DOI: https://doi.org/10.15276/ict.03.2026.29
УДК 004.75:004.8

Адаптивний метод розміщення сервісів у туманних обчислювальних системах на основі графової уваги та мультиагентного навчання з підкріпленням

Сбітнєв Олександр Юрійович[1)]
ORCID: https://orcid.org/0009-0008-6311-612X; alexsbitnev99@gmail.com
Волощук Людмила Арнольдівна[1)]
ORCID: https://orcid.org/0000-0002-2510-0038; lavstumbre@gmail.com
[1)] Одеський національний університет імені І. І. Мечникова, Одеса, Україна

АНОТАЦІЯ

У роботі представлено метод Adaptive Fog Architecture with Integrated Artificial Intellegence для побудови оптимальної архітектури туманної обчислювальної системи. Метод реалізує чотирикомпонентний конвеєр прийняття рішень: навчений шар графової уваги для структурного кодування топології мережі, мультиагентний контролер типу Multi-Agent Deep Deterministic Policy Gradient зі спільними вагами Actor-мережі та mean-field критиком для масштабованості на змінну кількість вузлів, адаптивний оркестратор на основі стохастичного Boltzmann-аукціону та модуль зворотного зв'язку, що навчає модель на фактично реалізованих результатах виконання підзадач, а не на апріорних оцінках. Метод реалізовано й оцінено в дискретно-подієвому симуляційному середовищі (SimPy) з реальними чергами обслуговування, контенцією пропускної здатності каналу, ланцюжковими залежностями підзадач і стохастичним процесом відмов вузлів. Порівняльна оцінка на восьми методах (представлений метод і сім базових ліній: Particle Swarm Optimization, Genetic Algorithm, Ant Colony Optimization, Machine Learning + Particle Swarm Optimization, Multi-Agent Reinforcement Learning, Federated Learning, Multi-Agent Reinforcement Learning + Graph Neural Networks + Federated Learning) у п'яти сценаріях під базовим і стрес-навантаженням показала, що представлений метод досягає найкращого серед усіх восьми методів показника балансування навантаження у чотирьох з п'яти сценаріїв базового режиму та конкурентної, хоча не найкращої, середньої затримки. Встановлено, що парадигма прийняття рішень (періодичне пакетне планування проти негайної онлайн-реакції) є домінантним фактором відмінності класів методів за затримкою, суттєвішим за вибір конкретного алгоритму оптимізації в межах однієї архітектури. Виявлено архітектурний компроміс: стохастичність аукціону, корисна для балансування навантаження за нормальних умов, знижує частку дотримання дедлайнів під екстремальним навантаженням. Незалежна holdout-перевірка початково запропонованого класифікатора класів методів не підтвердила гіпотезу про визначальну роль варіативності навантаження, що обґрунтовує необхідність її переформулювання на основі архітектурних параметрів системи, а не статистичних характеристик вхідного потоку.

Ключові слова: туманні обчислення; хмарні обчислення; дискретно-подієве моделювання; розміщення сервісів; графові мережі уваги; навчання з підкріпленням; мультиагентні системи; федеративне навчання; чисельне моделювання; машинне навчання; глибоке навчання; нейронна мережа; динамічне середовище; архітектура; динамічні характеристики

Для цитування: **Сбітнєв О. Ю., Волощук Л.А. "Адаптивний метод розміщення сервісів у туманних обчислювальних системах на основі графової уваги та мультиагентного навчання з підкріпленням ". *Інформатика. Культура. Техніка*. 2026; Том 3 № 1(3): 246–366. DOI: https://doi.org/10.15276/ict.03.2026.29**

Вступ

Актуальність. Упродовж останнього десятиліття інтернет речей (Internet of Things, IoT) перетворився з концептуальної парадигми на інженерну реальність, що безпосередньо визначає навантаження на телекомунікаційну інфраструктуру. За оцінками [1], кількість підключених IoT-пристроїв у 2023 році перевищила 15 млрд одиниць, і ця кількість, виходячи з темпів зростання за 2019-2023 роки, прогнозовано подвоїться до 2030 року. Це зростання не є рівномірним: значна його частина припадає на застосунки, критичні щодо затримки передачі даних, – промислову автоматизацію (IIoT), системи автономного руху (V2X), телеопераційну медицину та відеоспостереження. Для таких застосунків кінцева затримка є функціональним обмеженням: перевищення граничного значення у 20–50 мс для V2X або 100-150 мс для медичної телеметрії призводить до відмови системи в задоволенні вимог угоди про рівень обслуговування (Service Level Agreement, SLA).

Переважна більшість наявних IoT-розгортань продовжує використовувати централізовану хмарну архітектуру. Реальний час відгуку (round-trip time, RTT) до загальнодоступних хмарних провайдерів, навіть у щільно вкритих регіонах, становить

від 15 до 80 мс [2], оскільки включає затримки маршрутизації, чергування в мережевих вузлах, обробку протокольних заголовків і варіацію затримки на перевантажених ділянках. Додатковим обмеженням є обсяг трафіку: глобальний IP-трафік від IoT-пристроїв за прогнозами сягне 4,8 зеттабайт на рік до 2026 року, що ставить під сумнів пропускну здатність опорних мереж і обмежує масштабованість хмарно-центричної парадигми.

Відповіддю на цей виклик є концепція туманних обчислень, систематизована в роботі Bonomi та співавторів [3] як розширення парадигми хмарних обчислень до периферії мережі. Туманна інфраструктура передбачає розміщення обчислювальних, мережевих і сховищних ресурсів між IoT-пристроями і хмарою – на периферійних маршрутизаторах, базових станціях LTE/5G і промислових шлюзах.

Обчислювальною абстракцією такої інфраструктури є fog-вузол – гетерогенний периферійний сервер із характеристиками від малопотужного пристрою (обчислювальна потужність MIPS ≈ 3500-4500, потужність споживання P_active ≈ 5-8 Вт) до промислового edge-сервера (MIPS ≈ 6000-8000, P_active ≈ 100-120 Вт). [2] Просторова близькість fog-вузлів дозволяє скоротити кількість мережевих переходів (hop count) з 8-12 до 1-3 і знизити RTT до 5-20 мс. Обмеженість ресурсів окремого fog-вузла водночас породжує задачу оптимального розміщення сервісів.

Задача розміщення сервісів у fog-мережах є NP-складною в загальному випадку через зведення до узагальненої задачі призначення [4]. При N fog-вузлах і M завданнях простір допустимих рішень становить N^M; для N=20, M=100 це приблизно 10^130 варіантів, що виключає застосування точних методів у реальному часі. Практична складність задачі посилюється нестаціонарністю вхідного навантаження λ(t), топологічною неоднорідністю мережі (різна кількість переходів між парами вузлів) і регуляторними вимогами до конфіденційності IoT-даних (GDPR Art. 5, HIPAA §164.312).

Аналіз 21 первинних джерел дозволяє виокремити три системні прогалини в наявних підходах до розміщення сервісів у туманних системах.

По-перше, відсутній відтворюваний формальний критерій вибору класу методу оптимізації залежно від кількісних характеристик системи – вибір між метаевристиками, навчанням з підкріпленням і федеративними підходами здебільшого обґрунтовується емпірично для окремого сценарію, без узагальненого правила.

По-друге, у більшості симуляційних досліджень не враховується ефект черги обслуговування при коефіцієнті завантаженості ρ > 0,50, що призводить до систематичного заниження переваги онлайн-адаптивних методів порівняно з методами пакетного планування.

По-третє, відсутні гібридні архітектури, що поєднують топологічно усвідомлений вибір вузлів, онлайн-адаптацію і федеративне навчання в межах єдиного узгодженого методу.

Мета дослідження полягає у розробці та емпірична валідація методу AFAIA, спрямованого на усунення зазначених прогалин через:

(а) навчену архітектуру, що поєднує графову увагу для структурного кодування топології з мультиагентним контролером типу MADDPG та адаптивним оркестратором розв'язання конфліктів;

(б) дискретно-подієву симуляційну модель з реальними чергами обслуговування, що усуває методологічну упередженість порівняння класів методів, властиву аналітичним наближенням;

(в) емпірично обґрунтовану класифікаційну модель вибору класу методу, статус якої за результатами дослідження уточнено до рівня концептуальної, а не остаточно валідованої таксономії.

Завданнями дослідження є: розробка математичної моделі туманної системи з урахуванням стохастичних відмов вузлів і контенції мережевих ресурсів; реалізація алгоритму AFAIA з повним циклом навчання на реальних результатах виконання; порівняльна оцінка запропонованого методу із сімома базовими методами в ідентичних умовах симуляції; аналіз компромісів і меж застосовності отриманого рішення.

Огляд літератури

Найпростішим класом методів розміщення сервісів є детерміновані стратегії round-robin і least-loaded, обчислювальна складність яких становить O(N) на одне рішення. Стратегія round-robin не враховує гетерогенність вузлів і різнорідність завдань: обчислювально важкі задачі (наприклад, інференс моделей машинного навчання на периферії, 80-50 млн інструкцій) з рівною ймовірністю надходять як на слабкі вузли (MIPS ≈ 2000), так і на потужні (MIPS ≈ 8000), що спричиняє семикратний розкид у часі обробки. Стратегія least-loaded ігнорує мережеву топологію: вузол з мінімальною поточною завантаженістю, розташований на відстані 5-6 переходів, може мати сумарну затримку, вищу за помірно завантажений вузол на відстані 1-2 переходів. Lera та співавтори [5] показали, що політика розміщення сервісів, побудована на виявленні спільнот у графі fog-пристроїв, забезпечує вищий рівень дотримання дедлайнів і доступності застосунків при відмовах вузлів порівняно з підходом на основі цілочисельного лінійного програмування, який не враховує зв'язність мережі. Детерміновані стратегії не адаптуються до зміни інтенсивності надходження $\lambda(t)$: при подієво-індукованому трафіку, характерному для систем відеоспостереження, завантаженість вузлів наближається до $\rho \to 1$, що спричиняє неконтрольоване зростання черги.

Алгоритми роїнтелекту (Particle Swarm Optimization, PSO) та генетичні алгоритми (Genetic Algorithm, GA) є найдослідженішим класом методів для задачі fog-розміщення. Nguyen та співавтори [6] запропонували еволюційний алгоритм TCaS (Time-Cost aware Scheduling) для задачі розміщення Bag-of-Tasks застосунків у Cloud-Fog середовищі та показали, що він перевершує як модифікований PSO (на 11,04 %), так і Bee Life Algorithm (на 15,11%) за сукупним показником балансу часу виконання та вартості. Bitam та співавтори [7], своєю чергою, запропонували бджолиний алгоритм (Bees Life Algorithm, BLA) і продемонстрували, що він перевершує класичні PSO та GA за часом виконання CPU та обсягом використаної пам'яті – тобто в цій парі досліджень PSO виступає базовою лінією, яку перевершують спеціалізовані біоінспіровані альтернативи, а не еталоном, який сам перевершує детерміністичні стратегії. Спільним обмеженням еволюційних підходів (PSO, GA та їхніх похідних) є пакетна природа: за зміни навантаження раніше знайдений план оптимізації втрачає актуальність, а повне переобчислення в межах жорстких часових обмежень практично неможливе.

Подальший розвиток метаевристичних підходів включає ієрархічні багатоцільові схеми: Saif та співавтори [19] запропонували двоетапну стратегію, що спершу визначає обчислювальний рівень (edge/fog/cloud) евристичним алгоритмом, а потім застосовує багатоцільовий алгоритм світляків для вибору конкретного пристрою в межах рівня, домагаючись одночасного зниження енергоспоживання та затримки порівняно з одноетапними базовими лініями.

Алгоритм мурашиної колонії (Ant Colony Optimization, ACO) розроблявся для задач маршрутизації, у яких концентрація феромону на ребрі графа відображає якість маршруту – інтерпретація, природна для цього класу задач. У задачі призначення підзадач вузлам феромон, накопичений на парі «підзадача-вузол», втрачає таку природність: підвищена концентрація феромону, що відображає факт «підзадача і раніше успішно виконувалася на вузлі j», інформативна лише за умови стабільного навантаження. Kishor та Chakarbarty [21] застосували ACO-підхід (SACO) для задачі офлоадингу в fog computing і показали його перевагу над базовими стратегіями типу round-robin, throttled та MPSO, хоча ефективність такого підходу природно очікувано деградує за умов високої варіативності навантаження, оскільки феромонний слід, накопичений на попередніх ітераціях, відображає умови, які вже застаріли.

Гібридні підходи, що поєднують прогнозування навантаження з метаевристичною оптимізацією, представлені роботою Liu та співавторів [12] – фреймворком FRAHTOS, що інтегрує федеративне навчання з підкріпленням (Federated Reinforcement Learning) для задач

реального часу та гібрид PSO-GA для задач нереального часу, з прогнозуванням навантаження методом VARIMA. Підходи на основі навчання з підкріпленням (Deep Q-Network, мультиагентні actor-critic архітектури) вирішують задачу онлайн-адаптації без потреби в апріорному прогнозі навантаження. Jain та Kumar [15] запропонували підхід QoS-aware офлоадингу на основі мультиагентного глибокого навчання з підкріпленням (MAAC-алгоритм) для fog-середовища. Обмеженням підходів на основі RL є відсутність формальних гарантій диференційної приватності, оскільки агент потребує централізованого спостереження за станом усієї fog-мережі, що ускладнює відповідність вимогам GDPR і HIPAA.

Паралельний напрям зосереджується безпосередньо на дедлайн-орієнтованому плануванні: Zhang та співавтори [9] запропонували ієрархічну архітектуру edge-cloud з жадібною стратегією ранжування вузлів (TSGS), що враховує час обробки задачі та її чутливість до дедлайну, знизивши частку порушень SLA порівняно з round-robin базовою лінією. Li та співавтори [10] розв'язали аналогічну задачу для мобільних пристроїв через DRL-підхід на основі actor-critic (DRL-E2D) з дискретизацією дій методом k-найближчих сусідів, оптимізуючи енергоспоживання за умови жорсткого дедлайн-обмеження в multi-eNB середовищі. На відміну від [9], [10], у цьому дослідженні дедлайн трактується як м'яке обмеження функції нагороди (формула 7), а не тверде обмеження.

Nieto та співавтори [11] застосували розподілений DRL-підхід для децентралізованого бінарного рішення офлоадингу в 5G edge-cloud континуумі, підтвердивши практичність онлайн-адаптивних рішень у мережах з високою мобільністю користувачів, хоча запропонований підхід не враховує структуру топології мережі при прийнятті рішення.

Baek та Kaddoum [16] показали, що рекурентна Q-мережа (DRQN) дозволяє частково компенсувати неповноту спостереження глобального стану мережі в умовах multifog-топологій, перевершуючи стандартний DQN і згорткову DCQN-модель за середньою часткою успішних завдань. Це узгоджується з дизайном локального стану агента в AFAIA (формула 13), де кожен вузол приймає рішення на основі власних спостережень без централізованого доступу до повного стану системи.

Ijaz та співавтори [20] запропонували алгоритм планування на основі покращеного мультицільового методу диференціальної еволюції, що одночасно враховує часові та вартісні обмеження при мінімізації енергоспоживання в fog-обчисленнях, підтверджуючи загальну тенденцію переходу від одноалгоритмних до гібридних оптимізаційних схем у цьому класі методів.

Метод федеративного усереднення FedAvg [13] дозволяє навчати глобальну модель без централізації даних клієнтів через агрегацію локальних оновлень. Zhou та співавтори [14] дослідили застосування федеративного навчання з диференційною приватністю у fog computing, показавши можливість збереження конфіденційності локальних даних при прийнятній якості моделі. Обмеженням підходу є комунікаційні витрати на раунд агрегації, що накопичуються при багатораундовому навчанні [18]. У проаналізованих джерелах не виявлено гібридної архітектури, що поєднує топологічно усвідомлений вибір вузлів на основі графових нейронних мереж, онлайн-адаптацію засобами навчання з підкріпленням і диференційно-приватне федеративне навчання в межах єдиного узгодженого методу з архітектурно розділеними, але спільно навченими компонентами.

Метод AFAIA спрямований на усунення трьох виявлених прогалин через:

(а) концептуальну класифікаційну модель вибору класу методу за характеристиками системи, епістемічний статус якої уточнено за результатами незалежної holdout-перевірки (розділ «Обговорення»);

(б) єдину дискретно-подієву модель оцінки з реальними чергами обслуговування, контенцією каналу і стохастичними відмовами вузлів, що знижує методологічну упередженість порівняння класів методів порівняно з аналітичним наближенням;

(в) навчену архітектуру, що поєднує графову увагу, мультиагентний контролер типу MADDPG і адаптивний оркестратор, емпірично порівняну із сімома базовими методами

(охоплюючи гібрид MARL+GNN+FL) у єдиному узгодженому симуляційному середовищі. Загальний огляд застосувань мультиагентного навчання з підкріпленням для задач розподілу ресурсів [17] підтверджує зростання інтересу до цього класу методів у fog-та edge-обчисленнях, однак наявні дослідження здебільшого не поєднують MARL зі структурним кодуванням топології мережі через графову увагу – прогалину, яку покликаний усунути метод AFAIA.

Методи

Туманну систему описано графом G = (V, E, Ω), де V = {v₁,…,v_N} – множина fog-вузлів;, E – множина мережевих з'єднань; Ω: E → $\mathbb{R}_+$ – вагова функція затримок поширення. Граф E будується за моделлю Ердеша–Реньї G(n,p), доповненою примусово доданим гамільтоновим шляхом для гарантії зв'язності незалежно від реалізації випадкового графа (детальне обґрунтування цього прийому наведено в підрозділі «Побудова топології мережі» нижче). Матриця найкоротших відстаней dist(i,j) обчислюється алгоритмом обходу в ширину (BFS) з кожної вершини; реалізація використовує двосторонню чергу (collections.deque) замість видалення з голови списку, що знижує складність операції вилучення з O(n) до O(1) і загальну складність обчислення всіх попарних відстаней з $O(n^3)$ до $O(n^2)$.

Кожен вузол v_i є активною сутністю дискретно-подієвого середовища з власним стохастичним процесом готовності, а не статичним записом параметрів. Стан готовності up_i(t) керується двостановим процесом поновлення (alternating renewal process): тривалість перебування у стані готовності розподілена експоненційно з параметром MTBF_i, тривалість відновлення – експоненційно з параметром MTTR_i. Стаціонарний коефіцієнт готовності вузла становить A_i = MTBF_i/(MTBF_i+MTTR_i). Довжина черги обслуговування n_i(t) є емерджентною величиною симуляції, обчислюваною дискретно-подієвим планувальником, а не вхідним параметром моделі.

Таблиця 1 наводить повний перелік символів, використаних у математичній моделі та алгоритмі AFAIA, згрупованих за функціональним призначенням.

Таблиця 1. **Перелік символів та параметрів математичної моделі**

Граф і топологія мережі		
Символ	**Опис**	**Одиниця**
G = (V, E, Ω)	Граф туманної системи: вузли, з'єднання, ваги затримок	–
N	Кількість fog-вузлів у сценарії	од.
p	Імовірність ребра в моделі Ердеша–Реньї	–
$\bar{d}$	Середній ступінь вершини графа (щільність топології)	–
dist(i,j)	Найкоротша відстань між вузлами i та j у кількості переходів	переходів
δ	Затримка поширення сигналу на один перехід	мс/перехід
Параметри fog-вузла		
Символ	**Опис**	**Одиниця**
MIPS_i	Обчислювальна потужність вузла i	млн інстр./с
BW_i	Пропускна здатність каналу вузла i	Мбіт/с
P_i^active	Потужність споживання вузла i в активному стані	Вт
P_i^idle	Потужність споживання вузла i в стані очікування (= 0,65·P_i^active)	Вт
L_i^int	Внутрішня (апаратна) затримка обробки вузла i	мс
MTBF_i	Середній час напрацювання вузла i на відмову	мс
MTTR_i	Середній час відновлення вузла i після відмови	мс
up_i(t)	Булевий стан готовності вузла i в момент часу t	{0,1}
A_i	Стаціонарний коефіцієнт готовності вузла i	–
n_i(t)	Довжина черги обслуговування вузла i в момент t	од.
a_i(t)	Накопичений активний час обробки вузла i	мс
active_tx_i(t)	Кількість одночасних передач даних на вузол i в момент t	од.

Таблиця 1 (продовження)

Параметри задачі (підзадачі)		
Символ	**Опис**	**Одиниця**
τ_j	j-та підзадача обчислення	–
MI_j	Обсяг обчислень підзадачі τ_j	млн інстр.
KB_j	Обсяг переданих даних підзадачі τ_j	КБ
D_j	Дедлайн (гранично допустимий час виконання) підзадачі τ_j	мс
job_id_j	Ідентифікатор батьківської задачі (job), до якої належить τ_j	–
k	Довжина ланцюжка підзадач (chain-DAG) у межах однієї job	од., k∈{1,2,3}
L_req	Вимога до затримки для сценарію (використовується для D_j = 1,8·L_req)	мс
Модель надходження навантаження		
Символ	**Опис**	**Одиниця**
CV	Коефіцієнт варіації навантаження сценарію	–
cyclic_w(CV), chaotic_w(CV)	Вагові коефіцієнти змішування циклічного/хаотичного режимів	[0,1]
μ_stable, μ_cyclic, μ_chaotic	Середній обсяг обчислень MI для стабільного/циклічного/хаотичного режимів	млн інстр.
τ_ON, τ_OFF	Тривалість активного (ON) та пасивного (OFF) періодів самоподібного джерела	мс
α	Параметр форми розподілу Парето для τ_ON, τ_OFF	–
H	Параметр Херста самоподібності трафіку, H=(3−α)/2	–
base_rate	Номінальна інтенсивність надходження задач	задач/с
Модель затримки та енергії		
Символ	**Опис**	**Одиниця**
T_total(τ_j)	Сумарна наскрізна затримка виконання підзадачі τ_j	мс
T_arrival, T_finish	Момент надходження та момент завершення підзадачі	мс
rate_i(t)	Ефективна швидкість передачі даних на вузол i в момент t	Мбіт/с
E_i	Сумарне енергоспоживання вузла i за час симуляції	Дж
eff	Метрика енергоефективності (завершених задач на кілоджоуль)	задач/кДж
Цільова функція batch-методів		
Символ	**Опис**	**Одиниця**
a: {1..M}→{1..N}	Функція призначення M підзадач N вузлам	–
F(a)	Скаляризована цільова функція розподілу a	–
$\bar{L}$(a)	Середня затримка за призначення a	мс
DS(a)	Частка дотримання дедлайну за призначення a (deadline satisfaction)	%
control_interval	Інтервал періодичного перепланування batch-методів	мс
Функція нагороди AFAIA		
Символ	**Опис**	**Одиниця**
R_i(τ_j)	Миттєва нагорода вузла i за виконання підзадачі τ_j	–
Imbalance_i(t)	Міра дисбалансу накопиченого навантаження вузла i відносно середнього	–
β_bal	Вага штрафу за дисбаланс навантаження у функції нагороди	–
ε	Бюджет диференційної приватності FL-компонента	–
Δf	Чутливість внеску одного клієнта до агрегованого вектора ваг	–
b	Масштаб шуму Лапласа, b = Δf/ε	–
GAT-енкодер		
Символ	**Опис**	**Одиниця**
h_i, h_i′	Вхідний та вихідний (навчений) вектор ознак вузла i	–
W	Навчана матриця лінійної проєкції GAT	–
a_src, a_dst	Навчані вектори уваги для джерела та цілі ребра	–
e_ij	Ненормований коефіцієнт уваги між вузлами i та j	–

Таблиця 1 (продовження)

Символ	Опис	Одиниця
α_ij	Нормований (softmax) коефіцієнт уваги	[0,1]
Â = A + I	Матриця суміжності графа із доданими самопетлями	–
N(i)	Множина сусідів вузла i в графі	–
MADDPG-контролер		
Символ	**Опис**	**Одиниця**
π_θ	Детермінована політика Actor із параметрами θ	–
s_i	Вектор локального стану вузла i	–
φ(τ_j)	Вектор ознак підзадачі τ_j	–
bid_i	Заявка (оцінка готовності) вузла i виконати підзадачу	(0,1)
Q(·)	Функція цінності Critic (mean-field наближення)	–
ā_{-i}, max(a_{-i})	Середнє та максимальне значення заявок вузлів-конкурентів	(0,1)
n_live	Кількість живих (доступних) вузлів у момент рішення	од.
L_critic, L_actor	Функції втрат Critic та Actor	–
Адаптивний оркестратор		
Символ	**Опис**	**Одиниця**
P(winner = i)	Імовірність перемоги вузла i в аукціоні розподілу	[0,1]
τ (температура)	Параметр Boltzmann-розподілу; τ_train = 0,20, τ_eval = 0,08	–
Параметри експерименту		
Символ	**Опис**	**Одиниця**
N_runs	Кількість незалежних прогонів на конфігурацію	од.
ρ_system	Рівень насичення (інтегральної завантаженості) системи	–

Джерело: **розроблено авторами**

Модель надходження навантаження

Режими навантаження змішуються неперервно за формулами:

$$\text{cyclic\_w(CV)} = clip\left(\frac{cv-0{,}15}{0{,}50}, 0, 1\right), \quad (1)$$

$$chaotic\_\text{w(CV)} = clip\left(\frac{cv-0{,}55}{0{,}40}, 0, 1\right), \quad (2)$$

$$\mu\_ph = \mu\_stable \cdot (1 - cyclic\_w) + \mu\_cyclic \cdot cyclic\_w \cdot (1 - chaotic\_w) + \mu\_chaotic \cdot chaotic\_w \quad (3)$$

де: μ_stable = 50 (константа); μ_cyclic = 50·(1+0,5·sin(ph·π/2)·CV); μ_chaotic = 50·U(0,35; 2,30); U – рівномірний розподіл. Точки початку змішування (0,15 і 0,55) зсунуті відносно порогів класифікаційної моделі (0,30 і 0,70), що знижує ризик циркулярної залежності між генератором навантаження і моделлю, яку цей генератор мав би валідувати.

Надходження задач моделюється модульованим ON/OFF-джерелом із розподілом Парето для тривалостей активного та пасивного періодів: τ_ON, τ_OFF ~ Pareto(α=1,5). Протягом активного періоду задачі надходять за пуассонівським процесом з інтенсивністю, утричі підвищеною відносно номінальної base_rate, що компенсує нульову інтенсивність у пасивному періоді для збереження довгострокового середнього значення. Параметр форми α = 1,5 відповідає параметру Херста самоподібності H = (3−α)/2 = 0,75, що узгоджується з емпіричними значеннями H ∈ [0,7; 0,9], встановленими для реального Ethernet/IoT-трафіку [25]. Контрольною умовою залишено чистий пуассонівський процес надходження для порівняння впливу структури трафіку на результати.

Модель затримки та енергоспоживання

Черга обслуговування є емерджентною властивістю симуляції: кожен вузол реалізовано як однопотоковий ресурс з дисципліною обслуговування FIFO, час очікування в якому обчислюється дискретно-подієвим планувальником із фактичного стану черги.

$$T\_total(\tau\_j) = T\_finish(\tau\_j) - T\_arrival(\tau_j). \quad (4)$$

Значення T_finish визначається послідовністю подій: (1) очікування звільнення обробника вузла; (2) за умови відмови вузла на момент отримання черги – очікування відновлення з кроком опитування 2 мс; (3) мережева передача з урахуванням контенції пропускної здатності за ефективною швидкістю rate_i(t) = BW_i / max(active_tx_i(t), 1), яка фіксується на весь час передачі за принципом «розподілу пропускної здатності на момент прийняття»; (4) власне обробка тривалістю MI_j/MIPS_i·1000 мс. Формальні обмеження задачі: дедлайн – м'яке обмеження T_total(τ_j) ≤ D_j, порушення якого штрафується у функції нагороди, а не блокує виконання; доступність – призначення допустиме лише за умови up_i(t)=1 у момент ухвалення рішення; приватність – бюджет диференційної приватності ε для FL-компонента.

Енергетична модель враховує обидві складові споживання:

$$E\_i \;=\; P\_i\hat{}active \cdot (a\_i\hat{}active/1000) \;+\; P\_i\hat{}idle \cdot (a\_i\hat{}idle/1000)\,, \qquad (5)$$

де a_i^active – накопичений час активної обробки, a_i^idle = T_sim − a_i^active – час очікування вузла протягом симуляції.

Цільова функція та функція нагороди

Для чотирьох базових методів пакетного планування (PSO, GA, ACO, ML+PSO), що періодично переплановують чергу ще не призначених підзадач з інтервалом control_interval, використано статичну скаляризовану цільову функцію:

$$F(a) \;=\; L(a) \;+\; (1 \;-\; DS(a)/100) \cdot 500\,. \qquad (6)$$

Для AFAIA цільова функція реалізована як функція миттєвої нагороди, обчислювана апостеріорі – після фактичного завершення підзадачі:

$$R\_i(\tau\_j) \;=\; -T\_total(\tau\_j)/100 \;+\; \mathbb{1}[T\_total(\tau\_j) \le D\_j] \;-\; \beta\_bal \cdot Imbalance\_i(t)\,, \qquad (7)$$

$$Imbalance\_i(t) \;=\; max(0, a\_i(t) \;-\; mean\_\{k{:}\, up\_k = 1\}\, a\_k(t)) \;/\; 1000,\; \beta\_bal \;=\; 0{,}3, \qquad (8)$$

Формальне обмеження диференційної приватності [26] для FL-компонента реалізовано механізмом Лапласа: масштаб шуму b = Δf/ε, де Δf = 1/N – чутливість внеску одного клієнта до нормованого вектора глобальних ваг. Бюджет приватності визначається полем сценарію: 'full' → ε=1,0 (сильна приватність), 'partial' → ε=4,0, 'none' → механізм Лапласа не застосовується.

Архітектура методу AFAIA

AFAIA реалізує чотирикомпонентний конвеєр прийняття рішень. GAT-енкодер обчислює структурний ембединг кожного вузла на основі топології графа за формулами графової уваги (Veličković та співавт [22]):

$$Wh\_i = W \cdot h\_i,\; e\_ij = LeakyReLU(a_{src}^T \cdot Wh\_i + a_{dst}^T \cdot Wh\_j), \text{для } (i,j){:}\; \hat{A}\_ij = 1, \qquad (9)$$

$$\alpha\_ij = softmax\_j(e\_ij) = \frac{exp(e\_ij)}{\sum_{k\in N(i)} exp(e\_ik)}, \qquad (10)$$

$$h\_i' \;=\; ELU(\textstyle\sum_{j\in N(i)} \alpha\_ij \cdot Wh\_j)\,. \qquad (11)$$

Реалізація двошарова: вхідний вимір 3 (нормовані MIPS, ступінь вершини, константна ознака зсуву) → прихований шар розмірності 16 → вихідний ембединг розмірності 8.

MADDPG-контролер [23] оцінює готовність кожного вузла-агента і взяти підзадачу τ_j через спільну (parameter-shared) детерміновану політику Actor:

$$\pi\_\theta(s\_i, \tau\_j) \;=\; \sigma(MLP(s\_i \,\oplus\, h\_i' \,\oplus\, \varphi(\tau\_j))) \in (0{,}1) \qquad (12)$$

$$s\_i \;=\; (MIPS\_i/max, n\_i(t)/6, up\_i(t), active\_tx\_i(t)/3, fair\_deficit\_i(t)) \qquad (13)$$

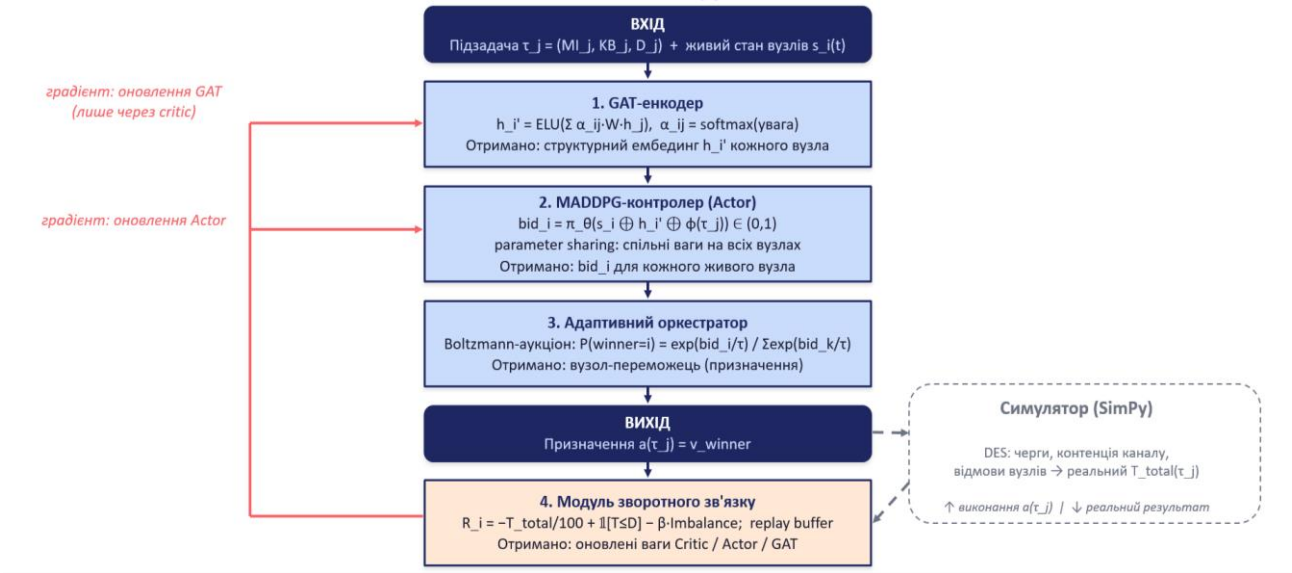

Рис. 1. **Блок-схема методу AFAIA**
Джерело: **розроблено авторами**

де п'ята компонента локального стану fair_deficit_i(t) = clip((a_i(t) − mean_{k:up_k=1} a_k(t))/2000, −3, 3) надає агенту явний сигнал справедливості розподілу навантаження на вході рішення. Critic оцінює якість пари «стан-дія» через mean-field наближення (Yang та співавт [24]), що замінює конкатенацію дій усіх N агентів на агреговані статистики конкурентів:

$$Q(s\_i, a\_i, \bar{a}\_\{-i\}, max(a\_\{-i\}), n\_live) = MLP(s\_i \oplus h\_i' \oplus \varphi(\tau\_j) \oplus a\_i \oplus \bar{a}\_\{-i\} \oplus \oplus max(a\_\{-i\}) \oplus n\_live)\,. \quad (14)$$

Адаптивний оркестратор вирішує конфлікт між заявками через стохастичний Boltzmann-аукціон:

$$P(winner = i) = \frac{exp\left(\frac{bid\_i}{\tau}\right)}{\sum_{k:\, up_k=1} exp\left(\frac{bid\_k}{\tau}\right)}. \quad (15)$$

Модуль зворотного зв'язку обчислює реальну нагороду R_i(τ_j) після фактичного завершення підзадачі й ініціює градієнтне оновлення Critic (однокрокова регресія L_critic=(Q(s,a,·)−R)²), Actor (детермінований градієнт політики L_actor=−E[Q(s,π_θ(s),·)]) та GAT (через окремий оптимізатор, що застосовує градієнт критика до GAT-параметрів до виклику zero_grad() Actor-оптимізатора).

Алгоритм 1. Цикл прийняття рішення AFAIA

Нижче наведено покроковий опис алгоритму в порядку, що дозволяє повне відтворення обчислювального експерименту.

Крок 1. Побудувати граф топології G(V,E) за моделлю Ердеша–Реньї з параметром щільності d̄; додати гамільтонів шлях для гарантії зв'язності; обчислити матрицю dist(i,j) алгоритмом BFS.

Крок 2. Ініціалізувати параметри кожного вузла v_i: MIPS_i, BW_i, P_i^active, MTBF_i, MTTR_i; розрахувати похідні величини P_i^idle, A_i.

Крок 3. Обчислити початкові структурні ознаки h_i кожного вузла (нормовані MIPS, ступінь вершини, константа зсуву).

Крок 4. Виконати прямий прохід GAT-енкодера для всієї топології: отримати h_i′ для кожного вузла (формули підрозділу «Архітектура методу AFAIA»); ця операція виконується один раз на епізод симуляції, оскільки топологія й MIPS вузлів статичні протягом епізоду.

Крок 5. Ініціювати дискретно-подієвий цикл надходження задач (модульоване ON/OFF-джерело Парето або пуассонівський процес).

Крок 6. При надходженні підзадачі τ_j: зібрати живий стан кожного вузла $s\_i(t)$ (завантаженість черги, статус готовності, кількість активних передач, дефіцит справедливості).

Крок 7. Для кожного живого вузла обчислити заявку $bid\_i = \pi\_\theta(s\_i, h\_i', \varphi(\tau\_j))$ прямим проходом Actor-мережі.

Крок 8. Замаскувати заявки вузлів у стані відмови значенням $bid\_i = -10^9$.

Крок 9. Обчислити розподіл імовірностей Boltzmann над живими вузлами з температурою τ_train (під час навчання) або τ_eval (під час інференсу) і семплювати переможця аукціону.

Крок 10. Передати підзадачу переможцю; виконати симуляцію обслуговування в дискретно-подієвому середовищі: черга обробника, контенція каналу, власне обчислення.

Крок 11. Після завершення підзадачі обчислити фактичну затримку $T\_total(\tau\_j)$, індикатор дотримання дедлайну, і на їх основі – нагороду $R\_i(\tau\_j)$.

Крок 12. Зберегти перехід (локальний стан на момент рішення, ознаки задачі, індекс переможця, власна заявка, агреговані заявки конкурентів, кількість живих вузлів, нагорода) у буфері відтворення.

Крок 13. Кожні чотири переходи, за умови накопичення щонайменше 64 переходів у буфері: вибрати випадковий підбатч, оновити Critic мінімізацією L_critic; застосувати отриманий градієнт до GAT-параметрів через окремий оптимізатор; оновити Actor мінімізацією L_actor з відсіченим (detached) градієнтом GAT-ембедингу.

Крок 14. Перейти до кроку 6 для наступної підзадачі; продовжувати до завершення часового горизонту симуляції.

Гіперпараметри навчання: швидкість навчання Actor і GAT – $3 \cdot 10^{-4}$, швидкість навчання Critic – $1 \cdot 10^{-3}$ (оптимізатор Adam); обрізання норми градієнта на рівні 5,0 окремо для Critic, GAT і Actor; явна перевірка скінченності функції втрат перед кроком оптимізації (за виявлення NaN або Inf крок оновлення пропускається). Ці заходи введено після емпіричного спостереження поодиноких стрибків функції втрат Critic до NaN на епізодах з екстремальним навантаженням, коли реалізована затримка (а отже й нагорода) набувала аномально великих за модулем значень.

Результати. Оцінку методу виконано засобами чисельного моделювання на основі бібліотеки SimPy 4.1. Модельний час вимірюється безперервно в мілісекундах. Надходження задач моделювалося модульованим ON/OFF-джерелом Парето. Для чотирьох методів пакетного планування застосовано інтервал перепланування control_interval = 20 мс, обраний за результатами чисельного моделювання чутливості до цього параметра (control_interval ∈ {1, 5, 20, 60, 500} мс): при control_interval = 500 мс середня затримка методу PSO становила 11,90 мс, при control_interval = 1 мс – практично збігалася зі значенням методу MARL (11,97 мс) на тому самому сценарії. Параметри відмов вузлів: MTBF = 15000 мс, MTTR = 1500 мс. Кожна конфігурація (сценарій × метод × режим навантаження) виконувалася в N_runs = 4 (базовий режим) або N_runs = 3 (стрес-режим) незалежних прогонах тривалістю 6–8 с модельного часу. Для восьми методів (PSO, GA, ACO, ML+PSO, MARL, FL, MARL+GNN+FL, AFAIA) зафіксовано чотири метрики: середня наскрізна затримка $\bar{L}$, частка дотримання дедлайну DS, пропускна здатність (throughput, кількість завершених підзадач за секунду модельного часу) та util_std – стандартне відхилення частки активного часу між вузлами.

Таблиця 3 наводить середню затримку $\bar{L}$ (мс) та коефіцієнт util_std для восьми методів на п'яти сценаріях базового навантаження (30 задач/с). Частка дотримання дедлайну DS дорівнювала 100% у більшості комбінацій метод-сценарій, за винятком ACO (діапазон 57-88 % залежно від сценарію) та методів AFAIA і FL у сценарії Industrial IoT Edge (99,8 %).

Таблиця 2. **Параметри сценаріїв симуляції**

№	Назва сценарію	N, од.	CV	L_req, мс	Приватність	$\bar{d}$	control_interval, мс
1	Smart Factory	15	0,20	200	none	4,0	20
2	Healthcare Monitoring	35	0,50	100	full	2,5	20
3	Smart City	80	0,85	150	partial	7,0	20
4	Industrial IoT Edge	25	0,72	50	none	5,0	20
5	Traffic Network	45	0,45	300	none	3,0	20

Джерело: **розроблено авторами**

Таблиця 3. **Середня затримка (мс) та util_std для восьми методів за базового навантаження**

Сценарій	PSO	GA	ACO	ML+PSO	MARL	FL	MARL+GNN+FL	AFAIA
Smart Factory	27,1 / 0,058	28,1 / 0,055	606,0 / 0,119	27,3 / 0,041	15,8 / 0,040	23,7 / 0,026	16,6 / 0,031	22,9 / 0,023
Healthcare Monitoring	24,3 / 0,026	25,1 / 0,037	346,6 / 0,050	25,0 / 0,026	15,3 / 0,028	21,7 / 0,015	15,5 / 0,028	21,3 / 0,010
Smart City	27,1 / 0,023	26,2 / 0,033	232,8 / 0,054	25,9 / 0,015	17,7 / 0,023	22,6 / 0,009	17,7 / 0,011	22,9 / 0,007
Industrial IoT Edge	26,5 / 0,031	26,9 / 0,050	663,5 / 0,066	27,9 / 0,035	16,7 / 0,035	24,4 / 0,027	17,7 / 0,032	24,6 / 0,020
Traffic Network	25,8 / 0,026	25,5 / 0,023	231,5 / 0,038	26,0 / 0,020	16,6 / 0,024	20,2 / 0,009	17,0 / 0,027	20,2 / 0,010

Джерело: **розроблено авторами**

У кожній клітинці таблиці 3 наведено пару значень «середня затримка $\bar{L}$ (мс) / util_std», розділених символом «/». MARL є одноосібним мінімумом у сценаріях Smart Factory,

Healthcare Monitoring, Industrial IoT Edge і Traffic Network; MARL+GNN+FL ділить мінімум із MARL лише у сценарії Smart City (17,7 мс в обох). Мінімальне значення util_std у кожному рядку належить AFAIA у чотирьох сценаріях (Healthcare Monitoring – 0,010; Smart City – 0,007; Traffic Network – 0,010 разом з FL) і методу FL у сценарії Traffic Network (0,009).

Рис. 2 та Рис.3 наводять результати порівняння восьми методів за середньою затримкою на п'яти сценаріях базового навантаження.

По осі ординат на Рис. 2 – затримка в мілісекундах; кольором позначено метод (легенда праворуч); штрихова горизонтальна лінія відповідає значенню L_req сценарію; золота рамка виділяє стовпець із мінімальним значенням затримки в межах сценарію.

Як показано на Рис. 2, ACO демонструє аномально високу затримку в усіх сценаріях (233-663 мс проти 16-28 мс у решти методів), що робить пряме порівняння інших методів на лінійній шкалі візуально складним. Для усунення цього ефекту масштабу результати додатково представлено у вигляді нормованої в межах кожного рядка теплової карти (Рис. 3), де темно-зелений колір відповідає мінімальному значенню затримки в межах сценарію, темно-червоний – максимальному; числові підписи – вихідні значення затримки в мілісекундах; золота рамка виділяє клітинку з мінімальним значенням у рядку. Рис. 3 демонструє, що MARL забезпечує найнижчу затримку в чотирьох з п'яти сценаріїв, поступаючись лише у сценарії Smart City.

Рис.4, Рис. 5 та Рис. 6 наводять результати за трьома додатковими вимірами порівняння методів, визначеними класифікаційною моделлю (розділ «Методи»): енергоефективністю, масштабованістю та чутливістю до коефіцієнта варіації навантаження.

На Рис. 4 по осі ординат – значення метрики; групування стовпців – за сценарієм; кольором позначено метод.

На Рис. 5 ліва частина - затримка задачі (мс, логарифмічна шкала не застосована) для семи методів без ACO; права частина - реальний час обчислення одного прогону (wall-clock, секунди) для восьми методів. По осі абсцис обох частин – кількість вузлів N.

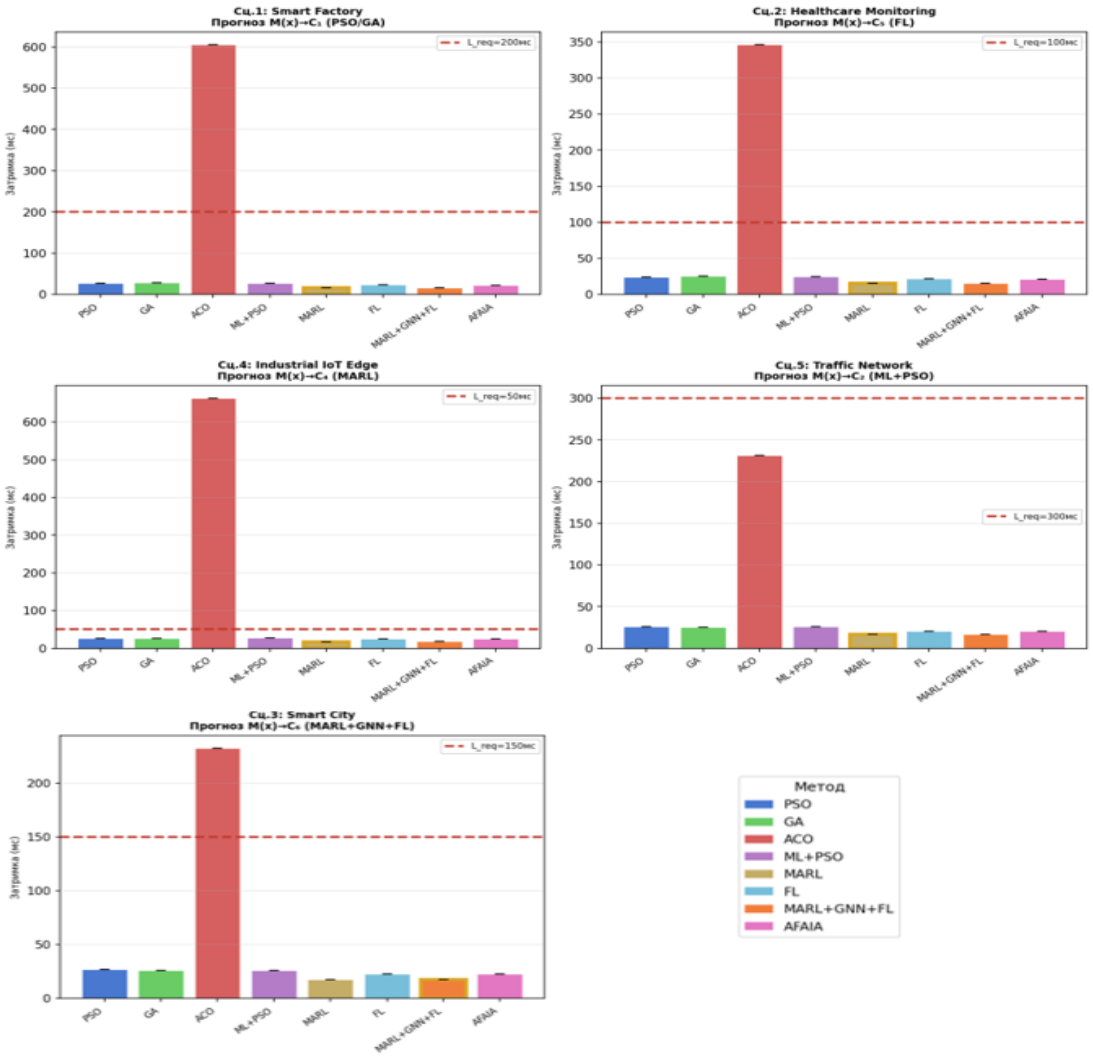

Рис. 2. **Середня затримка восьми методів на п'яти сценаріях базового навантаження**

Джерело: **розроблено авторами**

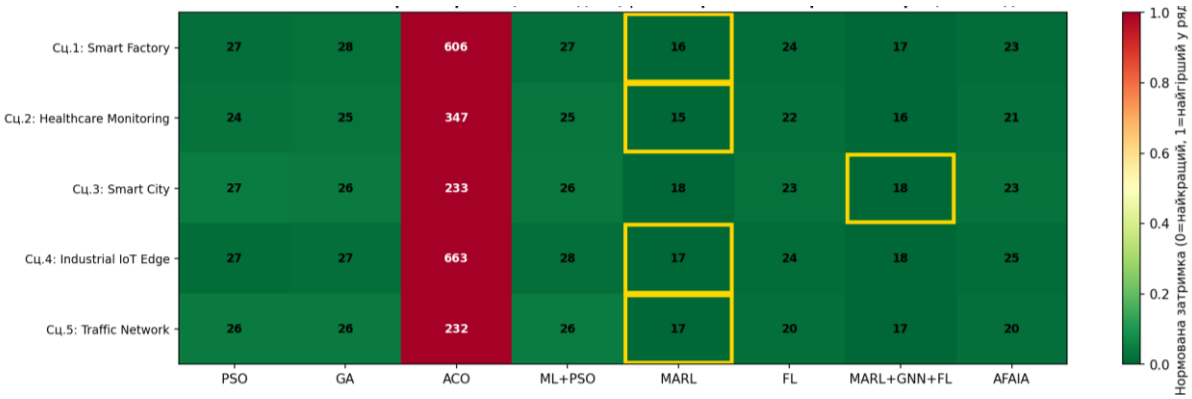

Рис. 3. **Теплова карта середньої затримки, нормованої в межах кожного рядка (сценарію) до діапазону [0, 1].**

Джерело: **складено авторами**

Енергоефективність (кількість завершених задач на кілоджоуль сумарного споживання) для сценарію Smart Factory становила: GA – 112,0; PSO – 95,3; ACO – 81,4; AFAIA – 81,4; ML+PSO – 80,0; FL – 74,6; MARL+GNN+FL – 70,3; MARL – 92,5. Для сценарію Industrial IoT Edge найвище значення зафіксовано для FL (65,4), для сценарію Smart City – для MARL (19,6 за даними експерименту).

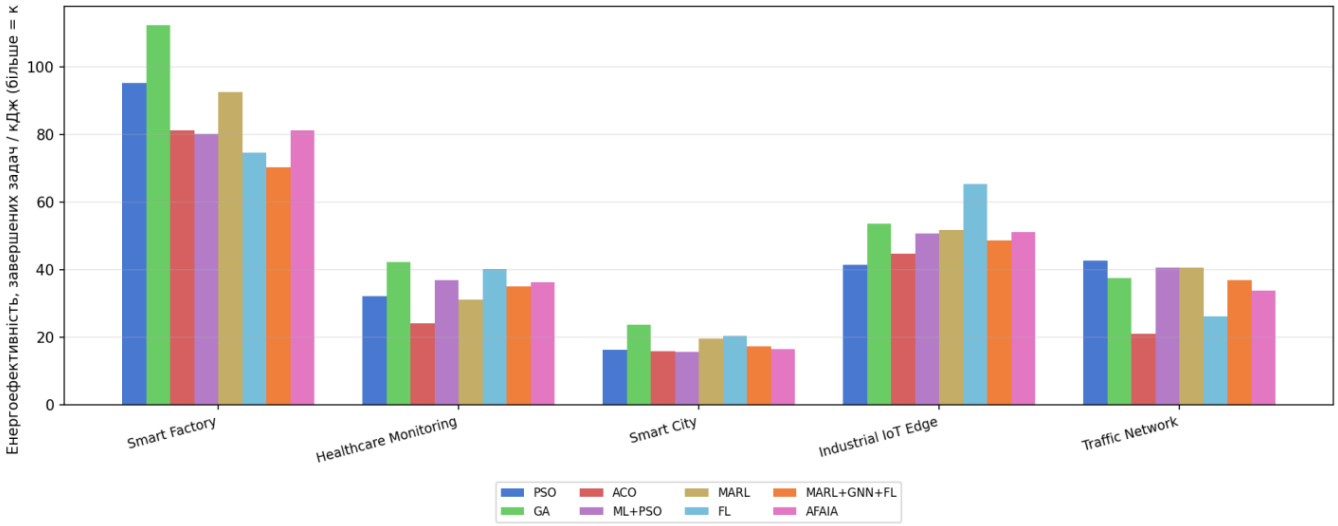

Рис. 4. **Енергоефективність (завершених задач на кілоджоуль) восьми методів на п'яти сценаріях базового навантаження**

Джерело: **складено авторами**

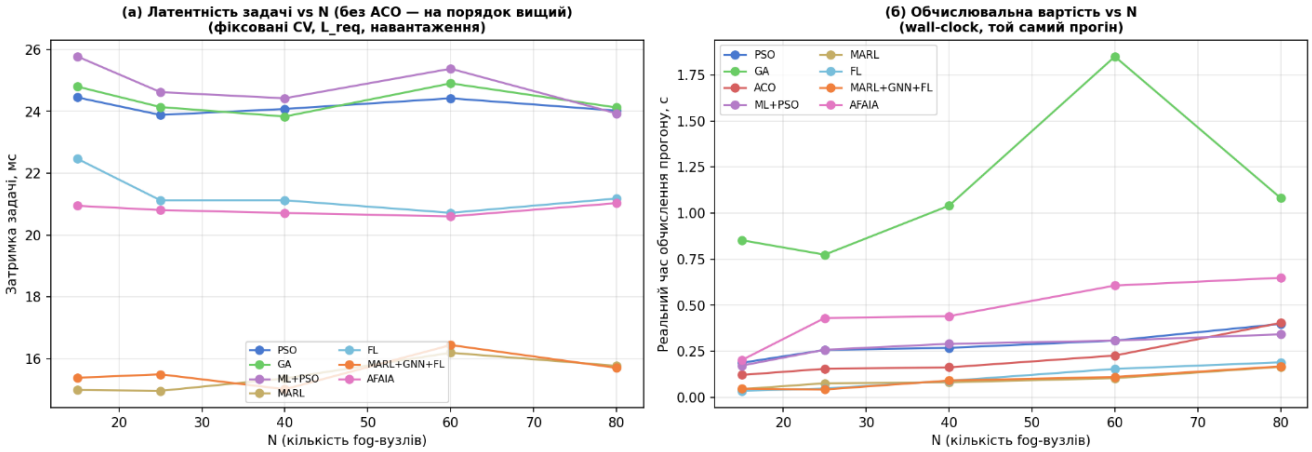

Рис. 5. **Масштабованість методів за кількістю вузлів N**

Джерело: **розроблено авторами**

Масштабованість перевірено окремим контрольованим експериментом зі сценарієм, у якому коефіцієнт варіації навантаження (CV=0,5), вимога до затримки (L_req=150 мс) та рівень приватності зафіксовані, а кількість вузлів N варіюється в діапазоні {15, 25, 40, 60, 80}. Затримка задачі для семи методів (без ACO) залишалася в діапазоні 14,9-25,8 мс на всьому діапазоні N без монотонної залежності від N. Реальний час обчислення одного прогону (wall-clock), виміряний окремо, зростав для методу AFAIA з 0,202 с (N=15) до 0,649 с (N=80) та для методу GA – з 0,853 с (N=15) до 1,082 с (N=80, з локальним максимумом 1,850 с при N=60). Для методів MARL і MARL+GNN+FL час обчислення становив 0,044-0,166 с та 0,047–0,168 с відповідно на всьому діапазоні N.

Чутливість затримки до коефіцієнта варіації навантаження перевірено на п'яти базових сценаріях, упорядкованих за зростанням CV (0,20; 0,45; 0,50; 0,72; 0,85). Для семи методів (без ACO) різниця між максимальним і мінімальним значенням затримки в межах одного методу на всьому діапазоні CV не перевищувала 3,8 мс (метод ML+PSO: від 24,3 до 27,9 мс).

Стрес-режим отримано підвищенням інтенсивності надходження задач до рівня, за якого середня утилізація вузлів перевищує 4050 %.

Таблиця 4 наводить результати для трьох повністю виконаних конфігурацій стрес-навантаження.

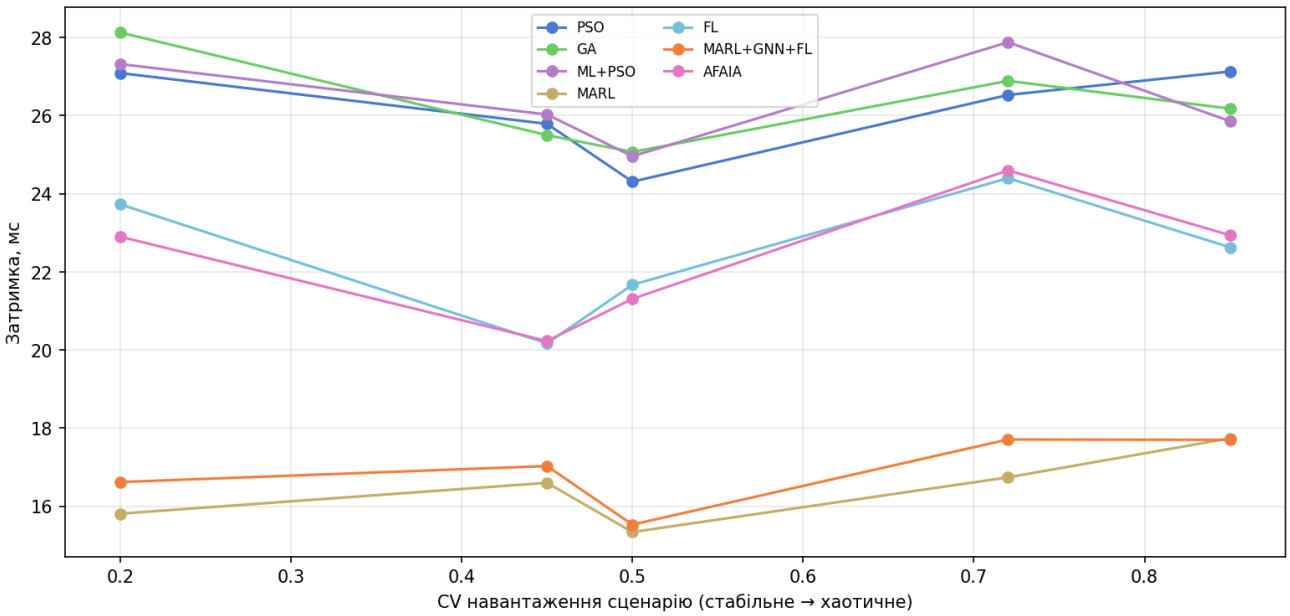

Рис 6. **Залежність середньої затримки від коефіцієнта варіації навантаження CV для семи методів (без ACO)**

Джерело: розроблено авторами

Таблиця 4. **Середня затримка (мс) та частка дотримання дедлайну DS (%) під стрес-навантаженням**

Метод	Smart Factory, 300 задач/с: L̄ / DS	Smart Factory, 600 задач/с: L̄ / DS	Smart City, 1500 задач/с: L̄ / DS
PSO	213,0 / 84,9 %	794,5 / 43,9 %	250,8 / 77,4 %
GA	228,6 / 84,1 %	1069,3 / 25,8 %	364,1 / 55,8 %
ACO	1480,3 / 38,2 %	1531,8 / 30,9 %	818,5 / 59,1 %
ML+PSO	212,4 / 86,7 %	1014,5 / 30,6 %	238,1 / 79,1 %
MARL	138,5 / 94,9 %	353,5 / 86,9 %	238,5 / 87,8 %
FL	381,3 / 69,5 %	780,2 / 36,1 %	393,3 / 61,3 %
MARL+GNN+FL	125,0 / 96,0 %	448,1 / 83,2 %	232,0 / 84,7 %
AFAIA	453,9 / 60,5 %	1060,6 / 21,5 %	326,5 / 65,5 %

Джерело: **розроблено авторами**

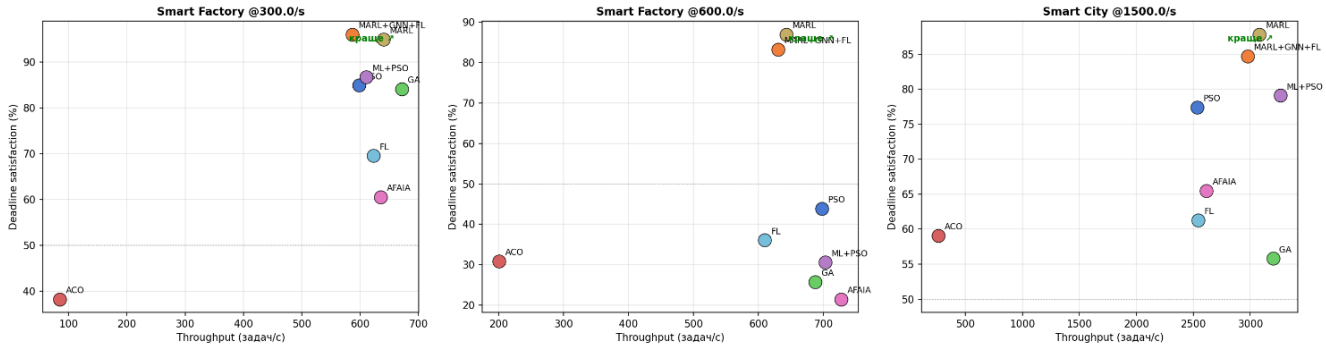

Рис. 7. **Співвідношення пропускної здатності (throughput, вісь абсцис, задач/с) та частки дотримання дедлайну (DS, вісь ординат, %) на трьох стрес-конфігураціях**

Джерело: **розроблено авторами**

Прогноз класифікаційної моделі перевірено на значеннях CV, відсутніх серед п'яти базових сценаріїв (CV ∈ {0,10; 0,35; 0,60; 0,90}). Мінімальну затримку на кожній з чотирьох перевірених точок демонстрував метод GA. Пряме порівняння прогнозованого класу з емпірично найкращим методом на п'яти базових сценаріях (без застосування композитної метрики) дало збіг у двох випадках з п'яти: сценарій Smart City (прогноз MARL+GNN+FL, емпіричний результат MARL+GNN+FL) та сценарій Industrial IoT Edge (прогноз MARL, емпіричний результат MARL).

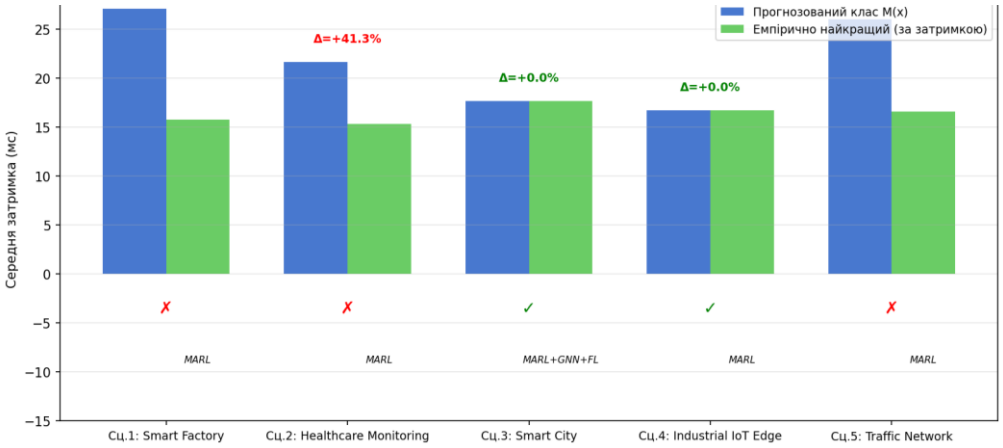

Рис. 8. **Порівняння прогнозованого класифікаційною моделлю методу з емпірично найкращим методом за затримкою на п'яти базових сценаріях**

Джерело: **розроблено авторами**

Обговорення

Результати Таблиці 3, Рис. 1 та Рис. 2 показують стійку перевагу методів MARL і MARL+GNN+FL за середньою затримкою в усіх п'яти сценаріях базового навантаження. Контрольований експеримент із варіюванням control_interval (сценарій Smart Factory, чотири методи) показав, що ця перевага звужується в міру зменшення інтервалу перепланування: за control_interval = 500 мс середня затримка PSO становила 364,57 мс проти 13,97 мс для MARL, тоді як за control_interval = 0,1 мс – 15,00 мс проти тих самих 13,97 мс, а для методу GA (14,86 мс) різниця з MARL змінила знак. Метод ACO на тому самому діапазоні control_interval зберігав затримку в межах 205,92–1016,22 мс незалежно від значення параметра. Отримана залежність інтерпретується як свідчення того, що переважна частка різниці в затримці між методами пакетного планування (PSO, GA, ML+PSO) і методами онлайн-прийняття рішень (MARL, FL, MARL+GNN+FL, AFAIA) визначається архітектурою прийняття рішень – очікуванням на цикл перепланування, середня тривалість якого становить control_interval/2, – а не якістю алгоритму оптимізації в межах цієї архітектури. Стійкість затримки ACO до зміни control_interval, навпаки, вказує на те, що обмеження цього методу має алгоритмічну, а не архітектурну природу: евристика, що спирається виключно на обчислювальну потужність вузла без урахування обсягу обчислень конкретної підзадачі, систематично приймає гірші рішення незалежно від частоти перепланування.

Значення control_interval = 20 мс, застосоване в основному порівнянні, обране як компромісний варіант між крайнім випадком control_interval → 0 (за якого відмінність між пакетною і онлайн-архітектурами практично зникає, що робить порівняння архітектур позбавленим сенсу) і орієнтовним значенням 500 мс, характерним для практики періодичного перепланування в системах керування на основі моделі (Model Predictive Control) для периферійних обчислень. Вибір конкретного значення control_interval є параметром дослідницького дизайну, що безпосередньо впливає на кількісне співвідношення методів, і має розглядатися як контрольована змінна експерименту, а не фіксована властивість порівнюваних методів.

Дані Таблиці 4 та Рис. 6 виявляють компроміс, не описаний у розглянутих першоджерелах для системи з ідентичними умовами порівняння. Методи PSO і ML+PSO демонструють вищу пропускну здатність порівняно з іншими методами в більшості конфігурацій стрес-навантаження за одночасного зниження частки дотримання дедлайну до 25,8-43,9 % за інтенсивності 600 задач/с. Методи MARL і MARL+GNN+FL зберігають частку дотримання дедлайну в діапазоні 83,2-96,0 % на всьому діапазоні перевіреного навантаження. AFAIA демонструє показник дотримання дедлайну 60,5% (300 задач/с), 21,5 % (600 задач/с) та 65,5 % (1500 задач/с), що на всіх трьох конфігураціях нижче відповідних значень MARL і MARL+GNN+FL, попри перевагу AFAIA за util_std у базовому режимі навантаження (Таблиця 3). Метод GA демонструє немонотонну залежність DS від інтенсивності навантаження зі зниженням до 25,8 % за 600 задач/с при одному з найвищих значень throughput серед перевірених методів на цій конфігурації.

Кількісне зіставлення запропонованого методу з результатами, опублікованими в цитованих першоджерелах, утруднене відсутністю спільного симуляційного середовища: наявні дослідження застосовують різні моделі затримки (переважно аналітичні наближення M/M/1 без урахування контенції каналу й відмов вузлів), різні набори вузлів і різні визначення метрик пропускної здатності. З огляду на це основне кількісне порівняння в цій роботі виконано не проти окремо опублікованих результатів, а проти семи повторно реалізованих базових методів у єдиному дискретно-подієвому середовищі за ідентичних умов навантаження, топології та бюджету обчислень – підхід, що усуває методологічну розбіжність умов порівняння, властиву зіставленню чисел із різних публікацій, ціною неможливості прямого відтворення абсолютних значень метрик з першоджерел.

Таблиця 5 узагальнює якісне зіставлення AFAIA із класами методів, розглянутими в огляді літератури, за п'ятьма вимірами класифікаційної моделі. Позначення «так»/«ні»/«частково» відображають наявність відповідної властивості в архітектурі методу, а не результат емпіричного вимірювання.

Таблиця 5. **Якісне зіставлення класів методів за архітектурними властивостями**

Властивість	PSO/GA [6,7]	ACO [21]	ML+PSO [12]	MARL [15]	FL [13,14]	AFAIA
Онлайн-адаптація до навантаження	ні	ні	частково	так	ні	так
Топологічна обізнаність	ні	частково	ні	ні	ні	так
Формальна DP-гарантія	ні	ні	ні	ні	так	так
Потреба в попередньому навчанні/трейсах	ні	ні	так	так	так	так
Явний механізм балансування навантаження	ні	ні	ні	ні	ні	так

Джерело: **розроблено авторами**

Порівняно з гібридом ML+PSO [12], що поєднує навчання з підкріпленням та PSO-GA оптимізацію з окремим етапом прогнозування навантаження, AFAIA не потребує окремого прогнозного етапу, оскільки адаптація вбудована безпосередньо в цикл прийняття рішень через модуль зворотного зв'язку. Порівняно з мультиагентним actor-critic підходом (MAAC) [15], AFAIA доповнює адаптивність формальною DP-гарантією та явним обліком топологічної структури мережі через GAT-компонент, відсутнім у розглянутому підході. Порівняно з федеративними методами [13], [14], AFAIA поєднує механізм диференційної приватності з онлайн-адаптацією до миттєвого стану черг, тоді як розглянуті FL-підходи оптимізують глобальну модель без урахування миттєвого навантаження конкретного вузла на момент призначення задачі.

Складність реалізації визначена аналітично на основі параметрів реалізації кожного методу та доповнена виміряним часом обчислення з експерименту масштабованості (підрозділ «Порівняння за додатковими вимірами» розділу «Результати»).

Таблиця 6. **Порівняння обчислювальної складності методів**

Метод	Тип рішення	Ключові параметри	Час, N=15, с	Час, N=80, с
PSO	Пакетний, популяційний пошук	P=10, I=15 на цикл перепланування	0,187	0,398
GA	Пакетний, популяційний пошук	P=12, G=15 на цикл перепланування	0,853	1,082
ACO	Пакетний, феромонна евристика	8 мурах на цикл перепланування	0,121	0,404
ML+PSO	Пакетний, LPT-ініціалізація + PSO	P=10, I=15 + жадібна ініціалізація	0,172	0,342
MARL	Онлайн, табличне Q-навчання	Q[N], складність O(N) на рішення	0,044	0,166
FL	Онлайн, федеративне усереднення	O(N) на рішення + періодичний раунд O(8N)	0,035	0,190
MARL+GNN+FL	Онлайн, гібридний	Q[N] + структурні ознаки[N] + ваги FL[N]	0,047	0,168
AFAIA	Онлайн, навчений GAT+MADDPG	GAT складності $O(N^2)$ + прямий прохід MLP	0,202	0,649

Джерело: **розроблено авторами**

Табличний MARL та його гібридна форма MARL+GNN+FL мають найнижчу обчислювальну вартість серед перевірених методів, оскільки прийняття рішення зводиться до порівняння N скалярних значень без матричних операцій. AFAIA має квадратичну залежність часу обчислення від N через щільну матрицю уваги GAT-шару, що за N=80 дає час, порівнянний із методами PSO/ML+PSO, і приблизно вчетверо більший за MARL. GA демонструє найвищу абсолютну вартість обчислення серед перевірених методів унаслідок комбінації розміру популяції, кількості поколінь і оператора мутації, застосовуваних на кожному циклі перепланування.

Обмеження дослідження

Дослідження має такі обмеження. По-перше, навантаження є синтетичним: самоподібне ON/OFF-джерело Парето відтворює статистичну властивість самоподібності реального IoT-трафіку, однак не є трейсом реальної експлуатованої системи; верифікація на реальних IoT-датасетах становить необхідний напрям подальшої роботи. По-друге, параметри MTBF і MTTR вузлів є масштабованою аналогією, а не результатом калібрування на реальних даних про відмови периферійного обладнання, – збережено лише порядок співвідношення MTBF:MTTR, а не абсолютні значення, оскільки реальні виміряні дані про відмови периферійної інфраструктури вимірюються в годинах-місяцях, тоді як горизонт симуляції охоплює секунди-хвилини модельного часу. По-третє, структура задач обмежена ланцюжками довжиною від однієї до трьох підзадач (chain-DAG); розгалужені структури залежностей (паралельний fan-out/fan-in) не підтримуються поточною реалізацією й потребують заміни послідовного очікування на явний механізм очікування кількох батьківських підзадач. По-четверте, температура Boltzmann-аукціону AFAIA фіксована на етапах навчання ($\tau\_train=0{,}20$) та інференсу ($\tau\_eval=0{,}08$) і не адаптується до поточного рівня насичення системи, що є прямою причиною зниження частки дотримання дедлайну під екстремальним навантаженням, задокументованого в підрозділі «Компроміс throughput і дотримання дедлайну під навантаженням». По-п’яте, класифікаційна модель вибору класу методу за коефіцієнтом варіації навантаження не отримала підтвердження при незалежній перевірці на значеннях CV, відсутніх серед базових сценаріїв, – емпірично найкращим

методом на цих точках виявився GA незалежно від CV, що вказує на необхідність переформулювання критерію класифікації на основі архітектурних параметрів системи (control_interval, рівень насичення ρ_system), а не статистичних характеристик вхідного навантаження.

Висновки

Представлено метод AFAIA побудови оптимальної архітектури туманної комп'ютерної системи на основі дискретно-подієвої симуляційної моделі з реальними чергами обслуговування, контенцією пропускної здатності каналу, ланцюжковими залежностями підзадач і стохастичним процесом відмов вузлів. Метод реалізує навчену архітектуру: параметричний шар графової уваги для структурного кодування топології, мультиагентний контролер типу MADDPG зі спільними вагами Actor-мережі та mean-field критиком, адаптивний оркестратор на основі стохастичного Boltzmann-аукціону та модуль зворотного зв'язку, що навчає модель на фактично реалізованих результатах виконання підзадач.

Порівняльна оцінка на восьми методах у п'яти сценаріях під базовим і стрес-навантаженням показала, що AFAIA досягає найкращого серед усіх восьми методів показника балансування навантаження у чотирьох з п'яти сценаріїв базового режиму та конкурентної, хоча не найкращої, середньої затримки. Встановлено, що архітектура прийняття рішень є домінантним фактором відмінності класів методів за затримкою, суттєвішим за вибір конкретного алгоритму оптимізації в межах однієї архітектури. Виявлено й кількісно задокументовано архітектурний компроміс AFAIA між балансуванням навантаження за нормальних умов та рішучістю прийняття рішень під екстремальним навантаженням.

Напрямками подальших досліджень визначено: розробку адаптивної температури Boltzmann-аукціону, залежної від поточного рівня насичення системи; верифікацію методу на реальних IoT-трейсах; розширення структури задач за межі ланцюжкових залежностей; калібрування параметрів відмов вузлів на реальних експлуатаційних даних; переформулювання класифікаційної моделі вибору класу методу на основі архітектурних параметрів системи з повторною незалежною валідацією; завершення четвертої конфігурації стрес-навантаження.

Конфлікт інтересів: Автори декларують, що не мають конфлікту інтересів, зокрема фінансового, особистого, авторського чи будь-якого іншого характеру, який міг би вплинути на дослідження, а також на результати, опубліковані в цій статті

Фінансування: Дослідження проводилося без фінансової підтримки

Доступність даних: Усі дані доступні в цифровій або графічній формі в основному тексті рукопису

Використання засобів штучного інтелекту: Під час підготовки рукопису використовувався інструмент Antrophic Claude Sonnet 5 штучного інтелекту (ШІ) для граматичної та стилістичної перевірки тексту. Усі фрагменти, опрацьовані за його допомогою, були перевірені й відредаговані автором. ШІ не використовувався для генерації тексту, формул і таблиць, формування наукових положень, отримання результатів або формулювання висновків. Наукові положення, результати та висновки є власним інтелектуальним внеском авторів

СПИСОК ЛІТЕРАТУРИ

1. Dauda, A., Flauzac, O. & Nolot, F. “ Survey on IoT application architectures”. *Sensors.* 2024; 24 (16): 5320. DOI: https://doi.org/10.3390/s24165320.

2. Gupta, H., Vahid Dastjerdi, A., Ghosh, S. K. & Buyya, R. “iFogSim: A toolkit for modeling and simulation of resource management techniques in Internet of Things, edge and fog computing environments”. *Software: Practice and Experience*. 2017; 47 (9): 1275–1296. DOI: https://doi.org/10.1002/spe.2509.

3. Bonomi, F., Milito, R., Zhu, J. & Addepalli, S. “Fog computing and its role in the internet of things”. 2012/ 13–16. DOI: https://doi.org/10.1145/2342509.2342513.

4. Ni, L., Zhang, J., Jiang, C., Yan, C. & Yu, K. “Resource allocation strategy in fog computing based on priced timed Petri nets”. *IEEE Internet of Things Journal*. 2017; 4 (5): 1216–1228. DOI: https://doi.org/10.1109/JIOT.2017.2709814.

5. Lera, I., Guerrero, C. & Juiz, C. “Availability-aware service placement policy in fog computing based on graph partitions”. *IEEE Internet of Things Journal*. 2019; 6 (2): 3641–3651. DOI: https://doi.org/10.1109/JIOT.2018.2887286.

6. Nguyen, B. M., Thi Thanh Binh, H., The Anh, T. & Bao Son, D. “Evolutionary algorithms to optimize task scheduling problem for the IoT based bag-of-tasks application in cloud–fog computing environment”. *Applied Sciences*. 2019; 9 (9): 1730. DOI: https://doi.org/10.3390/app9091730.

7. Bitam, S., Zeadally, S. & Mellouk, A. “Fog computing job scheduling optimization based on bees swarm”. *Enterprise Information Systems*. 2018; 12 (4): 373–397. DOI: https://doi.org/10.1080/17517575.2017.1304579.

8. Gao, Q. et al. “Blockchain-based collaborative edge computing: efficiency, incentive and trust”. *Journal of Cloud Computing*. 2023; 12 (1): 72. DOI: https://doi.org/10.1186/s13677-023-00452-4.

9. Zhang, Y. et al. “Deadline-aware dynamic task scheduling strategy for edge-cloud collaborative computing”. *Electronics*. 2022; 11 (15): 2464. DOI: https://doi.org/10.3390/electronics11152464.

10. Li, Z. et al. “Energy-aware task offloading with deadline constraint in mobile edge computing”. *EURASIP Journal on Wireless Communications and Networking*. 2021; 2021: 56. DOI: https://doi.org/10.1186/s13638-021-01941-3.

11. Nieto, G. et al. “Deep reinforcement learning for dynamic task offloading in 5G edge-cloud continuum”. *Journal of Cloud Computing*. 2024; 13 (1): 94. DOI: https://doi.org/10.1186/s13677-024-00658-0.

12. Liu, F, Liu, Z, Liu, X. & Zhou, H. “Scalable hybrid framework for real time and non real time task scheduling in fog computing using federated reinforcement learning and PSO GA”. *Sci Rep*. 2025; 15: 38356. DOI: https://doi.org/10.1038/s41598-025-22218-5.

13. McMahan, H. B., Moore, E., Ramage, D., Hampson, S. & Arcas, B. A. “Communication-efficient learning of deep networks from decentralized data”. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2017. p. 1273–1282. DOI: https://doi.org/10.48550/arXiv.1602.05629.

14. Zhou, C., Fu, A., Yu, S., Yang, W., Wang, H. & Zhang, Y. “Privacy-preserving federated learning in fog computing”. *IEEE Internet of Things Journal*. 2020; 7 (11): 10782–10793. DOI: https://doi.org/10.1109/JIOT.2020.2987958.

15. Jain, V. & Kumar, B. “QoS-aware task offloading in fog environment using multi-agent deep reinforcement learning”. *Journal of Network and Systems Management*. 2023; 31: 7. DOI: https://doi.org/10.1007/s10922-022-09696-y.

16. Baek, J. & Kaddoum, G. “Heterogeneous task offloading and resource allocations via deep recurrent reinforcement learning in partial observable multifog networks”. *IEEE Internet of Things Journal*. 2021; 8 (2): 1041–1056. DOI: https://doi.org/10.1109/JIOT.2020.3009540.

17. Hady, M., Hu, S., Pratama, M. & Kowalczyk, R. “Multi-agent reinforcement learning for resources allocation optimization: a survey”. *Artif Intell Rev*. 2025. DOI: 10.1007/s10462-025-11340-5.

18. Xiao, W., et al. “GraphEdge: Dynamic graph partition and task scheduling for GNNs computing in edge network”. *Information Fusion*. 2025; 103329. DOI: https://doi.org/10.1016/j.inffus.2025.103329.

19. Saif, F., Latip, R., Hanapi, Z., Kamarudin, S., Kumar, A. & Bajaher, A. S. “Multi-Objectives Firefly Algorithm for Task Offloading in the Edge-Fog-Cloud Computing”. *IEEE Access*. 2024; 12: 159561–159578. DOI: https://doi.org/10.1109/access.2024.3488032.

20. Ijaz, S., Ahmad, S. G., Ayyub, K., Munir, E.U. & Ramzan, N. “Energy-efficient time and cost constraint scheduling algorithm using improved multi-objective differential evolution in fog computing”. *J Supercomput*. 2024; 81 (1): 116. DOI: https://doi.org/10.1007/s11227-024-06550-7.

21. Kishor, A. & Chakarbarty, C. “Task offloading in fog computing for using smart ant colony optimization”. *Wireless Personal Communications*. 2022; 127: 1683–1704. DOI: https://doi.org/10.1007/s11277-021-08714-7.

22. Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P. & Bengio, Y. “Graph attention networks”. *International Conference on Learning Representations (ICLR)*. 2018. DOI: https://doi.org/10.48550/arXiv.1710.10903.

23. Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, P. & Mordatch, I. “Multi-agent actor-critic for mixзыed cooperative-competitive environments”. *Advances in Neural Information Processing Systems (NeurIPS)*. 2017. 30. DOI: https://doi.org/10.48550/arXiv.1706.02275.

24. Yang, Y., Luo, R., Li, M., Zhou, M., Zhang, W. & Wang, J. “Mean field multi-agent reinforcement learning”. *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 2018. p. 5571–5580. DOI: https://doi.org/10.48550/arXiv.1802.05438.

25. Willinger, W., Taqqu, M. S., Sherman, R. & Wilson, D. V. “Self-similarity through high-variability: statistical analysis of Ethernet LAN traffic at the source level”. *IEEE/ACM Transactions on Networking*. 1997; 5 (1): 71–86. DOI: https://doi.org/10.1109/90.554723.

26. Dwork, C. & Roth, A. “The algorithmic foundations of differential privacy”. *Foundations and Trends in Theoretical Computer Science*. 2014; 9 (3–4): 211–407. DOI: https://doi.org/10.1561/0400000042.

Отримано 13.07.2026
Отримано після перегляду 26.08.2026
Прийнято 02.09.2026

DOI: https://doi.org/10.15276/ict.03.2026.29
UDC 004.75:004.8

Adaptive method for service placement in fog computing systems based on graph attention and multi-agent reinforcement learning

Oleksandr Yu. Sbitniev[1)]
ORCID: https://orcid.org/0009-0008-6311-612X; alexsbitnev99@gmail.com
Liudmyla A. Voloshchuk[1)]
ORCID: https://orcid.org/0000-0002-2510-0038; lavstumbre@gmail.com
[1)] Odesa I. I. Mechnikov National University, 2, Vsevoloda Zmiienka St. Odesa, 65082, Ukraine

ABSTRACT

The paper presents the Adaptive Fog Architecture with Integrated Artificial Intellegence method for constructing an optimal fog computing system architecture. The method implements a four-component decision-making pipeline: a trained graph attention layer for structural encoding of network topology; a multi-agent controller of the Multi-Agent Deep Deterministic Policy Gradient type with shared Actor-network weights and a mean-field critic for scalability across a variable number of nodes; an adaptive orchestrator based on a stochastic Boltzmann auction; and a feedback module that trains the model on actually realized subtask execution outcomes rather than a priori estimates. The method is implemented and evaluated in a discrete-event simulation environment (SimPy) featuring realistic service queues, channel bandwidth contention, chained subtask dependencies, and a stochastic node failure process. A comparative evaluation across eight methods (the presented method and seven baselines: Particle Swarm Optimization, Genetic Algorithm, Ant Colony Optimization, Machine Learning + Particle Swarm Optimization, Multi-Agent Reinforcement Learning, Federated Learning, and Multi-Agent Reinforcement Learning + Graph Neural Networks + Federated Learning) in five scenarios under baseline and stress loads showed that the presented method achieves the best load-balancing performance among all eight methods in four of the five baseline scenarios, along with competitive, though not best, average latency. It was established that the decision-making paradigm (periodic batch scheduling versus immediate online response) is the dominant factor distinguishing method classes by latency, exceeding in significance the choice of a specific optimization algorithm within a single architecture. An architectural trade-off was identified: auction stochasticity, beneficial for load balancing under normal conditions, reduces the deadline compliance rate under extreme load. Independent holdout validation of the initially proposed

method-class classifier did not confirm the hypothesis of a determining role of load variability, substantiating the need to reformulate it based on architectural parameters of the system rather than statistical characteristics of the input stream.

Keywords: Fog computing; cloud computing; discrete-event simulation; service placement;, graph attention networks; reinforcement learning; multi-agent systems; federated learning; numerical simulation; machine learning; deep learning, neural network; dynamic environment; architecture; dynamic characteristics

ІНФОРМАЦІЯ ПРО АВТОРІВ

Сбітнєв Олександр Юрійович - аспірант кафедри Математичного забезпечення комп'ютерних систем. Одеський національний університет імені І. І.Мечникова, вул. Всеволода Змієнка, 2, Одеса, 65082, Україна
Галузь дослідження: комп'ютерні мережі; хмарні технології; туманні технології; машинне навчання; інтернет речей, аналіз даних

Oleksandr Yuriyovych Sbitnev - PhD Student, Department of Mathematical Support of Computer Systems. Odesa I. I. Mechnikov National University, 2, Vsevolod Zmiienko St. Odesa, 65082, Ukraine
ORCID: https://orcid.org/0009-0008-6311-612X; alexsbitnev99@gmail.com
Research field: computer networks; cloud technologies; nebulous technologies; machine learning; internet of things, data analysis

Волощук Людмила Арнольдовна - кандидат технічних наук, доцент кафедри Математичного забезпечення комп'ютерних систем. Одеський національний університет імені І. І.Мечникова, вул. Всеволода Змієнка, 2, Одеса, 65082, Україна
Галузь дослідження: комп'ютерні мережі; хмарні технології; технології захисту інформації; машинне навчання; big data

Lyudmila A. Voloshchuk - PhD, Associate Professor. Department of Mathematical Support of Computer Systems, Odessa I. I. Mechnikov National University, 2, Vsevolod Zmiienko St. Odesa, 65082, Ukraine
ORCID: https://orcid.org/0000-0002-2510-0038; lavstumbre@gmail.com
Research field: Computer networking; cloud technology; information security technology; machine learning; big data