Conclusion. *From a practical perspective, the results of this work highlight the importance of thorough data preparation and the development of adaptive strategies for cross-domain training. Despite these challenges, developing models capable of reliably operating on heterogeneous data remains a promising direction, as this approach most closely reflects the conditions of real-world medical practice.*

Keywords*: deep learning, transfer learning, computer vision, convolutional neural networks, automated melanoma detection, data domain, class imbalance, medical image classification.*

Одержано редакцією 22.07.2025 р.
Прийнято до публікації 24.09.2025 р.

УДК 004.75:004.052.42

DOI 10.31651/2076-5886-2025-1-18-32

PACS 07.05.Bx, 07.05.Mh

КИРПА Григорій Анатолійович
студент спеціальності «Прикладна математика» Черкаського національного університету імені Богдана Хмельницького

ДЗЮБА Вікторія Анатоліївна
кандидат технічних наук, викладач кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: viktoriya.dzyuba15@gmail.com
ORCID 0000-0003-1655-0333

ГЛАДКА Людмила Іванівна
к.ф.-м.н., доцент кафедри автоматизації та комп'ютерно-інтегрованих технологій, Черкаський національний університет імені Б. Хмельницького
e-mail: l_i_gladka@vu.cdu.edu.ua
ORCID 0000-0002-7030-9666

ПОРІВНЯЛЬНИЙ АНАЛІЗ АЛГОРИТМІВ ПЛАНУВАННЯ РЕСУРСІВ У ХМАРНИХ ОБЧИСЛЮВАЛЬНИХ СЕРЕДОВИЩАХ

У статті досліджуються алгоритми планування обчислювальних ресурсів у хмарних середовищах. Проведено порівняльний аналіз класичних та евристичних методів розподілу завдань: First Come First Served (FCFS), Round Robin (RR), Longest Processing Time (LPT), Shortest Processing Time (SPT) та Weighted Round Robin (WRR). Задача розподілу ресурсів формалізована як задача мінімізації показника Makespan у гетерогенному обчислювальному середовищі. Для проведення експериментів розроблено модульний програмний комплекс на мові Python, який реалізує імітаційне моделювання хмарних навантажень при трьох рівнях інтенсивності вхідного потоку: низькому (10 завдань), середньому (20 завдань) та високому (40 завдань). Оцінювання якості алгоритмів здійснюється за допомогою системи метрик, що включає Makespan, стандартне відхилення завантаженості вузлів, коефіцієнт варіації (CV) та інтегральний показник ефективності Score, який поєднує якість розкладу і обчислювальні витрати на планування. Встановлено, що алгоритм LPT демонструє найкращі показники Makespan за умов середнього та високого навантаження, тоді як WRR забезпечує найвищі значення інтегрального показника в сценаріях з гетерогенними вузлами. Результати підтверджують відсутність універсального оптимального алгоритму та обґрунтовують

доцільність впровадження адаптивних систем планування, здатних динамічно перемикатися між стратегіями залежно від характеристик поточного навантаження.

Ключові слова: *хмарні обчислення, планування ресурсів, алгоритми розподілу завдань, Makespan, балансування навантаження, LPT, Round Robin, WRR, інтегральний показник ефективності.*

Вступ

Стрімкий розвиток хмарних технологій та масштабна цифровізація економіки перетворили задачу ефективного управління обчислювальними ресурсами на один із ключових чинників конкурентоспроможності сучасних IT-інфраструктур [1]. Збільшення обсягів даних і ускладнення завдань, що обробляються у хмарі, вимагають впровадження інтелектуальних методів планування, здатних мінімізувати час виконання розкладу (Makespan) і забезпечити рівномірне завантаження серверів. Ефективне планування ресурсів безпосередньо впливає на дотримання угод про рівень послуг (SLA) і на операційні витрати провайдерів хмарних сервісів [2].

Хмарна парадигма базується на наданні обчислювальних потужностей як послуги, що вимагає від постачальників забезпечення якісного обслуговування при одночасному скороченні витрат. Проблема ефективного розподілу ресурсів є однією з центральних у цій галузі, оскільки динамічний характер запитів користувачів та неоднорідність фізичної інфраструктури ускладнюють прийняття рішень у реальному часі [3]. Планування ресурсів у хмарних середовищах – це процес призначення набору завдань на наявні обчислювальні вузли таким чином, щоб оптимізувати цільові показники продуктивності.

Серед ключових особливостей хмарного планування, що визначають складність задачі, виділяють такі: гетерогенність обчислювального середовища, що передбачає використання вузлів з різними технічними характеристиками процесорів, обсягом оперативної пам'яті та швидкістю дискових підсистем; еластичність ресурсів, яка дозволяє динамічно масштабувати кількість доступних віртуальних машин; багатокритеріальність оптимізації, оскільки планувальник повинен одночасно враховувати час виконання, енергоспоживання та вартість оренди ресурсів [4].

У науковій літературі задача планування ресурсів у хмарі класифікується як NP-повна задача комбінаторної оптимізації [2], що обумовлює необхідність застосування евристичних методів, які дозволяють знаходити прийнятні рішення за поліноміальний час. Задача ускладнюється відсутністю повної інформації про майбутні запити та необхідністю миттєвої реакції на зміни в стані інфраструктури.

Незважаючи на широку наявність досліджень окремих алгоритмів, порівняльний аналіз їхньої ефективності в однакових умовах тестування при різних рівнях навантаження є недостатньо систематизованим у вітчизняній науковій літературі. Зокрема, переважна більшість наявних робіт розглядає алгоритми ізольовано або тестує їх на синтетичних рівномірних навантаженнях, що не відображає реальних умов роботи промислових хмарних платформ. Крім того, у більшості досліджень оцінювання обмежується лише показником Makespan, без урахування обчислювальних витрат самого планувальника – витрат, які в масштабних хмарах, де планування виконується тисячі разів на секунду, можуть виявитися суттєвим вузьким місцем. Потреба у створенні комплексних інструментів для симуляції та порівняння різних стратегій планування на реальних моделях навантаження, а також у розробці багатокритеріального показника оцінювання обумовлює актуальність даної роботи.

Мета статті – дослідження ефективності класичних і евристичних алгоритмів планування завдань у хмарних середовищах на основі імітаційного моделювання та розробки комплексної методики їхньої кількісної оцінки. Для досягнення поставленої мети необхідно: проаналізувати методологічну базу алгоритмів розподілу завдань;

формалізувати задачу планування; реалізувати програмний модуль для порівняльного тестування; провести серію експериментів при різних рівнях навантаження; запропонувати та апробувати інтегральний показник ефективності, що враховує як якість розкладу, так і обчислювальні витрати на роботу самого планувальника.

Виклад основного матеріалу:

1. Огляд методів розв'язання задачі

Алгоритми розподілу завдань є механізмом реалізації обраної стратегії планування. За десятиліття розвитку обчислювальних систем було розроблено велику кількість методів – від найпростіших детермінованих схем до складних інтелектуальних систем. У контексті хмарних обчислень найпоширенішими є класичні алгоритми, що базуються на чергах, та евристичні підходи, орієнтовані на мінімізацію затримок або збалансування навантаження.

Основні концептуальні стратегії, що визначають логіку прийняття рішень у системах планування, можна систематизувати таким чином [5, 6]:

1. *Статичне планування* базується на припущенні, що всі параметри завдань та стан ресурсів відомі заздалегідь. Це дозволяє використовувати потужні математичні методи для знаходження глобального оптимуму, проте цей підхід є вразливим до будь-яких відхилень від початкових даних.
2. *Динамічне планування* адаптується до поточних змін у системі та здатне реагувати на раптову появу нових завдань або вихід з ладу обчислювальних вузлів, що робить його стандартом для комерційних хмарних платформ.
3. *Централізоване планування* використовує єдиний логічний центр управління. Це забезпечує точність розподілу, але обмежує масштабованість при зростанні кількості вузлів до тисяч одиниць.
4. *Децентралізоване планування* передає функції прийняття рішень локальним агентам. Це забезпечує високу відмовостійкість, проте часто призводить до локальних екстремумів.
5. *Гібридне планування* намагається поєднати переваги централізованого контролю з гнучкістю розподілених систем, використовуючи ієрархічні структури управління [7].

Серед конкретних алгоритмів, що реалізують ці стратегії, особливої уваги заслуговують п'ять методів, які є найбільш поширеними у комерційних хмарних платформах.

First Come, First Served (FCFS) – базовий метод, де завдання обробляються в порядку їхнього надходження. Його головною перевагою є простота реалізації і відсутність додаткових витрат на аналіз характеристик завдань. Проте у гетерогенних середовищах FCFS часто демонструє низьку ефективність, оскільки коротке завдання може тривалий час очікувати в черзі за ресурсомістким процесом, що призводить до простою частини ресурсів [8].

Round Robin (RR) використовує часові кванти для розподілу ресурсів. Кожному завданню виділяється фіксований проміжок часу на процесорі, після чого воно передається наступному об'єкту в черзі. Це забезпечує рівномірність обробки запитів, але вибір неоптимального розміру кванта може призвести або до надмірних витрат на перемикання контексту, або наблизити поведінку алгоритму до FCFS [9].

Longest Processing Time (LPT) спрямований на мінімізацію загального часу виконання розкладу (Makespan). Логіка полягає у першочерговому призначенні найдовших завдань на вільні ресурси. Це дозволяє уникнути ситуації, коли в кінці циклу обробки велике завдання виконується на одному вузлі, тоді як інші вузли простоюють. Математично доведено, що LPT забезпечує результат, який не перевищує

оптимальний більше ніж у 4/3 рази [10].

Shortest Processing Time (SPT) фокусується на мінімізації середнього часу перебування завдання в системі. Пріоритет надається завданням з найменшою тривалістю виконання. Це дозволяє швидко вивільняти чергу, що є корисним для систем з великою кількістю дрібних запитів, проте великі завдання можуть страждати від «голодування» (starvation) – постійно відкладатися на користь нових коротких задач [11].

Weighted Round Robin (WRR) є модифікацією RR, яка враховує різну продуктивність обчислювальних вузлів. Кожному вузлу присвоюється вага, пропорційна його потужності. Вузли з більшою вагою отримують більше завдань або довші кванти часу. Це робить алгоритм ефективним у гетерогенних хмарах, де використання однакових квантів на швидких і повільних процесорах було б нераціональним [12].

Окрему увагу слід приділити етапам процесу планування, які в реальних системах виконуються послідовно. Перший етап – виявлення ресурсів – передбачає збір актуальної інформації про стан фізичних та віртуальних вузлів, їхню доступність і поточне завантаження. Другий – фільтрація та вибір вузлів – відсіює ресурси, які не відповідають технічним вимогам конкретного завдання (наприклад, за обсягом RAM). Третій – ранжування ресурсів – присвоює вузлам пріоритет на основі прогнозованої ефективності виконання завдання. Четвертий – призначення та виконання – передбачає безпосередню передачу завдання на обраний вузол і запуск моніторингу. П'ятий – оновлення стану системи – актуалізує інформацію у базі даних планувальника після завершення обробки. Таке структурування процесу дозволяє мінімізувати помилки при розподілі навантаження та забезпечує прозорість роботи системи [2].

Аналіз наведених методів дозволяє зробити висновок, що жодного універсального алгоритму не існує: детерміновані методи (FCFS, RR) найкраще підходять для однорідних навантажень з низькою варіативністю, пріоритетні методи (LPT, SPT) потребують попереднього знання часу виконання завдань, а гетерогенні алгоритми (WRR) є необхідними для реальних хмарних структур з різними поколіннями обладнання.

2. Постановка задачі

Задачу розподілу ресурсів у хмарних обчислювальних середовищах формалізуємо як процес динамічного призначення незалежних завдань на множину доступних серверів. Нехай задано фіксований набір серверів $S = \{s_1, s_2, \ldots, s_n\}$, де n змінюється від 2 до 6, та пакет завдань $T = \{t_1, t_2, \ldots, t_m\}$, де m залежить від рівня навантаження. Кожне завдання t_i має випадкову тривалість виконання d_i, рівномірно розподілену в діапазоні [1, 10] умовних одиниць.

Ефективність розкладу вважається високою, якщо всі обчислювальні вузли завершують роботу приблизно одночасно, уникаючи ситуацій, коли один сервер перевантажений, а інші простоюють. Формально, задача є NP-повною при $n \geq 3$ серверах і довільних тривалостях завдань, що обумовлює застосування евристичних підходів.

Головна мета планування – мінімізувати показник *Makespan M*, що визначається як загальний час від початку виконання першого завдання до завершення останнього на найбільш завантаженому сервері [13]:

$$M = \max_j \left(C_j \right), \tag{1}$$

де C_j – час завершення виконання всіх завдань на j-му обчислювальному вузлі.

Для оцінки якості балансування навантаження аналізується завантаженість кожного вузла. Нехай L_j – сумарний час роботи j-го вузла. Тоді середня завантаженість обчислюється як:

$$L_{avg} = \frac{1}{n}\sum_{j=1}^{n} L_j \ . \tag{2}$$

Нерівномірність розподілу кількісно оцінюється через стандартне відхилення σ, що демонструє абсолютне розсіювання значень завантаженості навколо середнього рівня:

$$\sigma = \sqrt{\frac{1}{n}\sum_{j=1}^{n}\left(L_j - L_{avg}\right)^2} \ . \tag{3}$$

Оскільки системи з різними масштабами навантаження необхідно порівнювати між собою, застосовується відносний показник – коефіцієнт варіації V, що обчислюється як відношення стандартного відхилення до середньої завантаженості:

$$V = \frac{\sigma}{L_{avg}} . \tag{4}$$

Якщо V перевищує 30%, це вказує на суттєвий дисбаланс у розподілі ресурсів.

Для комплексної оцінки, що поєднує часові та ресурсні параметри, запропоновано інтегральний показник ефективності *Score* – зважена сума нормованих значень *Makespan* і часу виконання самого алгоритму планування:

$$Score = \alpha \cdot \frac{M}{M_{max} + \beta} \cdot \frac{t_{exec}}{t_{exec\,max}}, \tag{5}$$

де α = 0.7 та β = 0.3 – вагові коефіцієнти значущості *Makespan* та часу роботи планувальника відповідно, M_{max} і $t_{exec\,max}$ – максимальні значення відповідних показників серед усіх алгоритмів у поточному сценарії. Вибір ваг обґрунтований пріоритетністю кінцевого результату для користувача (швидке завершення завдань) при одночасному врахуванні витрат керуючого вузла. Чим менше значення *Score*, тим ефективнішим є алгоритм.

Параметри експерименту охоплюють три рівні навантаження: Low Load (10 завдань), Medium Load (20 завдань), High Load (40 завдань). Кожен цикл тестів повторюється 10 разів для отримання усереднених статистично стійких значень та мінімізації впливу випадкових відхилень при генерації вхідних даних.

3. Методи розв'язання: програмна реалізація

Для проведення порівняльного аналізу розроблено модульний програмний комплекс мовою Python. Вибір Python як мови реалізації обумовлений розвиненою екосистемою бібліотек для наукових обчислень (NumPy, Pandas), можливостями статистичної обробки даних та зручністю швидкого прототипування. Такий підхід дозволяє відокремити логіку планування від логіки збору статистики та візуалізації. Ключовою особливістю реалізації є використання структури «словника стратегій»

(dictionary of strategies) – шаблону проєктування, що забезпечує легке масштабування системи та уніфікований інтерфейс тестування:

```
ALGORITHMS = {
    "FCFS": fcfs_scheduler,
    "Round Robin": round_robin_scheduler,
    "LPT (Longest Processing Time)": lpt_scheduler,
    "SPT (Shortest Processing Time)": spt_scheduler,
    "Weighted Round Robin": wrr_scheduler
}
```

Така структура дозволяє додавати нові алгоритми (наприклад, на основі генетичних методів) просто шляхом додавання нового запису до словника, не змінюючи логіку проведення самих експериментів.

Архітектура програмного модуля включає три основні компоненти:

– *модуль симуляції (run_experiments)* – відповідає за створення віртуального середовища, де сервери представлені як масиви, що накопичують тривалість призначених завдань. Ядро симуляції виконує роль оркестратора – ініціалізує стан середовища, генерує ідентичну послідовність завдань для всіх алгоритмів у межах однієї ітерації та фіксує метрики;
– *модуль аналітики*: розраховує статистичні метрики (M , σ , V , *Score*) відразу після завершення кожного прогону;
– *модуль експорту*: автоматично трансформує результати у формат Microsoft Excel та будує графічні залежності, що є критичним для обробки великих масивів даних.

Ключовою умовою об'єктивності є забезпечення чистоти експерименту: перед запуском кожного нового алгоритму стан серверів повністю обнуляється, а для кожного прогону генерується ідентична послідовність завдань для всіх методів у межах однієї ітерації. Це гарантує, що різниця у результатах зумовлена виключно логікою планування, а не випадковими факторами.

Для вимірювання часу роботи алгоритму використовується функція `time.perf_counter()`, яка забезпечує прецизійне вимірювання з роздільністю в наносекундах, що є необхідним для коректного порівняння методів, час виконання яких відрізняється на порядки величини.

Методика генерації навантажень ґрунтується на імітаційному моделюванні з використанням генератора псевдовипадкових чисел для відтворення хаотичного характеру реального хмарного навантаження. Тривалість кожного завдання задається рівномірним розподілом на відрізку [1, 10] умовних одиниць, що забезпечує достатню варіативність для прояву структурних переваг евристичних підходів порівняно з детермінованими методами. Параметри тестування налаштовуються через конфігураційний файл, що дозволяє автоматично масштабувати кількість серверів та завдань без модифікації вихідного коду. Після виконання 10 незалежних повторень на кожній конфігурації результати усереднюються, що мінімізує вплив статистичних викидів при генерації вхідних послідовностей.

4. Отримані результати

Сценарій низького навантаження (Low Load: 10 завдань)

Перший етап аналізу присвячений оцінці роботи алгоритмів за умов низького навантаження (Low Load), що відповідає надходженню 10 завдань на групу від 2 до 6 серверів (рис. 1). Даний сценарій моделює роботу невеликих приватних хмар або крайових обчислювальних вузлів (Edge Computing), де ресурси обмежені, а кількість запитів є мінімальною.

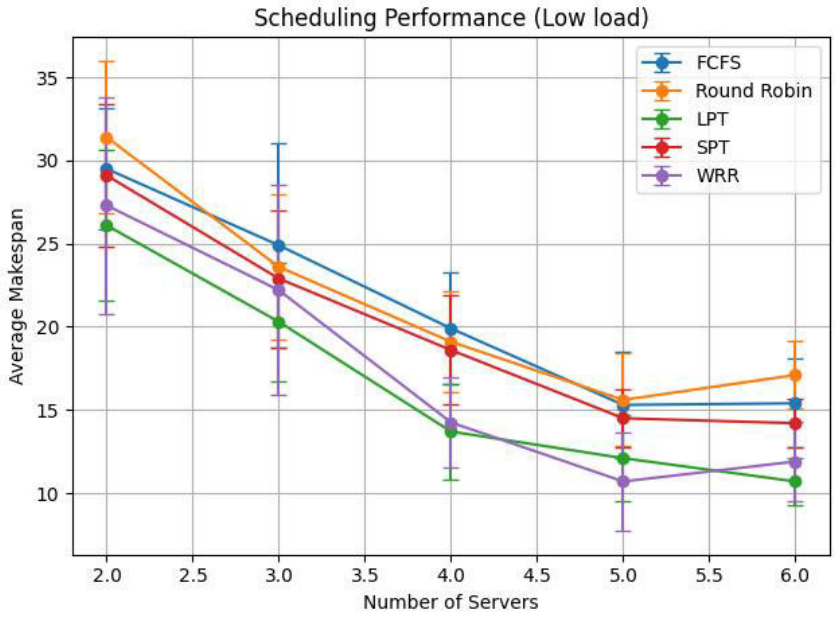

Рис. 1. Продуктивність алгоритмів планування (низьке навантаження). Побудовано автором на основі результатів імітаційного моделювання

На графіку простежується закономірність зниження загального часу виконання розкладу зі збільшенням кількості доступних обчислювальних вузлів для всіх досліджуваних методів. При мінімальній конфігурації з 2 серверами різниця між алгоритмами є найбільш помітною, що пояснюється обмеженим простором для маневрування ресурсами.

Алгоритм LPT демонструє найкращий показник Makespan у більшості конфігурацій: стратегія першочергового призначення найдовших завдань дозволяє максимально щільно заповнити обчислювальні потужності вже на перших ітераціях. За умов 6 серверів LPT зберігає перевагу над базовим FCFS у 10-12%, що підтверджує ефективність пріоритизації навіть за незначних обсягів даних.

Аналіз коефіцієнта варіації для низького навантаження свідчить про певну нестабільність класичних методів FCFS та Round Robin. Зокрема, FCFS при використанні 3 серверів демонструє підвищений CV, що вказує на значну залежність результату від випадкового порядку надходження завдань. Евристичні підходи SPT та LPT демонструють більш передбачувану поведінку, хоча SPT поступається значенні Makespan через ефект накопичення довгих завдань наприкінці розкладу.

Статистичні дані підтверджують, що для 6 серверів значення Makespan у всіх алгоритмів починають зближуватися, оскільки ресурсів стає достатньо для паралельної обробки більшості завдань. Однак навіть у таких умовах LPT зберігає лідерство.

Сценарій середнього навантаження (Medium Load: 20 завдань)

Перехід до середнього навантаження дозволяє оцінити масштабованість алгоритмів та їхню здатність до ефективного перерозподілу ресурсів при зростанні щільності вхідного потоку (рис. 2). Цей етап симуляції є найбільш наближеним до реальних корпоративних середовищ, де спостерігається постійна, але помірна активність користувачів.

Результати вказують на стабілізацію переваги алгоритму LPT: при збільшенні кількості серверів до 6 він забезпечує Makespan на рівні 17.5 умовних одиниць, що є суттєво кращим порівняно з FCFS або SPT. LPT успішно уникає ситуацій, коли великі завдання залишаються на фінальну стадію розкладу, завантажуючи сервери рівномірно протягом усього циклу.

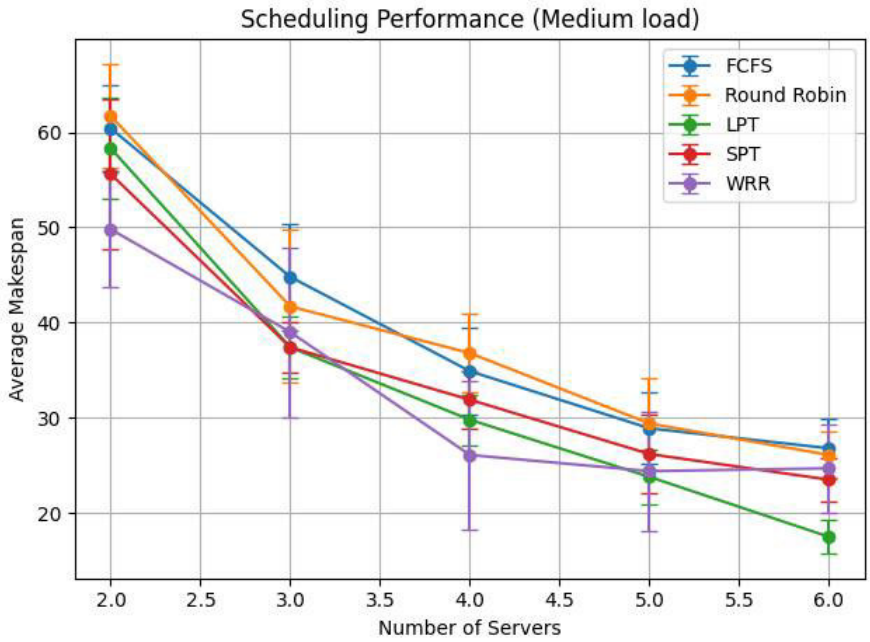

Рис. 2. Продуктивність алгоритмів планування (середнє навантаження). Побудовано автором на основі результатів імітаційного моделювання

Особливий інтерес викликає поведінка алгоритму WRR, який демонструє різке зниження Makespan при 4 серверах, проте це супроводжується аномально високим коефіцієнтом варіації. Це вказує на нерівномірність завантаження вузлів через специфіку вагових коефіцієнтів, коли частина ресурсів перевантажена, тоді як інші залишаються недовантаженими.

Детерміновані методи Round Robin та FCFS показують стійку лінійну залежність від кількості серверів, проте їхня ефективність залишається на 15-20% нижчою за спеціалізовані евристики. Це пояснюється тим, що при 20 завданнях ймовірність нерівномірного розподілу навантаження зростає, і прості стратегії не в змозі компенсувати цей дисбаланс. Алгоритм SPT демонструє середні показники – через «хвіст» довгих накопичених завдань наприкінці розкладу його Makespan є гіршим, ніж у LPT, попри швидке спустошення черги.

Статистичний аналіз свідчить, що оптимальна точка використання ресурсів досягається при 4–5 серверах. Подальше розширення інфраструктури до 6 вузлів дає менший приріст продуктивності, що вказує на досягнення межі ефективності алгоритмів при даній кількості завдань.

Сценарій високого навантаження (High Load: 40 завдань)

Найбільш показовим є дослідження алгоритмів в умовах високого навантаження, де 40 завдань значно перевищують кількість доступних ресурсів. Цей сценарій моделює ситуацію пікового навантаження на дата-центри, коли черга запитів є критично довгою, а час простою обладнання має бути зведеним до нуля (рис. 3).

В умовах дефіциту ресурсів розрив між результатами різних методів набуває критичного значення. Алгоритм LPT продовжує утримувати лідерство за показником Makespan, забезпечуючи найшвидше завершення всієї пачки завдань навіть при мінімальній кількості серверів. Його стратегія дозволяє ефективно «ущільнювати» розклад, що при 40 завданнях дає суттєву економію часу порівняно з іншими підходами.

Round Robin при великій кількості завдань починає демонструвати стабільні результати завдяки закону великих чисел: середня тривалість завдань у круговій черзі

вирівнюється, що знижує ймовірність критичного перекосу навантаження.

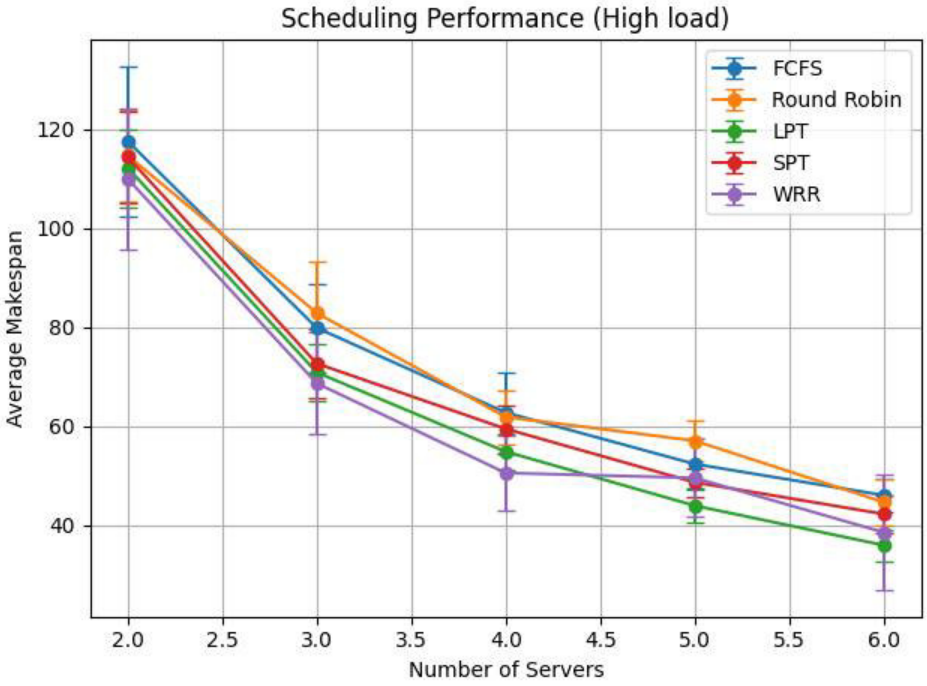

Рис. 3. Продуктивність алгоритмів планування (високе навантаження). Побудовано автором на основі результатів імітаційного моделювання

Алгоритм FCFS показує найвищі значення стандартного відхилення при 2 серверах серед усієї серії експериментів, що остаточно підтверджує його непридатність для висококонкурентних систем. Випадковий розподіл великих завдань у довгій черзі створює некеровані затримки, що робить FCFS найбільш ризикованим вибором для комерційних хмарних сервісів із суворими вимогами SLA.

SPT при 40 завданнях показує найгірші результати за Makespan через проблему «голодування» великих завдань: поки виконуються десятки дрібних запитів, ресурсомісткі задачі чекають у черзі, що розтягує загальний час завершення. Примітно, що CV для WRR при високому навантаженні досягає значення 30,36%, що вказує на роботу деяких серверів на межі потужності.

Загальний аналіз часових показників за трьома сценаріями дозволяє стверджувати, що зі зростанням обсягу завдань переваги інтелектуальних евристик стають вирішальними. Якщо при низькому навантаженні помилка у виборі алгоритму коштує частки секунди, то при високій інтенсивності вона призводить до затримок, що сягають 40% від загального часу виконання. Це обґрунтовує необхідність впровадження складних систем планування в сучасні центри обробки даних. Зокрема, стандартне відхилення для FCFS при 2 серверах у сценарії High Load набуває максимальних значень у всій серії, що остаточно підтверджує його непридатність для висококонкурентних хмарних платформ з вимогами до гарантованого часу відгуку.

Порівняльний аналіз за інтегральним показником ефективності

Для отримання комплексної оцінки, що враховує не лише час завершення розкладу, а й обчислювальну складність самого алгоритму, було використано інтегральний показник *Score*. Нагадаємо, що цей параметр поєднує нормований Makespan (вага 0.7) та нормований час роботи алгоритму (вага 0.3).

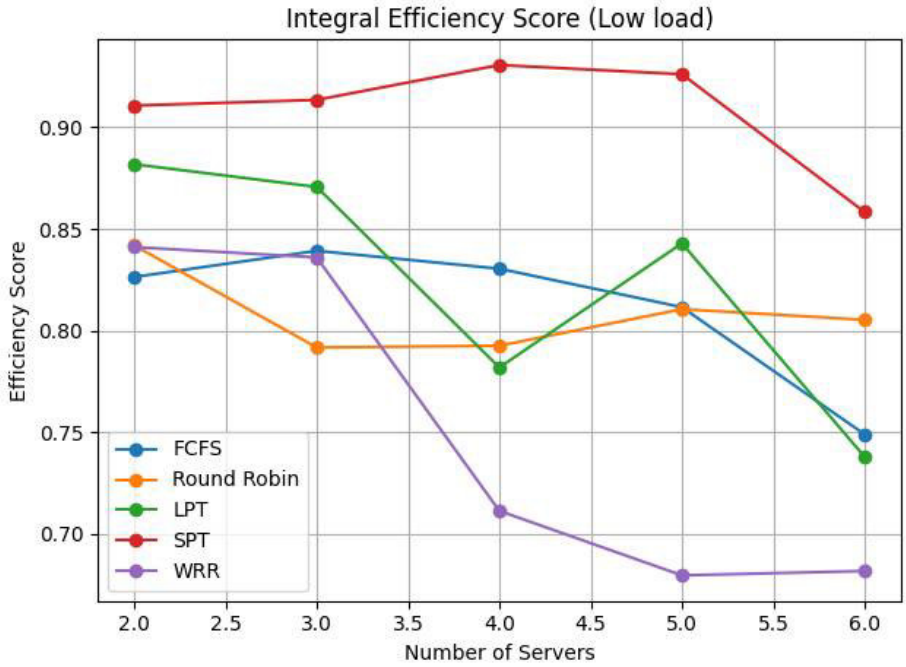

Рис. 4. Інтегральний показник ефективності (низьке навантаження). Побудовано автором на основі результатів імітаційного моделювання

За умов низької інтенсивності вхідного потоку найбільш збалансовані результати демонструє алгоритм WRR. Його здатність враховувати продуктивність вузлів забезпечує високу якість розкладу при мінімальних витратах обчислювального часу на саме планування. Це дозволяє системі швидко реагувати на запити, не витрачаючи зайві цикли процесора на складне сортування даних.

При середньому навантаженні пріоритети зміщуються в бік алгоритму LPT. Отримані дані підтверджують, що витрати на реалізацію стратегії «найдовше завдання першим» повністю виправдовують себе за рахунок значного скорочення *Makespan*. Round Robin утримує конкурентні позиції завдяки екстремальній швидкодії, що робить його ефективною альтернативою у сценаріях, де важлива швидкість ініціалізації обчислень.

В умовах екстремального навантаження інтегральний показник виявляє суттєву перевагу WRR, який стає найбільш збалансованим інструментом управління ресурсами. Висока щільність завдань вимагає від системи максимальної утилізації кожного вузла відповідно до його потужності, що і забезпечує ваговий підхід. LPT при 40 завданнях дещо втрачає позиції в рейтингу *Score*: витрати на сортування та постійний пошук мінімального навантаження починають відбирати значний час, що знижує його загальну привабливість у динамічних системах.

Узагальнені результати за ключовими метриками при максимальній кількості серверів (6 вузлів) представлено в таблиці 1.

Алгоритм LPT демонструє найнижчий *Makespan* (36.0 умовних одиниць) серед усіх досліджених методів, що свідчить про його найвищу ефективність у мінімізації загального часу виконання завдань. Це робить його оптимальним для сценаріїв з високим навантаженням, де критично важливе швидке виконання обчислень. WRR показує найкращі результати за інтегральним показником при низькому навантаженні (0.682) і зберігає лідерство при високому (0.716), що підкреслює його переваги у гетерогенних середовищах з урахуванням вагових коефіцієнтів серверів. Натомість SPT демонструє найгірші значення Score при обох рівнях навантаження, підтверджуючи

непридатність алгоритму для пакетної обробки задач, де ключовим критерієм є Makespan.

Таблиця 1

Узагальнені показники ефективності алгоритмів (6 серверів)

Алгоритм	**Makespan (High)**	**CV (%) (High)**	**Score (Low)**	**Score (High)**
FCFS	46.1	7.04	0.749	0.785
Round Robin	44.7	10.34	0.805	0.76
LPT	36	8.78	0.738	0.847
SPT	42.3	9.06	0.858	0.923
WRR	38.6	30.36	0.682	0.716

Примітка: *Score* – нормований інтегральний показник (чим менше, тим краще); CV – коефіцієнт варіації завантаженості серверів.

5. Аналіз результатів дослідження та наукова новизна

Проведене дослідження дозволяє сформулювати ряд важливих узагальнень щодо порівняльної ефективності алгоритмів планування ресурсів у хмарних середовищах.

По-перше, зі зростанням обсягу завдань переваги інтелектуальних евристик стають вирішальними. Якщо при низькому навантаженні помилка у виборі алгоритму коштує частки секунди, то при високій інтенсивності вона призводить до затримок, що сягають 40% від загального часу виконання. Це обґрунтовує необхідність впровадження складних систем планування в сучасні центри обробки даних.

По-друге, встановлено, що не існує універсального алгоритму, оптимального для всіх сценаріїв. Вибір має залежати від пріоритетів конкретного застосування: для максимальної швидкості виконання пакетних задач обирають LPT; для збалансованості та економії ресурсів керуючого вузла в гетерогенному середовищі – WRR; для систем реального часу з інтенсивним вхідним потоком коротких запитів доцільно використовувати Round Robin або FCFS завдяки їхній мінімальній обчислювальній складності.

По-третє, запропонований інтегральний показник Score є науковою новизною роботи. На відміну від традиційного підходу, що оцінює алгоритми виключно за Makespan або CV, показник Score враховує «ціну» отриманого розкладу – обчислювальні витрати на роботу самого планувальника. Це важливо для систем реального часу, де занадто повільний планувальник може стати вузьким місцем і знизити пропускну здатність системи. Формула (5) з ваговими коефіцієнтами $\alpha = 0.7$ та $\beta = 0.3$ дозволяє системному архітектору гнучко налаштовувати баланс між якістю розкладу і витратами на його формування.

По-четверте, отримані результати підтверджують теоретичні передбачення щодо поведінки алгоритмів:

- гарантія LPT про відхилення від оптимуму не більше ніж у 4/3 рази [10] підтверджується експериментально для всіх сценаріїв навантаження;
- висока варіація CV для WRR (30.36% при High Load) свідчить про необхідність впровадження механізмів зворотного зв'язку для динамічної корекції вагових коефіцієнтів вузлів залежно від їхньої реальної поточної продуктивності;
- проблема «голодування» в SPT [11] чітко проявляється при обробці пакетів із значною варіативністю тривалості завдань.

По-п'яте, аналіз показника CV дозволяє виявити важливу залежність: алгоритми з низьким значенням Makespan не завжди забезпечують рівномірне завантаження

серверів. Зокрема, WRR при 2 серверах демонструє суттєву нерівномірність розподілу, що в реальних умовах може призвести до передчасного зносу або перегріву «сильних» вузлів. Це підкреслює необхідність використання комплексної системи метрик замість одного показника при проєктуванні систем управління хмарними ресурсами.

По-шосте, результати підтверджують доцільність впровадження концепції адаптивних «інтелектуальних перемикачів алгоритмів» (algorithm selectors) у сучасні Cloud Management Systems. Такий підхід передбачає безперервний моніторинг характеристик вхідного потоку (розміру черги, дисперсії тривалості завдань, кількості активних вузлів) і динамічний вибір оптимального планувальника на основі правил або навченої моделі класифікації. Наприклад, при коефіцієнті варіації вхідних даних нижче 20% можна автоматично перемикатися на Round Robin або FCFS, заощаджуючи ресурси керуючого вузла, а при зростанні гетерогенності завдань – задіювати LPT або WRR. Таке адаптивне управління дозволяє досягти компромісу між якістю розкладу і накладними витратами, що особливо важливо для мультитенантних хмарних платформ, де одночасно обслуговуються тисячі різнотипних запитів.

Практичне значення отриманих результатів полягає у формуванні конкретних рекомендацій для розробників хмарного програмного забезпечення та системних адміністраторів. По-перше, застосування LPT у сценаріях пакетної обробки (Big Data аналітика, рендеринг, наукові симуляції) дозволяє скоротити загальний час оренди хмарних потужностей на 15–20% порівняно з базовими методами. По-друге, для систем реального часу з інтенсивним вхідним потоком коротких запитів (IoT, мікросервісні архітектури, API-шлюзи) доцільно використовувати Round Robin або FCFS, які мінімізують накладні витрати планувальника і забезпечують передбачуваний час відгуку. По-третє, для хмарних середовищ з гетерогенними серверами різних поколінь оптимальним вибором є WRR, який автоматично враховує відмінності у продуктивності вузлів при розподілі навантаження. По-четверте, для збалансованого розвитку інфраструктури провайдерам варто орієнтуватися на інтегральний показник Score для динамічного перемикання між стратегіями залежно від поточної черги – підхід, що є логічним підґрунтям для впровадження інтелектуальних перемикачів алгоритмів у сучасні Cloud Management Systems.

Висновки

У статті проведено комплексне дослідження алгоритмів планування обчислювальних ресурсів у хмарних середовищах з використанням методів імітаційного моделювання. Основна увага приділена порівняльному аналізу п'яти методів розподілу завдань (FCFS, Round Robin, LPT, SPT, WRR) за критеріями часової ефективності та збалансованості навантаження.

На основі проведеного дослідження можна зробити такі висновки:

1. Задача розподілу ресурсів у хмарних обчислювальних середовищах формалізована як задача мінімізації Makespan у системі з гетерогенними вузлами. Встановлено, що для об'єктивного порівняння алгоритмів необхідна багатокритеріальна система метрик, яка враховує як якість розкладу, так і обчислювальні витрати на його формування.
2. Алгоритм LPT продемонстрував найкращі показники Makespan в усіх сценаріях навантаження, і особливо при обробці великих пакетів завдань (High Load). Стратегія першочергової обробки найдовших завдань дозволяє ефективно заповнювати обчислювальні потужності та уникати критичного перекосу наприкінці циклу обробки.
3. Алгоритм WRR забезпечує найкращий інтегральний показник Score у сценаріях з гетерогенними серверами та при динамічному навантаженні. Разом з тим висока

варіація CV (30,36%) при пікових навантаженнях вказує на необхідність механізмів адаптивної корекції вагових коефіцієнтів.

4. Класичні методи Round Robin та FCFS демонструють задовільну ефективність для простих сценаріїв з однорідним навантаженням та обмеженими ресурсами керуючого вузла, однак поступаються евристикам на 15–20% у середовищах з нерівномірним вхідним потоком.
5. Запропонований інтегральний показник Score є практичним інструментом для порівняльної оцінки алгоритмів планування, що враховує «ціну» отриманого результату. Гнучка система вагових коефіцієнтів у формулі показника дозволяє адаптувати вибір алгоритму залежно від пріоритетів конкретної системи.

Перспективою подальших досліджень є розробка гібридних адаптивних систем планування, які динамічно перемикаються між стратегіями залежно від поточних характеристик навантаження. Перспективним напрямом також є застосування методів машинного навчання – зокрема, моделей регресії та нейронних мереж – для прогнозування часу виконання завдань ще до їхнього потрапляння в чергу планувальника. Такий прогностичний підхід дозволив би усунути одне з ключових обмежень алгоритмів LPT та SPT, які потребують апріорного знання тривалості задач. Окрім того, доцільним є дослідження поведінки алгоритмів в умовах часткових відмов вузлів та мережевих затримок між компонентами розподіленої інфраструктури – сценаріїв, які в рамках даної роботи не розглядалися, проте є критичними для реальних промислових хмарних платформ.

Список використаної літератури:

1. Шишкіна М. Тенденції розвитку і стандартизації вимог до засобів ІКТ навчального призначення на базі хмарних обчислень. Науковий вісник Мелітопольського державного педагогічного університету. Серія: Педагогіка. 2014. № 2. С. 223–231.
2. Гребенюк Д. С. Аналіз методів розподілення ресурсів у середовищах віртуалізації. Системи управління, навігації та зв'язку. 2018. № 6. С. 98–103.
3. Петровська І., Кучук Г. Розподіл обчислювальних ресурсів у хмарних системах. Системи управління, навігації та зв'язку. 2022. Вип. 2 (68). С. 75–78.
4. Сидор К., Щербина Ю. Технологія хмарних обчислень: архітектура, моделі та аспекти інформаційної безпеки. Information Systems and Networks. 2025. Issue 18, part 2. P. 184–191.
5. Дайновський Ю. А., Гліненко Л. К. Бізнес-моделі хмарного надання ІТ-послуг. Маркетинг і цифрові технології. 2019. Т. 3, № 2. С. 18–44.
6. Kosarevskyi B., Tetskyi A. Сучасні підходи до розгортання інфраструктури мобільних інтелектуальних систем. Innovative Technologies and Scientific Solutions for Industries. 2025. No. 2 (32). P. 33–48.
7. Новохатський Д. Є. Оптимізаційна багатофакторна модель оцінки показників функціонування розподіленої комп'ютерної системи: дипломна робота бакалавра. Київ: Національний технічний університет України «КПІ імені Ігоря Сікорського», 2023. 120 с.
8. FCFS – First Come First Serve CPU Scheduling. GeeksforGeeks. URL: https://www.geeksforgeeks.org/dsa/first-come-first-serve-cpu-scheduling-non-preemptive/ (дата звернення: 31.12.2025).
9. Round Robin Scheduling in Operating System. GeeksforGeeks. URL: https://www.geeksforgeeks.org/operating-systems/round-robin-scheduling-in-operating-system/ (дата звернення: 31.12.2025).
10. Anwar A., Rochman D. D., Ferdian R. Parallel Machine Scheduling with Shortest Processing Time (SPT) and Longest Processing Time (LPT) to Minimize Makespan at PT. ABC. Rigeo. 2021. Vol. 11, No. 6.
11. Shortest Processing Time Definition. Fiveable. URL: https://fiveable.me/key-terms/introduction-industrial-engineering/shortest-processing-time (дата звернення: 31.12.2025).
12. Round Robin (RR) vs. Weighted Round Robin (WRR). FS Community. URL: https://www.fs.com/blog/round-robin-rr-vs-weighted-round-robin-wrr-7151.html (дата звернення: 31.12.2025).

13. Зайцев С. В., Василенко В. М., Семендяй С. М. Огляд адаптивних методів забезпечення достовірності передачі інформації при використанні завадостійкого кодування у системах бездротового зв'язку. Інформатика та математичні методи в моделюванні. 2021. С. 277.

References:

1. Shishkina, M. Trends in the Development and Standardization of Requirements for Cloud-Based Educational ICT Tools. Scientific Bulletin of Melitopol State Pedagogical University. Series: Pedagogy. 2014. No. 2. pp. 223–231.
2. Grebenyuk, D. S. Analysis of Resource Allocation Methods in Virtualization Environments. Control, Navigation, and Communication Systems. 2018. No. 6. Pp. 98–103.
3. Petrovska, I., & Kuchuk, G. Allocation of Computing Resources in Cloud Systems. Control, Navigation, and Communication Systems. 2022. Issue 2 (68). Pp. 75–78.
4. Sydor K., Shcherbina Y. Cloud computing technology: architecture, models, and information security aspects. Information Systems and Networks. 2025. Issue 18, part 2. P. 184–191.
5. Dainovskyi Yu. A., Hlinenko L. K. Business models for cloud-based IT service delivery. Marketing and Digital Technologies. 2019. Vol. 3, No. 2. P. 18–44.
6. Kosarevskyi B., Tetskyi A. Modern approaches to deploying the infrastructure of mobile intelligent systems. Innovative Technologies and Scientific Solutions for Industries. 2025. No. 2 (32). P. 33–48.
7. Novokhatskyi D. E. An optimization-based multifactor model for evaluating the performance indicators of a distributed computer system: bachelor's thesis. Kyiv: National Technical University of Ukraine "Igor Sikorsky KPI," 2023. 120 pp.
8. FCFS – First Come First Serve CPU Scheduling. GeeksforGeeks. URL: https://www.geeksforgeeks.org/dsa/first-come-first-serve-cpu-scheduling-non-preemptive/ (accessed: 12/31/2025).
9. Round Robin Scheduling in an Operating System. GeeksforGeeks. URL: https://www.geeksforgeeks.org/operating-systems/round-robin-scheduling-in-operating-system/ (accessed December 31, 2025).
10. Anwar A., Rochman D. D., Ferdian R. Parallel machine scheduling using shortest processing time (SPT) and longest processing time (LPT) to minimize makespan in PT. ABC. Rigeo. 2021. Vol. 11, No. 6.
11. Determining the shortest processing time. Fiveable. URL: https://fiveable.me/key-terms/introduction-industrial-engineering/shortest-processing-time (accessed: 12/31/2025).
12. Round Robin (RR) vs. Weighted Round Robin (WRR). FS Community. URL: https://www.fs.com/blog/round-robin-rr-vs-weighted-round-robin-wrr-7151.html (accessed: 12/31/2025).
13. Zaitsev S. V., Vasilenko V. M., Semendyai S. M. Review of adaptive methods for ensuring the reliability of information transmission using error-correcting coding in wireless communication systems. Informatics and Mathematical Methods in Modeling. 2021. P. 277.

KIRPA Grigory,
Student, Department of Applied Mathematics and Informatics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
DZYUBA Viktoriya,
PhD in Technical Sciences, Lecturer, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
HLADKA Liudmyla,
PhD in Physical and Mathematical Sciences, Associate Professor, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine

COMPARATIVE ANALYSIS OF RESOURCE SCHEDULING ALGORITHMS IN CLOUD COMPUTING ENVIRONMENTS

Summary. Introduction. *The rapid growth of cloud technologies and large-scale digitalisation of the economy have made efficient management of computing resources one of the key factors in the competitiveness of modern IT infrastructures. The problem of effective resource allocation directly affects compliance with Service Level Agreements (SLA) and the operational costs of cloud service providers.*

Purpose *of this article is to study the effectiveness of classical and heuristic task scheduling algorithms in cloud environments based on simulation modelling and to develop a comprehensive methodology for their quantitative evaluation.*

Results. *A modular Python-based software complex was developed to conduct simulation experiments at three load levels: Low Load (10 tasks), Medium Load (20 tasks), and High Load (40 tasks) across clusters of 2 to 6 servers. Five algorithms were compared: FCFS, Round Robin, LPT, SPT, and Weighted Round Robin. A composite efficiency metric, Score, was proposed to jointly evaluate schedule quality (Makespan) and computational overhead of the scheduling algorithm itself. Experimental results confirm that the LPT algorithm achieves the best Makespan values under medium and high load conditions, while WRR delivers the best integral Score in heterogeneous server configurations. No universal optimal algorithm was identified.*

Conclusion. *The choice of scheduling algorithm must depend on system priorities: LPT is recommended for batch processing workloads where minimising total execution time is critical; WRR – for heterogeneous infrastructures where balanced resource utilisation is paramount; Round Robin and FCFS – for lightweight real-time scenarios with strict latency constraints. The proposed Score metric provides a practical tool for adaptive algorithm selection in Cloud Management Systems.*

Keywords: *cloud computing, resource scheduling, task distribution algorithms, Makespan, load balancing, LPT, Round Robin, WRR, integral efficiency metric.*

Одержано редакцією 04.11.2025 р.
Прийнято до публікації 17.12.2025 р.

УДК 004.42:004.421

DOI 10.31651/2076-5886-2025-1-32-45

PACS89.20.Ff, 02.70.Wz

ЛИСЕНКО Олександр Володимирович
студент спеціальності «Прикладна фізика та наноматеріали» Черкаського національного університету імені Богдана Хмельницького
e-mail: lysenko.oleksandr1623@vu.cdu.edu.ua

ТАТАРЧУК Євгеній Вікторович
кандидат фізико-математичних наук, доцент кафедри фізики Черкаського національного університету імені Богдана Хмельницького
e-mail: Tatarchuk@vu.cdu.edu.ua

РОЗРОБКА ЗАСТОСУНКУ З МЕТОЮ КОГНІТИВНОГО ТРЕНУВАННЯ ТА АЛГОРИТМУ ДЛЯ АВТОМАТИЧНОЇ ГЕНЕРАЦІЇ АЛГЕБРАЇЧНИХ РІВНОСТЕЙ

У статті розглянуто проблему зниження когнітивних функцій як у літніх людей, так і у молодого покоління, що пов'язано зі зменшенням навичок усного рахування на тлі зростаючої залежності від цифрових технологій. Проаналізовано існуючі ігрові рішення для когнітивної стимуляції (Math Puzzle, Math Fight, Mathemagics), визначено їх переваги та недоліки. На основі проведеного огляду запропоновано власний підхід до створення веб-застосунку для розвитку когнітивних здібностей, який реалізує автоматичну генерацію алгебраїчних рівностей із використанням власного алгоритму розбору виразів на основі абстрактного синтаксичного дерева (AST). Розроблена бібліотека дозволяє ефективно перевіряти алгебраїчні рівності з мінімальним споживанням оперативної пам'яті порівняно з вбудованою функцією eval у Python. Представлено архітектуру веб-застосунку, REST API та результати порівняльного аналізу продуктивності.