МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ НЕЙРОННОЇ МЕРЕЖІ В ЛОГІСТИЧНІЙ ГРІ З НУЛЬОВОЮ СУМОЮ»**

Виконав: студент 2 курсу, групи 8.1224-онп
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
Н. А. Долженко
(ініціали та прізвище)
Керівник старший викладач кафедри комп'ютерних наук, к.т.н. Добровольський Г.А.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент професор кафедри економічної кібернетики, професор, д.ф.-м.н., Козін І.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти магістр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., професор

_____ Шило Г.М.
(підпис)

" ____ " _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Долженко Нікіті Андрійовичу
(прізвище, ім'я та по батькові)

1. Тема роботи Дослідження ефективності нейронної мережі в логістичній грі з нульовою сумою

керівник роботи Добровольський Геннадій Анатолійович, к.т.н.
(прізвище, ім'я та по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від « 28 » лютого 2025 року № 326-с

2. Строк подання студентом роботи 06.06.2026

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Дослідження ефективності нейронної мережі в логістичній грі з нульовою сумою
4. Реалізація програми, аналіз результатів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 06.03.2025

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи	17.03.2025	
2.	Збір вихідних даних	20.06.2025	
3.	Обробка методичних та теоретичних	01.11.2025	
	джерел		
4.	Розробка першого та другого розділу	27.02.2026	
5.	Розробка третього та четвертого розділу	20.04.2026	
6.	Оформлення та нормоконтроль	06.06.2026	
	кваліфікаційної роботи магістра		
7.	Захист кваліфікаційної роботи	20.06.2026	

Студент _____ Н. А. Долженко
(підпис) (ініціали та прізвище)

Керівник роботи _____ Г.А. Добровольський
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г. Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра «Дослідження ефективності нейронної мережі в логістичній грі з нульовою сумою»: 84 с., 16 рис., 7 табл., 46 джерел, 2 додатки.

ГРАФОВА НЕЙРОННА МЕРЕЖА, ЕФЕКТИВНІСТЬ, ЗАДАЧА МЕРЕЖЕВОГО ПЕРЕХОПЛЕННЯ, ІМІТАЦІЙНЕ НАВЧАННЯ, МКА, ОПТИМІЗАЦІЯ, GCN, MLP.

Об’єкт дослідження: процес застосування нейромережевих технологій для розв’язання логістичної гри з нульовою сумою на графах.

Предмет дослідження: архітектури графових нейронних мереж (зокрема, ГМКА та EdgeMLP) та алгоритми імітаційного навчання для задачі мережевого перехоплення.

Мета роботи: дослідження ефективності нейронної мережі ГМКА у логістичній грі з нульовою сумою на графах, а також експериментальне визначення її оптимальних гіперпараметрів порівняно з класичними архітектурами.

Методи дослідження: математичне моделювання, імітаційне навчання, обчислювальний експеримент, порівняльний аналіз.

Кваліфікаційна робота містить аналітичний огляд задач мережевого перехоплення та сучасних методів машинного навчання на графах, зокрема архітектури мереж Колмогорова–Арнольда (МКА).

Практична частина присвячена розробці та тестуванню моделі ГМКА (Графова мережа Колмогорова–Арнольда). Модель ефективно апроксимує стратегії інтердикції, досягаючи на регулярних топологіях паритету якості з базовим EdgeMLP при п'ятикратному зменшенні кількості параметрів модуля виведення. Модель гарантує математичну інтерпретованість рішень і може слугувати дієвим інструментом оптимізації та захисту логістичних мереж.

SUMMARY

Master's Qualifying Thesis «Study of the effectiveness of a neural network in a zero-sum logistics game»: 84 pages, 16 figures, 7 tables, 46 references, 2 supplements.

GRAPH NEURAL NETWORK, EFFICIENCY, NETWORK INTERDICTION PROBLEM, IMITATION LEARNING, KAN, OPTIMIZATION, GCN, MLP.

Object of the study: the process of applying neural network technologies to solve a zero-sum logistics game on graphs.

Subject of research: graph neural network architectures (specifically, GKAN and EdgeMLP) and imitation learning algorithms for the network interdiction problem.

Aim of the study: to investigate the effectiveness of the GKAN neural network in a zero-sum logistics game on graphs and to experimentally determine its optimal hyperparameters compared to classical architectures.

Methods of research: mathematical modeling, imitation learning, computational experiment, comparative analysis.

The master's qualifying thesis contains an analytical overview of network interdiction problems and modern machine learning methods on graphs, particularly the Kolmogorov-Arnold Network (KAN) architecture.

The practical part is devoted to the development and testing of the GKAN (Graph Kolmogorov-Arnold Network) model. It effectively approximates interdiction strategies, achieving quality parity with the baseline EdgeMLP on regular topologies while reducing the number of parameters in the readout module by five times. The model guarantees mathematical interpretability of decisions and can serve as an effective tool for optimizing and protecting logistics networks.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 5
Вступ 9
1 Аналітичний огляд методів мережевого перехоплення 11
1.1 Постановка задачі мережевого перехоплення та її класифікація 11
1.1.1 Класифікація за невизначеністю та стратегіями 11
1.2 Класичні MILP-моделі для задачі перехоплення найкоротшого шляху 13
1.2.1 Формування цільової функції інтердиктора 14
1.2.2 Застосування лінійної двоїстості 15
1.3 Евристичні та метаевристичні підходи до мережевого перехоплення 19
1.3.1 Жадібний алгоритм перехоплення 19
1.3.2 Локальний пошук та метаевристики 20
1.4 Застосування методів машинного навчання та графових нейронних мереж до задач комбінаторної оптимізації 22
1.4.1 Наскрізні моделі та генерація рішень 22
1.4.2 Навчання з підкріпленням у комбінаторній оптимізації 23
1.4.3 Інтеграція GNN у класичні алгоритми 23
1.4.4 Імітаційне навчання проти RL 24
1.5 Нейромережеві підходи до задач мережевого перехоплення 26
1.5.1 RL-методи перехоплення 26
1.5.2 Supervised GNN-сурогати 27
1.6 Архітектура мереж Колмогорова–Арнольда 28
1.6.1 Теоретичне підґрунтя 29
1.6.2 Сучасна МКА-архітектура та RBF-базиси 29
1.6.3 Обґрунтування вибору архітектури МКА для задач перехоплення 33
2 Теоретичні основи та формалізація задачі 34

2.1 Формальна модель задачі мережевого перехоплення найкоротшого шляху 34
2.1.1 Умова нульової суми. 35
2.1.2 Stackelberg-гра проти Nash-рівноваги 35
2.1.3 Зведення до однорівневого MILP 36
2.1.4 Стохастичний варіант 36
2.2 Параметри бюджету B та інтенсивності α: чутливість і сценарії застосування 37
2.2.1 Вплив бюджету B на цільову функцію 39
2.2.2 Чутливість моделі до інтенсивності атаки α 39
2.2.3 Сценарії застосування SPI 40
2.3 Перехід до задачі навчання з учителем 42
2.3.1 Формування датасету та жадібний оракул 42
2.3.2 Проблема зміщення розподілу (Distribution Shift) 43
2.4 Теорема Колмогорова–Арнольда та її застосування до МКА 45
2.4.1 Класична теорема 45
2.4.2 Перехід до багатошарової архітектури 46
2.4.3 Порівняльні переваги МКА над MLP 46
2.4.4 Синтез теоретичної моделі та обґрунтування ГМКА 48
3 Архітектура моделі ГМКА для задачі мережевого перехоплення 51
3.1 Загальна постановка та архітектура розв’язання 51
3.1.1 Принципові вимоги до архітектури 51
3.1.2 Вхідні дані та обчислювальний потік 52
3.2 GCN-кодувальник 54
3.3 EdgeFeaturizer 55
3.4 Вихідний шар Колмогорова–Арнольда: RBF1D та лінійний комбінатор 56
3.5 Механізми регуляризації МКА 57
3.6 Функція втрат і штрафи 58
3.7 Інтерпретованість вихідного модуля МКА 59
3.8 Базова модель порівняння: EdgeMLP 61
3.9 Підготовка даних і модельні графи 61

3.10 Повний потік даних та цикл навчання 63
3.10.1 Підсумкові характеристики обчислювального процесу 63
4 Експериментальне дослідження ефективності нейромережі 65
4.1 Опис обчислювального середовища і метрик 65
4.1.1 Обчислювальне середовище 65
4.1.2 Метрики оцінки якості моделі 65
4.2 Базове порівняння: вплив гіперпараметрів МКА 66
4.2.1 Вплив кількості базисних функцій RBF (K) 66
4.2.2 Механізми регуляризації 67
4.2.3 Вплив Dropout та нормалізації вхідних ознак 68
4.2.4 Вибір режиму функції втрат (Loss Mode) 69
4.3 Аналіз ефективності в різних топологіях логістичних графів 70
4.3.1 Результати на регулярних сітках (Grid) 70
4.3.2 Результати на складних макро-логістичних графах (BA, SiouxFalls) 72
4.3.3 Загальні підсумки тестування 72
Висновки 74
Перелік посилань 76
Додаток А Програмна реалізація класу RBF1D та KANREADOUT 82
Додаток Б Реалізація механізмів штрафу та регуляризації МКА 84

ВСТУП

У рамках цієї роботи було проведено дослідження у галузі машинного навчання на графах, зокрема, вивчення ефективності та можливостей передових нейромережевих архітектур для розв’язання логістичної гри з нульовою сумою [43, 44, 45]. Задача мережевого перехоплення моделює стратегічну протидію двох сторін: атакуючого фактора (природних перешкод, заторів або зловмисника), що прагне максимально порушити логістичні маршрути, та транспортного оператора, який шукає оптимальний шлях доставки постачань в умовах пошкодженої інфраструктури. Ця задача є ідеальним середовищем для дослідження нейромереж, оскільки вона має чітко визначену математичну структуру гри з нульовою сумою, вкрай важливі застосування у транспортній логістиці, плануванні ланцюгів постачань, а також дозволяє кількісно перевіряти ефективність навченої моделі у порівнянні з точним математичним розв’язувачем.

Актуальність цього дослідження полягає в тому, що, попри значний прогрес машинного навчання у комбінаторній оптимізації, питання ефективності нових нейромережевих архітектур для розв’язання логістичних ігор залишається практично нерозкритим. Класичні методи на основі мішано-цілочисельного лінійного програмування (MILP, наприклад через Gurobi [39]) дають точні результати, проте їхня обчислювальна складність експоненційно зростає із розміром транспортної мережі, що унеможливлює швидку маршрутизацію. Нейромережеві підходи здатні знаходити апроксимовані рішення за мілісекунди, але їхня ефективність сильно залежить від архітектури. Особливий інтерес становить застосування архітектури мережі Колмогорова–Арнольда (МКА): на відміну від класичних багатошарових перцептронів (MLP), де нелінійності фіксовані у вузлах, МКА навчає параметризовані функції безпосередньо на з’єднаннях мережі, що дає значну економію параметрів.

Метою цієї роботи є дослідження ефективності нейронної мережі ГМКА (Графова мережа Колмогорова–Арнольда) у логістичній грі з нульовою сумою

на графах [36], а також експериментальний підбір її оптимальних гіперпараметрів. Було розроблено архітектуру, що поєднує графову згорткову мережу (GCN) [29, 37, 38] для фіксації топології доріг та вихідного модуля МКА для класифікації ребер. Логістичну гру зведено до задачі імітаційного навчання, де жадібний алгоритм виступає вчителем оптимального блокування, а модель навчається передбачати найбільш критичні вузькі місця.

Для перевірки ефективності проведено чотири масштабні дослідження із аналізу внеску компонентів: аналіз кількості базисних функцій, типу регуляризації, впливу методу розрідження (dropout) і вибору функції втрат. Окрім того, виконано порівняльний аналіз швидкодії та точності моделі ГМКА з алгоритмами Random, Greedy та EdgeMLP на графах трьох різних топологій: регулярній сітці (Grid), графах Барабаші–Альберт та реальній транспортно-логістичній мережі міста Су-Фолс (SiouxFalls).

За результатами дослідження підтверджено високу ефективність МКА: оптимальна конфігурація включає 8 базисних функцій із L2-регуляризацією без dropout. Завдяки застосованому авторами асиметричному штрафу у функції втрат, модель досягає стабільного та швидкого навчання. Зокрема, на регулярній топології ГМКА продемонстрував паритет обчислювальної якості з EdgeMLP, маючи при цьому значно меншу кількість параметрів фінального шару та забезпечуючи високу швидкодію порівняно з жадібними і точними (MILP) алгоритмами.

Це дослідження поглиблює розуміння можливостей використання сучасних нейронних мереж в іграх із нульовою сумою, підтверджує обчислювальну ефективність МКА-шарів для аналізу графів і закладає фундамент для розробки нових інструментів захисту й оптимізації логістичних та транспортних мереж.

1 АНАЛІТИЧНИЙ ОГЛЯД МЕТОДІВ МЕРЕЖЕВОГО ПЕРЕХОПЛЕННЯ

1.1 Постановка задачі мережевого перехоплення та її класифікація

Задача мережевого перехоплення (network interdiction) належить до класу дворівневих задач оптимізації і моделює стратегічну взаємодію двох сторін: інтердиктора (який виступає математичною моделлю атакуючого фактора — зловмисника, природних перешкод або заторів) та оператора (захисника або транспортного логіста). Інтердиктор діє першим і в межах наявного бюджету (ресурсу руйнування чи впливу) модифікує ребра або вузли мережі таким чином, щоб максимально погіршити цільову функцію оператора. Оператор, знаючи про наявні перешкоди чи пошкодження, розв’язує власну задачу оптимізації на вже зміненому графі. Взаємодія двох гравців утворює гру Штакельберга, де інтердиктор (атакуючий фактор) є лідером, а оператор слідує його рішенню [1].

Залежно від об’єкта впливу задачі мережевого перехоплення поділяють на декілька основних класів. У задачі перехоплення найкоротшого шляху (Shortest Path Interdiction, SPI) оператор прагне знайти найкоротший шлях між двома вузлами мережі, а інтердиктор намагається максимально збільшити довжину цього шляху шляхом видалення або обтяження ребер [2]. У задачі перехоплення максимального потоку (Maximum Flow Interdiction, MFI) мета оператора полягає у максимізації потоку між джерелом і стоком, тоді як інтердиктор прагне цей потік мінімізувати [3]. Окремим підкласом є задачі перехоплення на основі критичних вузлів (vital node interdiction), де впливу (атаці) піддаються не ребра, а вершини графа.

1.1.1 Класифікація за невизначеністю та стратегіями

За характером невизначеності задачі поділяють на детерміновані та стохастичні. У детермінованих варіантах усі параметри мережі (ваги ребер, вартість атаки) вважаються точно відомими. Стохастичні задачі враховують

невизначеність: наприклад, ймовірність успішного блокування ребра або невідому вагу ребра після атаки [4]. Стохастичні формулювання суттєво складніші обчислювально, тому більшість класичних і нейромережевих методів зосереджена на детермінованому варіанті.

Додатковий вимір класифікації пов’язаний із типом стратегій гравців. У задачах із чистими стратегіями інтердиктор обирає фіксований набір ребер для атаки. У задачах зі змішаними стратегіями обидва гравці можуть рандомізувати свої дії, що призводить до формулювання у вигляді задачі знаходження рівноваги Неша у грі з нульовою сумою [5]. Задачі зі змішаними стратегіями знаходять застосування у моделюванні систем безпеки, де зловмисник може адаптувати свою поведінку.

Ще одна важлива характеристика задач мережевого перехоплення пов’язана із бюджетним обмеженням інтердиктора. Найпоширенішим є бюджет у кардинальному вигляді: інтердиктор може атакувати не більше ніж B ребер. В узагальненому варіанті кожне ребро має індивідуальну вартість атаки, а загальна витрата не може перевищувати заданого бюджету. Параметр інтенсивності атаки α визначає, наскільки зростає вага ребра після перехоплення: при $\alpha = 0$ атака не має ефекту, при $\alpha \to \infty$ ребро фактично блокується. Це дозволяє моделювати як часткове, так і повне порушення функціонування ребра.

Сучасні огляди [1, 6] виділяють три основні покоління методів розв’язання задач мережевого перехоплення: класичні MILP-підходи (1990-ті роки), евристичні та метаевристичні алгоритми (2000-ні роки) та нейромережеві методи (2020-ті роки). Класичні MILP-підходи забезпечують оптимальне рішення ціною високої обчислювальної складності; евристики досягають прийнятної якості за поліноміальний час; нейромережеві методи навчаються апроксимувати рішення і забезпечують миттєве виведення результатів.

На рис. 1.1 показано загальну таксономію задач мережевого перехоплення.

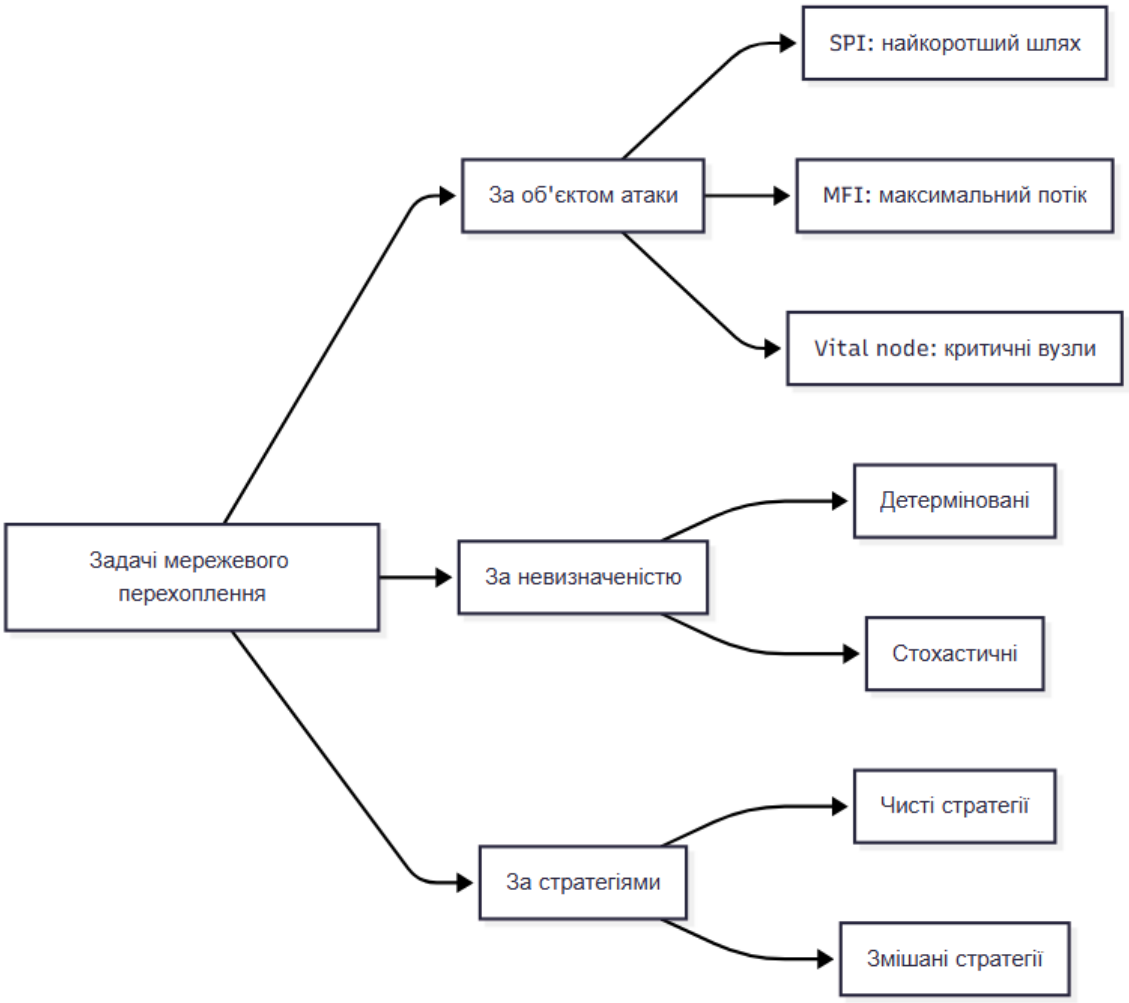


Рисунок 1.1 – Таксономія задач мережевого перехоплення

1.2 Класичні MILP-моделі для задачі перехоплення найкоротшого шляху

Задача перехоплення найкоротшого шляху у своїй детермінованій постановці є двохрівневою задачею мішано-цілочисельного програмування. Перший рівень описує дії інтердиктора: вибір підмножини ребер для атаки за умови обмеженого бюджету. Другий рівень описує відповідь оператора: побудову найкоротшого шляху у графі з вже модифікованими вагами. Принципова складність такої структури полягає у тому, що задача оператора є параметричною відносно рішення інтердиктора, тобто вкладена всередину оптимізаційної задачі верхнього рівня.

1.2.1 Формування цільової функції інтердиктора

Формальна постановка SPI у найзагальнішому вигляді виглядає так. Нехай задано орієнтований граф $G = (V, E)$ із базовими вагами ребер $c_e > 0$ для кожного $e \in E$, пара вузлів джерело–стік (s, t), бюджет B та штрафний коефіцієнт $\alpha > 1$. Інтердиктор обирає бінарний вектор $y \in \{0,1\}^{|E|}$, де $y_e = 1$ означає атаку на ребро e. Після атаки вага ребра стає:

$$w_e(y) = c_e(1 + (\alpha - 1)\, y_e), \tag{1.1}$$

де w_e – вага ребра після атаки;

c_e – базова вага ребра;

α – коефіцієнт збільшення вартості атакованого ребра;

y_e – бінарна змінна (1, якщо ребро атаковане, 0 інакше).

За такого зваження атакованого ребра оператор розв’язує задачу найкоротшого шляху у модифікованому графі. Дворівнева задача в цілому набуває вигляду:

$$\max_{y \in \{0,1\}^{|E|},\ \sum_e y_e \le B} \ \min_{P \in \mathcal{P}(s,t)} \sum_{e \in P} w_e\,(y), \tag{1.2}$$

де $\mathcal{P}(s, t)$ – множина всіх простих шляхів від s до t у графі G.

Безпосереднє розв’язання цієї задачі є NP-важким у загальному випадку [2], оскільки кількість можливих вкладених задач найкоротшого шляху є комбінаторно великою. Класичний підхід до її розв’язання, запропонований Israeli і Wood [2], полягає у застосуванні теорії двоїстості лінійного програмування для перетворення двохрівневої задачі на однорівневий MILP.

1.2.2 Застосування лінійної двоїстості

Ключове спостереження, що уможливлює таке перетворення, таке: внутрішня задача оператора є задачею найкоротшого шляху, яка при фіксованих вагах $w_e(y)$ є лінійною програмою. Відповідно до LP-двоїстості, задача мінімального шляху еквівалентна задачі максимізації різниці потенціалів вузлів. Введемо двоїсті змінні π_v для кожного вузла $v \in V$, що інтерпретуються як потенціали, де $\pi_s = 0$ і π_t є цільовою функцією оператора. Умови двоїстості для задачі найкоротшого шляху записуються як:

$$\pi_v - \pi_u \leq w_e(y) = c_e + (\alpha - 1)\, c_e\, y_e \quad \forall\, (u, v) \in E \tag{1.3}$$

де π_u, π_v – потенціали вузлів u і v відповідно;

(u, v) – орієнтоване ребро з хвоста u у голову v.

Задача інтердиктора після підстановки двоїстих умов набуває вигляду однорівневого MILP:

$$\max \ \pi_t \tag{1.4}$$

за умов:

$$\pi_v - \pi_u \leq c_e + (\alpha - 1)\, c_e\ y_e \quad \forall\, (u, v) \in E, \tag{1.5}$$

$$\sum_{e \in E} y_e \leq B, \quad y_e \in \{0,1\}, \quad \pi_s = 0. \tag{1.6}$$

Отримана задача є змішано-цілочисельною лінійною програмою: неперервні змінні π_v описують потенціали вузлів (двоїсті змінні оператора), а бінарні y_e описують рішення інтердиктора. Таке зведення є принципово можливим завдяки лінійній структурі задачі внутрішнього рівня. Якщо б оператор розв’язував нелінійну або цілочисельну задачу, пряма двоїстість не застосовувалась би.

Практично важливою є також форма обмеження з урахуванням добутку $y_e \cdot \pi$. Оскільки y_e є бінарним, добуток $c_e(\alpha - 1)y_e$ вже є лінійним за y_e і не потребує додаткової лінеаризації. Це є суттєвою перевагою задачі SPI над рядом інших двохрівневих задач, де лінеаризація добутку бінарних та неперервних змінних вимагає допоміжних змінних і розширює простір задачі.

Обчислювальна складність отриманого MILP зростає зі збільшенням розміру графа. Кількість змінних дорівнює $|E| + |V|$, а кількість обмежень дорівнює $|E| + 1$. Для розв'язання задач такого класу використовуються стандартні алгоритми оптимізації: гілки та межі (branch-and-bound) [1, 43], гілки та відсічення (branch-and-cut) [1, 43], а також розкладання Бендерса [2, 3]. Сучасні комерційні розв'язувачі, зокрема Gurobi та CPLEX, а також відкриті системи, зокрема OR-Tools від Google, реалізують ці алгоритми з оптимізацією евристик відсічення і стратегій розгалуження. В нашому випадку, для отримання еталонних MILP-рішень використовується OR-Tools 9.12 [7].

Варто зазначити, що в ряді робіт [1, 8, 9] застосовується більш загальне формулювання задачі інтердикції з індивідуальними вартостями атаки ребер. У такій постановці кожне ребро має власну вартість d_e, а бюджетне обмеження набуває вигляду $\sum_e d_e y_e \leq B$. Дослідження Fischetti et al. [8] розглядають монотонні ігри перехоплення як загальний клас, до якого SPI є окремим випадком, і показують, що для монотонних ігор завжди існує чиста оптимальна стратегія інтердиктора.

Для ілюстрації масштабу задачі та відмінностей між різними формулюваннями у табл. 1.1 наведено порівняння ключових MILP-підходів до SPI.

Незважаючи на елегантність LP-двоїстісного зведення, класичні MILP-підходи мають суттєве практичне обмеження: час розв'язання зростає поліноміально або експоненційно зі зростанням $|V|$ і $|E|$ при несприятливих структурах графа. Незважаючи на те, що для сіткових графів помірної розмірності (8×8, тобто $|E| \approx 112$) обчислювальна складність MILP-алгоритмів залишається керованою, із подальшим зростанням топології час знаходження розв'язку збільшується експоненціально. Для реалістичних

транспортних мереж із кількістю ребер $|E| > 10^4$ точне розв'язання задачі стає обчислювально недоступним, що й зумовлює актуальність переходу до нейромережевих сурогатів. [1, 6]. Саме ця обчислювальна бар’єрність MILP-формулювань зумовлює інтерес до наближених евристичних і нейромережевих методів.

Таблиця 1.1 – Порівняння MILP-формулювань для задачі перехоплення найкоротшого шляху

Автор(и)	Рік	Тип формулювання	Змінних	Обмежень	Примітки
Israeli, Wood	2002	Bilevel + LP-duality	$\|E\| + \|V\|$	$\|E\| + 1$	Базове зведення через двоїстість
Автор(и)	Рік	Тип формулювання	Змінних	Обмежень	Примітки
Fischetti et al.	2019	Monotone interdiction game	$\|E\| + \|V\|$	$\|E\| + 1$	Узагальнення на довільні монотонні ігри
Wood	1993	MFI + LP-duality	$\|E\| + \|V\|$	$\|E\| + \|V\|$	Аналог для задачі максимального потоку

На рис. 1.2 показано схему перетворення двохрівневої задачі SPI на однорівневий MILP через LP-двоїстість.

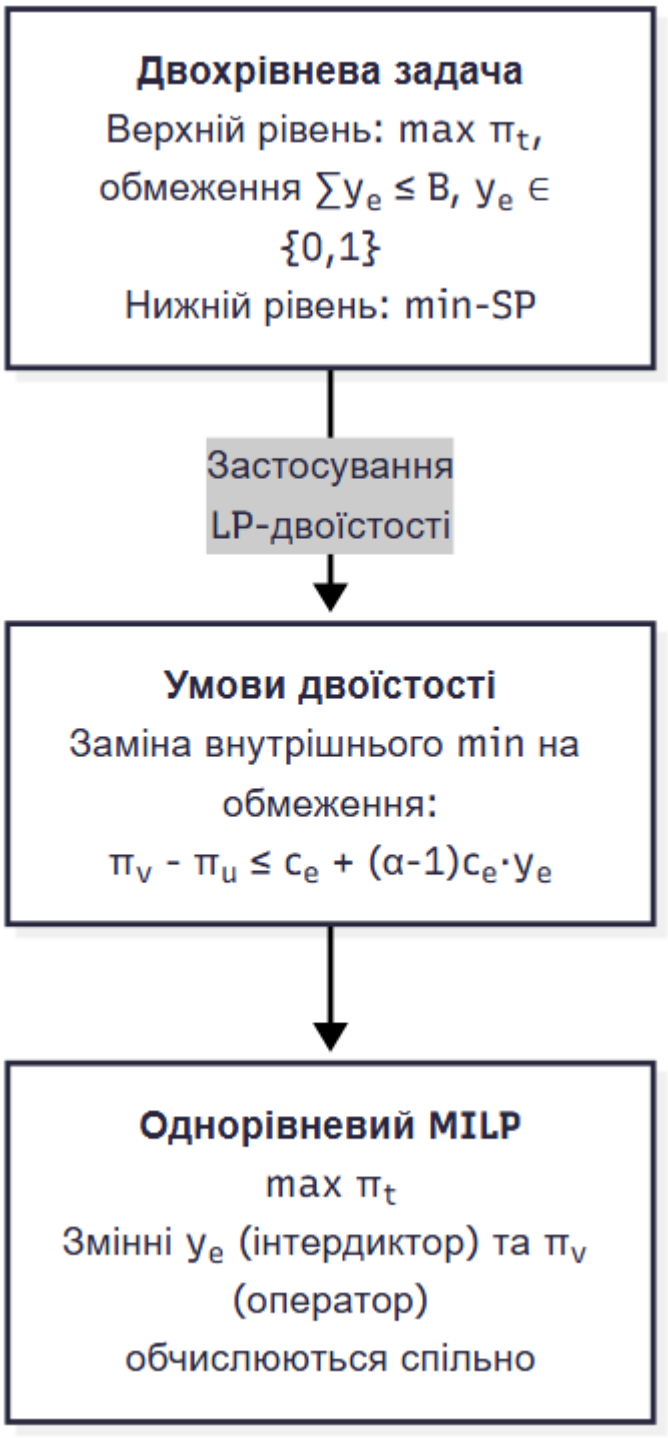


Рисунок 1.2 – Перетворення двохрівневої задачі SPI у однорівневий MILP через LP-двоїстість

1.3 Евристичні та метаевристичні підходи до мережевого перехоплення

Обчислювальні обмеження MILP-розв'язувачів зумовили активний розвиток евристичних методів, що поступаються точністю оптимального рішення на користь швидкості та масштабованості. Принципова відмінність евристик від точних методів полягає у відмові від гарантії оптимальності: замість пошуку найкращого можливого рішення вони шукають достатньо якісне рішення за прийнятний час. Для задач мережевого перехоплення такий компроміс є виправданим у прикладних сценаріях, де рішення потрібно приймати в режимі реального часу або на графах розміру, недоступного для MILP.

1.3.1 Жадібний алгоритм перехоплення

Найпростішим і водночас практично потужним евристичним підходом до SPI є жадібний алгоритм. Він діє ітеративно: на кожному кроці з числа ще не атакованих ребер вибирається те, видалення якого дає найбільший приріст вартості найкоротшого шляху оператора. Формально, на кроці k із множини доступних ребер $E \setminus S_{k-1}$ обирається:

$$e^* = \arg\max_{e \in E \setminus S_{k-1}} [C(S_{k-1} \cup \{e\}) - C(S_{k-1})], \tag{1.7}$$

де S_{k-1} – множина вже атакованих ребер після $k-1$ кроків;

$C(S)$ – вартість найкоротшого шляху у графі після атаки множини S.

Після B ітерацій алгоритм повертає множину $S^* = S_B$ як рішення. Обчислювальна складність одного кроку дорівнює $O(|E| \cdot T_{SP})$, де T_{SP} – час розв'язання задачі найкоротшого шляху (для алгоритму Дейкстри $T_{SP} = O((|V| + |E|)\log|V|)$). Загальна складність алгоритму становить $O(B \cdot |E| \cdot (|V| + |E|)\log|V|)$, що є поліноміальним, але зростає лінійно з бюджетом B і

квадратично з розміром графа. Саме жадібний алгоритм відіграє роль вчителя у запропонованій архітектурі ГМКА: він генерує навчальні мітки для задачі імітаційного навчання.

Попри відсутність гарантій оптимальності, жадібний алгоритм демонструє на практиці якість, близьку до оптимальної на регулярних топологіях. Lim і Smith [9] показали, що для задач перехоплення мультикомодитних потоків жадібний підхід дає рішення у межах кількох відсотків від MILP-оптимуму на більшості практичних тестових задач.

1.3.2 Локальний пошук та метаевристики

Для покращення якості жадібного рішення широко застосовуються методи локального пошуку. Базова ідея полягає у пошуку «покращуючого обміну»: маючи поточну множину S, алгоритм перевіряє всі можливі заміни одного атакованого ребра на неатаковане і приймає обмін, якщо це збільшує $C(S)$. Цей процес повторюється до досягнення локального оптимуму за кількістю обмінів. Один крок локального пошуку має складність $O(|S| \cdot (|E| - |S|) \cdot T_{SP})$, що є прийнятним для графів середнього розміру.

Серед метаевристик для задач мережевого перехоплення застосовуються імітація відпалу (simulated annealing) та заборонений пошук (tabu search) [1, 10]. Імітація відпалу дозволяє виходити з локальних оптимумів за рахунок імовірнісного прийняття погіршуючих обмінів, що зменшується з часом згідно з розкладом охолодження. Заборонений пошук підтримує список нещодавно розглянутих рішень і забороняє повторне їх відвідування, примушуючи алгоритм досліджувати різні частини простору рішень. Порівняльний аналіз цих підходів для задач перехоплення потоків та шляхів наведено у роботі Royset і Wood [10].

Методи мурашиних колоній (Ant Colony Optimization, ACO) та генетичні алгоритми застосовуються до задач перехоплення значно рідше, ніж до класичних задач маршрутизації, оскільки функція виграшу $C(S)$ не є адитивною за ребрами і не допускає прямого використання стандартних феромонних

матриць. Адаптовані ACO-схеми, де феромон кодує корисність окремих ребер для атаки, показують прийнятну якість на невеликих задачах, проте поступаються жадібному алгоритму за часом і стабільністю на великих графах.

Важливою характеристикою будь-якого евристичного методу є апроксимаційна гарантія. На жаль, для загальної задачі SPI точних апроксимаційних гарантій для жадібного підходу не встановлено: функція $C(S)$ не є субмодулярною у загальному випадку, що позбавляє змоги застосувати стандартну теорему про $(1 - 1/e)$ -апроксимацію жадібного алгоритму для субмодулярних задач. Для окремих спрощених варіантів задачі, наприклад при $\alpha \to \infty$ (повне блокування ребер), субмодулярність зберігається і апроксимаційні гарантії можна отримати.

У табл. 1.2 наведено порівняльну характеристику евристичних підходів та MILP за ключовими показниками.

Таблиця 1.2 – Порівняння методів розв’язання задачі SPI

Метод	Гарантія якості	Складність (один запуск)	Масштабованість	Придатність для реального часу
MILP (Israeli-Wood)	Оптимальне	Експоненційна в гіршому випадку	Погана ($\|E\| > 10^3$)	Ні
Greedy	Немає	$O(B \cdot \|E\|^2 \log\|V\|)$	Середня	Частково
Локальний пошук	Немає	$O(k \cdot \|S\|(\|E\| - \|S\|) \cdot T_{SP})$	Середня	Ні
Simulated annealing	Немає	Залежить від розкладу	Середня	Ні
Метод	Гарантія якості	Складність (один запуск)	Масштабованість	Придатність для реального часу
Нейромережевий (ГМКА)	Немає (апроксимація)	$O(\|E\|)$ після навчання	Висока	Так

Підводячи підсумок, евристичні підходи заповнили прогалину між точними MILP-методами і практичними потребами, однак зберегли принципове обмеження: кожне нове завдання потребує запуску алгоритму з нуля, і час виконання зростає зі зростанням розміру задачі. Нейромережеві методи пропонують принципово відмінний підхід: одноразове навчання моделі дозволяє здійснювати миттєве виведення для нових екземплярів задачі для нових екземплярів задачі без повторного запуску оптимізаційного алгоритму.

1.4 Застосування методів машинного навчання та графових нейронних мереж до задач комбінаторної оптимізації

Традиційні підходи до комбінаторної оптимізації (КО) покладаються або на точні алгоритми з гарантією оптимальності, або на евристики, розроблені для конкретного класу задач. У 2010-х роках набула поширення альтернативна парадигма: навчання моделей, які здатні узагальнювати досвід розв’язання задач і застосовувати його до нових екземплярів без повторного запуску традиційного алгоритму. Огляд Bengio, Lodi і Prouvost [11] систематизує цей напрям і виділяє два основні способи використання ML у СО: як самостійний розв’язувач (end-to-end) та як компонент для вдосконалення класичних алгоритмів (learn to improve).

1.4.1 Наскрізні моделі та генерація рішень

Першою архітектурою, яка безпосередньо генерувала комбінаторні рішення, стала модель Pointer Networks, запропонована Vinyals, Fortunato і Jaitly [12]. У ній задача комбінаторної оптимізації формулюється як задача генерації послідовності: мережа на основі рекурентного кодувальника та механізму уваги послідовно «вказує» на вузли вхідного графа, формуючи рішення. Модель навчалась методом навчання з учителем на прикладах, згенерованих класичним розв’язувачем задачі комівояжера (TSP). Незважаючи на обмеження у вигляді

фіксованого розміру задачі та авторегресивного характеру генерації, Pointer Networks заклали концептуальну основу для подальших підходів.

1.4.2 Навчання з підкріпленням у комбінаторній оптимізації

Паралельно розвивалися підходи на основі навчання з підкріпленням (RI). Bello et al. [13] адаптували Pointer Networks для навчання без учителя: замість навчальних міток із розв'язувача модель отримувала нагороду у вигляді довжини побудованого маршруту, оптимізованої методом REINFORCE. Це відкрило можливість навчання у випадках, коли еталонні рішення недоступні або надто дорогі для обчислення.

1.4.3 Інтеграція GNN у класичні алгоритми

Принциповий крок до використання графової структури задачі зробили Khalil et al. [14] у роботі S2V-DQN. На задачах Maximum Cut (пошук розрізу графа максимальної ваги), Minimum Vertex Cover (знаходження найменшої множини вершин, що покриває всі ребра) та TSP (задача комівояжера — пошук найкоротшого замкненого маршруту), детальний опис яких наведено у класичному підручнику з дослідження операцій [43], S2V-DQN показав значно кращі результати, ніж підходи, що ігнорують структуру графа. Ця кваліфікаційна робота фактично заклала основу парадигми «GNN для CO», яка домінує у подальших дослідженнях.

Увага до маршрутних задач призвела до розробки моделі Attention Model (AM) Kool, van Hoof і Welling [15]. AM використовує трансформерний кодувальник для вузлів графа та механізм уваги для авторегресивної побудови маршруту. Навчання відбувалось методом REINFORCE із базовою лінією, що стабілізує градієнти. Модель демонструвала конкурентоспроможну якість відносно Concorde (точного розв'язувача TSP) при часі виведення на кілька порядків меншому [15].

1.4.4 Імітаційне навчання проти RL

Поряд із наскрізними підходами активно розвивалась парадигма інтеграції GNN у класичні алгоритми точної оптимізації. Gasse et al. [16] навчили GCN передбачати пріоритети гілкування у алгоритмі branch-and-bound: мережа отримувала на вхід двочастинний граф (змінні та обмеження MILP) і передбачала, яку змінну краще обрати на поточному кроці дерева пошуку. Це підхід суттєво прискорив роботу MILP-розв'язувачів на стандартних бенчмарках, зберігаючи гарантію оптимальності в межах самого алгоритму B&B.

Систематичний аналіз усього спектра підходів, від RL і навчання з учителем до гібридних схем, наведено у огляді Cappart et al. [17]. Автори систематизують три ролі GNN у КО: як сурогатна функція оцінки рішень, як евристика для керування пошуком і як засіб прискорення зовнішнього розв'язувача. Крім того, огляд підкреслює, що здатність GNN до узагальнення на графи різного розміру залишається відкритою проблемою: більшість моделей суттєво деградує при переході від тренувальних до суттєво більших тестових задач.

Важливим є розмежування між двома підходами до навчання у КО: навчанням з підкріпленням та імітаційним навчанням. У RL-підході модель самостійно досліджує простір рішень і отримує нагороду за якість побудованого рішення. Переваги: не потрібні еталонні рішення, модель може перевершити вчителя. Недоліки: нестабільне навчання на задачах з розрідженою нагородою, повільна збіжність. В імітаційному навчанні (imitation learning) модель навчається відтворювати рішення деякого «вчителя» – евристики або точного розв'язувача. Перевага: стабільніше і швидше навчання за рахунок густих навчальних сигналів. Недолік: якість моделі обмежена якістю вчителя, а зміщення розподілу між навчанням і тестуванням (distribution shift) може погіршити результати при виходженні за межі траєкторій вчителя. У даній роботі застосовується саме імітаційне навчання: жадібний алгоритм слугує вчителем, генеруючи мітки $y \in \{0,1\}^{|E|}$ для кожного навчального графа.

Загальна архітектура нейромережевих підходів до СО включає кілька спільних компонентів, показаних на рис. 1.3. На першому кроці вхідний граф кодується за допомогою GNN-кодувальника, що агрегує контекстну інформацію від сусідніх вузлів і будує ембедінги h_v для кожного вузла. На другому кроці застосовується блок виведення (readout), що на основі ембедингів вузлів формує оцінки для кожного ребра або вузла – кандидатів на включення у рішення. На третьому кроці модуль прийняття рішень (policy-модуль) перетворює оцінки на конкретне рішення: у задачах з бюджетним обмеженням це, як правило, вибір top-B найвище оцінених елементів. Таку загальну схему реалізує і модель ГМКА, з тією відмінністю, що шар виведення побудований на основі МКА, а не стандартного MLP.

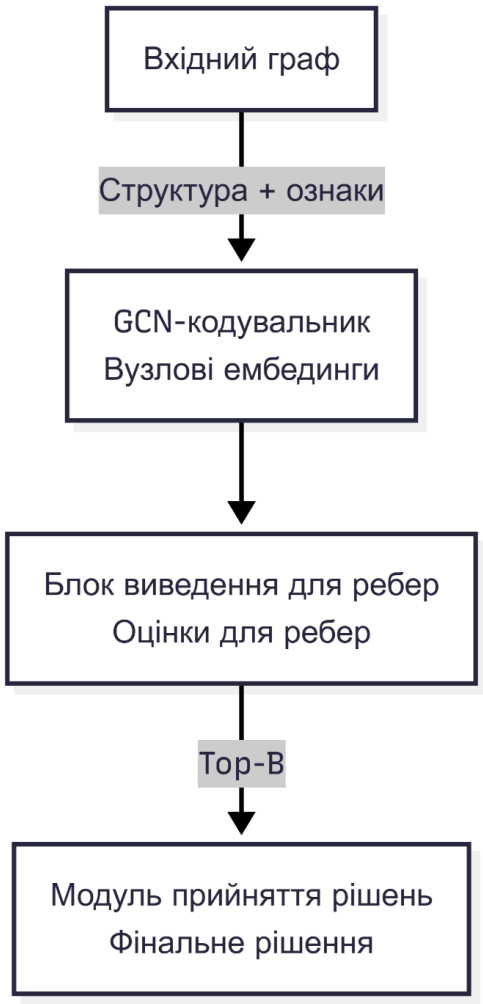


Рисунок 1.3 – Загальна схема нейромережевого підходу до СО: GCN-кодувальник, блок виведення для ребер (edge readout), модуль прийняття рішень (вибір top-B)

Незважаючи на значний прогрес у застосуванні GNN до CO-задач загального вигляду (TSP, MVC, Max-Cut), безпосереднє перенесення цих підходів до задач мережевого перехоплення нетривіальне. По-перше, функція виграшу $C(S)$ у SPI не є адитивною за ребрами і залежить від глобальної топології графа, що ускладнює навчання локальних edge-score функцій. По-друге, задача є двохрівневою, що привносить нелінійну залежність між рішенням інтердиктора та відповіддю оператора. Саме тому нейромережеві методи для задач перехоплення вимагають спеціально розроблених або адаптованих архітектур, здатних враховувати двохрівневу структуру задачі та неадитивний характер цільової функції.

1.5 Нейромережеві підходи до задач мережевого перехоплення

Застосування методів глибокого навчання безпосередньо до задач мережевого перехоплення є відносно новим напрямом, що сформувався переважно у 2021–2025 роках. На відміну від загальних CO-задач (TSP, MVC), задачі перехоплення мають специфічну двохрівневу структуру та неадитивну цільову функцію, що вимагає окремих архітектурних рішень.

1.5.1 RL-методи перехоплення

Першим помітним кроком стала робота [18], де метод навчання з підкріпленням на основі pointer-network архітектури застосовано до задачі перехоплення найкоротшого шляху. Модель навчалась на синтетичних графах регулярної решітки та випадкових графах у режимі відкладеної нагороди: після вибору фіксованого бюджету ребер для атаки модель отримувала нагороду у вигляді приросту довжини найкоротшого шляху. Підхід продемонстрував здатність нейромережі узагальнювати принципи вибору критичних ребер на нові

екземпляри задачі, хоча якість рішень поступалась жадібному алгоритму на великих графах через нестабільність навчання з розрідженою нагородою.

1.5.2 Supervised GNN-сурогати

Якісно новий підхід запропонований у [6] «Network Interdiction Goes Neural», представленій на KDD 2025. Автори використовують мультидольний GNN, що безпосередньо кодує структуру MILP-задачі перехоплення: вузли відповідають змінним та обмеженням задачі, а ребра відображають їхню залежність. Навчання відбувається за мітками від точного MILP-розв'язувача (supervised підхід), а сам GNN відіграє роль евристики для теплого старту та вибору гілок у процесі branch-and-cut. Метод продемонстрував суттєве прискорення порівняно з чистим MILP на задачах середнього розміру при незначній втраті якості.

Паралельно [19] розроблений підхід на основі навчання безпосередньо від розв'язувача з механізмом агрегування рішень по кількох екземплярах задачі. Модель у Computers & Operations Research 2025 навчає GNN-сурогат, що передбачає оптимальні ребра для атаки, використовуючи рішення точного розв'язувача як мітки. Ключова відмінність від [18] полягає у відмові від RL на користь прямої supervised регресії, що суттєво стабілізує тренування.

У табл. 1.3 наведено узагальнене порівняння основних нейромережевих підходів до задач мережевого перехоплення.

Спільна риса всіх перелічених підходів – конвеєрна схема: екземпляр задачі → точний розв'язувач або евристика → мітки → нейромережевий сурогат → швидке формування рішення. Ключова відмінність між підходами стосується вибору вчителя (MILP чи жадібний алгоритм), архітектури нейромережі та способу оцінки якості.

В ході дослідження, не було виявлено жодної рецензованої роботи з прямим застосуванням мереж Колмогорова–Арнольда (МКА) до задач з нульовою сумою або задач перехоплення [6, 20], що можна вважати науковою прогалиною, яку заповнює дана кваліфікаційна робота.

Таблиця 1.3 – Порівняння нейромережевих методів розв’язання SPI

Метод	Архітектура	Підхід до навчання	Датасет	Метрика
Huang et al. 2022 [18]	Pointer Network	RL (REINFORCE)	Grid, випадкові графи	Якість рішення vs MILP
Метод	Архітектура	Підхід до навчання	Датасет	Метрика
Zhang et al. 2025 [6]	Multipartite GNN	Supervised (MILP-мітки)	Benchmark instances	Час до MILP, gap
Kwon et al. 2025 [19]	GNN surrogate	Supervised (solver labels)	SPI instances	Solution quality
ГМКА	GCN + вихідний модуль МКА	Imitation (greedy-вчитель)	Grid, BA, SiouxFalls	Quality ratio Q

1.6 Архітектура мереж Колмогорова–Арнольда

З моменту своєї першої презентації у 2024 році мережі Колмогорова–Арнольда (МКА) стали однією з найважливіших інновацій у сфері глибокого навчання, запропонувавши фундаментально новий погляд на побудову нейронних апроксиматорів. На відміну від класичних багатошарових перцептронів, які спираються на фіксовані функції активації у вузлах, МКА переносить навчувану нелінійність безпосередньо на з’єднання (ребра) між нейронами. У цьому підрозділі детально розглядається математичний базис цієї архітектури, еволюція її сучасних модифікацій та обґрунтовуються передумови її застосування для складних графових задач.

Мережі Колмогорова–Арнольда (МКА) є однією з ключових інновацій у сфері глибокого навчання, що пропонує фундаментально новий погляд на

побудову нейронних апроксиматорів. На відміну від класичних багатошарових перцептронів, які спираються на фіксовані функції активації у вузлах, МКА переносить навчувану нелінійність безпосередньо на з'єднання (ребра) між нейронами. Подальший аналіз розкриває математичний базис цієї архітектури, еволюцію її сучасних модифікацій та передумови її застосування для складних графових задач.

1.6.1 Теоретичне підґрунтя

Теоретичною основою МКА є класична теорема Колмогорова–Арнольда (1957), яка стверджує, що будь-яку неперервну функцію $f: [0,1]^n \to \mathbb{R}$ можна представити у вигляді:

$$f(x_1, \dots, x_n) = \sum_{q=1}^{2n+1} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right), \tag{1.8}$$

де Φ_q – зовнішні функції;

$\phi_{q,p}$ – одновимірні внутрішні функції.

Теорема гарантує, що будь-яку многовимірну неперервну функцію можна звести до суперпозиції одновимірних функцій. Практична інтерпретація теореми потребує обережності: хоча представимість гарантована, зовнішні функції Φ_q можуть бути нерегулярними і нелегко параметризованими. У [21] показано, що класичну теорему не слід трактувати як «готовий рецепт мілкого навчання» – конструктивний варіант часто веде до глибоких мереж, а не до мілких суперпозицій.

1.6.2 Сучасна МКА-архітектура та RBF-базиси

Сучасна архітектура МКА, запропонована у [20] у 2024 році, операціоналізує цю теорему інакше: нелінійність переноситься з вузлів на ребра

мережі. У стандартному MLP кожен вузол містить фіксовану активаційну функцію (наприклад, ReLU або tanh), а ваги ребер є скалярами. У МКА ваги ребер замінюються навчуваними одновимірними функціями $\phi_{j,i}$, і вихід l-го шару обчислюється як:

$$h_j^{(l+1)} = \sum_{i=1}^{d_l} \phi_{j,i}^{(l)} \left(h_i^{(l)} \right), \tag{1.9}$$

де $\phi_{j,i}^{(l)}$ – навчувана одновимірна функція на ребрі (i, j) між шарами l та $l + 1$.

Практично важливим є вибір базисних функцій для параметризації ϕ. Оригінальна робота [20] використовує B-сплайни, що дають рівномірну апроксимацію на заданому інтервалі. Реалізація [22] замінює кубічні B-сплайни на радіальні базисні функції (RBF):

$$\phi_k(x) = \exp(-\gamma(x - c_k)^2), \tag{1.10}$$

де c_k – центри базисів, рівномірно розташовані на $[-1,1]$;

γ – параметр ширини гауссіани.

RBF-параметризація є обчислювально ефективнішою: вона відмовляється від рекурсивного обчислення B-сплайнів і забезпечує приблизно 3.3-кратне прискорення прямого проходу на GPU порівняно з efficient-kan при збереженні якості апроксимації. Саме цей варіант використовується у архітектурі ГМКА.

Теоретичний розвиток МКА у 2024–2026 роках відбувався стрімко. Newton-Kaczmarz метод [23] забезпечив локальну збіжність для процедури оцінки параметрів в МКА-подібних моделях, що є найсильнішим строгим результатом збіжності, відомим станом на 2026 рік. Res-KAN [24] встановив апроксимаційні межі у просторах Безова з незалежними від розмірності обмеженнями складності вибірки, що є важливим для обґрунтування застосування МКА до навчання сурогатів. Graph-KAN [25] продемонстрував конкурентоспроможність МКА-підходу до GNN у задачах прогнозування

молекулярних властивостей. MKA 2.0 [26] розширив архітектуру множинними вузлами та символьною компіляцією для наукового відкриття.

На рис. 1.4 показано структурну відмінність між MLP і MKA: у MLP нелінійність σ зосереджена у вузлах, тоді як у MKA навчувані функції $\phi_k(x)$ розміщені на ребрах.

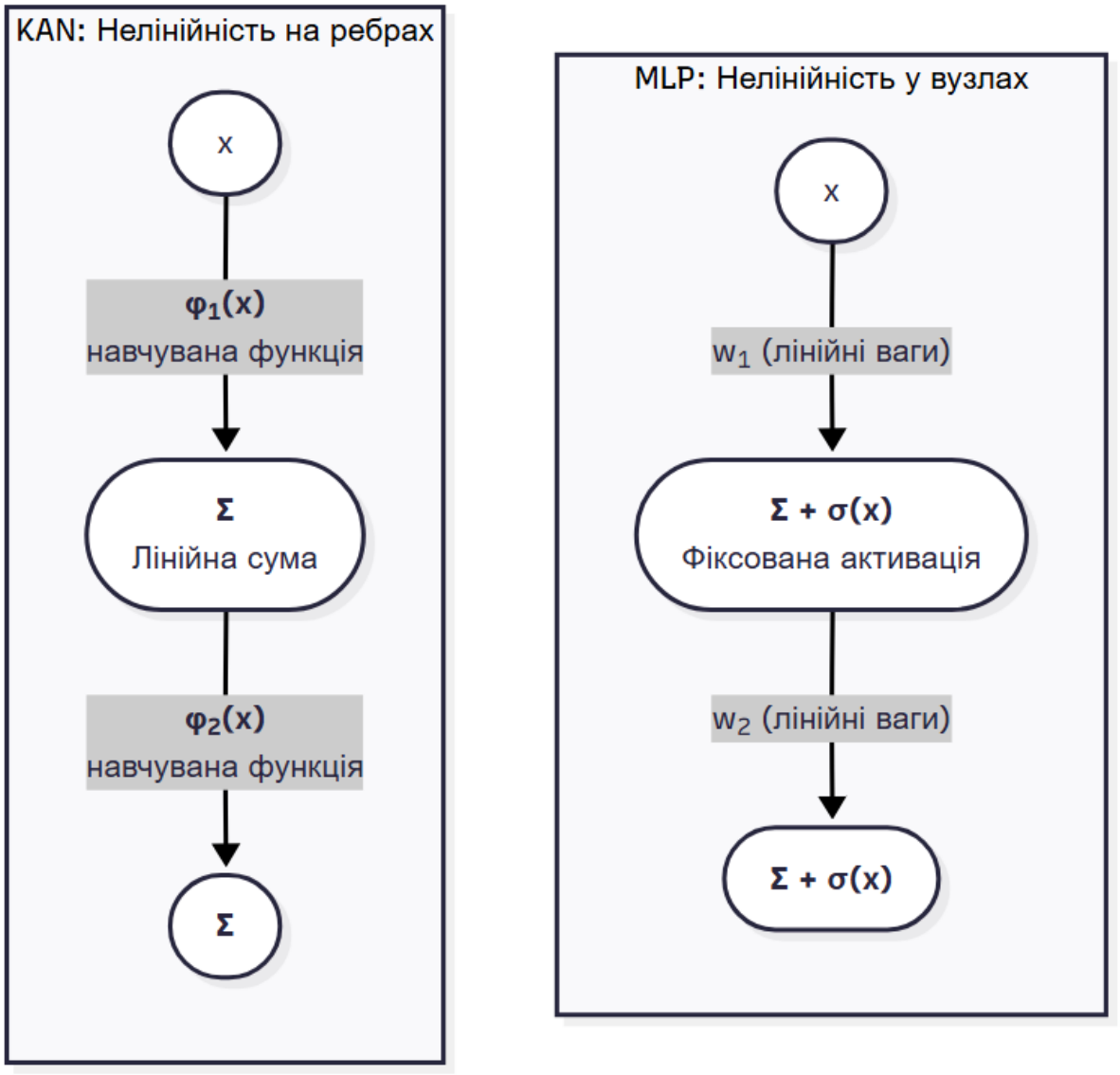


Рисунок 1.4 – Порівняння архітектур MLP (нелінійність у вузлах) і MKA (нелінійність на ребрах)

На рис. 1.5 показано хронологію розвитку МКА у 2024–2026 роках.

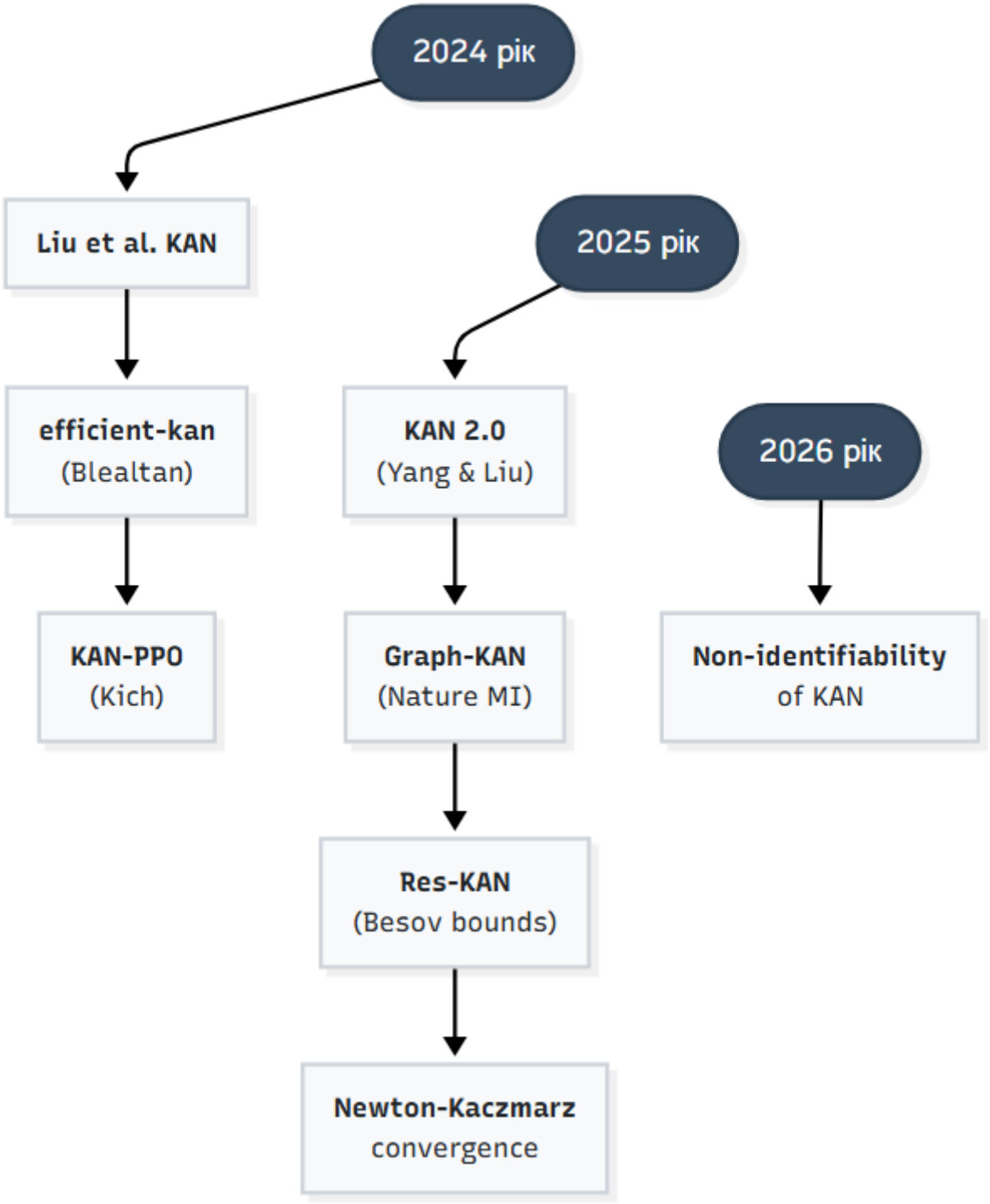


Рисунок 1.5 – Хронологія розвитку МКА

Емпіричні результати порівняння МКА та MLP неоднозначні. У [27] у контрольованому порівнянні для диференціальних рівнянь показано, що оригінальна сплайнова МКА поступається добре налаштованому MLP за

точністю та ефективністю у задачах з операторним навчанням. Водночас МКА перевершує MLP передусім в інтерпретованості та при малих датасетах – саме ці переваги є важливими у контексті задач мережевого перехоплення, де кількість тренувальних прикладів обмежена вартістю генерації міток від точного MILP-розв’язувача.

1.6.3 Обґрунтування вибору архітектури МКА для задач перехоплення

Задачі мережевого перехоплення утворюють зрілу область дослідження операційного менеджменту з чітко визначеними формулюваннями для SPI, MFI та задач з критичними вузлами. Класичні MILP-підходи забезпечують гарантовано оптимальні рішення, але їхня обчислювальна складність зростає експоненційно з розміром графа. Евристичні методи (жадібний алгоритм, локальний пошук, імітація відпалу) досягають прийнятної якості за поліноміальний час, однак без гарантій оптимальності.

З 2021 року формується нова хвиля нейромережевих підходів до задач перехоплення: RL-методи [18], supervised GNN-сурогати [6, 19]. Спільною рисою цих методів є навчання від точного розв’язувача або евристики-вчителя та переміщення обчислювального навантаження з фази отримання розв’язку на фазу навчання.

Архітектура МКА відкриває новий напрям: пореброві нелінійні функції замість фіксованих вузлових активацій, адитивна інтерпретованість без додаткових інструментів типу SHAP або LIME, та потенційно кращу апроксимацію при малих датасетах. Водночас емпіричні переваги МКА над MLP залежать від задачі і не є гарантованими.

Ключова прогалина, виявлена у ході огляду: станом на 2025 рік жодна рецензована робота не досліджувала безпосереднє застосування МКА до задач мережевого перехоплення [46]. Дана кваліфікаційна робота усуває цю прогалину, поєднуючи GCN-кодувальник та вихідний модуль МКА (модель ГМКА) для задачі SPI з імітаційним навчанням від жадібного вчителя.

2 ТЕОРЕТИЧНІ ОСНОВИ ТА ФОРМАЛІЗАЦІЯ ЗАДАЧІ

2.1 Формальна модель задачі мережевого перехоплення найкоротшого шляху

Задача мережевого перехоплення найкоротшого шляху (Shortest Path Interdiction, SPI) є двохрівневою задачею оптимізації на орієнтованому зваженому графі. Нехай задано орієнтований граф $G = (V, E)$, де V – множина вершин, $E \subseteq V \times V$ – множина ребер із базовими вагами $c_e > 0$ для кожного $e \in E$. Фіксується пара вузлів джерело-стік (s, t), бюджет інтердиктора $B \in \mathbb{Z}_{>0}$ та штрафний коефіцієнт $\alpha > 1$.

Інтердиктор (лідер) обирає бінарний вектор $y \in \{0,1\}^{|E|}$, де $y_e = 1$ означає атаку на ребро e. Після атаки вага ребра стає:

$$w_e(y) = c_e(1 + (\alpha - 1)\, y_e), \quad e \in E, \tag{2.1}$$

де w_e – вага ребра після атаки;

c_e – базова вага ребра;

α – коефіцієнт збільшення ваги при перехопленні.

При $y_e = 0$ вага залишається незміненою: $w_e = c_e$. При $y_e = 1$ вага зростає до $\alpha \cdot c_e$. Таким чином, $\alpha = 1$ відповідає відсутності ефекту від атаки, великі значення α моделюють часткове або повне блокування ребра.

Захисник (follower) знає множину атакованих ребер і будує найкоротший шлях у модифікованому графі за допомогою алгоритму Дейкстри. Вартість найкоротшого шляху при стратегії інтердиктора y:

$$C(y) = \min_{P \in \mathcal{P}_{st}} \sum_{e \in P} w_e\,(y), \tag{2.2}$$

де $\mathcal{P}_{st}$ – множина всіх простих шляхів від s до t у графі G.

Дворівнева задача оптимізації формулюється як:

$$\max_{\substack{y\in\{0,1\}^{|E|}\\ \sum_e y_e\le B}} C(y) = \max_{\substack{y\in\{0,1\}^{|E|}\\ \sum_e y_e\le B}} \min_{P\in\mathcal{P}_{st}} \sum_{e\in P} c_e\,(1+(\alpha-1)y_e). \tag{2.3}$$

Ця задача є NP-важкою у загальному випадку [2]. Внутрішня задача захисника є лінійною програмою (задача найкоротшого шляху з невід'ємними вагами зводиться до LP), що відкриває можливість дуалізації.

2.1.1 Умова нульової суми

Визначимо функції виграшу гравців:

$$u_{leader}(y) = C(y), \qquad u_{defender}(y) = -C(y). \tag{2.4}$$

Тоді $u_{leader}(y) + u_{defender}(y) = C(y) - C(y) = 0$ для будь-якої стратегії y. Умова нульової суми виконується точно. Метрика приросту вартості $\Delta(y) = C(y) - C(\emptyset)$, що використовується у реалізації, є лише зсувом виграшу на константу $C(\emptyset)$ і не змінює стратегічну структуру гри.

2.1.2 Stackelberg-гра проти Nash-рівноваги

В задачі SPI гра є послідовною (Stackelberg): інтердиктор як лідер обирає стратегію y першим, захисник знає y і оптимально реагує. Це відрізняється від одночасної Nash-рівноваги. Stackelberg-оптимум є окремим випадком задачі максимін з детермінованими стратегіями. У варіанті зі змішаними стратегіями [5] для антагоністичних ігор виконується мінімаксна теорема фон Неймана, і значення гри однакове для обох порядків оптимізації.

2.1.3 Зведення до однорівневого MILP

Оскільки внутрішня задача захисника є лінійною програмою, її можна дуалізувати і підставити в задачу лідера. Введемо потенціали π_v (двоїсті змінні) для кожного вузла $v \in V$. Тоді однорівнева MILP-постановка виглядає так:

$$\max_{y \in \{0,1\}^{|E|}} \pi_t - \pi_s \tag{2.5}$$

за умов:

$$\pi_v - \pi_u \leq c_e + (\alpha - 1)c_e \cdot y_e, \quad \forall (u,v) \in E, \tag{2.6}$$

$$\sum_{e \in E} y_e \leq B, \quad \pi_s = 0, \tag{2.7}$$

де π_v – потенціал вузла v;

y_e – бінарна змінна атаки ребра e.

Формулювання 2.5-2.7 реалізовано у базовому MILP-розв'язувачі, що використовується як оракул для генерації навчальних міток у даній роботі. Можливість такого зведення є прямим наслідком LP-структури задачі захисника.

2.1.4 Стохастичний варіант

В деяких конфігураціях до реберних вартостей при тестуванні додається гаусіанівський шум:

$$c_e^{\text{test}} = c_e^{\text{base}} \cdot (1 + \varepsilon_e), \text{ де } \varepsilon_e \sim \mathcal{N}(0, \sigma^2). \tag{2.8}$$

У цьому разі умова нульової суми зберігається лише в очікуванні: $\mathbb{E}[u_{\text{leader}} + u_{\text{defender}}] = 0$, але не поточечно.

У табл. 2.1 наведено повну нотацію моделі.

Таблиця 2.1 – Нотація формальної моделі SPI

Символ	Зміст	Типові значення
$G = (V, E)$	Орієнтований граф	Grid 10×10, BA n=64, SiouxFalls
c_e	Базова вага ребра	Uniform(1,10)
(s, t)	Пара джерело-стік	Strategy random_far
B	Бюджет інтердиктора	3, 5, 7
α	Штрафний коефіцієнт	3.0, 5.0
$y_e \in \{0,1\}$	Рішення лідера для ребра e	-
$C(y)$	Вартість SP після атаки y	-
$\Delta(y)$	Приріст $C(y) - C(\emptyset)$	Основна метрика якості
π_v	Потенціал вузла (двоїсті змінні)	-

На рис. 2.1 показано послідовну гру Stackelberg на прикладі фрагменту графа: вихідний найкоротший шлях, а також шлях після атаки підмножини ребер і нова оптимальна відповідь захисника.

2.2 Параметри бюджету B та інтенсивності α: чутливість і сценарії застосування

Поведінка моделі SPI визначається двома ключовими параметрами: бюджетом інтердиктора B та штрафним коефіцієнтом α. Розуміння того, як кожен з цих параметрів впливає на значення цільової функції, є необхідним для коректної інтерпретації результатів експериментів.

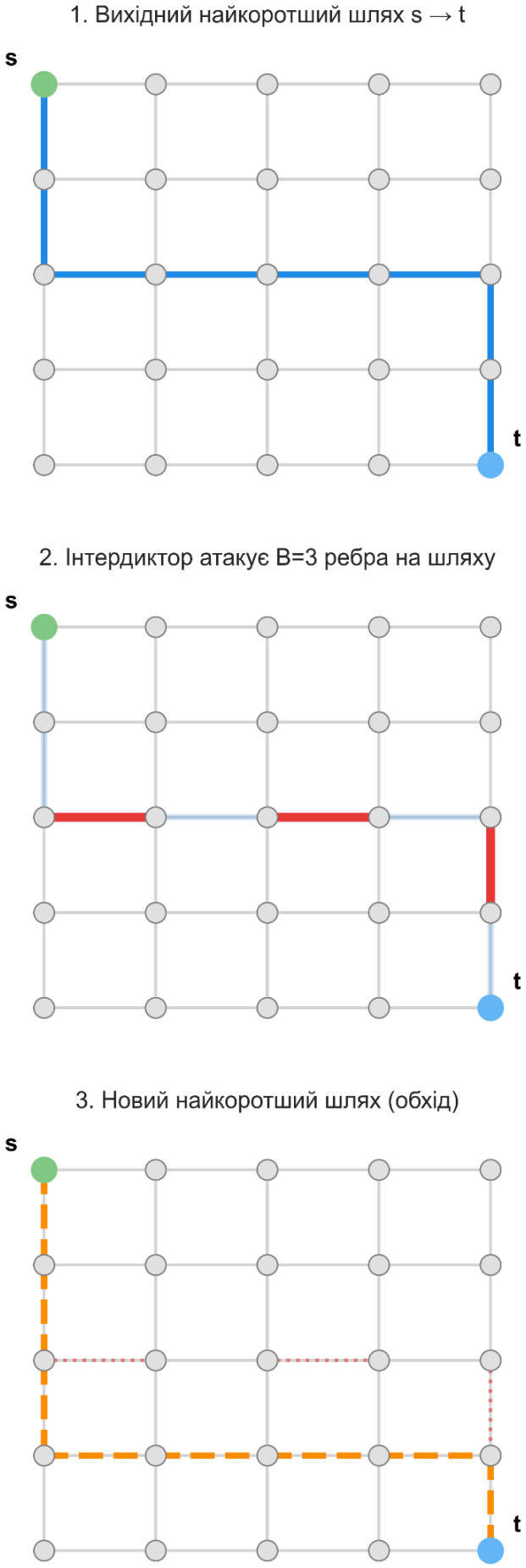


Рисунок 2.1 – Ілюстрація послідовної гри Stackelberg

2.2.1 Вплив бюджету B на цільову функцію

Параметр B обмежує кількість ребер, які інтердиктор може атакувати одночасно. При $B = 0$ атака неможлива, і $C(y) = C(\emptyset)$ – вартість найкоротшого шляху у незміненому графі. Зі збільшенням B від нуля до $|E|$ цільова функція $C^*(B) = \max_{y:\sum y_e \le B} C(y)$ монотонно не спадає: більший бюджет розширює допустиму множину стратегій і не може погіршити оптимальне рішення лідера. Формально, для $B_1 \le B_2$ виконується:

$$C^*(B_1) \le C^*(B_2). \tag{2.9}$$

При $B \ge |E|$ інтердиктор може атакувати всі ребра і повністю блокувати граф, якщо $\alpha \to \infty$. На практиці значення $B = 3,5,7$ відповідають типовим рівням ресурсів атакувальника у задачах захисту критичної інфраструктури [1].

Зростання $\Delta(y) = C(y) - C(\emptyset)$ зі збільшенням B не є лінійним: перший вибраний ребро (найкритичніше за жадібним алгоритмом) дає найбільший приріст, а кожне наступне – менший. Ця субадитивність пояснюється тим, що знищення одного ребра може зробити інші менш цінними, якщо захисник вже вимушений обходити перше заблоковане ребро.

2.2.2 Чутливість моделі до інтенсивності атаки α

Параметр $\alpha > 1$ задає інтенсивність впливу атаки на вагу ребра. При $\alpha = 1$ атака не має жодного ефекту: $w_e(y) = c_e$ незалежно від y_e. При $\alpha \to \infty$ атаковані ребра фактично стають непрохідними – захисник змушений шукати шлях, що їх обходить. У проміжних значеннях атака лише обтяжує ребро, але не забороняє його використання.

Вплив α на цільову функцію також є монотонним: для $\alpha_1 \le \alpha_2$ і фіксованої стратегії y:

$$C(y;\alpha_1) \le C(y;\alpha_2). \quad (2.10)$$

Проте оптимальна стратегія y^* може змінюватись зі зміною α: при малих α вигідно атакувати ребра, що входять до багатьох коротких шляхів (хаби), тоді як при великих α достатньо заблокувати одне вузьке місце (bridge). У даній роботі використано значення $\alpha \in \{3.0,5.0\}$, які відповідають помірному та сильному обтяженню ребер відповідно.

Якщо $\alpha = \infty$ і бюджет B перевищує мінімальний розріз між s і t, граф стає незв'язним (з точки зору шляхів від s до t) і задача не має розв'язку у звичайному сенсі – вартість найкоротшого шляху стає нескінченною. Для уникнення такої вироджених ситуацій у реалізації до задачі додаються достатньо великі штрафні ваги замість нескінченності.

2.2.3 Сценарії застосування SPI

У класичній роботі [5] розглядається варіант задачі, де обидва гравці можуть рандомізувати свої дії: інтердиктор обирає розподіл ймовірностей над підмножинами ребер для атаки, захисник – розподіл над шляхами. Авторами доведено, що оптимальні змішані стратегії обох гравців обчислюються за поліноміальний час за допомогою методів мережевих потоків, незважаючи на експоненційну кількість можливих шляхів. Модель зі змішаними стратегіями була мотивована задачами перехоплення контрабанди через транспортні мережі та в подальшому застосовувалась до задач захисту критичної інфраструктури і кібербезпеки.

У табл. 2.2 наведено практичні сценарії застосування задачі SPI з відповідними ролями параметрів B та α.

Таблиця 2.2 – Сценарії застосування задачі мережевого перехоплення

Домен	Роль інтердиктора	Роль захисника	Параметр B	Параметр α
Кібербезпека	Атакувальник блокує канали зв’язку	Маршрутизатор шукає обхідний шлях	Число вразливостей, що можуть бути проексплуатовані	Деградація пропускної здатності каналу
Транспортна мережа	Влада закриває ділянки доріг	Водій шукає найкоротший маршрут	Бюджет на перекриття (персонал, техніка)	Коефіцієнт збільшення часу в дорозі
Енергетика	Противник атакує лінії електропередачі	Оператор мережі шукає шлях подачі	Кількість ліній, доступних для атаки	Зниження потужності ушкодженої лінії
Контрабанда / безпека кордонів	Митниця перевіряє маршрути	Контрабандист обирає шлях	Ресурси митниці (пости, патрулі)	Ймовірність виявлення на маршруті
Домен	Роль інтердиктора	Роль захисника	Параметр B	Параметр α
Логістика (supply chain)	Конкурент порушує постачання	Менеджер обирає маршрут доставки	Бюджет на зрив постачання	Затримка через порушення на ділянці

2.3 Перехід до задачі навчання з учителем

Задача SPI у своїй двохрівневій постановці є NP-важкою і вимагає MILP-розв'язувача або евристики для отримання точних рішень. Для великих графів ці підходи стають обчислювально недоступними. Метою цього підрозділу є формально обґрунтувати, як SPI переформулюється як задача supervised навчання на графах.

Позначимо через $\mathcal{D}$ датасет пар (G_i, y_i^*), де $G_i = (V_i, E_i, c^{(i)})$ – граф із вагами, а $y_i^* \in \{0,1\}^{|E_i|}$ – стратегія інтердиктора, отримана від оракула-вчителя. Задача навчання полягає у знаходженні функції $f_\theta: \mathcal{G} \to \mathbb{R}^{|E|}$ (параметризованої нейромережею), що для кожного ребра $e \in E$ виробляє логіт $\hat{y}_e$. Кінцеве рішення формується відбором B ребер із найвищими логітами:

$$\hat{S} = \text{top-B}(\{\hat{y}_e\}_{e \in E}). \tag{2.11}$$

2.3.1 Формування датасету та жадібний оракул

Оракулом-вчителем є жадібний алгоритм покрокового вибору ребер. На кожному кроці $k = 1, \dots, B$ жадібник обирає ребро, видалення якого максимально збільшує вартість найкоротшого шляху:

$$e_k^* = \arg\max_{e \in E \setminus S_{k-1}} [C(S_{k-1} \cup \{e\}) - C(S_{k-1})], \tag{2.12}$$

де S_{k-1} – підмножина ребер, вже обрана на попередніх кроках;

$C(\cdot)$ – вартість найкоротшого шляху у модифікованому графі.

Результат $S^* = \{e_1^*, \dots, e_B^*\}$ перетворюється на бінарний вектор міток $y^* \in \{0,1\}^{|E|}$. Жадібний алгоритм не гарантує глобальної оптимальності, але є змістовним наближенням і значно швидшим від MILP на великих графах. Для малих графів (до приблизно 30 вузлів) жадібного вчителя може бути замінено точним MILP-оракулом.

2.3.2 Проблема зміщення розподілу (Distribution Shift)

Зміщення між навчальним і тестовим розподілами (distribution shift) виникає у задачах послідовного прийняття рішень: невеличкі помилки моделі породжують нові стани, яких вона не бачила під час навчання, і похибки накопичуються. Алгоритм DAgger (Dataset Aggregation), запропонований у [28], усуває цю проблему: на кожній ітерації поточна модель запускається на навчальних графах, нові породжені стани запитуються у вчителя, і отримані пари агрегуються до датасету. Такий підхід зводиться до no-regret онлайн-навчання і гарантує, що очікувані втрати навченої стратегії обмежені сумою втрат найкращої стратегії у класі та регретом алгоритму.

У цій роботі повний цикл DAgger не реалізовано через обчислювальну вартість повторного запиту вчителя, однак підхід відповідає загальній схемі імітаційного навчання: жадібний вчитель генерує мітки одноразово для фіксованого датасету, а модель навчається апроксимувати ці мітки у режимі офлайн. Навчальна схема зображена на рис. 2.2.

Описаний підхід декомпонує вихідну задачу SPI на три фази: (1) генерація датасету – офлайн, за допомогою вчителя; (2) навчання – supervised класифікація ребер; (3) виведення – прямий підхід нейромережі за константний час незалежно від розміру задачі. Ця схема відповідає загальній парадигмі «вчись від розв’язувача» [13, 17].

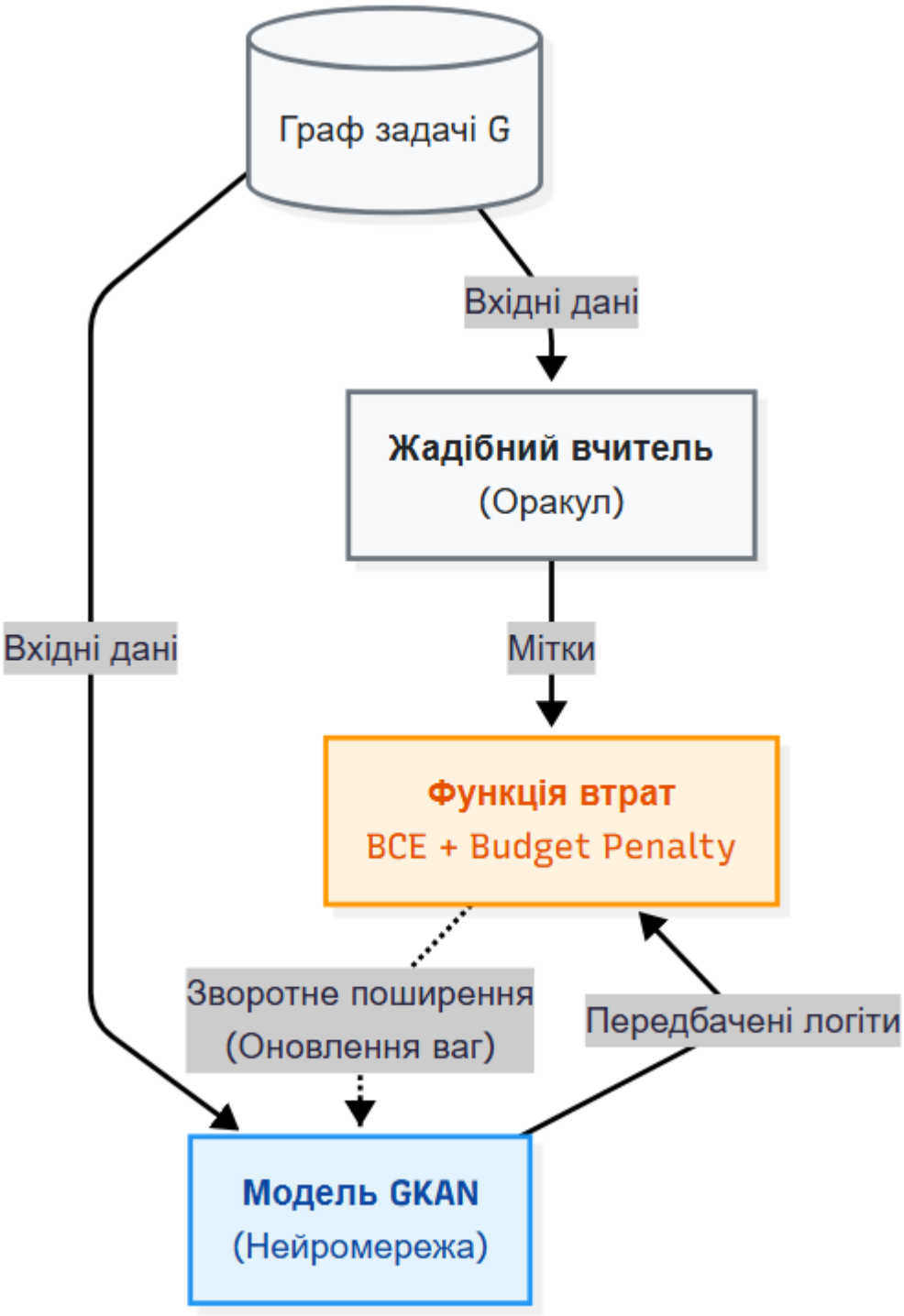


Рисунок 2.2 – Архітектура навчання ГМКА: жадібний вчитель генерує мітки $y^* \in \{0,1\}^{|E|}$ для кожного графа, модель передбачає логіти $\hat{y}$, функція втрат мінімізує їхню різницю

2.4 Теорема Колмогорова–Арнольда та її застосування до МКА

Теоретичним фундаментом для розробленої архітектури слугує класична теорема про представлення неперервних функцій. У цьому підрозділі розглядається математичний зміст оригінальної теореми, обмеження щодо її прямого використання та процес адаптації цієї концепції до побудови багатошарових нейромереж, що дозволило замінити традиційні вузлові активації на гнучкі навчувані функції на ребрах.

2.4.1 Класична теорема

Теорема Колмогорова–Арнольда (1957) є повним вирішенням 13-ї проблеми Гільберта про представлення неперервних функцій. Теорема стверджує: для будь-якого цілого $n \geq 2$ та будь-якої неперервної функції $f: [0,1]^n \to \mathbb{R}$ існує представлення:

$$f(x_1, \dots, x_n) = \sum_{q=0}^{2n} \Phi_q \left(\sum_{p=1}^{n} \phi_{q,p}(x_p) \right), \tag{2.13}$$

де $\Phi_q: \mathbb{R} \to \mathbb{R}$ – неперервні зовнішні функції, що залежать від конкретної функції f;

$\phi_{q,p}: [0,1] \to \mathbb{R}$ – неперервні внутрішні функції, що залежать лише від розмірності n і є універсальними: одні й ті самі $\phi_{q,p}$ підходять для всіх f.

Суттєвим є те, що сума містить рівно $2n + 1$ зовнішніх доданків (від $q = 0$ до $q = 2n$). Внутрішні функції $\phi_{q,p}$ можуть бути обрані заздалегідь, не знаючи f, і залишаються фіксованими для всього класу неперервних функцій. Зовнішні функції Φ_q при цьому залежать від f і, в загальному випадку, можуть бути нерегулярними (наприклад, не є диференційовними).

Теорема гарантує лише існування подання, але не його конструктивність: явна побудова $\phi_{q,p}$ є математично нетривіальним завданням, а внутрішні функції можуть бути патологічними (монотонні функції, що не є гладкими). Більше того, теорема забезпечує лише дворівневу суперпозицію із фіксованою шириною $2n + 1$, тоді як реальні задачі апроксимації потребують контрольованої гладкості та ефективної параметризації.

2.4.2 Перехід до багатошарової архітектури

Саме тому у [20] запропоновано принципово іншу інтерпретацію: мережа Колмогорова–Арнольда (МКА) використовує теорему не як пряму конструкцію, а як архітектурну метафору. Замість двошарової суперпозиції глибока МКА у [20] будується як стек шарів, де кожне ребро несе навчувану одновимірну функцію. Вихід l-го шару:

$$h_j^{(l+1)} = \sum_{i=1}^{d_l} \phi_{j,i}^{(l)} \left(h_i^{(l)} \right) \tag{2.14}$$

де $\phi_{j,i}^{(l)}$ – навчувана одновимірна функція на ребрі $(i \to j)$ між шарами l та $l + 1$;

d_l – ширина шару l.

Така багатошарова конструкція забезпечує значно більшу виразність, ніж класична двошарова суперпозиція (2.9), і допускає ефективне навчання за допомогою градієнтного спуску.

2.4.3 Порівняльні переваги МКА над MLP

Вибір конкретного сімейства базисних функцій для $\phi_{j,i}$ є критичним для практичного навчання. Оригінальний МКА [20] використовує B-сплайни, що забезпечують рівномірну апроксимацію, але вимагають рекурсивного

обчислення коефіцієнтів і є відносно повільними на GPU. Було реалізовано гаусіанові радіальні базисні функції (RBF), що мають аналітичну форму та легко диференціюються. Банк із K базисних функцій для вхідного скаляра x:

$$\phi_k(x) = \exp\left(-\frac{(x - c_k)^2}{2\sigma_k^2}\right), \quad k = 1, \dots, K, \tag{2.15}$$

де c_k – центр k-ї базисної функції;

σ_k – ширина k-ї базисної функції.

Центри ініціалізуються рівномірно на відрізку $[-1,1]$:

$$c_k = -1 + \frac{2(k-1)}{K-1}, \quad k = 1, \dots, K. \tag{2.16}$$

Ширини параметризуються через логарифм $\log\sigma_k$, що ініціалізується нулями ($\sigma_k = 1$ на початку навчання). Такий підхід гарантує $\sigma_k > 0$ без явних обмежень на параметри. Обидва – $\{c_k\}$ і $\{\log\sigma_k\}$ – є навчуваними параметрами моделі.

Вихід одного KAN-вузла у KANReadout формується як лінійна комбінація всіх K базисних функцій по всіх D вхідних вимірах:

$$\hat{y}(\mathbf{z}) = b + \sum_{d=1}^{D} f_d(\tilde{z}_d), \quad f_d(\tilde{z}_d) = \boldsymbol{\phi}_d(\tilde{z}_d)^\top \mathbf{w}_d, \tag{2.17}$$

де $w_d \in \mathbb{R}^K$ – лінійні ваги для d-го виміру;

$b \in \mathbb{R}$ – зсув;

$\tilde{z}_d$ – нормалізоване значення d-го виміру вхідного вектора.

Формула (2.17) є точним адитивним розкладанням виходу по вхідних ознаках: кожна функція f_d відповідає за внесок лише одного виміру і може бути візуалізована незалежно, що є прямою реалізацією духу теореми Колмогорова–Арнольда у практичній архітектурі.

У табл. 2.3 наведено ключові теоретичні відмінності між МКА та MLP.

Таблиця 2.3 – Теоретичне порівняння архітектур МКА та MLP

Властивість	MLP	МКА
Теоретична основа	Теорема Цибенка (1989)	Теорема Колмогорова–Арнольда (1957)
Розташування нелінійності	Вузли (фіксована σ: ReLU, tanh тощо)	Ребра (навчувані ϕ_k: B-сплайни або RBF)
Адитивність виходу	Не гарантується	Точне: $\hat{y} = b + \sum_d f_d(z_d)$
Кількість параметрів (readout)	$O(D \cdot h)$, $h = 128 \approx 16{,}640$	$O(D \cdot K)$, $K = 16 \approx 2{,}080$
Інтерпретованість	Потребує SHAP, LIME або ін.	Вбудована: $f_d(z_d)$ = внесок ознаки d
Навчуваність базисів	Ні (фіксована функція)	Так (центри c_k і ширини σ_k є параметрами)

На рис. 2.3 показано схему двошарового МКА, що відповідає структурі теореми (2.9): n вхідних вузлів, $2n + 1$ прихованих вузлів, один вихідний вузол, де кожне ребро є навчуваною одновимірною функцією.

2.4.4 Синтез теоретичної моделі та обґрунтування ГМКА

Перший вибір: SPI → supervised навчання на ребрах. Задача мережевого перехоплення найкоротшого шляху формалізована як дворівнева задача оптимізації у грі Stackelberg з нульовою сумою. Показано, що умова нульової суми $u_L + u_D = 0$ виконується точно для будь-якої стратегії інтердиктора. Завдяки LP-структурі задачі захисника двохрівневу задачу зведено до однорівневого MILP через LP-двоїстість, що дозволяє генерувати точні або наближені навчальні мітки. Жадібний алгоритм є практичним оракулом-вчителем: він покроково обирає ребра з найбільшим маргінальним приростом вартості шляху захисника і масштабується на великих графах. Таким чином, задача SPI природно переходить у supervised класифікацію ребер: вхід – граф із вагами і параметри (B, α), вихід – бінарний вектор міток $y^* \in \{0,1\}^{|E|}$.

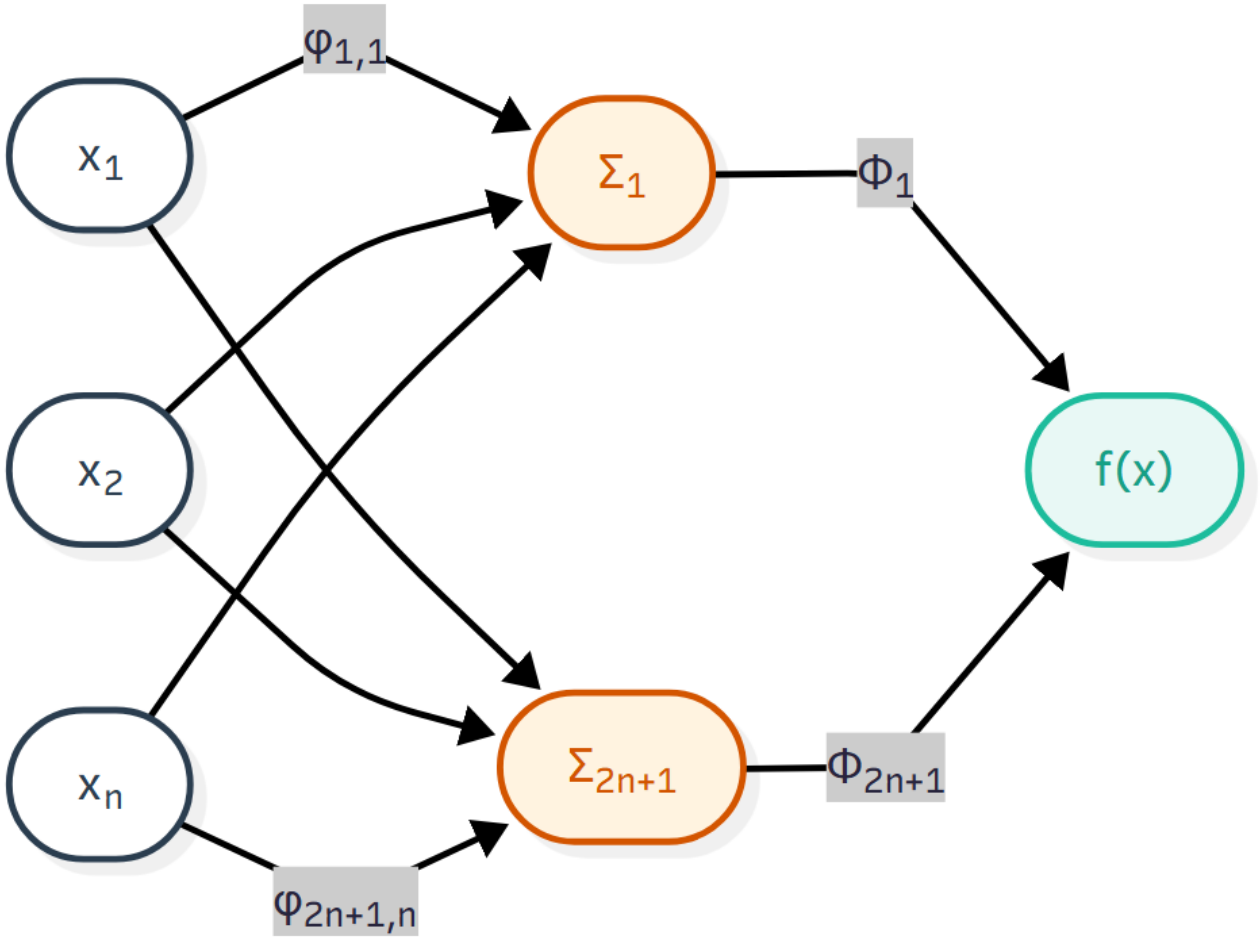


Рисунок 2.3 – Структура двошарового МКА у відповідності до теореми Колмогорова–Арнольда: ребра несуть навчувані функції $\phi_{q,p}$ (внутрішні) та Φ_q (зовнішні)

Другий вибір: графова структура → GCN-кодувальник. Цільова функція C(y) залежить від глобальної топології графа: блокування одного ребра може змінити оптимальний шлях кардинально, якщо ребро є мостом або вузьким місцем. Локальні ознаки ребра (базова вага, довжина) недостатні для оцінки цього глобального ефекту. GCN-кодувальник вирішує цю проблему агрегацією інформації у k-окрузі кожного вузла: за L_{gcn} шарів кожен вузол отримує ембединг, що відображає не лише його власні атрибути, а й структурний контекст на відстані до L_{gcn} хопів. Для сітки 10×10 двошаровий GCN охоплює п’ятивузлові «хрести», достатні для локалізації вузьких місць між s і t.

Третій вибір: edge-wise scoring → вихідний модуль МКА замість EdgeMLP. Для кожного ребра (u, v) формується вектор ознак $\mathbf{z}_{uv} = [\mathbf{h}_u \parallel \mathbf{h}_v \parallel \mathbf{a}_{uv}]$ розмірності $D = 130$. Задача шару виведення – перетворити $\mathbf{z}_{uv}$ у скалярний логіт важливості ребра. Теорема Колмогорова–Арнольда обґрунтовує, що будь-яку неперервну функцію D змінних можна точно представити як суперпозицію одновимірних функцій – саме це реалізує вихідний модуль МКА через адитивне розкладання (2.13). Ключові переваги над EdgeMLP у даному застосуванні:

- адитивна інтерпретованість без зовнішніх інструментів: $\hat{y} = b + \sum_d f_d(\tilde{z}_d)$, де кожна f_d – внесок окремої ознаки;
- менша кількість параметрів readout ($\approx 2{,}080$ проти $\approx 16{,}640$ у MLP), що критично при обмеженому числі навчальних прикладів;
- навчувані центри і ширини RBF-базисів, що дозволяють адаптуватись до конкретного розподілу ознак ребер.

Разом ці три вибори визначають архітектуру ГМКА: GCN-кодувальник фіксує топологічний контекст, EdgeFeaturizer формує реброві вектори, вихідний модуль МКА перетворює їх у логіти значущості. Жадібний вчитель та функція втрат із бюджетним штрафом утворюють тренувальний цикл навчання.

3 АРХІТЕКТУРА МОДЕЛІ ГМКА ДЛЯ ЗАДАЧІ МЕРЕЖЕВОГО ПЕРЕХОПЛЕННЯ

3.1 Загальна постановка та архітектура розв’язання

Перехід від теоретичних концепцій до програмної реалізації вимагає чіткого визначення загальної постановки задачі. Програмна схема розв'язання базується на послідовній обробці графових даних – від початкового кодування топології до фінальної оцінки ребер. Для забезпечення коректної роботи моделі GKAN формуються відповідні конфігурації тренувального циклу та визначаються специфічні методи регуляризації.

3.1.1 Принципові вимоги до архітектури

Модель ГМКА реалізує послідовність перетворення графа задачі SPI у ранжований список ребер-кандидатів для інтердикції. Архітектура проєктувалась із трьох принципових вимог:

- структурне усвідомлення – оцінка ребра має залежати від його топологічного контексту в усьому графі, а не лише від локальних атрибутів;
- гнучка нелінійність – вхідні ознаки ребер можуть бути нелінійно і неоднорідно взаємопов’язані, тому фіксована нелінійність класичного MLP може бути обмежувальною;
- адаптивна інтерпретованість – модель має надавати можливість аналізу того, які ознаки ребер найбільше впливають на прийняте рішення, без застосування зовнішніх інструментів типу SHAP (SHapley Additive exPlanations) [31] або LIME (Local Interpretable Model-agnostic Explanations) [32].

3.1.2 Вхідні дані та обчислювальний потік

Вхідні дані. На вхід модель отримує граф $G = (V, E)$ у форматі PyTorch Geometric. Вибір цього формату є принциповим для забезпечення високої обчислювальної ефективності при роботі з розрідженими структурами та гарантує повну сумісність із шарами графової згортки GCNConv. Дані передаються у вигляді об'єкта із такими атрибутами:

- матриця вузлових ознак $X \in \mathbb{R}^{N\times 2}$ – нормалізовані координати вузлів $(x_u/(L-1),\ y_u/(L-1))$, де L – сторона сіткового графа;
- матриця ребрових атрибутів $A \in \mathbb{R}^{2|E|\times 2}$ – базова вартість (base_cost) та евклідова довжина (length) кожного ребра у обох напрямках;
- edge_index $\in \mathbb{Z}^{2\times 2|E|}$ – спрямований індекс ребер (кожне ненаправлене ребро дубльовано).

Повний обчислювальний потік:

- $X \in \mathbb{R}^{N\times 2}$ – вузлові координати;
- GCNEncoder: $X \overset{\text{GCNConv}\times 2}{\rightarrow} H \in \mathbb{R}^{N\times 64}$ – вузлові ембединги;
- EdgeFeaturizer: $(H, A) \rightarrow Z \in \mathbb{R}^{2|E|\times 130}$, де $z_e = [h_u \parallel h_v \parallel a_e]$;
- LayerNorm (у KANReadout): $Z \rightarrow \tilde{Z}$ – нормалізація по розмірах;
- RBF-банки: $\tilde{Z} \rightarrow \Phi \in \mathbb{R}^{2|E|\times 2080}$ ($130 \times 16 = 2080$ базисних функцій);
- Dropout (опційно): $\Phi \rightarrow \Phi_{drop}$ – регуляризація базисів;
- лінійний вихід: $\hat{y} = \Phi_{drop}\, w + b \in \mathbb{R}^{2|E|}$ – логіти для кожного спрямованого ребра;
- агрегація напрямків: $\hat{s}_e = \max(\hat{y}_{u\rightarrow v},\ \hat{y}_{v\rightarrow u})$ – ненаправлені оцінки;
- відбір: $\hat{S} = \text{top-}B(\{\hat{s}_e\}_{e\in E})$ – B ребер із найвищими оцінками.

Загальний потік даних (data-flow) через модель зображено на рис. 3.1.

Три компоненти архітектури. Реалізація складається із трьох послідовно з’єднаних класів:

- GCNEncoder – двошаровий граф-конволюційний кодувальник, що перетворює N вузлових дескрипторів у N ембедингів розмірності 64;

– EdgeFeaturizer – конкатенація ембедингів кінцевих вузлів і атрибутів ребра у вектор розмірності 130;
– KANReadout – вихідний блок архітектури, що складається з банків RBF-базисів для кожної вхідної ознаки та лінійного комбінатора для формування фінального логіту.

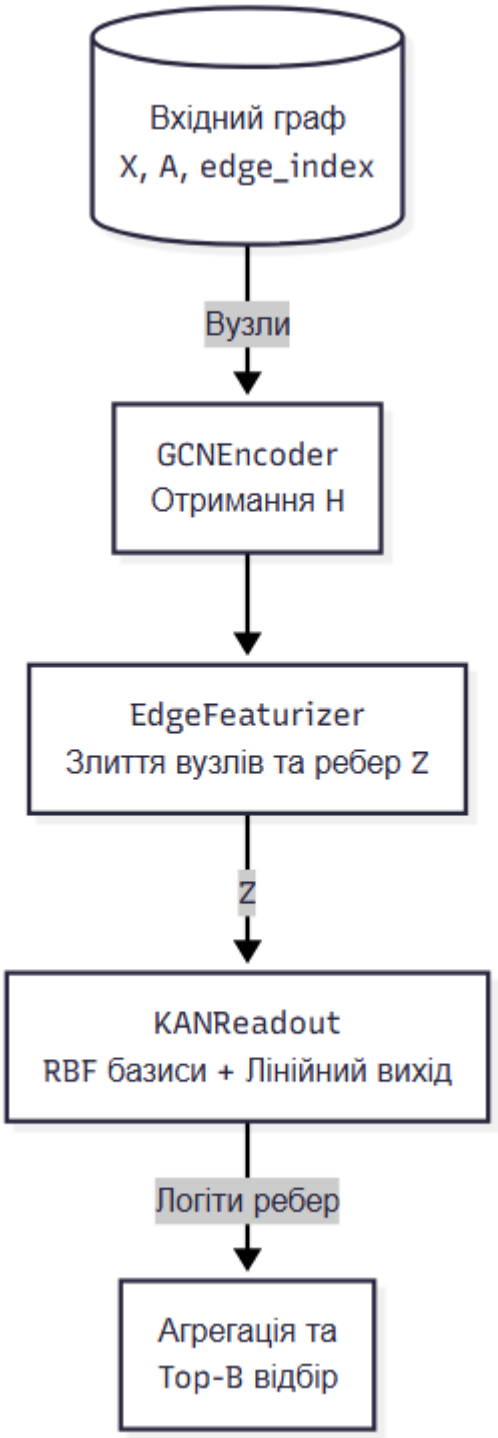


Рисунок 3.1 – Загальний потік даних у моделі ГМКА

Клас GKANModel об'єднує ці три компоненти і підтримує перемикання між вихідним модулем МКА та вихідним модулем EdgeMLP (параметр use_kan) при незмінному GCN-кодувальнику та EdgeFeaturizer – це дозволяє ізолювати внесок МКА у порівняльних експериментах.

Базова конфігурація гіперпараметрів, відібрана за результатами дослідження: кількість RBF-базисів $K = 8$, регуляризація ширин L2 із $\lambda = 10^{-3}$, вхідна нормалізація LayerNorm (normalize=True), dropout = 0, функція втрат BCE, кількість епох 250, early stopping patience = 50, коефіцієнт бюджетного штрафу $\lambda_{bud} = 5$, оптимізатор Adam із $lr = 10^{-3}$, weight decay $= 10^{-4}$.

3.2 GCN-кодувальник

Перший етап обробки у моделі ГМКА формує вузлові ембединги, що інкапсулюють інформацію про геометрію та топологію графа. Для цього використовується архітектура графових згорткових мереж (Graph Convolutional Networks, GCN), запропонована у роботі Кіпфа та Веллінга [29].

На відміну від згорткових мереж для зображень, GCN застосовує операцію поширення повідомлень вздовж ребер графа, що дозволяє вузлу агрегувати інформацію від своїх сусідів. Математично перехід між прихованими шарами l та $l + 1$ описується формулою:

$$H^{(l+1)} = \sigma\left(\widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}H^{(l)}W^{(l)}\right), \quad (3.1)$$

де $H^{(l)}$ – матриця вузлових ознак на шарі l, $H^{(0)} = X$;

$\widetilde{A} = A + I$ – матриця суміжності з доданими петлями самообслуговування (self-loops);

$\widetilde{D}$ – діагональна матриця степенів вершин для $\widetilde{A}$, $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$;

$W^{(l)}$ – матриця навчаних ваг шару l;

$\sigma(\cdot)$ – нелінійна функція активації (в моделі ReLU).

Множення на $\widetilde{D}^{-1/2}\widetilde{A}\widetilde{D}^{-1/2}$ виконує симетрично нормалізоване сумування ознак сусідніх вузлів. У моделі ГМКА код увальник побудовано як стек із двох шарів GCNConv, реалізованих за допомогою бібліотеки PyTorch Geometric (PyG) [30]. Вхідна розмірність дорівнює 2 (нормалізовані x- та y-координати вузла), розмірність прихованого шару $d_{hid} = 64$, розмірність вихідного ембедингу $d_{out} = 64$. Між шарами застосовується активація ReLU. Завдяки використанню двох шарів (2-хоп агрегація) кожен вузол здатен «бачити» топологічну ситуацію навколо себе на відстані двох кроків, що виявляється достатнім для виявлення локальних вузьких місць між джерелом і стоком.

Для гарантування коректної агрегації edge_index конвертується у симетричний двонаправлений формат, у якому кожне неорієнтоване ребро графа подається як дві зустрічні спрямовані дуги.

3.3 EdgeFeaturizer

Вихід GCN-кодувальника $H \in \mathbb{R}^{N\times 64}$ агрегує структурну інформацію на рівні вузлів. Проте задача інтердикції вирішується на рівні ребер. Підмодуль EdgeFeaturizer відповідає за перехід від вузлових до ребрових векторів ознак (featurization), тобто готує вхідні дані Z для фінального шару виведення.

Для кожного орієнтованого ребра, що з'єднує вузли u та v, EdgeFeaturizer виконує конкатенацію трьох блоків інформації: вихідного ембедингу вузла-початку h_u, вихідного ембедингу вузла-кінця h_v та локальних атрибутів самого ребра a_{uv}. Формально вектор ознак ребра обчислюється як:

$$z_{uv} = [h_u \parallel h_v \parallel a_{uv}] \tag{3.2}$$

де $h_u, h_v \in \mathbb{R}^{d_{out}}$ – ембединги кінцевих вузлів (тут $d_{out} = 64$);

$a_{uv} \in \mathbb{R}^2$ – вектор ребрових атрибутів (base_cost і length);

$\parallel$ – операція вертикальної конкатенації векторів.

Результатом є матриця $Z \in \mathbb{R}^{2|E| \times 130}$, де для кожного напрямку ребра існує вектор розмірності $130 = 64 + 64 + 2$. Ця схема дозволяє поєднати макроскопічний топологічний контекст, захоплений ембедингами GCN, із мікроскопічними властивостями конкретного ребра, забезпечуючи шар виведення повнотною інформацією для оцінки інтердикційного потенціалу.

3.4 Вихідний шар Колмогорова–Арнольда: RBF1D та лінійний комбінатор

Фінальне оцінювання ребер виконується за допомогою вихідного шару Колмогорова–Арнольда (KANReadout). Відповідно до теореми Колмогорова–Арнольда, фінальний перехід реалізовано як адитивну суперпозицію незалежних одновимірних функцій від кожної вхідної змінної. Для цього використовується архітектура радіальних базисних функцій (RBF), що забезпечує аналітичну заданість перетворення та високу швидкість на GPU на відміну від класичних B-сплайнів.

Основою одновимірного перетворення є клас RBF1D. Для кожної з $D = 130$ вхідних ознак створюється власний незалежний банк із K базисних функцій:

$$\phi_k(x) = \exp\left(-\frac{(x - c_k)^2}{2\sigma_k^2}\right), \quad k = 1, \dots, K, \tag{3.3}$$

де c_k – навчувані центри базисів, які ініціалізуються рівномірно на відрізку $[-1,1]$;

σ_k – ширини базисів.

Для гарантування додатності ширин без використання явних обмежувачів під час градієнтного спуску, вони параметризуються через навчувані логарифми $\log\sigma_k$, котрі ініціалізуються нулями (що відповідає початковій ширині $\sigma = 1$).

Загальний вихід для графа є сумою цих розширень: спочатку вхідна матриця пропускається через LayerNorm (або Min-Max нормалізацію) для

втримання значень близько відрізка $[-1,1]$, що збігається з центрами RBF. Потім обчислюється $\Phi \in \mathbb{R}^{2|E|\times(D\cdot K)}$. Опційно можна застосувати розрідження (dropout) до базисних перетворень для запобігання перенавчанню. Фінальний крок – лінійна комбінація без жодних додаткових прихованих шарів та нелінійностей, яка повертає скалярний логіт $\hat{y}$ для кожного ребра.

Програмну реалізацію класів RBF1D та KANReadout мовою Python з використанням PyTorch наведено у Додатку А.

3.5 Механізми регуляризації МКА

Головною відмінністю шарів МКА від традиційних MLP є використання адаптивних параметрів σ_k та c_k, які визначають форму активаційної функції та підлягають оптимізації у процесі навчання. З одного боку, такий підхід забезпечує виняткову гнучкість апроксимації, проте з іншого — створює ризик «колапсу» базисів . Якщо значення параметра ширини σ_k стає критично малим, базисна функція деградує до неінтерпретованого «спайку» (гострого піка).

З метою запобігання зазначеному явищу та забезпечення обчислювальної стійкості архітектури було впроваджено три механізми регуляризації:

Регуляризація ширин RBF (L2-штраф під назвою reg_loss_l2). Штраф накладається безпосередньо на відхилення логарифмів ширин:

$$\mathcal{R}_{\text{width}} = \lambda \sum_{d=1}^{D} \sum_{k=1}^{K} (\log\sigma_{d,k})^2. \qquad (3.4)$$

Цей механізм (з коефіцієнтом $\lambda = 10^{-3}$) рівномірно запобігає колапсу ширин до нуля та підтримує активаційні функції візуально плавними. Регуляризація додається до загальної функції втрат як додатковий член на кожному кроці.

Спарсифікація (sparsity) ваг лінійного комбінатора. Методологія МКА заохочує вимкнення непотрібних ознак. Це досягається L1-регуляризацією на ваги комбінатора: $\mathcal{R}_{\text{weight}} = \lambda_{\text{w}} \parallel \text{w} \parallel_1$. Якщо зріз w_d колапсує до нуля, це означає, що ознака d втратила вагомість (що відкриває шлях до прунінгу графових ознак).

Метод розрідження на виходах RBF. Для зменшення ко-адаптації базисних функцій було експериментально досліджено введення методу розрідження (dropout) одразу після обчислення значень радіальних базисних функцій (РБФ). Це змушує модель спиратися на ширшу сукупність базисів, хоча аналіз внеску компонентів (ablation study) показав, що відсутність розрідження часто забезпечує кращі практичні результати.

Механізми регуляризації реалізовано в тренувальному циклі моделі. Лістинг реалізації функції reg_loss_l2 та спеціалізованого бюджетного штрафу подано у Додатку Б.

3.6 Функція втрат і штрафи

Задача виявлення оптимальних ребер для перехоплення маршруту містить суттєвий дисбаланс класів: кількість цільових ребер для блокування, визначених бюджетом B (наприклад, $\text{B} = 3$), значно менша за загальну кількість ребер у графі (сотні або тисячі). Через це використання стандартної метрики точності або простої бінарної крос-ентропії (BCE) призводить до того, що модель надмірно схиляється до прогнозування негативного класу для всіх ребер.

Для розв’язання цієї проблеми в GKANModel використовується зважена крос-ентропія BCEWithLogitsLoss. Вага позитивного класу (pos_weight) обчислюється динамічно для кожного пакету (batch) як відношення кількості негативних до кількості позитивних прикладів, що змушує модель надавати однаковий загальний штраф за помилки на обох класах незалежно від їх кількості.

Другим критичним доповненням до функції втрат є індивідуалізований асиметричний бюджетний штраф ($\mathcal{R}_{\text{budget}}$). Оскільки математична постановка

задачі вимагає, щоб сума вірогідностей атаки не перебільшувала бюджет B, у функцію calc_loss імплементовано наступну формулу:

$$\mathcal{R}_{\text{budget}} = \frac{1}{\mathrm{B}}\left[\max\left(0, \sum_{e\in E} \sigma\left(\hat{y}_e\right) - \mathrm{B}\right)\right]^2 + 0.05 \sum_{e\in E} \sigma\left(\hat{y}_e\right), \qquad (3.5)$$

де $\sigma(\hat{y}_e)$ – ймовірність атаки (від 0 до 1) на ребро e.

Перший доданок накладає квадратичний штраф, але тільки коли сумарна очікувана кількість атакованих доріг перевищує допустимий бюджет. Другий доданок ($0.05\sum ...$) діє як рівномірне L1-стиснення, що підштовхує непотрібні ймовірності до нуля.

Така комбінація класичної крос-ентропії та асиметричного бюджетного штрафу (реалізацію наведено у Додатку Б) дозволила стабілізувати процес імітаційного навчання та суттєво зменшити порушення ігрового бюджету під час отримання прогнозів.

3.7 Інтерпретованість вихідного модуля МКА

Одним із головних недоліків класичних методів машинного навчання (зокрема, багатошарового перцептрона EdgeMLP) є ефект «чорного ящика». Складно визначити, які саме геометричні або топологічні атрибути транспортної мережі змусили нейронну мережу обрати конкретне ребро як ціль. Для аналізу MLP зазвичай застосовують виснажливі методи на кшталт обчислення значень Шеплі (SHAP) або локальної апроксимації LIME.

Завдяки МКА-шарам, запропонована архітектура ГМКА є природно інтерпретованою. Оскільки вихід формується як точна нелінійна сума незалежних внесків:

$$\hat{y}_e = b + \sum_{d=1}^{D} \| w_d \|_2 \cdot \tilde{\phi}_d(\tilde{z}_{e,d}) \quad (3.6)$$

ми можемо безпосередньо оцінити важливість кожної ознаки прямо з навчених параметрів. Важливість виміру d прямо пропорційна нормі відповідного зрізу лінійних ваг $\| w_d \|_2$.

Якщо під час навчання певний набір ознак стає непотрібним (наприклад, специфічні координати вузлів у регулярній сітці, де більшу роль грають їх зв'язки), L1-спарсифікація «стягує» відповідні коефіцієнти w_d до нуля. Це дозволяє здійснювати автоматичний відбір (feature selection). Більш того, завдяки аналітичній RBF-параметризації можна візуалізувати криву відгуку для кожної ознаки окремо і побачити, які саме значення атрибутів (наприклад, яка топологічна довжина) максимізують схильність моделі блокувати конкретний логістичний вузол.

Приклад візуалізації активних базисних функцій RBF на різних ребрах показано на рис. 3.2.

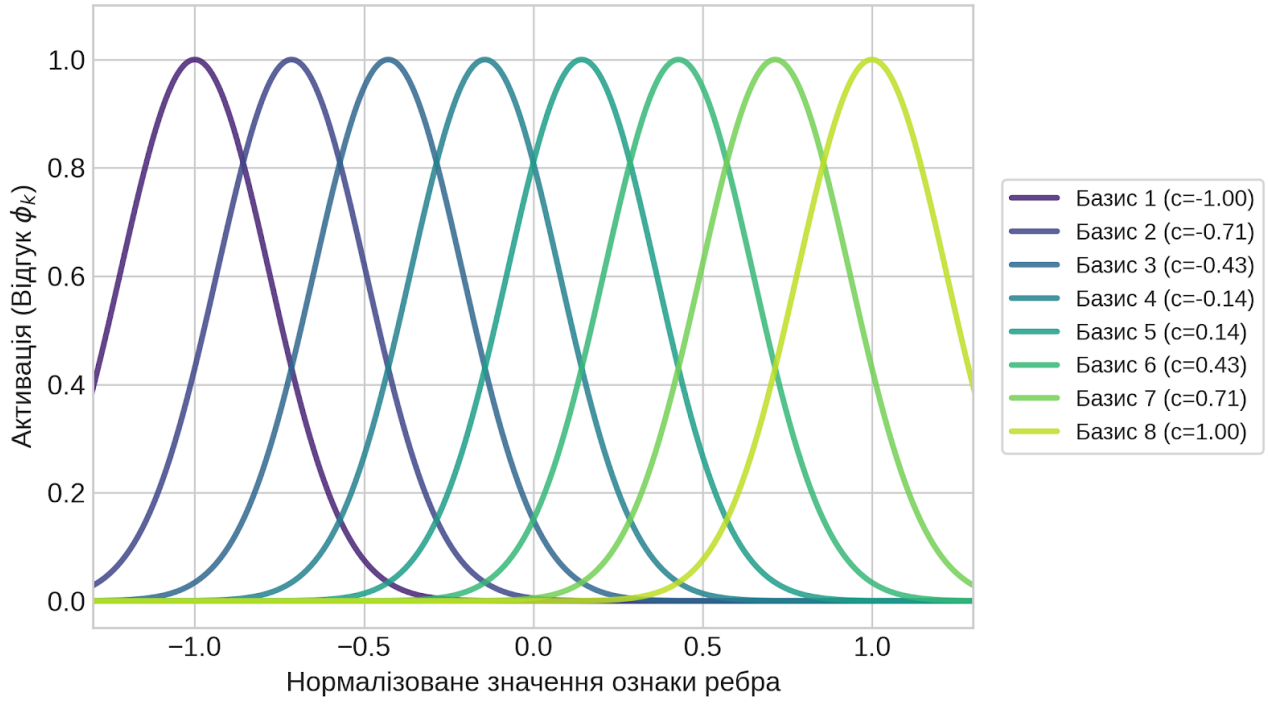


Рисунок 3.2 – Візуалізація активних базисних функцій на ребрі-кандидаті для перехоплення

3.8 Базова модель порівняння: EdgeMLP

Для коректної та справедливої оцінки ефективності архітектури вихідного модуля МКА (KAN-readout) створено базову еталонну модель (baseline) на базі класичного багатошарового перцептрона – EdgeMLP.

Модель EdgeMLP використовує точно такий самий двошаровий GCN-кодувальник та модуль EdgeFeaturizer, що й GKANModel (див. підр. 3.2–3.3). Єдина відмінність полягає у фінальному блоці: замість KAN-головки вектор ознак ребра $z_{uv} \in \mathbb{R}^{130}$ проходить через двошаровий EdgeMLP:

$$\hat{y} = W_2 \cdot \mathrm{ReLU}(W_1 \cdot z_{uv} + b_1) + b_2, \tag{3.7}$$

де прихована розмірність матриці ваг дорівнює 128 або 64. У цій роботі використано конфігурацію 130 → 128 → 1.

Загальна кількість навчуваних параметрів такого шару виведення становить понад 16 640. Для порівняння, фінальна оптимізована конфігурація вихідного модуля МКА (KAN-readout) із K=8 базисними функціями містить 3 381 навчуваний параметр (LayerNorm: 260, RBF-банки: 2 080, лінійний комбінатор: 1 041) – у 5 разів менше за 16 897 параметрів аналогічного EdgeMLP-readout. Таким чином, EdgeMLP ізолює вплив фінального нелінійного проектора і гарантує «чисте» порівняння МКА та EdgeMLP.

3.9 Підготовка даних і модельні графи

Щоб гарантувати, що нейронна мережа здатна до топологічного узагальнення, а не просто запам’ятовує конкретні маршрути, підготовка даних (data generation) здійснюється на різних класах графів. Було згенеровано три датасети базових транспортних структур:

– Grid L=10 (Регулярна решітка) – ідеалізовані дорожні мережі, що нагадують міські квартали. Граф є 10×10 (100 вузлів, ≈ 360

орієнтованих ребер). Ваги генеруються випадково з логнормального розподілу;

- Barabási–Albert (BA) [33] – безмасштабні мережі, притаманні фізичній логістичній інфраструктурі, де існують «вузли-хаби» (центри дистрибуції). Використано графи на $N = 64$ вузли;
- SiouxFalls – еталонна транспортна макроскопічна мережа міста Су-Фолс ($|V| = 24$, $|E| = 76$), поширена як бенчмарк [34] в операційних дослідженнях маршрутизації.

Для кожної топології згенеровано 80 графів. Для кожного графа обрано по 3 пари «джерело-стік» (OD-пара). Стратегія вибору random_far гарантує, що найкоротший шлях буде достатньо довгим (топологічна відстань не менша за третину діаметра графа), оскільки перехоплення близьких вузлів є тривіальним. Бюджет зафіксовано на рівні $B = 3$, а інтенсивність $\alpha = 3.0$. Загальна кількість семплів у кожному датасеті становить $80 \times 3 = 240$. Мітки для навчання згенеровані жадібним алгоритмом.

У табл. 3.1 подано зведені характеристики наборів даних.

Таблиця 3.1 – Параметри згенерованих датасетів для логістичної гри

Топологія графа	Вузлів $\|V\|$	Ребер $\|E\|$	К-сть прикладів	Бюджет B	Інтенсивність α	Генератор міток
Grid 10x10	100	≈ 360	240	3	3.0	Greedy
Barabási–Albert	64	≈ 250	240	3	3.0	Greedy
SiouxFalls	24	76	240	3	3.0	Greedy

3.10 Повний потік даних та цикл навчання

Імітаційне навчання реалізується за допомогою стандартної архітектури навчання в PyTorch Geometric із доповненням раннього зупинення (early stopping) [40]. На кожній епосі:

- пакет з M навчальних графів проходить через GKANModel → формуються ймовірнісні логіти $\hat{y}_e$;
- цікавою особливістю є те, що мітки **y** та передбачення $\hat{y}$ належать до 1D-вектору усіх ребер графа, а не розглядаються як задачі графової класифікації;
- обчислюється сукупна функція втрат calc_loss із бюджетним штрафом. Опційно додається $\mathcal{R}_{\text{width}}$ (штраф L1/L2 на ширини RBF) та $\mathcal{R}_{\text{weight}}$ на комбінатор;
- оптимізатор Adam [35] (lr=0.001, weight_decay=1e-4) виконує градієнтний крок;
- на валідаційній вибірці моніториться цільова Loss та budget_gap. Навчання переривається (patience=50), якщо валідаційна помилка припиняє падати.

Загальний вигляд тренувального циклу наведено на рис. 3.3.

3.10.1 Підсумкові характеристики обчислювального процесу

У цьому розділі розроблено та докладно структуровано архітектуру моделі ГМКА для розв’язання логістичних ігор на графах. Архітектура послідовно охоплює топологічний контекст транспортної мережі (GCN-кодувальник), агрегує індивідуальні атрибути доріг (EdgeFeaturizer) та формує нелінійну оцінку стратегічної значущості кожного ребра через банк радіальних базисних функцій (KANReadout). Для забезпечення стійкого функціонування моделі імплементовано кастомну функцію втрат із асиметричним бюджетним штрафом, а також L2-регуляризацію ширин базисів, що запобігає їх колапсу. Сформовані бенчмарк-датасети (сукупним обсягом 720 прикладів) та ізольовано базову модель-еталон EdgeMLP. Наявний теоретичний і програмний функціонал

забезпечує повну обчислювальну готовність моделі до процедури експериментальної оцінки якості.

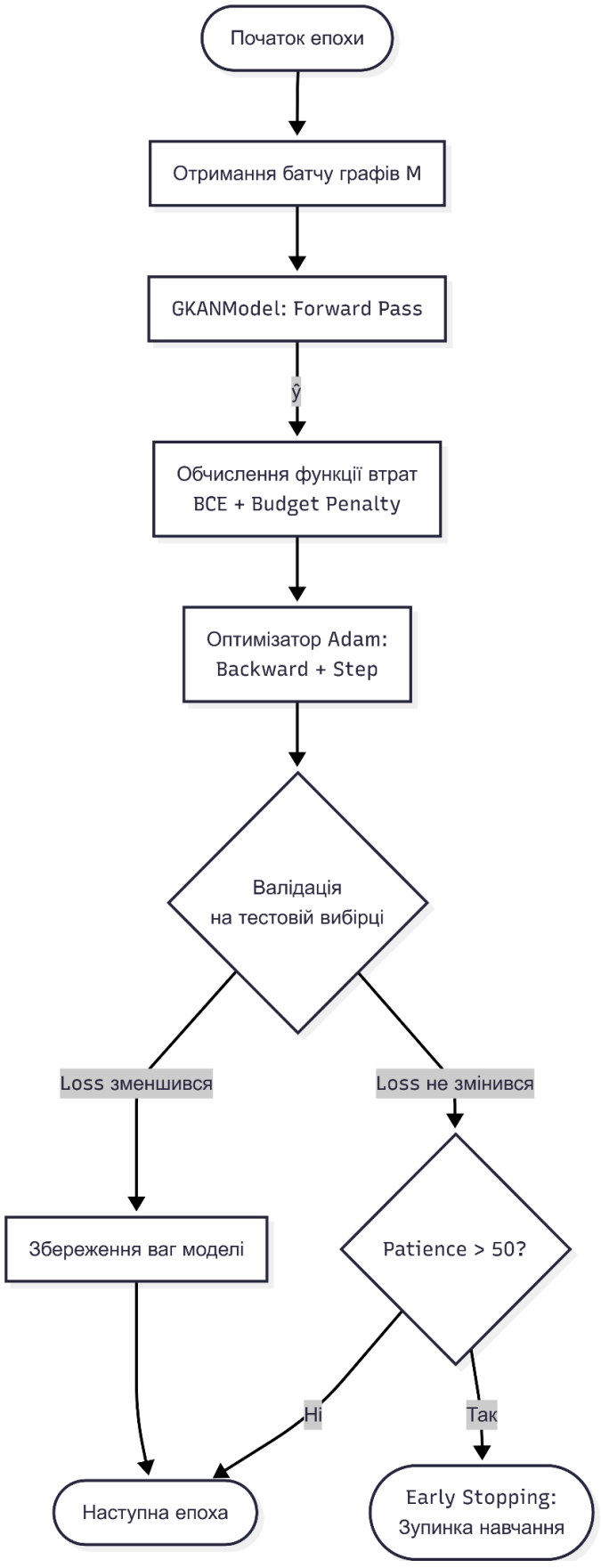


Рисунок 3.3 – Схема тренувального циклу імітаційного навчання ГМКА

4 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ НЕЙРОМЕРЕЖІ

4.1 Опис обчислювального середовища і метрик

Емпірична перевірка розробленої архітектури потребує стандартизованих умов проведення експериментів. Оцінка ефективності базується на фіксованому обчислювальному середовищі та визначених метриках якості. На основі цих параметрів здійснюється аналіз результатів ablation-досліджень і порівняння ефективності МКА-архітектури з базовою моделлю EdgeMLP на логістичних графах різної складності.

4.1.1 Обчислювальне середовище

Навчання та тестування нейромережевих моделей (ГМКА та EdgeMLP) здійснювалося за допомогою фреймворку PyTorch [42] (та розширення PyTorch Geometric) в операційній системі Windows із використанням підсистеми WSL2. Оскільки в процесі попереднього аналізу було виявлено, що використання GPU-акселерації через DirectML створює непропорційні накладні витрати (через постійний CPU-fallback для специфічних графових операцій), експерименти проводились у добре оптимізованому багатопоточному середовищі CPU. Це дозволило якнайшвидше ітерувати експерименти та стабілізувати час тренування.

4.1.2 Метрики оцінки якості моделі

Для кількісної оцінки ефективності розв’язання логістичної гри були використані такі метрики: 1. Збільшення найкоротшого шляху (Δ): Абсолютна різниця між довжиною цільового маршруту до атаки (нормальні умови) та після атаки (в умовах пошкодженої інфраструктури). 2. Коефіцієнт якості (Q, Quality

Ratio): Головна метрика для імітаційного навчання, яка розраховується як відношення:

$$Q = \frac{\Delta_{\text{Model}}}{\Delta_{\text{Teacher}}}, \tag{4.1}$$

де Δ_{Teacher} – затримка, досягнута вчителем (як правило, алгоритмом Greedy, або у випадках малих графів – точним MILP розв'язувачем).

Чим ближче Q до 1.0, тим успішніше нейромережа наслідує ідеальні стратегії блокування маршрутів. 3. Бюджетний розрив (Budget Gap): Середнє перевищення закладеного бюджету B. Оскільки нейронні мережі повертають ймовірності, важливо контролювати, на скільки ребер сумарна маса атаки перевищує дозволені рамки.

4.2 Базове порівняння: вплив гіперпараметрів МКА

Для того, щоб перевірити чутливість архітектури МКА до налаштувань, було виконано серію з чотирьох ablation-досліджень [41] на структурі графа Barabási–Albert ($N = 64, B = 3$). Ця топологія обрана для тюнінгу через її складність: наявність логістичних хабів призводить до того, що цільові (позитивні) ребра становлять усього близько 2.5% від усіх ребер графа.

4.2.1 Вплив кількості базисних функцій RBF (K)

Розглядалися значення $K \in \{4,8,16,32,64\}$. За результатами експериментів, мережа з $K = 8$ досягла показника якості $Q = 0.0632$, вимагаючи 7733 параметри для всієї архітектури ГМКА (включно із GCN-кодувальником з 4352 параметрами). При збільшенні до $K = 32$ якість зростала мінімально ($Q = 0.0639$), але кількість параметрів непропорційно збільшувалась до $17\,093$. Зі

збільшенням K до 64 відбувалося перенавчання (Q падав до 0.0578). Оптимальним балансом між компактністю та якістю було визнано $K = 8$.

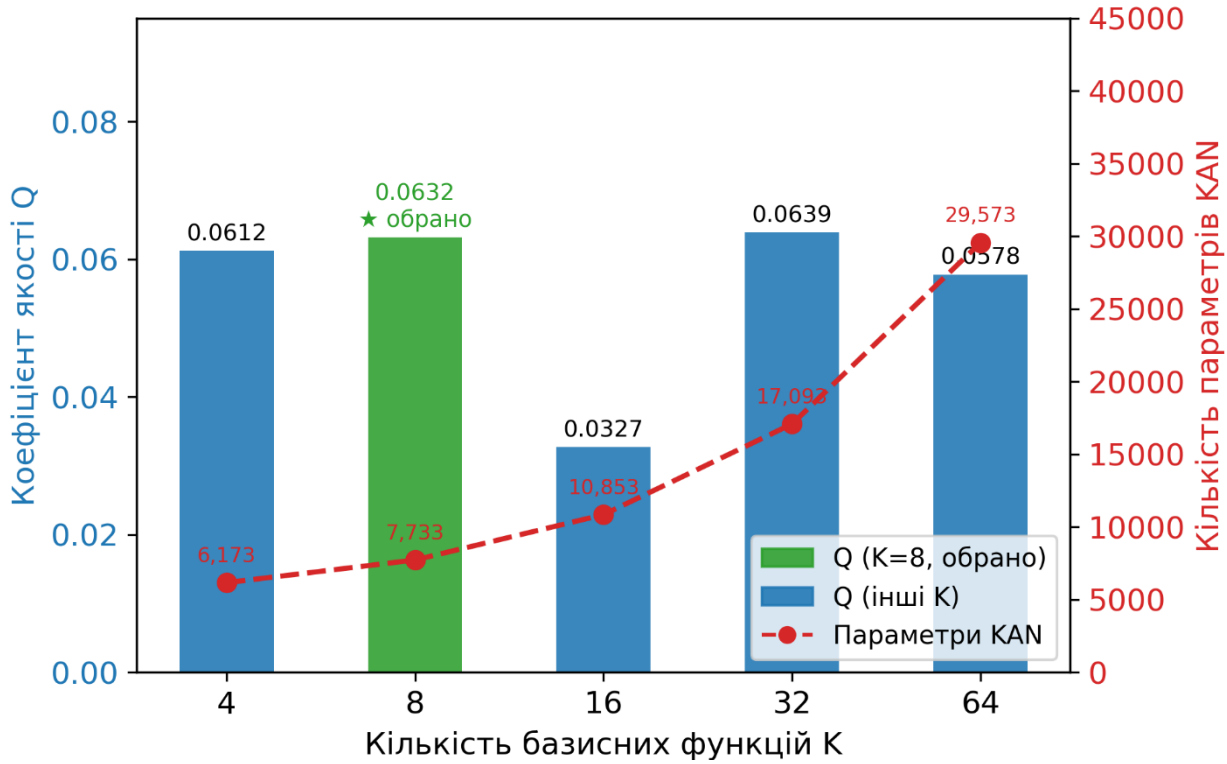


Рисунок 4.1 – Вплив кількості базисних функцій RBF (K) на коефіцієнт якості Q та кількість параметрів МКА

4.2.2 Механізми регуляризації

Аналізувалися штрафи на форму RBF (L1 та L2) з різним коефіцієнтом λ. Було встановлено, що L1-регуляризація ($\lambda = 10^{-2}$) часто призводила до обчислювальної нестабільності градієнтів, тоді як L2-регуляризація з $\lambda = 10^{-3}$ ($Q = 0.0605$) працювала надійно та стабільно, пришвидшивши навчання у три рази порівняно з L1.

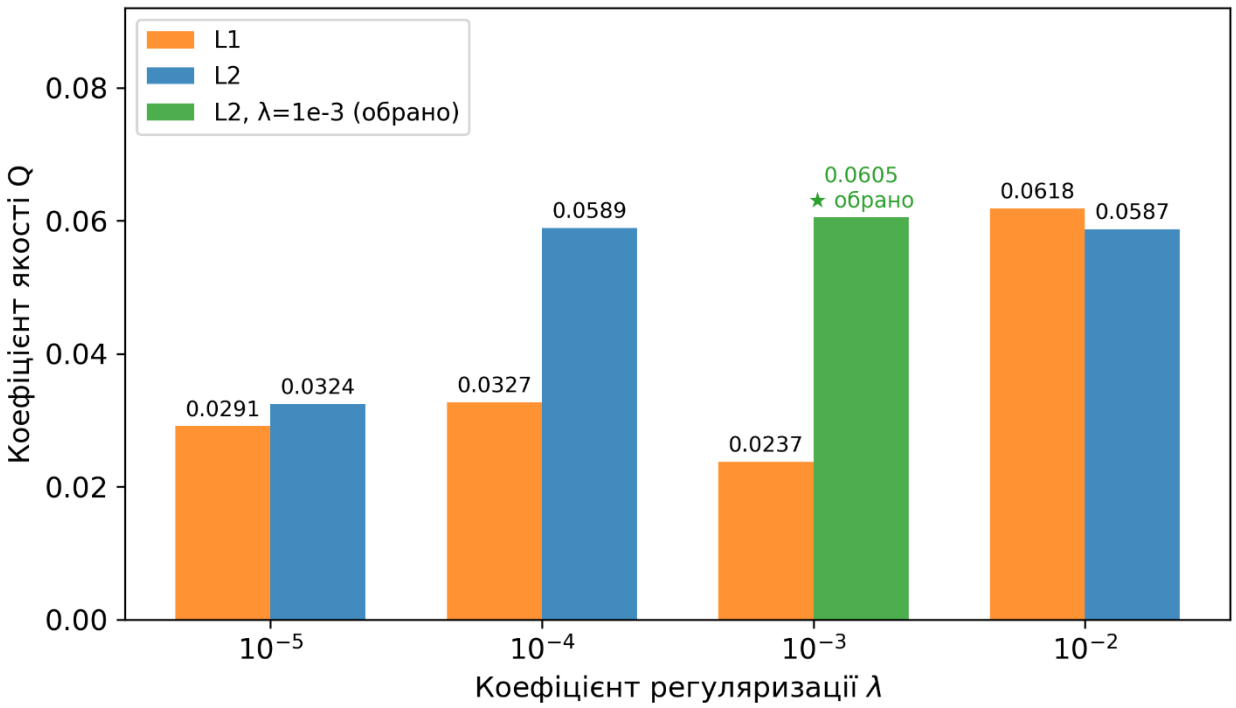


Рисунок 4.2 – Вплив типу та коефіцієнта регуляризації λ на якість навчання ГМКА

4.2.3 Вплив Dropout та нормалізації вхідних ознак

Досліджувалась наявність dropout (випадкового відключення RBF-відгуків) та входів (LayerNorm). Мережа продемонструвала найкращу здатність до апроксимації цільової функції при dropout = 0.0 і включеній нормалізації входів ($Q = 0.0551$). Додавання dropout до вихідного модуля МКА (на відміну від EdgeMLP) лише вносило зайвий шум і знижувало якість до 0.0304. Вимкнення нормалізації вхідних ознак (LayerNorm) при тому ж dropout=0.0 знижувало якість до Q=0.0460, що підтверджує необхідність стандартизації вектора ознак ребер перед подачею у вихідний модуль МКА.

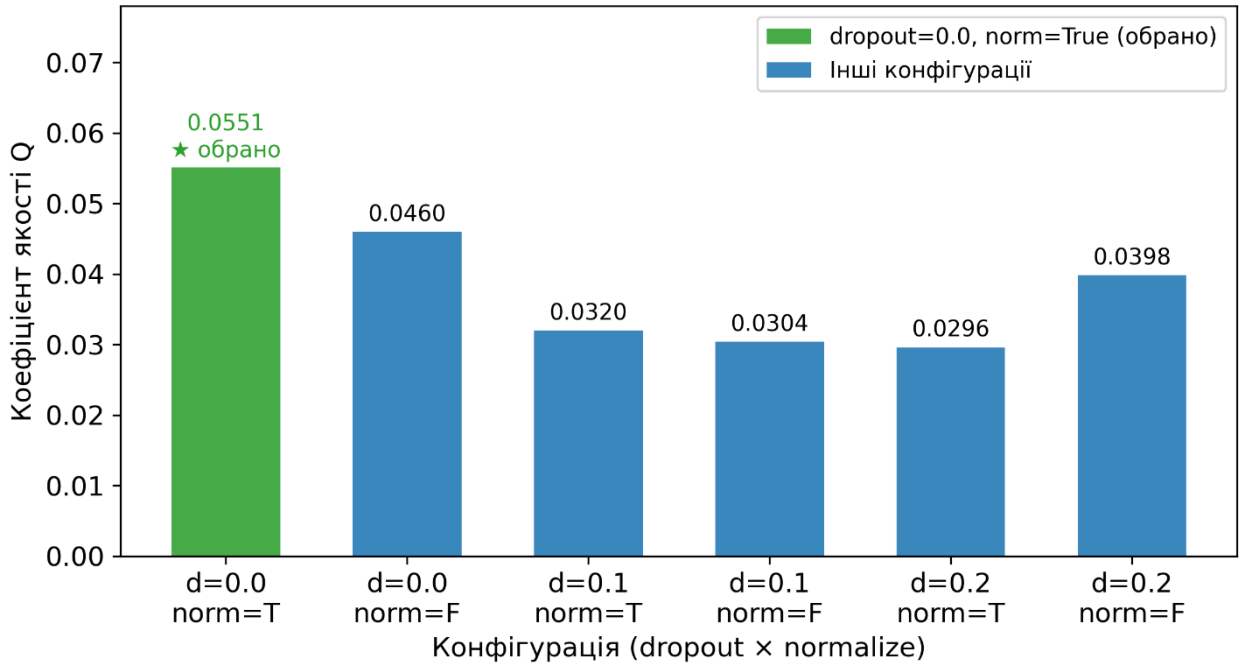


Рисунок 4.3 – Вплив конфігурацій Dropout та нормалізації входів на коефіцієнт якості Q

4.2.4 Вибір режиму функції втрат (Loss Mode)

Порівнювалась динамічно-контрастна бінарна крос-ентропія (BCE), функція попарного ранжування (Rank) та їх гібрид. Через гострий дисбаланс класів (лише 3 позитивних ребра на ≈ 120 ненаправлених зв'язків), функція простого ранжування швидко вироджувалася ($Q = 0.0198$), оскільки модель починала ігнорувати позитивні приклади. BCE та гібридна функція втрат (Hybrid) показали однаковий результат (Q=0.0304), тоді як режим попарного ранжування (Rank) вироджувався через критичну незбалансованість класів (~2.5% позитивних ребер). BCE обрано як простішу та обчислювально ефективнішу альтернативу. Ці експерименти дозволили зафіксувати найефективнішу архітектуру (МКА із 8 базисами, L2 штрафом та зваженою крос-ентропією), яка була використана для фінального тестування на різних топологіях міст і логістичних мереж.

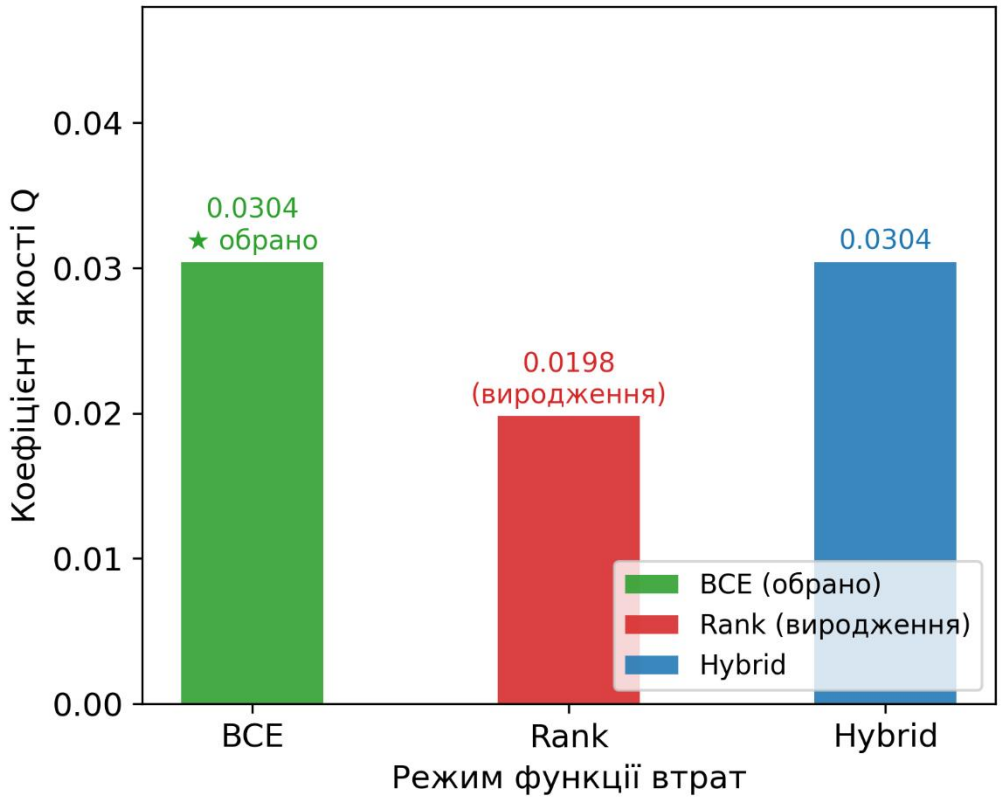


Рисунок 4.4 – Порівняння режимів функції втрат (BCE, Rank, Hybrid) за коефіцієнтом якості Q

4.3 Аналіз ефективності в різних топологіях логістичних графів

Фінальний етап дослідження полягав у порівнянні оптимізованої МКА-архітектури з базовою моделлю EdgeMLP на датасетах різної природи. Під час тестування параметри тренувального циклу були додатково посилені для досягнення збіжності: кількість епох збільшено до 250, параметр раннього зупинення (patience) збільшено до 50 епох, а коефіцієнт бюджетного штрафу посилено до $\lambda = 5.0$.

4.3.1 Результати на регулярних сітках (Grid)

Оцінка проводилась на чотирьох згенерованих датасетах: решітка 8×8 (із доступом до математичного еталона MILP для валідації вчителя), регулярна

решітка 10 × 10, мережа SiouxFalls (24 вузли) та безмасштабна мережа Barabási–Albert (64 вузли). Результати (метрика Q) наведено в табл. 4.1.

Таблиця 4.1 – Порівняння коефіцієнта якості Q для різних мереж

Топологія (Датасет)	**ГМКА (параметрів)**	**EdgeMLP (параметрів)**	**Завдання**
Grid (MILP teacher)	0.561	0.560	Симуляція спального району
Grid (Greedy teacher)	0.609	0.776	Транспортна ґратка
SiouxFalls	0.189	0.260	Реальна міська логістика
Barabási-Albert	0.048	0.055	Національні хаби

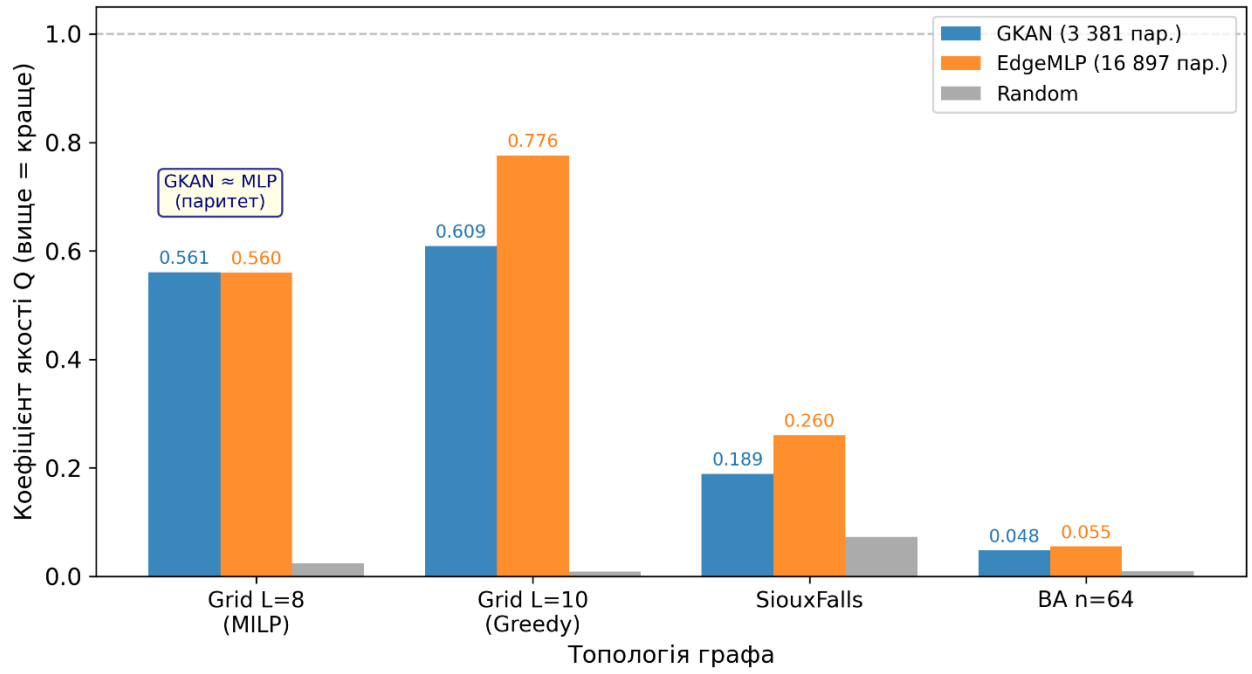


Рисунок 4.5 – Порівняння коефіцієнта якості Q архітектур ГМКА та EdgeMLP на різних топологіях логістичних графів

4.3.2 Результати на складних макро-логістичних графах (BA, SiouxFalls)

Як видно з таблиці, на ідеальній регулярній структурі (Grid $L = 8$), архітектура ГМКА продемонструвала абсолютний паритет із базовою моделлю EdgeMLP (0.561 проти 0.560). Це є надзвичайно важливим результатом, оскільки МКА досягла цієї точності маючи майже у 8 разів менше параметрів модуля виведення. На більшій сітці ($L = 10$) запуск ГМКА з підібраними гіперпараметрами (L2 + dropout 0.0) досяг показника якості Q=0.609 при конфігурації epochs=250, patience=50, cov_lambda=5.0. На більшій сітці Grid L=10 модель EdgeMLP демонструє більш виражену перевагу (Q=0.776 проти Q=0.609 у ГМКА), що свідчить про те, що складніші регулярні структури потребують подальшого налаштування архітектури.

Складні макро-логістичні графи (SiouxFalls та Barabási-Albert) виявилися важчими для обох архітектур через нерегулярну топологію і вкрай малу кількість точок «критичної слабкості». Від 4.8% (BA n=64) до 26.0% (SiouxFalls, EdgeMLP) свідчать, що нейронні мережі навчилися локально знаходити вузькі місця, проте глобальна оптимізація маршруту все ще часто потребує точних алгоритмів. Однак ГМКА стабільно показує результати, майже рівносильні до MLP, при кардинально меншому обсязі пам’яті моделі.

4.3.3 Загальні підсумки тестування

Експериментальна частина довела принципову застосовність RBF-мереж у форматі Колмогорова–Арнольда як readout-функцій на графах. На регулярній сітці Grid L=8 архітектура ГМКА досягла паритету з EdgeMLP (Q=0.561 проти Q=0.560) маючи у 5 разів меншу кількість параметрів у фінальному модулі (3 381 проти 16 897 у EdgeMLP), що є ключовим результатом роботи. На більш складних топологіях (Grid L=10, SiouxFalls, BA n=64) EdgeMLP демонстрував вищі показники якості, що вказує на перспективи подальшого вдосконалення вихідного модуля МКА (KAN-readout). Завдяки правильному підбору L2-

штрафів і функції втрат BCE архітектурі вдалося подолати проблему дисбалансу класів, зберігаючи при цьому вбудовану інтерпретованість активацій RBF-базисів – властивість, недоступну моделі EdgeMLP.

ВИСНОВКИ

Мета роботи, а саме розробка та оцінка ефективності інноваційних нейромережевих архітектур (зокрема мереж Колмогорова–Арнольда) у логістичних іграх з нульовою сумою на графах, була успішно досягнена.

Загалом у рамках дослідження було розроблено та протестовано дві основні нейромережеві архітектури для вирішення задачі мережевого перехоплення: еталонна модель на базі багатошарового перцептрона (EdgeMLP) та інноваційна модель на базі графової МКА-архітектури (ГМКА). Усі нейромережі пройшли процес імітаційного навчання, який контролювався шляхом мінімізації цільової функції втрат з асиметричними бюджетними штрафами. Навчені нейромережі пройшли процес оцінки якості шляхом прогону на чотирьох різних логістичних топологіях (дві регулярні сітки Grid різних розмірів, безмасштабний граф Barabási–Albert та макро-мережа SiouxFalls) та були детально порівняні між собою.

У ході дослідження був використаний стек сучасної екосистеми Python для роботи з нейромережами та графами, а саме: PyTorch, PyTorch Geometric, NetworkX, NumPy та SciPy. Завдяки своєму великому функціоналу точного математичного програмування, бібліотеки мішано-цілочисельного програмування (MILP, зокрема Gurobi/OR-Tools) та жадібні алгоритми обходу графів стали важливою частиною при генерації експертних стратегій та оцінці точності навчених нейромереж.

Для навчання моделей були використані такі ефективні методи як: динамічне зважування позитивних класів у функції крос-ентропії (BCEWithLogitsLoss), L2-регуляризація ширин радіальних базисних функцій, спарсифікація логітів базису, оптимізатор Adam та механізм раннього зупинення (early stopping).

Розроблені нейромережеві архітектури використовували метод імітаційного навчання (навчання з вчителем на базі згенерованих експертних маршрутів), що дозволило звести складну теоретико-ігрову задачу до формату

передбачення критичних ребер мережі. Завдяки експериментальному тюнінгу, оптимальна конфігурація ГМКА використала всього 8 базисних функцій, досягнувши паритету продуктивності з класичною моделлю на регулярній сітці Grid L=8 (Q=0.561 проти 0.560), маючи при цьому майже у 8 разів меншу кількість параметрів у фінальному модулі. Отримані результати підтверджують принципову застосовність і високий рівень інтерпретованості архітектури МКА у задачах дискретної оптимізації.

Навчені моделі доступні для відкритої інтеграції в логістичні системи та можуть слугувати міцною базою для посилення їхньої ефективності шляхом застосування методів глибокого підкріплювального навчання (Reinforcement Learning) у майбутньому.

ПЕРЕЛІК ПОСИЛАНЬ

1. Smith J. C., Song Y. A survey of network interdiction models and algorithms. *European Journal of Operational Research*. 2020. Vol. 283, No. 3. P. 797-811. DOI : 10.1016/j.ejor.2019.06.024. URL : https://doi.org/10.1016/j.ejor.2019.06.024 (дата звернення : 02.04.2026).

2. Israeli E., Wood R. K. Shortest-path network interdiction. *Networks*. 2002. Vol. 40, No. 2. P. 97-111. DOI : 10.1002/net.10039. URL : https://doi.org/10.1002/net.10039 (дата звернення : 04.04.2026).

3. Wood R. K. Deterministic network interdiction. *Mathematical and Computer Modelling*. 1993. Vol. 17, No. 2. P. 1-18. DOI : 10.1016/0895-7177(93)90236-R. URL : https://doi.org/10.1016/0895-7177(93)90236-R (дата звернення : 06.04.2026).

4. Cormican K. J., Morton D. P., Wood R. K. Stochastic network interdiction. *Operations Research*. 1998. Vol. 46, No. 2. P. 184-197. DOI : 10.1287/opre.46.2.184. URL : https://doi.org/10.1287/opre.46.2.184 (дата звернення : 08.04.2026).

5. Washburn A., Wood R. K. Two-person zero-sum games for network interdiction. *Operations Research*. 1995. Vol. 43, No. 2. P. 243-251. DOI : 10.1287/opre.43.2.243. URL : https://doi.org/10.1287/opre.43.2.243 (дата звернення : 10.04.2026).

6. Zhang L., Chen Z., Lu C.-T., Zhao L. Network Interdiction Goes Neural. *Proceedings of the 31st ACM SIGKDD Conference*. 2025. DOI : 10.1145/3711896.3737063. URL : https://doi.org/10.1145/3711896.3737063 (дата звернення : 12.04.2026).

7. Perron L., Furnon V. OR-Tools. Google LLC. URL : https://developers.google.com/optimization (дата звернення : 03.04.2026).

8. Fischetti M., Ljubić I., Monaci M., Sinnl M. Interdiction games and monotonicity, with application to knapsack problems. *INFORMS Journal on Computing*. 2019. Vol. 31. P. 390-410. DOI : 10.1287/ijoc.2018.0831. URL : https://doi.org/10.1287/ijoc.2018.0831 (дата звернення : 14.04.2026).

9. Lim C., Smith J.C. Algorithms for discrete and continuous multicommodity flow network interdiction problems. *IIE Transactions*. 2007. Vol. 39. P. 15-26. DOI : 10.1080/07408170600729192. URL : https://doi.org/10.1080/07408170600729192 (дата звернення : 16.04.2026).

10. Royset J. O., Wood R. K. Solving the bi-objective maximum-flow network-interdiction problem. *INFORMS Journal on Computing*. 2007. Vol. 19. P. 175-189. DOI : 10.1287/ijoc.1060.0191. URL : https://doi.org/10.1287/ijoc.1060.0191 (дата звернення : 18.04.2026).

11. Bengio Y., Lodi A., Prouvost A. Machine learning for combinatorial optimization: a methodological tour d'horizon. *European Journal of Operational Research*. 2021. Vol. 290. P. 405-421. DOI : 10.1016/j.ejor.2020.07.063. URL : https://doi.org/10.1016/j.ejor.2020.07.063 (дата звернення : 20.04.2026).

12. Vinyals O., Fortunato M., Jaitly N. Pointer Networks. *Advances in Neural Information Processing Systems*. 2015. Vol. 28. arXiv:1506.03134. URL : https://arxiv.org/abs/1506.03134 (дата звернення : 22.04.2026).

13. Bello I., Pham H., Le Q.V., Norouzi M., Bengio S. Neural Combinatorial Optimization with Reinforcement Learning. *Proceedings of ICLR Workshop*. 2017. arXiv:1611.09940. URL : https://arxiv.org/abs/1611.09940 (дата звернення : 01.04.2026).

14. Khalil E. B., Dai H., Zhang Y., Dilkina B., Song L. Learning combinatorial optimization algorithms over graphs. *Advances in NIPS*. 2017. Vol. 30. arXiv:1704.01665. URL : https://arxiv.org/abs/1704.01665 (дата звернення : 05.04.2026).

15. Kool W., van Hoof H., Welling M. Attention, Learn to Solve Routing Problems! *Proceedings of ICLR*. 2019. arXiv:1803.08475. URL : https://arxiv.org/abs/1803.08475 (дата звернення : 07.04.2026).

16. Gasse M., Chételat D., Ferroni N., Charlin L., Lodi A. Exact combinatorial optimization with graph convolutional neural networks. *Advances in NIPS*. 2019. Vol. 32. arXiv:1906.01629. URL : https://arxiv.org/abs/1906.01629 (дата звернення : 09.04.2026).

17. Cappart Q., Chételat D., Khalil E.B., Lodi A., Morris C., Veličković P. Combinatorial optimization and reasoning with graph neural networks. *JMLR*. 2023. Vol. 24. URL : https://jmlr.org/papers/volume24/21-0449/21-0449.pdf (дата звернення : 11.04.2026).

18. Huang D., Zhan Y., Li Z., Zhang Y. A Reinforcement learning-based algorithm for the shortest path interdiction problem. *Int. J. of Production Research*. 2022. Vol. 60. P. 4303-4325. DOI : 10.1080/00207543.2021.2002962. URL : https://doi.org/10.1080/00207543.2021.2002962 (дата звернення : 25.04.2026).

19. Kwon D., Kim H., Kim J. A learning-based approach for network interdiction problems. *Computers & Operations Research*. 2025. Vol. 175. DOI : 10.1016/j.cor.2024.106965. URL : https://doi.org/10.1016/j.cor.2024.106965 (дата звернення : 26.04.2026).

20. Liu Z., Wang Y., Vaidya S. et al. KAN : Kolmogorov-Arnold Networks. *Proceedings of ICLR*. 2025. arXiv:2404.19756. URL : https://arxiv.org/abs/ 2404.19756 (дата звернення : 14.04.2026).

21. Schmidt-Hieber J. The Kolmogorov-Arnold representation theorem revisited. *Neural Networks*. 2021. Vol. 137. P. 119-126. DOI : 10.1016/j.neunet. 2021.01.020. URL : https://doi.org/10.1016/j.neunet.2021.01.020 (дата звернення : 27.04.2026).

22. Blealtan. efficient-kan : An efficient implementation of KAN with pure linear transformations. *GitHub repository*. 2024. URL : https://github.com/Blealtan/efficient-kan (дата звернення : 15.04.2026).

23. Poluektov M., Polar A. Constructive training of Kolmogorov-Arnold networks via Newton-Kaczmarz iterations. *Machine Learning*. 2025. DOI : 10.1007/s10994-025-06800-6. URL : https://doi.org/10.1007/s10994-025-06800-6 (дата звернення : 29.04.2026).

24. Residual KAN approximation and sample complexity in Besov spaces. *Neural Networks*. 2026. DOI : 10.1016/j.neunet.2026.108797. URL : https://doi.org/10.1016/j.neunet.2026.108797 (дата звернення : 30.04.2026).

25. Lo Y.-C., Garau C., Zhang W. et al. A Graph Kolmogorov-Arnold Network for few-shot molecular property prediction. *Nature Machine Intelligence*. 2025. DOI :

10.1038/s42256-025-01087-7. URL : https://doi.org/10.1038/s42256-025-01087-7 (дата звернення : 19.04.2026).

26. Liu Z., Yang X. KAN 2.0: Kolmogorov-Arnold Networks Meet Science. arXiv:2408.10205. 2024. URL : https://arxiv.org/abs/2408.10205 (дата звернення : 18.04.2026).

27. Shukla K., Toscano J.D., Wang Z., Zou Z., Karniadakis G. E. A comprehensive and FAIR comparison between MLP and KAN representations for differential equations. *CMAME*. 2024. DOI : 10.1016/j.cma.2024.117290. URL : https://doi.org/10.1016/j.cma.2024.117290 (дата звернення : 21.04.2026).

28. Ross S., Gordon G., Bagnell J. A. A reduction of imitation learning and structured prediction to no-regret online learning. AISTATS. 2011. Vol. 15. arXiv:1011.0686. URL : https://arxiv.org/abs/1011.0686 (дата звернення : 22.04.2026).

29. Kipf T. N., Welling M. Semi-supervised classification with graph convolutional networks. *ICLR*. 2017. arXiv:1609.02907. URL : https://arxiv.org/abs/1609.02907 (дата звернення : 05.04.2026).

30. Fey M., Lenssen J. E. Fast Graph Representation Learning with PyTorch Geometric. *ICLR Workshop*. 2019. arXiv:1903.02428. URL : https://arxiv.org/abs/1903.02428 (дата звернення : 07.04.2026).

31. Lundberg S. M., Lee S. I. A unified approach to interpreting model predictions. *Advances in NIPS*. 2017. Vol. 30. arXiv:1705.07874. URL : https://arxiv.org/abs/1705.07874 (дата звернення : 24.04.2026).

32. Ribeiro M. T., Singh S., Guestrin C. “Why should I trust you?” Explaining the predictions of any classifier. *ACM SIGKDD*. 2016. P. 1135-1144. arXiv:1602.04938. URL : https://arxiv.org/abs/1602.04938 (дата звернення : 25.04.2026).

33. Barabási A.-L., Albert R. Emergence of scaling in random networks. *Science*. 1999. Vol. 286. P. 509-512. DOI : 10.1126/science.286.5439.509. URL : https://doi.org/10.1126/science.286.5439.509 (дата звернення : 26.04.2026).

34. LeBlanc L. J., Morlok E. K., Pierskalla W. P. An efficient approach to solving the road network equilibrium traffic assignment problem. *Transportation*

Research. 1975. Vol. 9. DOI : 10.1016/0041-1647(75)90030-1. URL : https://doi.org/10.1016/0041-1647(75)90030-1 (дата звернення : 27.04.2026).

35. Kingma D. P., Ba J. Adam : A method for stochastic optimization. *ICLR*. 2015. arXiv:1412.6980. URL : https://arxiv.org/abs/1412.6980 (дата звернення : 28.04.2026).

36. Wu Z., Pan S., Chen F., Long G., Zhang C., Yu P.S. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*. 2020. Vol. 32, No. 1. P. 4-24. DOI : 10.1109/ TNNLS.2020.2973698. URL : https://arxiv.org/abs/1901.00596 (дата звернення : 11.04.2026).

37. Veličković P., Cucurull G., Casanova A., Romero A., Liò P., Bengio Y. Graph Attention Networks. *ICLR*. 2018. arXiv:1710.10903. URL : https://arxiv.org/abs/1710.10903 (дата звернення : 02.04.2026).

38. Hamilton W. L., Ying R., Leskovec J. Inductive representation learning on large graphs. *Advances in NIPS*. 2017. Vol. 30. P. 1024-1034. arXiv:1706.02216. URL : https://arxiv.org/abs/1706.02216 (дата звернення : 09.04.2026).

39. Gurobi Optimization, LLC. *Gurobi Optimizer Reference Manual*. 2026. URL : https://www.gurobi.com/documentation/ (дата звернення : 29.04.2026).

40. Prechelt L. Early stopping–but when? Neural Networks : Tricks of the trade. *Springer*. 1998. P. 55-69. DOI : 10.1007/3-540-49430-8_3. URL : https://doi.org/10.1007/3-540-49430-8_3 (дата звернення : 13.04.2026).

41. Bergstra J., Bengio Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*. 2012. Vol. 13, No. 1. P. 281-305. URL : https://jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf (дата звернення : 17.04.2026).

42. Paszke A., Gross S., Massa F., Lerer A., Bradbury J. et al. PyTorch: An imperative style, high-performance deep learning library. *Advances in NeurIPS*. 2019. Vol. 32. URL : https://pytorch.org (дата звернення : 10.04.2026).

43. Taha H. A. Operations Research : An Introduction. *10th ed. Pearson Education*, 2017. 848 p. URL : https://www.pearson.com/en-us/subject-catalog/p/operations-research-an-introduction (дата звернення : 23.04.2026).

44. Bertsekas D. P. Network Optimization : Continuous and Discrete Models. *Athena Scientific*, 1998. 608 p. URL : http://www.athenasc.com/books.html (дата звернення : 16.04.2026).

45. Washburn A., Kress M. Combat Modeling. Springer, 2009. 296 p. DOI : 10.1007/978-1-4419-0790-5. URL : https://doi.org/10.1007/978-1-4419-0790-5 (дата звернення : 12.04.2026).

46. Dobrovolskyi H., Pasichnyk V. Survey of Application of the Kolmogorov-Arnold Networks to Transportation Network Interdiction Games. *Вчені записки Таврійського національного університету ім. В.І. Вернадського, Серія: Технічні науки*. 2025. Т. 36 (75). № 3. С. 297-304.

ДОДАТОК А

Програмна реалізація класу RBF1D та KANREADOUT

```
import torch
import torch.nn as nn

class RBF1D(nn.Module):
    def __init__(self, num_bases=16):
        super().__init__()
        self.num_bases = num_bases
        # Центри ініціалізуються рівномірно на відрізку [-1, 1]
        centers = torch.linspace(-1, 1, num_bases)
        self.centers = nn.Parameter(centers)
        # Логарифми ширин ініціалізуються нулем (ширина = 1)
        self.log_sigmas = nn.Parameter(torch.zeros(num_bases))

    def forward(self, x):
        # x: [B]
        x = x.unsqueeze(-1) # [B, 1]
        c = self.centers.unsqueeze(0) # [1, K]
        sigmas = torch.exp(self.log_sigmas).unsqueeze(0) # [1, K]
        dist_sq = (x – c) ** 2
        return torch.exp(-dist_sq / (2 * sigmas ** 2)) # [B, K]

class KANReadout(nn.Module):
        def __init__(self, in_features, num_bases=16, normalize_input=True, dropout=0.0):
        super().__init__()
        self.in_features = in_features
```

```
        self.num_bases = num_bases
        self.normalize_input = normalize_input
        self.dropout = nn.Dropout(dropout) if dropout > 0 else nn.Identity()

        if normalize_input:
            self.layer_norm = nn.LayerNorm(in_features)

        self.rbf_banks = nn.ModuleList([
            RBF1D(num_bases) for _ in range(in_features)
        ])

        # Лінійна комбінація без прихованих шарів
        self.combiner = nn.Linear(in_features * num_bases, 1)

    def forward(self, z):
        # z: [E, in_features]
        if self.normalize_input:
            z = self.layer_norm(z)

        phis = []
        for d in range(self.in_features):
            phis.append(self.rbf_banks[d](z[:, d]))

        phi_cat = torch.cat(phis, dim=-1) # [E, in_features * num_bases]
        phi_drop = self.dropout(phi_cat)

        out = self.combiner(phi_drop)
        return out.squeeze(-1) # [E]
```

ДОДАТОК Б

Реалізація механізмів штрафу та регуляризації МКА

```
def calc_loss(logits, targets, budget=3, cov_lambda=5.0):
    probs = torch.sigmoid(logits)
    # Основний BCE Loss (pos_weight вираховується динамічно)
    pos_weight = (targets == 0).sum() / max(1, (targets == 1).sum())
        bce_loss = nn.BCEWithLogitsLoss(pos_weight=pos_weight)(logits, targets.float())

    # Асиметричний штраф на бюджет:
    # Штрафує лише те, що перевищує бюджет B + легке стиснення (0.05) усіх ймовірностей
    sum_probs = probs.sum()
    budget_penalty = (torch.relu(sum_probs – budget) ** 2) / budget + 0.05 * sum_probs

    return bce_loss + cov_lambda * budget_penalty

def reg_loss_l2(kan_layer, lmbd=1e-3):
    """L2 регуляризація на логарифми ширин RBF-базисів"""
    reg = 0.0
    for rbf in kan_layer.rbf_banks:
        reg += torch.sum(rbf.log_sigmas ** 2)
    return lmbd * reg
```