УДК 004.032.26:37.091.214

Ю. О. Лотиш
Я. В. Іванчук

ГЕНЕТИЧНИЙ АЛГОРИТМ ОПТИМІЗАЦІЇ ЗАДАЧІ АВТОМАТИЗОВАНОГО ПЛАНУВАННЯ РОЗКЛАДУ ЗАНЯТЬ

Вінницький національний технічний університет

Анотація

У тезах розглянуто проблему автоматизованого планування навчального розкладу як складну NP-складну комбінаторну задачу оптимізації. Обґрунтовано доцільність застосування генетичного алгоритму як ефективного евристичного методу пошуку наближених до оптимальних рішень в умовах великої кількості обмежень. Особливу увагу приділено формалізації та реалізації функції пристосованості з урахуванням м'яких обмежень. Запропоновано алгоритм оцінки пристосованості на прикладі обмеження щодо проведення занять однієї дисципліни, одного типу та однієї групи підряд в одній аудиторії. Розроблено математичні вирази для розрахунку штрафів за порушення критерію та описано структуру даних для реалізації алгоритму. Отримані результати можуть бути використані при створенні інтелектуальних систем автоматизованого планування розкладу занять та розширені шляхом урахування додаткових критеріїв оптимізації.

Ключові слова: генетичний алгоритм; автоматизоване планування; розклад занять; функція пристосованості; м'які обмеження; комбінаторна оптимізація; евристичні методи.

Abstract

The theses address the problem of automated timetable scheduling as a complex NP-hard combinatorial optimization task. The feasibility of applying a genetic algorithm as an effective heuristic method for searching near-optimal solutions under numerous constraints is substantiated. Particular attention is paid to the formalization and implementation of the fitness function with consideration of soft constraints. A fitness evaluation algorithm is proposed based on the constraint requiring that classes of the same discipline, type, and group be scheduled consecutively in the same classroom. Mathematical expressions for penalty calculation in case of constraint violations are developed, and the data structure required for algorithm implementation is described. The obtained results can be applied in the development of intelligent automated timetable scheduling systems and extended by incorporating additional optimization criteria.

Keywords: genetic algorithm; automated scheduling; timetable; fitness function; soft constraints; combinatorial optimization; heuristic methods.

Вступ

Проблема формування ефективного навчального розкладу є однією з ключових в організації освітнього процесу, оскільки від його якості безпосередньо залежить рівень засвоєння знань студентами, комфортність умов праці викладачів та раціональне використання матеріально-технічної бази навчального закладу. Тому актуальним є автоматизація процесу планування, яка дозволяє не лише суттєво скоротити час на розробку розкладу, але й забезпечити дотримання усіх вимог навчального плану та індивідуальних побажань викладачів [1].

Складання навчального розкладу – типова NP-складна комбінаторна задача, де класичні алгоритмічні методи часто не справляються через велику кількість обмежень і різні, часом суперечливі, критерії ефективності [2]. У такому разі доцільно застосовувати евристичні методи, зокрема генетичний алгоритм, який завдяки операціям селекції, схрещування і мутації здатен ефективно досліджувати простір можливих розкладів і шукати наближення до оптимального рішення [3].

Метою роботи є розробка алгоритму автоматизованого планування розкладу занять для складного критерію.

Результати дослідження

Генетичні алгоритми – це методи, засновані на принципах природного відбору та еволюції, які використовуються для розв'язання задач оптимізації, де традиційні методи можуть виявитися

неефективними. Цей підхід імітує процес природної еволюції для пошуку оптимальних або наближених до оптимальних рішень у великих і складних просторах пошуку [4].

Замість того, щоб починати з однієї точки (або припущення) в просторі пошуку, генетичні алгоритми ініціалізуються з популяцією припущень. Зазвичай вони є випадковими і розподіляються по всьому простору пошуку. Типовий алгоритм використовує три оператори: відбір, кросовер (схрещення) і мутацію (вибрані частково за аналогією з природним світом), щоб скерувати популяцію (протягом серії часових кроків або поколінь) до збіжності в глобальному оптимуму [5].

Відбір – це обрання найефективнішої моделі та передача її генів іншому поколінню. Пари особин (батьків) відбирають залежно від пристосованості. Кількість батьків може бути довільною, але однаковою для кожного наступного циклу. Ключові методи відбору – турнірний і ранговий. У турнірному випадковим чином відбирають кілька особин. Серед них визначають найкращу. Процес повторюється, поки проміжна популяція не буде заповнена. В свою чергу у ранговому популяцію розбивають на кілька груп (рангів). Ранги присвоюють залежно від функції пристосованості та пропорційно відбирають деяку частину особин із кожного рангу [6].

Функція пристосованості – це цільова функція в генетичних алгоритмах, яка чисельно оцінює якість кожного можливого розв’язку та визначає, які з них матимуть більше шансів на відбір у процесі еволюційної оптимізації [7]. Функція пристосованості складається з набору критеріїв оцінювання розв’язку, їхніх ваг (якщо критеріїв кілька), а також правил обчислення штрафів за порушення обмежень, що разом формують числову міру якості рішення [8]. Обмеження поділяються на жорсткі та м’які. Жорсткі обмеження – це обмеження, які повинні неодмінно задовольнятись; такі, які фізично не можуть бути порушені. Приклад жорсткого обмеження: «В один і той же час у зазначеній аудиторії має бути заняття одного викладача з однієї дисципліни». М’які обмеження (критерії оптимізації) – це обмеження, які можна порушувати, але ці порушення повинні бути зведеними до мінімуму. Їх виконання не є таким же обов’язковим як жорстких [9].

Певні обмеження можуть мати важке формулювання і/або бути важкими у реалізації. Розглянемо алгоритм оцінки пристосованості на прикладі одного м’якого обмеження, а саме: «заняття з однієї дисципліни, одного типу та однієї групи проводити підряд в одній аудиторії». Для того щоб перевірити чи уроки підряд та в одній аудиторії можна зберігати інформацію про них у описі відповідного дня. Такий опис складатиметься із списку порядкових номерів уроків та множини аудиторій, це дозволить зручно знаходити кількість пропусків між уроками, а також швидко розуміти чи кількість аудиторій не перевищує потрібну. Враховуючи що ці уроки можуть бути розподілені по різним дням тижня, і що це є лише частиною критерію, потрібно також зробити прив'язку до іншої впливової характеристики, наприклад, груп. Таким чином, інформація про групу міститиме дані про те у які дні проводяться уроки, також їх порядкові номери та аудиторії.

Подання інформації про те, коли відбуваються уроки та у яких груп уже є, але потрібно також додати дисципліну та тип заняття для того щоб можна було повністю представити обмеження. Враховуючи що дисципліна розділяється на типи – у неї має бути вищий рівень у ієрархії. У навчальних програмах існує певна множина дисциплін, тому потрібно додати новий рівень, який буде включати всі дисципліни. Такий рівень не має певної назви, як наприклад «тип заняття», але враховуючи що він описує множину дисциплін, можна назвати його списком/множиною дисциплін. Усі отримані залежності між даними більш наочно подані на рисунку 1.

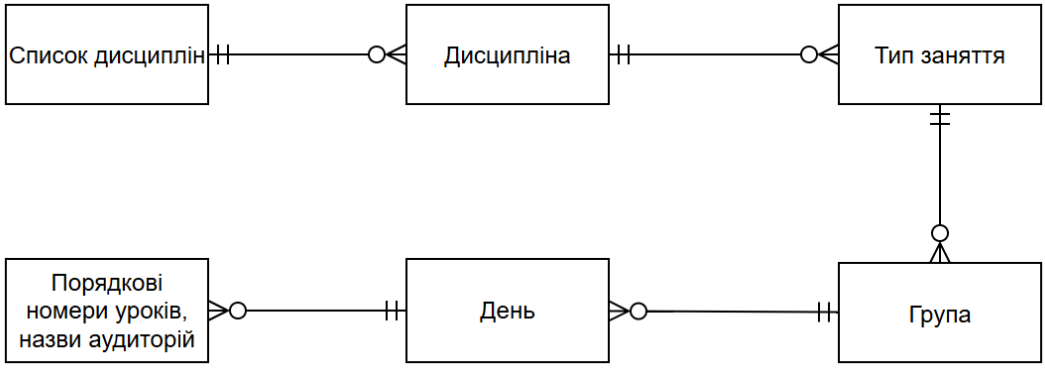

Рис. 1. Схема зв’язку між даними

Залежності, зображені на рисунку 1, можна описати таким чином: «Одна дисципліна може містити багато типів заняття. Один тип заняття обов’язково належить одній дисципліні».

Маючи такі залежності, можна повністю визначити чи є початкове твердження, «заняття однієї дисципліни, одного типу та однієї групи проводити підряд в одній аудиторії», правдивим і розробити схему алгоритму планування розкладу занять (рис. 2).

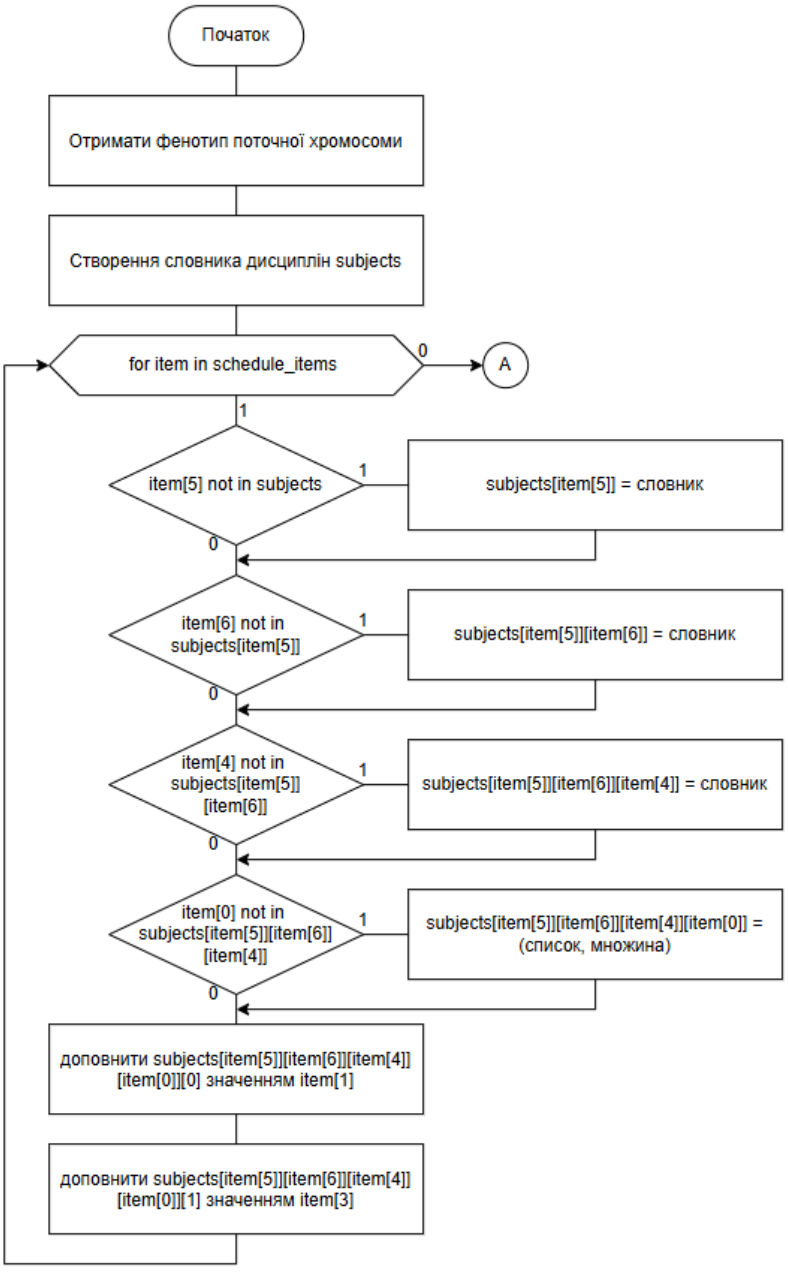

Рис. 2. Схема алгоритму планування розкладу занять

Для цього, починаючи з найвищого рівня ієрархії, опустимося до рівня груп, та перевіримо чи містить опис однієї групи (враховуючи ієрархію, відомо що це відбувається у рамках одного типу конкретної дисципліни) кілька днів. Якщо так, то це є великим порушенням у рамках пропусків між уроками, яке можна оцінити за формулою

$$penalty = penalty + (len(days) - 1) * MAX\_LESSONS, \quad (1)$$

де *penalty* – штраф; *days* – список днів; *MAX_LESSONS* – максимальна кількість уроків за день; *len(x)* – довжина об’єкту; «=» – оператор присвоєння.

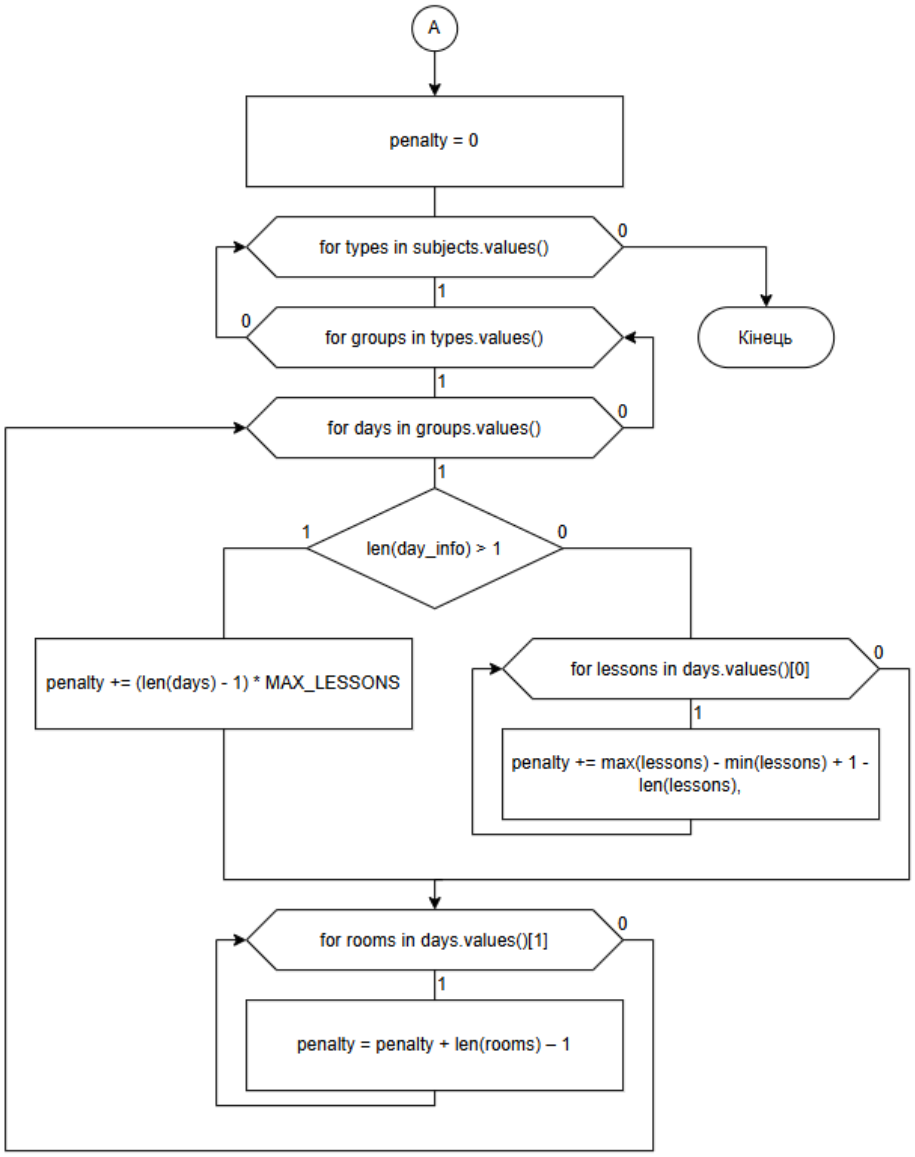

Рис. 2. Продовження

Якщо день лише один, можна знайти кількість уроків, що розділяють заняття однієї дисципліни, за формулою

$$penalty = penalty + max(lessons) - min(lessons) + 1 - len(lessons), \quad (2)$$

де *penalty* – штраф; *lessons* – список порядкових номерів уроків; *max(x)* – максимальне значення об'єкту; *min*(*x*) – мінімальне значення об'єкту; *len*(*x*) – довжина об'єкту; «=» – оператор присвоєння.

Після визначення штрафу за пропуски між уроками, до отриманого значення потрібно додати штраф за різні аудиторії, який можна обрахувати за наступною формулою

$$penalty = penalty + len(rooms) – 1, \quad (3)$$

де *penalty* – штраф; *rooms* – множина аудиторій; *len*(*x*) – довжина об'єкту; *max*(*x*) – максимальне значення об'єкту; «=» – оператор присвоєння.

У випадку мінімізації функції пристосованості штраф потрібно додавати, а у випадку максимізації – віднімати.

На рисунку 2 *schedule_items* – двовимірний масив, кожна комірка якого містить наступну інформацію:

- день проведення уроку (індекс 0);
- порядковий номер уроку (індекс 1);

- викладач (індекс 2);
- аудиторія (індекс 3);
- назва групи (індекс 4);
- дисципліна (індекс 5);
- тип заняття (індекс 6).

Цикли *for*, що використовуються, мають властивості циклів *for* мови програмування Python, або ж *foreach* мови програмування C#. Наприклад *for x in data* означає, що змінній *x* поступово присвоюється значення кожного елементу масиву *data*, це можна інтерпретувати як *for (int i = 0; i < data.Length; i++) {x = data[i];}*.

Твердження *value not in Iterable* має властивості однойменного твердження мови програмування Python, або, наприклад, *Iterable.Contains(value)* мови програмування C#, що повертає булеве значення відповіді на питання «чи міститься об`єкт *value* у ітераційному об`єкті *Iterable*».

Метод «.values()» повертає лише значення із пар «ключ – значення», що є у словнику. Функція *len(Iterable)* повертає довжину ітераційного об`єкта *Iterable*.

Висновки

У роботі розглянуто проблему автоматизованого планування навчального розкладу як складну комбінаторну задачу оптимізації та обґрунтовано доцільність застосування генетичних алгоритмів для її розв'язання. Запропоновано алгоритм оцінки пристосованості на прикладі одного м'якого обмеження, що дозволяє кількісно оцінювати штраф за його порушення. Також було представлено і описано схему алгоритму.

Результати роботи створюють передумови для підвищення якості сформованих розкладів, зменшення конфліктів і покращення організації освітнього процесу. Розроблений алгоритм може бути використаний при створенні інтелектуальних систем автоматизованого планування розкладу занять та розширені шляхом врахування додаткових критеріїв оптимізації.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Наукова праця з репозиторію НУ «Львівська політехніка» [Електронний ресурс]. – Режим доступу: https://ena.lpnu.ua:8443/server/api/core/bitstreams/44f95415-9cdb-4ca7-9d00-b849b899436c/content (дата звернення: 22.12.2025).

2. Турчина В. А., Танасієнко Д. О. Застосування генетичного алгоритму до задачі складання навчального розкладу // Питання прикладної математики і математичного моделювання. – Вип. 18. – [Електронний ресурс]. – Режим доступу: https://doi.org/10.15421/321820 (дата звернення: 22.12.2025).

3. Наукова праця з репозиторію НТУ «ХПІ» [Електронний ресурс]. – Режим доступу: https://repository.kpi.kharkov.ua/server/api/core/bitstreams/04a366f1-7d20-4bcc-b0bf-ba7b5a14b133/content (дата звернення: 24.12.2025).

4. Наукова стаття з репозиторію КНТЕУ [Електронний ресурс]. – Режим доступу: https://ur.knute.edu.ua/server/api/core/bitstreams/bded0dee9a74-4733-a7e2-7554e6483e70/content (дата звернення: 24.12.2025).

5. Coley D. A. An Introduction to Genetic Algorithms for Scientists and Engineers [Електронний ресурс]. – Режим доступу: http://ftp.demec.ufpr.br/CFD/bibliografia/an_introduction_to_genetic_algorithms_for_scientists_and_engineers_coley.pdf (дата звернення: 24.12.2025).

6. Як працюють генетичні алгоритми [Електронний ресурс]. – Режим доступу: https://robotdreams.cc/uk/blog/186-kak-rabotayut-geneticheskie-algoritmy (дата звернення: 21.02.2026).

7. Goldberg D. E. Genetic Algorithms in Search, Optimization and Machine Learning [Електронний ресурс]. – Режим доступу: https://www2.fiit.stuba.sk/~kvasnicka/Free%20books/Goldberg_Genetic_Algorithms_in_Search.pdf (дата звернення: 21.02.2026).

8. Mitchell M. An Introduction to Genetic Algorithms [Електронний ресурс]. – Режим доступу: https://www.boente.eti.br/fuzzy/ebook-fuzzy-mitchell.pdf (дата звернення: 22.02.2026)

9. Снитюк В. Є. Технологія еволюційного формування розкладів у закладах вищої освіти [Електронний ресурс]. – Режим доступу: https://www.researchgate.net/profile/Vitaliy-Snytyuk/publication/366759465_Tehnologia_evolucijnogo_

formuvanna_rozkladiv_u_zakladah_visoi_osviti/links/63b1a335a03100368a45d195/Tehnologia-evolucijnogo-formuvanna-rozkladiv-u-zakladah-visoi-osviti.pdf (дата звернення: 22.02.2026).

Лотиш Юлія Олександрівна – студентка групи 3КН-22б, факультет інтелектуальних інформаційних технологій та автоматизації, Вінницький національний технічний університет, Вінниця, e-mail: lotisula077@gmail.com

Іванчук Ярослав Володимирович – д-р техн. наук, професор, професор кафедри комп`ютерних наук, Вінницький національний технічний університет, Вінниця

Lotysh Yuliia O. – Department of Intelligent Information Technologies and Automation, Vinnytsia National Technical University, Vinnytsia, email: lotisula077@gmail.com

Ivanchuk Yaroslav V. – Dr. Sc. (Eng.), Professor of Computer Science, Vinnytsia National Technical University, Vinnytsia.