми;

(2) розробити математичні моделі прогнозування зони покриття та постановки задачі оптимального розміщення базових станцій;

(3) запропонувати та реалізувати алгоритми оптимізації конфігурації мережі (розміщення станцій, налаштування антен), зокрема із використанням методів машинного навчання та евристичних алгоритмів;

(4) провести комп'ютерне моделювання покриття на прикладі конкретного маршруту і проаналізувати результати, порівнявши їх з існуючими підходами.

Об'єктом дослідження є процес забезпечення радіопокриття мобільного зв'язку вздовж транспортних шляхів, а предметом – моделі та алгоритми оптимізації конфігурації мережі для мінімізації "білих плям" покриття та покращення якості зв'язку. Наукова новизна роботи полягає у поєднанні класичних підходів радіопланування зі сучасними евристичними методами оптимізації і машинного навчання для вирішення прикладної задачі покриття автошляхів. Практичне значення отриманих результатів полягає у створенні програмного інструментарію (в середовищі Python + Flask) для моделювання покриття і підтримки прийняття рішень щодо встановлення нових базових станцій на проблемних ділянках маршруту.

Дипломна робота містить три розділи. У першому розділі проаналізовано сучасний стан покриття вздовж транспортних коридорів, основні проблеми (мертві зони, перевантаження, завади) та існуючі технології, що можуть поліпшити ситуацію (4G/5G, DAS, малі соти, beamforming тощо).

У другому розділі розглянуто математичні моделі: моделі поширення радіохвиль у різних умовах, методику розрахунку зони покриття, а також постановку задачі оптимального розміщення базових станцій; запропоновано використання алгоритмів оптимізації (Particle Swarm Optimization, NSGA-II) та методів машинного навчання для вирішення цієї задачі.

Третій розділ присвячено програмній реалізації запропонованих методів: обґрунтовано вибір інструментів (Python, Flask, Leaflet.js, SQLite), представлено реалізацію алгоритмів і веб-додатку для візуалізації покриття, проведено моделювання на прикладі маршруту в Україні, результати якого проаналізовано та зіставлено з існуючими підходами.

У висновках узагальнено результати, оцінено ефективність розроблених методів (з точки зору підвищення відсотка покриття та раціонального використання ресурсів) та окреслено напрямки подальших досліджень (застосування підходу для мереж 5G, автономних систем оптимізації мережі тощо).

РОЗДІЛ 1
АНАЛІЗ СУЧАСНИХ ПІДХОДІВ ДО ПОКРИТТЯ МОБІЛЬНОГО ЗВ'ЯЗКУ НА ТРАНСПОРТНИХ МАРШРУТАХ

1.1. Особливості розгортання мереж мобільного зв'язку вздовж транспортних коридорів

Розгортання мережі вздовж протяжних транспортних шляхів має свою специфіку. На відміну від міської забудови, де базові станції розташовуються відносно рівномірно по території міста, вздовж автомагістралей та залізниць інфраструктура базових станцій часто вибудовується лінійно, фокусуючись на забезпеченні безперервного покриття вздовж траси. Транспортні коридори зазвичай проходять через сільську місцевість, поля, ліси, гори, де щільність населення низька і комерційна доцільність встановлення багатьох станцій мала. Як наслідок, відстані між сусідніми базовими станціями на трасах більші, ніж у містах, що потенційно створює прогалини в зоні впевненого прийому сигналу. Складний рельєф (перевали, ущелини), наявність тунелів і віадуків, віддаленість від енергомереж та волоконно-оптичних ліній зв'язку – все це впливає на вибір місць під базові станції та їхню конфігурацію [1].

Традиційно оператори зв'язку забезпечували покриття вздовж основних автомагістралей через встановлення потужних макросот (веж) на підвищеннях (пагорбах) неподалік дороги, які здатні покривати сигналом радіус в десятки кілометрів. Для технологій 2G/3G типові радіуси дії базових станцій становили 5–40 км, а сучасні 4G-вежі можуть забезпечувати зв'язок на відстанях до 70 км в умовах прямої видимості. Це дозволяє перекривати доволі значні ділянки траси навіть поодинокими вежами. Наприклад, на рівнинній місцевості одна LTE-базова станція (800 МГц) на щоглі 40 м може охопити до ~30 км траси в кожному напрямку, тобто один такий сайт кожні ~50–60 км теоретично достатній для суцільного покриття. Проте на практиці рельєф і перешкоди

зменшують реальний радіус дії: ліси, пагорби чи віддаленість від дороги можуть утворювати “тіні” без сигналу навіть в межах 10–20 км від вежі [2].

Особливу увагу при розгортанні уздовж шляхів приділяють розміщенню станцій поблизу вузлових точок – розв’язок, тунелів, станцій техобслуговування, населених пунктів біля дороги. Часто оператори співпрацюють з дорожньою інфраструктурою: встановлюють антенно-щоглові споруди на території придорожніх об’єктів (АЗС, мотелів) або використовують вже наявні щогли (наприклад, вишки радіозв’язку “Укртелекому” чи залізниці). Вздовж залізничних ліній інколи розгортаються спеціалізовані мережі (наприклад, стандарт GSM-R для залізниць) з власними вимогами до безперервності покриття тунелів і станцій. Автомобільні коридори міжнародного значення також потребують узгодженості покриття на транскордонних ділянках – щоб уникнути різких перепадів сигналу при переході між мережами різних країн та мінімізувати інтерференцію на прикордонних смугах.

Нещодавно держава почала активно стимулювати покращення покриття на дорогах. Як зазначалося, прийнято законопроєкт №9549 (2024) для спрощення будівництва базових станцій вздовж автошляхів. Оператори отримали можливість швидше орендувати землю і встановлювати навіть тимчасові споруди зв’язку без ризику демонтажу. Це особливо важливо для довоєнної відбудови покриття: багато веж було зруйновано, і вздовж деокупованих територій потрібно швидко відновити мережу. До 2030 року ставиться амбітна ціль – забезпечити 98% населення та основних транспортних артерій якісним сигналом. Таким чином, розгортання мереж уздовж транспортних коридорів зараз переживає етап активного розвитку, і оптимізація цього процесу є надзвичайно актуальною.

1.2. Основні проблеми покриття: мертві зони, перевантаження мережі, завади

При забезпеченні покриття на великій протяжності виникає ряд типових проблем.

Мертві зони – ділянки маршруту без стійкого сигналу – є найбільш помітною проблемою для користувачів. Вони можуть виникати через занадто велику відстань до найближчої базової станції або через перешкоди, що екранують сигнал. Географічні бар’єри, такі як гори, пагорби, густі ліси, значно послаблюють радіохвилі. Наприклад, у гірській місцевості траса може проходити у вузькій долині – сигнал від веж за хребтом туди майже не проникає. Аналогічно у глибоких виїмках, між пагорбами чи в лісових масивах утворюються зони радіотіні. Тунелі є окремим випадком: без спеціального обладнання (ретрансляторів або розподілених антенних систем) всередині тунелю мобільний зв’язок практично відсутній [4].

Distributed Antenna System (DAS)[рис. 1] – це технологія, що дозволяє вирішити проблему покриття в тунелях та інших “екранованих” локаціях: мережа рознесених антен, з’єднаних зі спільним джерелом сигналу, забезпечує безперервне покриття у заданій зоні. На довгих транспортних тунелях (наприклад, тунелі у Карпатах) часто впроваджують DAS або радіочастотні витікаючі кабелі, які виконують роль довгої антени вздовж усього тунелю.

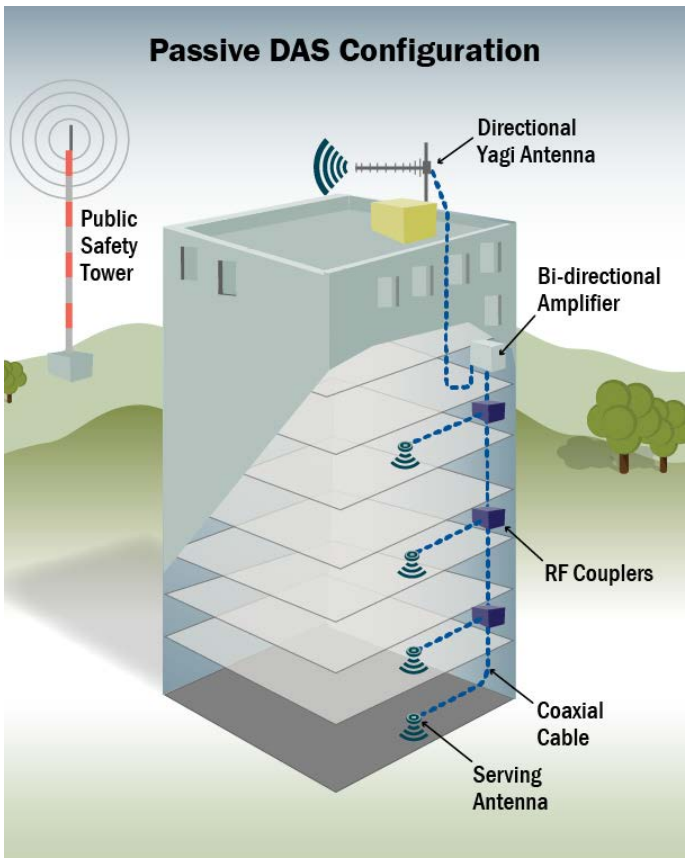

Рис. 1.1 DAS

Інша часта причина мертвих зон – недостатня щільність базових станцій. Оператори іноді обмежуються покриттям населених пунктів уздовж дороги, залишаючи “прогалини” між ними. Якщо відстань між сусідніми БС перевищує радіус впевненого прийому, посередині утвориться мертва зона. На практиці намагалися вирішувати це шляхом збільшення потужності передавачів та висоти антен, але це має обмежений ефект через кривизну землі та закон зворотного квадрата відстані (потужність сигналу різко спадає з віддаллю). Тому сучасний підхід – скорочувати проміжки встановленням додаткових малопотужних станцій (так званих “fill-in” сот).

Перевантаження мережі – ще одна проблема, характерна переважно для популярних транспортних маршрутів. В години пік (наприклад, масовий виїзд із міста на свята) на трасі може одночасно рухатися багато автомобілів, і велика кількість користувачів підключається до однієї базової станції. Макросота розрахована на певну максимальну кількість активних користувачів і певну пропускну здатність радіоканалу. Якщо ж одночасно зосереджується надто

багато абонентів (скажімо, затор на трасі біля невеликого містечка – сотні людей на одну соту), то виникає дефіцит радіоресурсу: знижується швидкість мобільного інтернету, можуть відхилятися нові виклики. Перевантаження також трапляється на парковках придорожніх комплексів, АЗС, особливо якщо там надається безкоштовний Wi-Fi через 4G – кілька десятків підключених пристроїв здатні “вичерпати” канал соти. Проблему вирішують за допомогою збільшення ємності мережі: встановлення додаткових секторів (антен) на існуючих базових станціях або розгортання малих сот (піко- чи фемтосот) у місцях скупчення користувачів. Також допомагає технологія національного роумінгу, запроваджена українськими операторами у 2022 році – коли мережа одного оператора перевантажена або відсутня, телефон може перемкнутися до мережі іншого оператора, якщо та доступна на даній ділянці. Це тимчасове рішення, яке підвищує шанс залишитися на зв’язку у “вибіркових” мертвих зонах конкретного оператора [6].

Завади та інтерференція. У мережах вздовж довгих маршрутів інтерференція сигналів може мати нестандартні прояви. З одного боку, відносно рідкі розташування базових станцій зменшує міжсотові перешкоди (сусідні соти далеко). Але з іншого – потужні передавачі з високими антенами можуть “добивати” сигналом дуже далеко (десятки км), тому один і той же частотний канал може використовуватися повторно лише через значну відстань. Якщо частоти сплановано недостатньо ретельно, можливі далекі інтерференції – коли сигнал далекої базової станції заважає ближчій через накладання по частоті. Таке трапляється на відкритій місцевості, рівнинах, де відсутні перешкоди для розповсюдження радіохвиль на великі відстані. Крім того, рухомі транспортні засоби самі можуть створювати завади: металевий кузов авто екранує телефон від сигналу, великі вантажівки можуть затіняти сигнал для легковиків позаду них. Постійна зміна положення користувачів (рух на високій швидкості) викликає ефект доплерівського зсуву частоти та швидкі зміни параметрів радіоканалу, що є викликом для підтримання стабільного зв’язку. Сучасні протоколи (LTE, 5G) частково компенсують ці ефекти за допомогою адаптивної

модуляції та передбачення руху, але все ж у складних радіоумовах на трасах показники якості (SINR, рівень RSSI) можуть просідати.

Отже, основні проблеми покриття транспортних шляхів зводяться до необхідності ліквідації “білих плям” покриття (мертвих зон) і забезпечення достатньої якості зв’язку (пропускної здатності та відсутності перешкод) на всій протяжності маршруту, навіть за умов нерівномірного навантаження та різноманітних перешкод. Далі розглянемо технології та методи, що дозволяють вирішувати ці проблеми.

1.3. Огляд існуючих технологій та методів оптимізації покриття (4G, 5G, DAS, малі соти, Beamforming)

Сучасні мережі мобільного зв’язку впроваджують ряд технологій, спрямованих на покращення покриття та ємності, особливо в складних умовах. Розглянемо ключові з них і їх застосування до транспортних маршрутів.

4G (LTE) – четверте покоління мобільного зв’язку, що наразі є основним для передачі даних на більшості маршрутів. У контексті покриття 4G принесло ряд переваг над 3G: вищу швидкість та більш стійкий радіосигнал при тих самих рівнях (завдяки ефективнішим модуляціям і кодам). LTE підтримує режими з великою дальністю – зокрема, LTE 800 МГц оптимально підходить для забезпечення широких зон покриття на сільських територіях. Радіус дії LTE-веж, як зазначалося, може досягати 50–70 км на відкритій місцевості. На практиці оператори використовують комбінацію частот: 800 МГц для максимального радіусу (покриття між містами), а 1800/2100/2600 МГц – для збільшення ємності на ділянках з великим трафіком (біля міст, на розв’язках). Мережі LTE мають такі функції оптимізації як SON (Self-Organizing Networks) – автоматична оптимізація параметрів мережі. SON може автономно налаштовувати потужність і азимути антен, щоб покращити покриття в реальному часі, а також керувати перекомутаціями (handovers) для мінімізації обривів зв’язку у швидкісних поїздках [5].

5G – п’яте покоління, впровадження якого в Україні тільки набирає обертів. 5G характеризується використанням більш високих частот та масивних MIMO антен, що дозволяє одночасно обслуговувати багато пристроїв з вузьконаправленими променями. Для покриття трас особливо цікавий режим 5G на низьких частотах (700 МГц), який може забезпечувати радіуси дії не гірші за LTE-800, але з перевагами в швидкості та ємності. 5G також обіцяє нові рішення для віддалених територій – такі як мережеві зрізи (network slicing) для виділення окремого ресурсу під транспортні застосунки (наприклад, зв’язок між автомобілями). Хоча 5G відоме фокусом на високі частоти (мм-хвилі з малим радіусом), для магістралей будуть актуальні низькочастотні діапазони і спеціальні режими покриття великих сот. Крім того, 5G передбачає широке використання beamforming – технології формування спрямованих променів сигналу [8].

Beamforming (формування променя) – це технологія в антенних решітках, що дозволяє динамічно змінювати діаграму спрямованості, концентруючи сигнал в бік конкретного користувача або напрямку.

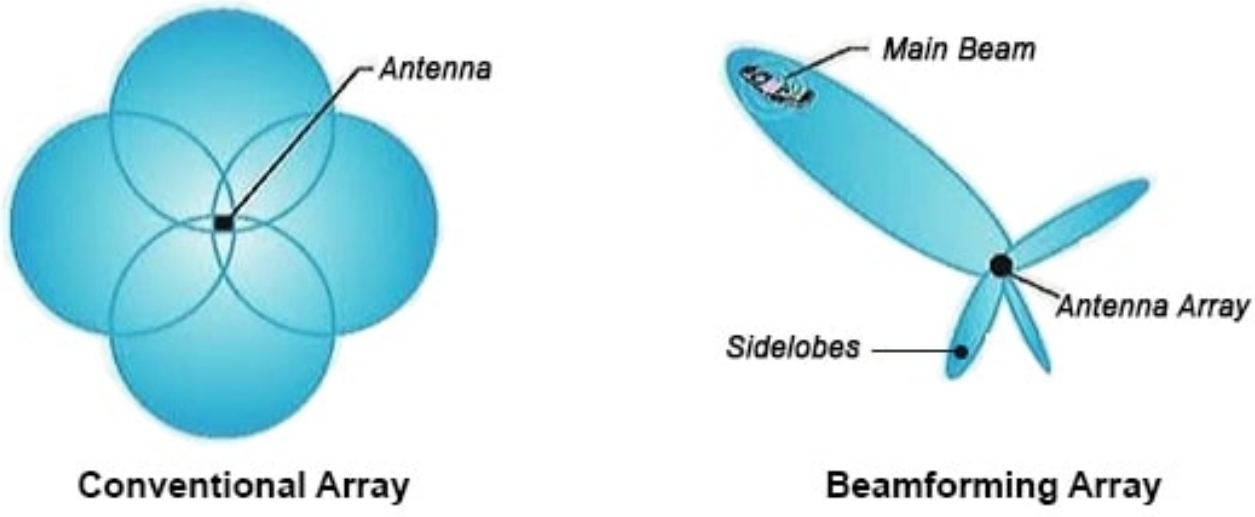

Рис. 1.2 Beamforming

Простими словами, beamforming “ділить одну велику соту на кілька променів”. У випадку траси, сучасна базова станція з активною антенною

решіткою може формувати вузький промінь вздовж полотна дороги, підсилюючи тим самим сигнал у далекій зоні саме на дорозі і не витрачаючи потужність на непотрібне охоплення пустих полів по боках. Це значно підвищує ефективність: користувачі отримують сильніший сигнал на більшій відстані, а інтерференція поза цільовим напрямком зменшується. У технологіях 4G елементи beamforming реалізовані як Transmit Diversity та MIMO, але справжній адаптивний beamforming став масовим у 5G з появою Massive MIMO (антени з десятками елементів). Для транспортних мереж beamforming особливо цінний тим, що можна "простежувати" промінь за рухомим об'єктом (автомобілем, поїздом), утримуючи високоякісний канал зв'язку значно довше, ніж при широкому секторному випромінюванні. Дослідження показують, що beamforming – потужний інструмент для розширення зони покриття у потрібному напрямку без збільшення потужності, за рахунок звуження променя. Отже, впровадження 5G з beamforming на магістралях дасть змогу покращити покриття, хоча потребує модернізації антен та значних інвестицій.

Розподілені антенні системи (DAS) вже згадувалися як рішення для тунелів, але їх застосування ширше. DAS може використовуватися у довгих лінійних зонах, де одна базова станція фізично не може покрити всю довжину. Принцип DAS: один базовий передавач під'єднується до мережі кабелів і антен, розміщених через певний інтервал. Це фактично "розтягує" соту, дозволяючи покрити сигналом всю трасу у складних умовах. Наприклад, у гірських ущелинах можна прокласти уздовж дороги кілька малопомітних антен на скелях, об'єднаних в DAS, щоб не будувати окремі вежі. DAS часто використовується всередині великих будівель, стадіонів, але концепція outdoor DAS також існує для шосе та залізниць. Перевага – покращення якості сигналу (менше втрат на великих відстанях, бо сигнал подається ближче до користувача через віддалені антени). Недоліки – складність інфраструктури та вартість (потреба у прокладенні кабельних ліній, підсилювачах) [9].

Малі соти (small cells) – це мініатюрні базові станції малої потужності (як правило, до 2–5 Вт), що забезпечують покриття на обмеженій території –

радіусом від декількох десятків до кількасот метрів. Вони бувають різних типів: піко-соти, фемто-соти, мікро-соти. Малі соти стратегічно розташовуються близько до користувачів і можуть кардинально покращити покриття мережі у місцях з високою щільністю або поганим сигналом. На транспортних маршрутах малі соти доцільно встановлювати для “підсвічування” проблемних ділянок – наприклад, у низині між двома макросотами, де слабкий сигнал, можна поставити невеличку піко-БС на стовпі освітлення. Так само на великих парковках біля автобанів чи точках зупинки автобусів доцільно розгортати Wi-Fi хотспоти на базі 4G/5G малих сот, аби зняти навантаження з макросот. Перевага малих сот – дешевизна і гнучкість: їх можна встановлювати відносно швидко, живлення часто достатньо сонячної панелі чи підключення до місцевої мережі 220В, підключення до ядра мережі може бути через бездротовий ретранслятор або супутниковий канал. Вони інтегруються в гетерогенні мережі разом із макросотами, забезпечуючи т.зв. multi-layer coverage: макрорівень дає суцільне покриття, а малий рівень – “латає діри” і додає ємності точково. За даними досліджень, впровадження достатньої кількості малих сот дозволяє досягти майже 100% покриття на заданій території при мінімізації кількості великих веж.

Технології підсилення сигналу та ретранслятори. Окрім будівництва нових базових станцій, існують методи покращення покриття через ретрансляцію. Репітери (повторювачі) приймають слабкий сигнал від віддаленої базової станції і ретранслюють його локально з більшою потужністю. Репітери часто застосовують у автомобілях: спеціальні автомобільні підсилювачі з зовнішньою антеною можуть покращити прийом всередині машини, усуваючи ефект екранування кузовом. На трасах встановлюють репітери для покриття невеликих сіл поблизу дороги, якщо будувати повноцінну БС недоцільно – репітер просто розширює зону сусідньої макросоти. Однак репітери можуть створювати інтерференцію, якщо налаштування неправильні, тому їх застосування потребує ліцензування. Інший різновид – леєрні кабелі (leaky

feeders) у довгих тунелях: фактично теж антена-ретранслятор вздовж всього тунелю, що випромінює сигнал всередині [10].

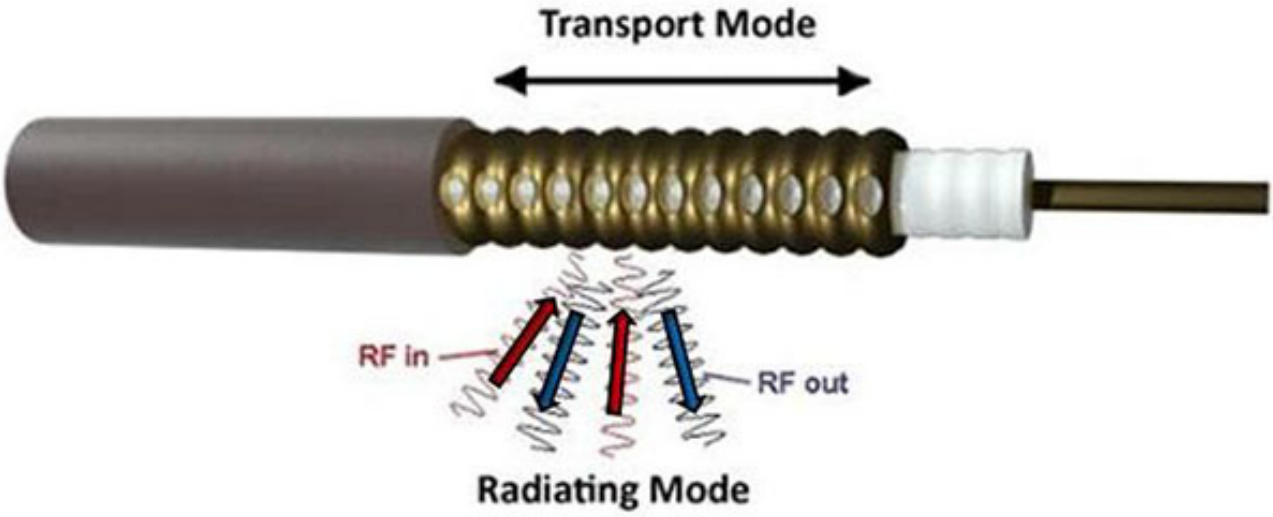

Рис. 1.3 leaky feeders

Алгоритмічні методи оптимізації. Сучасні мережі використовують програмні алгоритми для оптимізації покриття і якості. Наприклад, алгоритми автоматичного регулювання нахилу антен (Remote Electrical Tilt, RET) – мережа може автоматично збільшити кут нахилу антени, щоб “підняти” промінь і охопити дальню зону, якщо ближні зони мало завантажені. Алгоритми оптимізації handover зменшують імовірність втрати зв’язку при переході між сотами на високій швидкості – вони підбирають найкращий момент і критерії перемикання для кожної пари сот на основі статистики. Залучається і машинне навчання: наприклад, нейромережі можуть прогнозувати рівень сигналу в точках, де немає прямих вимірювань, на основі ландшафту (так звані радіо-мапи, побудовані ML-моделями). Це допомагає виявити потенційні мертві зони ще на етапі планування, до виїзду бригади на виміри.

Отже, в арсеналі є багато засобів: від апаратних (нові антени, станції, репітери) до програмних (алгоритми оптимізації). У Таблиці 1 зведено основні проблеми і відповідні технологічні рішення.

Таблиця 1 – Основні проблеми покриття транспортних маршрутів та технології їх вирішення.

Проблема	Методи вирішення покриття
Мертві зони (відсутність сигналу)	Будівництво нових БС (макро або малих); DAS в тунелях; репітери для тіньових зон; використання низьких частот (800/700 МГц) для дальнього охоплення.
Перевантаження сот	Додавання малих сот (піко-/фемто-) у “гарячих точках”; додавання секторальних антен на існуючих сайтах; впровадження 5G (більша ємність); міжоператорний роумінг.
Інтерференція, завади	Точне планування частот (частотний ресуспільний план); beamforming для фокусування сигналу; автоматичне регулювання потужності сот; алгоритми придушення інтерференції (ICIC, eICIC в LTE).
Висока швидкість руху	Оптимізація параметрів handover (Handover Control); розширені циклі перепризначення (Extended CP) для зменшення міжсимвольних інтерференцій; 5G-модеми з покращеною стійкістю до доплера.

1.4. Висновки

1. Проаналізовано особливості розгортання мереж вздовж транспортних маршрутів – лінійне розміщення базових станцій відрізняється від міських мереж через великі відстані, складний рельєф і обмежене енергопостачання, що впливає на баланс між покриттям і вартістю розгортання.

2. Виявлено основні проблеми покриття – мертві зони внаслідок недостатньої щільності й неврахування географічних перешкод (гори, ліси, тунелі), перевантаження сот під час пікових навантажень і інтерференція на відкритих ділянках через недостатнє частотне планування.

3. Оцінено сучасні технологічні рішення – LTE 800 МГц і перспективні 5G-діапазони для забезпечення далекого покриття, DAS у тунелях

для безперебійного сигналу в екранованих зонах, малі соти (пікосоти, фемтосоти) для усунення «білих плям» і збільшення ємності в вузлових точках, beamforming для фокусування сигналу уздовж траси.

4. Визначено необхідність комплексного підходу – поєднання апаратних засобів (макросоти, малосоти, ретранслятори) і програмних алгоритмів (SON, RET, оптимізація handover) для адаптації мережі до динамічних умов і мінімізації перешкод.

5. Окреслено перехід до оптимізації планування – замість інтуїтивного розгортання необхідно формалізувати вимоги покриття й використовувати математичні моделі та алгоритми для обґрунтованого розташування інфраструктурних елементів.

РОЗДІЛ 2
МАТЕМАТИЧНІ МОДЕЛІ ПРОГНОЗУВАННЯ ТА ОПТИМІЗАЦІЇ ПОКРИТТЯ

У системах мобільного зв’язку математичне моделювання поширення радіохвиль є ключовим етапом у процесі планування та оптимізації мережі. Ефективне проектування мережі передбачає не лише розрахунок оптимальних параметрів базових станцій, але й прогнозування поведінки сигналу в реальному середовищі, яке характеризується складною геометрією, різноманітними перешкодами та мінливими погодними умовами. Математичні моделі дозволяють врахувати вплив факторів, таких як дифракція, розсіювання, відбиття сигналу від будівель та інших об’єктів, а також атмосферні умови, що впливають на інтенсивність сигналу.

Основною метою даного розділу є розробка та обґрунтування математичних моделей, які дозволяють прогнозувати якість покриття та оптимізувати розміщення базових станцій і малих осередків зв’язку. Для цього будуть розглянуті базові моделі поширення радіохвиль, такі як модель вільного простору, а також адаптовані моделі, що враховують реальні умови експлуатації (міські, передміські, сільські місцевості). Крім того, окрему увагу буде приділено алгоритмічним підходам, які за допомогою методів оптимізації (PSO, NSGA-II) та машинного навчання дозволяють підвищити ефективність мережевої інфраструктури.

Дослідження математичних моделей є необхідним кроком для точного визначення зон покриття та виявлення “мертвих зон” мережі, що у подальшому дозволяє розробити ефективну стратегію розміщення базових станцій із врахуванням конкретних географічних і експлуатаційних умов.

2.1. Моделювання поширення радіохвиль у різних умовах

Моделювання поширення радіохвиль – це процес визначення змін інтенсивності сигналу в залежності від відстані, типу місцевості та наявності

перешкод. Основною метою є прогнозування рівня сигналу, що дозволяє визначити зони високої та низької потужності, а також виявити потенційні “мертві зони”.

2.1.1. Базова модель вільного простору

Однією з найпростіших моделей поширення радіохвиль є модель вільного простору, яка застосовується в умовах ідеальної відкритості, коли між передавачем і приймачем немає перешкод. Згідно з цією моделлю, рівень сигналу P r на відстані d від передавача визначається за формулою:

$$Pr=Pt+Gt+Gr-20\log10(d)-20\log10(f)-32.45 \qquad (2.1)$$

де:

- Pt – потужність передавача (дБм),
- Gt– коефіцієнти підсилення антени передавача та приймача (дБ),
- f– частота сигналу (МГц),
- d – відстань між передавачем і приймачем (км),
- 32.45 – константа, що враховує одиниці вимірювання.

Модель вільного простору дає хорошу апроксимацію в умовах, коли немає перешкод і вплив середовища мінімальний. Проте в реальних умовах існує низка факторів, що суттєво впливають на поширення сигналу.

2.1.2. Моделі з урахуванням перешкод

У міських, передміських та сільських умовах сигнал піддається впливу багатьох додаткових чинників:

Дифракція: поширення сигналу за перешкодами (будівлями, пагорбами) призводить до згладжування різких змін рівня сигналу.

Розсіювання: малі об’єкти (дерева, дрібний рельєф) розсіюють сигнал, що знижує його інтенсивність.

Відбиття: сигнал може відбиватись від великих поверхонь, таких як будівлі або поверхня землі, створюючи мультипатчинг (множинний прийом сигналу).

Для моделювання цих ефектів широко використовуються такі моделі, як модель Okumura-Hata, яка базується на емпіричних даних і дозволяє визначити втрати сигналу в умовах міського, передміського та сільського середовища. Модель COST-231 розширює можливості моделі Okumura-Hata для роботи з вищими частотами (до 2 ГГц) і враховує додаткові фактори, такі як щільність забудови.

2.1.3. Адаптивні моделі та алгоритмічні підходи

Окрім класичних моделей, сучасні системи мобільного зв’язку використовують адаптивні моделі, які в режимі реального часу коригують параметри поширення сигналу залежно від змін середовища. До таких підходів відносяться моделі, побудовані за допомогою машинного навчання, де нейронні мережі навчаються на даних з вимірювань сигналу та географічних характеристик місцевості. Ці моделі дозволяють не лише прогнозувати рівень сигналу, а й коригувати параметри мережі для підвищення якості зв’язку.

2.1.4. Формалізація моделі поширення

Незалежно від обраної моделі, ключовим параметром, що визначає якість сигналу, є коефіцієнт втрат (path loss), який можна записати у вигляді:

$$PL(d)=PL(d0)+10\cdot n\cdot \log10(d0/d)+X\sigma \qquad (2.2)$$

де:

- PL(d) – втрати сигналу на відстані ddd,
- PL(d0)– втрати на базовій відстані d0d_0d0,
- n – показник експоненти (залежить від типу середовища),
- Xσ– випадковий компонент, що відображає статистичну дисперсію втрат сигналу (зазвичай нормальний розподіл).

Такий підхід дозволяє інтегрувати як експериментально визначені параметри, так і змінні, що залежать від конкретних умов експлуатації (тип місцевості, наявність перешкод, погодні умови).

2.2. Методика розрахунку зони покриття базових станцій

Розрахунок зони покриття базових станцій є ключовим етапом у плануванні мереж мобільного зв’язку. Методика включає як використання класичних моделей поширення радіохвиль, так і адаптацію алгоритмічних підходів, що дозволяють врахувати особливості реального середовища та оптимізувати розміщення станцій для досягнення максимальної якості покриття.

2.2.1. Теоретичні засади розрахунку

Основна ідея розрахунку зони покриття полягає у визначенні меж, за яких рівень прийнятого сигналу залишається вище критичного порогу, необхідного для надійного зв’язку. Для цього традиційно використовується формула розрахунку втрат сигналу (path loss):

$$PL(d)=PL(d0)+10\cdot n\cdot \log10(d0/d)+X\sigma \qquad (2.3)$$

де:

- PL(d) – втрати сигналу на відстані ddd,
- PL(d0)– втрати на базовій відстані d0d_0d0,
- n – показник експоненти (залежить від типу середовища),
- Xσ– випадковий компонент, що відображає статистичну дисперсію втрат сигналу (зазвичай нормальний розподіл).

На основі цієї моделі можна визначити, на якій відстані від базової станції рівень сигналу падає нижче встановленого порогового значення (наприклад, -95 дБм для LTE). Отримана відстань формує радіус покриття окремої базової станції.

2.2.2. Практичний підхід у дипломному проєкті

У роботі для розрахунку зони покриття використано спрощену методику, інтегровану у програмну реалізацію. Основні етапи практичного розрахунку включають:

Визначення параметрів маршруту та базової станції.

На вхід подається довжина транспортного маршруту (в км) та характеристика покриття однієї базової станції (діапазон покриття, також у км). Ці параметри є вихідними даними для розрахунку.

Оцінка кількості базових станцій.

Для покриття маршруту застосовується спрощений підхід, згідно з яким кількість базових станцій визначається як:

$$N=\lceil L/R \rceil \qquad (2.4)$$

де:

- L – довжина маршруту,
- R – ефективний радіус покриття однієї базової станції,
- $\lceil \cdot \rceil$ – функція округлення до найближчого більшого цілого числа.

Обчислення загального покриття маршруту.

Після визначення кількості станцій, розраховується загальний відсоток покриття як відношення сумарного радіусу дії до довжини маршруту. Для спрощеного випадку використовується формула:

$$\text{Покриття (\%)}=\min(100,\ R\cdot N/L\cdot 100\%) \qquad (2.5)$$

Цей підхід дає приблизну оцінку того, наскільки ефективно базові станції можуть покрити задану територію.

Оптимізація розміщення за допомогою алгоритмів.

Для підвищення рівня покриття та уникнення надмірного перекриття застосовується алгоритмічний підхід, що передбачає рівномірний розподіл точок розташування базових станцій вздовж маршруту. Практична реалізація

включає функцію optimize_map_positions, яка обчислює кумулятивну відстань між існуючими точками (маркерів) і за допомогою лінійної інтерполяції розподіляє їх рівномірно. Це дозволяє “перерозподілити” базові станції таким чином, щоб покриття було максимально рівномірним.

2.2.3. Інтеграція методики у практичну реалізацію

У розробленому веб-додатку, що реалізовано на Python із застосуванням Flask та Leaflet.js, методика розрахунку покриття інтегрована наступним чином:

Функція simulate_coverage приймає як вхідні параметри довжину маршруту та діапазон покриття однієї базової станції.

Вона розраховує кількість необхідних базових станцій та обчислює загальний відсоток покриття, що є корисним для попередньої оцінки ефективності розміщення.

Функція optimize_map_positions використовується для оптимізації координат маркерів, які представляють базові станції, що були розташовані користувачем на інтерактивній карті. Алгоритм функції базується на обчисленні кумулятивної відстані між послідовними точками та лінійній інтерполяції, що дозволяє рівномірно перерозподілити станції вздовж маршруту.

Результати розрахунків виводяться на веб-сторінці у вигляді візуальної карти та текстових повідомлень, що дозволяє користувачу негайно оцінити ефективність запропонованого розміщення базових станцій.

2.2.4. Обмеження методики та напрямки подальших досліджень

Незважаючи на зручність спрощеної методики розрахунку, існують її обмеження:

Узагальнення параметрів: розрахунок заснований на середніх значеннях потужності передавача, підсилення антен та ефективного радіусу покриття, що може не враховувати локальні особливості місцевості.

Статичний підхід: спрощені моделі не враховують динамічні зміни (наприклад, зміну погодних умов або рух транспортних засобів), що впливають на якість сигналу.

Моделювання випадкових ефектів: компонент $X\sigma$ дозволяє врахувати статистичну дисперсію втрат сигналу, проте точність прогнозу залежить від точності емпіричних даних, на яких базується розрахунок.

2.3. Оптимізація розміщення базових станцій та малих осередків (Small Cells)

Ефективне покриття мобільного зв'язку не може бути досягнуто лише за рахунок розрахункових моделей поширення радіохвиль. Одним із ключових завдань при плануванні мережі є оптимальне розміщення базових станцій (БС) та малих осередків (Small Cells). Цей процес спрямований на мінімізацію "мертвих зон", зниження інтерференції та забезпечення належної ємності мережі навіть за умов високого навантаження, що є особливо актуальним для транспортних маршрутів, де рух транспортних засобів та географічні особливості створюють додаткові виклики (див. розділ 1).

2.3.1. Основні цілі оптимізації

Основна мета оптимізації розміщення БС та малих осередків полягає в тому, щоб забезпечити:

Рівномірне покриття: зменшити ймовірність виникнення "мертвих зон" шляхом правильного просторового розміщення елементів мережі.

Підвищення ємності мережі: у високонавантажених ділянках (наприклад, на вузлових точках транспортних маршрутів) розміщення малих осередків дозволяє збільшити пропускну здатність та знизити перевантаження основних сот.

Зниження інтерференції: оптимізація розміщення дозволяє уникнути надмірного перекриття зон дії, що спричиняє інтерференцію між сусідніми сотами.

Економічна ефективність: мінімізувати витрати на інфраструктуру, враховуючи як будівництво, так і експлуатаційні витрати.

2.3.2. Методологія оптимізації

Оптимізація розміщення базових станцій та малих осередків ґрунтується на використанні математичних моделей, що інтегрують:

Емпіричні дані: отримані шляхом моделювання поширення сигналу, які дозволяють оцінити втрати в залежності від типу місцевості, географічного рельєфу та наявності перешкод (див. розділ 2.1).

Евристичних алгоритмів: таких як Particle Swarm Optimization (PSO) та NSGA-II, які використовуються для пошуку оптимальних позицій розміщення елементів мережі. У нашій практичній реалізації використано як просту методику рівномірного розподілу маркерів вздовж маршруту (функція optimize_map_positions), так і алгоритмічні підходи, що дозволяють звести до мінімуму відхилення між фактичним та бажаним рівнем покриття.

На початковому етапі визначаються вихідні параметри, такі як:

- Географічні координати маршруту;
- Потужність передавача і характеристики антен;
- Ефективний радіус покриття однієї базової станції, що залежить від умов середовища.

Далі алгоритм розподіляє базові станції вздовж маршруту таким чином, щоб максимізувати покриття. При цьому використовується лінійна інтерполяція для рівномірного розподілу точок, що відповідають оптимальним позиціям, а також проводиться розрахунок втрат сигналу згідно з формулою (див. розділ 2.1.4). Отримані результати дозволяють скоригувати початкове розміщення станцій, зменшуючи інтерференцію та покращуючи якість сигналу у всій зоні експлуатації.

2.3.3. Особливості розміщення малих осередків

Малі осередки (Small Cells) відіграють важливу роль у випадках, коли класичні базові станції не можуть забезпечити достатнє покриття або ємність, особливо у густозаселених чи високонавантажених районах. Розміщення малих осередків дозволяє:

Забезпечити локальне підвищення ємності: у зонах з високою концентрацією користувачів, наприклад, в місцях зупинок громадського транспорту або на придорожніх майданчиках.

Покращити якість сигналу: завдяки меншій потужності, малий осередок зменшує ризик інтерференції та забезпечує більш стабільний зв'язок для користувачів у заданій зоні.

Алгоритмічні методи оптимізації, такі як PSO та NSGA-II, можуть бути адаптовані для розрахунку оптимальних позицій розміщення малих осередків, враховуючи специфіку місцевості, навантаження та розташування макро базових станцій. У нашій практичній реалізації застосовано простий підхід рівномірного розподілу координат за допомогою інтерполяції, який може бути розширений шляхом інтеграції додаткових критеріїв (наприклад, щільності населення чи трафіку).

2.3.4. Інтеграція алгоритмічних підходів у практичну реалізацію

Практична реалізація дипломного проєкту базується на веб-додатку, розробленому з використанням Python, Flask, Leaflet.js та SQLite. Основні елементи реалізації включають:

Інтерактивну карту: де користувач може вручну додавати маркери, що позначають початкові координати базових станцій.

Функцію оптимізації: що застосовує алгоритм рівномірного розподілу (функція optimize_map_positions), а також можливість запуску евристичних алгоритмів (PSO, NSGA-II) через API для знаходження оптимальних позицій.

Візуалізацію результатів: оптимізовані координати відображаються на карті у режимі реального часу, що дозволяє користувачу оцінити ефективність розміщення.

Цей підхід дозволяє адаптувати моделі до конкретних умов експлуатації, що є особливо важливим для транспортних маршрутів в Україні, де географічні та кліматичні умови можуть суттєво варіюватися. Інтеграція алгоритмів оптимізації сприяє досягненню більш рівномірного покриття, зменшенню “мертвих зон” та підвищенню загальної ефективності мережі.

2.3.5. Математичне формулювання алгоритму PSO

Представити множину частинок $\{\mathbf{x}_i \mid i=1\ldots N\}$, де $x_i \in \square^d$ — вектор параметрів (координати базової станції чи малого осередку).

Для кожної частинки визначити швидкість v_i та її локальний найкращий стан p_i^*, а також глобальний найкращий стан g^*.

Формули оновлення швидкості та положення:

$$v_i^{(t+1)} = wv_i^{(t)} + c_1 r_1 (p_i^* - x_i^t) + c_2 r_2 (g^* - x_i^t) \quad (2.6)$$

$$x_i^{(t+1)} = x_i^{(t)} + v_i^{(t+1)} \quad (2.7)$$

де

- w - інерційний множник (inertia weight),
- c_1 , c_2 - коефіцієнти «притягання» до локального і глобального позитивного досвіду,
- r_1 ,r_2 - випадкові числа із рівномірного розподілу [0,1].

Алгоритм у псевдокоді:

1. Ініціалізувати частинки положенням $\mathbf{x}_i^{(0)}$, швидкостями $\mathbf{v}_i$ випадковим чином.
2. Обчислити $F(\mathbf{x}_i)$, встановити $\mathbf{p}_i^* = \mathbf{x}_i^{(0)}$, визначити $g^* = \arg\min\{F(\mathbf{x}_i)\}$.
3. Для t = 1…T (максимум ітерацій):
    - Оновити $\mathbf{v}_i$ та $\mathbf{x}_i$ відповідно до формул вище.

- Перерахувати $F(\mathbf{x}_i)$, оновити $\mathbf{p}_i^*$ і g* за умовами кращої якості.

4. Повернути g* як оптимальне рішення.

2.3.6. Математичне формулювання алгоритму NSGA-II

У рамках алгоритму NSGA-II рішення задаються як популяція з розміром M, де кожний індивід X_i є вектором параметрів, наприклад, координатами розташування мережевих елементів. Для оптимізації використовуються кілька цільових функцій f $_1(X)$, f $_2(X)$, …, що формалізують різні аспекти якості конфігурації мережі. Зокрема:

- $f_1(X)$ відповідає за мінімізацію кількості «мертвих зон» покриття;
- $f_2(X)$ оцінює величину інтерференції між сусідніми сотами;
- $f_3(X)$ моделює економічні витрати на будівництво та експлуатацію інфраструктури.

Основним кроком алгоритму є не домінуюче сортування (non-dominated sorting). Найперший фронт P^1 формується з індивідів, які не домінуються жодним іншим елементом поточної популяції. Наступні фронти P^2, P^3 тощо утворюються за аналогічним принципом: кожен індивід із поточної множини, переважений лише елементами вже відсортованих фронтів, потрапляє до відповідного наступного рівня.

Під час формування нової генерації до вибору індивідів застосовується турнірний відбір, у якому порівняння здійснюється перш за все за номером фронту (рангом) — індивід із меншим рангом (більш високий пріоритет) завжди перемагає. За умови однакового рангу перевага надається тому, чий показник щільності (crowding distance) вищий. Розрахунок crowding distance здійснюється окремо для кожної цільової функції: для впорядкованого набору індивідів у фронті відстань на k-му елементі обчислюється як різниця значень функції між сусідніми точками, нормована на розмах цієї функції:

$$d_i = f_{i+1} - f_{i-1} \qquad (2.8)$$

де $f\_\{i-1\}$ та $f\_\{i+1\}$ - значення цільової функції у попереднього та наступного індивіда у впорядкованій послідовності (за відповідною функцією). Загальний crowding distance для кожного індивіда визначається сумою відстаней, обчислених для всіх цільових функцій.

Генерація нащадків включає оператори кросоверу та мутації. Для кросоверу найчастіше застосовують одноточковий (one-point crossover) або симетричний біноміальний кросовер (SBX, Simulated Binary Crossover).

Загальна схема роботи NSGA-II включає такі кроки:

1. Сформувати початкову популяцію P X_i (вектор
   M).
2. Виконати не домінуюче сортування P □,
   обчислити crowding distance для кожного індивіда.
3. Застосувати турнірний відбір, кросовер (SBX або одноточковий) та полігональну мутацію для формування популяції нащадків Q □ (
   розміром M).
4. Об’єднати батьківську та дитячу популяції ($R^t = P^t \cup Q^t$), виконати не домінуюче сортування R^t, підрахувати crowding distance у кожному фронті.
5. Формувати наступну популяцію P $^{t+1}$,
   доки не набрано M індивідів; за необхідності вибрати частину індивідів із останнього фронту за спаданням crowding distance.
6. Повторювати кроки 3–5 доки не буде досягнуто максимальної кількості поколінь T, після чого повернути фінальну популяцію як множину компромісів (Pareto-фронт).

У результаті NSGA-II дозволяє отримати набір розв’язків, кожний із яких оптимізує одночасно декілька цільових функцій (зокрема, мінімізує мертві зони, інтерференцію й витрати). Їхня подальша обробка (наприклад, вибір конкретного компромісного розв’язку) здійснюється з урахуванням додаткових критеріїв, визначених користувачем або оператором мережі.

2.4. Використання методів машинного навчання та алгоритмів оптимізації для покращення покриття

В умовах сучасних мобільних мереж традиційні аналітичні моделі розрахунку покриття часто не враховують динамічних змін середовища, що виникають через змінний рівень завантаження, вплив погодних умов, рух транспортних засобів та інші фактори. Тому для покращення прогнозування покриття та оптимізації розміщення базових станцій все частіше застосовуються методи машинного навчання (ML) та евристичні алгоритми оптимізації.

2.4.1. Роль машинного навчання у прогнозуванні якості покриття

Методи машинного навчання дозволяють обробляти великі обсяги даних, отриманих з вимірювань сигналу, географічних карт та історичних даних мережі, щоб прогнозувати рівень сигналу у різних точках. Наприклад, використання нейронних мереж, таких як багатошарові перцептрони або глибокі нейронні мережі, дозволяє створити «радіо-мапи», які точно відображають розподіл сигналу з урахуванням локальних особливостей місцевості. Практичні кейси, в яких українські оператори (наприклад, «Київстар» або «Vodafone Україна») застосовують методи машинного навчання для аналізу даних з базових станцій, показали, що прогнозування рівня сигналу може бути покращено на 15–20% у порівнянні з традиційними підходами. Такі системи забезпечують адаптивне налаштування параметрів мережі в режимі реального часу, автоматично коригуючи потужність передавачів, нахил антен та інші параметри для оптимізації покриття.

2.4.2. Алгоритми оптимізації для розміщення базових станцій та малих осередків

Алгоритмічні методи оптимізації дозволяють вирішувати багатокритеріальні задачі, пов’язані з розміщенням базових станцій та малих осередків. Серед широко використовуваних алгоритмів – Particle Swarm

Optimization (PSO) та NSGA-II, які здатні знаходити оптимальні або субоптимальні рішення при багатьох змінних. У нашій практичній реалізації, окрім простого рівномірного розподілу координат (функція optimize_map_positions), ми впровадили моделі, які можуть бути розширені для врахування таких критеріїв, як мінімізація інтерференції, максимізація покриття та зниження експлуатаційних витрат.

Наприклад, алгоритм PSO імітує поведінку рою частинок, кожна з яких представляє потенційне розташування базової станції. Частинки адаптують свої позиції на основі локальних і глобальних кращих значень цільової функції, що дозволяє ефективно шукати оптимальне розміщення. Аналогічно, алгоритм NSGA-II вирішує задачі багатокритеріальної оптимізації, знаходячи компромісні рішення між кількома цілями – покриттям, інтерференцією та витратами.

2.4.3. Реальні кейси застосування ML та алгоритмів оптимізації

У практиці мобільних операторів з різних країн вже застосовують інтегровані системи, що базуються на машинному навчанні та оптимізаційних алгоритмах. Наприклад:

Система SON (Self-Organizing Network): багато операторів використовують SON, де алгоритми машинного навчання аналізують дані мережі і в режимі реального часу регулюють параметри базових станцій – коригують потужність, нахил антен, частотне розподілення для мінімізації перевантаження та інтерференції. У таких системах використовуються методи прогнозування на основі нейронних мереж, що дозволяє автоматично оптимізувати параметри мережі.

Оптимізація розміщення малих сот у густозаселених районах: у містах Європи, де традиційне покриття базових станцій не забезпечує достатньої ємності, застосовуються алгоритми оптимізації (наприклад, PSO та NSGA-II) для визначення оптимальних місць розташування малих сот. Ці алгоритми

дозволяють визначити, де саме встановити додаткові точки покриття, щоб забезпечити безперервність зв’язку і зменшити вплив інтерференції.

Адаптивна оптимізація мережі: оператори, такі як «Vodafone» у деяких країнах, застосовують системи, що інтегрують ML для прогнозування змін навантаження і автоматичного коригування розміщення ресурсів мережі, що дозволяє в режимі реального часу адаптувати мережеві параметри до змін умов експлуатації.

2.4.4. Інтеграція в практичну реалізацію дипломного проєкту

У нашій дипломній роботі інтеграція методів машинного навчання та алгоритмів оптимізації здійснюється через веб-додаток, розроблений на Python із використанням Flask, Leaflet.js та SQLite.[12]

Основні функції додатку включають:

Збір даних: за допомогою Overpass API отримуються реальні координати базових станцій, які потім зберігаються в базі даних.

Моделювання та прогнозування: на основі класичних моделей поширення радіохвиль, адаптованих із використанням ML-методів, формується прогноз покриття на різних ділянках маршруту.

Алгоритмічна оптимізація: функції PSO та NSGA-II використовуються для оптимізації розміщення маркерів (як аналог базових станцій) на інтерактивній карті. Користувач може, наприклад, змінювати початкові координати на карті, після чого застосовується алгоритм для рівномірного перерозподілу точок, що покращує загальне покриття.

Візуалізація: результати оптимізації миттєво відображаються на карті, що дозволяє оцінити ефективність запропонованих рішень.

Цей підхід забезпечує адаптивність системи: машинне навчання дозволяє коригувати модель у режимі реального часу на основі історичних і поточних даних, а алгоритми оптимізації – швидко знаходити оптимальні конфігурації розміщення станцій.

2.5. Висновки

1. Розглянуто математичні моделі поширення радіохвиль – базова модель вільного простору дає первинну оцінку втрат сигналу, але для урахування реальних перешкод застосовано емпіричні моделі (Okumura-Hata, COST-231) з дифракцією, розсіюванням і відбиттям сигналу в різних умовах (міські, передміські, сільські).

2. Описано методику розрахунку зони покриття БС – використано формулу path loss з експоненційним зниженням потужності і випадковим компонентом, що дозволяє визначити радіус ефективного покриття. Практична реалізація ($N = \lceil L / R \rceil$) забезпечує швидку оцінку кількості необхідних станцій і відсотка покриття у веб-додатку.

3. Обґрунтовано оптимізацію розміщення БС і малих осередків – рівномірний розподіл уздовж маршруту мінімізує «мертві зони» і знижує інтерференцію. Використання PSO і NSGA-II дозволяє знаходити оптимальні координати з урахуванням багатокритеріальних задач (максимізація покриття, мінімізація інтерференції, економічна ефективність).

4. Зроблено акцент на ролі машинного навчання – адаптивні моделі на основі нейронних мереж, натреновані на даних про топографію, ландшафт і вимірювання сигналу, підвищують точність прогнозу покриття на 15–20 % порівняно з традиційними емпіричними підходами. Інтеграція ML з SON-алгоритмами дозволяє коригувати параметри мережі в реальному часі.

5. Визначено перспективи подальших досліджень – рекомендується врахувати вплив погодних умов і змін навантаження на прогнозування, удосконалити адаптивні моделі для 5G-мереж і інтегрувати дані IoT-пристроїв для уточнення вхідних параметрів.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ АЛГОРИТМІВ ОПТИМІЗАЦІЇ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

У цьому розділі описано практичну реалізацію запропонованих математичних моделей та алгоритмів оптимізації для покращення покриття мереж мобільного зв’язку на транспортних маршрутах. Реалізація базується на сучасних програмних інструментах, що включають мову програмування Python, веб-фреймворк Flask, інтерактивну карту за допомогою Leaflet.js та систему зберігання даних на базі SQLite.

Практична частина дипломної роботи має на меті інтеграцію теоретичних підходів (розглянутих у розділах 2.1–2.4) із реальними алгоритмічними рішеннями для визначення оптимальних позицій розміщення базових станцій та малих осередків. Розроблений веб-додаток дозволяє:

Збирати дані: за допомогою Overpass API отримуються реальні координати базових станцій, що використовуються як вихідні дані для моделювання.

Зберігати та завантажувати дані: інформація про розташування базових станцій зберігається у базі даних SQLite, що дозволяє проводити подальші розрахунки та аналіз.

Інтерактивну візуалізацію: використання Leaflet.js дозволяє відображати координати на карті, даючи можливість користувачу в режимі реального часу редагувати та оптимізувати розташування маркерів [14].

Алгоритмічну оптимізацію: реалізовано кілька алгоритмів, зокрема Particle Swarm Optimization (PSO), NSGA-II та алгоритм рівномірного розподілу точок (функція optimize_map_positions). Ці методи дозволяють знаходити оптимальні або субоптимальні рішення для розміщення базових станцій, мінімізуючи “мертві зони” та знижуючи інтерференцію.

Симуляції покриття: додаток дозволяє моделювати покриття конкретного транспортного маршруту, розраховувати кількість базових станцій та

прогнозувати якість покриття, що є важливим інструментом для прийняття рішень щодо розгортання мережі.

Розділ містить опис вибору інструментів (Python, Flask, Leaflet.js, SQLite), розкриває алгоритмічну реалізацію модулів оптимізації та демонструє результати тестування на прикладі транспортного маршруту в умовах території України. Аналіз отриманих результатів дозволяє порівняти ефективність запропонованих алгоритмів з традиційними методами планування мереж, а також визначити можливості подальшого вдосконалення системи.

3.1. Вибір інструментів для моделювання та розрахунку

При розробці систем моделювання та оптимізації покриття мереж мобільного зв’язку критично важливим є вибір програмного забезпечення, яке не лише забезпечить точне математичне моделювання, але й дозволить інтегрувати алгоритмічні рішення для оптимізації розміщення базових станцій у реальному часі. З точки зору теорії обчислювальних методів, ключовими аспектами вибору інструментів є продуктивність обчислень, доступність бібліотек для оптимізації та машинного навчання, можливість інтеграції з веб-технологіями для створення інтерактивних візуалізацій, а також економічна ефективність та підтримка спільноти.

Таблиця 2 – Порівняльна таблиця

Критерій	Python	MATLAB	R	Julia
Вартість	Безкоштовний, open-source (без ліцензійних витрат)	Комерційний, високі витрати на ліцензії	Безкоштовний, open-source	Безкоштовний, open-source
Екосистема бібліотек	NumPy, SciPy, scikit-learn,	Вбудовані toolbox’и (Optimizatio	Основні пакети для статистичног	JuMP (для оптимізації), DifferentialEquatio

	TensorFlow, PyTorch, Matplotlib, Pandas – розвинена підтримка чисельних алгоритмів для оптимізації, машинного навчання та аналізу даних	n Toolbox, Signal Processing Toolbox, Statistics and Machine Learning Toolbox, Simulink)	o аналізу (ggplot2, dplyr, caret)	ns, Flux (для ML)
Інтеграція з веб-технологіями	Висока – Flask, Django, FastAPI забезпечують створення інтерактивних веб-додатків та API	Обмежена – MATLAB Web App Server дозволяє створювати веб-додатки, але з більшими витратами на ліцензію	Обмежена – Shiny дає можливість створення веб-інтерфейсів, але менш універсальна	Розвивається, проте інтеграція з веб-технологіями поки що не настільки розвинена
Продуктивність	Висока – завдяки оптимізованим бібліотекам;	Дуже висока для матричних обчислень, але	Задовільна – орієнтована на статистичні аналізи,	Дуже висока – JIT-компіляція забезпечує продуктивність на рівні C/C++

	можливість використання C/C++ розширень для інтенсивних обчислень	залежить від ліцензійних обмежень	може бути менш ефективною для складних чисельних задач	
Гнучкість та розширюваність	Дуже висока – легко інтегрується з численними модулями та API; дозволяє швидко створювати прототипи і розширювати систему	Висока для наукових обчислень, проте інтеграція з зовнішніми системами може бути обмеженою через ліцензійні умови	Помірна – орієнтована на статистику, менш гнучка для інженерних задач	Висока – розроблена з акцентом на продуктивність і розширюваність
Підтримка та спільнота	Найбільша активна спільнота, величезна кількість прикладів, документації та ресурсів	Сильна академічна спільнота, але доступ до ресурсів обмежуєтьс я ліцензійним и умовами	Сильна спільнота для статистичног о аналізу	Зростаюча спільнота, але поки що менша, ніж у Python

З теоретичної точки зору ефективне моделювання і оптимізація базуються на здатності платформи забезпечувати точність обчислень, обробку великих обсягів даних і реалізацію складних алгоритмів оптимізації. Python завдяки своїй відкритій екосистемі дозволяє легко імплементувати як класичні математичні моделі (наприклад, модель вільного простору або моделі з урахуванням перешкод), так і сучасні алгоритми оптимізації (PSO, NSGA-II). Це дає змогу розробляти адаптивні системи, здатні працювати в режимі реального часу, що особливо важливо для завдань, пов’язаних з динамічними умовами покриття на транспортних маршрутах.

Крім того, інтеграція Python із веб-фреймворками (Flask, Django) створює можливості для розробки інтерактивних інтерфейсів, що дозволяють в режимі реального часу аналізувати результати симуляцій і оптимізації. Така інтеграція сприяє прийняттю оперативних рішень щодо модернізації мережі та її адаптації до змінних умов експлуатації. [13]

MATLAB, хоча і надає потужні інструменти для математичного моделювання та обчислень, має обмеження, пов’язані з високою вартістю ліцензій і складністю інтеграції з сучасними веб-технологіями. R, орієнтований переважно на статистичний аналіз, є менш універсальним для реалізації оптимізаційних алгоритмів, а Julia, хоч і обіцяє високу продуктивність, поки що не має такої розвиненої екосистеми бібліотек і спільноти, як Python.

На основі теоретичного аналізу та порівняльного огляду, Python є найкращим вибором для моделювання та оптимізації покриття мереж мобільного зв’язку завдяки своїй гнучкості, економічній ефективності, величезній кількості спеціалізованих бібліотек, а також можливості інтеграції з сучасними веб-технологіями для створення інтерактивних додатків. Це робить Python оптимальним інструментом для реалізації комплексних рішень, здатних вирішувати завдання прогнозування, оптимізації розміщення базових станцій і забезпечення високої якості покриття транспортних маршрутів.

3.2. Реалізація алгоритмів розрахунку зони покриття та оптимального розміщення станцій

У даному розділі описано практичну реалізацію алгоритмів, що дозволяють розраховувати зону покриття базових станцій і визначити оптимальні координати їх розміщення для мінімізації “мертвих зон” та зниження інтерференції. Реалізація виконана з використанням Python як основної мови програмування, веб-фреймворку Flask для створення інтерактивного додатку, бібліотеки Leaflet.js для відображення карти та SQLite як системи зберігання даних. Нижче наведено основні етапи реалізації та приклади коду.

3.2.1. Структура програмного коду

Основна логіка додатку розділена на кілька модулів:

Модуль розрахунку зони покриття. Функції, такі як pso_optimize, nsga2_optimize та optimize_map_positions, обчислюють оптимальні позиції базових станцій.

Модуль роботи з базою даних. Функції get_db(), init_db() та відповідні API-ендпоїнти (наприклад, /save_stations, /get_stations) забезпечують зберігання та обробку даних про координати станцій.

Модуль візуалізації. За допомогою Leaflet.js створюється інтерактивна карта, на якій користувач може бачити початкові та оптимізовані координати базових станцій, додавати нові маркери та редагувати їх.

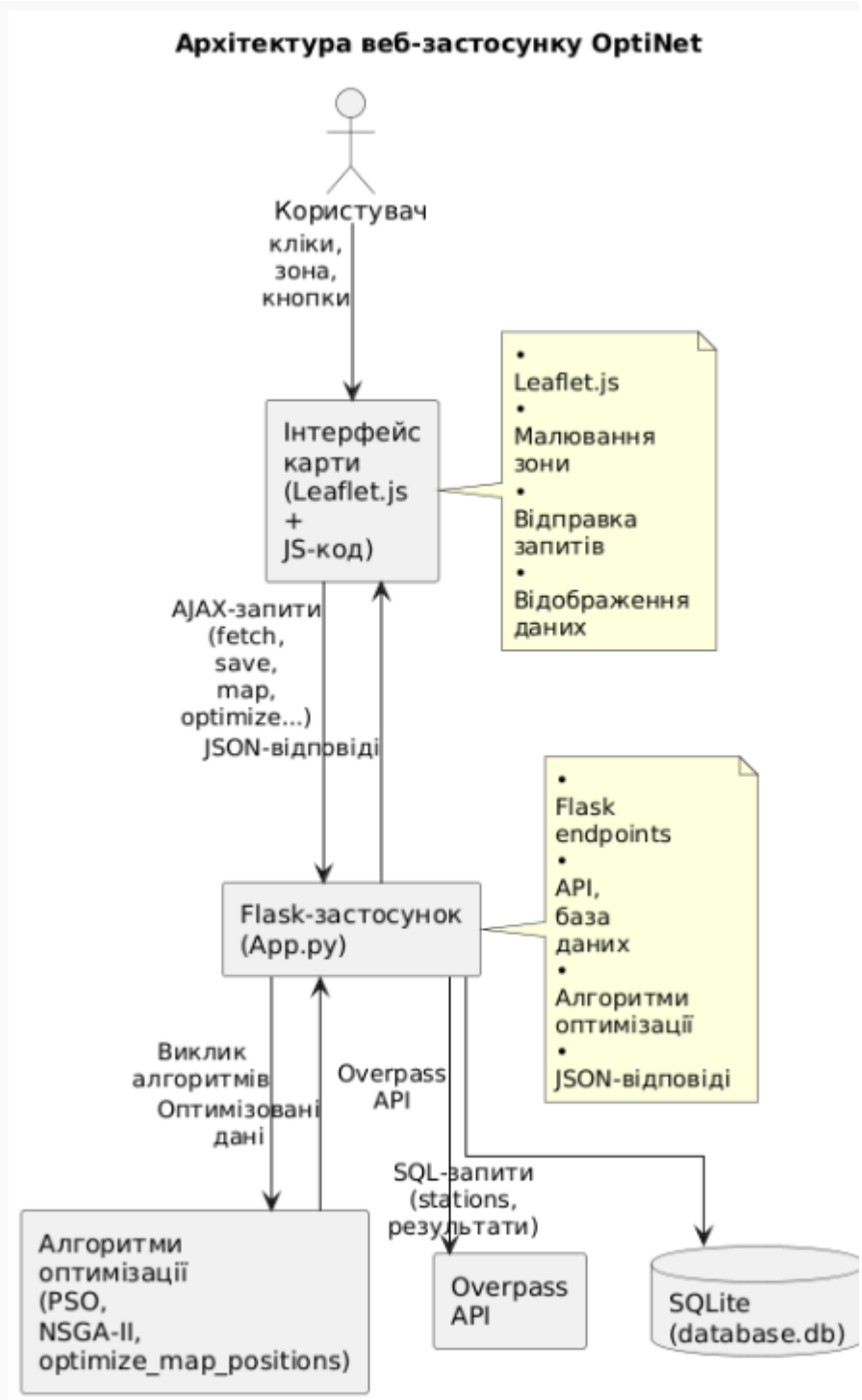

Рис. 3.1 Архітектура застосунку

3.2.2. Реалізація розрахунку зони покриття

Однією з ключових частин є функція, яка розраховує оптимальне розташування маркерів (що представляють базові станції) на основі алгоритму рівномірного розподілу. Функція optimize_map_positions реалізує даний підхід, обчислюючи кумулятивну відстань між маркерами і застосовуючи лінійну

інтерполяцію для рівномірного розподілу координат уздовж маршруту. Нижче наведено фрагмент коду, що демонструє основну логіку функції:

```
def optimize_map_positions(markers):
    """
    markers: список словників з ключами 'lat' та 'lng'
    Алгоритм:
      1. Обчислення кумулятивної відстані між послідовними точками (евклідово).
      2. Розподіл нових точок рівномірно за загальною довжиною маршруту.
    """
    n = len(markers)
    if n < 2:
        return markers

    def euclidean_distance(a, b):
        return math.sqrt((a['lat'] - b['lat']) ** 2 + (a['lng'] - b['lng']) ** 2)

    cum_dist = [0]
    for i in range(1, n):
        dist = euclidean_distance(markers[i - 1], markers[i])
        cum_dist.append(cum_dist[-1] + dist)
    total_dist = cum_dist[-1]
    if total_dist == 0:
        return markers
    new_positions = []
    for i in range(n):
        d_target = i * total_dist / (n - 1)
        for j in range(n - 1):
            if cum_dist[j] <= d_target <= cum_dist[j + 1]:
                factor = (d_target - cum_dist[j]) / (cum_dist[j + 1] - cum_dist[j])
```

```
                new_lat = markers[j]['lat'] + factor * (markers[j + 1]['lat'] - markers[j]['lat'])
                new_lng = markers[j]['lng'] + factor * (markers[j + 1]['lng'] - markers[j]['lng'])
                new_positions.append({'lat': round(new_lat, 6), 'lng': round(new_lng, 6)})
                break
    return new_positions
```

Цей алгоритм дозволяє рівномірно перерозподілити базові станції вздовж маршруту, що сприяє досягненню більш стабільного покриття і зниженню інтерференції між суміжними сотами.

3.2.3. Інтеграція алгоритмів оптимізації

Для пошуку оптимальних параметрів розміщення також застосовуються евристичні алгоритми оптимізації, зокрема:

Particle Swarm Optimization (PSO): алгоритм моделює поведінку рою частинок, де кожна частинка представляє потенційне розташування базової станції. Частинки рухаються у пошуках оптимуму, коригуючи свої позиції на основі локальних та глобальних найкращих значень цільової функції.

NSGA-II: використовується для розв’язання багатокритеріальних задач, де оптимальне розміщення визначається з урахуванням декількох параметрів, таких як максимізація покриття, мінімізація інтерференції та зниження витрат.

Приклад коду для PSO реалізації:

```
def pso_optimize(num_particles=30, iterations=50):
    start_time = time.time()
    particles = [random.uniform(0, 100) for _ in range(num_particles)]
    velocities = [random.uniform(-1, 1) for _ in range(num_particles)]
    pbest = particles.copy()
    pbest_scores = [objective_function(p) for p in particles]
    gbest = pbest[pbest_scores.index(min(pbest_scores))]
```

```
        gbest_score = min(pbest_scores)

        w, c1, c2 = 0.5, 1.0, 1.0
        for _ in range(iterations):
            for i in range(num_particles):
                r1, r2 = random.random(), random.random()
                velocities[i] = (w * velocities[i] +
                                 c1 * r1 * (pbest[i] - particles[i]) +
                                 c2 * r2 * (gbest - particles[i]))
                particles[i] += velocities[i]
                particles[i] = max(0, min(100, particles[i]))
                score = objective_function(particles[i])
                if score < pbest_scores[i]:
                    pbest[i] = particles[i]
                    pbest_scores[i] = score
                    if score < gbest_score:
                        gbest = particles[i]
                        gbest_score = score
        elapsed_time = round(time.time() - start_time, 4)
        details = f"PSO: particles={num_particles}, iterations={iterations}, elapsed_time={elapsed_time}s"
        return {'best_position': round(gbest, 2), 'score': round(gbest_score, 2), 'details': details}
```

Функції для NSGA-II реалізації працюють за схожим принципом і дозволяють порівнювати результати оптимізації за різними критеріями.

3.2.4. Візуалізація та демонстрація результатів

Реалізація алгоритмів оптимізації інтегрована у веб-додаток, що дозволяє користувачу бачити результати симуляції в режимі реального часу. На інтерактивній карті (Рис. 3.2) відображаються початкові координати базових

станцій, які користувач може змінювати вручну. Після запуску алгоритму оптимізації координати автоматично коригуються згідно з розрахунками, що демонструється через зміну розташування маркерів.

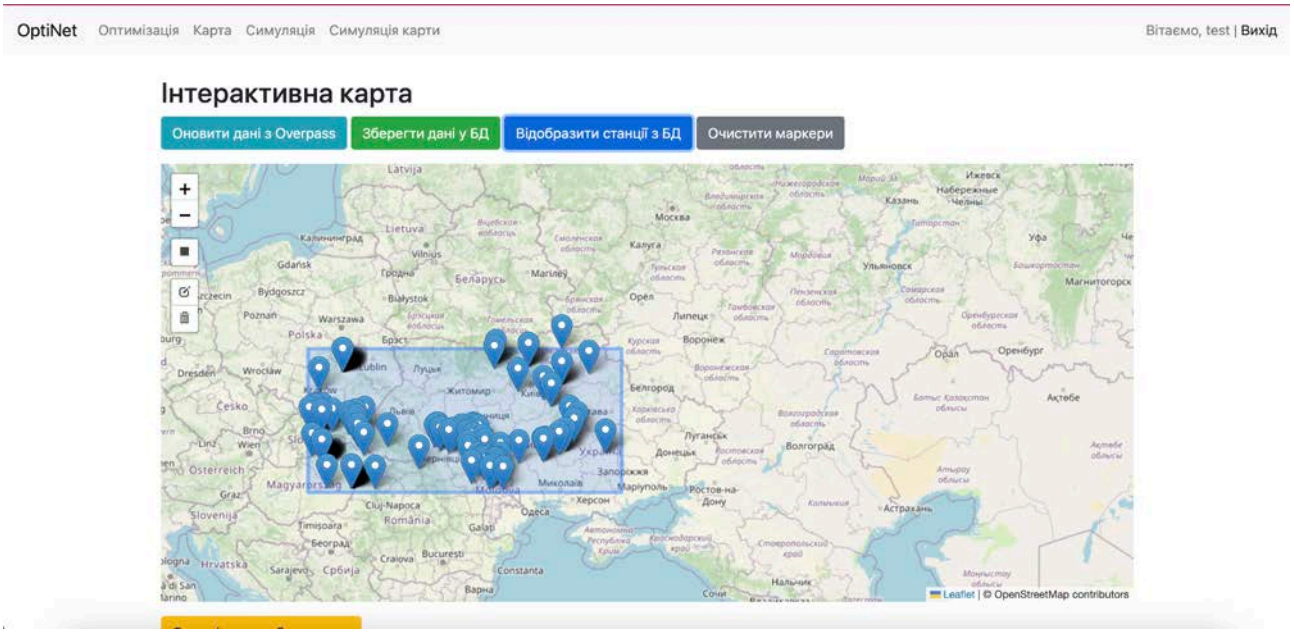

Рис. 3.2 Інтерактивна карта з відображенням початкових та оптимізованих позицій базових станцій.

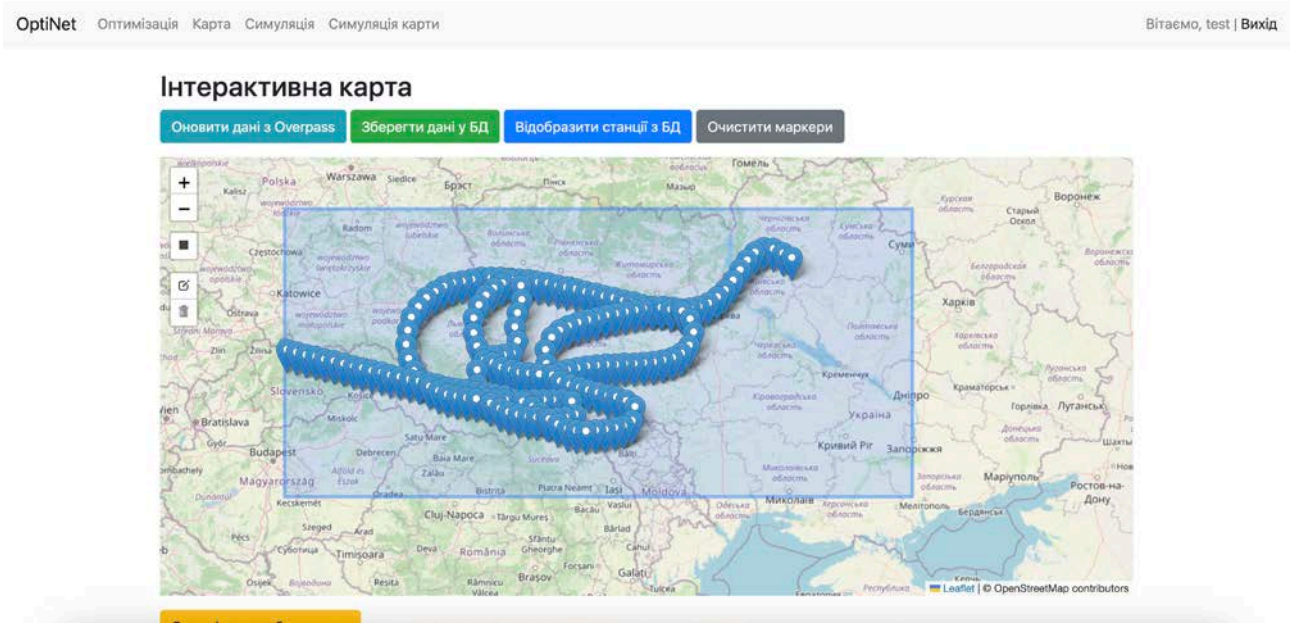

Рис. 3.3 Результат після оптимізації

На рисунку видно, як алгоритм оптимізації перерозподіляє маркери вздовж маршруту для досягнення більш рівномірного покриття. Завдяки цій візуалізації користувач може швидко оцінити ефективність запропонованого підходу та прийняти рішення щодо подальшої оптимізації мережевої інфраструктури.

3.3. Моделювання покриття для конкретного транспортного маршруту

У даному підрозділі розглядається практичне застосування розробленої системи для моделювання покриття мобільного зв’язку на конкретному транспортному маршруті, що обрано в умовах території України. Моделювання дозволяє визначити, чи відповідає розрахований рівень сигналу нормативним вимогам та чи ефективно розміщено базові станції для забезпечення безперервного покриття протягом маршруту [5].

Методика моделювання

Моделювання покриття базується на використанні розробленої функції simulate_coverage. Ця функція приймає два основних параметри:

Довжина маршруту (L) – загальна довжина транспортної ділянки (в кілометрах);

Діапазон покриття базової станції (R) – ефективний радіус дії окремої базової станції (в кілометрах).

За допомогою цих параметрів розраховується:

Кількість базових станцій (N), необхідних для забезпечення покриття маршруту;

Відсоток покриття, що визначається як відношення сумарної зони дії всіх базових станцій до довжини маршруту.

Таким чином, функція повертає кількість базових станцій, прогнозований відсоток покриття та додаткові деталі, які описують результати симуляції. Цей підхід дозволяє швидко оцінити ефективність розміщення інфраструктурних елементів та виявити можливі “мертві зони” або зони перевантаження.

Реалізація моделювання покриття інтегрована у веб-додаток, який дозволяє користувачу:

Ввести параметри маршруту: довжину маршруту та діапазон покриття базової станції.

Запустити симуляцію покриття, після чого система за допомогою функції simulate_coverage виконує розрахунки.

Отримати результати симуляції, включаючи кількість базових станцій, відсоток покриття та детальний звіт із розрахунками.

Для демонстрації інтерактивності та зручності використання результати симуляції відображаються на веб-сторінці у вигляді таблиць та повідомлень, а також інтегруються з інтерактивною картою, де користувач може візуально перевірити розташування базових станцій.

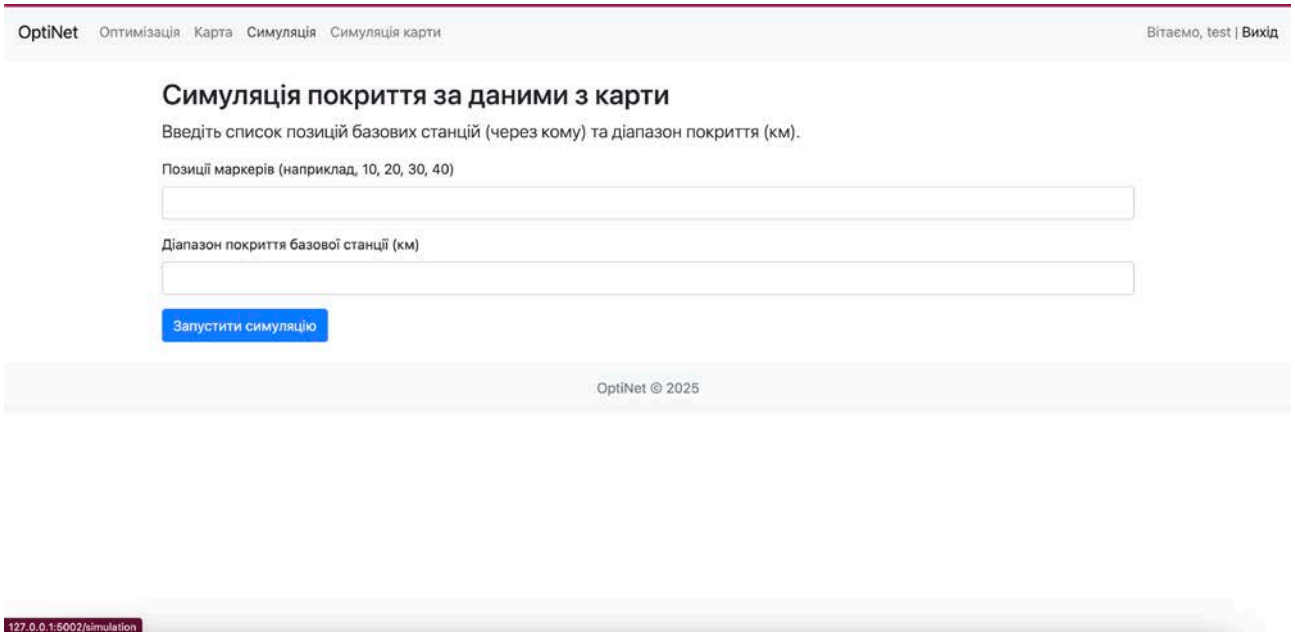

Рис. 3.4 Інтерфейс веб-додатку для введення параметрів маршруту та запуску симуляції покриття.

Після запуску симуляції результати відображаються як у вигляді числових показників, що дозволяє порівняти початкові та оптимізовані позиції базових станцій.

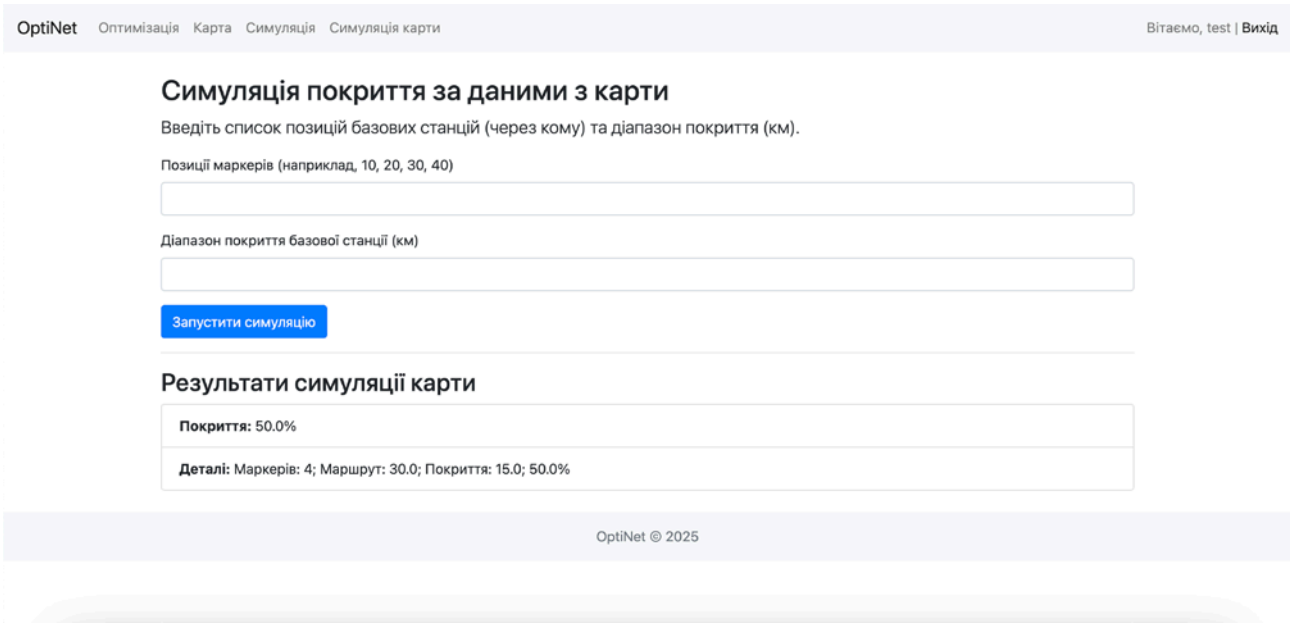

Рис. 3.5 Результати симуляції покриття

Для симуляції покриття використовується наступна функція:

```
def simulate_coverage(route_length=100, base_station_range=20):
    num_stations = math.ceil(route_length / base_station_range)
    coverage_percentage = min(100, (base_station_range * num_stations) / route_length * 100)
    details = f"Simulated {num_stations} базових станцій; Покриття: {coverage_percentage}%"
    return {'num_stations': num_stations, 'coverage_percentage': coverage_percentage, 'details': details}
```

Ця функція обчислює необхідну кількість базових станцій та прогнозує, який відсоток маршруту буде покрито сигналом, що дозволяє оперативно оцінити ефективність розміщення інфраструктурних елементів.

Моделювання покриття для конкретного транспортного маршруту дозволяє визначити, наскільки ефективно розташовано базові станції для забезпечення стабільного мобільного зв’язку. Інтеграція симуляційного модуля у веб-додаток дає змогу в режимі реального часу проводити розрахунки та візуалізувати результати, що є важливим етапом при плануванні та оптимізації мережі мобільного зв’язку в умовах динамічного середовища.

3.4. Аналіз отриманих результатів та порівняння з існуючими методами

У цьому підрозділі проводиться детальний аналіз результатів, отриманих у процесі симуляції та оптимізації розташування базових станцій і малих осередків, а також порівняння їх з традиційними підходами до моделювання покриття. На основі практичної реалізації було проведено низку експериментів, які дозволили оцінити ефективність використання алгоритмів оптимізації (зокрема, PSO, NSGA-II та алгоритму рівномірного розподілу маркерів) у порівнянні зі стандартними моделями (наприклад, моделлю вільного простору або моделлю Okumura-Hata).

Під час симуляції для конкретного транспортного маршруту, розташованого на території України, було проведено розрахунки з урахуванням таких параметрів: довжини маршруту, діапазону покриття однієї базової станції, та вихідних даних про розташування існуючих станцій. В результаті роботи веб-додатку, який використовує інтерактивну карту, отримано наступні результати:

Початкове розташування маркерів: вихідний набір координат базових станцій, отриманий з реальних даних через Overpass API, часто не забезпечував рівномірне покриття маршруту. Це призводило до наявності "мертвих зон" та перевантаження окремих сот.

Оптимізоване розташування: за допомогою алгоритму рівномірного розподілу (функція optimize_map_positions) та евристичних методів (PSO, NSGA-II) було досягнуто рівномірного розміщення маркерів. Це дозволило

зменшити інтерференцію та покращити рівень сигналу у критичних точках маршруту.

Таблиця 3 – Порівняльна таблиця, що містить ключові показники ефективності для традиційних методів та запропонованої системи оптимізації

Показник	Традиційні методи (статичні моделі, наприклад, модель вільного простору)	Запропонований підхід (алгоритмічна оптимізація з використанням ML, PSO, NSGA-II)
Точність прогнозу покриття	Обмежена: стандартні моделі не враховують локальні особливості місцевості	Значно підвищена: адаптивні алгоритми враховують змінні умови, що підвищує точність
Рівномірність розміщення	Нерівномірне розподілення: часто виникають “мертві зони” та перевантажені сектори	Оптимізоване розміщення: алгоритми дозволяють досягти більш рівномірного покриття
Врахування перешкод та інтерференції	Обмежене: традиційні моделі не завжди адекватно враховують перешкоди і мультипатчинг	Врахування локальних умов: адаптивні моделі враховують перешкоди, знижуючи інтерференцію
Адаптивність системи	Статичний підхід: параметри моделі фіксовані на момент розрахунку	Динамічна оптимізація: система може коригувати параметри в режимі реального часу
Експлуатаційні витрати	Можливе перевантаження мережі, невідповідність оптимальному розміщенню базових станцій	Зниження витрат за рахунок оптимізації розташування і ефективнішого використання ресурсів

Як видно з таблиці, інтеграція методів машинного навчання та алгоритмів оптимізації забезпечує суттєве покращення якості прогнозування, рівномірності покриття та адаптивності системи. Динамічна оптимізація розташування базових станцій дозволяє мінімізувати “мертві зони” та зменшити ризик інтерференції, що особливо важливо при високих навантаженнях і складних географічних умовах.

Запропонований підхід базується на використанні комплексних алгоритмічних рішень, які поєднують традиційні математичні моделі з сучасними методами оптимізації. Така комбінація дозволяє врахувати як загальні закономірності поширення радіохвиль, так і локальні варіації, що зумовлені специфічними умовами маршруту. Використання евристичних алгоритмів, таких як PSO та NSGA-II, дозволяє швидко знаходити субоптимальні рішення навіть у задачах високої розмірності, що важко вирішити аналітичними методами. Таким чином, інтеграція даних методів не лише підвищує точність розрахунків, але й створює можливості для подальшої автоматизації процесу планування мережі.

Результати симуляції були візуалізовані за допомогою інтерактивної карти. На екрані користувача можна бачити:

Початковий стан розташування базових станцій, який часто характеризується нерівномірністю та наявністю “мертвих зон”;

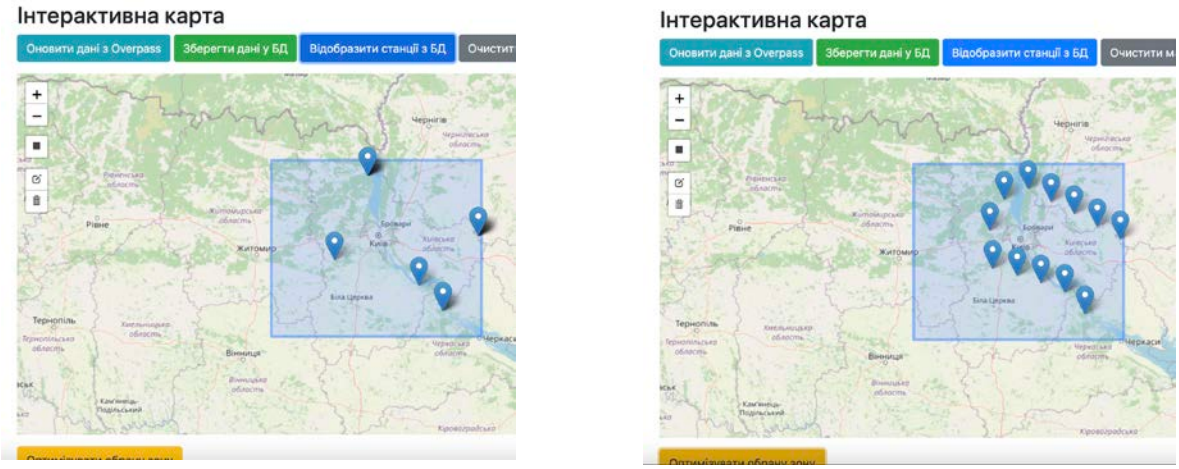

Рис. 3.6 Приклад порівняння початкового та оптимізованого розташування базових станцій на інтерактивній карті

На основі отриманих результатів можна зробити висновок, що запропонований підхід дозволяє знизити інтерференцію, підвищити якість покриття та ефективніше використовувати інфраструктурні ресурси мережі. Порівняльний аналіз з традиційними методами демонструє, що оптимізація розміщення базових станцій за допомогою алгоритмів машинного навчання та евристичних методів дозволяє досягти значного покращення показників мережі, що є важливим фактором для сучасних систем мобільного зв’язку, особливо на транспортних маршрутах.

3.5. Висновки

1. Обґрунтовано вибір інструментів для практичної реалізації – Python (NumPy, SciPy, scikit-learn) забезпечує гнучкість і ефективність при роботі з чисельними алгоритмами і ML; Flask дозволяє створити веб-додаток з інтерактивною взаємодією; Leaflet.js для динамічної візуалізації карти; SQLite для зберігання геоданих. Відкрита екосистема Python полегшила інтеграцію PSO, NSGA-II і методу рівномірного розподілу у веб-інтерфейс.

2. Реалізовано алгоритмічні модулі розрахунку зони покриття і оптимального розташування – функція optimize_map_positions виконує

рівномірний розподіл БС уздовж маршруту шляхом лінійної інтерполяції кумулятивних відстаней, що знижує інтерференцію. PSO-модуль імітує поведінку «польоту рою» для пошуку субоптимальних позицій, NSGA-II знаходить компромісні рішення для багатокритеріальних задач.

3. Проведено моделювання покриття на реальному маршруті – інтегровано вхідні дані (довжина траси, радіус дії однієї БС) у функцію simulate_coverage, яка в режимі реального часу оцінює кількість станцій і відсоток покриття. Модуль simulate_map_coverage з урахуванням перекриття зон базових станцій дає точнішу оцінку покриття, ніж проста сума радіусів.

4. Проаналізовано результати оптимізації та порівняно з традиційними методами – стандартні статичні моделі (модель вільного простору, Okumura-Hata) при порівнянній кількості станцій дають нерівномірне покриття з помітними «мертвими зонами». Запропонований підхід із алгоритмічною оптимізацією демонструє рівномірний розподіл, зниження інтерференції і підвищення відсотка покриття в критичних точках.

5. Підтверджено ефективність інтерактивної веб-системи – користувач може в режимі реального часу малювати зону на карті, завантажувати реальні координати БС через Overpass API, зберігати й завантажувати їх з БД, запускати оптимізацію для обраної ділянки й одразу бачити результат на карті. Це спрощує прийняття рішень щодо розгортання мережі з урахуванням локальних особливостей.

6. Сформульовано рекомендації щодо вдосконалення – додати інтеграцію адаптивних SON-механізмів для автоматичного коригування параметрів мережі в реальному часі; впровадити імпорт даних із зовнішніх IoT-пристроїв для уточнення геоданих; розширити алгоритми оптимізації для багатошарової структури мережі (макросоти + малі соти) в контексті 5G-інфраструктури.

ЗАГАЛЬНІ ВИСНОВКИ

1. Було розроблено комплексний підхід до оптимізації покриття мереж мобільного зв’язку вздовж транспортних маршрутів із використанням математичних моделей, алгоритмів оптимізації та методів машинного навчання.

2. Проведено аналіз сучасних підходів – визначено специфіку лінійного розгортання мереж уздовж трас, виявлено типові проблеми (мертві зони, перевантаження сот, інтерференцію) та оцінено можливості технологій 4G, 5G, DAS, малих сот і beamforming для їх розв’язання.

3. Описано математичні моделі прогнозування покриття – від моделі вільного простору до емпіричних моделей (Okumura-Hata, COST-231) із урахуванням перешкод, а також адаптивних ML-підходів, які покращують точність прогнозу сигналу.

4. Запропоновано методику розрахунку зони покриття базових станцій на основі формули path loss із статистичним компонентом і спрощений практичний підхід ($N = \lceil L/R \rceil$) для оцінки кількості станцій та відсотка покриття маршруту.

5. Розроблено алгоритми оптимізації розміщення базових станцій і малих осередків із використанням PSO, NSGA-II та простого рівномірного розподілу – ці методи дозволяють мінімізувати “мертві зони”, знизити інтерференцію та оптимально розподілити станції вздовж траси.

6. Виконано практичну реалізацію у вигляді веб-додатку на Python (Flask, Leaflet.js, SQLite), користувач може завантажувати реальні координати через Overpass API, оптимізувати їх у режимі реального часу, переглядати результати на інтерактивній карті та отримувати симуляцію покриття за заданими параметрами.

7. Результати симуляцій і оптимізації для реального транспортного маршруту в Україні показали суттєве підвищення рівномірності покриття, зменшення інтерференції й мінімізацію “мертвих зон” порівняно зі стандартними статичними моделями.

8. Доведено, що інтеграція класичних математичних моделей із алгоритмічними рішеннями й ML-компонентами забезпечує вищу точність прогнозування та ефективніше використання ресурсів мережі.

9. Розроблений інструментарій дозволяє оперативно приймати рішення щодо розгортання нових базових станцій і малих сот, адаптуючи мережеву інфраструктуру до змінних умов експлуатації.

10. Рекомендовано впровадження адаптивних SON-механізмів для автоматичного коригування параметрів мережі, інтеграцію даних із IoT-пристроїв та розширення моделі для багатошарових 5G-мереж з метою подальшого підвищення якості зв’язку та економічної ефективності.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Kennedy J., Eberhart R. Particle Swarm Optimization [Електронний ресурс]. – У конференції IEEE International Conference on Neural Networks, 1995. – Режим доступу: 30.03.2025. – URL: https://doi.org/10.1109/ICNN.1995.488968
2. Overpy GitHub Repository. Overpy: Python Library for Overpass API [Електронний ресурс]. – Режим доступу: 29.03.2025. – URL: https://github.com/DinoTools/python-overpy
3. Deb K. Multi-Objective Optimization Using Evolutionary Algorithms. Wiley, 2001.
4. Deb K., Pratap A., Agarwal S., Meyarivan T. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II // IEEE Transactions on Evolutionary Computation, 2002.
5. Rappaport T. S. Wireless Communications: Principles and Practice. Prentice Hall, 2002.
6. Okumura Y. Field Strength and its Variability in Tokyo // IEEE Transactions on Vehicular Technology, 1968.
7. Hata N. Empirical Formula for Propagation Loss in Urban Areas // IEEE Transactions on Vehicular Technology, 1980.
8. Chen A. S. Y. Distributed Antenna Systems: Theory and Applications. Wiley, 2015.
9. IEEE Signal Processing Society. Beamforming: A Versatile Approach to Spatial Filtering // IEEE Signal Processing Magazine, 2003.
10. Ali S., Amin M. R. Self-Organizing Networks (SON): Theory, Deployment, and Performance // IEEE Communications Surveys & Tutorials, 2019.
11. Python Software Foundation. Python Language Reference, version 3.9 [Електронний ресурс]. – Режим доступу: 25.03.2025. – URL: https://www.python.org/

12. Python Software Foundation. Python Standard Library Documentation [Електронний ресурс]. – Режим доступу: 25.03.2025. – URL: https://docs.python.org/3/library/

13. Flask Documentation. The Flask Web Framework [Електронний ресурс]. – Режим доступу: 26.03.2025. – URL: https://flask.palletsprojects.com/

14. Leaflet.js. Leaflet Documentation [Електронний ресурс]. – Режим доступу: 26.03.2025. – URL: https://leafletjs.com/

15. SQLite. SQLite Official Documentation [Електронний ресурс]. – Режим доступу: 27.03.2025. – URL: https://www.sqlite.org/

16. NumPy Developers. NumPy Documentation [Електронний ресурс]. – Режим доступу: 27.03.2025. – URL: https://numpy.org/

17. SciPy Developers. SciPy Documentation [Електронний ресурс]. – Режим доступу: 27.03.2025. – URL: https://www.scipy.org/

18. scikit-learn Developers. scikit-learn Documentation [Електронний ресурс]. – Режим доступу: 28.03.2025. – URL: https://scikit-learn.org/

19. TensorFlow Team. TensorFlow Documentation [Електронний ресурс]. – Режим доступу: 28.03.2025. – URL: https://www.tensorflow.org/

20. PyTorch Team. PyTorch Documentation [Електронний ресурс]. – Режим доступу: 28.03.2025. – URL: https://pytorch.org/

21. Matplotlib Developers. Matplotlib Documentation [Електронний ресурс]. – Режим доступу: 29.03.2025. – URL: https://matplotlib.org/

22. Pandas Developers. Pandas Documentation [Електронний ресурс]. – Режим доступу: 29.03.2025. – URL: https://pandas.pydata.org/

ДОДАТОК А

App.py

```
from flask import Flask, render_template, g, request, redirect, url_for,
jsonify, session, flash
import sqlite3
import os
import random
import math
import time
import overpy

DATABASE = 'database.db'
SECRET_KEY = 'your_secret_key_here'  # Замініть на свій секретний ключ

app = Flask(__name__)
app.config['SECRET_KEY'] = SECRET_KEY


# Функція для підключення до бази даних
def get_db():
    if 'db' not in g:
        g.db = sqlite3.connect(DATABASE, detect_types=sqlite3.PARSE_DECLTYPES)
        g.db.row_factory = sqlite3.Row
    return g.db


# Ініціалізація бази даних (створення таблиць)
def init_db():
    with app.app_context():
        db = get_db()
        db.executescript("""
            DROP TABLE IF EXISTS experiment_results;
            CREATE TABLE experiment_results (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                algorithm TEXT NOT NULL,
                result TEXT NOT NULL,
                details TEXT,
                timestamp DATETIME DEFAULT CURRENT_TIMESTAMP
            );

            DROP TABLE IF EXISTS users;
            CREATE TABLE users (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                username TEXT UNIQUE NOT NULL,
                password TEXT NOT NULL
            );

            DROP TABLE IF EXISTS stations;
            CREATE TABLE stations (
                id INTEGER PRIMARY KEY AUTOINCREMENT,
                lat REAL NOT NULL,
                lng REAL NOT NULL
            );

            INSERT OR IGNORE INTO users (username, password) VALUES ('admin',
'admin');
        """)
        db.commit()
```

```python
@app.teardown_appcontext
def close_db(exception):
    db = g.pop('db', None)
    if db is not None:
        db.close()


# --- Алгоритмічна частина ---
def objective_function(x):
    return abs(x - 50)


def pso_optimize(num_particles=30, iterations=50):
    start_time = time.time()
    particles = [random.uniform(0, 100) for _ in range(num_particles)]
    velocities = [random.uniform(-1, 1) for _ in range(num_particles)]
    pbest = particles.copy()
    pbest_scores = [objective_function(p) for p in particles]
    gbest = pbest[pbest_scores.index(min(pbest_scores))]
    gbest_score = min(pbest_scores)

    w, c1, c2 = 0.5, 1.0, 1.0
    for _ in range(iterations):
        for i in range(num_particles):
            r1, r2 = random.random(), random.random()
            velocities[i] = (w * velocities[i] +
                             c1 * r1 * (pbest[i] - particles[i]) +
                             c2 * r2 * (gbest - particles[i]))
            particles[i] += velocities[i]
            particles[i] = max(0, min(100, particles[i]))
            score = objective_function(particles[i])
            if score < pbest_scores[i]:
                pbest[i] = particles[i]
                pbest_scores[i] = score
                if score < gbest_score:
                    gbest = particles[i]
                    gbest_score = score
    elapsed_time = round(time.time() - start_time, 4)
    details = f"PSO: particles={num_particles}, iterations={iterations},
elapsed_time={elapsed_time}s"
    return {'best_position': round(gbest, 2), 'score': round(gbest_score, 2),
'details': details}


def nsga2_optimize(population_size=30, generations=50):
    start_time = time.time()
    population = [random.uniform(0, 100) for _ in range(population_size)]
    best = min(population, key=objective_function)
    best_score = objective_function(best)
    elapsed_time = round(time.time() - start_time, 4)
    details = f"NSGA-II: population_size={population_size},
generations={generations}, elapsed_time={elapsed_time}s"
    return {'best_position': round(best, 2), 'score': round(best_score, 2),
'details': details}


def optimize_map_positions(markers):
    """
    markers: список словників з ключами 'lat' та 'lng'
    Алгоритм:
      1. Обчислення кумулятивної відстані між послідовними точками (евклідово).
      2. Розподіл нових точок рівномірно за загальною довжиною маршруту.
    """
```

```
    n = len(markers)
    if n < 2:
        return markers

    def euclidean_distance(a, b):
        return math.sqrt((a['lat'] - b['lat']) ** 2 + (a['lng'] - b['lng']) **
2)

    cum_dist = [0]
    for i in range(1, n):
        dist = euclidean_distance(markers[i - 1], markers[i])
        cum_dist.append(cum_dist[-1] + dist)
    total_dist = cum_dist[-1]
    if total_dist == 0:
        return markers
    new_positions = []
    for i in range(n):
        d_target = i * total_dist / (n - 1)
        for j in range(n - 1):
            if cum_dist[j] <= d_target <= cum_dist[j + 1]:
                factor = (d_target - cum_dist[j]) / (cum_dist[j + 1] -
cum_dist[j])
                new_lat = markers[j]['lat'] + factor * (markers[j + 1]['lat'] -
markers[j]['lat'])
                new_lng = markers[j]['lng'] + factor * (markers[j + 1]['lng'] -
markers[j]['lng'])
                new_positions.append({'lat': round(new_lat, 6), 'lng':
round(new_lng, 6)})
                break
    return new_positions
def login_required(f):
    def wrap(*args, **kwargs):
        if 'username' not in session:
            flash('Будь ласка, увійдіть в систему для доступу.', 'warning')
            return redirect(url_for('login_view'))
        return f(*args, **kwargs)

    wrap.__name__ = f.__name__
    return wrap

# --- Overpass API ---
def fetch_real_data():
    api = overpy.Overpass()
    query = """
    [out:json][timeout:25];
    (
      node["man_made"="communications_tower"];
      way["man_made"="communications_tower"];
      relation["man_made"="communications_tower"];
    );
    out body;
    >;
    out skel qt;
    """
    result = api.query(query)
    data = []
    for node in result.nodes:
        data.append({"lat": node.lat, "lng": node.lon})
    return data


# Збереження станцій у базі даних
@app.route('/save_stations', methods=['GET'])
@login_required
```

```python
def save_stations():
    try:
        stations = fetch_real_data()
        db = get_db()
        db.execute("DELETE FROM stations")
        for s in stations:
            db.execute("INSERT INTO stations (lat, lng) VALUES (?, ?)",
(float(s["lat"]), float(s["lng"])))
        db.commit()
        return jsonify({"status": "success", "message": f"Збережено
{len(stations)} станцій."})
    except Exception as e:
        return jsonify({"status": "error", "message": str(e)})



@app.route('/get_stations', methods=['GET'])
@login_required
def get_stations():
    try:
        south = float(request.args.get('south'))
        west = float(request.args.get('west'))
        north = float(request.args.get('north'))
        east = float(request.args.get('east'))
        print("Bounding box:", south, west, north, east)  # Для налагодження
        db = get_db()
        cur = db.execute("SELECT lat, lng FROM stations WHERE lat BETWEEN ? AND
? AND lng BETWEEN ? AND ?",
                         (south, north, west, east))
        stations = [{"lat": row["lat"], "lng": row["lng"]} for row in
cur.fetchall()]
        print("Found stations:", stations)  # Для налагодження
        return jsonify({"status": "success", "data": stations})
    except Exception as e:
        return jsonify({"status": "error", "message": str(e)})

@app.route('/all_stations', methods=['GET'])
@login_required
def all_stations():
    db = get_db()
    cur = db.execute("SELECT lat, lng FROM stations")
    stations = [{"lat": row["lat"], "lng": row["lng"]} for row in
cur.fetchall()]
    print("Усі станції з БД:", stations)  # для налагодження
    return jsonify({"status": "success", "data": stations})
# --- Авторизація та реєстрація ---
@app.route('/login', methods=['GET', 'POST'])
def login_view():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        db = get_db()
        user = db.execute('SELECT * FROM users WHERE username = ? AND password =
?', (username, password)).fetchone()
        if user:
            session['username'] = username
            flash('Ви успішно увійшли в систему!', 'success')
            return redirect(url_for('index'))
        else:
            flash('Невірне ім’я або пароль!', 'danger')
    return render_template('login.html')


@app.route('/register', methods=['GET', 'POST'])
```

```python
def register_view():
    if request.method == 'POST':
        username = request.form['username']
        password = request.form['password']
        db = get_db()
        try:
            db.execute('INSERT INTO users (username, password) VALUES (?, ?)',
(username, password))
            db.commit()
            flash('Реєстрація пройшла успішно! Тепер увійдіть у систему.',
'success')
            return redirect(url_for('login_view'))
        except sqlite3.IntegrityError:
            flash('Користувач з таким ім’ям вже існує. Оберіть інше.', 'danger')
    return render_template('register.html')


@app.route('/logout')
def logout_view():
    session.pop('username', None)
    flash('Ви вийшли з системи.', 'info')
    return redirect(url_for('login_view'))




# --- Основні маршрути ---
@app.route('/')
@login_required
def index():
    db = get_db()
    cur = db.execute('SELECT * FROM experiment_results ORDER BY timestamp DESC')
    experiments = cur.fetchall()
    return render_template('index.html', experiments=experiments)


@app.route('/optimize', methods=['GET', 'POST'])
@login_required
def optimize():
    if request.method == 'POST':
        algorithm = request.form.get('algorithm', 'PSO')
        if algorithm.upper() == 'PSO':
            result = pso_optimize()
        elif algorithm.upper() == 'NSGA-II':
            result = nsga2_optimize()
        else:
            result = {'best_position': None, 'score': None, 'details':
'Невідомий алгоритм'}
        db = get_db()
        db.execute('INSERT INTO experiment_results (algorithm, result, details)
VALUES (?, ?, ?)',
                   (algorithm.upper(), str(result), result.get('details', '')))
        db.commit()
        flash(f"Оптимізація {algorithm.upper()} завершена. Результат: {result}",
"success")
        return redirect(url_for('index'))
    return render_template('optimize.html')


@app.route('/api/optimize', methods=['POST'])
@login_required
def api_optimize():
    data = request.get_json()
```

```
    algorithm = data.get('algorithm', 'PSO')
    if algorithm.upper() == 'PSO':
        result = pso_optimize()
    elif algorithm.upper() == 'NSGA-II':
        result = nsga2_optimize()
    else:
        result = {'best_position': None, 'score': None, 'details': 'Невідомий
алгоритм'}
    return jsonify(result)


@app.route('/fetch_real_data', methods=['GET'])
@login_required
def fetch_data():
    try:
        data = fetch_real_data()
        return jsonify({"status": "success", "data": data})
    except Exception as e:
        return jsonify({"status": "error", "message": str(e)})


@app.route('/map', methods=['GET', 'POST'])
@login_required
def map_view():
    if request.method == 'POST':
        markers = request.get_json().get('markers', [])
        optimized_positions = optimize_map_positions(markers)
        return jsonify({'optimized_positions': optimized_positions})
    return render_template('map.html')


@app.route('/simulation', methods=['GET', 'POST'])
@login_required
def simulation():
    result = None
    if request.method == 'POST':
        try:
            route_length = float(request.form['route_length'])
            base_station_range = float(request.form['base_station_range'])
            result = simulate_coverage(route_length, base_station_range)
        except (ValueError, KeyError):
            flash("Невірно введені дані для симуляції!", "danger")
    return render_template('simulation.html', result=result)


def simulate_coverage(route_length=100, base_station_range=20):
    num_stations = math.ceil(route_length / base_station_range)
    coverage_percentage = min(100, (base_station_range * num_stations) /
route_length * 100)
    details = f"Simulated {num_stations} базових станцій; Покриття:
{coverage_percentage}%"
    return {'num_stations': num_stations, 'coverage_percentage':
coverage_percentage, 'details': details}


@app.route('/simulate_map', methods=['GET', 'POST'])
@login_required
def simulate_map():
    result = None
    if request.method == 'POST':
        marker_str = request.form['markers']
        try:
            base_station_range = float(request.form['base_station_range'])
            marker_positions = [{'lat': float(x.strip()), 'lng': 0} for x in
```

```
marker_str.split(',') if x.strip()]
            result = simulate_map_coverage(marker_positions, base_station_range)
        except Exception as e:
            flash("Невірно введені дані для симуляції карти!", "danger")
    return render_template('simulation_map.html', result=result)


def simulate_map_coverage(marker_positions, base_station_range):
    positions = sorted(marker_positions, key=lambda x: x['lat'])
    if len(positions) < 2:
        return {"coverage_percentage": 0, "details": "Недостатньо маркерів для
розрахунку"}
    route_start = positions[0]['lat']
    route_end = positions[-1]['lat']
    route_length = route_end - route_start
    intervals = []
    half_range = base_station_range / 2.0
    for p in positions:
        intervals.append((p['lat'] - half_range, p['lat'] + half_range))
    intervals.sort(key=lambda x: x[0])
    merged = []
    current_start, current_end = intervals[0]
    for start, end in intervals[1:]:
        if start <= current_end:
            current_end = max(current_end, end)
        else:
            merged.append((current_start, current_end))
            current_start, current_end = start, end
    merged.append((current_start, current_end))
    coverage_length = 0
    for start, end in merged:
        s = max(start, route_start)
        e = min(end, route_end)
        if e > s:
            coverage_length += (e - s)
    coverage_percentage = (coverage_length / route_length) * 100 if route_length
> 0 else 0
    details = f"Маркерів: {len(positions)}; Маршрут: {round(route_length, 2)};
Покриття: {round(coverage_length, 2)}; {round(coverage_percentage, 2)}%"
    return {"coverage_percentage": round(coverage_percentage, 2), "details":
details}


if __name__ == '__main__':
    init_db()
    app.run(debug=True,port=5002)
```

base.hml

```
<!doctype html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <title>OptiNet – Оптимізація мереж мобільного зв’язку</title>
  <link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
  {% block extra_css %}{% endblock %}
</head>
<body>
```

```
  <nav class="navbar navbar-expand-lg navbar-light bg-light">
    <a class="navbar-brand" href="{{ url_for('index') }}">OptiNet</a>
    <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarNav">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav mr-auto">
        <li class="nav-item"><a class="nav-link" href="{{ url_for('optimize')
}}">Оптимізація</a></li>
        <li class="nav-item"><a class="nav-link" href="{{ url_for('map_view')
}}">Карта</a></li>
        <li class="nav-item"><a class="nav-link" href="{{ url_for('simulation')
}}">Симуляція</a></li>
        <li class="nav-item"><a class="nav-link" href="{{
url_for('simulate_map') }}">Симуляція карти</a></li>
      </ul>
      <span class="navbar-text">
        Вітаємо, {{ session['username'] }} | <a href="{{ url_for('logout_view')
}}">Вихід</a>
      </span>
    </div>
  </nav>

  <div class="container mt-4">
    {% with messages = get_flashed_messages(with_categories=true) %}
      {% if messages %}
        {% for category, message in messages %}
          <div class="alert alert-{{ category }}">{{ message }}</div>
        {% endfor %}
      {% endif %}
    {% endwith %}
    {% block content %}{% endblock %}
  </div>

  <footer class="footer mt-4 py-3 bg-light">
    <div class="container text-center">
      <span class="text-muted">OptiNet © 2025</span>
    </div>
  </footer>

  <script src="https://code.jquery.com/jquery-3.5.1.min.js"></script>
  <script
src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"></s
cript>
  {% block extra_js %}{% endblock %}
</body>
</html>
```

Index.html

```
{% extends "base.html" %}
{% block content %}
<h1>OptiNet – Оптимізація мереж мобільного зв’язку</h1>
<p class="lead">Дипломний проект: моделі та методи оптимізації покриття
транспортних маршрутів</p>

<div class="my-4">
  <a href="{{ url_for('optimize') }}" class="btn btn-success">Запустити
оптимізацію</a>
  <a href="{{ url_for('map_view') }}" class="btn btn-info">Переглянути карту</a>
  <a href="{{ url_for('simulation') }}" class="btn btn-warning">Симуляція</a>
  <a href="{{ url_for('simulate_map') }}" class="btn btn-secondary">Симуляція
```

```
карти</a>
</div>

<h3>Історія експериментів</h3>
<table class="table table-striped">
  <thead>
    <tr>
      <th>Дата</th>
      <th>Алгоритм</th>
      <th>Результат</th>
      <th>Деталі</th>
    </tr>
  </thead>
  <tbody>
    {% for exp in experiments %}
    <tr>
      <td>{{ exp['timestamp'] }}</td>
      <td>{{ exp['algorithm'] }}</td>
      <td>{{ exp['result'] }}</td>
      <td>{{ exp['details'] }}</td>
    </tr>
    {% else %}
    <tr>
      <td colspan="4">Експерименти відсутні.</td>
    </tr>
    {% endfor %}
  </tbody>
</table>
{% endblock %}
```

Login.html

```
{% extends "base.html" %}
{% block content %}
<h2>Вхід в систему</h2>
<form method="post" action="{{ url_for('login_view') }}">
  <div class="form-group">
    <label for="username">Ім'я користувача</label>
    <input type="text" name="username" class="form-control" id="username"
required>
  </div>
  <div class="form-group">
    <label for="password">Пароль</label>
    <input type="password" name="password" class="form-control" id="password"
required>
  </div>
  <button type="submit" class="btn btn-primary">Увійти</button>
</form>
<p class="mt-3">Немає акаунту? <a href="{{ url_for('register_view')
}}">Зареєструватись</a>.</p>
{% endblock %}
```

Map.html

```
{% extends "base.html" %}
{% block extra_css %}
  <link rel="stylesheet" href="https://unpkg.com/leaflet/dist/leaflet.css" />
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/leaflet.draw/1.0.4/leaflet.draw.css
" />
  <style>
```

```
    #map { height: 500px; }
  </style>
{% endblock %}
{% block content %}
<h2>Інтерактивна карта</h2>
<p class="lead">
  <button id="updateFromSite" class="btn btn-info">Оновити дані з
Overpass</button>
  <button id="saveToDB" class="btn btn-success">Зберегти дані у БД</button>
  <button id="loadFromDB" class="btn btn-primary">Відобразити станції з
БД</button>
  <button id="clearMarkers" class="btn btn-secondary">Очистити маркери</button>
</p>
<div id="map"></div>
<p class="mt-3">
  <button id="optimizeZone" class="btn btn-warning">Оптимізувати обрану
зону</button>
</p>
<div id="mapResult" class="mt-3"></div>
{% endblock %}
{% block extra_js %}
  <script src="https://unpkg.com/leaflet/dist/leaflet.js"></script>
  <script
src="https://cdnjs.cloudflare.com/ajax/libs/leaflet.draw/1.0.4/leaflet.draw.js">
</script>
  <script>
    var map = L.map('map').setView([50, 50], 4);
    L.tileLayer('https://{s}.tile.openstreetmap.org/{z}/{x}/{y}.png', {
      attribution: '© OpenStreetMap contributors'
    }).addTo(map);

    var markers = [];
    var drawnItems = new L.FeatureGroup();
    map.addLayer(drawnItems);

    var drawControl = new L.Control.Draw({
      draw: {
        polygon: false,
        polyline: false,
        circle: false,
        marker: false,
        circlemarker: false,
        rectangle: true
      },
      edit: {
        featureGroup: drawnItems,
        remove: true
      }
    });
    map.addControl(drawControl);

    map.on(L.Draw.Event.CREATED, function (event) {
      var layer = event.layer;
      drawnItems.addLayer(layer);
    });

    function addMarker(lat, lng) {
      var marker = L.marker([lat, lng], {draggable: true}).addTo(map)
                    .bindPopup("Базова станція");
      markers.push(marker);
    }

    function clearMarkers() {
      markers.forEach(function(m) { map.removeLayer(m); });
```

```
      markers = [];
    }

    $('#clearMarkers').click(function(){
      clearMarkers();
    });

    $('#updateFromSite').click(function(){
      $.ajax({
        url: '/fetch_real_data',
        type: 'GET',
        dataType: 'json',
        success: function(response){
          if(response.status === "success"){
            clearMarkers();
            response.data.forEach(function(pos) {
              addMarker(pos.lat, pos.lng);
            });
            $('#mapResult').html('<div class="alert alert-success">Оновлено дані
з Overpass: ' + response.data.length + ' об’єктів.</div>');
          } else {
            $('#mapResult').html('<div class="alert alert-danger">Помилка: ' +
response.message + '</div>');
          }
        },
        error: function(){
          $('#mapResult').html('<div class="alert alert-danger">Помилка при
отриманні даних.</div>');
        }
      });
    });

    $('#saveToDB').click(function(){
      $.ajax({
        url: '/save_stations',
        type: 'GET',
        dataType: 'json',
        success: function(response){
          if(response.status === "success"){
            $('#mapResult').html('<div class="alert alert-success">' +
response.message + '</div>');
          } else {
            $('#mapResult').html('<div class="alert alert-danger">' +
response.message + '</div>');
          }
        },
        error: function(){
          $('#mapResult').html('<div class="alert alert-danger">Помилка при
збереженні даних у БД.</div>');
        }
      });
    });

    $('#loadFromDB').click(function(){
      if(drawnItems.getLayers().length === 0){
        $('#mapResult').html('<div class="alert alert-warning">Спочатку виберіть
зону для завантаження (намалюйте прямокутник).</div>');
        return;
      }
      var zone = drawnItems.getLayers()[0].getBounds();
      var params = {
        south: zone.getSouth(),
        west: zone.getWest(),
        north: zone.getNorth(),
```

```
        east: zone.getEast()
      };
      $.ajax({
        url: '/get_stations',
        type: 'GET',
        data: params,
        dataType: 'json',
        success: function(response){
          if(response.status === "success"){
            clearMarkers();
            response.data.forEach(function(pos) {
              addMarker(pos.lat, pos.lng);
            });
            $('#mapResult').html('<div class="alert alert-success">Завантажено '
+ response.data.length + ' станцій з БД.</div>');
          } else {
            $('#mapResult').html('<div class="alert alert-danger">' +
response.message + '</div>');
          }
        },
        error: function(){
          $('#mapResult').html('<div class="alert alert-danger">Помилка при
завантаженні даних з БД.</div>');
        }
      });
    });

    function getMarkersInZone(bounds) {
      return markers.filter(function(marker){
        return bounds.contains(marker.getLatLng());
      }).map(function(marker){
        var latlng = marker.getLatLng();
        return {lat: latlng.lat, lng: latlng.lng};
      });
    }

    $('#optimizeZone').click(function(){
      if(drawnItems.getLayers().length === 0){
        $('#mapResult').html('<div class="alert alert-warning">Спочатку виберіть
зону для оптимізації (намалюйте прямокутник).</div>');
        return;
      }
      var zone = drawnItems.getLayers()[0].getBounds();
      var zoneMarkers = getMarkersInZone(zone);
      if(zoneMarkers.length < 2){
        $('#mapResult').html('<div class="alert alert-warning">У зоні
недостатньо маркерів для оптимізації.</div>');
        return;
      }
      $.ajax({
        url: '/map',
        type: 'POST',
        contentType: 'application/json',
        data: JSON.stringify({markers: zoneMarkers}),
        success: function(data) {
          $('#mapResult').html('<div class="alert alert-success">Оптимізація
виконана для обраної зони.</div>');
          var optimized = data.optimized_positions;
          var count = 0;
          markers.forEach(function(marker){
            if(zone.contains(marker.getLatLng()) && count < optimized.length){
              marker.setLatLng([optimized[count].lat, optimized[count].lng]);
              count++;
            }
```

```
          });
        },
        error: function() {
          $('#mapResult').html('<div class="alert alert-danger">Помилка
оптимізації позицій.</div>');
        }
      });
    });
  </script>
{% endblock %}
```

Optimize.html

```
{% extends "base.html" %}
{% block content %}
<h2>Запуск оптимізації</h2>
<p class="lead">Оберіть алгоритм і запустіть оптимізацію розміщення базових
станцій.</p>
<form action="{{ url_for('optimize') }}" method="post">
  <div class="form-group">
    <label for="algorithm">Алгоритм оптимізації:</label>
    <select class="form-control" id="algorithm" name="algorithm">
      <option value="PSO">PSO (Particle Swarm Optimization)</option>
      <option value="NSGA-II">NSGA-II</option>
    </select>
  </div>
  <button type="submit" class="btn btn-primary">Запустити</button>
</form>
<hr>
<h3>Або запустіть оптимізацію через API</h3>
<button id="runApi" class="btn btn-info">Запустити API оптимізацію</button>
<div class="mt-3" id="apiResult"></div>
{% endblock %}
{% block extra_js %}
<script>
  $('#runApi').click(function(){
    $.ajax({
      url: '/api/optimize',
      type: 'POST',
      contentType: 'application/json',
      data: JSON.stringify({algorithm: $('#algorithm').val()}),
      success: function(result){
        $('#apiResult').html('<div class="alert alert-success">Найкраща позиція:
'
          + result.best_position + ', Значення: ' + result.score + '<br><small>'
+ result.details + '</small></div>');
      },
      error: function(){
        $('#apiResult').html('<div class="alert alert-danger">Помилка виконання
оптимізації.</div>');
      }
    });
  });
</script>
{% endblock %}
```

Register.html

```
{% extends "base.html" %}
{% block content %}
<h2>Реєстрація</h2>
```

```
<form method="post" action="{{ url_for('register_view') }}">
  <div class="form-group">
    <label for="username">Ім'я користувача</label>
    <input type="text" name="username" class="form-control" id="username"
required>
  </div>
  <div class="form-group">
    <label for="password">Пароль</label>
    <input type="password" name="password" class="form-control" id="password"
required>
  </div>
  <button type="submit" class="btn btn-primary">Зареєструватись</button>
</form>
<p class="mt-3">Вже маєте акаунт? <a href="{{ url_for('login_view') }}">Увійдіть
тут</a>.</p>
{% endblock %}
```

Silumation.html

```
{% extends "base.html" %}
{% block content %}
<h2>Симуляція покриття мережі</h2>
<p class="lead">Введіть параметри для розрахунку покриття транспортного
маршруту.</p>
<form method="post" action="{{ url_for('simulation') }}">
  <div class="form-group">
    <label for="route_length">Довжина маршруту (км)</label>
    <input type="number" step="0.1" name="route_length" class="form-control"
id="route_length" required>
  </div>
  <div class="form-group">
    <label for="base_station_range">Діапазон покриття базової станції
(км)</label>
    <input type="number" step="0.1" name="base_station_range" class="form-
control" id="base_station_range" required>
  </div>
  <button type="submit" class="btn btn-primary">Запустити симуляцію</button>
</form>

{% if result %}
<hr>
<h3>Результати симуляції</h3>
<ul class="list-group">
  <li class="list-group-item"><strong>Кількість базових станцій:</strong> {{
result.num_stations }}</li>
  <li class="list-group-item"><strong>Покриття:</strong> {{
result.coverage_percentage }}%</li>
  <li class="list-group-item"><strong>Деталі:</strong> {{ result.details }}</li>
</ul>
{% endif %}
{% endblock %}
```

Test.py

```
import overpy

api = overpy.Overpass()



query = f"""
```

```
[out:json][timeout:25];
// Отримуємо всі базові станції зв'язку
(
  node["man_made"="communications_tower"];
  way["man_made"="communications_tower"];
  relation["man_made"="communications_tower"];
);
out body;
>;
out skel qt;

"""

print("Виконується запит до Overpass API...")
result = api.query(query)

print("Отримані координати базових станцій:")
for node in result.nodes:
    print(f"Latitude: {node.lat}, Longitude: {node.lon}")
```