КІБЕРНЕТИКА та КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 004.75:519.8 DOI:10.34229/2707-451X.25.2.3

В.В. ГОДЛЮК

МАТЕМАТИЧНІ МОДЕЛІ ДЛЯ ІНФОРМАЦІЙНИХ СИСТЕМ УПРАВЛІННЯ НА ЦИФРОВИХ ПЛАТФОРМАХ: ВІД УПРАВЛІННЯ РЕСУРСАМИ ДО ПРОГНОЗУВАННЯ ПОПИТУ

Розглянуто математичні моделі та методи, що використовуються в інформаційних системах управління цифрових платформ. Досліджено алгоритми оптимізації ресурсів, балансування навантаження, прогнозування попиту та побудови рекомендаційних систем. Описано застосування теорії графів, зокрема, алгоритмів Дейкстри та Флойда – Уоршелла, для маршрутизації даних. Аналізуються методи лінійного та цілочисельного програмування, градієнтного спуску, генетичних алгоритмів і дискретного моделювання для підвищення ефективності платформ. Запропоновані підходи сприяють адаптації цифрових платформ до змінних умов та забезпеченню високої продуктивності.

Ключові слова: *цифрові платформи, математичні моделі, управління ресурсами, балансування навантаження, прогнозування попиту, алгоритми Дейкстри та Флойда – Уоршелла, генетичні алгоритми, дискретне моделювання.*

Вступ. Сучасний світ мобільних та онлайн технологій корегує наше бачення доступу до інформації та її використання у нашому повсякденні. Електронна комерція, стрімінгові сервіси, хмарні обчислення та соціальні мережі в своїй роботі використовують цифрові платформи, які стали невід'ємно частиною отримання якісного та швидкого доступу до даних. Успіх роботи онлайн сервісів залежить від здатності ефективно управляти ресурсами, обробляти великі обсяги даних та адаптуватись до змінних умов у режимі реального часу.

Цифрові платформи стикнулися з проблемою пошуку балансу між складністю обчислювальних завдань, швидкістю обробки даних і ресурсними обмеженнями. Інтелектуальні системи, такі як штучний інтелект, машинне навчання, великі обчислювальні мережі, є основним драйвером розвитку цифрових платформ, але із їх удосконаленням виникли складнощі, що обмежують їх ефективність.

В зв'язку із збільшення потоків та об'ємів інформації, уповільнюється швидкість обробки інтелектуальних систем, що може спричинити затримки в роботі штучного інтелекту та машинного навчання, що в свою чергу знизить продуктивність та точність результатів. В сучасних умовах цифрові платформи повинні вміти розподіляти навантаження між різними компонентами системи, щоб уникнути перевантажень або недовантажень окремих елементів (процесорна потужність, оперативна пам'ять, сховища даних), бути надійними та стійкими до збоїв, добре захищеними в аспекті кібербезпеки та мати можливість до масштабування (можливість збільшення потужності системи без значного зниження продуктивності).

У цій статті розглядаються математичні моделі та методи, що використовуються для створення інформаційних систем управління цифровими платформами. Особливу увагу приділено їхньому застосуванню для вирішення задач управління ресурсами, балансування навантаження, прогнозування попиту, надання рекомендацій.

Опис моделей. Математичні моделі є важливим інструментом та невід'ємною частиною інформаційних систем управління цифрових платформ. Їх використання сприяє підвищенню ефективності, продуктивності, оптимізації ресурсів та навантажень. Моделі лінійного програмування дозволяють оптимізувати використання та розподіл ресурсів, забезпечуючи баланс між обчислювальними потужностями та навантаженням на сервери. Диференційні рівняння застосовуються для регулювання поведінки системи під час пікових навантажень. Алгоритми пошуку оптимальних рішень забезпечують підтримку процесів у режимі реального часу, що сприяє швидкій адаптації до змін у попиті. Стохастичні моделі використовуються для підтримки стабільності під час масштабування, прогнозуючи оптимальне використання ресурсів у разі зростання навантажень.

Теорія графів дозволяє відображати об'єкти (вузли) та їх взаємозв'язки (ребра) [1]. У цьому контексті графи можуть представляти користувачів та їхні взаємодії (кожен користувач може бути вузлом, а взаємодії між ними (наприклад, комунікації, обмін даними – ребрами), продукти та послуги (вузли можуть представляти різні продукти, а ребра – зв'язки між ними, такі як спільні категорії або сумісність).

На цифрових платформах, як соціальні мережі або системи доставки контенту, для оптимізації маршрутизації даних може використовуватися алгоритм Дейкстри [2], за допомогою якого є можливість знайти найкоротші шляхи в мережі.

Наприклад, у системі доставки контенту кожен вузол u може представляти сервер з якого надходять дані, v – сервер, до якого ці дані передаються, а вага $w(u, v)$ – час передачі даних між серверами. Мета – знайти шлях від початкового сервера s (точка відправлення даних у мережі, звідки здійснюється пошук оптимального маршруту) до кінцевого вузла, де знаходиться користувач, за найкоротший час. Використовуючи формулу

$$d(v) = \min\big(d(v), d(u) + w(u, v)\big), \tag{1}$$

де $d(u)$ – найкоротша відстань від початкового сервера s до вузла u, $d(u) + w(u, v)$ – потенційний новий шлях до вузла v, який проходить через вузол u.

На цифровій платформі є можливість динамічно обирати оптимальний шлях для кожного користувача, гарантуючи швидкий доступ до контенту, це важливо під час пікових навантажень, коли платформа повинна розподіляти трафік, щоб уникнути затримок і збоїв.

На цифрових платформах, які виконують функцію маркетплейсів або стрімінгових сервісів існує складна мережа зв'язків між користувачами, продуктами та контентом, алгоритм Флойда – Уоршелла [3] може допомогти у побудові рекомендаційної системи.

Припустимо, що в графі кожен вузол i представляє користувача або товар, а вага ребра $w(i, j)$ – схожість між користувачами чи продуктами (для користувачів це може бути кількість спільних інтересів, а для товарів – спільні категорії чи відгуки). Алгоритм Флойда – Уоршелла обчислює найкоротші відстані між усіма парами вузлів за допомогою формули:

$$D_{ij}^{(k)} = \min\left(D_{ij}^{(k-1)}, D_{ik}^{(k-1)} + D_{kj}^{(k-1)}\right), \tag{2}$$

де $D_{ij}^{(k)}$ – мінімальна відстань між вузлами i і j з урахуванням проміжних вузлів від 1 до k.

За допомогою цього підходу на платформі можна створити рекомендаційні списки для користувачів, враховуючи всі можливі зв'язки. Наприклад, якщо користувач переглядає певний продукт, алгоритм дозволить запропонувати схожі товари, навіть якщо між ними немає прямого зв'язку.

Для сегментації користувачів, в аспекті налаштування цільової реклами [4] та персоналізованого контенту, може використовуватись метод кластеризації, наприклад, k-середніх [5].

На платформі величина $X = \{x_1, x_2, \dots, x_n\}$ може представляти сукупність користувачів, а відстань $||x - \mu_j||$ – схожість між користувачами на основі їх поведінкових даних (наприклад, кількість переглядів або вподобань). Мета алгоритму – мінімізувати функцію вартості

$$J = \sum_{j=1}^{k} \sum_{x \in C_j} ||\, x - \mu_j||^2, \tag{3}$$

де k – загальна кількість кластерів, C_j – множина користувачів у кластері j, μ_j – центроїд кластера C_j, який є середнім значенням позицій усіх елементів у кластері, $||x - \mu_j||^2$ – квадрат відстані між користувачем x і центроїдом μ_j.

Це дає можливість створення на платформі сегментів користувачів і налаштування рекомендацій відповідно до їх інтересів. Завдяки цьому підходу користувачі отримують релевантні рекомендації, а на платформі підвищується залучення та утримання клієнтів.

Мережеві моделі є критично важливими для управління даними та забезпечення безперервного функціонування цифрової платформи, в свою чергу алгоритм максимального потоку [6] може допомогти визначити оптимальне навантаження f, яке може пройти через мережу платформи, що особливо корисно для підтримки високої доступності. Формально, задача максимального потоку може бути описана як:

$$max f(u, v) \text{ для всіх } (u, v) \in E, \tag{4}$$

де E – множина ребер (зв'язків) у мережі.

Кожне ребро $(u, v) \in E$ представляє шлях, по якому дані можуть передаватися з одного вузла u до іншого v, у мережі платформи ці ребра можуть відповідати з'єднанням між серверами або іншими елементами інфраструктури.

$f(u, v)$ – кількість даних або "потік", який проходить через ребро (u, v). Це значення показує, скільки інформації передається між вузлами u і v за певний проміжок часу. Потік $f(u, v)$ має обмеження, що залежить від пропускної здатності кожного ребра: для кожного ребра (u, v) має виконуватися умова $0 \leq f(u, v) \leq c(u, v)$ де $c(u, v)$ – пропускна здатність ребра.

Алгоритм максимального потоку під час передачі великих обсягів даних між серверами цифрової платформи забезпечує роботу мережі з максимальною продуктивністю без перевантаження, що важливо для безперебійної роботи сервісів.

За допомогою алгоритму мінімального розрізу [7], можна мінімізувати затримки в передачі даних, ефективно розподіляючи навантаження між різними мережевими шляхами. Ця оптимізація є критично важливою для балансування навантажень на цифрових платформах, таких як наприклад AmazonWebServices [8], де для розподілу ресурсів використовуються оптимізаційні методи. Формально, задача мінімального розрізу може бути представлена як:

$$min \sum_{(u,v) \in S} c(u, v), \tag{5}$$

де S – розріз, що розділяє вершини графа платформи, іншими словами представляє множину ребер, яка розділяє граф на дві частини: одну, яка містить початковий вузол і другу, яка містить кінцевий вузол, $c\ (u, v)$ – вартість (або пропускна здатність) ребра між вершинами u і v.

Цей підхід дозволяє забезпечити оптимальне використання ресурсів, що, в свою чергу, підвищує ефективність роботи платформи та задовольняє потреби користувачів, особливо при пікових навантаженнях для забезпечення плавності трафіку.

Лінійне програмування – це основний інструмент для оптимізації використання ресурсів у системах, де треба збалансувати навантаження на сервери, якщо цю задачу вдається формалізувати як задачу лінійного програмування [9].

Наприклад, на платформі з кількома серверами можна мінімізувати сумарний час обробки запитів користувачів. Нехай, x_i – кількість запитів, призначених серверу i, показує скільки запитів буде

оброблено кожним сервером. Значення x_i для кожного сервера має бути невід'ємним, оскільки кількість запитів не може бути від'ємною, що відображено в обмеженні $x_i \geq 0$, c_i – час обробки одного запиту на сервері i. Задача мінімізації сумарного часу обробки виглядає так:

$$\begin{aligned} &min \sum_{i=1}^{n} c_i x_i, \\ &\sum_{i=1}^{n} x_i = T, \end{aligned} \tag{6}$$

де T – загальна кількість запитів, які потрібно обробити $x_i \geq 0$ для всіх i.

Цільова функція обчислює загальний час обробки всіх запитів, враховуючи продуктивність кожного сервера. Мета – мінімізація цього часу для зниження затримок і підвищення ефективності роботи платформи.

Цілочисельне програмування використовується в ситуаціях, коли змінні повинні приймати тільки цілочисельні значення [10]. Наприклад, платформа може мати завдання мінімізувати кількість необхідних серверів для обробки певної кількості запитів, де кількість серверів це ціле число.

Припустимо, що кожен сервер може обробити q запитів, y – кількість серверів, яку ми хочемо визначити. Тоді задача мінімізації кількості серверів виглядає так:

$$\begin{aligned} &min\, y, \\ &y \cdot q \geq T, \end{aligned}$$

де q – максимальна кількість запитів, яку може обробити один сервер; T – загальна кількість запитів, які потрібно обробити всіма серверами, або загальне навантаження, яке платформа отримує від користувачів і його потрібно розподілити.

Дане обмеження гарантує, що загальна обчислювальна здатність усіх серверів (виражена як $y \cdot q$) буде достатньою для обробки T запитів.

Динамічне програмування – це метод оптимізації, який розбиває складну задачу на менші підзадачі [11]. Воно ефективне для задач, де рішення для кожного кроку залежить від рішень попередніх кроків.

Розглянемо випадок оптимізації часу відновлення даних з кешу. Нехай $f(i)$ – мінімальний час доступу до даних для перших i запитів, маємо два варіанти; або зберігати дані у кеші, або завантажувати їх заново; рекурентне співвідношення для функції $f(i)$, можемо записати як:

$$f(i) = min\big(f(i-1) + cost\_load, cost\_cache(i)\big), \tag{7}$$

де *cost_load* – час завантаження даних без кешу, *cost_cache(i)* – час доступу до даних з кешу для i-го запиту.

Це співвідношення означає, що для кожного запиту i ми маємо два варіанти: a) завантажити дані заново (що додає час завантаження *cost_load* до загального часу $f(i-1)$, b) використати кешовані дані з вартістю *cost_cache(i)*. Алгоритм обирає мінімальне з двох значень, забезпечуючи оптимальний час доступу до даних для поточного запиту.

Це рекурентне співвідношення дозволяє нам вибрати оптимальну стратегію кешування для кожного запиту, зводячи до мінімуму загальний час обробки запитів, що підвищує швидкість платформи та знижує затримки для користувачів.

Для оптимізації роботи цифрових платформ використовуються алгоритми на основі моделювання, а саме моделювання дискретних подій (Discrete Event Simulation) [12], що відображає систему як набір подій, які відбуваються у визначені моменти часу у відповідь на певні дії, та неперервне

моделювання (Continuous Simulation) [13], яке використовується для систем, де зміни відбуваються постійно, наприклад, під час потокової передачі даних або роботи у режимі реального часу.

Моделювання дискретних подій представляє систему як сукупність станів $S(t)$, де кожен стан змінюється через події, що виникають у певні моменти часу $t_1, t_2, \ldots, t_n$. Кожна подія E_i викликає зміну стану системи з $S(t_i^-)$до $S(t_i^+)$, де t_i^- і t_i^+ позначають стан системи безпосередньо до і після події відповідно.

Стан системи у момент часу t позначається як $S(t)$. Наприклад, у цифровій платформі стан $S(t)$може представляти кількість запитів користувачів, які обробляються сервером у цей момент. Якщо на платформі відбувається подія E_i (надходження нового запиту користувача), стан системи змінюється за правилом:

$$S(t_i^+) = S(t_i^-) + \Delta S_i, \tag{8}$$

де ΔS_i – зміна стану, яку викликає подія E_i.

Кількість подій, що надходять до системи, можна описати за допомогою інтенсивності потоку подій λ, яка показує середню кількість подій за одиницю часу (наприклад, інтенсивність запитів від користувачів на платформу):

$$N(t) = \lambda t, \tag{9}$$

де $N(t)$ – загальна кількість запитів, що надійшли за час t.

Для кожного запиту час очікування W та обробки T можна представити як:

$$W_i = t_i - t_{i-1}, \tag{10}$$

де W_i – час очікування для обробки події E_i, t_i, t_{i-1} – моменти часу для двох послідовних подій.

Загальне навантаження платформи $L(t)$ у час tможна оцінити як суму всіх змін стану, викликаних подіями до поточного моменту:

$$L(t) = \sum_{i=1}^{N(t)} \Delta S_i, \tag{11}$$

де $N(t)$ – загальна кількість подій, що сталися до моменту часу t.

Неперервне моделювання відображає систему як набір змінних, значення яких змінюються постійно протягом часу. Нехай $x(t)$ позначає поточний стан системи (наприклад, рівень завантаження платформи, кількість запитів, що обробляються або швидкість передачі даних) у момент часу t. Стан системи може змінюватися згідно з диференціальним рівнянням, що описує залежність стану від часу та інших факторів.

Постійні зміни стану системи можна представити як похідну функції стану $x(t)$ за часом:

$$\frac{dx(t)}{dt} = f(x(t), u(t), t), \tag{12}$$

де $x(t)$ – стан системи в момент часу t, $dx(t)$ – зміна стану системи $x(t)$ за короткий проміжок часу (у контексті неперервного моделювання це показує, як стан системи змінюється у кожен момент часу), $u(t)$ – вхідний сигнал або керуюча змінна, що впливає на систему (наприклад, швидкість надходження нових запитів), f – функція, що описує взаємодію між станом, керуючими змінними та часом.

Похідна $\frac{dx(t)}{dt}$ дозволяє моделювати, як швидко змінюється кількість підключень на платформу в реальному часі. Це допомагає прогнозувати навантаження і вживати заходів для уникнення перевантажень (наприклад, виділення додаткових ресурсів).

Цифрові платформи функціонують в умовах змінного навантаження протягом доби, тому важливим завданням є ефективний розподіл ресурсів і їхня оптимізація [14]. Нехай $R(t)$ – загальний

обсяг доступних ресурсів (обчислювальні потужності, пам'ять, пропускна здатність мережі та ін.) у момент часу t, а $D(t)$ – потреба в ресурсах у момент часу t. Оптимізація ресурсів означає, що у кожен момент часу виконується умова:

$$R(t) \geq D(t).$$

При цьому обчислюється оптимальна кількість ресурсів $R*(t)$, яка задовольняє попит $D(t)$ при мінімальних витратах:

$$\begin{gathered} R^*(t) = \arg\min_{R(t)} C\left(R(t)\right), \\ R(t) \geq D(t), \end{gathered} \tag{13}$$

де $C\big(R(t)\big)$ – функція вартості використання ресурсів.

Якщо навантаження змінюється з періодом T, його можна моделювати як функцію, періодичну в часі, де синусоїдальна функція моделює коливання навантаження з амплітудою A і періодом T:

$$D(t) = D_0 + A\sin\left(\frac{2\pi t}{T}\right), \tag{14}$$

де D_0 – середнє навантаження, яке система зазнає протягом періоду T, A – амплітуда змін у навантаженні, визначає максимально можливе відхилення навантаження від середнього значення D_0, чим більше значення A, тим значніші коливання навантаження, T – період коливань, протягом якого навантаження повертається до свого середнього значення D_0. Наприклад, якщо період T дорівнює 24 годинам, це може відображати добові коливання навантаження на платформі, коли активність користувачів змінюється залежно від часу доби.

Для стабільної роботи система повинна підтримувати стан $x(t)$ у межах допустимих значень. Це можна формалізувати як:

$$x_{\min} \leq x(t) \leq x_{\max}, \tag{15}$$

де x_{min} і x_{max} – межі, в яких система може працювати стабільно (можуть визначатися залежно від фізичних або економічних обмежень, наприклад, максимальної пропускної здатності мережі або обчислювальних ресурсів).

Цифрові платформи обробляють великі обсяги запитів і даних, їм необхідно постійно знаходити оптимальні налаштування для розподілу ресурсів, управління навантаженням і забезпечення максимальної продуктивності. Генетичні алгоритми допомагають вирішувати ці проблеми завдяки своїй здатності адаптуватися та знаходити оптимальні рішення в умовах обмежених ресурсів [15]. Сфокусуємося на тому, як ці алгоритми використовуються для оптимізації таких задач, як розподіл ресурсів, балансування навантаження та підвищення продуктивності платформ.

Цифрові платформи, такі як хмарні сервіси, повинні ефективно розподіляти обчислювальні ресурси між користувачами, щоб мінімізувати затримки та забезпечити максимальну продуктивність. Генетичні алгоритми використовуються для оптимізації розподілу обчислювальних потужностей $R(t)$, щоб досягти балансу між запитами користувачів $D(t)$ та доступними ресурсами. Нехай множина можливих рішень складається з наборів параметрів розподілу ресурсів, де кожен індивід x_i представляє конфігурацію розподілу ресурсів для певного набору серверів. Пристосованість $f(x)$ кожного індивіда вимірюється функцією, яка визначає якість розподілу ресурсів, наприклад, шляхом мінімізації затримки або зниження витрат на інфраструктуру:

$$f(x) = -\left(\sum_{t=1}^{T}\big(R(t) - D(t)\big)^2 + C(x)\right), \tag{16}$$

де $\sum_{t=1}^{T}\big(R(t) - D(t)\big)^2$ – частина формули вимірює відхилення між доступними ресурсами $R(t)$ та потребами $D(t)$ у кожен момент часу t, $C(x)$ – функція вартості використання ресурсів для конфігурації x.

Наприклад, на хмарній платформі, яка обслуговує величезну кількість запитів на обробку даних, генетичний алгоритм створює множину можливих конфігурацій розподілу ресурсів між серверами, де кожен індивід x_i представляє конкретне налаштування обчислювальних ресурсів для певного часу. За допомогою функції пристосованості [16], алгоритм відбирає найбільш ефективні конфігурації, які мінімізують відхилення між $R(t)$ і $D(t)$ та знижують загальні витрати на інфраструктуру. Найкращі конфігурації розподілу ресурсів передаються на наступне покоління, забезпечуючи поступове покращення продуктивності платформи.

Платформи, які обслуговують значні обсяги трафіку, потребують збалансованого розподілу запитів серед серверів, щоб уникнути перевантаження. Генетичні алгоритми можуть використовуватися для знаходження оптимальної конфігурації навантаження, щоб мінімізувати ризик затримок. Відбираються найбільш придатні конфігурації, де запити користувачів рівномірно розподілені між серверами. Ймовірність відбору $P(x_i)$ конфігурації x_i залежить від ефективності обробки навантаження:

$$P(x_i) = \frac{f(x_i)}{\sum_{j=1}^{N} f(x_j)}, \tag{17}$$

де $f(x_i)$ – значення функції пристосованості для конфігурації x_i, яке відображає рівень збалансованості навантаження.

Генетичні алгоритми також застосовуються для оптимізації продуктивності цифрових платформ шляхом комбінування найкращих конфігурацій (кросоверу) і внесення випадкових змін (мутації) для покращення роботи системи.

Найкраща конфігурація: два обраних рішення x_aі x_b, що добре розподіляють навантаження і мають низьку вартість, комбінуються для створення нових конфігурацій:

$$x_c = \alpha x_a + (1 - \alpha) x_b,$$

де α – коефіцієнт змішування, що визначає вагу кожної з конфігурацій у створенні нащадка x_c.

Випадкові зміни: щоб уникнути локальних максимумів і покращити стійкість платформи до раптових змін у навантаженні, генетичний алгоритм вводить невеликі випадкові зміни в конфігурації

$$\mathrm{x_i'} = \mathrm{x_i} + \delta,$$

де δ – випадкове значення, яке додається до параметрів конфігурації x_i, щоб змінити її характеристики.

Завдяки постійним ітераціям і селекції, генетичні алгоритми дозволяють цифровим платформам адаптуватися до динамічних змін, таких як підвищення навантаження у пікові години. Завдяки генетичним алгоритмам можна безперервно підлаштовувати конфігурації платформи, щоб зберігати оптимальний рівень продуктивності навіть за умов швидких змін.

Градієнтний спуск є одним із найважливіших алгоритмів для налаштування параметрів моделей на цифрових платформах. Він допомагає мінімізувати похибки моделі, дозволяючи платформам досягати точніших прогнозів і рекомендацій у реальному часі [17]. У контексті цифрових платформ градієнтний спуск використовується для задач, таких як рекомендації контенту, оптимізація реклами, класифікація користувачів, прогнозування попиту, що потребує оптимізації цільової функції для покращення роботи моделі.

Цільова функція $J(\theta)$ вимірює різницю між прогнозованими результатами моделі і фактичними даними. У задачах, таких як персоналізація контенту чи класифікація користувачів, цільова функція допомагає мінімізувати помилки передбачень, покращуючи таким чином точність рекомендацій та прогнозів. Наприклад, середньоквадратична похибка $J(\theta)$ для задач регресії на цифровій платформі визначається так:

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)} \right)^2, \tag{18}$$

де m – кількість навчальних прикладів у вибірці, $h_\theta\left(x^{(i)}\right)$ – прогнозоване значення моделі для вхідного прикладу $x^{(i)}$, $y^{(i)}$ – реальне значення для прикладу i.

Щоб зменшити похибку і покращити якість прогнозів, алгоритм градієнтного спуску поступово оновлює параметри θ моделі, змінюючи їх у напрямку найшвидшого зниження значення $J(\theta)$. Для цього використовується градієнт, що вказує напрямок найбільшого зменшення цільової функції.

Формула для оновлення параметрів

$$\theta \coloneqq \theta - \alpha \nabla J(\theta), \tag{19}$$

де α – коефіцієнт навчання (learningrate), який визначає, наскільки швидко модель оновлює свої параметри; градієнт $\nabla J(\theta)$ – вектор часткових похідних цільової функції по кожному параметру θ_J, що відображає, як змінювати кожен параметр, щоб зменшити похибку моделі.

Для середньоквадратичної похибки градієнт за параметром θ_J обчислюється так:

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m} \sum_{i=1}^{m} \left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right) x_j^{(i)}, \tag{20}$$

де $x_j^{(i)}$ – значення j-го ознаки для i-го прикладу.

На цифрових платформах, таких як стрімінгові сервіси або маркетингові платформи, градієнтний спуск застосовується для адаптації моделей до нових даних у реальному часі. Наприклад, якщо платформа регулярно отримує нові дані про вподобання користувачів або змінюється попит на певний контент, параметри моделі оновлюються на кожній ітерації, щоб забезпечити актуальність і точність рекомендацій. Ця адаптація знижує затримку обробки даних і дозволяє платформам реагувати на зміни у поведінці користувачів.

Важливим фактором успіху градієнтного спуску на цифрових платформах є вибір коефіцієнта навчання α. Якщо α занадто великий, модель може не збігатися і "перескакувати" мінімум, що призведе до неточностей у прогнозах. Якщо ж він занадто малий, навчання моделі буде повільним, що особливо важливо для платформ, де потрібно швидко реагувати на великі обсяги вхідних даних. Оптимальне значення α дозволяє платформі досягти балансу між швидкістю навчання і точністю результатів.

Висновки. Математичні моделі виконують ключову функцію та є важливим елементом в інформаційних системах управління на цифрових платформах. Вони дозволяють вирішувати широкий спектр задач, зокрема, оптимізацію ресурсів, балансування навантаження, прогнозування попиту та підвищення продуктивності. Використання алгоритмів Дейкстри та Флойд – Уоршелла, сприяє ефективній маршрутизації даних і побудові рекомендаційних систем. Методи оптимізації, наприклад, лінійне програмування або генетичні алгоритми, забезпечують адаптивність платформ до змінних умов роботи.

Прогнозування попиту та моделювання поведінки користувачів, основане на динамічних рядах і машинному навчанні, дає змогу платформам швидко реагувати на зміни в активності та забезпечувати високу якість обслуговування. Завдяки застосуванню стохастичних моделей цифрові платформи можуть масштабувати свої ресурси, зберігаючи стабільність навіть під час пікових навантажень.

Таким чином, математичні моделі це фундамент для розробки та вдосконалення інформаційних систем управління на цифрових платформах, сприяючи їхній ефективності, гнучкості та здатності задовольняти потреби користувачів у реальному часі.

У подальших дослідженнях буде розглянуто можливості та доцільність застосування сучасних програмних засобів для розв’язування оптимізаційних задач у мережах цифрових платформ. Це дозволить оцінити їхню ефективність у контексті реальних сценаріїв роботи інформаційних систем управління.

Список літератури

1. Gross J.L., Yellen J., Anderson M. Graph Theory and Its Applications, Chapman and Hall/CRC, 2018. 800 p.
2. Dijkstra E.W. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*. 1959. Vol. 1, No. 1. P. 269–271.
3. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms**,** 4th ed. Cambridge, MA: The MIT Press, 2022. 1312 p.
4. Парфенюк І. Персоналізація реклами в соціальних мережах: етичні виклики та загрози. *Цифрова платформа: інформаційні технології в соціокультурній сфері*. 2024. № 7. С. 148–158. https://doi.org/10.31866/2617-796X.7.1.2024.307017
5. Ткаченко О.М., Грійо-Тукало О.Ф., Дзісь О.В., Лаховець С.М. Метод кластеризації на основі послідовного запуску *k*-середніх з удосконаленим вибором кандидата на нову позицію вставки. *Наукові праці ВНТУ*, 2012. Вип. 2. С. 25–34.
6. Gallo G., Grigoriadis M.D., Tarjan R.E. A Fast Parametric Maximum Flow Algorithm and Applications. *SIAM Journal on Computin*. 1989. Vol. 18, No. 1. P. 30–55.
7. Stoer M., Wagner F.A. Simple Min-Cut Algorithm. *Journal of the ACM (JACM)*. 1997. Vol. 44, No. 4. P. 585–591.
8. Mathew S., Varia J. Overview of Amazon Web Services. Amazon Whitepapers, 2014. Vol. 105, No. 1. P. 22.
9. Задорожня Т., Шибко О. Використання лінійного програмування для розв'язання задач оптимізації. *Grail of Science*. 2023. № 26. С. 244–248. https://doi.org/10.36074/grail-of-science.14.04.2023.043
10. Wolsey L.A**.** Integer Programming. John Wiley & Sons, 2020.
11. Федік Л.Ю., Шабас М.Р. Методи рішення задач оптимізації. *Комп'ютерно-інтегровані технології: освіта, наука, виробництво*. 2013. № 12. С. 75–80.
12. Goldsman D., Goldsman P. Discrete-Event Simulation: Modeling and Simulation in the Systems Engineering Life Cycle: Core Concepts and Accompanying Lectures. London: Springer. 2015. P. 103–109.
13. Ruutu S., Casey T., Kotovirta V. Development and Competition of Digital Service Platforms: A System Dynamics Approach: Technological Forecasting and Social Change. 2017. Vol. 117. P. 119–130.
14. Mousavi S. et al. Dynamic Resource Allocation in Cloud Computing. *Acta Polytechnica Hungarica*. 2017. Vol. 14, No. 4. P. 83–104.
15. Kramer O. Genetic algorithm essentials. Studies in computational intelligence. Springer International Publishing, Cham, Switzerland, 1 edn. 2017. P. 11–19.
16. Гулаєва Н., Устілов А. Аналіз методів відбору в генетичних алгоритмах. *Наукові записки НаУКМА. Комп'ютерні науки*. 2021. **4**. С. 29–43. https://doi.org/10.18523/2617-3808.2021.4.29-43
17. Zhou S. Online Recommendation of Digital Trade Resources Based on Gradient Descent Optimization Algorithm: *2024 International Conference on Data Science and Network Security (ICDSNS)*. IEEE, 2024.

Одержано 01.04.2025

Годлюк Віктор Васильович,
аспірант Інституту кібернетики імені В.М. Глушкова, Київ, Україна.
https://orcid.org/0009-0007-4489-7058
goodiniv@ukr.net

УДК 004.75:519.8

В.В. Годлюк

Математичні моделі для інформаційних систем управління на цифрових платформах: від управління ресурсами до прогнозування попиту

Інститут кібернетики імені В.М. Глушкова НАН України, Київ
Листування: goodiniv@ukr.net

У сучасних умовах розвитку цифрових платформ ефективне управління ресурсами, оптимізація навантаження та точне прогнозування попиту є ключовими завданнями для забезпечення їх продуктивності та стабільності. У статті розглянуто математичні моделі та алгоритми, що застосовуються для інформаційних систем управління цифрових платформ. Досліджено використання методів оптимізації, графових алгоритмів, прогнозування та машинного навчання для підвищення ефективності цифрових систем.

Особливу увагу приділено математичним підходам до балансування навантаження, розподілу ресурсів і забезпечення масштабованості платформ. Описано застосування теорії графів для аналізу взаємозв'язків між користувачами, контентом і сервісами, а також алгоритмів Дейкстри та Флойда – Уоршелла для оптимальної маршрутизації даних. Розглянуто методи лінійного та цілочисельного програмування, що дозволяють знаходити оптимальні рішення для розподілу обчислювальних потужностей і зниження витрат.

Проаналізовано стохастичні та евристичні підходи до прогнозування попиту, включаючи методи машинного навчання, градієнтного спуску та генетичних алгоритмів. Описано використання алгоритмів максимального потоку та мінімального розрізу для ефективного управління мережевими ресурсами та мінімізації затримок. Запропоновано методи моделювання дискретних подій та неперервного моделювання для аналізу динамічних змін у цифрових системах.

Отримані результати можуть бути використані для підвищення адаптивності цифрових платформ до змінних умов роботи, зменшення ризиків перевантажень та покращення користувацького досвіду. Представлені математичні моделі сприяють розвитку ефективних механізмів управління цифровими сервісами, що є критично важливим для їхньої надійної та стійкої роботи.

Ключові слова: цифрові платформи, математичні моделі, управління ресурсами, балансування навантаження, прогнозування попиту, алгоритми Дейкстри та Флойда – Уоршелла, генетичні алгоритми, дискретне моделювання.

UDC 004.75:519.8

Viktor Godliuk

Mathematical Models for Management Information Systems on Digital Platforms: from Resource Management to Demand Forecasting

V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv
Correspondence: goodiniv@ukr.net

In the current conditions of development of digital platforms, efficient resource management, load optimization and accurate demand forecasting are key tasks to ensure their productivity and stability. The article deals with mathematical models and algorithms used for information management systems of digital platforms. The use of optimization methods, graph algorithms, forecasting, and machine learning to improve the efficiency of digital systems is investigated.

Particular attention is paid to mathematical approaches to load balancing, resource allocation, and platform scalability. The application of graph theory to analyze the relationships between users, content, and services, as well as Dijkstra and Floyd-Warshell algorithms for optimal data routing, is described. Linear and integer programming methods are considered to find optimal solutions for allocating computing power and reducing costs.

Stochastic and heuristic approaches to demand forecasting, including machine learning, gradient descent, and genetic algorithms, are analyzed. The use of maximum flow and minimum cut algorithms for efficient management of network resources and minimization of delays is described. The methods of discrete event and continuous modeling for analyzing dynamic changes in digital systems are proposed.

The results obtained can be used to increase the adaptability of digital platforms to changing operating conditions, reduce the risk of overloading, and improve user experience. The presented mathematical models contribute to the development of effective mechanisms for managing digital services, which is critical for their reliable and sustainable operation.

Keywords: digital platforms, mathematical models, resource management, load balancing, demand forecasting, optimization, Dijkstra and Floyd-Warshell algorithms, genetic algorithms, discrete simulation.