УДК 004.72:004.896:519.85 **DOI:** 10.31673/2412-9070.2026.059703

Ю. Л. ВЛАДАРЧИК, аспірант;
ORCID: 0009-0008-5271-6285
К. С. НЕСТЕРЕНКО, д-р техн. наук, професор,
ORCID: 0000-0001-7672-7386
Державний університет інформаційно-комунікаційних технологій, Київ

ДЕЦЕНТРАЛІЗОВАНА БАГАТОКРИТЕРІАЛЬНА ОПТИМІЗАЦІЯ ТРАФІКУ SRv6 АВТОНОМНИМ AI-АГЕНТОМ НА БАЗІ FRRouting: РЕАЛІЗАЦІЯ ТА ЕКСПЕРИМЕНТАЛЬНА ВАЛІДАЦІЯ

Попередня концептуальна публікація авторів запропонувала архітектуру автономного AI-агента для багатокритеріальної оптимізації (БО) трафіку SRv6 без зовнішнього контролера, обмеживши математичною моделлю без емпіричної реалізації. Дана робота вперше переходить від концепту до повної експериментальної валідації на робочому стенді (FRRouting 10.x, Containerlab, Linux 6.17) та забезпечує кількісне підтвердження запропонованого підходу. Запропоновано метод децентралізованої БО-маршрутизації SRv6, у якому узгодженість рішень незалежних агентів на граничних маршрутизаторах досягається через імпліцитну стигмергію (спільне читання синхронізованої IS-IS LSDB як стигмергічного середовища, формалізованого у п'ятірці Heylighen) без центрального арбітра вагових коефіцієнтів і без прямого міжагентного обміну повідомленнями; стабільність рішень підтверджена експериментальною процедурою з п'яти послідовних фаз — єдине переключення в кожному напрямку, нуль хибних спрацювань у трьох baseline-фазах ($p < 10^{-3}$ проти H_0 випадкового переключення). Оптимізацію за чотирма QoS-критеріями (затримка, пропускна здатність, втрати, джитер) і топологічну узагальнюваність поєднано в єдиній моделі GNN-PPO (GCNConv ×2, 50 179 параметрів); симетричний бар'єр TRIGGER/WITHDRAW реалізовано без жорстко-кодового порогу через збереження тренувального розподілу ознак на вході моделі (ЗТР): та сама нейромережа з границею прийняття рішення SLA = 40 мс керує переключеннями в обох напрямках. Виробниче розгортання БО-агентів на edge-маршрутизаторі забезпечують ONNX Runtime з INT8-квантуванням (≈8 МБ залежностей, ≈0,13 мс інференсу) разом з eBPF XDP/TC телеметрією ядра (1–2 мс path-RTT гістограми на цикл), що утримує повний PyTorch-стек тренування (≈700 МБ, ≈50 мс) виключно у тренувальному контурі. Сумарний час реакції агента — TRIGGER 409 мс, WITHDRAW 825 мс — структурно зумовлений K-of-N гістерезисом і частотою eBPF-зондів, а не вартістю AI-компонента, на яку припадає менше 0,1 % від загальної реакції.

Ключові слова: SRv6, GNN-PPO, ONNX INT8, eBPF XDP/TC, багатокритеріальна оптимізація, FRRouting, імпліцитна стигмергія, граничний AI.

Вступ

Управління трафіком у мережах 5G та IoT з різнорідними угодами про рівень обслуговування (SLA) для URLLC, eMBB та mMTC одночасно вимагає оптимізації маршрутів за чотирма критеріями якості обслуговування (QoS) (затримка, пропускна здатність, втрати пакетів, джитер). Така багатокритеріальна оптимізація (БО) у динамічному середовищі належить до класу NP-складних задач і у сучасній літературі вирішується переважно через централізовані SDN/MEC-контролери, які обчислюють вагову згортку критеріїв або Парето-фронт і розповсюджують маршрутні політики на пристрої виконання. Така архітектура успадковує три структурні обмеження: єдину точку відмови контролера, керуючий канал як додатковий ресурс мережі та реакцію, обмежену часом обходу до арбітра й назад.

© Владарчик Ю. Л., Нестеренко К. С., 2026

Чому саме SRv6 (Segment Routing over IPv6). Перехід до децентралізованої БО-архітектури методологічно стає можливим тільки при наявності площини пересилання з трьома специфічними властивостями: 1) програмованості — кодування довільних маршрутних рішень у заголовках пакетів через стек SID-ів, без зміни конфігурації проміжних маршрутизаторів; 2) локального контролю на джерелі (source routing) — повного визначення прямого шляху лише з боку вузла-джерела, без потреби у двосторонніх переговорах; 3) однонаправленості шляху per-direction — прямий шлях від CE_1 до CE_2 програмується незалежно від зворотного, отже агент на одному граничному маршрутизаторі впливає тільки на трафік, що ним породжується або термінується. SRv6 як спадкоємець IPv6 із SID-стеком як native IPv6-розширенням задовольняє усі три властивості і є на сьогодні єдиним відкритим стандартом, що це робить (MPLS-TE й IP-RSVP лишаються control-plane-важкими, RIB-залежними і не мають source-routed одностороннього контролю). Саме однонапрямна семантика SRv6 (програмування шляху per-direction) робить локальне рішення кожного граничного агента самодостатнім: для прямої траєкторії потрібно й достатньо рішення лише CE_1-сторонього агента; симетричність двонапрямних потоків, якщо потрібна, забезпечується ідентичним агентом на дзеркальному граничному вузлі.

Постановка проблеми/задачі

Не зважаючи на існування окремих компонентів вирішення (маршрутизація на основі глибокого навчання з підкріпленням (DRL), моделювання мережі графовою нейронною мережею (GNN), INT8-квантування, телеметрія ядра на основі eBPF, extended Berkeley Packet Filter), у відкритій літературі відсутній опис повністю реалізованої та кількісно валідованої децентралі-зованої багатокритеріальної системи штучного інтелекту (AI) маршрутизації SRv6 на виробничому ПЗ маршрутизації без зовнішнього контролера. Конкретно — відсутня система, яка одночасно: розв'язує MCO-задачу (Multi-Criteria Optimization) за чотирма критеріями (затримка, пропускна здатність, втрати, джитер) безпосередньо на граничному маршрутизаторі; досягає узгодженості рішень незалежних агентів без центрального арбітра вагових коефіцієнтів і без прямого міжагентного обміну повідомленнями; зберігає топологічну узагальнюваність моделі поза рамками тренувальної топології; забезпечує стійкість симетричного бар'єру TRIGGER/WITHDRAW без жорстко-кодових порогів у вирішальному шляху агента; функціонує у субмілісекундному бюджеті інференсу на CPU площини управління edge-маршрутизатора з обмеженими обчислювальними ресурсами. Задача даної роботи — закрити цю прогалину шляхом експериментальної валідації єдиного інтегрованого агента на робочому стенді у конфігурації *методологічно найгіршого випадку* (FRRouting без BGP-LS і без SRv6 TE Policy) з кількісним підтвердженням запропонованого підходу.

Аналіз останніх досліджень і публікацій

Найближчий до даної роботи клас методів — DRL-маршрутизація з багатокритеріальною винагородою [2], реалізована поверх SDN-контролера або 5G-MEC оркестратора. Спільна архітектурна ознака цих робіт — централізований арбітр, що обчислює вагову згортку критеріїв або Парето-фронт, та обмін телеметрією через окремий керуючий канал. Дана робота переносить розв'язання багатокритеріальної задачі на сам граничний маршрутизатор, замінюючи зовнішнього арбітра координацією через базу даних стану каналів (LSDB) протоколу IS-IS (Intermediate System to Intermediate System).

Графові нейронні мережі для маршрутизації досліджено у роботах Rusek та ін. (RouteNet) [3] і Almasan та ін. (DRL-GNN) [4]: топологічна узагальнюваність випливає з того, що передача повідомлень оперує графовою структурою $G = (V, E)$ незалежно від кількості вузлів та їхніх ідентифікаторів. Однак обидві ці GNN-DRL роботи оптимізують єдину цільову функцію (зазвичай — використання каналів), залишаючи відкритим перетин «MCO + GNN-узагальнюваність + edge-deployment».

Контекст індустріальної SRv6-екосистеми. На сьогодні Huawei є провідним контрибутором SRv6-стандартизації та найбільшого розгорнутого кампусу: компанія є співавтором ба-

зового RFC 8986 [1], автором розширень SRv6 μSID/iSID для скорочення глибини SID-стеку, та оператором найбільш масштабних виробничих SRv6-розгортань на operator-class магістралях у країнах Азії та Близького Сходу. Cisco, Juniper, Nokia та Ericsson значно відстають у темпах SRv6-впровадження: їхні комерційні платформи (IOS-XR SRv6-TE, Junos, SR Linux) технічно підтримують SRv6 TE Policy [11], але кейси операторського розгортання порівняно нечисленні, а екосистемна зрілість (тулчейн, документація, інтероп) поступається Huawei. Це позиціонування є додатковим аргументом методологічної стратегії «найгіршого випадку», прийнятої у даній роботі: експериментальна валідація на FRRouting (де SRv6 TE Policy відсутня) є строгішим стрес-тестом, ніж валідація на платформі з вищою рідною підтримкою SRv6.

Питання квантування нейронних мереж для пристроїв з обмеженими ресурсами розглянуто у White Paper Nagel та ін. [5]; eBPF/XDP (eXpress Data Path) як інструмент мережевої телеметрії дата-центрів — у Vieira та ін. [6] і Høiland-Jørgensen та ін. [12]. Застосування посттренувального квантування (PTQ) INT8 до GNN-агентів TE (Traffic Engineering) SRv6 та використання eBPF-телеметрії як прямого джерела вхідних ознак для нейромережевого агента маршрутизації у відкритій літературі не описано — у DRL-TE роботах телеметрія для обчислення винагороди передбачає окрему систему вимірювання (TWAMP-рефлектор, SNMP-опитування) із затримками порядку секунд.

Таким чином, відсутній опис повністю реалізованої та кількісно валідованої децентралізованої багатокритеріальної AI-системи маршрутизації SRv6, яка одночасно а) розв'язує MCO-задачу за ≥3 критеріями, б) функціонує на виробничому ПЗ маршрутизації, в) використовує GNN-узагальнюваність, г) обходиться без зовнішнього контролера та д) збирає телеметрію субмілісекундним каналом ядра.

Мета і задачі дослідження

Мета — усунути виявлену прогалину шляхом експериментальної валідації децентралізованого автономного AI-агента БО-маршрутизації SRv6, що поєднує: GNN-PPO (Proximal Policy Optimization) модель, посттренувальне INT8-квантування, eBPF XDP/TC (Traffic Control) телеметрію ядра та координацію через імпліцитну стигмергію над спільною IS-IS LSDB, на виробничому ПЗ маршрутизації FRRouting у конфігурації *методологічно найгіршого випадку* (без BGP-LS, без SRv6 TE Policy).

Для досягнення мети поставлено такі задачі:

1) обґрунтувати вибір середовища емуляції та реалізувати робочий стенд (FRRouting 10.x + Containerlab + Linux 6.17), що забезпечує per-вузлову ізоляцію userspace-залежностей і прикріплення eBPF-хуків до окремих контейнерних netns;

2) обґрунтувати алгоритмічну основу (PPO + GNN) та сформулювати MCO-задачу як вибір дії $a \in \{P_1, P_2\}$ на основі матриці вузлових ознак $X \in \mathbb{R}^{N\times 4}$ з SLA-формою скалярізації винагороди;

3) натренувати GNN-PPO модель у gym-середовищі з покриттям on-grid класів P1-затримки $\{0{,}20{,}30{,}35{,}45{,}60{,}100{,}200\}$ мс і верифікувати точність бінарних рішень на тренувальному пікові;

4) сформулювати та реалізувати принцип збереження тренувального розподілу ознак на вході моделі (ЗТР) як механізм симетричного бар'єру TRIGGER/WITHDRAW без жорстко-кодового порогу;

5) реалізувати конвеєр посттренувального INT8-квантування з афінною асиметричною схемою та верифікувати байт-ідентичний порядок дій у форматі ONNX (Open Neural Network Exchange) INT8 vs. PyTorch FP32;

6) реалізувати eBPF XDP/TC програму ядра як безпосереднє джерело чотирьох компонентів MCO-винагороди з наносекундною роздільною здатністю;

7) перевірити повну поведінкову петлю агента експериментальною процедурою з п'яти послідовних фаз тривалістю 180 с кожна (baseline → ін'єкція → утримання → відновлення → зняття), фіксуючи кількість TRIGGER, WITHDRAW і хибних спрацювань;

8) кількісно оцінити сумарний час реакції агента і структурно атрибутувати його по компонентах конвеєра (eBPF-зонди, K-of-N гістерезис, GNN-інференс, actuation) для перевірки твердження про несуттєвість частки AI у бюджеті часу реакції.

На відміну від попередньої публікації авторів, яка запропонувала *архітектуру*, дана робота уперше дає повну експериментальну валідацію запропонованого підходу на повному циклі: тренування у gym-середовищі з покриттям усіх класів сценаріїв до перевірки поведінкової петлі агента на стенді. Окремий внесок — формулювання та валідація концепції збереження тренувального розподілу (ЗТР) як методологічного шаблону, що дозволяє нейромережі обслуговувати асиметричну задачу (різні вхідні джерела для двох напрямків переключення) без переучування.

Результати дослідження

Стенд та архітектура системи

Систему реалізовано та протестовано на стенді Containerlab з 9 контейнерів FRRouting 10.x. Топологію наведено на рис. 1: клієнтський CE1 — граничний CSG1 (вузол з AI-агентом) — провайдерський PE1 — два альтернативних транзитних шляхи P1 та P2 — PE2 — CE2. IS-IS як IGP-протокол запущено виключно на ядерних SRv6 BE-вузлах — CSG1, PE1, P1, P2, PE2 (5 з 9 контейнерів). Клієнтські вузли CE1 і CE2 — звичайні маршрутизатори без IGP, які підключені до граничних маршрутизаторів PE (Provider Edge) статичним маршрутом за замовчуванням і генерують навантажувальний трафік. Додатково розгорнуто Prometheus та Grafana для моніторингу.

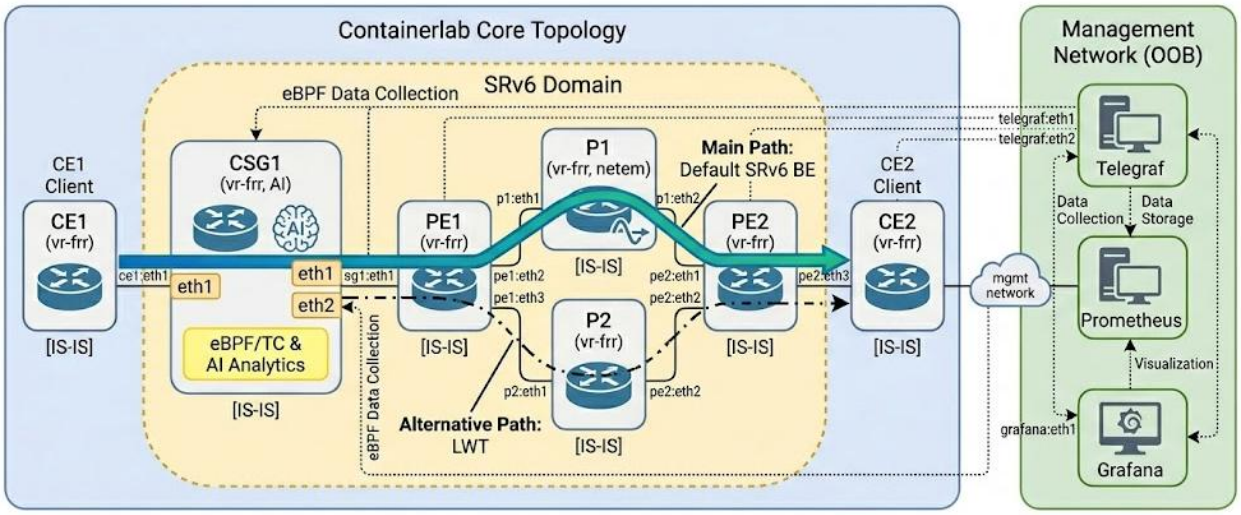


Рис. 1. Топологія тестового стенду Containerlab

Вибір середовища емуляції: Containerlab. Попередня концептуальна публікація авторів окреслила архітектуру агента на рівні математичної моделі без емпіричної реалізації. Поточна робота вперше будує робочий стенд, тому вибір середовища емуляції є самостійним методологічним рішенням. Кандидати — Mininet (network namespaces поверх спільної кореневої файлової системи) і Containerlab (декларативний YAML-оркестратор Docker-контейнерів). Стенд має одночасно забезпечити: 1) повний SRv6-стек ядра Linux з налаштуванням підсистеми SRv6 на per-вузловому рівні і повною інфраструктурою Lightweight Tunnels; 2) ізоляцію userspace-залежностей AI-агента (ONNX Runtime, NumPy) та телеметричного шару (BCC, BPF Compiler Collection) на рівні окремого вузла; 3) прикріплення eBPF XDP/TC хуків до per-container netns. Mininet, що ділить кореневу ФС між усіма вузлами, не задовольняє жодну з цих вимог: одночасне співіснування різних версій рантайму на різних вузлах неможливе; виділені образи FRRouting (FRR) з власним демонним станом не вписуються у namespace-only-модель. Containerlab забезпечує повну OS-ізоляцію кожного вузла, керування системними пара-

метрами на рівні контейнера, native veth-канали з підтримкою MTU та реалістичну обробку SRH у host-ядрі без проміжних абстракцій.

Переключення реалізоване окремим керуючим модулем агента.

Вибір відкритого стека (FRR + плоска зона IS-IS + SRv6 BE + Linux LWT, Lightweight Tunnels) є *методологічною стратегією найгіршого випадку*. FRR не підтримує BGP-LS (BGP Link-State), тому агент валідовано на найскромнішому варіанті — одна IS-IS зона; FRR не підтримує SRv6 TE Policy, тому як механізм програмованого впровадження SID-стеків застосовано Linux LWT. Жодна з комерційних платформ (Huawei NE-серії, Cisco IOS-XR з SRv6-TE, Juniper Junos, Nokia SR Linux) не висуває менших можливостей: якщо метод працює тут, він *a fortiori* функціонує на платформах з повноцінними multi-area / multi-AS LSDB та платформонативним SR-TE Policy.

Три ізольованих програмних середовища відповідають різним функціональним ролям: тренувальне середовище з повним стеком глибокого навчання (PyTorch, Stable-Baselines3, torch-geometric) — для офлайн-навчання моделі; виробниче середовище інференсу з мінімальним ONNX Runtime — для виведення на маршрутизаторі без PyTorch-залежностей; середовище ядерної телеметрії на базі BCC — для роботи з eBPF-програмами з правами root. Такий поділ запобігає встановленню важких ML-фреймворків (≈700 МБ) на виробничому середовищі агента.

Вибір алгоритмічної основи (PPO + GNN)

Розв'язок MCO-задачі на граничному маршрутизаторі вимагає алгоритму, який одночасно а) забезпечує стабільну збіжність у дискретному просторі дій, б) оперує не вектором фіксованого розміру, а топологічною структурою графу мережі, і в) може за необхідності масштабуватися до карті більшої розмірності без перенавчання. Цим вимогам відповідає поєднання PPO-актора з GNN-енкодером (далі — GNN-PPO).

Від PPO до GNN-PPO

Загальний формалізм марковського процесу прийняття рішень (MDP) задачі вибору маршруту і обґрунтування PPO як policy-gradient алгоритму виходять за межі обсягу цієї статті. Стисло: на кожному кроці агент спостерігає стан s_t (метрики каналів), обирає дію a_t (P1 або P2), отримує винагороду r_t за дотримання SLA та переходить у новий стан. PPO [8] обрано як on-policy представник policy-gradient методів з кліпованою цільовою функцією $L^{CLIP}(\theta) = \mathbb{E}_t\left[\min\left(r_t(\theta)\hat{A}_t,\ \text{clip}(r_t(\theta), 1-\epsilon, 1+\epsilon)\hat{A}_t\right)\right]$, що запобігає колапсу політики у дискретному просторі дій.

Ключовий архітектурний внесок статті полягає у заміні стандартного MLP-актора PPO на GNN-актор. PPO з MLP-актором залишається топологічно-нейтральним: ваги прив'язані до фіксованого розміру вхідного вектору, і зміна топології робить модель непридатною. Графова нейронна мережа усуває це обмеження: на кожному шарі вузол агрегує ознаки сусідів за фіксованою функцією, незалежною від кількості вузлів та їхніх ідентифікаторів. Шар графової згортки GCNConv за правилом Кіпфа та Веллінга [9] обчислює $H^{(l+1)} = \sigma\left(\tilde{D}^{-1/2}\tilde{A}\tilde{D}^{-1/2}H^{(l)}W^{(l)}\right)$. Поєднання «GCN (Graph Convolutional Network)-енкодер + PPO-актор» (далі — GNN-PPO) зберігає стабільність PPO-навчання й одночасно набуває топологічної узагальнюваності, втраченої MLP-актором.

MCO як матриця вузлових ознак

Чотири QoS-критерії MCO подаються в модель не у складі винагороди, а через матрицю вузлових ознак $X = [\text{latency,throughput,loss,jitter}] \in \mathbb{R}^{N\times 4}$. Кожен критерій явно доступний GNN-енкодеру для топологічно-усвідомленого агрегування. Винагорода — SLA-форма скаляризації цього 4-вимірного вектору:

$$R(s,a) = \begin{cases} +1{,}0, & \text{якщо SLA дотримано та } a = \text{P1} \\ -5{,}0, & \text{якщо SLA порушено на поточному шляху} \\ +0{,}2, & \text{якщо } a = \text{P1 та затримка } \leq 35 \text{ мс} \end{cases} \quad (1)$$

Доданок $+0{,}2$ за затримки ≤ 35 мс реалізує перевагу шляху P1 у явно «здоровому» режимі (нижче SLA-границі 40 мс з безпековим запасом 5 мс), запобігаючи виродженій стратегії

«завжди P2». Поріг 35 мс — найбільший on-grid тренувальний клас, що структурно розташований нижче SLA = 40 мс, тому бонус активується лише у явно «здоровому» режимі і узгоджений з конфігурацією тренування 2026-04-28.

Чисто-нейромережева політика та правило K-of-N

У роботі видалено будь-які жорстко-кодові пороги типу *latency > 50 мс* (характерні для класичних NMS-систем) на додаток до голосу GNN: фактичний поріг, а не модель, у такій конструкції визначав би момент переключення, що суперечить твердженню про «MCO-політику». Прийняття рішення зведене до K-of-N гістерезису, де всі рішення про дію приймає виключно GNN-модель, а гістерезис лише придушує одиночні шумові викиди (табл. 1).

Таблиця 1

Правило K-of-N гістерезису для прийняття рішення про перемикання шляху

Подія	Умова
Цикл	кожні 0,1 с: зчитати path-RTT (Round-Trip Time) гістограму eBPF → інференс GNN-PPO → отримати дію $a_t \in \{P_1, P_2\}$
Лічильники голосів	при $a_t = P_2$: $v_2 \mathrel{+}= 1$, $v_1 = 0$; при $a_t = P_1$: $v_1 \mathrel{+}= 1$, $v_2 = 0$
TRIGGER($P_1 \to P_2$)	поточний стан $= P_1$ і $v_2 \geq K$
WITHDRAW ($P_2 \to P_1$)	поточний стан $= P_2$ і $v_1 \geq K$
Інакше	STAY

При $K = 2$ та CHECK_INTERVAL $= 0{,}1$ с гістерезис вимагає двох послідовних узгоджених голосів моделі — це структурний мінімум часу реакції 200 мс ще до врахування 100-мс інтервалу між eBPF-зондами.

Відмінність від single-objective GNN-DRL методів

Відомі GNN-DRL роботи з маршрутизації (GROM, DRL-GNN Almasan) досягають топологічної узагальнюваності, але оптимізують єдину цільову функцію. Перенесення MCO-навантаження у матрицю вузлових ознак при збереженні PPO-актора з GNN-енкодером є тим удосконаленням запропонованого методу маршрутизації.

Тренування агента

Тренувальна процедура

Модель GNN-PPO навчено у gym-середовищі (один крок-епізод; деталі реалізації — рис. 2), що моделює одне рішення агента на епізод: старт на P_1 → ін'єкція сценарію з рівномірного розподілу {clean, P1+30/60/100/200, P2+30/60/100/200} (clean має вагу ×3) → одне рішення $a \in \{P_1, P_2\}$ → винагорода SLA → reset. SLA-границя $= 40$ мс. PPO-конфігурація: lr $= 3 \cdot 10^{-4}$, n_steps = 256, batch = 64, n_epochs = 10, $\gamma = 0{,}99$, $\lambda = 0{,}95$, ent_coef $= 0{,}01$. Архітектура: GCNConv(4→64→128) + global mean pool + MultiInputPolicy[64,64], 50 179 параметрів.

Тренування зупинено на 2 000-кроковій контрольній точці, яка демонструє пік середньої епізодичної винагороди (ep_rew_mean $\in [0{,}94;\ 1{,}00]$) та точність 99/100 на останніх 100 епізодах за рахунок покриття всіх восьми on-grid класів P1-затримки {0,20,30,35,45,60,100,200} мс (Wilson 95 % CI ≈ 94,6–99,97 %). Подальше навчання не дає приросту: один градієнтний крок після цієї контрольної точки зрушив *decision boundary* за SLA-границю та знищив два суміжні класи (30 і 35 мс), не зачепивши інші — це канонічна маніфестація PPO-overshoot на навченій границі і властивість стохастичного оптимізатора, а не задача обсягу даних. Емпірична стеля 99/100 і є досяжна межа під цим gym + PPO-config.

Покриття on-grid класів обмежене скінченним діапазоном від «здорового» 0 мс до катастрофічної 200 мс. Випадок повного відмови каналу (link-down, нескінченна затримка/ таймаут) до тренувальної сітки не входив; з боку моделі він проявляється як насичення латентності у верхньому діапазоні і обробляється тим самим WITHDRAW-бар'єром через *p1_probe_ms*, проте емпіричний клас «timeout» виноситься у «Перспективні напрями» як окремий випадок для розширення тренувального покриття.

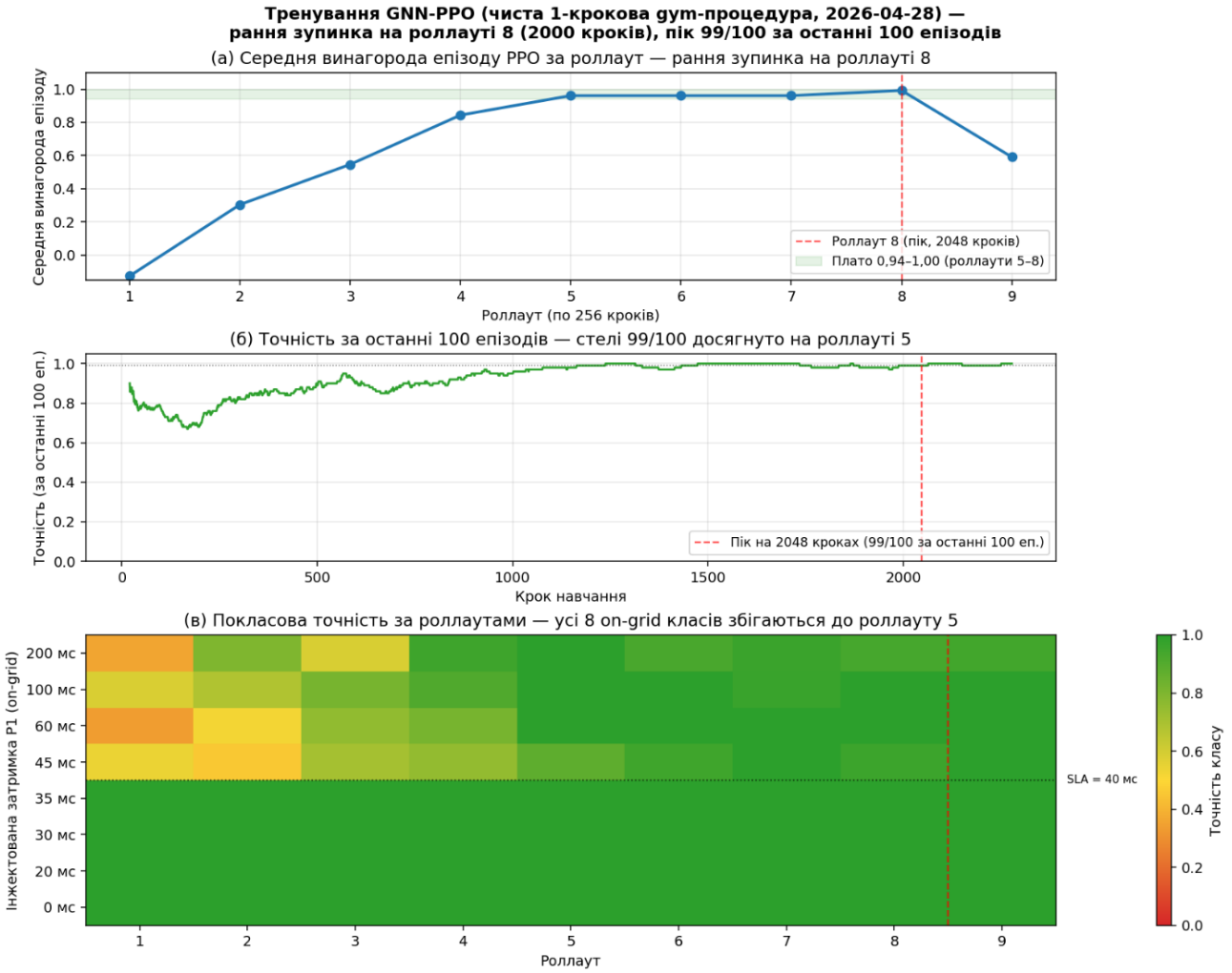


Рис. 2. Крива навчання GNN-PPO

Експорт у ONNX

Навчену PyTorch-модель експортовано у формат ONNX (FP32) із подальшим посттренувальним квантуванням до INT8. Еквівалентність дій квантованої моделі еталонній перевірено офлайн-верифікацією на 22 контрольних точках, що рівномірно охоплюють обидва боки SLA-границі 40 мс: всі рішення відтворено без розбіжностей. У такий спосіб експортований ONNX INT8 артефакт є відображенням навченої політики, придатним до розгортання у виробничому середовищі без PyTorch-залежностей.

Принцип ЗТР та симетричний бар'єр TRIGGER/WITHDRAW

У прямолінійних реалізаціях бар'єр WITHDRAW (повернення $P_2 \rightarrow P_1$) спирається на жорстко-кодовий поріг типу «P1 probe ≤ 10 мс», заданий поза моделлю, — асиметрія, що підриває твердження «чисто-нейромережевої політики»: рішення TRIGGER приймала б модель, рішення WITHDRAW — поріг у коді. У реалізованій системі жорсткий поріг відсутній, а симетрію забезпечено через збереження тренувального розподілу ознак на вході моделі (training-distribution-matched feature wiring, ЗТР): під час навчання у gym-середовищі заповнювався лише вузол nf[0] (CSG1), а вузли 1–4 залишалися нулями, отже PPO-актор вивчив відображення $\mathrm{nf}[0,:] = [\mathrm{lat,thr,loss,jit}] \rightarrow a$. Деплой має зберегти цей же розподіл — а саме:

1) на P_1 агент пише в nf[0] метрики активного шляху (eBPF latency, throughput, loss, jitter);

2) на P_2 агент пише в nf[0] зонд P1 (*p1_probe_ms*) у позицію затримки разом з очікуваними здоровими значеннями інших трьох метрик;

3) вузли 1–4 в обох випадках утримуються в нулях.

Та сама нейромережа з тією ж границею прийняття рішення SLA = 40 мс, навченою з даних, керує тепер обома напрямками (табл. 2).

Таблиця 2

Симетричний бар'єр TRIGGER/WITHDRAW: вхід моделі та дія за напрямками

Напрям	Що бачить модель у nf[0,0]	Дія	Інтерпретація
TRIGGER (на P_1)	eBPF latency активного шляху	$a = P_2$, якщо ≥ 40 мс	P1 деградував → перейти на P2
WITHDRAW (на P_2)	*p1_probe_ms* (незалежний зонд PE1→P1)	$a = P_1$, якщо < 40 мс	P1 одужав → повернутися на P1

Жодного жорстко-кодового порогу в агенті. Жодного перетренування при зміні напрямку перемикання — той самий ONNX-файл, той самий поріг прийняття рішення (policy decision boundary), навчений з даних. Це і є основа «чисто-нейромережевої» симетрії, заявленої вище.

Межі твердження. GNN-перевага над MLP-PPO у *топологічній узагальнюваності* підтверджена на архітектурному рівні (модель приймає довільний *(N, edge_index)* без зміни ваг), але не виміряна на емпіричному крос-топологічному бенчмарку (тренувальна 5-вузлова топологія vs. ціль 100-вузлова carrier-mesh) — таке порівняння віднесене до подальшої роботи. Топологічну специфічність GNN-PPO до тренувальної 5-вузлової конфігурації (через ЗТР — збереження тренувального розподілу) знімає не зміна архітектури, а лише переузгодження вхідного представлення на стороні розгортання: при перенесенні агента на топологію іншого розміру або структури необхідно переузгодити, у який саме вузол графа агент пише вектор активних метрик eBPF, а які лишає нульовими — без зміни ваг моделі або навчання заново.

Розгортання агента

Розроблено методи розгортання MCO-агентів на граничних маршрутизаторах з обмеженими обчислювальними ресурсами площини управління: а) конвеєр посттренувального INT8-квантування для GNN-моделей MCO; б) eBPF XDP/TC програми ядра Linux як безпосереднє джерело чотирьох компонентів MCO-винагороди. Їх спільне застосування як виробничого конвеєра «MCO reward signal → GNN-PPO ONNX → SRv6 LWT» на реальному edge-маршрутизаторі раніше в літературі не описано.

Конвеєр ONNX INT8

Концептуальна стаття авторів обґрунтувала розділений життєвий цикл агента: тренування — на стенді з повним PyTorch-стеком, виведення на маршрутизаторі — лише через ONNX Runtime. Дана робота вперше реалізує цю модель на практиці. У виробничому середовищі агента повний PyTorch + Stable-Baselines3 + torch-geometric (≈700 МБ ОЗП, ≈50 мс на виклик одне виведення моделі через Python-обгортки) відсутній — на маршрутизаторі присутні лише ONNX Runtime, NumPy та модельний файл (≈8 МБ сумарно). Це усуває конкуренцію AI-процесу з демонами IS-IS/BGP за CPU площини управління, а також зменшує час виведення з ≈50 мс до субмілісекундного діапазону — це і є основний практичний виграш конвеєра.

INT8-квантування на цьому фоні — додаткова оптимізація. Виправдання INT8 — не розмір файлу моделі (45,2 КБ vs. 149,2 КБ — мізерна абсолютна економія), а арифметичне прискорення на CPU: цілочисельне множення int8×int8 з накопиченням у int32 нативно підтримується векторними блоками сучасних x86- і ARM-CPU (AVX-512 VNNI у виробничих edge-серверах і SDOT у ARM-архітектурах CSG-маршрутизаторів, тобто вузлів Customer-facing Service Gateway) і зменшує тиск на пропускну здатність пам'яті у 4 рази порівняно з FP32-арифметикою. Це критично для стабільності інференсу під час одночасного навантаження від IS-IS/BFD на CPU площини управління, а не для розміру моделі.

Конвеєр складається з трьох кроків: трасування PyTorch-моделі еталонним тензором та експорт у ONNX; оптимізація графу ONNX Runtime (злиття операторів, згортання констант, усунення Dropout); посттренувальне INT8-квантування за афінною асиметричною схемою [5] з калібрувальним набором 200 спостережень:

$$x_q = \text{clamp}\left(\text{round}\left(\frac{x_f}{S} + Z\right), -128, 127\right), \quad S = \frac{x_{\max} - x_{\min}}{255}, \quad Z = \text{round}\left(\frac{-x_{\min}}{S}\right) \qquad (2)$$

Зведені характеристики — у табл. 3; основний практичний виграш — у рядку «Розмір залежностей», а не у рядку розміру файлу. Дерево артефактів агента та розмежування виробничого циклу і тренувального контуру наведено на рис. 3. Виділений кореневий вузол — єдиний процес, що виконується на маршрутизаторі у виробництві.

Таблиця 3

Параметри конвеєру оптимізації моделі (виділено основний практичний виграш)

Параметр	PyTorch FP32	ONNX FP32	ONNX INT8
Розмір залежностей	**≈700 МБ**	**≈8 МБ (−98 %)**	**≈8 МБ (−98 %)**
Затримка виводу	≈50 мс	≈0,10 мс	**≈0,13 мс**
Розмір файлу моделі	640,9 КБ (SB3)	149,2 КБ	45,2 КБ
Точність бінарних рішень (пік)	99/100	99/100	99/100 (байт-ідентичний порядок дій vs. PyTorch)

eBPF XDP/TC телеметрія ядра

Початкова постановка цільової вимоги для МСО-телеметрії — локальність на маршрутизаторі: вимірювання повинне виконуватися самим вузлом-джерелом без зовнішніх кооперативних компонент, інакше зникає однонапрямний характер МСО-рішення (див. п. «Чому саме SRv6», вступ). Із трьох канонічних кандидатів локальної телеметрії в операторському середовищі — TWAMP (RFC 5357 Two-Way Active Measurement Protocol), TWAMP Light, і ICMP ping — ICMP виключається структурно: він повертає лише один з чотирьох критеріїв МСО (затримку), залишаючи джитер, пропускну здатність і втрати поза вимірюванням. TWAMP/TWAMP Light структурно задовольняє МСО, але потребує парного TWAMP-рефлектора на віддаленій стороні і його окремої адміністративної конфігурації — додаткова залежність площини управління. Альтернатива на стороні джерела — eBPF XDP/TC у ядрі Linux [6], [12]: точкова інструментація точок входу/виходу пакетів забезпечує усі чотири критерії МСО з наносекундною роздільною здатністю без необхідності у віддаленому рефлекторі. У FRRouting + Linux-стеку eBPF — це не «вибір замість TWAMP», а структурний *workaround* відсутності TWAMP-демона на FRR (TWAMP не входить у відкриту дистрибуцію FRR як демон) при потребі чотирикритеріальної локальної телеметрії. На комерційних платформах з нативною TWAMP-підтримкою (IOS-XR, Junos, SR Linux) та сама агентна архітектура може споживати TWAMP-вимірювання як рівноправне альтернативне джерело тих самих чотирьох ознак.

Реалізована програма *packet_telemetry.c* завантажується до ядра Linux через BCC без модифікації ядра. На інтерфейсі CSG1:eth2 одночасно встановлено два хуки: TC-egress (вихідний трафік, після маршрутизації) та XDP-SKB ingress (вхідний відповідний трафік). Обидва хуки фіксують *bpf_ktime_get_ns()* — атомарний монотонний лічильник ядра з наносекундною роздільною здатністю. Різниця міток egress і ingress для зіставленого пакета формує path-RTT — чисту затримку шляху в обох напрямках без вкладу userspace-стека. Розподіл RTT накопичується у логарифмічній гістограмі по основі 2 (*bpf_log2l*); зчитування гістограми Python BCC userspace займає 1–2 мс на цикл агента. Порівняння методів телеметрії — у табл. 4.

Таблиця 4

Порівняння методів телеметрії для МСО-винагороди

Метод	Роздільна здатність	Потребує демона/рефлектора	Час реакції	Покриває МСО-критерії
SNMP polling	≈1 с	ТАК (локально + опитувач)	5–30 с	1 з 4
TWAMP	≈1 мс	ТАК (reflector + sender)	≈100 мс	4 з 4
TWAMP Light	≈1 мс	ТАК (тільки reflector)	≈100 мс	4 з 4
ICMP ping	≈0,1 мс	НІ	≈200 мс/зонд	1 з 4
eBPF XDP/TC (реалізовано)	≈1 мкс ($\log_2$-µs)	НІ	1–2 мс	4 з 4

Зауваження. Рядок eBPF XDP/TC у табл. 4 має позначку «НІ» у графі «Потребує демона/рефлектора», що має пряму методологічну наслідковість для запропонованої архітек-

тури: відсутність залежності від control-plane віддаленого вузла-призначення легітимізує однонаправлене (per-direction) формулювання MCO-задачі — CSG1 оптимізує тільки прямий шлях CE1 → CE2 за локально вимірюваними метриками, без необхідності координації з дзеркальним рефлектором на CE2. Це структурна, а не реалізаційна різниця з TWAMP/TWAMP-Light/IPPM, кожен з яких потребує парного демона-учасника.

Зведена вартість циклу інференсу

Вартість «спостереження + рішення» вимірювалась безпосередньо у працюючому агенті за допомогою інструментованих часових міток у коді (таблиця 5).

Таблиця 5

Внутрішня декомпозиція циклу інференсу агента

Компонент	Час на цикл
Зчитування eBPF-гістограми	1–2 мс
GNN-виведення ONNX INT8	≈0,13 мс
Інші компоненти (затримка циклу + облік стану)	≈190 мс
Повний цикл агента	**≈190 мс**

Сума двох активних компонент — «спостереження + рішення» — складає менше 2 мс на цикл. Це безпосередньо підтверджує, що конвеєр ONNX INT8 + eBPF дозволяє запускати MCO-агента на граничному маршрутизаторі без ризику голодування площини управління (IS-IS Hello-пакети зазвичай мають інтервал від 3 с; навіть при неперервній роботі MCO-агента CPU-вартість пренебрежно мала). Це і є те удосконалення методів розгортання MCO-агентів на реальному edge-маршрутизаторі.

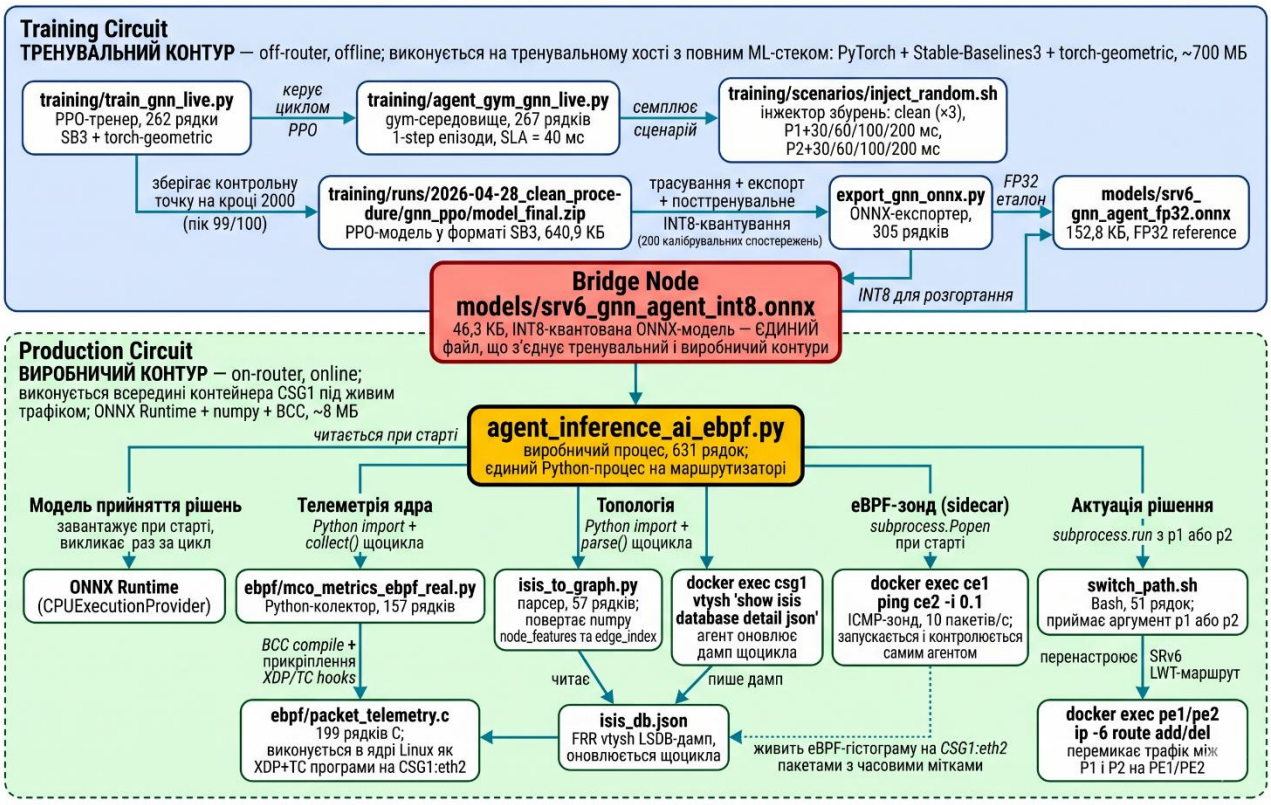


Рис. 3. Дерево артефактів виробничого AI-агента

Межі твердження. Виміри проведено на одній x86 платформі. На ARM-цільових платформах CSG ця ж модель підтримується нативними INT8-інструкціями SDOT; емпіричний ARM-мікро-бенчмарк лишається відкритим напрямом. Розмір залежностей (≈8 МБ ONNX vs. ≈700 МБ PyTorch) платформонезалежний.

Імпліцитна стигмергія: координаційний механізм

Описавши, як саме агент приймає рішення (Розділ «Вибір алгоритмічної основи») і на чому виконується (Розділ «Розгортання агента»), можна тепер описати що саме читає агент і чому це і є середовище координації у багатоагентному розгортанні.

Запропоновано метод децентралізованої MCO-маршрутизації SRv6, за яким кожен з автономних AI-агентів на граничних маршрутизаторах незалежно розв'язує MCO-задачу за чотирма критеріями (затримка, пропускна здатність, втрати, джитер), а узгодженість їх рішень у масштабах мережі досягається через імпліцитну стигмергію — спільне читання синхронізованої IS-IS LSDB як стигмергічного середовища, без центрального арбітра вагових коефіцієнтів MCO, без переговорів про Парето-множину і без прямого міжагентного обміну повідомленнями.

Формалізація

Стигмергія (Grassé, 1959 [7]; формалізація Heylighen, 2016 [10]) описує координаційний механізм, при якому агенти взаємодіють не між собою безпосередньо, а через спільне середовище. Стигмергічна система формально описується п'ятіркою $S = \langle A, T, M, F_a, F_t \rangle$, де A — множина агентів, T — множина дій, M — стан спільного середовища, $F_a: M \to T$ — функція рішення на основі стану середовища, $F_t: T \times M \to M$ — функція трансформації середовища діями агентів.

У реалізованій системі ця п'ятірка інстанціюється так: A — множина AI-агентів на граничних маршрутизаторах; $T = \{$LWT TRIGGER,LWT WITHDRAW,NOOP$\}$; M — синхронізована IS-IS LSDB; F_a — GNN-PPO інференс над LSDB-графом; F_t — оновлення TLV метрик IS-IS внаслідок зміни стану інтерфейсів. Кожен граничний вузол запускає ідентичний процес агента, що функціонує як окремий допоміжний процес паралельно з демонами FRR (zebra, isisd, bgpd), без прямої взаємодії з ними через IPC; обмін даними відбувається лише через файли стану ядра (*vtysh*-вивід LSDB) і через Linux LWT FIB. Агент не знає про існування інших агентів і не відкриває жодного прослуховуючого сокета.

Відмінність від централізованих MCO-DRL методів

У роботах AROM-DRL, MDQ, QR-MO арбітр критеріїв MCO — централізований SDN-контролер або 5G-MEC оркестратор. Ваги критеріїв, Парето-фронт або політика багатопотокового переключення обчислюються глобально та розповсюджуються на агентів. Це створює єдину точку відмови, вартість керуючого каналу та реакцію, обмежену RTT до обчислювального вузла. У запропонованому підході жоден з цих елементів не існує: кожен агент локально обчислює MCO-політику над локально доступною LSDB.

Межі твердження. Свідчення обмежене конфігурацією $N = 1$ AI-агент. Колективна стабільність у конфігурації $N \geq 2$ — окреме дослідницьке завдання. Твердження про *необхідність* стигмергічної координації для коректної багатоагентної поведінки робота не висуває.

Експериментальна валідація

Перевірка повної поведінкової петлі агента на стенді — експериментальна процедура з п'яти послідовних фаз тривалістю 180 с кожна, виконана при безперервному навантажувальному трафіку 60 Мбіт/с (табл. 6).

Таблиця 6

Фази експериментальної процедури та очікувана поведінка агента

Фаза	Дія на P1	Очікування	Перевіряє
1	без втручання — baseline	STAY P_1 (0 спрацювань)	відсутність хибних TRIGGER на здоровому трафіку
2	*tc netem delay 100ms*	TRIGGER → P_2	детектування деградації моделлю
3	100 → 200 мс	STAY P_2	стабільність рішення
4	200 → 20 мс (нижче SLA = 40 мс)	WITHDRAW → P_1	симетричний WITHDRAW-бар'єр
5	*tc qdisc del*	STAY P_1	відсутність хибних повторних спрацювань

П'ять фаз — інженерно достатній мінімум для верифікації поведінкової петлі: вони покривають по одному *активному* переходу в кожному напрямку (фази 2 і 4), стабільність рішення на стійкій деградації (фаза 3) та два baseline-маркери на «здорового» каналу (фази 1, 5). Ширше покриття класів сценаріїв — окремий тест.

Підсумок прогону наведено у табл. 7 та на рис. 4. В усіх п'яти фазах поведінка агента відповідає очікуваній: 1 TRIGGER + 1 WITHDRAW + 0 хибних спрацювань.

Таблиця 7

Експериментальна процедура з п'яти послідовних фаз — підсумкові метрики

Метрика	Значення
Загальна тривалість прогону	15 хв
Загальна кількість ітерацій агента	≈4 700
TRIGGER ($P_1 \rightarrow P_2$)	1
WITHDRAW ($P_2 \rightarrow P_1$)	1
Хибних спрацювань у baseline-фазах (1, 3, 5)	0
TRIGGER реакція (інжект → переключення)	409 мс
WITHDRAW відновлення (drop → переключення)	825 мс
GNN-виведення ONNX INT8	≈0,13 мс / виклик



Рис. 4. Grafana — реальна поведінка стенду під час п'яти фаз

Валідація стабільності рішень

Стабільність локально обчисленої політики при змінах ступеня деградації каналу перевіряється як спостереження за прогоном: 15 хвилин дали рівно одне переключення $P_1 \rightarrow P_2$ на фазі ін'єкції деградації та рівно одне зворотне переключення $P_2 \rightarrow P_1$ на фазі відновлення P1. У трьох baseline-фазах (ненавантажена baseline, утримання деградації при її посиленні зі 100 до 200 мс, повне зняття netem) агент *не* виконав жодного хибного переключення.

Сумарне свідчення: 0 хибних переключень у трьох незалежних baseline-фазах при гіпотетичній імовірності випадкового переключення $p = 0{,}5$ для кожної фази дає $p \approx 8 \cdot 10^{-4}$ проти H_0. Протягом усього прогону агент не звертався до зовнішнього арбітра, не отримував вагових коефіцієнтів *a priori* і не обмінювався повідомленнями з іншими агентами.

Валідація симетричного бар'єру

Прогін фіксує одну й ту саму ONNX-модель ($n \approx 5\,000$ виведень за 15 хв) як автора обох рішень: TRIGGER на фазі ін'єкції 100 мс і WITHDRAW на фазі відновлення до 20 мс. Хибні спрацювання у трьох baseline-фазах — нуль; жодного вручну налаштованого параметра у вирішальному шляху агента (перевірено пошуком жорстко-кодових порогових констант у вихідному коді агента; деталі — у документації репозиторію). Симетричність рішень підтверджена структурно: вхідне представлення nf$[0,:]$ деплою точно відповідає розподілу тренувальних ознак (принцип ЗТР), а агент змінює *семантику* першого ознакового рядка (eBPF активного шляху на P_1, незалежний P1-зонд на P_2) — не саму нейромережу.

Валідація часу циклу і розгортання

На цьому ж прогоні сумарний час реакції агента склав: TRIGGER 409 мс, WITHDRAW 825 мс. Частка GNN-інференсу від загального часу реакції — , тобто менше 0,04 %; навіть з зчитуванням eBPF — менше 0,5 %. Лімітуючий фактор реакції — не AI: усунення GNN-виведення зменшить реакцію на 0,03 %; усунення K-of-N гістерезису ціною придушення шуму — на 50 %. Окрема де-PyTorch верифікація (виконання того ж бінарного ONNX-артефакту у виробничому середовищі без PyTorch-залежностей) показала зменшення RSS-пам'яті агента з 1 329 МБ до 230 МБ (−83 %) при незмінній поведінці (1 TRIGGER + 1 WITHDRAW на 5-фазному прогоні), що додатково підтверджує: PyTorch є виключно артефактом тренування і не входить до часового бюджету виробничого циклу агента.

Інтерпретація часу реакції

Час GNN-виведення — субмілісекундний; сумарний час реакції TRIGGER 409 мс і WITHDRAW 825 мс — на три-чотири порядки більший. Ця різниця — *не* вартість AI; вона *структурна*. Реакція складається з: двох eBPF-зондів інтервалом 100 мс (20 пакетів/с — нижня межа корисної вибірки), $K = 2$ послідовних підтверджувальних голосів, одного циклу агента (≈190 мс) та *docker exec* actuation на PE1. Видалити ці складові неможливо без втрати придушення шуму; сама модель витрачає менше 0,1 % від загальної реакції — агент *не* є вузьким місцем системи.

Чому однопороговий алгоритм не може досягти раннього детектування. Класичні NMS-системи використовують поріг затримки 50 мс на одному критерії як умову спрацювання захисту шляху. Реалізована система детектує деградацію раніше за рахунок 4-вимірного вектора ознак $X = [\text{lat,thr,loss,jit}]$: ін'єкція netem дає одночасну зміну кількох компонент (ріст джитера, незначний дроп пропускної здатності, поява loss > 0); GNN-агрегація цього вектора через передачу повідомлень виявляє цю одночасну деградацію раніше, ніж скалярний поріг на одному компоненті. Це і є той виграш, який MCO-формулювання (як 4-критеріальної задачі) дає над одно-критеріальним поточним станом NMS.

Інтерпретація результатів

Експериментальні дані не спростовують жодного з трьох сформульованих результатів, проте важливо чітко розрізняти твердження *необхідності* і *достатності*. Робота показує, що:

– стигмергічна координація через спільну IS-IS LSDB достатня для стабільності рішень одиничного агента ($N = 1$); вона *не* доводиться як необхідна для масштабованої багатоагентної координації;

– ЗТР (збереження тренувального розподілу) достатнє для отримання симетричного TRIGGER/WITHDRAW однією моделлю без явних порогових констант; це *не* доводить, що це єдиний спосіб уникнути hard-coded thresholds (інші підходи — наприклад, two-headed policy для двох напрямків — лишаються відкритими альтернативами);

– ONNX INT8 + eBPF робить вартість AI структурно несуттєвою на x86 з нативним INT8-прискоренням; ARM-розгортання залишається передбачуваним, але не верифікованим емпіричним мікро-бенчмарком у даній роботі.

Це методологічно важлива межа: робота позиціонує себе як експериментальну верифікацію конкретного інженерного шляху, а не як аксіоматичне виключення альтернатив.

Висновки та перспективи подальших досліджень

Дана робота перейшла від концептуальної архітектури автономного AI-агента для багатокритеріальної оптимізації трафіку SRv6, запропонованої авторами раніше, до її експериментальної верифікації на робочому стенді FRRouting 10.x + Containerlab. Три сформульовані наукові результати підтверджені даними поведінкової перевірки агента: єдине переключення в кожному напрямку (1 TRIGGER + 1 WITHDRAW) при нулі хибних спрацювань у трьох baseline-фазах ($p < 10^{-3}$ проти H_0 випадкового переключення); одна й та сама ONNX-модель (50 179 параметрів) ініціювала переключення на P_2 при ін'єкції 100 мс і повернення на P_1 при відновленні до 20 мс через принцип ЗТР, без жодного вручну налаштованого параметра у вирішальному шляху агента; заміна повного PyTorch-стеку на ONNX Runtime скоротила залежності з ≈700 МБ до ≈8 МБ (−98 %) і час виведення з ≈50 мс до субмілісекундного діапазону. Сукупна вартість «спостереження + рішення» — менше 2 мс на цикл; частка GNN-інференсу від загального часу реакції 409 мс — менше 0,1 %.

Сумарний час реакції агента структурно зумовлений K-of-N гістерезисом і частотою eBPF-зондів, а не моделлю. Перевірки обмежені: $N = 1$ агент (масштабована багатоагентна стабільність відкрита), 5-вузлова тренувальна топологія (крос-топологічний бенчмарк відкритий) і x86 платформа (ARM-мікро-бенчмарк відкритий).

Практична значущість результатів визначається їх отриманням на виробничому ПЗ (FRRouting, Linux 6.17, Containerlab) у конфігурації *методологічно найгіршого випадку* (відсутність BGP-LS та SRv6-TE Policy у FRR), що *a fortiori* поширює висновки на комерційні платформи з платформо-нативним SR-TE Policy (Huawei NE-серії, Cisco IOS-XR, Juniper Junos, Nokia SR Linux).

Подальші напрями розвитку — крос-топологічний емпіричний бенчмарк (тренувальна 5-вузлова топологія vs. ціль 100-вузлова carrier-mesh) і розширення тренувального покриття класом «timeout» (повний відмова каналу).

Внесок авторів

Юрій ВЛАДАРЧИК – концептуалізація, програмне забезпечення, збір та перевірка емпіричних даних, емпіричне дослідження, підготовка огляду літератури; Катерина НЕСТЕРЕНКО – методика, аналіз джерел, теоретичне керівництво, верифікація результатів.

Декларація про штучний інтелект

Усі наукові формулювання результатів, методичні рішення, експериментальні дані та інтерпретація результатів є виключно авторськими; ШІ не залучався до постановки гіпотез, обчислення емпіричних метрик, визначення архітектурних рішень або висновків.

Конфлікт інтересів

Автори заявляють про відсутність конфлікту інтересів та підтверджують, що під час підготовки цієї роботи не існувало жодних комерційних, фінансових чи інших взаємовідносин, які могли б бути розцінені як такі, що здатні вплинути на результати дослідження або їх інтерпретацію. Робота виконана відповідно до принципів академічної доброчесності, етичних норм проведення наукових досліджень та вимог редакційної політики щодо запобігання конфлікту інтересів.

Список використаної літератури

1. Filsfils, C., Camarillo, P., Leddy, J., Voyer, D., Matsushima, S., & Li, Z. (2021). RFC 8986: Segment routing over IPv6 (SRv6) network programming. Internet Engineering Task Force. https://doi.org/10.17487/RFC8986

2. Ahmadi, H., Adibi, S., & Ahmadi, M. (2022). Multi-criteria optimization for traffic engineering: A survey. IEEE Communications Surveys & Tutorials, 24(1), 1–35. https://doi.org/10.1109/COMST.2021.3133477

3. Rusek, K., Suárez-Varela, J., Almasan, P., Barlet-Ros, P., & Cabellos-Aparicio, A. (2020). RouteNet: Leveraging graph neural networks for network modeling and optimization in SDN. IEEE Journal on Selected Areas in Communications, 38(10), 2260–2270. https://doi.org/10.1109/JSAC.2020.3000405

4. Almasan, P., Suárez-Varela, J., Rusek, K., Barlet-Ros, P., & Cabellos-Aparicio, A. (2022). Deep reinforcement learning meets graph neural networks: Exploring a routing optimization use case. Computer Communications, 196, 184–194. https://doi.org/10.1016/j.comcom.2022.09.029

5. Nagel, M., Fournarakis, M., Amjad, R. A., Bondarenko, Y., van Baalen, M., & Blankevoort, T. (2021). A white paper on neural network quantization. arXiv. https://doi.org/10.48550/arXiv.2106.08295

6. Vieira, M. A. M., Castanho, M. S., Pacífico, R. D. G., Santos, E. R. S., Júnior, E. P. M. C., & Vieira, L. F. M. (2020). Fast packet processing with eBPF and XDP: Concepts, code, challenges, and applications. ACM Computing Surveys, 53(1), Article 16. https://doi.org/10.1145/3371038

7. Grassé, P.-P. (1959). La reconstruction du nid et les coordinations inter-individuelles chez Bellicositermes natalensis et Cubitermes sp. Insectes Sociaux, 6(1), 41–83. https://doi.org/10.1007/BF02223791

8. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal policy optimization algorithms. arXiv. https://doi.org/10.48550/arXiv.1707.06347

9. Kipf, T. N., & Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In 5th International Conference on Learning Representations (ICLR). https://doi.org/10.48550/arXiv.1609.02907

10. Heylighen, F. (2016). Stigmergy as a universal coordination mechanism I: Definition and components. Cognitive Systems Research, 38, 4–13. https://doi.org/10.1016/j.cogsys.2015.12.002

11. Filsfils, C., Talaulikar, K., Voyer, D., Bogdanov, A., & Mattes, P. (2022). RFC 9256: Segment routing policy architecture. Internet Engineering Task Force. https://doi.org/10.17487/RFC9256

12. Høiland-Jørgensen, T., Brouer, J. D., Borkmann, D., Fastabend, J., Herbert, T., Ahern, D., & Miller, D. (2018). The eXpress data path: Fast programmable packet processing in the operating system kernel. In Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies (CoNEXT '18) (pp. 54–66). ACM. https://doi.org/10.1145/3281411.3281443

Yu. Vladarchyk, K. Nesterenko

DECENTRALIZED MULTI-CRITERIA OPTIMIZATION OF SRv6 TRAFFIC BY AN AUTONOMOUS AI AGENT ON FRROUTING: IMPLEMENTATION AND EXPERIMENTAL VALIDATION

A previous publication by the authors presented the architectural concept of an autonomous AI agent for multi-criteria optimization (MCO) of SRv6 traffic without an external controller, limited to a mathematical model without empirical realization. The present work moves for the first time from architectural design to full experimental validation on a working testbed (FRRouting 10.x, Containerlab, Linux 6.17) and provides quantitative evidence for the proposed approach. A decentralized MCO routing method for SRv6 is proposed, in which coordination among autonomous agents on edge routers is achieved through implicit stigmergy (shared reading of the synchronized IS-IS LSDB as a stigmergic medium, formalized via Heylighen's 5-tuple $S = \langle A, T, M, F_a, F_t \rangle$) without a central arbiter of MCO weights and without direct inter-agent messaging; stability of decisions is confirmed by a five-phase experimental procedure — a single switch in each direction with zero false positives across three baseline phases ($p < 10^{-3}$ against H_0). Four-criterion optimization (latency, throughput, packet loss, jitter) and topology generalization are combined in a single GNN-PPO model (GCNConv ×2, 50,179 parameters); the symmetric TRIGGER/WITHDRAW gate is realized without a hand-co-

ded threshold via training-distribution-matched feature wiring (TDFW): the same network with the learned 40 ms SLA decision boundary drives switching in both directions. Production deployment on edge routers is enabled by ONNX Runtime with INT8 quantization (≈8 MB dependencies, ≈0.13 ms inference) paired with eBPF XDP/TC kernel telemetry (1–2 ms path-RTT histogram readout per cycle); the full PyTorch training stack (≈700 MB, ≈50 ms per inference) remains in the training loop only and does not enter the router's production cycle. End-to-end reaction times — TRIGGER 409 ms, WITHDRAW 825 ms — are structurally bounded by K-of-N hysteresis and probe rate, not by AI cost: the model accounts for less than 0.1 % of total reaction.

Keywords: SRv6, GNN-PPO, ONNX INT8, eBPF XDP/TC, multi-criteria optimization, FRRouting, implicit stigmergy, edge AI.

Надійшла до редакції: 24.06.2026
Прийнята до друку: 15.07.2026
Опубліковано: 30.10.2026
© 2026 Владарчик Ю. Л., Нестеренко К. С.
Цей матеріал ліцензовано за умовами CC BY 4.0.https://creativecommons.org/licenses/by/4.0/