Інформаційні технології

УДК 004.9 **doi: 10.26906/SUNZ.2026.2.040**

С. В. Бондаренко[1], В. О. Мартовицький[1], Н. М. Бологова[1], В. Г. Рикун[2]

[1] Харківський національний університет радіоелектроніки, Харків, Україна
[2] Харківський національний університет Повітряних Сил ім. І. Кожедуба, Харків

ПЛАНУВАННЯ ЗАДАЧ У БАГАТОПРОЦЕСОРНИХ СИСТЕМАХ НА ОСНОВІ ГІБРИДНИХ МЕТОДІВ

Анотація. Стрімкий розвиток багатопроцесорних та розподілених обчислювальних систем обумовлює необхідність підвищення ефективності планування задач у гетерогенних середовищах. Об'єктом дослідження є процес планування задач у багатопроцесорних системах з неоднорідними обчислювальними ресурсами. Метою роботи є підвищення ефективності формування розкладу шляхом розробки гібридного підходу, що поєднує класичні евристичні алгоритми та методи навчання з підкріпленням. У роботі проведено аналіз існуючих методів планування задач, включаючи спискові евристики та підходи на основі машинного навчання. Запропоновано функціональну модель планування, яка інтегрує алгоритми HEFT і CPOP з агентом навчання з підкріпленням, реалізованим за схемою Actor–Critic із використанням алгоритму Proximal Policy Optimization. Розроблено математичну модель середовища планування, що враховує структуру DAG, матрицю очікуваного часу виконання та показники балансування навантаження. Для експериментальної перевірки створено потокову фабрику генерації різноманітних сценаріїв та реалізовано векторизоване середовище навчання. Результати експериментального дослідження показали, що запропоновані гібридні методи забезпечують стабільне зменшення makespan до 5-7% та 13-15% та середнього часу завершення задач порівняно з базовими евристиками, при контрольованому впливі на показники балансування навантаження. Отримані результати підтверджують можливість керованого компромісу між мінімізацією часу виконання та рівномірністю використання ресурсів. Наукова новизна роботи полягає у розробці узагальненої функціональної моделі гібридного планування задач, що забезпечує адаптивне коригування евристичних алгоритмів на основі навчання з підкріпленням у гетерогенному багатопроцесорному середовищі. Практична значущість полягає у можливості застосування запропонованого підходу в системах високопродуктивних, хмарних та розподілених обчислень для підвищення продуктивності та ефективності використання ресурсів.

Ключові слова: планування задач; багатопроцесорні системи; гібридні методи; навчання з підкріпленням; проксимальна оптимізація політики; критичний шлях на процесорі; балансування навантаження.

Вступ

Постановка проблеми. Стрімкий розвиток інформаційних технологій та зростання обсягів даних, що потребують обробки в режимі реального часу, призвели до повсюдного впровадження багатопроцесорних та розподілених обчислювальних систем. Сучасні архітектури, від хмарних (Cloud) та туманних (Fog) обчислень до вбудованих систем реального часу, вимагають ефективного управління ресурсами для забезпечення високої продуктивності та надійності. Ключовим елементом такого управління є планування задач (task scheduling), яке визначає порядок виконання завдань та їх розподіл між доступними процесорами. Як зазначають дослідники у свіжих оглядах, ефективність планування безпосередньо впливає на загальну пропускну здатність системи, час відгуку та енергоспоживання [1].

Планування задач у багатопроцесорних системах належить до класу NP-важких (NP-hard) задач комбінаторної оптимізації. Це означає, що зі збільшенням кількості завдань та процесорів складність знаходження оптимального розв'язку зростає експоненціально, що робить неможливим використання методів повного перебору для систем реальної розмірності. Особливої гостроти проблема набуває в системах реального часу, де порушення часових обмежень (deadlines) може призвести до критичних збоїв у роботі всієї інфраструктури [2]. Традиційні підходи до планування, такі як статичні евристики (наприклад, Round Robin, First-Come-First-Served) або класичні алгоритми спискового планування, часто виявляються неефективними в умовах динамічного навантаження та гетерогенності ресурсів. Водночас сучасні вимоги до обчислювальних систем стають багатоцільовими (multi-objective): окрім мінімізації часу виконання (makespan), необхідно враховувати вартість обчислень, балансування навантаження та енергоефективність. Наприклад, останні дослідження підкреслюють важливість одночасної оптимізації вартості та енергоспоживання в середовищах хмарних та граничних обчислень (Cloud-Edge computing), що вимагає застосування більш адаптивних інтелектуальних підходів [3].

Для подолання обмежень класичних методів в останні роки активно розробляються алгоритми, що базуються на метаевристиках (генетичні алгоритми, ройовий інтелект) та методах машинного навчання. Проте, як свідчить аналіз методів планування в системах реального часу, кожен окремий підхід має свої недоліки: евристики можуть "застрягати" в локальних оптимумах, а методи глибокого навчання потребують значних ресурсів для тренування моделей [4]. Саме тому науковий інтерес зміщується в бік гібридних методів (hybrid methods), які поєднують сильні сторони різних підходів. Гібридизація дозволяє компенсувати недоліки одного методу перевагами іншого, наприклад, використовуючи евристики для початкової ініціалізації популяції в генетичних алгоритмах або застосовуючи локальний пошук для покращення результа-

тів глобальної оптимізації. Дослідження показують, що багатоцільові гібридні алгоритми здатні забезпечити кращий баланс між часом пошуку розв'язку та його якістю порівняно з монолітними підходами [5].

Незважаючи на значний прогрес у цій галузі, проблема розробки універсальних та ефективних гібридних методів планування залишається відкритою. Існує потреба в детальному дослідженні механізмів поєднання різних алгоритмічних стратегій для адаптації до динамічних змін у багатопроцесорних середовищах. Це дослідження спрямоване на розробку та аналіз підходу до планування задач на основі гібридних методів, що дозволить підвищити ефективність використання ресурсів багатопроцесорних систем.

Аналіз останніх досліджень і публікацій. Проблема ефективного розподілу обчислювального навантаження у багатопроцесорних системах є предметом ґрунтовних наукових досліджень протягом останніх десятиліть. Аналіз літературних джерел дозволяє стверджувати, що підходи до її вирішення еволюціонували від класичних математичних моделей до складних адаптивних систем штучного інтелекту. Фундаментальну основу для розуміння процесів планування закладено в теорії масового обслуговування (Queuing Theory). Як зазначається у роботі [6], використання моделей черг дозволяє оцінити базові характеристики продуктивності системи, такі як середній час очікування задачі та коефіцієнт завантаження процесорів. Проте, класичні ймовірнісні моделі часто ідеалізують реальні умови функціонування, не враховуючи гетерогенність сучасних обчислювальних вузлів та складні залежності між задачами.

Для більш детального моделювання часових характеристик та синхронізації процесів у паралельних системах дослідники часто звертаються до апарату мереж Петрі. Зокрема, у дослідженні [7] показано, що цей математичний інструмент дозволяє ефективно виявляти потенційні блокування (deadlocks) та конфлікти доступу до ресурсів ще на етапі проектування розкладу. Однак, зі зростанням кількості вузлів та завдань, простір станів мережі Петрі вибухоподібно зростає, що ускладнює її використання для динамічного планування в режимі реального часу (online scheduling). Існуючі методи планування традиційно поділяють на статичні та динамічні. Статичні підходи, розглянуті у роботі [8], передбачають, що характеристики всіх завдань (час виконання, обсяг даних, директивні строки) відомі заздалегідь. Це дозволяє використовувати точні математичні методи або алгоритми на основі графа завдань (DAG – Directed Acyclic Graph) для побудови оптимального розкладу ще до початку виконання програм. Хоча такі методи гарантують детермінованість, вони виявляються безпорадними в умовах невизначеності, характерної для сучасних хмарних та туманних обчислень. У спробі подолати обмеження статичних методів та високу обчислювальну складність точних алгоритмів, значна частина досліджень була спрямована на розробку евристичних алгоритмів. Як зазначається в огляді методів та засобів планування [9], популярні евристики спискового планування (List Scheduling), такі як HEFT (Heterogeneous Earliest Finish Time) або CPOP (Critical Path on a Processor), забезпечують прийнятний компроміс між якістю розкладу та часом його генерації. Ці алгоритми базуються на пріоритезації завдань, часто використовуючи поняття критичного шляху в графі задач.

Розвиток цього напрямку призвів до появи рангових підходів. У роботі [10] досліджено застосування алгоритмів рангового підходу, де кожній задачі присвоюється певний ранг на основі її обчислювальної ваги та комунікаційних витрат. Це дозволяє більш гнучко керувати чергою виконання, проте більшість таких алгоритмів є «жадібними» (greedy) за своєю природою. Вони приймають локально оптимальні рішення на кожному кроці, що часто призводить до субоптимального глобального результату, особливо в системах з високим ступенем гетерогенності ресурсів.

Окремий клас проблем становить планування у розподілених системах реального часу, де порушення часових обмежень є неприпустимим. Дослідження [11] підкреслює, що в таких системах критерій оптимізації зміщується з мінімізації загального часу виконання (makespan) на максимізацію кількості завдань, виконаних до настання дедлайну. Традиційні евристики часто не здатні ефективно балансувати між цими суперечливими цілями, що спонукало науковців до пошуку більш досконалих методів оптимізації.

Недоліки детермінованих евристик призвели до широкого застосування метаевристичних методів, натхненних природними процесами. Узагальнюючий аналіз методів планування [12] виділяє генетичні алгоритми (GA), алгоритми імітації відпалу (Simulated Annealing) та ройовий інтелект (PSO, ACO) як потужні інструменти глобальної оптимізації. Ці методи здатні знаходити розв'язки, близькі до оптимальних, у величезних просторах пошуку, не вимагаючи повної інформації про систему. Проте, як вже згадувалося в попередніх оглядах [5], їхня головна вада – повільна збіжність та висока чутливість до налаштування гіперпараметрів, що робить їх проблематичними для використання у високодинамічних середовищах без додаткових модифікацій. Таким чином, аналіз класичних та метаевристичних методів демонструє, що жоден з «чистих» підходів не забезпечує універсального рішення для сучасних багатопроцесорних систем, які характеризуються динамічністю, гетерогенністю та багатокритеріальністю. Це створює передумови для звернення до методів машинного навчання та, зрештою, до гібридних архітектур, здатних адаптуватися до змін у навантаженні.

З появою концепцій інтелектуального управління ресурсами (Intelligent Resource Management), вектор наукових пошуків змістився в бік методів машинного навчання (Machine Learning, ML), зокрема навчання з підкріпленням (Reinforcement Learning, RL). На відміну від статичних евристик, RL-агенти здатні навчатися оптимальної стратегії планування через постійну взаємодію із середовищем, отримуючи "винагороду" за ефективні рішення. У роботі [13] пропонується підхід на основі ієрархічного глибокого навчання з підкріпленням (Hierarchical Deep Reinforcement Learning), який дозволяє декомпозувати складну задачу багатоцільового планування на підзадачі меншої розмірності. Це значно підвищує здатність системи адаптуватися до

непередбачуваних змін у потоці завдань, що є критичним для динамічних середовищ.

Особливу увагу в контексті планування залежних задач привертають графові нейронні мережі (Graph Neural Networks, GNN). Оскільки більшість паралельних програм моделюються як направлені ациклічні графи (DAG), використання GNN дозволяє ефективно витягувати просторові ознаки та залежності між вершинами графа. Дослідники [14] демонструють ефективність поєднання GNN з мультиагентним навчанням з підкріпленням (MARL) для задач розміщення навантаження в граничних обчисленнях (Edge Computing). Такий підхід дозволяє враховувати топологію мережі та мінімізувати затримки передачі даних. Аналогічний напрямок розвивається у роботі [15], де представлена архітектура ScheduleNet. Вона дозволяє агентам "навчитися" розв’язувати проблеми планування, оперуючи безпосередньо структурою графа завдань, що забезпечує кращу генералізацію на нові, раніше не бачені типи навантажень.

Практичне впровадження інтелектуальних методів знаходить своє відображення і в популярних системах оркестрації контейнерів. Зокрема, у дослідженні [16] розглядається вдосконалення стандартного планувальника Kubernetes за допомогою методів глибокого навчання та RL. Автори підтверджують, що інтеграція навчених моделей у реальні кластери дозволяє суттєво знизити час очікування подів та покращити утилізацію кластера порівняно зі стандартними алгоритмами пакування (bin packing). Також існують спроби вдосконалення самих нейромережевих архітектур для специфічних задач багатопроцесорної обробки, як показано у [17], де модифікована нейронна мережа використовується для прискорення збіжності процесу пошуку розкладу. Втім, попри значні переваги, методи на основі чистого машинного навчання мають суттєві обмеження: вони вимагають значних обчислювальних ресурсів для тренування, страждають від проблеми "холодного старту" та можуть демонструвати нестабільну поведінку при різкій зміні розподілу вхідних даних. Саме тому сучасна наукова думка схиляється до гібридних методів, які інтегрують швидкість та надійність евристик із адаптивністю штучного інтелекту. Наприклад, використання евристик для початкового розподілу або як "страхувального механізму" для RL-агента дозволяє нівелювати ризики та підвищити загальну надійність системи. Для верифікації таких складних гібридних моделей необхідні спеціалізовані інструменти, і, як зазначено в роботі [18], розробка адекватних систем моделювання є невід’ємною частиною процесу створення нових алгоритмів планування.

Постановка проблеми. Проведений аналіз літературних джерел дозволяє виявити суттєве протиріччя. З одного боку, сучасні багатопроцесорні системи вимагають планування, яке є одночасно швидким (як статичні евристики) та адаптивним до динамічних змін (як методи машинного навчання). З іншого боку, існуючі "чисті" підходи не здатні повною мірою задовольнити обидві вимоги одночасно: евристики дають локальні оптимуми та не адаптуються, а методи RL є ресурсомісткими та складними у навчанні. **Актуальною науковою проблемою** є розробка методології планування, яка б ефективно поєднувала різні класи алгоритмів у єдину гібридну структуру. Більшість існуючих гібридних рішень фокусуються на конкретних вузьких сценаріях (наприклад, тільки хмарні обчислення або тільки енергоефективність) і недостатньо досліджують загальні принципи взаємодії компонентів гібридної системи для досягнення балансу між часом планування та якістю розкладу (makespan, load balancing).

Таким чином, проблемою дослідження є недостатня ефективність існуючих методів планування задач у гетерогенних багатопроцесорних системах в умовах змінного навантаження. Вирішення цієї проблеми потребує розробки нового гібридного підходу, який інтегрує евристичні методи з адаптивними механізмами, забезпечуючи підвищення продуктивності системи без надмірних накладних витрат. Це обумовлює необхідність створення функціональної моделі такого планування та проведення експериментальних досліджень її ефективності.

Основний матеріал

Для досягнення поставленої мети та вирішення задач дослідження необхідно чітко визначити методологічну базу. Об’єктом дослідження є процес планування обчислювальних задач у багатопроцесорних системах з неоднорідними ресурсами. Предметом дослідження виступають гібридні методи та алгоритми, що забезпечують оптимізацію розподілу навантаження та мінімізацію часу виконання завдань. Основою для проведення експериментів та верифікації запропонованого підходу є розроблена математична модель, яка дозволяє формалізувати параметри обчислювального середовища та структуру завдань. Загальна структура математичної моделі, що відображає взаємозв'язок між параметрами вхідних даних, структурою графа задач (DAG), моделлю платформи та метриками ефективності, наведена на рис. 1.

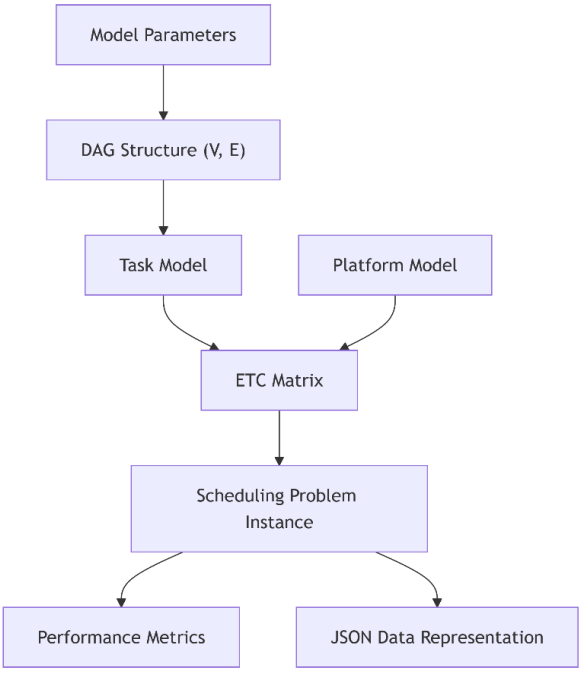

Рис. 1. Загальна структура математичної моделі

У даній роботі задача планування розглядається для моделі workflow, яка представлена у вигляді орієнтованого ациклічного графа (DAG):

$$G = (V, E), \quad (1)$$

де $V = \{v_1, v_2, \ldots, v_n\}$ – множина задач (вузлів графа), а E⊆V×V – множина залежностей між задачами. Наявність ребра $(v_i, v_j) \in E$ означає сувору технологічну залежність: задача v_j може розпочатися лише після повного завершення задачі v_i та отримання необхідних даних. Кожна задача $v_j \in V$ в моделі описується кортежем:

$$v_i = (id_i, level_i, type_i), \quad (2)$$

де id_i – унікальний ідентифікатор задачі, $type_i$ – тип задачі (обчислювальна, об'єднання даних або введення/виведення), а $level_i$ – топологічний рівень задачі у DAG. Топологічний рівень є критично важливим параметром для рангових евристик і визначається рекурсивно:

$$level_i = \begin{cases} 0, & \text{if } pred(v_i) = \varnothing, \\ \max_{v_j \in pred(v_i)}(level_j) + 1, & \text{otherwise}, \end{cases} \quad (3)$$

де $pred(v_i)$ – множина безпосередніх попередників задачі v_i. Процес формування параметрів задач та платформи візуалізовано на рис. 2.

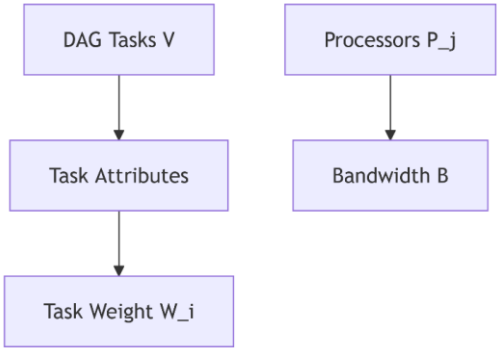

Рис. 2. Формування параметрів задач і платформи

Обчислювальна складність кожної задачі W_i не є статичною величиною, а моделюється як функція від її типу та рівня в графі, що дозволяє імітувати реальні сценарії, де складність обробки зростає по мірі проходження даних через конвеєр:

$$W_i = f(type_i, level_i). \quad (4)$$

Функція $f(\cdot)$ задається наступним чином:

$$W_i = \begin{cases} \alpha \cdot g(level_i), & \text{if } type_i = compute, \\ \beta \cdot g(level_i), & \text{if } type_i = merge, \\ \gamma, & \text{if } type_i = io, \end{cases} \quad (5)$$

де α,β,γ – коефіцієнти складності для різних типів задач, а $g(level_i)$ – монотонно зростаюча функція.

Паралельна обчислювальна система (платформа) моделюється множиною гетерогенних процесорів $P = \{p_1, p_2, \ldots, p_m\}$. Кожен процесор p_j характеризується коефіцієнтом відносної продуктивності $speed_j > 0$. Ключовим елементом моделі є матриця очікуваного часу виконання (Expected Time to Compute, ETC):

$$ETC = [ETC(i, j)]_{n \times m}, \quad (6)$$

де елемент *ETC(i,j)* визначає час виконання задачі v_i на процесорі p_j. Для наближення до реальних умов вводиться випадкове збурення $\epsilon_{i,j}$:

$$ETC(i, j) = \frac{w_i}{speed_j}(1 + \epsilon_{i,j}). \quad (7)$$

Окрім обчислень, модель враховує комунікаційні витрати. Кожне ребро $(v_i, v_j) \in E$ асоціюється з обсягом даних $data\_size_{i,j}$. Час передачі даних $Comm_{i,j}$ залежить від пропускної здатності каналу B та розміщення задач:

$$Comm_{i,j} = \begin{cases} \frac{data\_size_{i,j}}{B}, & \text{if } p(i) \neq p(j), \\ 0, & \text{if } p(i) = p(j), \end{cases} \quad (8)$$

де *p(i)* та p(j) – процесори, на які призначено задачі v_i та v_j відповідно. Для оцінки ефективності розроблених методів планування використовується набір метрик, взаємозв'язок яких показано на рис. 3.

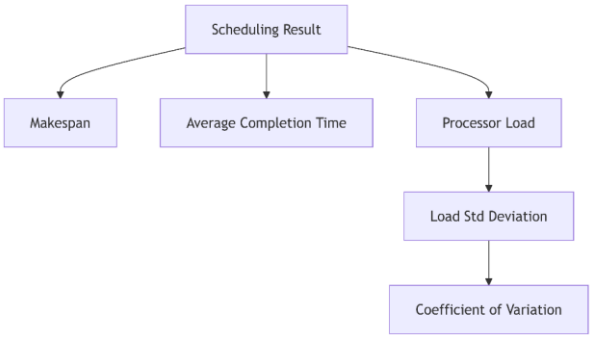

Рис. 3. Метрики оцінювання

Часові обмеження виконання визначаються двома параметрами: часом можливого початку $EST(v_i)$ та часом завершення $FT(v_i)$.

$$EST(v_i) = \max_{v_j \in pred(v_i)}(FT(v_j) + Comm_{i,j}), \quad (9)$$

$$FT(v_i) = EST(v_i) + EST(i, p(i)). \quad (10)$$

Основним критерієм оптимізації є Makespan (загальний час виконання workflow):

$$Makespan = \max_{v_i \in V} FT(v_i). \quad (11)$$

Додатковою метрикою, що дозволяє оцінити загальну затримку виконання незалежно від критичного шляху, є середній час завершення задач:

$$AvgCompletionTime = \frac{1}{|V|}\sum_{v_i \in V} FT(v_i). \quad (12)$$

Для аналізу якості використання ресурсів вводиться метрика балансування навантаження. Навантаження процесора p_j є сумою часів виконання всіх призначених на нього задач:

$$Load_j = \sum_{v_i \in V_j} ETC(i, j). \quad (13)$$

Якість розподілу оцінюється через середнє навантаження:

$$\overline{Load} = \frac{1}{m}\sum_{j=1}^{m} Load_j. \quad (14)$$

та його стандартне відхилення:

$$LB_{std} = \sqrt{\frac{1}{m}\sum_{j=1}^{m}(Load_j - \overline{Load})^2}. \quad (15)$$

Для можливості порівняння результатів на різних масштабах систем використовується нормалізований коефіцієнт варіації:

$$LB_{cv} = \frac{LB_{std}}{\overline{Load}}.$$

Менше значення LB_{cv} свідчить про більш рівномірний розподіл навантаження, що є індикатором ефективності гібридного методу планування.

Для проведення експериментального дослідження та верифікації розроблених методів, описану математичну модель необхідно трансформувати у формат, придатний для обробки програмним комплексом. У рамках даної роботи для серіалізації вхідних даних було обрано формат JSON через його наочність, розширюваність та легкість інтеграції з різними мовами програмування. Розроблена структура даних описує повний контекст одного експериментального сценарію, включаючи конфігурацію платформи, матрицю часу виконання та структуру графа завдань.

Структура вхідного файлу складається з трьох ключових блоків, що безпосередньо відображають компоненти математичної моделі: *processors* (модель платформи), ETC (матриця очікуваного часу виконання) та DAG (структурна модель workflow).

Блок processors являє собою масив об'єктів, що описує гетерогенний кластер обчислювальних вузлів *P*. Кожен об'єкт містить унікальний ідентифікатор *id* та коефіцієнт швидкодії *speed*, який відповідає параметру $speed_j$ у математичній моделі. Це дозволяє моделювати системи з різним ступенем гетерогенності, варіюючи значення швидкості процесорів.

Блок ETC реалізує матрицю *ETC(i,j)* у вигляді двовимірного масиву. Рядки масиву відповідають окремим завданням v_i, а стовпці – процесорам p_j. Значення у клітинці *[i][j]* визначає прогнозований час виконання *i*-ї задачі на j-му процесорі. Така структура дозволяє зберігати попередньо обчислені часові характеристики, що включають змодельовані випадкові збурення $\epsilon_{i,j}$.

Найбільш складною частиною структури є блок DAG, який описує граф завдань G(V,E) та складається з трьох підоб'єктів: dag_meta, tasks та edges.

Об'єкт dag_meta містить метадані, необхідні для налаштування генераторів навантаження та аналізу результатів. Сюди входять:

type – тип топології графа (наприклад, layered_fork_join);

num_tasks – загальна кількість вершин |V|;

max_level – глибина графа;

avg_parallelism – середня ширина графа, що впливає на потенціал розпаралелювання;

communication_model – параметри пропускної здатності мережі B.

Масив tasks відповідає множині вершин V. Кожен елемент масиву описує окрему задачу та містить поля *id* (ідентифікатор), *level* (топологічний рівень, необхідний для рангових евристик) та *type*. Поле type може приймати значення compute, merge або io, що визначає, яку саме формулу обчислення ваги W_i буде застосовано до даної задачі.

Масив edges описує множину дуг E та комунікаційні залежності. Кожен об'єкт зв'язку визначає задачу-попередника (from), задачу-наступника (to) та обсяг даних data_size, що передається між ними. Цей параметр використовується для розрахунку комунікаційної затримки $Comm_{i,j}$ у випадку, якщо зв'язані задачі плануються на різні процесори.

Загальний вигляд представлення даних у форматі JSON, що використовується в експериментах, наведено нижче:

Така організація даних забезпечує повну відповідність між теоретичною моделлю та її програмною реалізацією, дозволяючи проводити відтворювані експерименти для оцінки ефективності запропонованих гібридних методів планування.

Розроблена математична модель визначає задачу планування як проблему пошуку відображення множини задач V на множину процесорів P з метою мінімізації загального часу виконання workflow (*Makespan*) та забезпечення рівномірного розподілу навантаження між обчислювальними ресурсами (LB_{std}).

З урахуванням NP-складної природи задачі планування, а також гетерогенності сучасних багатопроцесорних систем, у даній роботі пропонується функціональна модель планування на основі гібридних методів, яка поєднує класичні евристичні алгоритми та методи глибокого навчання з підкріпленням (Deep Reinforcement Learning, DRL).

Функціональна модель системи планування формалізується у вигляді кортежу:

$$M = \langle E, S, A, H, \pi_\theta, R, L \rangle, \quad (17)$$

де V (Environment) – модель обчислювального середовища; S (State Space) – простір станів середовища; A (Action Space) – простір дій агента; H (Heuristic Guidance) – модуль евристичної підтримки, що використовується для маскування недопустимих дій та спрямування пошуку; π_θ (Policy) – політика RL-агента, що реалізована на базі FFN та оптимізується алгоритмом PPO у схемі Actor–Critic; R (Reward Function) – функція винагороди; L (Local post-processing) – необов'язковий модуль локального покращення (post-processing) готового розкладу.

Компонент E безпосередньо базується на математичній моделі і включає орієнтований ациклічний граф задач G=(V,E), матрицю очікуваного часу виконання ETC, характеристики гетерогенних процесорів P та модель комунікаційних витрат. Таким чином, середовище повністю визначає допустимі стани та обмеження планування. Стан системи на кожному кроці прийняття рішення t подається у вигляді вектора ознак фіксованої розмірності:

$$s_t = [\mathrm{DAG}_{\mathrm{embedding}}, \mathrm{P}_{\mathrm{status}}, \mathrm{Task}_{\mathrm{candidates}}], \quad (18)$$

де $\mathrm{DAG}_{\mathrm{embedding}}$ відображає агреговані характеристики поточного стану workflow (частка виконаних задач, оцінка довжини критичного шляху), $\mathrm{P}_{\mathrm{status}}$ – нормалізований вектор поточного завантаження процесорів:

$$P_{status} = [Load_1, Load_2, \ldots, Load_m], \quad (19)$$

а $\mathrm{Task}_{\mathrm{candidates}}$ описує множину готових до виконання задач, для яких виконані всі залежності $pred(v_i)$. Формування стану ґрунтується на метриках навантаження $Load_j$, визначених формулою (13), та оцінці критичного шляху DAG.

Простір дій A у загальному випадку визначається як вибір пари (v_i, p_j), де v_i – задача з множини готових, а p_j – процесор, на який вона призначається. Однак у запропонованій гібридній моделі RL-компонент використовується як спеціалізований модуль керування, тому структура простору дій A уточнюється залежно від схеми гібридизації.

У схемі Heuristic-guided RL для алгоритму HEFT (HG-RL–HEFT) евристика використовується для формування та ранжування множини готових задач:

$$Ready(t) = \{v \in V \mid pred(v) \subseteq Done(t)\}. \quad (20)$$

Із цієї множини формується підмножина $TopK(t)$ з k задач з найвищим пріоритетом. RL-актор на кроці t вибирає лише індекс задачі:

$$a_t \in \{1, \ldots, k\}, \quad (21)$$

тоді як вибір процесора виконується детерміновано за класичним правилом HEFT:

$$p^*(v) = \arg\min_{p_j \in P} EFT(v, p_j), \quad (22)$$

де EFT обчислюється на основі EST/FT із математичної моделі.

Тобто, для HG-RL–HEFT простір дій має вигляд:

$$A^{HG} = \{1, \ldots, k\}, \quad (v, p) = (TopK[a_t], \arg\min_{p_j} EFT(TopK[a_t], p_j)). \quad (23)$$

Таким чином, агент навчається коригувати вибір задачі, спираючись на глобальний стан системи, тоді як перевірка допустимості та розрахунок часових характеристик делегуються евристиці.

Для схеми RL-guided heuristic у випадку алгоритму CPOP (RL–GH–CPOP) агент керує параметризацією евристики.

Дія визначається як:

$$a_t = (p_{cp}, \mathrm{mode}), \quad (24)$$

де $p_{cp} \in P$ – процесор критичного шляху, а $mode \in \{soft, hard\}$ – режим обробки критичного шляху.

Далі застосовується стандартний алгоритм CPOP з фіксованими параметрами. Відповідно, простір дій має вигляд:

$$A^{RG} = P \times \{soft, hard\}. \quad (25)$$

Архітектура агента реалізується за схемою Actor–Critic із використанням алгоритму Proximal Policy Optimization (PPO).

- Актор (Actor) апроксимує політику $\pi_\theta(a/s)$, яка приймає стан s_t і повертає розподіл ймовірностей над допустимими діями, структура яких визначається обраною схемою гібридизації.

- Критик (Critic) оцінює функцію цінності поточного стану $V_\phi(s)$, прогнозуючи очікувану сумарну винагороду (Value function), що використовується для обчислення переваги (Advantage) у процесі навчання та дозволяє зменшити дисперсію градієнта.

Мережа реалізується у вигляді багатошарової нейронної мережі типу Feed-Forward Network (FFN) з 2–3 прихованими шарами по 128–256 нейронів та функцією активації ReLU.

Для гарантування допустимості рішень та прискорення збіжності навчання застосовується механізм евристичного маскування дій (Heuristic Guidance). Модифікована політика має вигляд:

$$\pi_{guided}(a \mid s) = Softmax(Logits(s) + M(s)), \quad (26)$$

де маска $M(s)$ приймає значення ∞ для недопустимих дій, що порушують топологічні обмеження DAG, та 0 для допустимих.

Функція винагороди формується на основі цільових метрик математичної моделі та визначається як:

$$r_t = \alpha \cdot (Makespan_{est}(t-1), -Makespan_{est}(t)) - \beta \cdot LB_{std}(t), \quad (27)$$

де $Makespan\_est(t)$ – евристична оцінка нижньої межі завершення workflow, яка може бути задана як $max\ \{FT_{current}, length_{CP}\}$, LB_{std} – стандартне відхилення навантаження процесорів, обчислене за формулою (15), а α, β – вагові коефіцієнти значущості метрик.

Перший доданок стимулює дії, що зменшують прогнозний час завершення всього workflow. Другий доданок (штраф) базується на стандартному відхиленні навантаження LB_{std}, запобігаючи дисбалансу системи.

Така функція винагороди стимулює одночасне зменшення часу виконання та покращення балансування навантаження.

Після побудови початкового розкладу може бути застосований необов'язковий модуль локального покращення $L(\cdot)$, який виконує обмежену кількість локальних перетворень (Move, Swap, Critical-path focusing), не порушуючи топологічних обмежень графа.

Типові локальні оператори:

Move: перемістити задачу v на інший процесор p', якщо це зменшує $FT(v)$ або зменшує критичний "хвіст" розкладу.

Swap: обміняти процесори двох задач v_i, v_j (за умови збереження допустимості за залежностями та ресурсами).

Critical-path focusing: обмежити пошук задачами, що належать до критичного шляху поточного розкладу або найближчого оточення CP.

У загальному вигляді процес побудови розкладу описується композицією:

$$Scheduler = L(S_{base}(G, P, ETC, Comm, \pi_\theta)), \quad (28)$$

де S_{base} – базовий гібридний планувальник (HG-RL–HEFT або RL–GH-CPOP).

Таким чином, запропонована функціональна модель забезпечує чітке розмежування ролей між евристичними та інтелектуальними компонентами, поєднуючи детермінованість і швидкодію класичних алгоритмів з адаптивністю методів навчання з підкріпленням. Це створює основу для експериментальної перевірки ефективності гібридного підходу.

Для перевірки ефективності запропонованої функціональної моделі планування задач у багатопроцесорних системах на основі гібридних методів було проведено серію експериментальних досліджень, спрямованих на оцінювання якості сформованих розкладів за ключовими часовими та ресурсними метриками. Метою експерименту було встановлення ступеня покращення класичних евристичних алгоритмів HEFT та CPOP шляхом їх інтеграції з методами глибокого навчання з підкріпленням у схемах HG-RL-HEFT та RL-GH-CPOP відповідно.

Програмна реалізація гібридних методів виконана мовою Python із використанням сучасних бібліотек машинного навчання та моделювання середовища. Для побудови та навчання нейромережевих моделей використовувалася бібліотека PyTorch (torch v2.10.0+cu128), яка забезпечує ефективну роботу з тензорними обчисленнями та підтримку апаратного прискорення. Формування середовища навчання з підкріпленням реалізовано на базі Gymnasium v1.2.3, що дозволило інтегрувати математичну модель (розділ 4) у стандартний RL-інтерфейс. Для реалізації алгоритму Proximal Policy Optimization (PPO) використовувалися бібліотеки Stable-Baselines3 v2.7.1 та SB3-Contrib v2.7.1, що забезпечують перевірені механізми Actor-Critic навчання та стабілізацію процесу оновлення політики.

Навчання моделей проводилося в середовищі з підтримкою GPU-прискорення, що дозволило забезпечити достатню швидкість обробки великої кількості епізодів. Апаратна конфігурація включала багатоядерний процесор (6 ядер), 32 ГБ оперативної пам'яті та графічний прискорювач NVIDIA GeForce RTX 5070 з підтримкою CUDA 13.1. Така конфігурація є типовою для сучасних дослідницьких задач машинного навчання та забезпечує відтворюваність експерименту. Для формування навчального потоку використовувалася спеціалізована фабрика генерації екземплярів задач, яка формує неперервний потік різноманітних DAG-графів із варіацією кількості задач, рівнів графа, середньої ширини, коефіцієнтів складності та гетерогенності процесорів. Базовий екземпляр передбачав генерацію workflow із базовою кількістю 100 задач, максимальною глибиною 10 рівнів, 6 процесорами та випадковими варіаціями гетерогенності, пропускної здатності каналу та шуму ETC-матриці. При цьому активовано механізми семплювання кількості задач (від 50 до 300), пропускної здатності (логарифмічно-однорідний розподіл у діапазоні 0.5–100.0), числа процесорів (2, 4, 8, 16), а також варіацій гетерогенності та коефіцієнтів складності. Такий підхід забезпечує формування неперервного потоку максимально різноманітних екземплярів, що запобігає переадаптації (overfitting) політики до конкретного типу графа та підвищує її узагальнювальну здатність

Фабрика інтегрується безпосередньо до середовища навчання через потоковий механізм генерації екземплярів, що дозволяє на кожному епізоді формувати новий DAG із незалежною конфігурацією. Таким чином, агент взаємодіє не з фіксованим набором сценаріїв, а з безперервною стохастичною послідовністю задач, що суттєво підвищує узагальнювальну здатність моделі. Для прискорення навчання використовувався механізм векторизованого середовища (vectorized environments). Одночасно запускалося по 6 паралельних екземплярів середовища для кожної гібридної схеми (n_envs_hg = 6, n_envs_rg = 6), що дозволяло накопичувати rollout-фрагменти з кількох незалежних сценаріїв паралельно. Така організація значно зменшує дисперсію градієнтної оцінки та підвищує стабільність алгоритму PPO, особливо в умовах високої варіативності графів.

Навчання моделі HG-RL-HEFT виконувалося з використанням GPU-прискорення, тоді як модель RL-GH-CPOP тренувалася на CPU. Це обумовлено різною обчислювальною складністю кроків взаємодії із середовищем: у схемі HG-RL-HEFT агент приймає рішення на кожному кроці побудови розкладу, тоді як у RL-GH-CPOP дія виконується одноразово на епізод (rg_action_frequency = "once").

На відміну від навчального режиму, для тестування та фінальної оцінки моделей використовувався окремий типовий сценарій із фіксованою конфігурацією параметрів та заздалегідь визначеною кількістю екземплярів. Це забезпечує коректне порівняння результатів між базовими евристиками та їх RL-реалізаціями в однакових умовах.

Сам процес навчання відбувався у режимі взаємодії агента з середовищем, де кожен епізод відповідав побудові повного розкладу для згенерованого DAG. Після завершення епізоду обчислювалися метрики makespan та балансування навантаження, які формували інтегральну функцію винагороди (формула (27)). Оновлення параметрів політики здійснювалося за алгоритмом PPO із використанням міні-пакетного градієнтного спуску.

З метою забезпечення відтворюваності результатів у всіх модулях, де це не порушувало логіку навчання, було зафіксовано значення random seed. Це дозволило мінімізувати вплив стохастичних

факторів та забезпечити стабільність експериментальних результатів.

У процесі дослідження було навчено декілька моделей із різними гіперпараметрами та тривалістю навчання. Відбір фінальної моделі для кожної гібридної схеми здійснювався на основі інтегральної оцінки якості розкладів на валідаційній вибірці. При цьому враховувався не лише мінімальний досягнутий показник оцінки, а й стабільність та відтворюваність отриманих результатів на інших наборах даних.

Для схеми RL-GH-CPOP спостерігалося стабільне та відтворюване покращення порівняно з базовою евристикою як на контрольних точках із найкращим значенням інтегральної оцінки (близько 0.0157), так і на більш стабільних контрольних точках з показником близько 0.065. Обидва варіанти демонстрували узгоджене покращення на різних тестових сценаріях.

Натомість для HG-RL-HEFT найкращі контрольні точки з мінімальним значенням оцінки (приблизно 0.148) часто характеризувалися нестійкою поведінкою: покращення спостерігалося лише на окремих (першому) тестових сценаріях, тоді як на інших екземплярах результати були близькими до базової евристики або навіть гіршими. Використання більш стабільної контрольної точки (близько 0.067) дозволило отримати повністю відтворюване та стабільне покращення на всіх тестових екземплярах, що і стало підставою для вибору фінальної моделі.

Таким чином, у межах даного дослідження критерій відбору фінальної моделі враховував компроміс між мінімальним значенням інтегральної оцінки та стабільністю узагальнення на нові сценарії.

Необов'язковий модуль локального покращення (L, Local post-processing) у межах даного експериментального дослідження за замовчуванням було вимкнено. Додаткові експерименти показали, що його внесок у покращення метрик є незначним (у межах статистичної похибки), тому основна увага зосереджена на ефекті саме гібридної інтеграції евристики та RL-компонента. Таким чином, організація експерименту забезпечує:

- навчання на максимально різноманітному потоці екземплярів;
- незалежне тестування на типовому сценарії;
- відтворюваність результатів;
- коректне порівняння базових та гібридних методів.

Налаштування алгоритму навчання з підкріпленням для обох гібридних схем виконувалося на основі алгоритму Proximal Policy Optimization (PPO), реалізованого в межах бібліотеки Stable-Baselines3. Для моделі RL-GH-CPOP використовувалися такі гіперпараметри: коефіцієнт навчання learning_rate = 0.0003, коефіцієнт дисконтування $\gamma = 0.99$, параметр GAE $\lambda = 0.95$, коефіцієнт ентропійної регуляризації ent_coef = 0.01, коефіцієнт функції цінності vf_coef = 0.5, обмеження норми градієнта max_grad_norm = 0.5, параметр обрізання clip_range = 0.2, кількість кроків збору траєкторії n_steps = 2, розмір міні-пакета batch_size = 6 та кількість епох оптимізації n_epochs = 2. Невелике значення n_steps обумовлене специфікою схеми RL-GH-CPOP, у якій агент приймає рішення лише один раз на епізод (rg_action_frequency = "once"), а подальше формування розкладу виконується детермінованою евристикою CPOP.

Для моделі HG-RL-HEFT застосовувалися аналогічні базові параметри PPO (learning_rate = 0.0003, $\gamma = 0.99$, $\lambda = 0.95$, ent_coef = 0.01, vf_coef = 0.5, clip_range = 0.2, max_grad_norm = 0.5), однак із суттєво більшими значеннями n_steps = 72, batch_size = 72 та n_epochs = 3. Це пов'язано з тим, що у схемі HG-RL-HEFT агент приймає рішення на кожному кроці побудови розкладу, тобто кількість взаємодій із середовищем у межах одного епізоду є значно більшою. Відповідно, збільшення rollout-фрагмента дозволяє стабілізувати оцінку переваги (advantage) та зменшити дисперсію градієнта при оновленні політики.

Гібридні конфігурації також відрізнялися параметрами інтеграції з евристичними модулями. Для RL-GH-CPOP використовувалося обмеження topk = 5 та два режими роботи з критичним шляхом (soft, hard), а вагові коефіцієнти функції винагороди становили $\alpha = 1.0$ та $\beta = 0.1$. Для HG-RL-HEFT застосовувалося topk = 8 із меншою вагою штрафу за дисбаланс ($\beta = 0.05$), що дозволяло агенту агресивніше оптимізувати makespan із помірним урахуванням балансування навантаження. В обох випадках використовувалася стратегія маскування допустимих дій ready_only та нормалізація завантаження процесорів у векторі стану.

Тривалість експерименту визначалась планом навчання, що передбачав 10 контрольних точок (checkpoints), по 12 екземплярів для оцінювання на кожній контрольній точці, із фіксованим seed = 12345. Така схема дозволила порівнювати динаміку якості моделі на різних етапах навчання та обирати фінальний варіант не лише за мінімальним значенням інтегральної оцінки, а й за стабільністю результатів.

Обрана конфігурація експериментального плану є результатом компромісу між статистичною надійністю оцінювання та обчислювальними витратами. Кількість паралельних середовищ n_envs=6 визначалася апаратними можливостями платформи: при меншій кількості не повністю використовувалися доступні ресурси процесора та GPU, тоді як збільшення до 8–12 середовищ призводило до суттєвого (у 1.5–2 рази) уповільнення навчання через накладні витрати синхронізації та зростання затримок обробки rollout-фрагментів. Відповідно, кількість екземплярів на контрольну точку було обрано кратною 12, що дозволяє ефективно розподіляти сценарії між 6 паралельними середовищами та забезпечує коректне агрегування результатів.

Використання 10 контрольних точок у межах одного експерименту є достатнім для формування моделі з інтегральною оцінкою на рівні близько 0.05 та прийнятною стабільністю узагальнення. Практичні спостереження показали, що контрольна точка з оптимальним співвідношенням мінімального зна-

чення оцінки та стабільності результатів зазвичай з'являється у межах перших десяти ітерацій. Подальше збільшення кількості контрольних точок лише розширює вибірку, але не призводить до якісно нового покращення політики, водночас суттєво збільшуючи час експерименту.

Середній час навчання моделі HG-RL-HEFT становить 25–32 хвилини, тоді як для RL-GH-CPOP – 1–2 хвилини через значно меншу кількість взаємодій із середовищем. Тестування усіх пар контрольних точок займає додатково 8–10 хвилин, що в сумі формує типовий час одного повного експерименту близько 40 хвилин. Збільшення кількості екземплярів на контрольну точку до 24 дозволило б дещо підвищити статистичну стабільність оцінювання, проте практичні результати показали, що це не суттєво впливає на загальну тенденцію метрик, натомість подвоює обчислювальні витрати.

Таким чином, обрана конфігурація експериментального плану забезпечує оптимальний баланс між тривалістю експерименту та якістю отриманої оцінки, дозволяючи в стислі строки перебирати варіації ключових гіперпараметрів (n_steps, batch_size, n_epochs) та знаходити їх раціональне співвідношення, яке забезпечує як прийнятний рівень інтегральної оцінки, так і стабільність узагальнення.

Після завершення навчання було проведено порівняння базових евристик HEFT та CPOP із відповідними гібридними модифікаціями на послідовності типових тестових сценаріїв. Агреговані результати для типового сценарію з 20 незалежних екземплярів наведено у табл. 1.

Таблиця 1 – **Порівняння агрегованих метрик базових та гібридних методів**

Метод	Makespan	Avg Completion Time	Avg Load	LB Std	LB CV
HEFT	57.1236	26.9557	22.1507	3.2682	0.1478
HG-RL-HEFT	53.3066	24.9436	22.3979	4.6743	0.2181
Δ (%)	+6.68	+7.46	−1.12	−43.03	−47.55
CPOP	64.8530	29.4950	22.7164	3.5333	0.1568
RL-GH-CPOP	55.3996	25.4111	23.2109	3.7997	0.1690
Δ (%)	+14.58	+13.85	−2.18	−7.54	−7.82

З наведених результатів видно, що для обох гібридних схем спостерігається суттєве зменшення makespan. Для HG-RL-HEFT середнє скорочення становить 6.68 %, тоді як для RL-GH-CPOP – 14.58 %. Аналогічна тенденція спостерігається і для середнього часу завершення задач (Average Task Completion Time), де покращення становить відповідно 7.46 % та 13.85 %. Значне покращення спостерігається і для Average Task Communication Time, але ця метрика він не є суттєвою для даної моделі.

Водночас для метрик балансування навантаження (Avg Load , LB Std та LB CV) спостерігається певне погіршення порівняно з базовими евристиками. Зокрема, у випадку HG-RL-HEFT стандартне відхилення навантаження зросло на 43.03 % (варіюється у діапазоні 35-45%), а коефіцієнт варіації – на 47.55 % (зазвичай 36-48%). Для RL-GH-CPOP зростання є менш вираженим (7–8 %). На фоні відчутного погіршення стандартного відхилення та коефіцієнту варіацій, середнє навантаження процесорів залишається у нормі з погіршенням лише на 1-2%. Ці погіршення пояснюються тим, що функція винагороди орієнтована насамперед на мінімізацію makespan, а балансування навантаження виступає як допоміжний штрафний компонент із меншою вагою β. Таким чином, отримані результати підтверджують, що гібридна інтеграція RL-компонента дозволяє суттєво скоротити загальний час виконання workflow, навіть якщо це супроводжується помірним зростанням нерівномірності завантаження процесорів. З практичної точки зору така поведінка є очікуваною, оскільки мінімізація критичного шляху часто потребує концентрації задач на швидших процесорах.

Графічний аналіз результатів дозволяє оцінити характер змін показників на рівні окремих тестових екземплярів та простежити стабільність отриманих покращень. На рис. 4 представлено порівняння значень Makespan для всіх чотирьох методів: HEFT, HG-RL-HEFT, CPOP та RL-GH-CPOP.

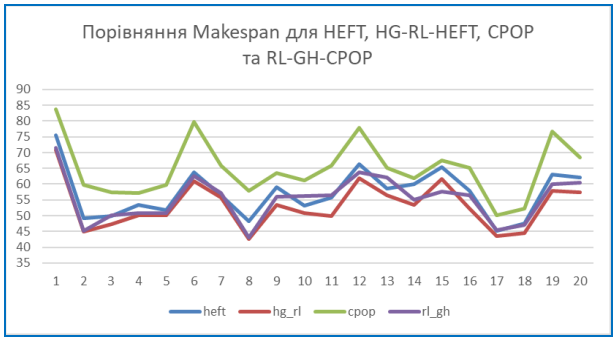

Рис. 4. Порівняння Makespan для HEFT, HG-RL-HEFT, CPOP та RL-GH-CPOP

З графіка видно, що обидві гібридні модифікації систематично зменшують загальний час виконання workflow порівняно з відповідними базовими евристиками. При цьому RL-GH-CPOP демонструє найбільш виражене зниження makespan, що узгоджується з агрегованим покращенням на рівні близько 14–15 %. HG-RL-HEFT забезпечує помірне, але стабільне скорочення (приблизно 6–7 %). Візуально лінії гібридних методів у більшості точок розташовані нижче відповідних базових алгоритмів, що свідчить про узгоджений характер покращення на різних екземплярах, а не лише про локальні виграші.

На рис. 5 наведено порівняння середнього часу завершення задач (Average Task Completion Time) для всіх чотирьох методів.

Форма кривих загалом повторює тенденцію makespan, що є очікуваним, оскільки зменшення довжини критичного шляху та оптимізація розподі-

лу задач скорочують не лише максимальний час завершення, але й середній час виконання задач. Найбільше зниження спостерігається для RL-GH-CPOP, де покращення перевищує 13 %. HG-RL-HEFT також демонструє стабільне зменшення показника. Важливо, що жоден із гібридних методів не демонструє систематичного погіршення середнього часу завершення, що підтверджує ефективність інтеграції RL-компонента.

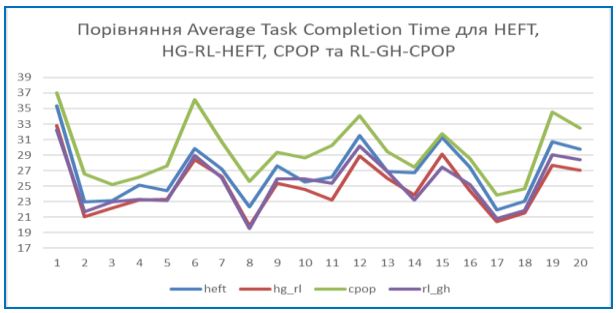

Рис. 5. Порівняння Average Task Completion Time для HEFT, HG-RL-HEFT, CPOP та RL-GH-CPOP

Average Task Communication Time також повторює ту саму тенденцію, але тут значення для базових евристик (HEFT, CPOP) повністю рівні, а у rl-методів окрім значного зниження, спостерігається невеликий шум, при чому HG-RL-HEFT трохи гірший за RL-GH-CPOP приблизно на 0,1-0,2%.

На рис. 6 подано порівняння середнього навантаження процесорів (Average Load).

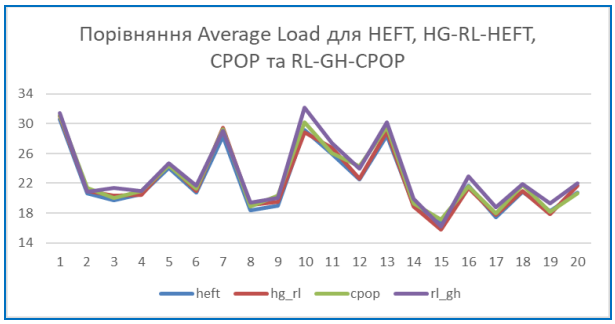

Рис. 6. Порівняння Average Load для HEFT, HG-RL-HEFT, CPOP та RL-GH-CPOP

На відміну від makespan, значення Average Load для всіх методів є близькими. Це пояснюється тим, що загальний обсяг обчислень у межах одного workflow є сталим, а зміни стосуються переважно розподілу задач між процесорами. Невелике зростання середнього навантаження у гібридних методів (близько 1–2 %) є наслідком концентрації задач на швидших процесорах для скорочення критичного шляху. Візуально графік не демонструє різких відхилень або аномальних піків, що свідчить про збереження загальної стабільності розподілу обчислювальної роботи.

Метрики балансування навантаження характеризуються значно більшою дисперсією значень між екземплярами. З цієї причини результати для Standard Deviation–based Load Balance та Coefficient of Variation–based Load Balance представлено у вигляді попарних графіків, що дозволяє уникнути накладання ліній та покращити інтерпретацію тенденцій. На рис. 7 наведено порівняння стандартного відхилення навантаження (Standard Deviation–based Load Balance) для HEFT та HG-RL-HEFT.

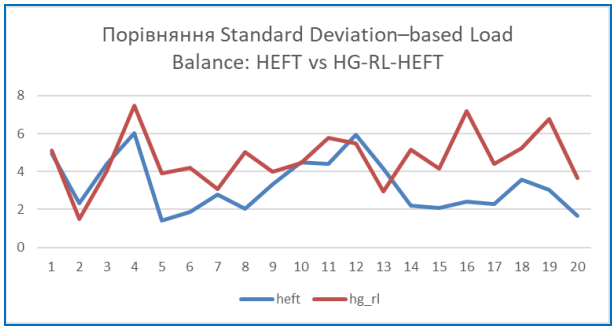

Рис. 7. Порівняння Standard Deviation–based Load Balance: HEFT vs HG-RL-HEFT

Графік демонструє, що HG-RL-HEFT у багатьох випадках має вищі значення стандартного відхилення навантаження порівняно з базовою евристикою. Це узгоджується з агрегованим зростанням показника приблизно на 40 %. Така поведінка пояснюється тим, що мінімізація makespan досягається за рахунок більш інтенсивного використання окремих (швидших) процесорів, що природно призводить до зростання нерівномірності розподілу задач.

На рис. 8 подано аналогічне порівняння для CPOP та RL-GH-CPOP.

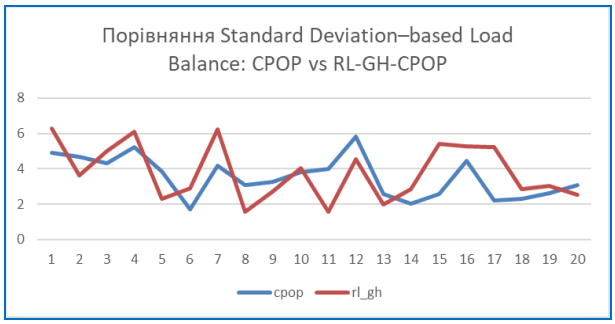

Рис. 8. Порівняння Standard Deviation–based Load Balance: CPOP vs RL-GH-CPOP

У цьому випадку збільшення дисбалансу є значно менш вираженим (близько 7–8 %). Лінії методів часто розташовані близько одна до одної, а пікові значення не мають різких відхилень. Це свідчить про те, що RL-GH-CPOP зберігає балансувальні властивості базової евристики, водночас забезпечуючи істотне зниження makespan.

Аналогічна структура результатів спостерігається для коефіцієнта варіації навантаження (Coefficient of Variation–based Load Balance). На рис. 9 наведено порівняння HEFT та HG-RL-HEFT.

Тут зростання показника є незначним та має стабільний характер, без різких піків. Це свідчить про більш збалансований компроміс між мінімізацією часу виконання та рівномірністю навантаження.

Таким чином, експериментальні результати підтверджують, що гібридні методи забезпечують

стабільне зменшення makespan та середнього часу завершення задач, при цьому характер впливу на балансування навантаження залежить від обраної схеми гібридизації (інтеграції RL-компонента).

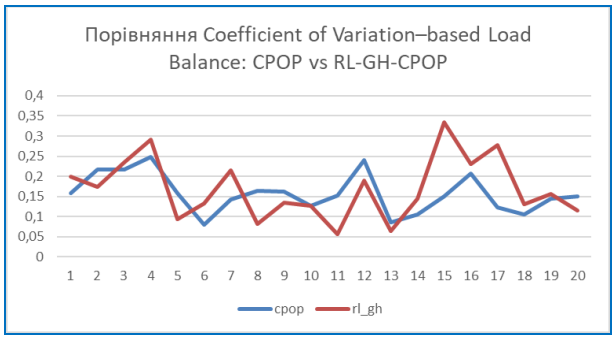

Рис. 9. Порівняння Coefficient of Variation–based Load Balance: CPOP vs RL-GH-CPOP

Висновки

У роботі розроблено модель планування задач у багатопроцесорних системах, що базується на гібридному підході та поєднує класичні евристичні алгоритми спискового планування (HEFT, CPOP) з адаптивним компонентом навчання з підкріпленням на основі алгоритму PPO. Запропонована модель формалізує взаємодію між середовищем, простором станів і дій, евристичним модулем та політикою агента, а також передбачає можливість регулювання компромісу між мінімізацією часу виконання задач і балансуванням навантаження за допомогою параметрів функції винагороди. Для перевірки ефективності підходу було реалізовано експериментальну інфраструктуру, що використовує потокову генерацію графів задач (DAG) та векторизоване середовище навчання. Така організація експерименту дала змогу сформувати узагальнену політику планування для широкого спектра сценаріїв із різною кількістю задач, різним рівнем гетерогенності процесорів та варіативністю комунікаційних параметрів, а також забезпечити стабільність і відтворюваність отриманих результатів.

Проведене експериментальне дослідження показало, що інтеграція компонентів навчання з підкріпленням у класичні евристичні методи планування дозволяє досягти систематичного зменшення часу виконання розкладу (makespan) та середнього часу завершення задач. Найбільш стабільне покращення продемонструвала схема RL-GH-CPOP, тоді як підхід HG-RL-HEFT забезпечує більш агресивне скорочення часу виконання за рахунок деякого зростання нерівномірності навантаження. Це підтверджує можливість керованого компромісу між часовими показниками виконання та ефективністю використання обчислювальних ресурсів.

Практична значущість запропонованого підходу полягає у можливості його застосування в гетерогенних багатопроцесорних середовищах, зокрема у високопродуктивних обчислювальних системах, хмарних та розподілених платформах, де важливо забезпечити мінімальний час виконання workflow за умови раціонального використання обчислювальних ресурсів.

Отримані результати свідчать, що гібридні методи планування, які поєднують евристичні алгоритми та навчання з підкріпленням, є перспективним напрямом розвитку систем керування обчислювальними ресурсами та можуть стати основою для подальших досліджень у сфері інтелектуального планування в складних обчислювальних середовищах.

Конфлікт інтересів. Автори декларують, що не мають конфлікту інтересів стосовно даного дослідження, в тому числі фінансового, особистісного характеру, авторства чи іншого характеру, що міг би вплинути на дослідження та його результати, представлені в даній статті.

Використання засобів штучного інтелекту. Автори підтверджують, що не використовували технології штучного інтелекту при створенні представленої роботи.

Список літератури

1. Shaima Rahim et al. A survey of machine learning-driven task scheduling approaches for multiprocessor systems. *Journal of Systems Architecture*. 2025. Vol. 171. DOI: https://doi.org/10.1016/j.sysarc.2025.103628
2. Xu Jian et al. Real-time scheduling of parallel tasks with tight deadlines. *Journal of Systems Architecture*. 2020. Vol. 108. DOI: https://doi.org/10.1016/j.sysarc.2020.101742
3. Wenfan Zhang, Haijiao Ou. Reinforcement learning based multi objective task scheduling for energy efficient and cost effective cloud edge computing. *Scientific Reports*. 2025. DOI: https://doi.org/10.1038/s41598-025-25666-1
4. Пасічник М. Ю., Зайцев В. Г. Методи диспетчеризації завдань у системах реального часу. *Комп'ютерно-інтегровані технології: освіта, наука, виробництво*. 2024. №56. DOI: https://doi.org/10.36910/6775-2524-0560-2024-56-04
5. Gaurav Agarwal, et al. Multiprocessor task scheduling using multi-objective hybrid genetic Algorithm in Fog–cloud computing. *Knowledge-Based Systems*. 2023. Volume 272. DOI: https://doi.org/10.1016/j.knosys.2023.110563
6. Kusay Hameed Al-Salami, Zaid Taha Sawadi. Task Scheduling for Multiprocessor Systems Using Queuing Theory. *Computer Engineering and Intelligent Systems*. 2016. Vol. 7, No. 2. URL: https://www.iiste.org/Journals/index.php/CEIS/article/view/28602
7. Зайцев В. Г., Цибаєв Є. І. Оцінка часових характеристик задач в багатопроцесорних системах реального часу з використанням сіток Петрі. *Управління розвитком складних систем*. 2020. № 42. С. 43–50. DOI: https://doi.org/10.32347/2412-9933.2020.42.43-50
8. Wafa Hantom et al. A Survey on Scheduling Algorithms in Real-Time Systems. *International Journal of Computer Science and Network Security*. 2022. Vol. 22, No. 4. DOI: https://doi.org/10.22937/IJCSNS.2022.22.4.80
9. Люлька А. В. Методи та засоби планування обчислювальних завдань в комп'ютерній системі: робота на здобуття кваліфікаційного ступеня магістра: спец. 123 - комп'ютерна інженерія. Тернопільський національний технічний університет імені Івана Пулюя. 2024. 74 с. URL: https://elartu.tntu.edu.ua/handle/lib/48105
10. Голубничий Д. Ю., Головченко О. С. Застосування алгоритмів рангового підходу при плануванні розподілу задач в багатопроцесорних системах *Радіотехніка*. 2024. Вип. 219. С. 16–27. DOI: https://doi.org/10.30837/rt.2024.4.219.02

11. Станко П., Охремчук О., Саламатіна Д., Свердлова Д. Оптимізація планування завдань в розподілених обчислювальних системах реального часу. *Наукоємні технології*. 2023. №4(60). С. 386–393. DOI: https://doi.org/10.18372/2310-5461.60.18267
12. Головченко О. С. Методи планування розподілу задач в багатопроцесорних системах : поясн. записка до кваліф. роботи здобувача вищої освіти на другому (магістерському) рівні, спец. 123 Комп'ютерна інженерія. М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. 2025. 87 с. URL: https://openarchive.nure.ua/handle/document/32427
13. Yoni Birman et al. Hierarchical Deep Reinforcement Learning Approach for Multi-Objective Scheduling with Varying Queue Sizes. arXiv. 2020. DOI: https://doi.org/10.48550/arXiv.2007.09256
14. Yihong Li et al. Task Placement and Resource Allocation for Edge Machine Learning: A GNN-based Multi-Agent Reinforcement Learning Paradigm. arXiv. 2023. DOI: https://doi.org/10.48550/arXiv.2302.00571
15. Junyoung Park, et al. ScheduleNet: Learn to solve multi-agent scheduling problems with reinforcement learning. arXiv. 2021. DOI: https://doi.org/10.48550/arXiv.2106.03051
16. Zheng Xu, et al. Enhancing Kubernetes Automated Scheduling with Deep Learning and Reinforcement Techniques for Large-Scale Cloud Computing Optimization. *arXiv*. 2024. DOI: https://doi.org/10.48550/arXiv.2403.07905
17. Xiao Fang Li. Simulation on Task Scheduling for Multiprocessors Based on Improved Neural Network. *Applied Mechanics and Materials*. 2014. Vol. 513-–517. P. 2293–2296. DOI: https://doi.org/10.4028/www.scientific.net/AMM.513-517.2293
18. Олександренко М. А. Система для моделювання багатопроцесорних алгоритмів планування : пояснювальна записка до кваліфікаційної роботи здобувача вищої освіти на першому (бакалаврському) рівні, спеціальність 123 Комп'ютерна інженерія. М-во освіти і науки України, Харків. нац. ун-т радіоелектроніки. 2025. 55 с. DOI: https://openarchive.nure.ua/handle/document/32703

Received (Надійшла) 22.12.2025
Accepted for publication (Прийнята до друку) 01.04.2026
Publication date (Дата публікації) 22.05.2026

ВІДОМОСТІ ПРО АВТОРІВ / ABOUT THE AUTHORS

Бондаренко Сергій Вікторович – студент кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Serhii Bondarenko –student, Department of Electronic Computing Machines, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: serhii.bondarenko@nure.ua; ORCID Author ID: https://orcid.org/0009-0002-8255-1764.

Мартовицький Віталій Олександрович – кандидат технічних наук, доцент**,** доцент кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Vitalii Martovytskyi – Candidate of Technical Sciences, Associate Professor, Associate Professor of Electronic Computing Machines Department, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: vitalii.martovytskyi@nure.ua; ORCID ID: https://orcid.org/0000-0003-2349-0578;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57196940070.

Бологова Наталія Миколаївна – кандидат технічних наук, доцент**,** доцент кафедри електронних обчислювальних машин, Харківський національний університет радіоелектроніки, Харків, Україна;
Nataliia Bolohova – Candidate of Technical Sciences, Associate Professor, Associate Professor of Electronic Computing Machines Department, Kharkiv National University of Radio Electronics, Kharkiv, Ukraine;
e-mail: nataliia.bolohova@nure.ua; ORCID ID: https://orcid.org/0000-0001-8927-0055;
Scopus Author ID: https://www.scopus.com/authid/detail.uri?authorId=57203140922&origin=resultslist.

Рикун Володимир Георгійович – кандидат технічних наук, доцент**,** доцент Харківського національного університету Повітряних Сил ім. І. Кожедуба, Харків, Україна;
Volodymyr Rykun – Candidate of Technical Sciences, Associate Professor, Associate Professor of Ivan Kozhedub Kharkiv National Air Force University,Kharkiv, Ukraine;
e-mail: Volodymyr_Rykun@gmail.com; ORCID ID: https://orcid.org/0000-0003-0162-1178.

Tasks scheduling in multi-processor systems based on hybrid methods

Serhii Bondarenko, Vitalii Martovytskyi, Nataliia Bolohova, Volodymyr Rykun

Abstract. Relevance. The rapid development of multiprocessor and distributed computing systems necessitates improving task scheduling efficiency in heterogeneous computing environments. Efficient resource utilization and reduction of execution time are critical challenges for modern high-performance and cloud computing systems. **Object of study:** task scheduling processes in heterogeneous multiprocessor systems. **Purpose of the article:** to improve scheduling efficiency by developing a hybrid approach that combines classical heuristic algorithms with reinforcement learning methods. **Research results.** The paper analyzes existing task scheduling approaches, including list heuristics and machine learning-based methods. A functional scheduling model integrating the HEFT and CPOP algorithms with a reinforcement learning agent based on the Actor–Critic architecture using the Proximal Policy Optimization algorithm is proposed. A mathematical model of the scheduling environment considering DAG structure, expected time to compute matrix, and load balancing metrics is developed. Experimental evaluation demonstrates that the proposed hybrid approach reduces makespan by 5–7% and 13–15% and improves average task completion time compared to baseline heuristics while maintaining balanced resource utilization. **Conclusions.** The proposed hybrid scheduling model enables adaptive adjustment of heuristic algorithms through reinforcement learning in heterogeneous multiprocessor environments and demonstrates potential for improving performance and resource utilization in high-performance, cloud, and distributed computing systems. **Scope of application of the obtained results:** high-performance computing systems, cloud infrastructures, and distributed computing environments for improving task scheduling efficiency and resource utilization.

Keywords: task scheduling, multiprocessor systems, hybrid methods, reinforcement learning, PPO (Proximal Policy Optimization), CPOP (Critical Path on a Processor), load balancing.