DOI: 10.35681/1560-9189.2024.26.1.308332

УДК 004.93

Є. М. Федорченко, А. О. Олійник, О. О. Степаненко, Т.А. Зайко, К. В. Мєдвєдєв, Ю. В. Федорченко, Т. В. Федорончак, Т. О. Колпакова
Національний університет «Запорізька політехніка»
вул. Жуковського, 64, 69063 Запоріжжя, Україна

Генетичний метод розв'язання задачі складання розкладу навчальних занять

Розглянуто проблему складання оптимального розкладу, яка полягає в пошуку оптимального розподілу навчальних занять на певний період часу при заданих обмеженнях. Розроблено послідовний і паралельний методи складання розкладу на основі генетичного пошуку. Запропоновані методи використовують адаптовані та модифіковані оператори ініціалізації, схрещування та селекції. Алгоритми, використовуючи цільову функцію, мінімізують конфлікти між заняттями та проміжок часу між заняттями, враховують рекомендований час і місце проведення. Розроблені методи дозволяють скоротити час на планування навчального процесу та уникнути помилок при створенні розкладу. Проведено порівняльний аналіз між класичним і модифікованим генетичним алгоритмом, і встановлено, що модифікований алгоритм працює швидше та ефективніше за класичний. Також порівняно роботу модифікованого алгоритму з різними операторами та параметрами генетичного алгоритму для встановлення найкращих. Отримані результати дозволяють запропонувати ефективні методи для підвищення якості складання розкладу та покращення процесу навчання в університеті.

Ключові слова: *генетичний алгоритм, розклад, еволюційний алгоритм, заняття, обмеження.*

Вступ

Управління часом і ресурсами є критично важливим аспектом для досягнення успіху в будь-якій сфері, зокрема в університетській освіті. Складання розкладу занять — це складна задача, що вимагає великої кількості ресурсів, оскільки треба враховувати одночасно багато обмежень: дотримання навчального плану, проведення занять у спеціалізованих аудиторіях, врахування потреб викладачів і студентів тощо [1]. Всі ці фактори неможливо врахувати при ручному створенні розкла-

ду. Тому доцільно автоматизувати цей процес для оптимального використання ресурсів.

Через значну кількість обмежень і складність побудови математичної моделі задачу складання розкладу можна віднести до задачі класу NP-повних. Для цієї задачі кількість можливих розв'язків може бути дуже великою залежно від розміру вхідних даних. Зазвичай використовуються алгоритми, які шукають найкращі наближені розв'язки. Одним із варіантів вирішення задачі планування занять є використання метаевристичних алгоритмів, які починають з одного або декількох початкових розв'язків і використовують стратегії пошуку, за допомогою яких наближаються до оптимального розв'язку.

Генетичний алгоритм — метаевристичний алгоритм, який є потужним інструментом для розв'язання задача оптимізації. Генетичні алгоритми є адаптованими до різних задач та умов, мають можливість знаходити рішення в глобальному масштабі. У даній роботі запропоновано модифікований генетичний алгоритм рішення складання занять для університету.

Аналіз літературних даних і постановка проблеми

Впровадження автоматичних систем для складання розкладів університету є актуальним завданням, яке має велике значення для покращення управління навчальним процесом. За останні десятиліття багато досліджень і розробок було проведено з метою оптимізації процесу розкладання робочого часу та ресурсів в університетських навчальних закладах.

У статті [1] автори використовують генетичний алгоритм та евристичний пошук для розв'язання проблеми планування розкладу в університеті. Методика створення розкладу базується на генетичних алгоритмах, які спрямовані на максимізацію кількості успішно запланованих одиниць уроків у розкладі з урахуванням різноманітних обмежень, таких як доступність викладачів, аудиторій та ін.

У роботі [2] використовується метод генетичних алгоритмів для вирішення проблеми складання розкладу занять в університеті чи коледжі. Автори запропонували систему, де складні об'єкти та процеси, зокрема розклади занять, час лекцій і доступні аудиторії кодуються у вигляді бінарних послідовностей, і на їхній основі створюються рішення з використанням генетичних операцій, таких як відбір, кросовер і мутація.

Стаття [3] розглядає проблему планування графіка занять і досліджує різні методи для її оптимізації. У статті наводиться огляд методу генетичного пошуку для створення розкладу, проте зазначено, що такий метод працює досить повільно, що обумовлює доцільність розробки паралельної реалізації відповідного методу.

Задача складання розкладу полягає в створенні оптимального розкладу занять з урахуванням різних обмежень і вимог. Ця задача належить до класу NP-повних задач, тому що являє собою комбінаторну оптимізаційну задачу, в якій кількість рішень зростає зі збільшенням розміру вхідних даних, обмежень і параметрів.

Цільова функція для цієї задачі повинна відображати основні критерії для оптимізації. Основними критеріями оцінки для розкладу є: 1) мінімізація кількості вікон між парами: зменшити простій часу викладачів і груп; 2) рівномірне розподілення навантаження протягом тижня: рівномірно розподілити навантаження на

викладачів і студентів для підвищення їхньої працездатності; 3) мінімізація конфліктів і накладань занять: передбачає уникнення перекриття між заняттями, які мають спільні групи, викладачів, або проводяться в одній аудиторії; 4) урахування пропозицій щодо розкладу: розклад повинен бути складений таким чином, щоби максимально задовольнити потреби студентів або викладачів.

Отже цільова функція для задачі складання розкладу — це сума штрафних функцій, які показують наскільки порушується обмеження при генерації розкладу. Штрафні функції можуть мати різну вагомість, залежно від того, наскільки важливим є дотримання певного обмеження. Ведемо наступні позначення: *ng* — кількість груп; *nt* — кількість викладачів; *na* — кількість аудиторій; *ns* — кількість занять; *D* — кількість навчальних днів; *P* — кількість занять на день; *Y* — тип заняття; $G = [g_1, g_2, ..., g_{ng}]$ — розклад відносно груп; $T = [t_1, t_2, \ldots, t_{nt}]$ — розклад відносно викладачів; $A = [a_1, a_2, \ldots, a_{na}]$ — розклад відносно аудиторій; $C = [c_1, c_2, \ldots, c_{ns}]$ — загальний розклад; $R = [r_{i,1}, r_{i,2}, \ldots, r_{i,nr}]$ — рекомендований час проведення занять, де *i* — номер заняття, а *nr* — кількість рекомендованого часу; $S = [s_{i,1}, s_{i,2}, \ldots, s_{i,ns}]$ — номер заняття для занять, де *i* — номер заняття, а *ns* — кількість пар у занятті; $X = [x_1, x_2, \ldots x_{nx}]$ — розклад окремої групи, викладача або аудиторій, де *nx* — кількість занять в окремій групі, викладача або аудиторії.

Введемо наступні функції: day(*x*) — генерує випадковий номер тижня для заняття *x*; pair(*x*) — генерує випадковий номер проведення заняття *x*, type(*x*) — генерує випадковий номер типу заняття *x*: чисельник, знаменник, загальна пара.

Уведемо штрафну функцію $f_w(X, y)$, яка розраховує вікна між заняттями та має відповідний ваговий коефіцієнт k_1. Ця функція обчислює різницю між номерами проведення наступного та попереднього заняття в межах одного дня. Спочатку треба порахувати вікна окремо по чисельнику, знаменнику та загальне. Нехай чисельнику відповідає значення 1, знаменнику — 2, загальному заняттю — значення 3. Слід також зазначити, що перед використанням цієї функції, обов'язково потрібно відсортувати масив занять: спочатку за днем тижня, потім за номером заняття, потім за типом заняття.

Нехай $g(x_i, x_{i+1}, y)$ — це функція відмінності занять, де *i* — це конкретне заняття:

$$g(x_i, x_{i+1}, y) = \begin{cases} k_1 \cdot \big(pair(x_{i+1}) - pair(x_i)\big), \text{якщо} \\ day(x_{i+1}) = day(x_i) \text{ та } type(x_{i+1}) = type(x_i) = y, \\ 0, \text{інакше.} \end{cases} \quad (1)$$

де k_1 — ваговий коефіцієнт; y — тип заняття; x_i та x_{i+1} — два заняття в розкладі; day(*x*) — функція, яка генерує випадковий номер тижня для заняття *x*; pair(*x*) — генерує випадковий номер проведення заняття *x*; type(*x*) — генерує випадковий номер типу заняття *x*: чисельник, знаменник, загальна пара.

Отримуємо таку формулу для підрахунку вікон:

$$f_w(X, y) = \sum_{i=1}^{nx-1} g(x_i, x_{i+1}, y), \quad (2)$$

Також уведемо функції $f_1(X)$, $f_2(X)$, $f_3(X)$, що враховують особливості навчального процесу. Для того, щоб прорахувати кількість простоїв між заняттями, треба врахувати, що заняття можуть проводитися з різною частотою: раз на тиждень, раз

на два тижні. Функція $f_1(X)$ — це сума простоїв між заняттями по чисельнику, знаменнику та загальних:

$$f_1(X) = f_w(X, 1) + f_w(X, 2) + f_w(X, 3). \quad (3)$$

Функція $f_2(X)$ розраховує накладання занять і має відповідний ваговий коефіцієнт k_2. Накладанням занять можна вважати ситуацію, коли в один і той же час заплановано два або більше занять. Виняткові ситуації виникають, коли ці два заняття розташовані в розкладі і в чисельнику, і в знаменнику.

Введемо функцію $h(x_i, x_j)$, яка обробляє всі випадки накладання занять, де x_i та x_j — окремі заняття:

$$h(x_i, x_j), = \begin{cases} 1, if\ type(x_i) = 1 \text{ та } type(x_j) = 1 \text{ або } 3, \\ 1, if\ type(x_i) = 2 \text{ та } type(x_j) = 2 \text{ або } 3, \\ 1, if\ type(x_i) = 3 \text{ та } type(x_j) = 1 \text{ або } 2 \text{ або } 3, \\ 0, \text{інакше.} \end{cases} \quad (4)$$

Функція, яка розраховує накладання занять:

$$f_2(X) = k_2 \cdot \sum_{i=1}^{nx} \sum_{j=1}^{nx} h(x_i, x_j). \quad (5)$$

Функція f_3 враховує дотримання рекомендацій щодо часу проведення заняття та має відповідний ваговий коефіцієнт k_3.

Введемо функцію $q(r_i, r_j, s_i, s_j)$, яка обробляє всі випадки дотримання рекомендацій щодо часу проведення, де r_i та r_j — окремі заняття в поточному розкладі, s_i та s_j — рекомендований час для цих занять:

$$q(r_i, r_j, s_i, s_j) = \begin{cases} 1, if\ day(r_{i,j}) \neq day(s_{i,j}) \\ \text{або } pair(r_{i,j}) \neq pair(s_{i,j}) \\ 0, \text{інакше.} \end{cases} \quad (6)$$

Функція виглядає наступним чином:

$$f_3 = k_3 \cdot \sum_{i=1}^{nc} \sum_{j=1}^{nr} q(r_i, r_j, s_i, s_j). \quad (7)$$

Загальна формула розрахунку штрафу для розкладу з використанням штрафних функцій (2), (3) буде мати вигляд:

$$F = \sum_{i=1}^{ng} f_1(g_i) + f_2(g_i) + \sum_{i=1}^{nt} f_1(t_i) + f_2(t_i) + \sum_{i=1}^{na} f_2(a_i) + f_3. \quad (8)$$

За допомогою вагових коефіцієнтів можемо коригувати важливість дотримання відповідного обмеження. Якщо значення коефіцієнта є високим, то в пріоритеті в алгоритму буде мінімізувати саме це обмеження.

Планування навчальних занять є складною комбінаторною задачею з великою кількістю обмежень і великим обсягом даних. Традиційні методи розкладу, які застосовуються до деяких інших задач, не підходять для цієї конкретної задачі. Проте існують евристичні та обчислювальні методи, які можуть бути успішно використані для її розв'язання.

Зазвичай для розв'язання задачі планування використовують евристичні методи, які базуються на евристичних алгоритмах та інтуїтивних підходах для пошуку оптимальних рішень. Ці методи швидше знаходять прийнятні розв'язки, але

не гарантують знаходження найкращого можливого варіанту через відсутність математичного обґрунтування їхньої роботи. Для визначення, наскільки близьким є знайдений розклад до оптимального розв'язку, часто порівнюють його з результатами методу грубої сили. Незважаючи на це обмеження, цей підхід є ефективним у випадках, коли знайти абсолютно найкращий варіант надто складно або неможливо.

Під час аналізу наявних методів розв'язання проблеми було встановлено, що найбільш підхожим алгоритмом для складання оптимального розкладу занять є генетичний алгоритм. Тому було вирішено розробити адаптований і модифікований генетичний алгоритм для розв'язання поставленої задачі.

Алгоритм для складання розкладу повинен мінімізувати значення визначеної цільової функції (8). Для оцінювання якості результатів роботи алгоритму буде використовуватись описана вище цільова функція, яка показує кількість порушень заданих обмежень.

Розробка модифікації генетичного алгоритму

Для оптимізації цільової функції було обрано еволюційний генетичний метод. Генетичний алгоритм поступово наближається до оптимального розкладу шляхом підбору, комбінування та варіації можливих розв'язків.

Основними кроками генетичного алгоритму є: ініціалізація вхідних даних і параметрів алгоритму; ініціалізація початкової популяції; обчислення пристосованості індивідів; схрещування індивідів; мутація осіб у популяції; обчислення пристосованості отриманих індивідів; селекція; продовження кроків, поки буде задовільнено критерій [5].

Вхідними даними для алгоритму є: список занять, кожне заняття містить викладача або викладачів, одну або декілька груп, дисципліну, тип заняття, кількість разів проведення заняття; список рекомендованого розкладу для занять; кількість навчальних днів; максимальна кількість занять на день; список аудиторій.

Алгоритм має наступні параметри: розмір популяції; кількість ітерацій; імовірність мутації; імовірність схрещування; імовірність мутації гену.

Ініціалізація в генетичному алгоритмі — це процес створення початкової популяції індивідів, які в подальшому будуть еволюціонувати [6].

Для ініціалізації пошуку можна використовувати метод випадкової ініціалізації, яка дає можливість включити в початкову множину різноманітні рішення, що збільшує ймовірність знаходження оптимального розкладу. Для кожного заняття випадково визначається день тижня, номер проведення, тип та аудиторія. Якщо заняття має рекомендований розклад, то він буде враховуватися [6].

Використання випадкової ініціалізації може провести до суттєвого збільшення часу пошуку. Тому запропоновано модифікований метод генерації початкової популяції генетичного пошуку, який використовує апріорну інформацію про особливості розкладу, відому із заданих обмежень. У модифікованому методі випадкової ініціалізації генерується день тижня та обирається аудиторія, після чого відбувається пошук можливого часу проведення заняття. Якщо це перше заняття в обраному дні, то можна його вставити випадково. Якщо вже є заняття, тоді підбирається номер таким чином, щоб мінімізувати порушення обмеження. В результаті отримаємо розклад, в якому будуть відсутні накладання занять і зменшена кількість

вікон. Використання цього методу ініціалізації дозволяє значно прискорити роботу генетичного алгоритму.

Схрещування — це операція в генетичному алгоритмі, яка використовується для створення нової популяції. Схрещування полягає в обміні генів між двома батьківськими особинами з метою створення нащадків з новою комбінацією генів [6]. Було прийнято рішення обмінювати лише по одному випадковому заняттю в двох випадкових розкладах. Це дає можливість іншим генам змінюватися в наступному поколінні та зберегти корисні комбінації.

Батьківські розклади — це дві особини (розклади), які використовуються для створення нових розкладів шляхом операції схрещування.

Для задачі розкладу із заняттями, хромосома представляє весь розклад, а гени — це окремі заняття та їхні часові слоти в цьому розкладі.

Математично функцію схрещування для обміну одного випадкового заняття між двома розкладами батьків можна виразити такою формулою.

Нехай P_1 та P_2 — батьківські розклади із заняттями, де G_1 представляє вибране заняття з P_1, і G_2 — вибране заняття з P_2. Тоді C_1 та C_2 — це нащадки, які формуються шляхом обміну цими заняттями:

$$C_1 = (P_1 \setminus \{G_1\}) \cup \{G_2\}, \quad (9)$$

$$C_2 = (P_2 \setminus \{G_2\}) \cup \{G_1\}. \quad (10)$$

Також для порівняння було реалізовано метод k-точкового схрещування. При k-точковому схрещенні випадково вибирається k точок на батьківських хромосомах. Ці точки поділяють хромосому на $(k + 1)$ сегментів. Далі обмінюються сегментами між двома батьківськими хромосомами, щоб створити дві нові хромосоми-потомки. Зазвичай, використовують парний кратний k, щоби забезпечити, що кожен сегмент батьківських хромосом буде розбитий на дві рівні частини [6].

Нехай P_1 та P_2 — дві батьківські хромосоми, де $P_1 = [p_1, p_2, p_3, \dots, p_m]$ і $P_2 = [q_1, q_2, q_3, \dots, q_n]$. C_1 і C_2 — два нащадки, які формуються за допомогою k-точкового схрещування. k — це кількість точок, які випадково обираються для поділу хромосом на сегменти.

K-точкове схрещування можна описати таким чином. Випадково вибираємо k точок на батьківських хромосомах P_1 та P_2, позначимо ці точки як $p_1, p_2, \dots, p_k$ для P_1 та $q_1, q_2, \dots, q_k$ для P_2. Створюємо нових нащадків C_1 та C_2. C_1 буде формуватися шляхом об'єднання сегментів із P_1 та P_2, де використовуються всі сегменти P_1, крім сегментів між точками p_1 та p_2, p_2 і p_3, ..., p_k та p_{k+1}, і всі сегменти P_2, між відповідними точками q_1 та q_2, q_2 та q_3, ..., q_k і q_{k+1}. C_2 буде формуватися об'єднанням сегментів із P_2 та P_1, де використовуються всі сегменти P_2, крім сегментів між точками q_1 і q_2, q_2 і q_3, ..., q_k і q_{k+1}, і всі сегменти P_1 між відповідними точками p_1 та p_2, p_2 і p_3, ..., p_k і p_{k+1}. Математично це можна представити таким чином:

$$C_1 = P_1[1{:}\,p_1] \cup P_2[q_1{:}\,q_2] \cup P_1[p_{1+1}{:}\,p_2] \cup P_2[q_{2+1}{:}\,q_3] \cup \dots \cup P_1[p_{k+1}{:}\,p_k] \cup P_2[q_k{:}\,] \quad (11)$$

$$C_2 = P_2[1{:}\,q_1] \cup P_1[p_1{:}\,p_2] \cup P_2[q_{1+1}{:}\,q_2] \cup P_1[p_{2+1}{:}\,p_3] \cup \dots \cup P_2[q_{k+1}{:}\,q_k] \cup P_1[p_k{:}\,], \quad (12)$$

де $P_1[\text{a}{:}\,b]$ позначає сегмент від a до b в хромосомі P_1.

Перевагою k-точкового схрещення є те, що воно забезпечує різноманітність у нащадках, дозволяє зберегти корисні комбінації генетичного матеріалу батьків, а також може допомогти уникнути попадання у локальний мінімум в оптимізації функції витрат. Однак його недоліком є те, що він може вести до втрати корисної інформації від батьківських хромосом, особливо при великих значеннях k [6].

Мутація використовується для збільшення різноманітності популяції і запобігання затяганню в локальних оптимумах. Вона випадковим чином змінює один або більше генів з метою створення нових можливих рішень [6].

Якщо ми будемо змінювати багато генів у індивіда, то це може зробити індивіда не придатним, тому що більшість змін не буду корисною. Тому було реалізовано мутацію, яка змінює розклад у одного випадкового заняття.

Можна описати операцію мутації математично наступним чином:

$$P_{mut} = (P \backslash \{G_{old}\}) \cup \{G_{new}\}, \tag{13}$$

де P — батьківська хромосома до мутації; P_{mut} — хромосома після мутації, в якій один випадковий ген був замінений; G_{old} — ген, який був видалений з батька P під час мутації; G_{new} — новий ген, який додали до батька P під час мутації.

Мутація всіх генів з шансом — це модифікація генетичного алгоритму, яка полягає у зміні значення всіх генів у кожному окремому рішенні з певною ймовірністю. Цей метод використовується для того, щоби допомогти генетичному алгоритму виходити з локальних оптимумів, які можуть з'являтися під час процесу пошуку рішення. При мутації генів з шансом випадкові зміни генів можуть привести до виявлення нових, більш оптимальних рішень [6].

У генетичному алгоритмі селекція — це процес відбору кращих рішень для формування наступної популяції. Суть селекції полягає в збереженні найкращих особин і усуненні гірших, що призводить до поступового поліпшення рішень з кожною наступною ітерацією. Було реалізовано наступні методи селекції:

— турнірна: випадково обираються два рішення, пріоритет відається рішенню з більшим фітнес-значенням;

— ранжуванням: метод вибору рішень на основі їхнього ранжування за пристосованістю;

— рулеткою: відбір кандидатів наступного покоління на основі ймовірностей, які відповідають їхній пристосованості [6].

Опишемо математично формулою турнірну селекцію. Нехай S_1 та S_2 — два випадково обрані рішення. S — рішення, яке отримує пріоритет на основі їхньої фітнес-функції Fitness:

$$S = \begin{cases} S_1, Fitness(S_1) \geq Fitness(S_2), \\ S_2, Fitness(S_1) < Fitness(S_2); \end{cases} \tag{14}$$

Математично селекцію ранжуванням можна описати наступним чином. Розраховуємо ранг кожного рішення S_n в популяції на основі його фітнес-значення.

Обчислюємо суму рангів усіх рішень у популяції:

$$R_s = \sum_{i=1}^{n} i, \tag{15}$$

де n — кількість індивідів у популяції; i — ранг індивіда.

Для кожного рішення S_n розраховуємо ймовірність вибору, використовуючи його ранг:

$$P(S_n) = R(S_n)/R_s, \tag{16}$$

де $P(S_n)$ — імовірність вибору індивіда S_n; $R(S_n)$ — ранг індивіда S_n.

Обране рішення S у селекції за ранжуванням вибирається наступним чином:

$$S = S_n, \text{якщо } P(S_{n-1}) < rand(0,1) \le P(S_n), \tag{17}$$

де $rand(0,1)$ — випадкове число від 0 до 1.

У методі селекції рулеткою ймовірність вибору індивіда пропорційна його фітнес-значенню.

Обчислюємо суму фітнес-значень усіх рішень у популяції:

$$F_s = \sum_{i=1}^{n} Fitness(i), \tag{18}$$

де n — кількість індивідів у популяції; $Fitness(i)$ — фітнес індивіда i.

Імовірність вибору кожного рішення розраховується так:

$$P(S_n) = Fitness(S_n)/F_s, \tag{19}$$

де $P(S_n)$ — імовірність вибору індивіда S_n; $Fitness(S_n)$ — фітнес індивіда S_n.

Селекція за допомогою рулетки розраховується наступним чином:

$$S = S_i, \text{якщо } i \in [1; n] \text{ та } \sum_{i=1}^{n} P(S_i) \ge rand(0,1), \tag{20}$$

де n — кількість індивідів у популяції; $P(S_i)$ — імовірність вибору індивіда; $rand(0,1)$ — випадкове число від 0 до 1.

У селекції було реалізовано масштабування пристосованості індивідів, що дозволяє збільшити швидкість збіжності алгоритму. Основна ідея полягає у тому, щоб змінити масштаб значень фітнесу таким чином, щоб вони лежали в діапазоні від a до b. Це може бути зроблено за допомогою функції масштабування, яка перетворює вихідний діапазон значень фітнесу на новий діапазон, що відповідає діапазону від a до b [7]. Масштабування пристосованості індивідів допомагає зменшити вплив різних масштабів фітнес-функцій на результати генетичного алгоритму і дозволяє більш ефективно знаходити оптимальне рішення [7].

Також для збільшення ймовірності знаходження оптимального рішення можна використовувати елітизм, який зберігає частину найкращих індивідів з одного покоління в інше без змін. Зберігання елітних індивідів з попередньої популяції дозволяє зберегти різноманітність і запобігти втраті корисної інформації [8].

Критерії зупинки алгоритму такі:

— якщо досягнуто максимальної кількості ітерацій;

— якщо знайдено такий розклад, значення пристосованості якого дорівнює 0.

Одним із методів покращення класичного генетичного алгоритму є його острівна модель. Острівна модель ГА — це модель, в якій популяція розділена на декілька груп (островів). Кожен острів містить свою підпопуляцію, яка еволюціонує незалежно від інших островів [9].

За допомогою механізму міграції індивіди можуть переміщуватися з одного острова до іншого для пришвидшення збіжності алгоритму. Це допоможе зменшити ризик попадання в локальні мінімуми [9].

В острівній моделі підпопуляції можуть використовувати різні параметри алгоритму, різні оператори схрещування, мутації і селекції [9, 10].

Експерименти та результати

Вхідними даними для системи є: дані про аудиторії: назва аудиторії, тип, місткість, закріплені кафедри; дані про кафедри: назва кафедри та її скорочена назва; дані про спеціальності: назва спеціальності, код, кафедра; дані про дисципліни: назва, спеціальність, семестр, в якому викладається дисципліна; дані про групи: назва групи, кількість студентів, поточний семестр, спеціальність; дані про викладачів: ПІБ викладача та кафедра, на якій він працює; дані про заняття: дисципліна, тип заняття, кількість занять на тиждень, викладачі, групи, рекомендовані аудиторії, рекомендований час проведення [10–27].

Вихідними даними є оптимальний розклад занять, що складений для обраної кафедри, в якому враховуються рекомендації до проведення та мінімізовано такі параметри: кількість накладань занять для груп, вчителів та аудиторій; кількість вікон між заняття для груп і вчителів.

Для порівняння роботи алгоритму використовувався параметр «швидкість збіжності», який показує, наскільки швидко або ефективно алгоритм наближається до свого оптимального розв'язку. Цей параметр можна розрахувати за такою формулою:

$$S = \frac{|F_s - F_f|}{|t_f - t_s|}, \tag{21}$$

де S — швидкість збіжності; F_s — початкове значення фітнес; F_f — кінцеве значення фітнес; t_s — початковий час роботи алгоритму; t_f — кінцевий час.

Також треба позначити наступні скорочення:

а) P_e — значення елітизму (elitism), тобто відсоток від загального числа популяції;

б) T_c — тип схрещування (crossing), який може приймати такі значення: 1 — «custom_one_gene»; 2 — «k_point»;

в) N_k — кількість k-точок схрещування;

г) P_c — імовірність схрещування;

е) T_m — тип мутації (mutation), який може приймати такі значення: 1 — «custom_one_gene»; 2 — «all_genes»;

є) P_{mg} — імовірність мутації гена;

ж) P_m — імовірність мутації;

з) T_s — тип вибірки (selection), який може приймати такі значення: 1 — «roulette», 2 — «ranging»; 3 — «tournament»;

и) T_i — тип ініціалізації (initialization), який може приймати такі значення: 1 — «random»; 2 — «simple_algorithm»;

і) N_g — максимальна кількість ітерацій роботи алгоритму;

ї) N_p — розмір популяції.

Для створення розкладу враховувалися наступні ключові значення. Кількість днів для складання розкладу: 6 днів. Кількість проведених занять на день для складання розкладу: 6 занять. Штраф за вікна у груп: 1, це означає, що було приділено увагу уникненню незаповненого часу між заняттями для студентських груп. Штраф

за вікна у викладачів: 1, підкреслюючи важливість уникання вільних перерв між заняттями викладачів. Штраф за накладання занять: 5, що показує важливість уникання конфліктів і перекриття в розкладі. Штраф за невідповідність рекомендованого часу проведення занять: 5, вказуючи на важливість дотримання встановлених годин проведення занять. Ці параметри були використані для підтримки оптимізації розкладу, забезпечення відповідності вимогам та уникнення конфліктів і накладань при його створенні.

У табл. 1 наведено параметри запуску тестів для генетичного алгоритму, включаючи розмір популяції (500) і максимальну кількість ітерацій (2000). Згідно з табл. 2, наведеними є результати цих тестів. Важливо зазначити, що кожен запис у табл. 2 представляє собою середнє значення результатів трьох проведених експериментів з однаковими параметрами.

Таблиця 1. Параметри запуску тестів генетичного алгоритму

#	P_e	T_c	N_k	P_c	T_m	P_{mg}	P_m	T_s	T_i
1	0,1	1	–	0,7	1	–	0,2	1	1
2	0,1	1	–	0,7	1	–	0,4	1	1
3	0,1	1	–	0,7	1	–	0,6	1	1
4	0,1	1	–	0,7	2	0,05	0,2	1	1
5	0,1	1	–	0,7	2	0,1	0,2	1	1
6	0,1	1	–	0,7	2	0,2	0,2	1	1
7	0,1	1	–	0,4	1	–	0,2	1	1
8	0,1	1	–	0,6	1	–	0,2	1	1
9	0,1	1	–	0,8	1	–	0,2	1	1
10	0,2	1	–	0,6	1	–	0,2	1	1
11	0,3	1	–	0,6	1	–	0,2	1	1
12	0,1	1	–	0,6	1	–	0,2	2	1
13	0,1	1	–	0,6	1	–	0,2	3	1
14	0,1	2	2	0,6	1	–	0,2	3	1
15	0,1	2	4	0,6	1	–	0,2	3	1
16	0,1	2	6	0,6	1	–	0,2	3	1
17	0,1	1	–	0,6	1	–	0,2	3	2
18	0,1	1	–	0,6	1	–	0,4	3	2
19	0,1	1	–	0,8	1	–	0,2	3	2
20	0,1	1	–	0,6	2	0,1	0,2	3	2
21	0,3	1	–	0,6	1	–	0,2	3	2
22	0,1	2	4	0,6	1	–	0,2	3	2

Параметри генетичного алгоритму, такі як розмір популяції і максимальна кількість ітерацій, залишалися постійними для всіх тестів, щоб забезпечити порівнянність результатів між різними експериментами.

У табл. 3 представлено параметри, що використовувалися під час тестів модифікації генетичного алгоритму — острівної моделі. У табл. 4 містяться результати цих тестів, і кожен тест було проведено три рази, після чого обчислено середнє значення результатів. У всіх тестах використовувались однакові параметри: розмір популяції 500 і максимальна кількість ітерацій 2000.

Таблиця 2. Результати генетичного алгоритму

#	t_s, с	t_f, с	F_s	F_f	S
1	0,983	1937,365	1128,667	22,000	0,577
2	0,968	1852,033	1157,333	24,667	0,613
3	0,981	1884,862	1148,000	119,000	0,546
4	0,991	1840,709	1178,667	75,000	0,600
5	0,964	1865,484	1164,333	80,333	0,581
6	1,008	1881,805	1156,667	54,333	0,586
7	0,953	1851,585	1183,667	22,000	0,628
8	0,952	1796,640	1165,000	17,667	0,639
9	0,966	1808,654	1164,000	15,000	0,636
10	0,960	1794,841	1158,333	14,000	0,638
11	0,979	1840,432	1148,333	18,000	0,615
12	0,983	1878,421	1165,333	10,000	0,615
13	1,003	1839,763	1155,333	14,667	0,620
14	1,023	1980,765	1159,000	15,667	0,578
15	1,135	2196,617	1156,667	10,000	0,522
16	1,023	1980,765	1159,000	15,667	0,578
17	1,116	947,481	8,333	1,333	0,014
18	0,922	1334,211	9,667	1,333	0,008
19	0,914	1320,637	9,667	2,000	0,009
20	0,929	680,211	10,333	2,667	0,076
21	1,021	859,483	14,667	1,000	0,019
22	1,075	1559,583	6,333	1,333	0,016

Слід увести наступні позначення: N_i — кількість островів; N_{st} — крок збільшення значень у островах; N_{it} — кількість ітерацій через яку проводити міграцію між островами; P_{mig} — яка кількість індивідів буде приймати участь у міграції.

Таблиця 3. Параметри запуску тестів модифікації генетичного алгоритму — острівної моделі

#	P_e	T_c	P_c	T_m	P_m	T_s	T_i	Ni	N_{st}	N_{it}	P_{mig}
1	0,1	1	0,4	1	0,2	1	1	6	2	10	0,1
2	0,1	1	0,4	1	0,2	1	1	12	2	10	0,1
3	0,1	1	0,4	1	0,2	1	1	18	2	10	0,1
4	0,1	1	0,4	1	0,2	1	1	6	3	10	0,1
5	0,1	1	0,4	1	0,2	1	1	6	5	10	0,1
6	0,1	1	0,4	1	0,2	1	1	6	7	10	0,1
7	0,1	1	0,4	1	0,2	1	1	6	7	5	0,1
8	0,1	1	0,4	1	0,2	1	1	6	7	8	0,1
9	0,1	1	0,4	1	0,2	1	1	6	7	13	0,1
10	0,1	1	0,4	1	0,2	1	1	6	7	5	0,05
11	0,1	1	0,4	1	0,2	1	1	6	7	5	0,15
12	0,1	1	0,4	1	0,2	1	1	6	7	5	0,25
13	0,1	1	0,4	1	0,2	1	2	12	5	5	0,05
14	0,1	1	0,4	1	0,4	1	2	12	5	5	0,05
15	0,1	1	0,6	1	0,4	1	2	12	5	5	0,05
16	0,3	1	0,6	1	0,4	1	2	12	5	5	0,05

Таблиця 4. Результати роботи тестів модифікації генетичного алгоритму — острівної моделі

#	t_s, с	t_f, с	F_s	F_f	S
1	1,144	1924,096	1123,667	14,667	0,577
2	2,211	3663,413	1101,000	6,333	0,299
3	3,068	5399,514	1118,667	4,000	0,207
4	1,123	1835,984	1119,333	9,333	0,605
5	1,121	1892,263	1112,667	6,000	0,585
6	1,100	1893,736	1094,333	5,333	0,576
7	1,160	1898,343	1131,000	5,000	0,593
8	1,060	1929,846	1124,000	8,000	0,579
9	1,120	1880,434	1152,333	10,667	0,608
10	1,070	1899,741	1143,000	5,667	0,599
11	1,080	1913,294	1109,000	7,000	0,576
12	1,161	1906,083	1146,667	11,000	0,596
13	1,497	1314,078	383,000	3,333	0,217
14	1,640	1819,898	10,667	0,667	0,012
15	1,606	1485,881	7,000	1,000	0,011
16	1,971	640,981	9,667	0,000	0,015

Обговорення результатів дослідження

Проаналізувавши табл. 2, можна зробити наступні висновки щодо найбільшої швидкості збіжності у параметрів ГА: ймовірність мутації — 0,2; тип мутації — мутація одного гену; ймовірність схрещування — 0,6; значення елітизму — 0,2; тип вибірки — рулетка; тип схрещування — схрещування одного гену.

Для аналізу впливу параметра $T_i = 2$ (використання модифікованого параметра ініціалізації) порахуємо, наскільки в середньому скоротився час роботи алгоритму порівняно зі значеннями $T_i = 1$. Для цього розрахуємо середнє значення часу виконання алгоритму для обох варіантів. Середнє значення часу роботи можна розрахувати за такою формулою:

$$t_{avr} = (\sum_{i=1}^{n}(t_{fi} - t_{si}))/n, \quad (22)$$

де t_{fi} — кінцевий час роботи в запуску; t_{si} — початковий час роботи алгоритму в запуску; n — кількість запусків.

Середнє значення часу роботи ГА при $T_i = 1$ — 1888,429 с.

Середнє значення часу роботи ГА при $T_i = 2$ — 1115,938 с.

Скорочення часу роботи алгоритму ГА з $T_i = 2$ порівняно з $T_i = 1$ становить приблизно 40,9 %. Скорочення часу роботи на 40,9 % вказує на поліпшення продуктивності алгоритму за рахунок використання значення $T_i = 2$.

Проаналізувавши табл. 4, можна зробити наступні висновки щодо найбільшої швидкості збіжності у параметрів острівної моделі: кількість островів — 6 (кількість логічних ядер процесора); крок розбіжності параметрів — 7; через скільки ітерацій проводити міграцію — 10; кількість індивидів для участі в міграції — 0,1.

Середнє значення часу роботи острівної моделі при $T_i = 1$ — 2335,027 с.

Середнє значення часу роботи острівної моделі при $T_i = 2$ — 1313,531 с.

При використанні модифікованого методу ініціалізації острівна модель ГА в середньому працює швидше в 1,8 разів. Це покращує продуктивність алгоритму, що дозволяє зекономити час і ресурси при розв'язанні задачі.

Протестуємо алгоритми з параметрами, які дають найкращі результати (табл. 5).

На основі результатів можна зробити висновок, що острівна модель генетичного алгоритму є кращою з точки зору швидкості та якості знаходження рішення порівняно з класичним генетичним алгоритмом. У середньому, острівна модель ГА працює значно швидше — час виконання роботи в середньому скоротився на 41,3 %, має кращий результат фітнес-функції, середнє значення фітнесу дорівнює 0, це значить, що алгоритм знайшов ідеальне рішення та робить його більш ефективним алгоритмом для даної задачі.

Таблиця 5. Результати тестів алгоритмів з найкращими параметрами

Алгоритм	№	t_s, с	t_f, с	F_s	F_f
Генетичний алгоритм	1	1,077	847,888	8,000	0,000
	2	1,020	1320,838	11,000	0,000
	3	1,178	1946,044	12,000	2,000
	Середнє	1,092	1371,590	10,333	0,667
Острівна модель ГА	4	1,857	2016,956	3,000	0,000
	5	1,009	349,973	12,000	0,000
	6	1,048	50,780	4,000	0,000
	Середнє	1,305	805,903	6,333	0,000

Висновки

Розроблено модифікований генетичний метод, що використовує оператор ініціалізації на основі апріорної інформації про навчальний процес, яка доступна із заданих обмежень. Використання розробленого підходу до ініціалізації генетичного методу дозволяє суттєво (в рази) скоротити час пошуку.

Розроблено модифіковану острівну модель розробленого генетичного методу для вирішення задачі складання оптимального розкладу навчальних занять. Принципова відмінність запропонованого методу від існуючих аналогів полягає у використанні модифікованого оператора ініціалізації, який намагається шляхом простого перебору можливих варіантів зменшити початкове значення фітнес. За допомогою модифікованого оператора ініціалізації час роботи алгоритму класичного генетичного алгоритму скоротився на 40,9 %. Також варто зазначити, що час роботи острівної моделі генетичного алгоритму з модифікованим оператором ініціалізації скоротився в 1,8 рази порівняно із використанням класичного методу ініціалізації. Це означає, що застосування цього методу дозволило значно зекономити час виконання алгоритму. Модифікований метод ініціалізації сприяє покращенню продуктивності та швидкості виконання генетичного алгоритму для складання розкладу навчальних занять, що є важливим для ефективного вирішення задачі.

Виконано експериментальне дослідження запропонованих генетичних методів. Острівна модель генетичного алгоритму виявилася більш ефективною як з точки зору швидкості, так і з точки зору якості отриманих рішень. У середньому, острівна модель ГА працює значно швидше — час виконання роботи в середньому скоротився на 41,3 %, має кращий результат фітнес-функції, середнє значення фітнесу дорівнює 0, це значить, що алгоритм знайшов ідеальне рішення та робить його більш ефективним алгоритмом для даної задачі.

1. Solving Timetable Problem by Genetic Algorithm and Heuristic Search Case Study:Universitas Pelita Harapan Timetable. URL: https://www.researchgate.net/publication/221927228_Solving_Timetable_Problem_by_Genetic_Algorithm_and_Heuristic_Search_Case_Study_Universitas_Pelita_Harapan_Timetable

2. An Evolutionary Algorithm for Solving Academic Courses Timetable Scheduling Problem. URL: https://www.researchgate.net/publication/356044560_An_Evolutionary_Algorithm_for_Solving_Academic_Courses_Timetable_Scheduling_Problem

3. A Review of Optimization Algorithms for University Timetable Scheduling. URL: https://www.researchgate.net/publication/347802207_A_Review_of_Optimization_Algorithms_for_University_Timetable_Scheduling

4. Some Methods of Solving the NP-difficult Problem of Optimal Schedule for the University. URL: https://www.sciencedirect.com/science/article/pii/S187705091930417X#:~:text=However%2C%20there%20are%20a%20number%20of%20heuristic%20and,method%2C%20graph%20coloring%20method%2C%20intelligent%20method%2C%20genetic%20algorithm

5. Гайтан О.М. Автоматизація генерації розкладу навчального процесу університету. *Інформатика, обчислювальна техніка та автоматизація*. 2020. Т. 31(70), Ч. 1. № 2. С. 58–66.

6. Снитюк В.Є., Сіпко О.М. Технологія еволюційного формування розкладів у закладах вищої освіти/монографія. Київ: Видавець ФОП Піча Ю.В., 2022. 136 с.

7. Мулява І.Я. Вирішення задачі автоматизованого формування розкладу навчального закладу за допомогою генетичних алгоритмів. *Міжнародний науковий журнал «Інтернаука»*. 2018. № 9(1). С. 77–83.

8. Кисіль В.В., Драч І.В., Кисіль Т.М. Модель задачі складання та оптимізації розкладу занять за умови задоволення об'єктивних та суб'єктивних вимог навчального закладу. *Інформатика, обчислювальна техніка та автоматизація*. 2019. Том 30(69) Ч. 1, № 6. С. 65–70.

9. Томашевський В.М., Новіков Ю.Л., Камінська П.А. Складання розкладів занять у дистанційних системах навчання. *Вісник Національного технічного університету України «КПІ» Іформатика, управління та обчислювальна техніка*. 2010. Вип. 52. С. 118–130.

10. Genetic Algorithm in Machine Learning. URL: https://www.javatpoint.com/genetic-algorithm-in-machine-learning.

11. Генетичні алгоритми. Ключові поняття і методи реалізації. URL: http://www.znannya.org/?view=ga_general

12. An Illustrated Guide to Genetic Algorithm. URL: https://towardsdatascience.com/an-illustrated-guide-to-genetic-algorithm-ec5615c9ebe

13. Genetic Algorithm — Advantages & Disadvantages. URL: https://electricalvoice.com/genetic-algorithm-advantages-disadvantages/

14. Семенчук В.М. Особливості використання острівної моделі генетичних алгоритмів. Тернопільський національний технічний університет імені Івана Пулюя, 2020. С. 114.

15. Oliinyk A., Fedorchenko I., Stepanenko A., Katschan A., Fedorchenko Y., Kharchenko A., Goncharenko D. Development of genetic methods for predicting the incidence of volumes of emissions of pollutants in air. 2019 2nd International Workshop on Informatics and Data-Driven Medicine, IDDM, CEUR Workshop Proceedings. 2019.Vol. 2488. P. 340–353.

16. Fedorchenko I., Oliinyk A., Stepanenko A., Svyrydenko A, Goncharenko D. Genetic method of image processing for motor vehicle recognition. 2019 2nd International Workshop on Computer Modeling

and Intelligent Systems, CMIS, 2019. Zaporizhzhia, April 15–19, CEUR Workshop Proceedings. Vol. 2353. P. 211–226.

17. Fedorchenko I., Oliinyk A., Stepanenko A., Zaiko T., Korniienko S., Burtsev N. Development of a genetic algorithm for placing power supply sources in a distributed electric network. *European Journal of Enterprise Technologies*. 2019. Issue 5/101. P. 6–16. doi: 10.15587/1729-4061.2019.180897.

18. Oliinyk A., Fedorchenko I., Stepanenko A., Rud M., Goncharenko D. Implementation of evolutionary methods of solving the travelling salesman problem in a robotic warehouse. *Lecture Notes on Data Engineering and Communications Technologie*s. 2021. **48**. P. 263–292.

19. Fedorchenko I., Oliinyk A., Stepanenko A., Zaiko T., Korniienko S., Kharchenko A. Construction of a genetic method to forecast the population health indicators based on neural network models. *Eastern-European Journal of Enterprise Technologies*. 2020. **1**(4–103). P. 52–63. DOI: 10.15587/1729-4061.2020.197319

20. Phang F.A., Pusppanathan J., Nawi N.D., Zulkifli N.A., Zulkapri I., Harun F.K.C., Wong A.Y.K., Alsayaydeh J.A.J., Sek T.K. Integrating Drone Technology in Service Learning for Engineering Students. *International Journal of Emerging Technologies in Learning*. 2021. **16**(15). P. 78–90.

21. Zakir Hossain A.K.M., Hassim N.B., Alsayaydeh J.A.J., Hasan M.K., & Islam M.R. A tree-profile shape ultra wide band antenna for chipless RFID tags. *International Journal of Advanced Computer Science and Applications*. 2021. **12**(4). P. 546–550. doi:10.14569/IJACSA.2021.0120469.

22. Jamil Abedalrahim Jamil Alsayaydeh*, Azwan Aziz, A. I. A. Rahman, Syed Najib Syed Salim, Maslan Zainon, Zikri Abadi Baharudin, Muhammad Inam Abbasi and Adam Wong Yoon Khang. DEVELOPMENT OF PROGRAMMABLE HOME SECURITY USING GSM SYSTEM FOR EARLY PREVENTION, ARPN. *Journal of Engineering and Applied Sciences*. 2021. Vol. 16, No. 1. P. 88–97.

23. Mishra S., Sachan R., and Rajpal D. Deep convolutional neural network based detection system for real-time corn plant disease recognition. *Procedia Comput. Sci*. 2020. Vol. 167 P. 2003–2010. Accessed: Nov. 8, 2023. [Online]. Available: https://doi.org/10.1016/j.procs.2020.03.236

24. Indra W.A., Zamzam N.S., Saptari A., Alsayaydeh J.A.J, Hassim N.B. Development of Security System Using Motion Sensor Powered by RF Energy Harvesting. 2020 IEEE Student Conference on Research and Development, SCOReD. 2020. 9250984. P. 254–258.

25. Jung M. et al. Construction of deep learning-based disease detection model in plants. *Scientific Rep*. May 2023. Vol. 13, No. 1. Accessed: Nov. 8, 2023. [Online]. Available: https://doi.org/10.1038/s41598-023-34549-2

26. Adam Wong Yoon Khang, Shamsul J. Elias, Nadiatulhuda Zulkifli, Win Adiyansyah Indra, Jamil Abedalrahim Jamil Alsayaydeh, Zahariah Manap, Johar Akbar Mohamat Gani, 2020. Qualitative Based QoS Performance Study Using Hybrid ACO and PSO Algorithm Routing in MANET. *Journal of Physics*. Conference Series 1502 (2020) 012004. doi:10.1088/1742-6596/1502/1/012004.

27. Lee S.-H., Wu C.-C., and Chen S.-F. Development of image recognition and classification algorithm for tea leaf diseases using convolutional neural network. In 2018 Detroit, Mich. July 29 – August 1, 2018. St. Joseph, MI: Amer. Soc. Agricultural Biol. Engineers, 2018. Accessed: Nov. 8, 2023. [Online]. Available: https://doi.org/10.13031/aim.201801254.

Надійшла до редакції 10.11.2023