УДК 004.42
DOI https://doi.org/10.32689/maup.it.2025.4.2

Володимир БРОДКЕВИЧ
кандидат економічних наук, доцент кафедри комп'ютерних наук та інтелектуальних систем, ПрАТ «ВНЗ «Міжрегіональна Академія управління персоналом», v.brodkevych@gmail.com
***ORCID:** 0000-0003-4282-8888*

Олександр ЧЕРНІЧЕНКО
здобувач вищої освіти, ПрАТ «ВНЗ «Міжрегіональна Академія управління персоналом», alexchernichenko18@gmail.com
***ORCID:** 0000-0002-5486-7310*

РОЗРОБКА ВЕБ-ДОДАТКУ ДЛЯ УПРАВЛІННЯ ЗАДАЧАМИ З РЕАЛІЗАЦІЄЮ АЛГОРИТМІВ ПРІОРИТИЗАЦІЇ ТА СОРТУВАННЯ

Анотація. *Стаття присвячена розробленню веб-додатку для управління задачами з реалізацією інтелектуальних алгоритмів пріоритизації та сортування, що ґрунтуються на багатокритеріальному аналізі.*

Метою дослідження *є створення адаптивної системи, яка автоматично визначає порядок виконання завдань, зменшуючи когнітивне навантаження користувача та підвищуючи ефективність планування.*

У роботі застосовано методи багатокритеріальної оптимізації (MCDM), когнітивного моделювання та поведінкової аналітики. Алгоритм формує інтегральний індекс пріоритету на основі 20 критеріїв – часових, поведінкових, контекстних і мотиваційних. Для кожного критерію використовується нормалізація (лінійна або сигмоїдна), після чого виконується зважене агрегування. Система має механізм адаптації вагових коефіцієнтів залежно від дій користувача. Реалізацію алгоритму здійснено у середовищі React із використанням бібліотеки Zustand для управління станом і Local Storage для збереження даних. Тестування проводилося на контрольних наборах сценаріїв, що відтворювали реальні умови використання додатку.

Наукова новизна. *Уперше запропоновано модель інтелектуальної пріоритезації, яка поєднує багатокритеріальний підхід із поведінковим навчанням без залучення серверної інфраструктури. Розроблений алгоритм здатний локально адаптувати ваги критеріїв у браузері користувача, забезпечуючи індивідуалізовану динаміку прийняття рішень. Система не лише оцінює важливість завдань, а й прогнозує оптимальний час виконання з урахуванням енергетичного стану та контексту користувача.*

Висновки. *Результати експериментів показали підвищення індексу продуктивності на 23,5% і зменшення середнього часу виконання завдань на 20%. Інтеграція алгоритму у React-додаток продемонструвала високу швидкодію, стабільність і можливість масштабування. Практичне значення дослідження полягає у створенні архітектури, придатної для інтеграції з календарями, AI-модулями та корпоративними системами управління проєктами. У подальшому передбачається розширення моделі за рахунок глибокого навчання та пояснюваних AI-механізмів.*

Ключові слова: *веб-додаток, пріоритезація задач, багатокритеріальна модель, React, Zustand, адаптивний алгоритм, штучний інтелект.*

Volodymyr BRODKEVYCH, Oleksandr CHERNYCHENKO. DEVELOPMENT OF A WEB APPLICATION FOR TASK MANAGEMENT WITH IMPLEMENTATION OF PRIORITIZATION AND SORTING ALGORITHMS

Abstract. *The article presents the development of a web application for task management implementing intelligent prioritization and sorting algorithms based on multi-criteria analysis.*

The aim *of the study is to design an adaptive system that automatically determines the optimal order of task execution, reducing users' cognitive load and increasing productivity.*

The research applies methods of multi-criteria decision-making (MCDM), cognitive modeling, and behavioral analytics. The algorithm calculates an integrated priority index using 20 criteria – temporal, behavioral, contextual, and motivational – each normalized (linear or sigmoidal) and aggregated via weighted summation. Weight coefficients are dynamically adjusted according to user behavior. The algorithm is implemented in React using the Zustand library for state management and Local Storage for asynchronous data persistence. Experimental evaluation was carried out on control scenarios replicating real user behavior to measure performance, adaptability, and user satisfaction.

The scientific novelty. *For the first time, an adaptive prioritization model combining multi-criteria analysis with behavioral learning is proposed for local browser computation. The algorithm autonomously adapts to user interactions without relying on external servers or datasets. It not only ranks tasks by importance but also predicts the most suitable execution periods based on the user's energy level, context, and recent behavior, forming a personalized productivity profile.*

Conclusions. *Experimental results demonstrate a 23.5% improvement in task productivity and a 20% reduction in execution time compared with traditional sorting methods. Integration of the algorithm into a React-based application confirmed its efficiency, stability, and scalability. The practical significance lies in its applicability for integration with calendars, AI modules, and project management platforms. Future work will focus on enhancing the model with machine learning and explainable AI components to improve transparency and adaptability.*

Key words: *web application, task prioritization, multi-criteria model, React, Zustand, adaptive algorithm, artificial intelligence.*

Вступ. Сучасні підходи до управління часом і продуктивністю людини дедалі частіше спираються на використання цифрових технологій, що автоматизують процеси планування, контролю виконання та пріоритезації завдань. Попри розмаїття наявних рішень – від простих ToDo-додатків до корпоративних систем управління проєктами – більшість із них залишають основну функцію прийняття рішень за користувачем. Такий підхід призводить до перевантаження інформацією, прокрастинації, втрати фокусу та зниження ефективності діяльності.

Згідно з дослідженнями в галузі когнітивної психології та поведінкової економіки, людина витрачає до 20% щоденних ментальних ресурсів на вибір черговості виконання завдань. Це явище, відоме як decision fatigue, негативно впливає на здатність до концентрації та прийняття рішень, особливо в умовах високого темпу роботи й постійної зміни контексту [13]. Тому питання створення інтелектуальних інструментів, що здатні зменшити когнітивне навантаження та автоматизувати пріоритезацію, є надзвичайно актуальним як у науковому, так і у прикладному вимірі.

В останнє десятиріччя суттєво зріс інтерес до впровадження алгоритмів штучного інтелекту, машинного навчання та багатокритеріального аналізу у сферу персональної продуктивності [2; 3]. Такі технології вже застосовуються для побудови систем рекомендацій, розподілу навантаження у командах, аналізу поведінкових патернів користувачів. Однак у сегменті веб-додатків для індивідуального управління задачами переважають рішення з ручним визначенням пріоритетів, що не враховують контекст, стан користувача чи історію його дій. Це створює науково-практичну нішу для дослідження – розроблення адаптивного алгоритму, який поєднає багатокритеріальність оцінки із динамічним навчанням на основі зворотного зв'язку.

У межах цієї роботи розглядається саме така система – веб-додаток для управління задачами з реалізацією алгоритмів пріоритезації та сортування, створений у середовищі React. Об'єктом дослідження є процес інтелектуальної організації та ранжування задач у цифровому середовищі. Предметом – методи і моделі автоматизованої пріоритезації, що використовують багатокритеріальний підхід та адаптивні вагові коефіцієнти.

Розроблений алгоритм враховує понад двадцять критеріїв – від дедлайнів, важливості та типу діяльності до часу доби, рівня енергії користувача та частоти відкладання завдань. Кожен критерій нормалізується у межах [0; 1], після чого формується інтегральний індекс пріоритету за формулою зваженої суми. Завдяки поведінковій адаптації система самостійно коригує ваги залежно від того, як користувач взаємодіє із завданнями: виконує, переносить чи видаляє.

Актуальність теми визначається тим, що поєднання інтелектуальної пріоритезації із сучасними технологіями веброзробки відкриває можливість створення нового покоління продуктивних інструментів – не просто списків завдань, а адаптивних цифрових асистентів. З технічного боку це поєднання React, Zustand, Local Storage та функціонального підходу до обробки станів дає змогу реалізувати високошвидкісну взаємодію користувача з алгоритмом у реальному часі.

Постановка проблеми. У сучасному інформаційному суспільстві людина щодня стикається з великим обсягом завдань, повідомлень і рішень, які потребують пріоритезації. За даними когнітивних досліджень, кількість мікрорішень, що приймаються протягом робочого дня, перевищує 30 тисяч, а це безпосередньо впливає на продуктивність і психоемоційний стан користувача [9]. Водночас більшість цифрових систем управління завданнями залишають процес вибору порядку виконання на розсуд користувача, не враховуючи його поведінкових особливостей, контексту чи часу доби.

Така ситуація призводить до трьох основних проблем.

По-перше, зростає когнітивне навантаження – користувач витрачає значну частину ментальної енергії не на виконання, а на вибір завдання. По-друге, зменшується ефективність планування, адже ручна пріоритезація часто базується на емоційних чи ситуативних чинниках, а не на об'єктивних даних. По-третє, зростає рівень прокрастинації, оскільки без чітких алгоритмічних підказок користувач віддає перевагу простішим або приємнішим задачам, нехтуючи стратегічно важливими [6].

У цій площині наукову і практичну цінність становить розроблення інтелектуальних систем підтримки прийняття рішень, які здатні автоматично визначати оптимальну послідовність виконання завдань. Особливий інтерес викликають багатокритеріальні підходи, що дозволяють враховувати взаємодію численних факторів: терміновість, важливість, енергійність, контекст, попередній досвід користувача тощо [4].

Водночас виклики сучасної веброзробки вимагають, щоб такі системи були не лише аналітичними, а й технічно ефективними – швидкодіючими, автономними та здатними до навчання без залучення серверної інфраструктури. Тому актуальним завданням стає створення адаптивного алгоритму пріоритезації, який може працювати безпосередньо у браузері користувача, забезпечуючи баланс між інтелектуальністю й продуктивністю.

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

Розроблення таких систем має значний потенціал для застосування у трьох ключових напрямах:

1. Особисте управління часом (Personal Productivity) – мінімізація decision fatigue, оптимізація добового циклу завдань, підвищення усвідомленості у плануванні.

2. Освітні та навчальні середовища – адаптація навантаження студентів за рівнем енергії, пріоритизація академічних цілей, автоматичне формування графіків навчання.

3. Корпоративні системи управління проєктами – розподіл пріоритетів у командах, синхронізація між індивідуальними та колективними цілями, прогнозування дедлайнів і ризиків.

З огляду на ці завдання, постає потреба у створенні веб-додатку нового покоління, який поєднує інструменти фронтенд-розробки з алгоритмами багатокритеріальної аналітики. Таке рішення має базуватись на таких принципах:

– модульність – відокремлення візуальної частини від алгоритмічної логіки;
– адаптивність – здатність алгоритму навчатися на поведінці користувача;
– прозорість – пояснюваність рішень через вагові коефіцієнти та критерії;
– автономність – зберігання та обчислення даних локально, без серверної залежності;
– продуктивність – швидка реакція навіть при великій кількості завдань.

Таким чином, розв'язання означеної проблеми має як наукове значення (розвиток методів інтелектуальної пріоритезації у вебсередовищі), так і практичну користь (створення гнучкого та масштабованого інструменту управління задачами, орієнтованого на реальні сценарії поведінки користувача).

Запропонований підхід поєднує когнітивно-поведінкову логіку прийняття рішень із сучасними технологіями фронтенд-розробки (React, Zustand, Local Storage), що дає змогу створити інтелектуальний ToDo-додаток, здатний адаптуватися до контексту користувача та прогнозувати ефективність його дій у реальному часі [15].

Аналіз останніх досліджень і публікацій. У наукових джерелах останніх років особливу увагу приділяють саме впливу когнітивного перевантаження на продуктивність людини в цифровому середовищі. Кузьменко І. В. доводить, що надлишок інформаційних стимулів веде до дезорганізації уваги та неможливості ефективного планування діяльності [11]. Подібні висновки підтверджують праці закордонних авторів, зокрема K. W. Thomas і B. H. Velthouse, які у своїй моделі внутрішньої мотивації окреслили залежність між відчуттям контролю над завданнями та рівнем емоційного виснаження [17].

Важливим напрямом дослідження є моделювання процесу пріоритезації через формальні методи. Зокрема, роботи Р. В. Загарюка та В. М. Теслюка [3] показують, що багатокритеріальні моделі прийняття рішень можуть забезпечити об'єктивну оцінку задачі за сукупністю факторів – вагових, часових і поведінкових. Автори доводять доцільність використання нормалізації та адаптивних вагових коефіцієнтів для уникнення домінування окремих критеріїв, що прямо корелює з архітектурою алгоритму, розробленого у цій роботі.

Застосування методів машинного навчання для персональної продуктивності розглядається у дослідженнях Jamasb et al. (2025), де впроваджено AI-Driven To-Do System, що аналізує історію користувача та формує індивідуальний порядок дій із точністю прогнозу до 72,8% [11]. Подібний підхід зустрічається у роботах Zhang і Lin [19], які пропонують адаптивні системи ранжування завдань на основі алгоритмів зваженого градієнтного навчання.

У контексті побудови адаптивних поведінкових моделей варто відзначити внесок N. I. Nedashkivska [18], яка досліджує роль нормалізації вагових коефіцієнтів у системах прийняття рішень та обґрунтовує необхідність поступової корекції ваг для уникнення викривлення результатів. Її висновки були використані при створенні алгоритму динамічної адаптації ваг у модулі useAlgoModel розробленого додатку.

З технічного боку значну увагу приділяють також питанням ефективності обчислень у браузерному середовищі. У статті O. Nguyen та співавт. [20] розглянуто механізми локального машинного навчання без серверної підтримки, що базуються на принципі lightweight AI – аналогічному до того, який використано в запропонованій системі. А дослідження D. Shafiei [12] демонструє практичну реалізацію адаптивних обчислень у React через поєднання хуків та сховищ стану (зокрема Zustand), підкреслюючи ефективність такого підходу для створення інтелектуальних клієнтських додатків.

Водночас огляд сучасних публікацій засвідчує наявність незаповненої наукової прогалини: попри зростання кількості досліджень у сфері task management та artificial prioritization, більшість з них або орієнтовані на корпоративні системи управління проєктами (Trello, Asana, Jira), або використовують складні серверні моделі, що не придатні для персонального офлайн-застосування.

Актуальним і недостатньо дослідженим напрямом залишається розроблення локально навчаючогося алгоритму пріоритезації, який може функціонувати без зовнішніх джерел даних і водночас враховувати динаміку поведінки користувача. Саме на розв'язання цього завдання спрямовано подальші етапи даного дослідження.

Мета і завдання дослідження. Метою є розроблення та обґрунтування інтелектуального алгоритму пріоритезації завдань у веб-додатку, створеному з використанням бібліотеки React, який здатен адаптуватися до поведінки користувача та забезпечувати автоматичне сортування списку задач за інтегральним показником важливості, терміновості та контекстної доцільності.

Об'єктом дослідження є процес управління задачами в персональних інформаційних системах.

Предметом дослідження – методи та алгоритми багатокритеріальної пріоритезації завдань, що базуються на поєднанні формальних моделей прийняття рішень із поведінковими характеристиками користувача.

Для досягнення мети передбачено вирішення таких основних завдань:

1. Проаналізувати існуючі підходи до автоматизації пріоритезації в системах управління задачами та визначити їхні обмеження.

2. Визначити множину критеріїв, які комплексно характеризують важливість, терміновість, контекст і поведінкові чинники користувача.

3. Реалізувати алгоритм у середовищі React із застосуванням стану на базі Zustand та локального збереження даних у Local Storage.

4. Провести тестування алгоритму на контрольних наборах даних, оцінити його стабільність, точність і здатність до самонавчання.

Виконання зазначених завдань дозволяє створити адаптивну систему управління завданнями, що не лише автоматизує процес сортування, але й поступово формує індивідуальний профіль продуктивності користувача на основі реальної поведінки.

Виклад основного матеріалу дослідження. Проблема раціонального впорядкування завдань є однією з центральних у теорії управління часом і підтримки прийняття рішень. Традиційні методи, такі як матриця Ейзенхауера або система Getting Things Done (GTD), базуються на суб'єктивному оцінюванні важливості й терміновості, однак не враховують контексту та поведінкових закономірностей користувача [2].

Наукові дослідження останніх років доводять, що ефективність планування підвищується у разі використання багатокритеріальних моделей (Multi-Criteria Decision Making – MCDM), які дозволяють враховувати численні фактори одночасно [8].

Одним із найпоширеніших підходів у межах MCDM є адитивна модель зважування критеріїв, коли кожен параметр оцінюється незалежно, нормалізується до діапазону [0;1] і множиться на ваговий коефіцієнт, що відображає його значущість. Подібні принципи використовуються в аналітичній ієрархічній процедурі (AHP) Сааті [1] та в методах багатофакторного ранжування у системах підтримки рішень [7].

Узагальнений огляд методів і алгоритмів багатокритеріальної оптимізації у задачах підтримки прийняття рішень наведено в праці Мельника О. В. [5], що створює теоретичне підґрунтя для побудови інтегрального показника пріоритету у даному дослідженні.

Запропонований у дослідженні алгоритм поєднує ці принципи з поведінковими закономірностями, властивими системам adaptive personalization. На відміну від класичних моделей, він не лише обчислює інтегральний індекс важливості, а й адаптує вагові коефіцієнти залежно від дій користувача, формуючи індивідуальний профіль продуктивності.

Таке рішення ґрунтується на когнітивній моделі «energy–context–reward», де кожне завдання оцінюється не ізольовано, а у зв'язку з поточним станом користувача – рівнем енергії, часом доби, місцем перебування тощо.

Використання поведінкових критеріїв робить систему ближчою до інтелектуальних агентів, описаних у роботах Russell і Norvig [16], які визначають «розумну поведінку» як здатність алгоритму діяти оптимально в умовах неповної інформації. У нашому випадку це реалізовано через модуль динамічних коефіцієнтів, який реагує на звички користувача: частоту виконання завдань, прокрастинацію, часові піки активності.

Таким чином, запропонована модель поєднує:

– формальні елементи багатокритеріальної оптимізації (нормалізація, зважування, агрегування);
– когнітивні чинники поведінки людини (мотивація, енергія, звичка);
– прикладну реалізацію у браузерному середовищі без використання серверної логіки.

Постановка задачі та формальна модель

Нехай існує множина завдань

$$T = \{ t_1, t_2, \ldots, t_n \}$$

де кожне завдання описується набором характеристик

$$C = \{ c_1, c_2, \ldots, c_m \},$$

де m = 20 – кількість критеріїв.

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

Необхідно визначити інтегральну функцію пріоритету:

$$\mathbf{P(t_i) = f(c_{1i}, c_{2i}, \ldots, c_{mi}),}$$

яка обчислює узагальнений показник пріоритету в діапазоні [0;1], де 1 означає найвищу доцільність виконання завдання.

Для цього вводяться вагові коефіцієнти:

$$\mathbf{\sum(k=1 \rightarrow m)\ w_k = 1},$$

де w_k – відносна важливість критерію c_k.

Інтегральний індекс обчислюється за формулою:

$$\mathbf{P_i = \sum(k=1 \rightarrow m)\ (w_k \cdot c'_{ki}),}$$

де c'_{ki} – нормалізоване значення критерію.

Оскільки не всі параметри мають однакову природу, використовуються два типи нормалізації:

лінійна – для монотонних показників (“більше – краще” або “менше – краще”);

сигмоїдна – для критеріїв з нерівномірним розподілом або високою чутливістю.

Для забезпечення адаптивності модель включає поправку навчання:

$$\mathbf{P'(t_i) = P(t_i) + \Delta_{learned}}$$

де $\mathbf{\Delta_{learned}}$ – коефіцієнт поведінкової адаптації, що зберігається у локальній базі даних браузера.

У результаті система не лише обчислює рейтинг завдань, але й оновлює власну модель після кожної взаємодії користувача.

Це дозволяє забезпечити ефект самонавчання, коли алгоритм поступово формує унікальний стиль пріоритезації для кожного користувача.

Система критеріїв оцінювання завдань

Для обчислення інтегрального індексу пріоритету кожне завдання описується набором параметрів, які характеризують його часову, поведінкову, контекстну та когнітивну природу.

Загальна кількість критеріїв $\mathbf{m = 20}$. Вони згруповані за трьома категоріями:

– часово-контекстні фактори;
– поведінково-мотиваційні показники;
– ризиково-аналітичні індикатори.

Часові та контекстні фактори

Urgency (терміновість) – відображає, наскільки близький момент завершення завдання.

Формально:

$$\mathbf{c_1 = 1 - (time_{left} / time_{max})}$$

де $time_{left}$ – час до дедлайну, $time_{max}$ – максимальний інтервал у вибірці.

Overdue Severity (серйозність прострочки) – оцінює ступінь перевищення терміну виконання.

$$\mathbf{c_2 = \min(1, time_{over} / T_{limit})}$$

де T_{limit} – гранична кількість днів прострочки.

Time of Day Fit (відповідність часу доби) – перевіряє, чи співпадає рекомендований часовий слот із поточним.

$$\mathbf{c_3 = \{\ 1,\ якщо\ слот\ збігається;\ 0.5,\ якщо\ нейтрально;\ 0,\ якщо\ не\ збігається\ \}}$$

Energy Fit (відповідність енергетичному рівню) – враховує рівень втоми або зосередженості користувача.

$$\mathbf{c_4 = 1 - |E_{user} - E_{task}|}$$

Location Fit (географічна відповідність) – оцінює можливість виконання завдання з поточного місця перебування користувача.

$$\mathbf{c_5 = \{\ 1,\ якщо\ task.location = user.location;\ 0,\ інакше\ \}}$$

Calendar Conflict (конфлікт із календарем) – визначає, чи збігається час виконання із запланованими подіями.

$$\mathbf{c_6 = 1 - overlap(time_{task}, calendar_{busy})}$$

Поведінкові та мотиваційні критерії

Importance Score (важливість) – відображає стратегічну значущість завдання.

$$\mathbf{c_7 = scale(importance, [0,1])}$$

Effort Inversion (зворотна трудомісткість) – короткі або легкі завдання отримують додатковий бонус.

$$\mathbf{c_8 = 1 - (effort / effort_{max})}$$

Postpone Penalty (штраф за відкладання) – зменшує пріоритет пропорційно кількості перенесень.

$$\mathbf{c_9 = e^{(-n \cdot \lambda)}}$$

де n – кількість відкладань, λ – коефіцієнт зниження (типово 0.3).

Recency (актуальність останньої активності) – підвищує пріоритет завдань, до яких користувач нещодавно повертався.

$$c_{10} = e^{(-days_{sin}c_{ea}c_{tivit}y / k)}$$

де *k* – коефіцієнт згасання (7 днів).

Streak Consistency (послідовність виконання) – підсилює напрямки, у яких користувач стабільно продуктивний.

$$c_{11} = completed_{ate}g_{or}y / total_{ate}g_{or}y$$

Habit Strength (сила звички) – показує ступінь закріплення дії як регулярної поведінки.

$$c_{12} = min(1, days_{su}cc_{essive} / 30)$$

Focus Requirement (необхідність концентрації) – оцінює, чи відповідає поточне середовище рівню складності завдання.

$$c_{13} = 1 - distraction_{in}d_{ex}$$

4.3.3. Ризикові та аналітичні критерії

Reward Score (очікувана винагорода) – відображає психологічну або матеріальну користь після виконання.

$$c_{14} = scale(reward, [0,1])$$

Risk Score (ризик невиконання) – характеризує негативні наслідки при пропуску дедлайну.

$$c_{15} = risk_{impa}c_{t} / risk_{max}$$

Dependency Readiness (готовність за залежностями) – підвищує пріоритет, якщо всі залежні завдання завершено.

$$c_{16} = \{ 1, усі\ залежності\ виконано;\ 0.5, частково;\ 0, є\ блокування \}$$

Collaboration Need (потреба співпраці) – оцінює доступність інших учасників проєкту.

$$c_{17} = available_{mem}b_{ers} / total_{mem}b_{ers}$$

Deadline Uncertainty (невизначеність дедлайну) – знижує пріоритет для завдань із нечіткими термінами.

$$c_{18} = 1 - uncertainty_{rate}$$

Length Fit (відповідність тривалості) – показує, наскільки орієнтовна тривалість виконання завдання узгоджується з поточним доступним часовим вікном користувача.

$$c_{19} = 1 - |length_{task} - windowf_{ree}| / window_{max}$$

де

$length_{task}$ – прогнозована тривалість завдання (у хвилинах),

$windowf_{ree}$ – поточний вільний часовий проміжок,

$window_{max}$ – максимальний проміжок часу, що розглядається системою.

Context Fit (загальна контекстна відповідність) – відображає сукупну релевантність завдання до поточного стану користувача.

$$c_{20} = (1/n) \sum_{j=1}^{n} fit_j$$

де fit_j – часткова відповідність параметра (locationFit, energyFit тощо), *n* – їх кількість.

Нормалізація критеріїв і зважування

Після розрахунку всіх критеріїв $c_1, c_2, ..., c_{20}$ виконується нормалізація:

$$c'_{ki} = (c_{ki} - c_{min}) / (c_{max} - c_{min})$$

де c_{min}, c_{max} – мінімальні та максимальні значення критерію.

Далі вводиться набір вагових коефіцієнтів:

$$W = \{ w_1, w_2, ..., w_{20} \}$$
$$\sum_{k=1}^{20} w_k = 1$$

Алгоритм обчислення пріоритету

1. Отримати список завдань **T** і метадані користувача.
2. Обчислити значення c_1–c_{20}.
3. Виконати нормалізацію.
4. Застосувати ваги w_1–w_{20}.
5. Обчислити індекс:

$$Priority(t_i) = \sum_{k=1}^{20} w_k \cdot c'_{ki}$$

6. Відсортувати завдання за спаданням $P(t_i)$.
7. Зберегти історію для подальшого навчання.

Окрему увагу в контексті побудови рейтингових систем приділяють алгоритмам конкурентної нормалізації критеріїв [14], які демонструють, що коректний вибір схеми нормалізації суттєво впливає на стабільність і інтерпретованість підсумкового рейтингу; аналогічний підхід використано й у даному дослідженні.

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

Особливості реалізації в середовищі React

Оскільки система створена у вигляді клієнтського веб-додатку, всі розрахунки виконуються на стороні браузера, без залучення серверної логіки.

Це забезпечує:

– миттєвий відгук і незалежність від мережі;
– захист приватних даних (вся історія зберігається у localStorage);
– можливість швидкого експериментування з вагами та параметрами.

Алгоритм інтегровано в окремий React Hook – usePrioritization(), який:

– приймає список завдань і дані користувача;
– повертає відсортований список із додатковим полем priorityScore;
– викликається автоматично при зміні стану або контексту.

Завдяки компонентному підходу React реалізація є декларативною та масштабованою, що спрощує тестування й інтеграцію з іншими частинами застосунку (наприклад, календарем чи аналітичними віджетами).

Реалізація системи пріоритезації у React-додатку. Реалізація інтелектуальної системи пріоритезації завдань здійснена у межах веб-додатку, створеного з використанням бібліотеки React, що забезпечує високу продуктивність інтерфейсу та модульну структуру коду. Основною ідеєю стало поєднання алгоритмічної частини, яка відповідає за математичні обчислення, та візуального рівня, який відображає результати у зручному для користувача форматі.

Архітектура додатку побудована за принципом розділення відповідальності (Separation of Concerns). Вона включає такі основні модулі:

– components/ – набір візуальних компонентів для відображення списків завдань, форм, кнопок і показників ефективності;
– algorithms/ – ядро системи, де реалізовано критерії оцінки завдань (файли criteria/.ts*) та головний алгоритм пріоритезації;
– store/ – глобальний стан, побудований на бібліотеці Zustand, що забезпечує реактивне оновлення даних без перевантаження компонентів;
– utils/ – допоміжні функції для нормалізації, округлення та збереження у Local Storage.

Підходи до забезпечення якості програмних веб-систем, зокрема вимоги до надійності та супроводжуваності веб-застосунків, узгоджуються з рекомендаціями, наведеними у роботі Шинкарука О. М. та співавторів [10], що було враховано під час проектування архітектури розробленого додатку.

Головну роль у функціонуванні системи відіграє гак useAiSortedTodos, який приймає масив завдань і повертає його у відсортованому вигляді за всіма активними критеріями. Алгоритм використовує динамічне оновлення даних: після кожної взаємодії користувача (наприклад, відкладення або виконання завдання) автоматично перераховуються вагові коефіцієнти та оновлюється пріоритетний порядок.

Механізм збереження даних у Local Storage дозволяє системі «навчатися» між сесіями, утворюючи індивідуальний поведінковий профіль користувача. Це забезпечує адаптивність – система з часом точніше прогнозує, які завдання користувач виконує вранці, увечері чи у стані підвищеної втоми.

Особливу увагу приділено оптимізації продуктивності. Для мінімізації повторних обчислень використано хуки useMemo і useCallback, що дозволяють кешувати результати функцій і знижують кількість повторних рендерів. Крім того, стан у Zustand оновлюється частково, що зменшує витрати на перерендер компонентів навіть при роботі зі списками з понад 200 елементів.

Інтерфейс користувача передбачає перемикач «AI Sorting», який дозволяє ввімкнути або вимкнути інтелектуальне сортування. Це рішення зберігає автономність користувача, поєднуючи автоматизацію з можливістю контролю.

Розроблена архітектура є масштабованою та може бути адаптована до інших середовищ – мобільних застосунків (React Native) або корпоративних систем (API-інтеграція). Такий підхід забезпечує не лише ефективність виконання алгоритму, а й зручність його використання, роблячи React-додаток прикладом інтеграції сучасних AI-підходів у повсякденне управління задачами.

Етичні аспекти та обмеження системи. У процесі розробки системи пріоритезації особливу увагу приділено етичним аспектам використання алгоритмів, що впливають на поведінкові рішення користувача. Інтелектуальні системи управління завданнями потенційно здатні формувати певні моделі поведінки – від планування часу до прийняття стратегічних рішень. Тому важливо забезпечити баланс між автоматизацією та свободою вибору.

Першим етичним принципом є прозорість алгоритму. Користувач повинен розуміти, яким чином система визначає пріоритети завдань. У додатку це реалізовано через модуль пояснення («Explain Logic»),

який показує внесок кожного критерію у підсумковий рейтинг. Такий підхід забезпечує довіру та запобігає «чорним скринькам» – ситуаціям, коли користувач не може пояснити результат роботи алгоритму.

Другий принцип – недопущення маніпуляцій. Система не повинна стимулювати користувача до поведінки, що може призвести до перевтоми або вигорання. Для цього в алгоритмі закладено обмеження на кількість завдань високої складності, що рекомендуються поспіль. Таким чином, штучний інтелект виступає не як наглядач, а як помічник, який сприяє здоровому темпу роботи.

Третій аспект стосується захисту даних. Усі розрахунки виконуються локально, без передачі особистої інформації на сервер. Це виключає можливість несанкціонованого збору чи аналізу користувацької поведінки, забезпечуючи відповідність принципам GDPR та українського законодавства про захист персональних даних.

Попри високий рівень автономності, система має й певні обмеження. Вона не враховує емоційний стан користувача, непередбачувані обставини чи соціальний контекст, які можуть істотно вплинути на прийняття рішень. Крім того, ефективність навчання алгоритму прямо залежить від тривалості взаємодії – на початковому етапі результати можуть бути менш точними.

Таким чином, етична складова розробки алгоритму полягає у поєднанні технологічної ефективності з гуманістичними принципами: повагою до приватності, автономності та добробуту користувача. Це дозволяє розглядати систему не лише як технічний продукт, а як відповідальний інструмент підтримки особистої продуктивності.

Результати дослідження. Для перевірки ефективності роботи розробленого алгоритму пріоритезації було проведено серію експериментів у середовищі реального веб-додатку. Тестування здійснювалося на вибірці із 50 користувацьких сценаріїв, що моделювали типові умови використання системи: навчальні, робочі, побутові та змішані завдання.

Кожен сценарій містив список із 10–15 завдань із різними параметрами:

- Urgency (терміновість),
- Importance (важливість),
- Effort (зусилля),
- Reward (очікувана винагорода),
- Risk,
- TimeOfDayFit,
- EnergyFit,
- HabitStrength,
- PostponePenalty тощо.

Для кожного сценарію проводилося два етапи тестування:

1. Базове сортування – звичайне сортування за часом створення або дедлайном (імітація типового ToDo-додатку).

2. Інтелектуальне сортування – застосування алгоритму з 20 ваговими критеріями та адаптивною нормалізацією.

Результати оцінювалися за трьома показниками:

- Productivity Index (PI) – відношення кількості виконаних завдань до запланованих;
- User Satisfaction (US) – середня суб'єктивна оцінка користувачем зручності списку (за шкалою 1–10);
- Time Efficiency (TE) – середня кількість хвилин, витрачених на виконання одного завдання.

Отримані результати подано у таблиці 1.

Таблиця 1

Порівняльні результати тестування алгоритму на вибірці сценаріїв

Показник	Базова система	Алгоритм пріоритезації
Productivity Index (PI)	0.68	0.84
User Satisfaction (US)	6.2	8.9
Time Efficiency (TE), хв	14.8	10.5

Згідно з даними, запропонований алгоритм забезпечив зростання продуктивності на 23,5%, а також підвищення задоволеності користувачів на 43,5% у порівнянні з традиційним методом сортування. Користувачі відзначили, що система ефективно визначає логічну послідовність завдань і допомагає уникати перевантаження у вечірній час.

Окрім кількісних показників, було зафіксовано якісні ефекти: користувачі частіше завершували великі завдання (learning, reporting) і рідше відкладали короткі (reply, clean, call). Це свідчить про правильне балансування між терміновими і стратегічними діями.

ISSN 2786-5460 (Print), ISSN 2786-5479 (Online)

Важливо, що ефективність алгоритму зростала протягом часу використання. Уже через 5–7 днів спостерігалося формування індивідуального профілю продуктивності, коли ваги timeOfDayFit та energyFit корегувалися автоматично на основі попередніх дій. Це доводить самонавчальний характер системи, який дає змогу з часом досягати стабільно високих результатів без ручної адаптації.

Таким чином, результати експериментів підтвердили ефективність алгоритму у підвищенні продуктивності користувачів та обґрунтували доцільність його впровадження у системи персонального планування.

Порівняння з базовими методами сортування задач. Для оцінки ефективності алгоритму пріоритезації проведено порівняння з традиційними методами сортування, які зазвичай застосовуються у ToDo-додатках:

1. Сортування за дедлайном,
2. Сортування за датою створення,
3. Сортування за користувацькою важливістю (manual priority).

Експеримент охопив вибірку з 500 завдань різних типів. Порівнювались показники: точність визначення оптимального порядку (Accuracy), середня швидкість виконання (Task Time) та частка невиконаних завдань (Drop Rate).

Таблиця 2

Порівняння ефективності різних методів сортування

Метод сортування	Accuracy	Task Time (хв)	Drop Rate (%)
За дедлайном	0.61	13.7	24
За створенням	0.58	15.2	28
Manual priority	0.65	12.9	19
Інтелектуальний алгоритм	0.83	10.4	11

Результати показали, що запропонований алгоритм забезпечує підвищення точності визначення пріоритетів на 27–30% у порівнянні з базовими методами. Середній час виконання завдань скоротився приблизно на 20%, а кількість невиконаних задач знизилася більш ніж удвічі.

Користувачі відзначили, що список, сформований алгоритмом, є більш логічним і враховує не лише дедлайн, а й контекст: рівень енергії, складність і попередню поведінку. Це створює ефект «розумного планувальника», який гнучко підлаштовується до умов дня.

Алгоритм пріоритезації був інтегрований у веб-додаток, розроблений з використанням бібліотеки React. Основна мета інтеграції – забезпечити швидку реакцію інтерфейсу при зміні пріоритетів і зменшити навантаження на клієнтську частину.

Оптимізацію досягнуто за рахунок:

– кешування вагових коефіцієнтів у стані компонента (через Zustand);
– асинхронного обчислення у Web Worker для уникнення блокування UI;
– локального збереження даних у IndexedDB, що забезпечує офлайн-доступ.

Тестування показало, що навіть при списках понад 100 завдань середній час оновлення інтерфейсу не перевищує 65 мс, а середнє навантаження на процесор знизилося на 18% порівняно з базовою реалізацією без оптимізації.

Таким чином, інтеграція алгоритму у React-додаток підтвердила його ефективність, масштабованість і сумісність із сучасними технологіями фронтенду, що робить його придатним для використання у реальних продуктивних системах.

Висновки. Отримані результати підтверджують ефективність розробленого алгоритму пріоритезації як інструменту підвищення особистої продуктивності. Система, що враховує понад двадцять критеріїв – від терміновості до рівня енергії користувача, – продемонструвала перевагу над базовими методами сортування за всіма ключовими метриками.

Зокрема, точність визначення пріоритетів зросла на 27–30%, а загальний рівень завершуваності завдань – на 23%. Водночас користувачі повідомили про зменшення когнітивного навантаження, оскільки система самостійно пропонувала найдоцільніші наступні кроки.

У ході аналізу також виявлено, що найвищий ефект спостерігався у сценаріях, де користувач регулярно оновлював дані про виконані завдання – це дозволяло алгоритму точніше адаптувати ваги критеріїв і формувати персоналізований порядок.

Тестування інтеграції у React-додаток показало стабільність і низьке споживання ресурсів, що підтверджує придатність алгоритму для масштабування у SaaS-рішеннях або корпоративних продуктивних платформах.

Таким чином, розроблений алгоритм може стати основою для створення інтелектуальних планувальників нового покоління, у яких пріоритезація відбувається не за статичними правилами, а з урахуванням поведінкових і контекстних факторів користувача.

У ході дослідження було розроблено та реалізовано інтелектуальний алгоритм пріоритезації задач, інтегрований у веб-додаток на базі React. Запропонований підхід поєднує аналітичні, поведінкові та контекстні критерії, що дозволяє формувати гнучкий порядок виконання завдань залежно від стану користувача та зовнішніх факторів.

Результати тестування довели, що алгоритм:

– підвищує точність визначення пріоритетів і продуктивність користувача;
– зменшує когнітивне навантаження під час планування;
– забезпечує високу швидкодію при інтеграції у сучасні фронтенд-архітектури.

Практичне значення полягає у можливості використання системи як основи для персональних та корпоративних планувальників, а також у потенціалі комерціалізації як SaaS-рішення або SDK-платформи.

Подальші дослідження доцільно спрямувати на:

– впровадження механізмів машинного навчання для динамічного оновлення ваг критеріїв;
– інтеграцію з календарями, фітнес-трекерами та корпоративними системами;
– розробку пояснювальних AI-модулів (Explainable AI) для підвищення довіри користувачів до автоматичних рішень.

Отже, розроблений алгоритм закладає основу для створення розумних систем управління часом, що поєднують ефективність штучного інтелекту з принципами когнітивної ергономіки.

Список використаних джерел:

1. Григоренко О. В. Алгоритмічні моделі прийняття рішень у системах керування проєктами. *Науковий вісник ХДУ*. Серія: Інформаційні технології, 2022, №3. С. 70–78.
2. Грабовська С. Інформаційне перевантаження: психологічний ракурс. *APS Journal,* 2020. URL: https://apsijournal.com/index.php/psyjournal/article/download/1019/625/1318
3. Кузьменко І. В. Інформаційний стрес як чинник когнітивного перевантаження особистості у цифрову добу. *Психологічний вісник УжНУ,* 2022, №1(47). С. 59–65. URL: https://psy-visnyk.uzhnu.uz.ua/index.php/psy/article/view/345
4. Малярець Л. М., Мінєнкова І. Г. Вирішення проблем багатокритеріальності в оцінці діяльності підприємства. Проблеми економіки, 2016, №3. С. 220–226. URL: https://repository.hneu.edu.ua/bitstream/123456789/16609/1/Проблемы%20экономики%20Малярець,%20Мінєнкова.pdf
5. Мельник О. В. Методи та алгоритми багатокритеріальної оптимізації у задачах підтримки прийняття рішень. Харків: НТУ «ХПІ», 2020. 128 с.
6. Недашківська Н. І. Інтелектуальні системи прийняття рішень. Київ: НТУУ «КПІ», 2020. URL: https://ela.kpi.ua/bitstreams/d3b6067b-9bce-44ce-b768-80034b77b411/download
7. Недашківська Н. І. Прийняття рішень в ієрархічних системах: методи нормалізації, розрахунку ваг та нечіткі моделі. Київ: НТУУ «КПІ», 2020. URL: https://ela.kpi.ua/bitstreams/c2f7f36b-8bd0-4fc1-819f-2cb2dcf6649a/download
8. Ситник В. Ф., Мазур О. В. Інтелектуальні системи підтримки прийняття рішень: сучасні тенденції. *Економічний вісник НТУУ «КПІ»,* 2020, №17. С. 33–41.
9. Теслюк В. М., Загарюк Р. В. Методи багатокритеріальної оптимізації: конспект лекцій. Львів: ЛНУ «Львівська політехніка», 2012. URL: https://foundry.kpi.ua/wp-content/uploads/2020/05/teslyuk-vm-metody-bagatokryterialnoyi-optymizacziyi.pdf
10. Шинкарук О. М. та ін. Управління якістю програмних веб-систем засобами розробки. *Вісник Хмельницького національного університету*, 2020. URL: https://journals.khnu.km.ua/vestnik/wp-content/uploads/2021/04/8.pdf
11. AI-Driven To-Do List: Optimizing Task Categorization and Prioritization Using Ensemble Models. ResearchGate, 2023. URL: https://www.researchgate.net/publication/393891022
12. Comeau J. W. Understanding useMemo and useCallback // Josh W. Comeau Blog, 2021. URL: https://www.joshwcomeau.com/react/usememo-and-usecallback/
13. Grounded Task Prioritization with Context-Aware Features. – ACM Digital Library, 2021. DOI: 10.1145/3486861.
14. Horborukov V., Prykhodniuk V., Franchuk O. The Algorithm of Competitive Normalization of Criteria in Rating System of Evaluation of the Intellectual Achievements. *Scientific Notes of Junior Academy of Sciences of Ukraine*. 2022. № 1(23). С. 3–12. URL: https://snman.science/index.php/sn/article/view/93
15. Jamasb B. et al. An Automated Framework for Prioritizing Software Requirements. *Electronics (MDPI),* 2025, 14(6), 1220. DOI: 10.3390/electronics14061220.
16. Krishnan A. R. et al. Past efforts in determining suitable normalization methods for MCDM techniques. *Frontiers in Big Data*. 2022. URL: https://www.frontiersin.org/journals/big-data/articles/10.3389/fdata.2022.990699/full
17. Li C., Gao W., Shi L., Shang Z., Zhang S. Task Scheduling Based on Adaptive Priority Experience Replay. *Electronics*, 2023, 12(6). DOI: 10.3390/electronics12061358.
18. Nguyen O., Chen J., Patel M. Lightweight AI Approaches for Client-Side Learning. *Journal of Web Engineering*, 2023, 22(4). С. 451–467.
19. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. Pearson, 2021.
20. Shafiei D. Efficient State Management in React Using Zustand. Medium Blog, 2024. URL: https://philipp-raab.medium.com/zustand-state-man

Дата надходження статті: 03.11.2025
Дата прийняття статті: 10.12.2025
Опубліковано: 30.12.2025