МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Черкаський національний університет
імені Богдана Хмельницького
MINISTRY OF EDUCATION AND SCIENCE OF UKRAINE
Bohdan Khmelnytsky
National University of Cherkasy

DOI: 10.31651/2076-5886-2021-1

ISSN 2076-5886 (Print)

ВІСНИК ЧЕРКАСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ БОГДАНА ХМЕЛЬНИЦЬКОГО

Серія
ПРИКЛАДНА МАТЕМАТИКА. ІНФОРМАТИКА

BULLETIN OF THE CHERKASY BOHDAN KHMELNYTSKY NATIONAL UNIVERSITY

APPLIED MATHEMATICS. INFORMATICS

ВИПУСК 1
ISSUE 1

Черкаси, 2021
Cherkasy, 2021

Засновник, редакція, видавець і виготовлювач –
Черкаський національний університет імені Богдана Хмельницького.
Свідоцтво про державну реєстрацію КВ № 16161-4633 ПР від 11.12.2009.
Свідоцтво про державну реєстрацію КВ № 23973-13813 Р від 21.05.2019.

Журнал розрахований на математиків, спеціалістів у галузі ІТ, викладачів, науковців, аспірантів, студентів.

Випуск № 1 наукового журналу «Вісник Черкаського національного університету імені Богдана Хмельницького. Серія «Прикладна математика. Інформатика» рекомендовано до друку та до поширення через мережу Інтернет Вченою радою Черкаського національного університету імені Богдана Хмельницького (протокол № 3 від 13.12.2021 року).

Журнал індексується Google Scholar.

За дотримання права інтелектуальної власності, достовірність матеріалів та обґрунтування висновків відповідають автори.

Адреса редакційної колегії:
18031, Черкаси, бул. Шевченка, 79
Черкаський національний університет імені Богдана Хмельницького, корпус № 3, к. 307
e-mail: ami.cdu@ukr.net

З електронною версією журналу можна ознайомитися за адресою: http://ami-ejournal.cdu.edu.ua/

Founder, editorial, publisher and manufacturer –
Bohdan Khmelnytsky National University of Cherkasy.
State registration certificate: KV No. 16161-4633 PR dated 11.12.2009.
State registration certificate: KV No. 23973-13813 R dated 21.05.2019.

This journal is meant for mathematicians, IT specialists, teachers, researchers, postgraduates and students.

Issue № 1 of the scientific journal «Bulletin of the Cherkasy Bohdan Khmelnytsky national university. Applied mathematics. Informatics» is recommended for publication and dissemination through the Internet by the Academic Council of Bohdan Khmelnytsky National University of Cherkasy (protocol number 3 dated 13.12.2021).

The journal is indexed in Google Scholar.

The authors are responsible for the observance of the intellectual property right, for the reliability of the materials and for the substantiation of the conclusions.

Editorial office address:
18031, Cherkasy, Shevchenko Blvd., 79
Bohdan Khmelnytsky National University of Cherkasy, building 3, ap. 307
e-mail: ami.cdu@ukr.net

All electronic versions of articles are available on the website edition http://ami-ejournal.cdu.edu.ua/

СЕКЦІЯ «ПРИКЛАДНА МАТЕМАТИКА»

УДК 517.9:539.3

DOI 10.31651/2076-5886-2021-1-4-13

PACS 02.70.-c

СТЕБЛЯНКО Павло Олексійович,
доктор-фізико математичних наук,
професор, професор кафедри кібербезпеки
та інформаційних технологій, Університет
митної справи та фінансів
e-mail: caf-vmi@ukr.net
ORCID: 0000-0003-0789-4409

ДЬОМІЧЕВ Костянтин Едуардович,
кандидат технічних наук, доцент,
професор кафедри комп'ютерних наук,
Київський міжнародний університет
e-mail: demichevk@gmail.com
ORCID: 0000-0002-3428-4094

ПЕТРОВ Олександр Дмитрович,
доктор філософії, молодший науковий
співробітник, Дніпровський національний
університет ім. О.Гончара
e-mail: caf-vmi@ukr.net
ORCID: 0000-0001-8688-043X

МОДЕЛЮВАННЯ ПОВЕДІНКИ ЛОКАЛЬНО НАВАНТАЖЕНОЇ ПОЛОСИ З ПСЕВДО-ПРУЖНО-ПЛАСТИЧНОГО МАТЕРІАЛУ ПРИ ВЕЛИКИХ ПЛАСТИЧНИХ ДЕФОРМАЦІЯХ

У роботі проведено моделювання поведінки локально навантаженої послабленої полоси з псевдо-пружно-пластичного матеріалу при її нестаціонарному навантаженні. При цьому використанно нелінійну феноменологічну модель яка описує властивості сплавів з пам'яттю форми та термо-псевдо-пластичну поведінку (ТППМ) матеріалу саме в точці. Використано діаграму псевдо-пружного матеріалу, що складається з трьох криволінійних ділянок. Такий підхід призводить до нестійкої діаграми напруження-деформація і для описання термомеханічної поведінки зразків різної форми необхідно мати рішення граничної задачі з урахуванням розвитку фронту деформації фазового перетворення. Здійснено порівняння результатів отриманих в геометрично лінійній і нелінійній постановках при великих пластичних деформаціях. Встановлено, що при пластичних деформаціях до 6% розбіжність результатів в точках локалізації деформації не перевищує 5%. При збільшенні значень пластичної деформації розбіжність результатів може значно зростати і в околі послаблення досягати 20% .

Ключові слова: *феноменологічна модель, нелінійна модель матеріалу, матеріали з паматтю форм, термо-пвевдо-пластичність, великі пластичні деформації.*

Вступ

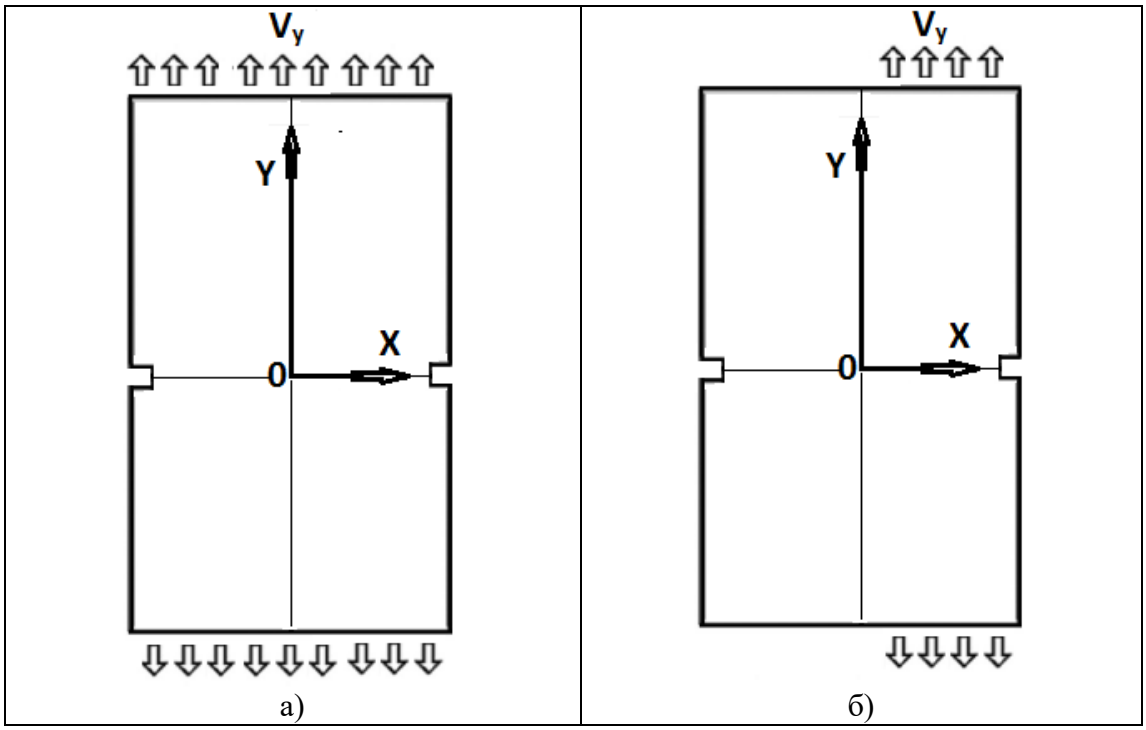
На даний час відомий цілий ряд моделей для опису термомеханічної поведінки функціонально-неоднорідних матеріалів, зокрема сплавів з пам'яттю форми (СПФ) [3; 4]. Більшість з них будуються на підставі класичних уявлень, тобто ставлять собі за

мету безпосереднє описати експериментальні дані, отримані на різних макрозразках при простому і складному навантаженні. Однак, як встановлено в експериментальних дослідженнях поведінка матеріалу в точці тіла в загальному випадку може бути відмінною від поведінки зразка вцілому [5]. В розглянутих задачах розроблено числову процедуру розрахунку діаграми матеріалу, яка представляє собою криву, що огинає сімейство діаграм матеріалу, побудованих для певних законів зміни швидкості фронту розриву деформацій.

Мета роботи полягає у застосуванні **нелінійної** феноменологічної моделі [7], яка описує властивості сплавів з пам'яттю форми та термо-псевдо-пластичну поведінку (ТППМ) матеріалу саме в точці, до задачі моделювання локально навантаженої полоси з псевдо-пружно-пластичного матеріалу при великих пластичних деформаціях. Порівняти результати моделювання в геометрично лінійній і нелінійній постановках при великих пластичних деформаціях.

Виклад основного матеріалу

Розглянемо двовимірну задачу про нестаціонарне деформування полоси із сплаву NiTi з малими послабленнями. Нехай геометрія полоси $x \in [-H/2; H/2]$, $y \in [-L/2; L/2]$. Для послаблюючого розрізу запишемо: $y \in [-l/2; l/2]$, $x \in [-H/2; -H/2+l] \cup [H/2-l; -H/2]$.



а) б)

Рисунок 1. Геометрія полоси з послабленнями

На краях з послабленнями ($x = -H/2, x = H/2$) поверхня полоси вільна від напруження. Поверхні самого послаблення полоси також вільні від напруження.

Для всіх точок полоси, для яких $y = 0$ швидкість вертикального переміщення v_y дорівнює нулю. На краї $y = L/2, x \in [H/2 - x_*]$. задається ненульова швидкість

переміщення $v_y = V_0$, яка змінюється по відомому закону. Для іншого края $y = -L/2, x \in [H/2 - x_*]$ маємо $v_y = -V_0$.

Шуканими величинами будуть дев'ять складових вектора $\vec{W}$: швидкості переміщень v_x, v_y; складові тензора напруження $\sigma_x, \sigma_y, \sigma_z, \sigma_{xy}$; деформації $\varepsilon_x, \varepsilon_y, \varepsilon_{xy}$ і температура T.

У випадку плоскої деформації ($u_3 \equiv 0$, $v_3 \equiv 0$, $\partial(\cdots)/\partial\alpha_3 \equiv 0$) при використанні ортогональної декартової системи координат ($H_1 = H_2 = H_3 = 0$) для геометрично лінійних складових тензора деформації отримаємо:

$$e_{11} = \frac{\partial u_1}{\partial x}; e_{22} = \frac{\partial u_2}{\partial y}; e_{33} = 0; e_{12} = \frac{\partial u_2}{\partial x} + \frac{\partial u_1}{\partial y}; e_{23} = 0; e_{31} = 0; \tag{1}$$

$$\omega_1 = 0; \omega_2 = 0; 2\omega_3 = \frac{\partial u_2}{\partial x} - \frac{\partial u_1}{\partial y}.$$

Тензор деформації і складові вектора переміщень пов'язані такими нелінійними співвідношеннями [1]:

$$\varepsilon_{11} = e_{11} + \frac{1}{2}\left[e_{11}^2 + \left(\frac{e_{12}}{2} + \omega_3\right)^2\right], \varepsilon_{22} = e_{22} + \frac{1}{2}\left[e_{22}^2 + \left(\frac{e_{21}}{2} + \omega_3\right)^2\right], \varepsilon_{33} = 0, \tag{2}$$

$$\varepsilon_{12} = e_{12} + e_{11}\left(\frac{e_{12}}{2} - \omega_3\right) + e_{22}\left(\frac{e_{12}}{2} + \omega_3\right), \varepsilon_{23} = 0, \varepsilon_{31} = 0.$$

З урахуванням цього після диференціювання за часом в геометрично нелінійному випадку для швидкостей деформацій при плоскій деформації можна записати:

$$\frac{\partial \varepsilon_{11}}{\partial t} = (1 + e_{11})\frac{\partial v_1}{\partial x} + \left(\frac{e_{12}}{2} + \omega_3\right)\frac{\partial v_2}{\partial x},$$

$$\frac{\partial \varepsilon_{22}}{\partial t} = \left(\frac{e_{21}}{2} - \omega_3\right)\frac{\partial v_1}{\partial y} + (1 + e_{22})\frac{\partial v_2}{\partial y},$$

$$\frac{\partial \varepsilon_{33}}{\partial t} = 0, \frac{\partial \varepsilon_{31}}{\partial t} = 0, \frac{\partial \varepsilon_{23}}{\partial t} = 0, \tag{3}$$

$$\frac{\partial \varepsilon_{33}}{\partial t} = (1 + e_{22})\frac{\partial v_2}{\partial x} + (1 + e_{11})\frac{\partial v_1}{\partial y} + \left(\frac{e_{12}}{2} - \omega_3\right)\frac{\partial v_1}{\partial x} + \left(\frac{e_{12}}{2} + \omega_3\right)\frac{\partial v_2}{\partial y}.$$

Рівняння руху мають вигляд:

$$\frac{\partial v_1}{\partial t} = \frac{1}{\rho}\cdot\frac{\partial \sigma_{11}}{\partial x} + \frac{1}{\rho}\cdot\frac{\partial \sigma_{21}}{\partial y} + \frac{1}{\rho}B_1,$$

$$\frac{\partial v_2}{\partial t} = \frac{1}{\rho}\cdot\frac{\partial \sigma_{12}}{\partial x} + \frac{1}{\rho}\cdot\frac{\partial \sigma_{22}}{\partial y} + \frac{1}{\rho}B_2, \tag{4}$$

де:

$$B_1 = K_1 + \frac{\partial}{\partial x}\left[\sigma_{11}^0 \frac{\partial}{\partial x}\left(u_1 + u_2 + u_3\right)\right],$$
$$B_2 = K_2 + \frac{\partial}{\partial y}\left[\sigma_{22}^0 \frac{\partial}{\partial y}\left(u_1 + u_2 + u_3\right)\right].$$

Після виключення з визначальних фізичних співвідношень швидкостей деформацій у випадку плоскій деформації отримаємо:

$$\frac{\partial \sigma_x}{\partial t} = \left\{a_{1111}\left(1+e_{11}\right) + a_{1112}\left(\frac{e_{12}}{2} - \omega_3\right)\right\}\frac{\partial v_1}{\partial x} +$$
$$+\left\{a_{1112}\left(1+e_{11}\right) + a_{1122}\left(\frac{e_{21}}{2} - \omega_3\right)\right\}\frac{\partial v_1}{\partial y} +$$
$$+\left\{a_{1111}\left(\frac{e_{12}}{2} + \omega_3\right) + a_{1112}\left(1+e_{22}\right)\right\}\frac{\partial v_2}{\partial x} +$$
$$+\left\{a_{1122}\left(1+e_{22}\right) + a_{1112}\left(\frac{e_{12}}{2} + \omega_3\right)\right\}\frac{\partial v_2}{\partial y} + b_*,$$

$$\frac{\partial \sigma_y}{\partial t} = \left\{a_{2211}\left(1+e_{11}\right) + a_{2212}\left(\frac{e_{12}}{2} - \omega_3\right)\right\}\frac{\partial v_1}{\partial x} +$$
$$+\left\{a_{2212}\left(1+e_{11}\right) + a_{2222}\left(\frac{e_{21}}{2} - \omega_3\right)\right\}\frac{\partial v_1}{\partial y} +$$
$$+\left\{a_{2211}\left(\frac{e_{12}}{2} + \omega_3\right) + a_{2212}\left(1+e_{22}\right)\right\}\frac{\partial v_2}{\partial x} + \qquad (5)$$
$$+\left\{a_{2222}\left(1+e_{22}\right) + a_{2212}\left(\frac{e_{12}}{2} + \omega_3\right)\right\}\frac{\partial v_2}{\partial y} + b_*,$$

$$\frac{\partial \sigma_{xy}}{\partial t} = \left\{a_{1211}\left(1+e_{11}\right) + a_{1212}\left(\frac{e_{12}}{2} - \omega_3\right)\right\}\frac{\partial v_1}{\partial x} +$$
$$+\left\{a_{1212}\left(1+e_{11}\right) + a_{1222}\left(\frac{e_{21}}{2} - \omega_3\right)\right\}\frac{\partial v_1}{\partial y} +$$
$$+\left\{a_{1211}\left(\frac{e_{12}}{2} + \omega_3\right) + a_{1212}\left(1+e_{22}\right)\right\}\frac{\partial v_2}{\partial x} +$$
$$+\left\{a_{1222}\left(1+e_{22}\right) + a_{1212}\left(\frac{e_{12}}{2} + \omega_3\right)\right\}\frac{\partial v_2}{\partial y}.$$

В результаті повна система для цієї задачі має вигляд:

$$\frac{\partial \vec{W}}{\partial t} = A_1^N \frac{\partial \vec{W}}{\partial x} + A_2^N \frac{\partial \vec{W}}{\partial y} + \vec{B}. \qquad (6)$$

$$W_1 = \nu_x, W_2 = \nu_y, W_3 = \sigma_x, W_4 = \sigma_y, W_5 = \sigma_{xy}.$$

$$A_i^N = \begin{pmatrix} 0 & 0 & \frac{\delta_{i1}}{\rho} & 0 & \frac{\delta_{i2}}{\rho} \\ 0 & 0 & 0 & \frac{\delta_{i2}}{\rho} & \frac{\delta_{i1}}{\rho} \\ \alpha_{41} & \alpha_{42} & 0 & 0 & 0 \\ \alpha_{51} & \alpha_{52} & 0 & 0 & 0 \\ \alpha_{71} & \alpha_{72} & 0 & 0 & 0 \end{pmatrix},$$

$$\alpha_{41} = \left\{ a_{1111}\left(1+e_{11}\right) + a_{1112}\left(\frac{e_{12}}{2} - \omega_3\right) \right\} \delta_{i1} + \left\{ a_{1112}\left(1+e_{11}\right) + a_{1122}\left(\frac{e_{21}}{2} - \omega_3\right) \right\} \delta_{i2};$$

$$\alpha_{42} = \left\{ a_{1111}\left(\frac{e_{12}}{2} + \omega_3\right) + a_{1112}\left(1+e_{22}\right) \right\} \delta_{i1} + \left\{ a_{1112}\left(\frac{e_{12}}{2} + \omega_3\right) + a_{1122}\left(1+e_{ee}\right) \right\} \delta_{i2};$$

$$\alpha_{51} = \left\{ a_{2211}\left(1+e_{11}\right) + a_{2212}\left(\frac{e_{12}}{2} - \omega_3\right) \right\} \delta_{i1} + \left\{ a_{2212}\left(1+e_{11}\right) + a_{2222}\left(\frac{e_{21}}{2} - \omega_3\right) \right\} \delta_{i2};$$

$$\alpha_{52} = \left\{ a_{2211}\left(\frac{e_{12}}{2} + \omega_3\right) + a_{2212}\left(1+e_{22}\right) \right\} \delta_{i1} + \left\{ a_{2212}\left(\frac{e_{12}}{2} + \omega_3\right) + a_{2222}\left(1+e_{ee}\right) \right\} \delta_{i2};$$

$$\alpha_{71} = \left\{ a_{1211}\left(1+e_{11}\right) + a_{1212}\left(\frac{e_{12}}{2} - \omega_3\right) \right\} \delta_{i1} + \left\{ a_{1212}\left(1+e_{11}\right) + a_{1222}\left(\frac{e_{21}}{2} - \omega_3\right) \right\} \delta_{i2};$$

$$\alpha_{72} = \left\{ a_{1211}\left(\frac{e_{12}}{2} + \omega_3\right) + a_{1212}\left(1+e_{22}\right) \right\} \delta_{i1} + \left\{ a_{1212}\left(\frac{e_{12}}{2} + \omega_3\right) + a_{1222}\left(1+e_{ee}\right) \right\} \delta_{i2}.$$

Температуру в точках тіла визначимо як розв'язок рівняння теплопровідності:

$$\frac{\partial T}{\partial t} = a^2 \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) + W_*. \tag{7}$$

Через W_* позначена функція, що враховує тепло, яке виникає в тілі при фазовому переході з точок A в точки B на діаграмі матеріалу.

Введемо сітки за часом і по координатам:

$$\omega_t = \left\{ t_p; t_{p+1} = t_p + \tau; t_0 = 0; p = 0;1;2;\ldots \right\},$$

$$\omega_h = \left\{ \begin{array}{l} x_i; x_{i+1} = x_i + h_1; x_0 = 0; h_1 = \dfrac{H}{n}; i = 0;1;2;\ldots;n \\ \\ y_j; y_{j+1} = y_j + h_2; y_0 = 0; h_2 = \dfrac{L}{m}; j = 0;1;2;\ldots;m \end{array} \right\}. \tag{8}$$

В якості розрахункової візьмемо наступну формулу ітераційної процедури і перепишемо її так

$$\left(\overline{w}^{p+1}\right)_{m+1} = \left[\overline{w}^{p} + \tau\left(\overline{L}_{12} + \overline{B}^{p}\right)\right]_{m}, \tag{9}$$
$$\overline{L}_{12} = \Lambda_1\left(\alpha_{11}\overline{w}^{p+1} + \alpha_{12}\overline{w}^{p}\right) + \Lambda_2\left(\alpha_{21}\overline{w}^{p+1} + \alpha_{22}\overline{w}^{p}\right).$$

Температурне поле в полосі визначаємо за допомогою двовимірного варіанта ітераційної процедури:

$$\left(T^{p+1}\right)_{m+1} = \left\{T^{p} + \tau M_{12} + W_{*}^{p}\right\}_{m},$$
$$M_{12} = P_1\left(\alpha_{11}T^{p+1} + \alpha_{12}T^{p}\right) + P_2\left(\alpha_{21}T^{p+1} + \alpha_{22}T^{p}\right). \tag{10}$$

Зазначимо, що після визначення основних невідомих, окремо розшукуються шляхом інтегрування за часом решта невідомих функцій:

$$\begin{aligned}\frac{\partial\sigma_{zz}}{\partial t} &= \left\{a_{3311}\left(1+e_{11}\right) + a_{3312}\left(\frac{e_{12}}{2} - \omega_3\right)\right\}\frac{\partial v_1}{\partial x} + \\ &+ \left\{a_{3312}\left(1+e_{11}\right) + a_{3322}\left(\frac{e_{21}}{2} - \omega_3\right)\right\}\frac{\partial v_1}{\partial y} + \\ &+ \left\{a_{3311}\left(\frac{e_{12}}{2} + \omega_3\right) + a_{3312}\left(1+e_{22}\right)\right\}\frac{\partial v_2}{\partial x} + \\ &+ \left\{a_{3322}\left(1+e_{22}\right) + a_{3312}\left(\frac{e_{12}}{2} + \omega_3\right)\right\}\frac{\partial v_2}{\partial y} + b_*,\end{aligned} \tag{10}$$

$$\frac{\partial\varepsilon_x}{\partial t} = \left(1+e_{11}\right)\frac{\partial v_x}{\partial x} + \left(\frac{e_{12}}{2} + \omega_3\right)\frac{\partial v_y}{\partial x}; \frac{\partial\varepsilon_y}{\partial t} = \left(\frac{e_{21}}{2} - \omega_3\right)\frac{\partial v_x}{\partial y} + \left(1+e_{22}\right)\frac{\partial v_y}{\partial y};$$

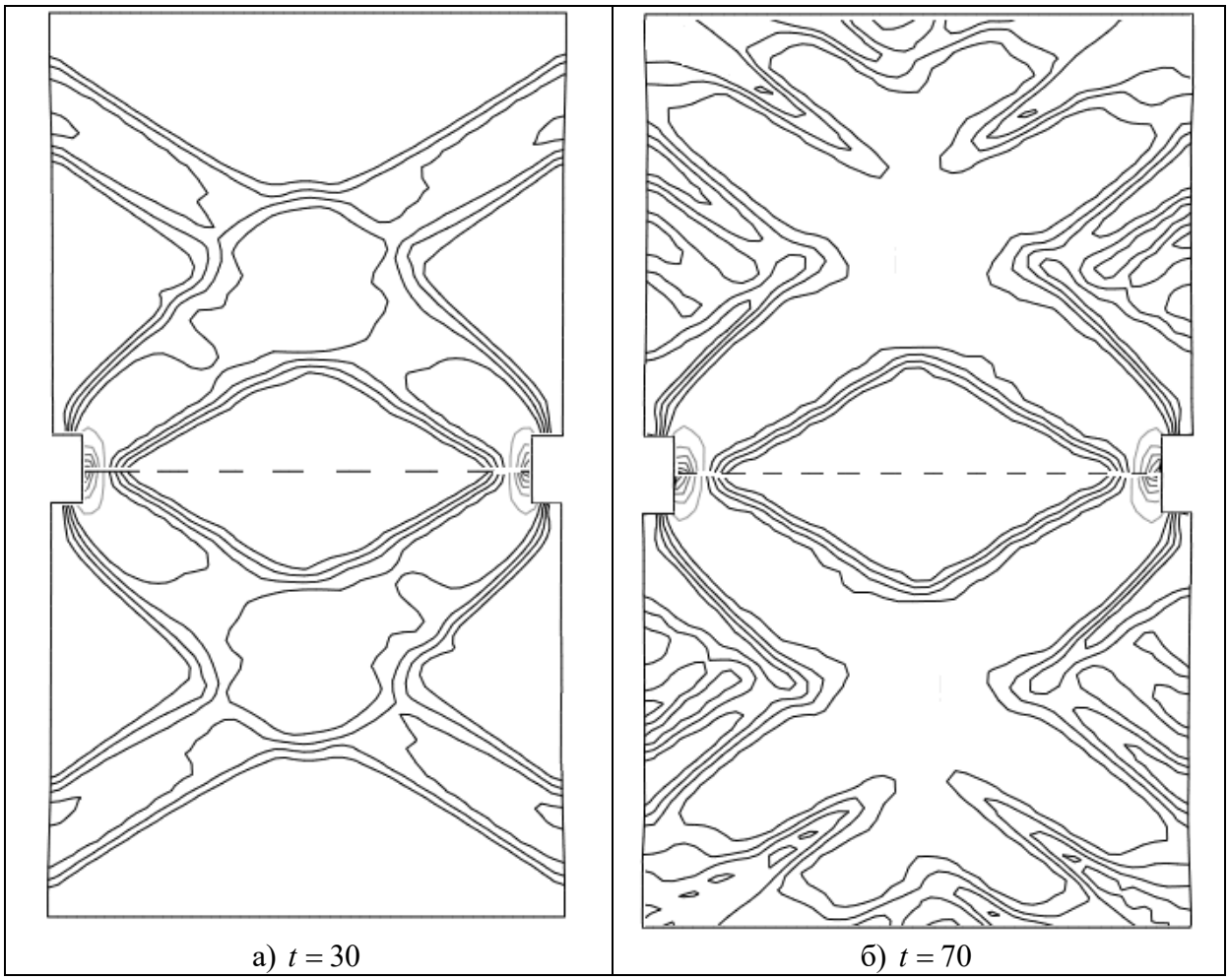
$$\frac{\partial\varepsilon_{xy}}{\partial t} = \left(1+e_{22}\right)\frac{\partial v_y}{\partial x} + \left(1+e_{11}\right)\frac{\partial v_x}{\partial y} + \left(\frac{e_{12}}{2} - \omega_3\right)\frac{\partial v_x}{\partial x} + \left(\frac{e_{12}}{2} + \omega_3\right)\frac{\partial v_y}{\partial y}.$$

Розглянемо числові результати, отримані для полоси при $L = 2H$, $h_1 = h_2 = H/10, \tau = 0{,}01$.

На рисунках 2 і 3 для відповідних моментів часу показано поле інтенсивності деформації , яке має місце в полосі при різних видах навантаження.

На рисунку 2 а) показано поле інтенсивності деформації при симетричному навантаженні (рисукок 2 а)) для безрозмірного моменту часу ($t = 30 = 3000\tau$). В момент $t = 50$ розтягування полоси зупинялось, але в тілі і далі йшов перерозподіл навантаження в режимі установлення. На рисунку 2 б) показано поле інтенсивності деформації (рисукок 1 а)) для $t = 70$.

На рисунку 3 а) показано поле інтенсивності деформації при несиметричному навантаженні (рисунок 3 б)) для безрозмірного моменту часу ($t = 30 = 3000\tau$). В момент $t = 50$ розтягування полоси зупинялось, але в тілі і далі йшов перерозподіл навантаження в режимі установлення. На рисунку 3 б) показано поле інтенсивності деформації (рисунок 1 а)) для $t = 70$.



а) $t = 30$ б) $t = 70$

Рисунок 2. Поле інтенсивності деформації при симетричному навантаженні

Висновки

Моделювання поведінки з псевдо-пружно-пластичного матеріалу при великих пластичних деформаціях вимагає застосування нелійних математичних моделей, які з більшою точністю могли б описати та спрогнозувати поведінку такого тіла. В роботі проведено моделювання поведінки локально навантаженої послабленої полоси з псевдо-пружно-пластичного матеріалу при її нестаціонарному навантаженні. Авторами, до розвязання вище вказаної задачі, застосовано нелінійну феноменологічну модель матеріалу, яка дозволяє описати ряд експерементальних даннних на різних зразках при різних умовах. Проведено порівняння результатів отриманих в геометрично лінійній і нелінійній постановках при великих пластичних деформаціях. Чисельно порівняно поля інтенсивності при симетричному і несиметричному навантаженні Встановлено, що при пластичних деформаціях до 6% (малі деформації) розбіжність результатів в точках локалізації деформації не перевищує 5%. При збільшенні значень пластичної деформації (більше 7%, великі деформації) розбіжність результатів може значно зростати і в околі можливого створення шийки досягати 20%.

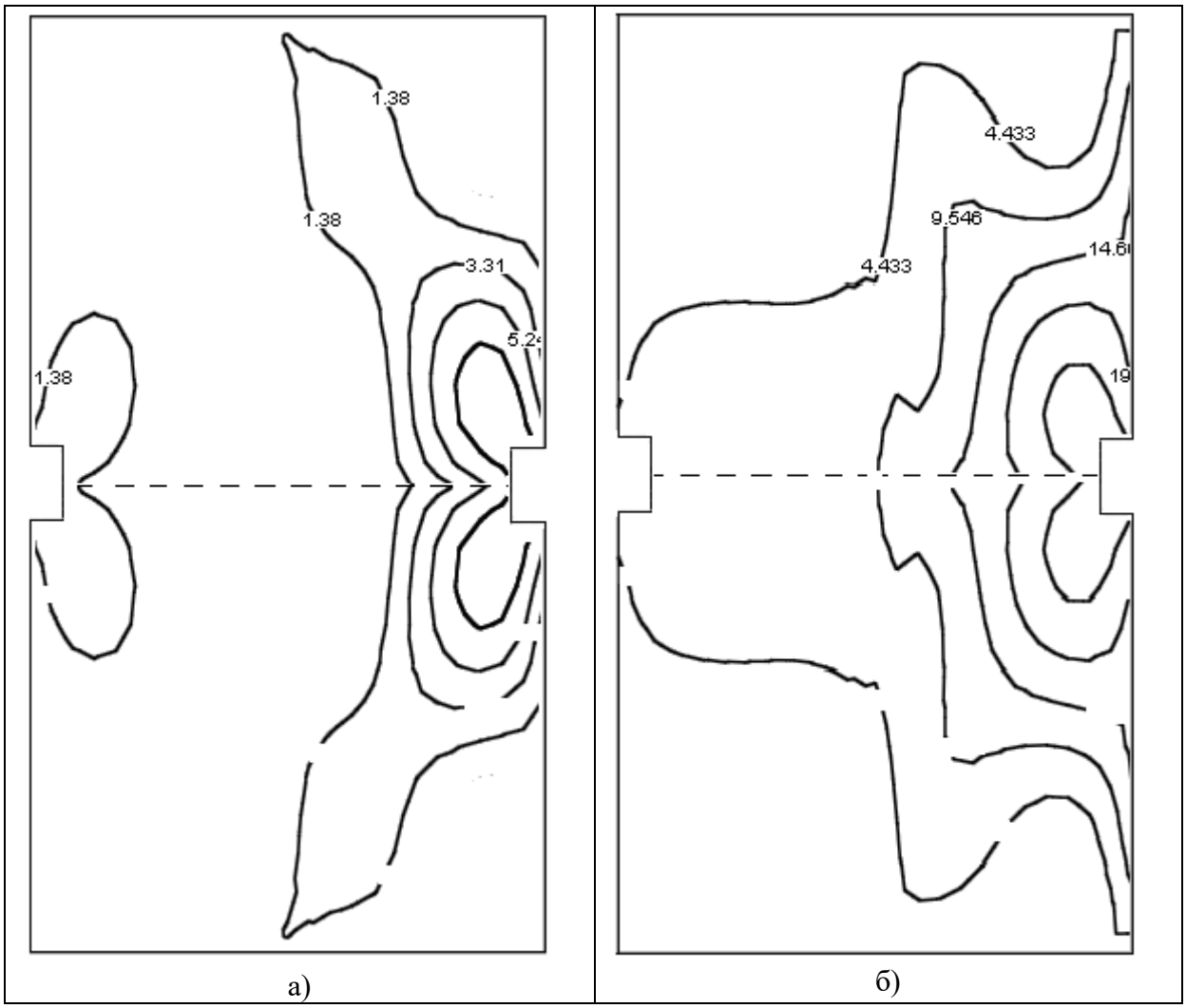


Рисунок 3 Поле інтенсивності деформації при несиметричному навантаженні

Список використаної літератури:

1. Domichev K. Modeling the behavior of the body with pseudo-elastic-plastic material at non-stationary loading / K. Domichev, P. Steblyanko, A. Petrov // Металофізика та новітні технології, Інститут металофізики ім. Г. В. Курдюмова НАН України, 2021 – Том 43, випуск 1 – с. 107-128.
2. Стеблянко П.А. Методы расщепления в пространственных задачах теории пластичности /П.А. Стеблянко – Киев: Наукова думка, 1998. – 304 с.
3. Abeyaratne R., Knowles J.K. Evolution of phase transitions /R. Abeyaratne, J.K. Knowles– Cambridge University Press, 2006. – 258 p.
4. Shaw J. A., Kyriakides S. Thermomechanical aspects of NiTi. / J. A Shaw, S. Kyriakides – Mechanics and Physics of Solids, 1995. – No 43, p.1243-1281.
5. Shaw, J. A., Kyriakides, S. On the nucleation and propagation of phase transformation fronts in a NiTi alloy / J. A Shaw, S. Kyriakides – Acta Materialia,1997. – No 45, p. 683-700.
6. Petrov A. Development of the method with enhanced accuracy for solving problems from the theory of thermo-psevdoelastic-plasticity / A. Petrov, Yu. Chernyakov, P. Steblyanko, K. Demichev, V. Haydurov – Eastern-European Journal of Enterprise Technologies. 2018. Vol. 4/7 (94). P. 25-33.
7. Дьомічев К.Е. Нелінійна феноменологічна модель поведінки функціонально-неоднорідних матеріалів / К.Е. Дьомічев, П.О. Стеблянко, О.Д. Петров // Вісник Черкаського національного університету ім. Б. Хмельницько. Серія Прикладна математика. Інформатика №1(1). – 2020–С. 4-12.
8. Steblyanko P. Phenomenological Model of Pseudo-Elastic-Plastic Material Under Nonstationary Combining Loading / P. Steblyanko, Y. Chernyakov, A. Petrov, V. Loboda – Structural Integrity, Volume 8, Theoretical, Applied and Experimental Mechanics, Springer Verlag, 2019. – P. 205-208.

9. Петров О.Д. Комп'ютерне моделювання поведінки стриженя з трилінійного двофазного матеріалу при розтягуванні / О.Д. Петров – Інформаційні технології та комп'ютерне моделювання; матеріали статей МНПК (ISBN 978-617-7468-26-3) – 2018. – Івано-Франківськ. – 2018. – С. 234-237.

References:

1. Domichev K., Steblyanko P., Petrov A. (2021) Modeling the behavior of the body with pseudo-elastic-plastic material at non-stationary loading – Metallophysics and new technologies, Institute of Metal Physics, 107-128.
2. Steblyanko P.A. (1998). Methods of decomposition in space problems of the theory of plasticity. – Kyiv: Naukova dumka [in Russian].
3. Abeyaratne R., Knowles J.K. (2006) Evolution of phase transitions. – Cambridge University Press
4. Shaw, J.A., Kyriakides S. (1995) Thermomechanical aspects of NiTi. J. Mechanics and Physics of Solids 43, 1243-1281
5. Shaw, J.A., Kyriakides S. (1997) On the nucleation and propagation of phase transformation fronts in a NiTi alloy. Acta Materialia
6. Petrov A., Chernyakov Yu., Steblyanko P. , Demichev K. , Haydurov V. (2018). Development of the method with enhanced accuracy for solving problems from the theory of thermo-psevdoelastic-plasticity. Eastern-European Journal of Enterprise Technologies.
7. Domichev K., Steblyanko P., Petrov A. (2020) Visnyk of Cherkasy National University named after B. Khmelnytsky. Applied Mathematics Series. Computer Science, 4-12 [in Ukraine]
8. Steblyanko P. Y. Chernyakov, A. Petrov, V. Loboda (2019) Phenomenological Model of Pseudo-Elastic-Plastic Material Under Nonstationary Combining Loading Theoretical, Applied and Experimental Mechanics, Springer Verlag.
9. Petrov A.D. (2018) Computer modeling of the shearing behavior of trilinear two-phase material during stretching. Information Technologies and Computer Modeling; materials of articles Annual Intern. Scientific [in Ukraine]

STEBLYANKO Pavel,
Doctor of Physics and Mathematics, Professor, Professor of Department of Cybersecurity and Information Technology, University of Customs and Finance, Ukraine
DOMICHEV Konstantin,
Candidate of Technical Sciences, Associate Professor, Professor of Department of Computer Science, Kyiv International University, Ukraine
PETROV Alexander,
Doctor of Philosophy (PhD), Junior Researcher, Department of Theoretical and Computer Mechanics Dnieper National University Named After Oles Honchar, Ukraine

SIMULATION OF LOCALLY LOADED STRIP BEHAVIOR FROM PSEUDO-ELASTIC-PLASTIC MATERIAL UNDER LARGE PLASTIC DEFORMATIONS

Summary. Introduction. *Currently, a number of models are known to describe the thermomechanical behavior of functionally inhomogeneous materials, in particular alloys with shape memory (SPF) [3; 4]. Most of them are based on classical ideas, ie aim to directly describe the experimental data obtained on different macrosamples under simple and complex loads. However, as established in experimental studies, the behavior of the material at a point in the body in the General case may be different from the behavior of the sample as a whole [5]. In the considered problems the numerical procedure of calculation of the diagram of material which represents a curve which surrounds a family of diagrams of the material constructed for certain laws of change of speed of a front of rupture of deformations is developed.*

Purpose. *The aim of the work is to apply a nonlinear phenomenological model, which describes the properties of alloys with shape memory and thermo-pseudo-plastic behavior (TPPM) of the material at the point to the problem of modeling a locally loaded strip of pseudo-elastic-plastic material at large plastic deformations. Compare the results of modeling in geometrically linear and nonlinear formulations for large plastic deformations.*

Results. *The behavior of the locally loaded weakened strip of pseudo-elastic-plastic material under its non-stationary loading is modeled in the work. The authors used a nonlinear phenomenological model of the material to solve the above problem, which allows to describe a number of experimental data on different samples under different conditions. A comparison of the results obtained in geometrically linear and nonlinear formulations with large plastic deformations.*

Conclusion. *Modeling the behavior of pseudo-elastic-plastic material with large plastic deformations requires the use of nonlinear mathematical models that could more accurately describe and predict the behavior of such a body. The behavior of the locally loaded weakened strip of pseudo-elastic-plastic material under its non-stationary loading is modeled in the work. The authors used a nonlinear phenomenological model of the material to solve the above problem, which allows to describe a number of experimental data on different samples under different conditions. A comparison of the results obtained in geometrically linear and nonlinear formulations with large plastic deformations. Numerically compared intensity fields at symmetric and asymmetric loading It is established that at plastic deformations up to 6% (small deformations) the discrepancy of results in points of localization of deformation does not exceed 5%. With increasing values of plastic deformation (more than 7%, large deformations), the discrepancy of the results can increase significantly and in the vicinity of the possible creation of the neck to reach 20%.*

Keywords: *phenomenological model, nonlinear model of material, materials with memory of forms, thermo-pseudo-plasticity, large plastic deformations.*

Одержано редакцією 07.10.2021 р.
Прийнято до публікації 24.11.2021 р.

UDC 519.6, 556.54

DOI 10.31651/2076-5886-2021-1-13-22

PACS 43.28.-g;92.60.Sz

LUKIANOV Petro,
PhD in Physics and Mathematics, Senior Researcher, Associate Professor, Department of Aviation and Rocket Engineering, Institute of Aerospace Ingineering, NTTU “KPI”
e-mail: luk_ptr@yahoo.com
ORCID: 0000-0002-7584-1491

NUMERICAL-ANALYTICAL METHOD FOR THE PROBLEMS OF ENVIROMENTAL SAFETY

In problems of modeling pollution of water and atmospheric resources of the Earth’s ecosystems, models of potential flow are often used. Recently, considerable attention has also been paid to protecting metropolis from atmospheric pollution with acoustic noise. Very often, without focusing on local flow features, the Laplace equation is used to describe the potential equation motion of a fluid. Acoustic problems are modeled based on the Helmholtz equation.

In the work presented below, the features of the numerical-analytical method for the Helmholtz and Laplace equations are considered. The sound potential is a rapidly oscillating function given at the boundary of the computational domain. In addition, the numerically-analytical method peculiarities are presented for the Laplace equation that uses the expansion of the boundary condition in a Fourier series in eigenvalues of the Sturm-Liouville problem. Despite the fact that the data presented in the work were obtained for canonical domains, the scheme of the method implies the possibility of using it for domains with an arbitrary curvilinear boundary. The numerical-analytical method proposed in the paper allows for low computational costs to solve numerically the problems for the Laplace equation and the acoustic equations. The results of the research conducted can be used as new information technology to address the environmental security challenges of water and air resources.

Keywords: *Computational modeling, Numerical method, Geo-environmental monitoring.*

Introduction

One of the major problems of today is the problem of preventing pollution of the Earth's water and air resources. It is directly related to the geo-environmental monitoring of the planet. To prevent global man-made disasters, scientists are developing mathematical models that describe the processes of occurrence and spread of pollution in aquatic environments.

There is also the problem of protecting the environment from noise pollution by helicopters. Recently, considerable attention has been paid to the development of information technologies in the field of environmental safety. In particular, much attention is paid to the study of processes of pollution of water resources and the atmosphere, as well as to the acoustic pollution (noisiness) of megapolices. In civil aviation [1], in the operation of helicopters, very often unforeseen situations occur that result in local environmental disasters that lead to undesired environmental pollution.

It is known that in most situations, the main equation for modeling fluid motion is the Laplace equation [2], [3]. Modeling of aerodynamic noise, helicopter noise, is performed on the basis of the equation of propagation of small perturbations from a thin wing [4]. Numerical methods and algorithms for its solution are constantly being improved. Numerical circuits are being searched for, allowing smaller computer resources to achieve the desired result. In this paper, we consider a numerical method for numerical modeling these processes. The numerical-analytical method is an implicit analogue of the finite-difference method: in the finite-difference method, all derivatives are a priori expressed in terms of values in the calculation nodes, and in the numerical-analytical method they are expressed implicitly directly during the solution of a specific differential equation, a boundary-value problem. Thus, this method allows you to take into account the specifics of the differential operator the behavior of the function on the boundary.

In a number of problems of hydromechanics, one has to deal with the Laplace equation:

$$\Delta f = 0, \tag{1}$$

where $f \in C^2(A)$, $A \subset R^2$.

If the problem is solved in the canonical domain, for example, in a rectangular 2-dimensional domain, then the desired solution based on the Fourier method is represented as a trigonometric series in eigenvalues functions. A similar situation holds for the Helmholtz equation. The Laplace and Helmholtz equations are to some extent similar in structure: on the basis of the classification existing in mathematical physics, both equations are elliptic equations. They are particular cases of the general Poisson equation. The use of the numerical - analytical method for the Helmholtz equation was considered [5] for cases of moderate values of the wave number. In this paper, we consider the behavior of this circuit for large wave number.

The aim of this work is to study and develop the application of the numerical-analytical method for boundary-value problems for which a rapidly oscillating function is given at the boundary. The study is based on the Helmholtz equation. The numerical solution of the Laplace equation is constructed in the same way as the Helmholtz equation, but using some modification.

1. Application of the method for the Helmholtz equation. The case of a rapidly oscillating function at the boundary

In [6], a scheme for applying the numerical-analytical method for the Helmholtz equation was considered using the example of a two-dimensional domain $[a,b]\times[c,d]$:

$$\Delta f + k^2 f = 0 \tag{2}$$

$$f_x(x,y) = 0,\ x = 0,\ f_x(x,y) = 0,\ x = a; \tag{3}$$

$$-f_y(x,y) = V_0,\ y = 0,\ -f_y(x,y) = 0,\ y = b. \tag{4}$$

In this case, the sound potential, following the idea of the method, is represented in the form:

$$f(y) = f(y_0) + f_y(y_0)(y - y_0) + \frac{1}{2!} f_{yy}(y_0)(y - y_0)^2 + \\ + \frac{1}{3!} f_{yyy}(y_0)(y - y_0)^3 + o((y - y_0)^3), \quad (5)$$

where $f(y_0),\ f_y(y_0),\ f_{yy}(y_0),\ f_{yyy}(y_0)$ unknown row expansion coefficients. We denote

$$x_1 = f(y_0),\ \ x_2 = f_y(y_0),\ x_3 = f_{yy}(y_0),\ x_4 = f_{yyy}(y_0).$$

Applying a 4-point scheme, we obtain the following recurrence relations [5]:

$$x_3 = \frac{(-0.5f(0) + 2f(\Delta) - 2.5f(2\Delta)) \cdot k^2}{0.5\Delta^2 k^2 + 1} \quad (6)$$

$$x_4 = \frac{(2f(0) - 3f(\Delta) - 3\Delta^2 x_3) \cdot k^2 - x_3}{-5\Delta^3 k^2} \quad (7)$$

$$x_2 = -\frac{f(0) - f(\Delta) - 2.5x_3\Delta^2 + \frac{19}{6}\Delta^3 x_4}{\Delta} \quad (8)$$

$$x_1 = f(0) + 3\Delta \cdot x_2 - 4.5\Delta^2 \cdot x_3 + 4.5\Delta^3 \cdot x_4. \quad (9)$$

Approximations of the third order are quite enough for this method so that the calculated interval $[0;1]$, broken with a step $\Delta = 0,02$, already gives us order accuracy $10^{-3} - 10^{-4}$. However, the range of wave numbers k was $0 \le k \le 8$. In this case, the calculation was performed for a little more than one wavelength. How will the method behave if 3 or 5 wavelengths are placed on the calculation interval? In this problem, this corresponds to order wavenumbers $k = 20, 30$. The numerical calculation showed that the step $\Delta = 0,02$ here will be too large: if you do not change it, then, starting with $y \approx 0,3$, a solution shift is observed according to this numerical method in comparison with the analytical solution by the Fourier method.

If we grind the step $\Delta = 1/120$, then for $k = 20$ (a little more than 3 wavelengths) we get results that coincide with the analytical solution with good accuracy. In Fig. 1, the dashed curve corresponds to the solution according to the numerical analytical method, the solid line is the exact solution according to the Fourier method. For $k = 30$ (about 5 wavelengths) the step $\Delta = 1/120$ must be ground to $\Delta = 1/180$. Nevertheless, we see that there is a slight deviation at the end of the calculation interval of the numerical solution according to the method from the analytical solution.

In terms of accuracy, the solution here is slightly inferior to the solution for small values k [5].

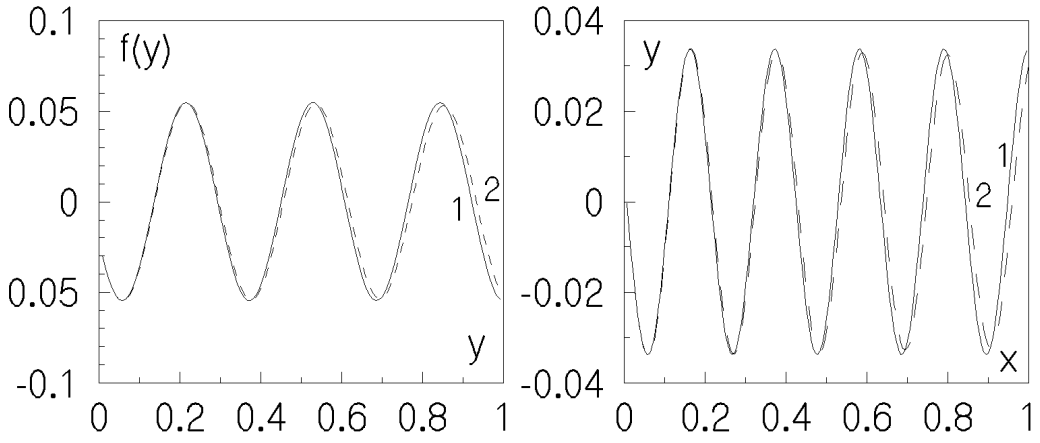


Fig.1 Sound pressure potential for wave numbers k=20, 30.

2. Application of the method to the Laplace equation

Let we have a rectangular region $[-a,a]\times[-h,h]$ symmetric with respect to the origin in which the Laplace equation (1) is solved. At the boundary of the region, the following boundary conditions are specified:

$$f = 0, \quad x = \pm a\,, \tag{10}$$

$$f = A_0 \cos\frac{(2n+1)}{2a}\pi x, \quad y = \pm h\,, \tag{11}$$

where A_0 – is some constant.

If we assume that at the boundary $y = \pm h$ the function f depends only on x, then the Laplace equation takes the form:

$$f_{xx} = 0\,. \tag{12}$$

And this means that $f = C_1 x + C_2$, where C_1, C_2 are some integration constants of equation (12). But this contradicts the fact that the function $f = A_0 \cdot \cos\frac{(2n+1)\pi}{2a}x$ is at the given boundary. Therefore, the direct application of the method by analogy with the Helmholtz equation does not work.

It is possible to solve the arisen problem if we know what structure the desired solution should be. Based on the Fourier method, it is easy to obtain:

$$f(x,y) = \sum_{n=0}^{\infty} A_n \cos\lambda_n x \cdot (C_n e^{-\lambda_n y} + D_n e^{\lambda_n y})\,, \tag{13}$$

where $\lambda_n = \frac{2n+1}{2a}\pi$.

However, let us pay an attention to expression (13). If we differentiate both sides of (13) by a variable y, we obtain:

$$f_{yy} = \lambda_n^2 f\,. \tag{14}$$

Let us return to the numerical-analytical method. Replace in the Laplace equation f_{yy} by $\lambda_n^2 f$:

$$f_{xx} + \lambda_n^2 f = 0. \tag{15}$$

In appearance, this equation exactly matches the Helmholtz equation, with one exception, which is k^2 instead of λ_n^2 here. Therefore, we can further use the scheme for applying the numerical-analytical method for the Helmholtz equation (2), replacing k^2 by λ_n^2:

$$x_3 = \frac{(-0.5f(0) + 2f(\Delta) - 2.5f(2\Delta)) \cdot \lambda_n{}^2}{0.5\Delta^2 \lambda_n{}^2 + 1}, \tag{16}$$

$$x_4 = \frac{(2f(0) - 3f(\Delta) - 3\Delta^2 x_3) \cdot \lambda_n{}^2 - x_3}{-5\Delta^3 \lambda_n{}^2}, \tag{17}$$

$$x_2 = -\frac{f(0) - f(\Delta) - 2.5x_3\Delta^2 + \frac{19}{6}\Delta^3 x_4}{\Delta}, \tag{18}$$

$$x_1 = f(0) + 3\Delta \cdot x_2 - 4.5\Delta^2 \cdot x_3 + 4.5\Delta^3 \cdot x_4. \tag{19}$$

The numerical calculation showed the same degree of convergence of the solution to the analytical solution (Fig. 2). The use of substitution (14) allows us to generalize the scheme of using the numerical-analytical method for the Laplace equation and a function ψ arbitrarily defined on the boundary.

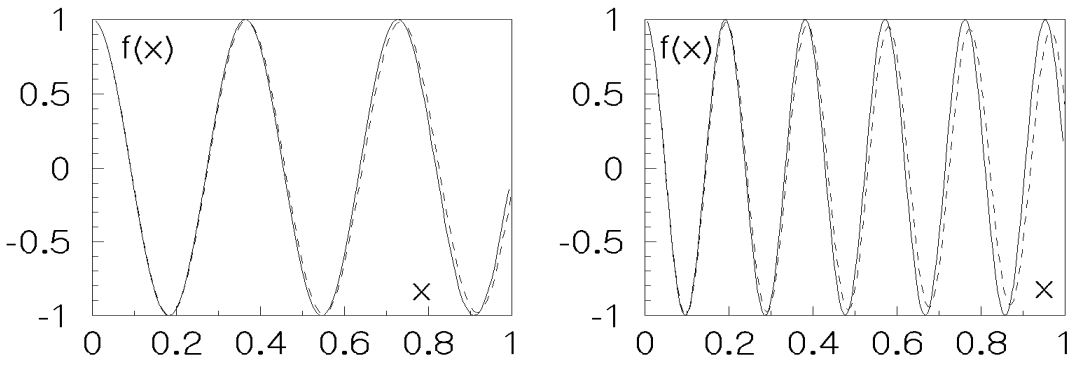


Fig. 2. The flow potential for $n = 5, 10$

3. A generalization of the application of the numerical analytical method for the Laplace equation

Let not one mode (11), but some arbitrary function ψ be given on the boundary of the region $[-a, a] \times [-h, h]$:

$$f(x, \pm h) = \psi, \tag{20}$$

which satisfies the Laplace equation. The boundary condition (10) remains the same. Applying the scheme of the Fourier method, we obtain (13). The set of eigenvalues λ_n depends on the type of boundary condition. For a fixed value $y = \pm h$ at the boundary, the two-dimensional function $f(x, y)$ is transformed into the usual trigonometric series:

$$\psi(x) = \sum_{n=0}^{\infty} A_n^* \cos \lambda_n x = \psi_0 + \psi_1 + \ldots + \psi_n, \quad A_n^* = C_n e^{-\lambda_n (\pm h)} + D_n e^{\lambda_n (\pm h)}. \tag{21}$$

Expression (21) is a trigonometric series composed of eigenfunctions obtained based on the solution of the Sturm-Liouville problem. It is not difficult to carry out rationing in such a way as to obtain an orthonormal closed system of functions. Therefore, according to the Riesz-Fisher and Steklov-Parseval theorems [6], expression (22) is a unique representation of the function in the form of a Fourier series in terms of the eigenfunctions of the Sturm-Liouville problem.

Now back to the function f. From the theory of linear differential equations it is known that a linear combination of individual solutions of a linear differential equation is also a solution to this differential equation. The Laplace equation is linear, therefore, if we present the desired solution in the form of a superposition of n solutions f_i:

$$f = \sum_{i=0}^{n} f_i, \tag{22}$$

then this superposition for each i will also be a solution to the Laplace equation:

$$\Delta f_i = 0. \tag{23}$$

This statement allows us to break down the original task for the function f on the $n+1$ task:

$$\begin{aligned} &\Delta f_i = 0, \\ &f_i = 0, \quad x = \pm a, \\ &f_i = \psi_i, \quad y = \pm h, \quad i = 0, n. \end{aligned} \tag{24}$$

It is easy to see that the obtained boundary-value problem (24) for each fixed value i coincides with the problem described above for a strongly oscillating function f, for which the scheme for applying the numerical-analytical method has already been considered.

Comment. Such a simple implementation of the scheme of the numerical analytical method is possible only for the cases of linear equations considered in the paper. If we take the quasi-linear equation [7], then the analytical representation of the solution in the a recursive form will not work: the nonlinear system of equations that is formed as a result of applying the numerical-analytical method has to be solved numerically.

4. An example of the application of the method for quasi-linear equations

This section provides a solution to the problem of transonic flow around a plane wing profile. The equation describing the flow around a wing profile is a quasilinear equation:

$$[1-\frac{1}{M_1^2}+\varepsilon\cdot(\gamma+1)f_\xi]f_{\xi\xi}-\frac{\lambda^2}{M_1^2}f_{\eta\eta}=0, \tag{25}$$

in dimensionless coordinates $\xi = x/c,\ \eta = \lambda y$, where $x,\ y$ – the Cartesian coordinates along and across the section of the blade, respectively. The coordinate η displays the surface shape of the cross section of the blade. Taking into account accepted notation, the boundary condition on the surface of the blade is written in the form:

$$\eta = g(\xi),\ 0<\xi<1; f_\eta = \delta g_\xi,$$

We turn to the application of the method. We believe that $f \in C^2([0;1]\times[0;1])$. Then it can be represented as a Taylor series in a neighborhood of an arbitrary point (ξ_i,η_i):

$$\begin{aligned} f(\xi_i,\eta_i) &= f(\xi_0,\eta_0)+f_\xi(\xi_0,\eta_0)(\xi_i-\xi_0)+f_\eta(\xi_0,\eta_0)(\eta_i-\eta_0)+\\ &+\frac{1}{2}[f_{\xi\xi}(\xi_0,\eta_0)(\xi_i-\xi_0)^2+2f_{\xi\eta}(\xi_0,\eta_0)(\xi_i-\xi_0)(\eta_i-\eta_0)+f_{\eta\eta}(\xi_0,\eta_0)(\eta_i-\eta_0)^2]+\\ &+\ o(\max\{(\xi_i-\xi_0)^2,(\xi_i-\xi_0)(\eta_i-\eta_0),(\eta_i-\eta_0)^2\}),\ i=1,5. \end{aligned} \tag{26}$$

Here the point (ξ_0,η_0) is some fixed point of profile, close to (ξ_i,η_i) (Fig.3).

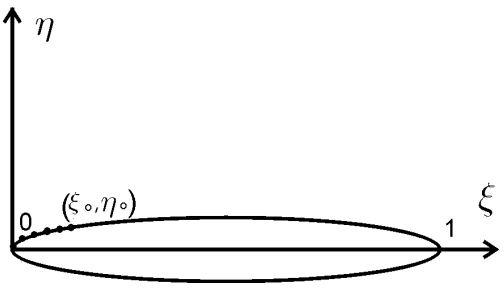


Fig. 3. Calculation pattern on the surface of the wing profile

Calculations will be carried out along the upper edge of the wing. We select five points at the beginning of the profile on the left with coordinates (ξ_i,η_i), $i=1,5$. We regard that (ξ_0,η_0) is the six point, that is situated on the upper boundary of wing just after the first five points. System (4) consists of five equations with six unknowns $f(\xi_0,\eta_0), f_\xi(\xi_0,\eta_0)$, $f(\xi_0,\eta_0), f_\xi(\xi_0,\eta_0), f_{\xi\eta}(\xi_0,\eta_0), f_{\eta\eta}(\xi_0,\eta_0)$.

As (ξ_0,η_0) is an arbitrary point, then equation (1) should be performed for it automatically, i.e.

$$[1-\frac{1}{M_1^2}+\varepsilon\cdot(\gamma+1)f_\xi(\xi_0,\eta_0)]f_{\xi\xi}(\xi_0,\eta_0)-\frac{\lambda^2}{M_1^2}f_{\eta\eta}(\xi_0,\eta_0)=0 \tag{27}$$

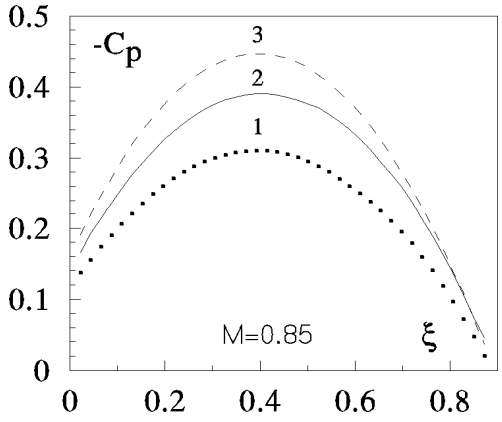


Fig. 4a

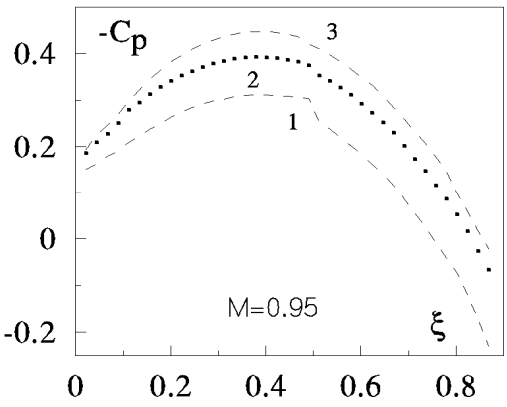


Fig. 4b

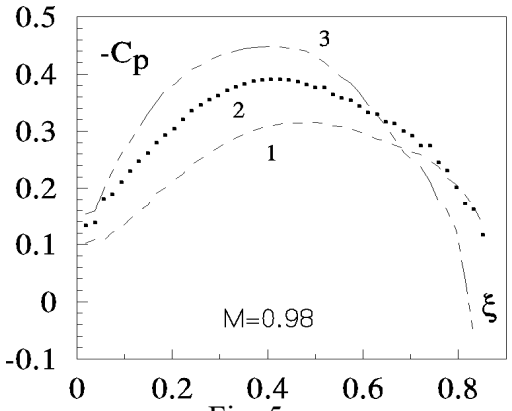


Fig. 5a

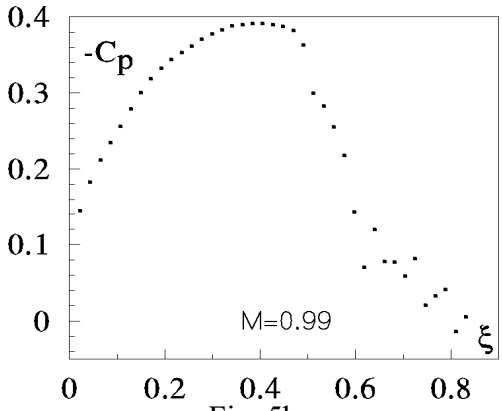


Fig. 5b

The system of equations (26)-(27) is closed relative to the indicated unknowns if we omit the second-order quantities of smallness. The data of calculating the pressure coefficient on the surface of the blade are shown in Fig. 4-5.

Close agreement is obtained with the calculated data of [8] in the case of a helicopter rotor blade remote from the end of the flow around the blade. At $M > 0,9$ the nucleation of a weak shock wave is observed (Fig. 3), which is realized in the form of a small jump literally immediately after $\xi = 0.5$.

As you approach $M = 1$ ($M = 0.98; 0.99$) it is noticeable instable behavior of C_p (fig.5). This is the cause of the instability of the flow itself. Since equation (25) changes type from elliptic to hyperbolic on a shock wave, it affects the numerical algorithm, where the series expansion was assumed to be continuous. Here, the method reveals an interesting regularity of the flow: the so-called “spurious vorticity”. In this zone, the method works stably even with unstable flow behavior.

The study of shock wave arising is very import in safety of flight air vehicles. So the numerical-analytical method can be used as an information technology for environment.

Conclusions

1. The paper considers the algorithm for applying the numerical-analytical method for the Laplace and Helmholtz equations for the case of strongly oscillating functions.

2. A comparison is made of the numerical convergence of the method with an analytical solution.

3. The features of setting the grid domain, the choice of the calculation step for the optimal use of the method scheme were studied.

4. The behavior of the helicopter blade in the mode of nucleation of a shock wave, the occurrence of flutter of the blade is studied. The calculation data can be used in the design of information flight control systems for aircraft, helicopters in order to prevent air crashes - local environmental incidents.

5. The numerical-analytical method proposed in the paper allows for low computational costs to solve numerically the problems for the Laplace equation and the acoustic equations. The results of the research conducted can be used as new information technology to address the environmental security challenges of water and air resources.

References:

1. *ICAO organization.* Manual of the Regulation of International Air Transport. – Third Edition, 2018.
2. *Ruixun Lai, Min Wang, Yang Zhang* Method based on the Laplace equations to reconstract the river terrain for two-dimensional hydrodynamic numerical modeling // Computer&Geociences. – Volume 111, February 2018. – P. 26-38.
3. *Beliaev M.M., Kirichenko P.S.* CFD modeling of the sea pollution in the case of mine water discharge // Geotechnical mechanics. – Issue 104, 2012. – P. 259-263.
4. *Lukianov P.V.* Unsteady propagation of small disturbances from a thin wing: The near and far field // Acoustic bulletin, 2009. – V. 12, N.3. –P. 41-55.
5. *Lukianov P.V.* Using of numerical-analytical method for acoustic problems solution. // "Consonant 2005", Acoustical symposium. Kiev-2005, Inst. Hydromechanics, 27-29 September. – P. 225-230.
6. *Vladimirov V.S.* Equation of mathematical physics. – M.: Science, 1981. – 512 p.
7. *Lukianov P.V.* On one numerically-analytical approach to solving of a problem on sound generation by a thin wing. Part I. General schematic of application to planer stationary problem. // Acoustic bulletin, 2011. – V.14, N3. – P. 46-52.
8. *Caradonna F.X., Isom M.P.* Subsonic and Transonic Potential Flow over Helicopter Rotor Blades. // AIAA Journal, 1972. – V.10, N.12. – P. 1606-1612.

Список використаної літератури:

1. *ICAO organization.* Manual of the Regulation of International Air Transport. – Third Edition, 2018.
2. *Ruixun Lai, Min Wang, Yang Zhang* Method based on the Laplace equations to reconstract the river terrain for two-dimensional hydrodynamic numerical modeling // Computer&Geociences. – Volume 111, February 2018. – P. 26-38.
3. Беляэв Н.Н. , Кириченко П.С. CFD моделирование загрязнения моря при сбросе шахтных вод // Геотехническая механика. Межвед.сб.науч.тр.-Днепропетровск: ИГТМ НАНУ, 2012.-Вып.104.С-259-263.
4. Лукьянов П.В. Нестационарное распространение малых возмущений от тонкого крыла: ближнее и дальнее поле //Акустичний вісник, 2009. – V. 12, N.3. –P. 41-55.
5. Лукьянов П.В. Применение численно-аналитического метода для решения задач акустики.// "Консонанс-2005", Акустический симпозиум. Киев-2005. Институт гидромеханики, 27-29 сентября,с.225-230.
6. Владимиров В.С. Уравнения математической физики.-М.:Наука,1981-512с.
7. Лукьянов П.В. Об одном численно-аналитическом подходе к решению задачи генерации звука тонким крылом. Часть І. Общая схема примпенения для плоской стационарной задачи.// Акустичний вісник, 2011.-V.14,N3,с.46-52.
8. *Caradonna F.X., Isom M.P.* Subsonic and Transonic Potential Flow over Helicopter Rotor Blades. // AIAA Journal, 1972. – V.10, N.12. – P. 1606-1612.

ЛУК'ЯНОВ Петро Володимирович,
кандидат фізико-математичних наук, старший науковий співробітник НАН України, доцент,
АРБ, ІАТ, НТТУ "КПІ"

ЧИСЕЛЬНО-АНАЛІТИЧНИЙ МЕТОД РОЗВ'ЯЗАННЯ ЗАДАЧ ЗАХИСТУ ОТОЧУЮЧОГО СЕРЕДОВИЩА

Анотація. *У задачах з моделювання забруднення води, атмосфери Землі часто використовують потенціальні моделі течії. Нещодавно значну увагу стали придіяти захисту метрополісів від акустичних шумових забруднень. Часто, не фокусуючи увагу на локальних рисах течії, для даного моделювання використовується рівняння Лапласа, яке описує*

потенціальний рух рідини. Акустичні задачі моделюються на основі рівняння Гельмгольца. У даній роботі розглянуто особливості використання чисельно-аналітичного методу для рівнянь Лапласа та Гельмгольца.. Звуковий потенціал є швидко осцилюючою функцією, заданою на границі розрахункової області. До того ж, представлено особливості чисельно-аналітичного методу для рівняння Лапласа з використанням розвинення граничної умови в ряд Тейлора за власними функціями задачі Штурма-Ліувілля. Не зважаючи на те, що дані, присутні у даній роботі, отримані для канонічних областей, схема методумає на увазі його використання для довільної криволінійної границі області. Чисельно-аналітичний метод, запропонований в даній роботі, дозволяє малими розрахунковими затратами чисельно розв'язувати задачі для рівняння Лапласа і акустичні рівняння. Результати даних досліджень можуть бути використані у якості нових інформаційних технологій для захисту оточуючого середовища.

Мета статті. *Метою статті є вивчення та розвиток застосування чисельно-аналітичного методу для граничних задач зі швидко-осцилюючими функціями, які задані на границі області.*

Висновки. *У даній роботі запропоновано алгоритм застосування чисельно-аналітичного методу для рівнянь Лапласа та Гельмгольца у випдку швидко осцилюючих функцій на границі. Виконано порівняння збіжності чисельного розв'язку задачи з аналітичним. Встановлено оптимальну кількість розбиття сітки розрахункової області для досягнення оптимальної збіжності чисельного розв'язку до аналітичного. Розв'язано задачу виникнення звукового поля для критичного діапазону обтікання лопаті, де виникають ударні хвилі, що можуть спричинити руйнування лопаті. Це дослідження може стати у нагоді екологічної безпеки польотів на гелікоптерах. Як показали дані розрахунку, чисельно-аналітичний метод здатний з порівняно малими затратами розв'язати граничні задачі для рівняння Лапласа та Гельмгольца. Результати даних досліджень спрямовані на розв'язання задач екологічної безпеки водних та повітряних ресурсів Землі.*

Ключові слова: *комп'ютерне моделювання, чисельний метод, екологічний моніторінг оточуючого середовища.*

Одержано редакцією 17.09.2021 р.
Прийнято до публікації 24.11.2021 р.

УДК 378:517

DOI 10.31651/2076-5886-2021-1-22-32

PACS 02.60.-x

СЕРДЮК Зоя Олексіївна,
кандидат педагогічних наук, доцент,
доцент кафедри математики та методики навчання математики, Черкаський національний університет імені Богдана Хмельницького
e-mail: serdyuk_z@ukr.net
ORCID: 0000-0002-9376-4346

ДЗЬОБА Микола Тарасович,
Національний університет «Львівська політехніка»
e-mail: mykola.lwiv@gmail.com

ЗАСТОСУВАННЯ ІНСТРУМЕНТАРІЮ ІНТЕГРАЛЬНОГО ЧИСЛЕННЯ ДО РОЗВ'ЯЗУВАННЯ ЗАДАЧ ГЕОМЕТРИЧНОГО ТА МЕХАНІЧНОГО ЗМІСТУ З МАТЕМАТИЧНОГО АНАЛІЗУ

У статті розглянуто семіотичні особливості вивчення математичних формул з курсу математичного аналізу, зокрема інтегрального числення; запропоновано закріплювати вивчені формули за трьома етапами (початковий, середній, заключний); показано застосування кожного

етапу на прикладі ряду завдань; враховано специфіку навчання математичному аналізу студентів спеціальностей 104 «Фізика та астрономія», 105 «Прикладна фізика та наноматеріали», 113 «Прикладна математика», 126 «Інформаційні системи та технології».

Ключові слова: *математичний аналіз, семіотичний підхід, математичні формули, застосування інтегралів у задача з геометрії та механіки, студенти-фізики, студенти-програмісти.*

Постановка проблеми

Курс математичного аналізу є одним із основних курсів математичного циклу дисциплін, які вивчають як студенти математичних, так і фізичних, технічних, ІТ спеціальностей ЗВО. Він насичений великою кількістю досить складних математичних фактів (теорем, формул тощо). За нашими спостереженнями засвоєння та, найголовніше, застосування тих чи тих фактів у процесі розв'язування задач часто зумовлюють появу труднощів у студентів. Тому методика їх вивчення має будуватися як специфічна для такої категорії студентів.

Аналіз останніх досліджень та публікацій

Питанням підвищення ефективності викладання курсу математичного аналізу та вдосконаленням методики його навчання студентів ЗВО присвячені роботи М. Бородіна, Л. Васяк, К. Власенко, М. Жалдак [1], Т. Крилової [2], О. Кондратьєвої [3], В. Клочка [4], І. Лов'янової, Т. Максимової, І. Михайлової, І. Михайленко, О. Потапової [5], С. Федорової, С. Федосєєва, Є. Ягової та ін.

Мета статті – розглянути особливості семіотичного компоненту навчання математичного аналізу, зокрема інтегрального числення та врахувати їх під час навчання студентів спеціальностей, як 104 «Фізика та астрономія», 105 «Прикладна фізика та наноматеріали», 113 «Прикладна математика», 126 «Інформаційні системи та технології», сформулювати відповідні рекомендації.

Виклад основного матеріалу

У Черкаському національному університеті імені Богдана Хмельницького навчальна дисципліна «Математичний аналіз» входить до циклу професійної підготовки зокрема студентів таких спеціальностей, як 104 «Фізика та астрономія», 105 «Прикладна фізика та наноматеріали», 113 «Прикладна математика», 126 «Інформаційні системи та технології». Ці дисципліни вивчаються у досить великому обсязі, як видно на рисунках 1-3. Крім того, цей предмет є важливим підґрунтям для подальшого вивчення дисциплін професійного спрямування за вказаними спеціальностями. Тому ґрунтовне вивчення саме даної дисципліни є важливим елементом подальшої професійної підготовки майбутніх фахівців з фізичних та ІТ-спеціальностей.

Оскільки вивчення математичного аналізу у зазначених спеціальностях розраховано на два семестри, то на другий семестр якраз припадає найбільший важіль – вивчення диференціального та інтегрального числення функцій кількох змінних (подвійні, потрійні інтеграли та їх застосування), криволінійні інтеграли 1-го та 2-го роду та їх застосування, поверхневі інтеграли 1-го та 2-го роду та їх застосування. Деякі особливості вивчення математичного аналізу студентами-фізиками та студентами-програмістами були нами описані у роботах [6-9] та ін. Проте комплексно методика вивчення математичного аналізу майбутніми фахівцями вищезазначених спеціальностей розглянута не була. Інструментарій математичного аналізу, зокрема інтегрального числення дуже потужний, тому й використання його під час розв'язання різних фізичних, математичних задач, завдань математичного моделювання,

програмування може бути надзвичайно важливим. Тому студенти мають вільно володіти даним матеріалом, зокрема обирати серед багатьох й застосовувати саме ті математичні формули, за допомогою яких задача буде розв'язана якнайпростіше.

1	2	3	4	5	6	7	8	9	10	11	12
1. ОБОВ'ЯЗКОВІ КОМПОНЕНТИ ОСВІТНЬО-ПРОФЕСІЙНОЇ ПРОГРАМИ											
1.1. Цикл професійної підготовки											
ОК 01	Інформатика	1			6	180	60	30	30		120
ОК 02	Програмування	1;2			12	360	120	60	60		240
ОК 03	Дискретна математика		2		5	150	50	24		26	100
ОК 04	Алгебра та геометрія	1;2			11	330	110	54		56	220
ОК 05	Програмне забезпечення та інформаційно-комунікаційні технології		2		5	150	50	20	30		100
ОК 06	Алгоритми та структури даних	3;4			12	360	120	60	60		240
ОК 07	Мови програмування	3;4			10	300	100	40	60		200
ОК 08	Математичний аналіз	4	3		10	300	100	48		52	200
ОК 09	Теорія ймовірностей та математична статистика		3		4	120	40	20		20	80
ОК 10	Архітектура обчислювальних систем	3			4	120	40	14	26		80
ОК 11	Веб-програмування	4			5	150	50	20	30		100
ОК 12	Об'єктно-орієнтоване програмування	5			6	180	60	20	40		120
ОК 13	Бази даних та інформаційні системи	5			6	180	60	20	40		120

Рис. 1. Фрагмент навчального плану спеціальності 126 «Інформаційні системи та технології»

1. ОБОВ'ЯЗКОВІ КОМПОНЕНТИ ОСВІТНЬО-ПРОФЕСІЙНОЇ ПРОГРАМИ											
1.1. Цикл професійної підготовки											
ОК 01	Інформатика	1			6	180	60	30	30		120
ОК 02	Програмування	1;2			12	360	120	60	60		240
ОК 03	Дискретна математика		2		5	150	50	24		26	100
ОК 04	Алгебра та геометрія	1;2			11	330	110	54		56	220
ОК 05	Програмне забезпечення та інформаційно-комунікаційні технології		2		5	150	50	20	30		100
ОК 06	Алгоритми та структури даних	3;4			12	360	120	60	60		240
ОК 07	Мови програмування	3;4			10	300	100	40	60		200
ОК 08	Математичний аналіз	4	3		10	300	100	48		52	200
ОК 09	Теорія ймовірностей та математична статистика		3		4	120	40	20		20	80
ОК 10	Архітектура обчислювальних систем	3			4	120	40	14	26		80
ОК 11	Веб-програмування	4			5	150	50	20	30		100
ОК 12	Об'єктно-орієнтоване програмування	5			6	180	60	20	40		120
ОК 13	Бази даних та інформаційні системи	5			6	180	60	20	40		120
ОК 14	Методи обчислень	5			6	180	60	30	30		120

Рис. 2. Фрагмент навчального плану спеціальності 113 «Прикладна математика»

1	2	3	4	5	6	7	8	9	10	11	12
1. ОБОВ'ЯЗКОВІ КОМПОНЕНТИ ОСВІТНЬО-ПРОФЕСІЙНОЇ ПРОГРАМИ											
1.1. Цикл професійної підготовки											
ОК01	Вступ до фаху		1		6	180	60	20	20	20	120
ОК02	Вища математика	1;2;3			15	450	180	60		120	270
ОК03	Математичний аналіз	1;2			9	270	120	40		80	150
ОК04	Механіка	2			9	270	90	30	30	30	180
ОК05	Молекулярна фізика	3			9	270	90	30	30	30	180
ОК06	Електрика і магнетизм	4			9	270	90	30	30	30	180
ОК07	Оптика	5			6	180	90	30	30	30	90
ОК08	Фізика атома	6			6	180	90	30	30	30	90
ОК09	Фізика ядра і елементарних частинок	7			6	180	60	30		30	120
ОК10	Курсова робота з фаху			8	3	90					90
ОК11	Класична механіка	4			6	180	60	30		30	120
ОК12	Електродинаміка	4			6	180	60	30		30	120

Рис. 3. Фрагмент навчального плану спеціальності 105 «Прикладна фізика та наноматеріали»

Виклад навчального матеріалу не варто будувати, спираючись лише на логіку розгортання змісту. Потрібно також враховувати й семіотичну його специфіку.

Особливої турботи викладача вимагає прогнозування утруднень та помилок, які виникають у студентів під час засвоєння теоретичного матеріалу та розв'язування задач. Опановуючи нові факти студенти-фізики дуже часто запам'ятовують лише їх оболонку. При цьому особливості змісту залишаються поза їх увагою. Згодом, на етапі засвоєння того чи того факту або його використання в конкретному завданні, у студентів виникають певні труднощі. Тому для ефективного засвоєння студентами математичних фактів необхідно приділяти спеціальну увагу процедурам упізнавання і розпізнавання. Першу з них ми пов'язуємо з візуальним аналізом, а другу – зі змістовим. Н. А. Тарасенкова [10] трактує візуальний аналіз як процес зорового упізнавання об'єкта засвоєння, а смисловий аналіз – як процес розпізнавання змісту. Своєю чергою, змістовий аналіз – це поєднання двох процесів: візуального й смислового аналізу. Як правило, візуальний аналіз опереджає в часі смисловий, однак іноді вони проходять одночасно. Проте діалектична єдність візуального і логічного не виключає появи протиріч між ними. Найчастіше такі протиріччя виникають у студентів на етапі первісного ознайомлення з об'єктом засвоєння. Виникнення конфліктів між візуальним і логічним у процесі вивчення математики студентами-фізиками доволі часто є неминучими. Однак, для того щоб уникати таких конфліктів, потрібно, щоб вони вміли вільно оперувати знаково-символічними оболонками тих чи тих об'єктів засвоєння та пов'язувати їх зі змістовими особливостями цих об'єктів.

Таким чином, пропонуємо поділити вивчення та засвоєння математичних формул на три умовних етапи: 1) 1 етап (*початковий*) – це звичайне упізнавання математичної формули серед схожих, тобто для цього використовуємо тестові завдання з вибором однієї правильної відповіді; 2) 2 етап (*середній*) – це вибір формули до певної математичної задачі серед інших, тобто є підказка; можна використовувати також тестові завдання з вибором однієї правильної відповіді або ж тестові завдання на відповідність; 3) 3 етап (*заключний*) – це розв'язування математичної задачі, вже без підказки, це завдання відкритої форми. Розглянемо завдання кожного етапу засвоєння математичних формул на прикладі різних тем інтегрального числення. Завдання можна формулювати у вигляді пошукових задач, квестів тощо для підвищення зацікавленості студентів.

Студенти саме зазначених спеціальностей (104 «Фізика та астрономія», 105 «Прикладна фізика та наноматеріали», 113 «Прикладна математика», 126 «Інформаційні системи та технології») вирізняються логічним складом мислення, структурованим та алгоритмічним підходом до виконання тих чи тих завдань, розвиненим візуальним сприйняттям. Крім того, вектор завдань, які ми пропонуємо як приклади для розв'язання, а саме застосування різних інтегралів до розв'язування задач з геометрії та механіки, можуть бути корисні для вищезазначених категорій студентів у їх подальшій професійній діяльності.

Завдання 1 етапу засвоєння

На цьому етапі доцільно засвоювати математичні формули спочатку всередині кожної теми (наприклад – подвійний інтеграл (*завдання 1*), криволінійний інтеграл 1–го роду (завдання 2)), потрійний інтеграл (*завдання 3*) тобто серед запропонованого набору схожих візуально формул, але в межах одного типу інтеграла студенти мають розпізнати потрібний їм. Тобто, вони не акцентують тут увагу на тип інтеграла, а лише на його змістове наповнення, на підінтегральний вираз.

Завдання 1. Для того, щоб отримати доступ до онлайн-аудиторії, необхідно розгадати п'ятизначний шифр кодового замка. Послідовно (не змінюючи порядку), треба набрати букви, які є правильними відповідями до наступних завдань.

1. Серед наведених нижче формул оберіть правильну (V – об'єм тіла).

A	B	C	D
$V = \iint\limits_D f(x;y)dxdy$	$V = \iint\limits_D dxdy$	$V = \iint\limits_D xdxdy$	$V = \iint\limits_D ydxdy$

2. Серед наведених нижче формул оберіть правильну (S – площа фігури).

A	B	C	D
$S = \iint\limits_D xdxdy$	$S = \iint\limits_D ydxdy$	$S = \iint\limits_D f(x;y)dxdy$	$S = \iint\limits_D dxdy$

3. Серед наведених нижче формул оберіть правильну (m – маса пластини, γ – густина пластини).

A	B	C	D
$m = \iint\limits_D \gamma dxdy$	$m = \iint\limits_D \gamma xdxdy$	$m = \iint\limits_D \gamma ydxdy$	$m = \iint\limits_D \gamma f(x;y)dxdy$

4. Серед наведених нижче формул оберіть правильну (S_x – статичний момент плоскої пластини щодо осі OX, γ – густина пластини).

A	B	C	D
$S_x = \iint\limits_D y \cdot \gamma(x;y)dxdy$	$S_x = \iint\limits_D x \cdot \gamma(x;y)dxdy$	$S_x = \iint\limits_D \gamma(x;y)dxdy$	$S_x = \iint\limits_D xydxdy$

5. Серед наведених нижче формул оберіть правильну (M_y – момент інерції плоскої пластини щодо осі OY, γ – густина пластини).

A	B	C	D
$M_y = \iint\limits_D x^2\gamma(x;y)dxdy$	$M_y = \iint\limits_D y^2\gamma(x;y)dxdy$	$M_y = \iint\limits_D \gamma(x;y)dxdy$	$M_y = \iint\limits_D x^2dxdy$

Завдання 2. Для того, щоб отримати розблокувати мобільний телефон необхідно розгадати чотиризначний шифр. Послідовно (не змінюючи порядку), треба набрати букви, які є правильними відповідями до наступних завдань.

1. Серед наведених нижче формул оберіть правильну (m – маса кривої, γ – густина кривої).

A	B	C	D
$m = \int\limits_{AB} \gamma(x;y)dl$	$m = \int\limits_{AB} x\gamma(x;y)dl$	$m = \int\limits_{AB} y\gamma(x;y)dl$	$m = \int\limits_{AB} xydl$

2. Серед наведених нижче формул оберіть правильну (l – довжина кривої).

A	B	C	D
$l = \int\limits_{AB} ydl$	$l = \int\limits_{AB} xdl$	$l = \int\limits_{AB} dl$	$l = \int\limits_{AB} xydl$

3. Серед наведених нижче формул оберіть правильну (S_y – статичний момент кривої щодо осі OY, γ – густина кривої).

A	B	C	D
$S_y = \int\limits_{AB} \gamma(x;y)dl$	$S_y = \int\limits_{AB} y\gamma(x;y)dl$	$S_y = \int\limits_{AB} x\gamma(x;y)dl$	$S_y = \int\limits_{AB} xy\gamma(x;y)dl$

4. Серед наведених нижче формул оберіть правильну (M_x – момент інерції кривої щодо осі OX, γ – густина кривої).

A	B	C	D
$M_x = \int\limits_{AB} x^2\gamma(x;y)dl$	$M_x = \int\limits_{AB} \gamma(x;y)dl$	$M_x = \int\limits_{AB} \left(x^2+y^2\right)\gamma(x;y)dl$	$M_x = \int\limits_{AB} y^2\gamma(x;y)dl$

Завдання 3. Домофон в будинку має тризначний код. Щоб його отримати, необхідно розв'язати наступні завдання. Послідовно (не змінюючи порядку), треба набрати букви, які й є правильними відповідями до запропонованих завдань.

1. Серед наведених нижче формул оберіть правильну (V – об'єм тіла).

A	B	C	D
$V = \iiint\limits_V f(x;y;z)dxdydz$	$V = \iiint\limits_V dxdydz$	$V = \iiint\limits_V xyzdxdydz$	$V = \iint\limits_S xydxdy$

2. Серед наведених нижче формул оберіть правильну (m – маса тіла, $\gamma(x;y;z)$ – об'ємна густина розподілу маси в точці $M(x;y;z)$).

A	B	C	D
$m = \iiint\limits_V x\gamma(x;y;z)dxdydz$	$m = \iiint\limits_V y\gamma(x;y;z)dxdydz$	$m = \iiint\limits_V \gamma(x;y;z)dxdydz$	$m = \iiint\limits_V dxdydz$

3. Серед наведених нижче формул оберіть правильну (I_{xy} – момент інерції тіла відносно площини XOY, $\gamma(x;y;z)$ – об'ємна густина розподілу маси в точці $M(x;y;z)$).

A	B	C	D
$I_{xy} = \iiint\limits_V \gamma(x;y;z)dv$	$I_{xy} = \iiint\limits_V z^2\gamma(x;y;z)dv$	$I_{xy} = \iiint\limits_V x^2\gamma(x;y;z)dv$	$I_{xy} = \iiint\limits_V y^2\gamma(x;y;z)dv$

Завдання 2 етапу засвоєння

На цьому етапі доцільно вже ускладнити завдання, тобто підвищити рівень знання та відтворення математичних формул. Доцільно запропонувати студентам дібрати до задачі математичну формулу, за допомогою якої її можна буде розв'язати, але серед запропонованих відповідей пропонувати вже різні типи інтегралів, а підінтегральну функцію, наприклад, зафіксувати, або ж ще ускладнити: змінювати і тип інтеграла і підінтегральну функцію, за бажанням викладача. Таким чином студенти візуально мають знайти потрібний їм інтеграл серед запропонованих (*завдання 4*).

Завдання 4. Дано астроїду $x = a\cos t$, $y = a\sin t$. Розв'яжіть наступні завдання:

1. Серед наведених нижче формул оберіть ту, за допомогою якої можна обчислити площу фігури, обмеженої астроїдою (S – площа фігури).

A	B	C	D
$S = \iint\limits_D dxdy$	$S = \iiint\limits_D dxdydz$	$S = \int\limits_l dl$	$S = \int\limits_l xdy$

2. Серед наведених нижче формул оберіть ту, за допомогою якої можна обчислити довжину астроїди (l – довжина кривої).

A	B	C	D
$l = \iint\limits_D dxdy$	$l = \iiint\limits_V dxdydz$	$l = \int\limits_l dl$	$l = \int\limits_l xdy$

3. Серед наведених нижче формул оберіть ту, за допомогою якої можна обчислити масу астроїди (m – маса кривої, γ – густина кривої).

A	B	C	D
$m = \int\limits_l y\gamma(x;y)dl$	$m = \int\limits_l x\gamma(x;y)dl$	$m = \int\limits_l dl$	$m = \int\limits_l \gamma(x;y)dl$

Завдання 3 етапу засвоєння

На цьому етапі вже доцільно пропонувати студентам розв'язувати комплексне завдання на знаходження площі, об'єму, маси, статичних моментів, моментів інерції, центрів ваги тощо. Під час виконання таких завдань студент повинен сам свідомо обрати тип інтеграла, який підходить до умови задачі та відповідну математичну формулу (*завдання 5 та завдання 6*). Проте перед виконанням такого завдання як узагальнення можна запропонувати повторити матеріал за відповідною схемою (рис. 4).

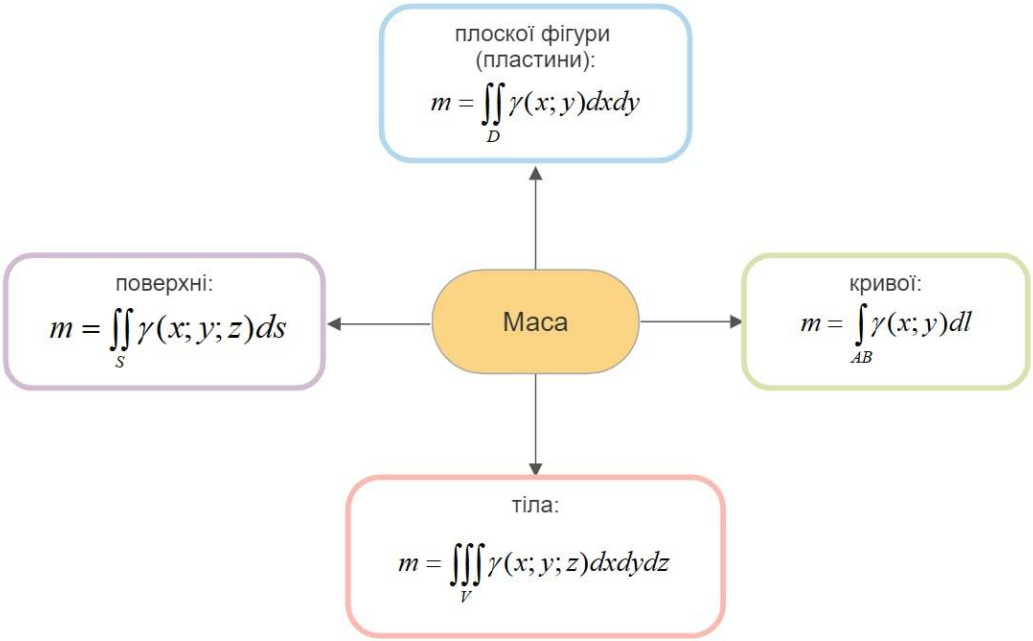


Рис. 4. Математичні формули для обчислення мас
(де γ – відповідно густина пластини, кривої тіла чи поверхні)

Завдання 5. Знайдіть центр ваги однорідного півкола, що лежить у верхній півплощині, якщо його густина $\gamma = 1$ в кожній точці кривої.

Розв'язання. Найважливішим моментом розв'язування даної задачі є вдале розміщення даної кривої у системі координат. У даному випадку вказано, що півколо знаходиться у верхній півплощині. Крім цього, найкраще розмістити його так, щоб центр півкола співпадав з початком координат, а вісь ординат була віссю його симетрії (рис. 5).

Звідси з міркувань симетрії зрозуміло, що центр ваги даного півкола знаходиться на осі *OY* (див. рис. 5). Тому $x_c = 0$.

Ординату центру ваги знаходимо за формулою $y_c = \frac{\int\limits_{AB} ydl}{\int\limits_{AB} dl}$. Варто звернути увагу студентів на те, що оскільки ми працюємо з кривою, то інтеграли у даній формулі – криволінійні, 1-го роду. Знаменник даного дробу – це довжина півкола, її ми можемо знайти, використовуючи формулу довжини кола з планіметрії: $l = \frac{1}{2} \cdot 2\pi R = \pi K$. А для обчислення чисельника скористаємося параметричними рівняннями кола, оскільки так знаходити інтеграл найзручніше: $x = R\cos t$, $y = R\sin t$, $0 \le t \le \pi$.

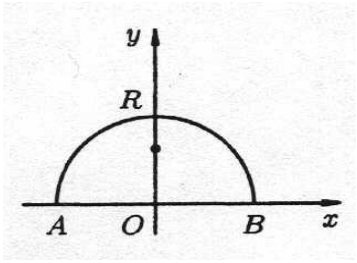


Рис. 5.

Маємо: $\int\limits_{AB} y \cdot dl = \int\limits_0^{\pi} R\sin \cdot \sqrt{R^2 \sin^2 t + R^2 \cos^2 t} \cdot dt = R^2 \int\limits_0^{\pi} \sin t \cdot dt = 2R^2.$

Звідси : $y_c = \frac{2R^2}{\pi R} = \frac{2R}{\pi}$.

Тому остаточно координати центра ваги півкола: $x_c = 0,\ y_c = \frac{2R}{\pi}$.

Відповідь: $C\left(0; \frac{2R}{\pi}\right)$.

Завдання 6. Знайдіть центр ваги однорідного півкруга, що лежить у верхній півплощині, якщо його густина $\gamma = 1$ в кожній точці даної фігури.

Розв'язання. Аналогічно як і для попередньої задачі, головне – це вдале розміщення даної плоскої фігури у системі координат. Оскільки півкруг знаходиться у верхній півплощині, то найкраще розмістити його так, щоб центр співпадав з початком координат, а вісь ординат була віссю його симетрії (рис. 6).

Звідси з міркувань симетрії зрозуміло, що центр ваги даного півкруга знаходиться на осі *OY* (див. рис. 6). Тому $x_c = 0$.

Ординату центру ваги знаходимо за формулою $y_c = \frac{\iint\limits_D ydxdy}{\iint\limits_D dxdy}$. Варто звернути увагу студентів на те, що оскільки в цій задачі ми вже працюємо з плоскою фігурою, то інтеграли у даній формулі – подвійні.

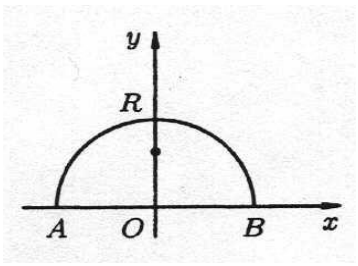


Рис. 6.

Знаменник даного дробу – це площа даної фігури, а саме площа півкруга, її ми можемо знайти, використовуючи формулу площі круга з планіметрії: $S = \frac{1}{2} \cdot \pi R^2 = \frac{\pi K^2}{2}$. А для обчислення чисельника скористаємося полярними координатами, оскільки так знаходити інтеграл найзручніше: $x = r\cos t$, $y = r\sin t$, $0 \le r \le R$, $0 \le t \le \pi$.

Маємо: $\iint\limits_D ydxdy = R\int\limits_0^{\pi} \sin t dt \int\limits_0^R rdr = R\int\limits_0^{\pi} \sin t \cdot \frac{R^3}{3} dt = \frac{R^4}{3} \cdot \left(-\cos\pi + \cos 0\right) = \frac{2R^3}{3}.$

Звідси : $y_c = \frac{2 \cdot 2R^3}{3\pi R^2} = \frac{4R}{3\pi}$.

Тому остаточно координати центра ваги півкола: $x_c = 0,\ y_c = \frac{4R}{3\pi}$.

Відповідь: $C\left(0; \frac{4R}{3\pi}\right)$.

Висновки

Подальші дослідження ми бачимо у розробці такої ж системи завдань до кожної теми математичного аналізу з урахуванням специфіки підготовки майбутніх фахівців вищезазначених спеціальностей.

Список використаної літератури:

1. Жалдак М. І. Математичний аналіз з елементами інформаційних технологій: навчальний посібник / М. І. Жалдак, Г. О. Михалін, С. Я. Деканов. – К. : Редакції газет природничо-математичного циклу, 2012. – 128 с.
2. Клочко В. І. Формування знань майбутніх інженерів з інформаційних технологій розв'язування диференціальних рівнянь : монографія / В. І. Клочко, З. В. Бондаренко. – Вінниця : ВНТУ, 2010. – 216 с.
3. Кондратьєва О. М. Методична система контролю і коригування знань та умінь студентів технічних спеціальностей у процесі навчання вищої математики : автореф. дис... канд. пед. наук : 13.00.02 – теорія та методика навчання математики / Оксана Марківна Кондратьєва ; Нац. пед. ун-т ім. М. П. Драгоманова. – К., 2007. – 20 с
4. Крилова Т. В. Наукові основи навчання математики студентів нематематичних спеціальностей (на базі металургійних, енергетичних і електромеханічних спеціальностей вищого закладу технічної освіти) : автореф. дис... д-ра пед. наук: 13.00.02; Нац. пед. ун-т ім. М. П. Драгоманова. – К., 1999. – 36 с.
5. Потапова О. М. Аналіз дослідження труднощів студентів технічних спеціальностей при вивченні математичного аналізу / О. М. Потапова // Вісник Черкаського університету. Серія Педагогічні науки. – № 12 (265). – Черкаси, 2013. – С. 83–90.
6. Сердюк З. О., Христенко Т. М. Математичний тезаурус як інтелектуальний засіб навчання студентів фізичних спеціальностей ВНЗ. Засоби і технології сучасного навчального середовища // Матеріали всеукраїнської науково-практичної конференції (20 – 21 травня 2011 року, м. Кіровоград). – С. 78–79.
7. Сердюк З. О. Особливості вивчення навчальної дисципліни «Математичний аналіз» для студентів фізичних спеціальностей ВНЗ / З. О. Сердюк // Актуальні проблеми і перспективи дидактики фізики // Матеріали Всеукраїнської науково-практичної конференції (26 – 28 квітня 2012 року, м. Черкаси). – Черкаси, ЧНУ ім. Б. Хмельницького. – С. 51–52.
8. Сердюк З. О. Реалізація компетентнісного підходу під час вивчення курсу математичного аналізу в ВНЗ. – Вісник Черкаського університету, Випуск № 8 (341): серія «Педагогічні науки». – Черкаси: Вид. від. ЧНУ ім. Б. Хмель¬ницького, 2015. – С. 101-106.
9. Тарасенкова Н. А., Сердюк З. О. Особливості викладання курсу математичного аналізу для фахівців з аналізу даних. Вісник Черкаського університету. Серія "Прикладна математика. Інформатика". 2020. № 1. Черкаси : Вид. від. ЧНУ ім. Б.Хмельницького. С. 57-73. **Published:** Mar 18, 2021. DOI 10.31651/2076-5886-2020-1-77-86
10. Тарасенкова Н. А. Використання знаково-символічних засобів у навчанні математики : [монографія] / Н. А. Тарасенкова. – Черкаси : Відлуння-плюс, 2002. – 400 с.

References:

1. Zhaldak, M., Mikhalin, G., Dekanov, S. (2012). Mathematical analysis with elements of information technology: [textbook] [in Ukrainian].
2. Klochko, V., Bondarenko, Z. (2010). Formation of knowledge of future engineers on information technologies for solving differential equations: [monograph] [in Ukrainian].
3. Kondratieva, O. (2007). Methodical system of control and correction of knowledge and skills of students of technical specialties in the process of learning higher mathematics: author. dis ... cand. ped. sciences: 13.00.02 - theory and methods of teaching mathematics [in Ukrainian].
4. Krylova, T. (1999). Scientific bases of teaching mathematics to students of non-mathematical specialties (based on metallurgical, energy and electromechanical specialties of higher technical education): author's ref. dis ... Dr. ped. Sciences: 13.00.02 [in Ukrainian].
5. Potapova, O. (2013). Analysis of the study of difficulties of students of technical specialties in the study of mathematical analysis. Bulletin of Cherkasy University. Pedagogical Sciences Series, 12 (265), 83–90 [in Ukrainian].
6. Serdiuk, Z. (2011). Mathematical thesaurus as an intellectual means of teaching students of physical specialties of higher education. Means and technologies of modern educational environment. Proceedings of the All-Ukrainian scientific-practical conference, 78–79 [in Ukrainian].
7. Serdiuk, Z. (2012). Peculiarities of studying the discipline "Mathematical analysis" for students of physical specialties of higher education. Actual problems and prospects of didactics of physics. Proceedings of the All-Ukrainian scientific-practical conference, 51–52 [in Ukrainian].
8. Serdiuk, Z. (2015). Implementation of the competence approach during the study of the course of mathematical analysis in higher education. Bulletin of Cherkasy University, 8 (341): series "Pedagogical Sciences", 101-106 [in Ukrainian].
9. Tarasenkova, N., Serdiuk, Z. (2020). Features of teaching a course of mathematical analysis for data analysis specialists. Bulletin of Cherkasy University. Series “Applied Mathematics. Informatics", 1, 57-73. Published: Mar 18, 2021. DOI 10.31651/2076-5886-2020-1-77-86
10. Tarasenkova, N. (2002). The use of sign-symbolic means in teaching mathematics: [monograph] [in Ukrainian].

SERDIUK Zoia,
PhD (Pedagogical Sciences), Associate Professor of the Department of Mathematics and Methods of Learning of Mathematics, Cherkasy Bohdan Khmelnytsky National University
Dzoba Mykola,
Lviv Polytechnic National University

APPLICATION OF INTEGRAL NUMBERING TOOLS FOR SOLVING PROBLEMS OF GEOMETRIC AND MECHANICAL CONTENT FROM MATHEMATICAL ANALYSIS

Abstract. Introduction. *Mathematical analysis has powerful tools, especially in integral calculus, for solving a number of problems for application in geometry and mechanics. However, it is difficult for students to memorize a large number of complex mathematical formulas. In this article we propose to carry out a step-by-step mastering of mathematical formulas and give a number of tasks that can be used. The task is designed for visual perception and assimilation, as the semiotic component of learning qualitatively helps mechanical memorization and facilitates the application of the studied material to solve practical problems. This approach is aimed not only at memorizing the basic mathematical formulas for integral calculation of functions of one or more variables and using them to calculate the lengths of curves, areas, volumes, masses, moments of inertia and static moments, etc., but also the formation of students' ability to and consistently think, analyze, compare, summarize, etc., in general, the ability to draw the right conclusions and make realistic predictions, to apply the acquired knowledge, skills and abilities to solve various practical problems. However, the specifics of the organization of the study of mathematical analysis taking into account the peculiarities of the content of education specialties 104 "Physics and Astronomy", 105 "Applied Physics and Nanomaterials", 113 "Applied Mathematics", 126 "Information Systems and Technologies" is still not considered carefully enough and remains at the center of our research.*

Purpose *– consider the features of the semiotic component of teaching mathematical analysis, in particular integral calculus and take them into account when teaching students specialties such as 104 "Physics and Astronomy", 105 "Applied Physics and Nanomaterials", 113 "Applied Mathematics", 126 "Information Systems and Technologies", formulate appropriate recommendations.*

Originality. *In this article, taking into account the latest trends in educational policy and the specifics of training in specialties 104 "Physics and Astronomy", 105 "Applied Physics and Nanomaterials", 113 "Applied Mathematics", 126 "Information Systems and Technologies", namely - the trend towards algorithmization when solving certain practical problems, it is advisable to use visualization in the study of mathematical analysis by the above students – and during the assimilation of theoretical material, and when solving problems. We have identified three stages of mastering mathematical formulas and proposed a number of exercises.*

Conclusion. *We see further research in the development of the same system of tasks for each topic of mathematical analysis, taking into account the specifics of training future professionals in the above specialties.*

Keywords: *mathematical analysis, semiotic approach, mathematical formulas, application of integrals in the exercises of geometry and mechanics, physics students, programming students*

Одержано редакцією 26.08.2021 р.
Прийнято до публікації 27.10.2021 р.

УДК 519.688

DOI 10.31651/2076-5886-2021-1-32-49

PACS 02.70.-c

ПОРУБЛЬОВ Ілля Миколайович,
викладач, Черкаський національний університет імені Богдана Хмельницького
e-mail: ilya@vu.cdu.edu.ua
ORCID: 0000-0001-7369-3862

ЩЕ ОДИН АЛГОРИТМ ПОШУКУ ЧИСЕЛ З МАКСИМАЛЬНОЮ КІЛЬКІСТЮ ДІЛЬНИКІВ (НА ОСНОВІ ІДЕЇ МНОЖИНИ НЕДОМІНОВАНИХ ПАР)

Ця стаття по суті є продовженням статті [6]. В ній розглянуто майже ту саму постановку задачі (пошук числа, що має максимальну кількість дільників серед всіх чисел проміжку, але цього разу лише від 1 до вказаного N, і треба шукати мінімальне з чисел, що мають цю максимальну кількість дільників). Попри схожість формулювань, запропоновано принципово інший спосіб розв'язання, що базується на введеному понятті недомінованих пар: пара (число, кількість його дільників) вважається недомінованою, якщо не існує менших чисел, що мали б більшу або рівну кількість дільників. Аналогічно [6], розглянуто узагальнення, коли максимальна кількість дільників шукається не серед усіх чисел проміжку, а лише серед не кратних деякому K (натуральному, більшому або рівному 2, не обов'язково простому).

Ключові слова: *кількість дільників, надскладені числа, недоміновані пари, впорядкована словникова структура даних, map, lower_bound, upper_bound.*

Вступ

Оскільки стаття є по суті продовженням [6], причини актуальності аналогічні — істотно різні кількості дільників у різних чисел, що впливає на тривалість роботи деяких алгоритмів. Використано спільну з [6] нумерацію: означення та твердження, що вже були там і виявилися важливими також і тут, повторені під тими ж номерами, а новим надані нові (не використані в [6]) номери.

Дотримано точки зору, що натуральними є цілі додатні чи*с*ла (а 0 не є).

Означення 1. Будемо, згідно [2], позначати кількість дільників числа n як $\tau(n)$.

Означення 2. Будемо, згідно [1], називати число n ***надскладеним***, якщо кількість його дільників строго перевищує кількість дільників будь-якого меншого числа. Інакше кажучи, n надскладене, коли

$$\forall j_{1\le j<n}\big(\tau(j)<\tau(n)\big). \qquad (1)$$

Як і в [6], зручно вважати, що прості чи*с*ла занумеровані як $p_0=2$, $p_1=3$, $p_2=5$, …, а коли з контексту ясно, про розкладення якого числ*а* йдеться, то k_0, k_1, k_2, … позначають відповідні показники степенів ($p_0=2$ підноситься до степеню k_0, $p_1=3$ до степеню k_1, …, причому k_i можуть бути як натуральними, так і нулями).

Мета статті

Описати принципово інший, чим у [6], розроблений алгоритм для розв'язання задачі пошуку чисел з великою кількістю дільників, у таких варіантах постановки:

1. Дано натуральне число N. Для проміжку всіх натуральних чисел від 1 до N (включно), знайти мінімальне з чисел, що має максимальну кількість дільників (інакше кажучи, останнє, воно ж найбільше, надскладене число цього проміжку).

3. Дані натуральні чи*с*ла N та K ($K\ge 2$). Для проміжку натуральних чисел від 1 до N (включно), але ігноруючи чи*с*ла, кратні K, знайти мінімальне з чисел, що має максимальну кількість дільників,

а) якщо гарантовано, що число K просте;

б) якщо такої гарантії нема.

В усіх варіантах постановки, потрібно знайти і максимальну кількість дільників, і мінімальне з чисел, для яких досягається така кількість дільників. Таку пару будемо називати також *повною відповіддю*. Коли виникатиме необхідність розрізнити, яке одне з цих двох чисел мається на увазі, вживатимуться словосполучення *відповідь-кількість* (дільників) та *відповідь-значення* (на якому числі досягається така кількість дільників).

Основною ідеєю цієї статті, повністю новою відносно [6], є ідея недомінованих пар, введена в озн. 4.

Виклад основного матеріалу

Повторимо потрібні зараз означення, твердження (без доведень) та наслідки з [6].

Твердження 1. Нехай відоме канонічне розкладення числа N на прості множники

$$N = p_1^{k_1}\cdot p_2^{k_2}\cdot\ldots\cdot p_m^{k_m} \qquad (2)$$

(всі p_i прості та різні, всі k_i натуральні). Тоді кількість різних дільників N дорівнює

$$\tau\left(p_1^{k_1}\cdot p_2^{k_2}\cdot\ldots\cdot p_m^{k_m}\right)=(k_1+1)\cdot(k_2+1)\cdot\ldots\cdot(k_m+1). \qquad (3)$$

(доведення можна бачити, наприклад, у [4, § 9])

Наслідок твердження 1. Якщо чи*с*ла a та b взаємно прості, то $\tau(a\cdot b)=\tau(a)\cdot\tau(b)$; інакше, $\tau(a\cdot b)<\tau(a)\cdot\tau(b)$.

Твердження 2. Для всіх надскладених чисел, $k_0\ge k_1\ge\ldots$

Означення 3. Для постановки № 3б, будемо позначати $z_0, z_1, \ldots$ показники, з якими прості $p_0=2$, $p_1=3$, $p_2=5$, … входять у розкладення K, кратність якому заборонена: $K=p_0^{z_0}\cdot p_1^{z_1}\cdot\ldots$ Значення z_0, z_1, … цілі невід'ємні (можуть дорівнювати 0).

На цьому основні повтори з [6] завершені, далі йде новий матеріал.

Будемо працювати з парами $(n_i, \tau(n_i))$, тобто числом та кількістю його дільників. Теоретично, другий елемент кожної такої пари визначається першим. Однак, будемо зберігати готові другі елементи, а не обчислювати їх через перші, бо так швидше.

Означення 4. Будемо казати, що пара $(a, \tau(a))$ ***домінує*** пару $(b, \tau(b))$, й позначати це знаком «$\succ$», як $(a,\tau(a)) \succ (b,\tau(b))$, коли

$$(a < b) \wedge (\tau(a) \geq \tau(b)); \qquad (12)$$

будемо також називати пару $(b, \tau(b))$ ***домінованою*** парою $(a, \tau(a))$, відповідно пару $(a, \tau(a))$ — ***домінуючою*** пару $(b, \tau(b))$.

Наприклад: пара (9, 3) домінує пару (13, 2), але пара (9, 3) домінована будь-якою з пар (6, 4) чи (4, 3). Це озн. 4 введено самостійно, взявши за основу Парето-домінування, описане у [7] та пізніше популяризоване в багатьох інших джерелах, зокрема [3].

Твердження 8. Введене в озн. 4 домінування є нелінійним (неповним) відношенням строгого порядку.

Доведення. Пари $(a, \tau(a))$ та $(b, \tau(b))$ однотипні та їх дві, тому домінування можна розглядати як бінарне відношення. Враховуючи строгу нерівність $a < b$ у перших дужках (12), відношення домінування іррефлексивне (жодна пара не домінує саму себе). Та сама умова $a < b$ гарантує також антисиметричність (не може бути двох різних пар, кожна з яких домінує іншу). Тепер транзитивність: якщо $(a,\tau(a)) \succ (b,\tau(b))$ та $(b,\tau(b)) \succ (c,\tau(c))$, то $a < b$, $b < c$, $\tau(a) \geq \tau(b)$, $\tau(b) \geq \tau(c)$, тому, за транзитивністю звичайних числових "<" та "≥", маємо також $a < c$, $\tau(a) \geq \tau(c)$, тобто $(a,\tau(a)) \succ (c,\tau(c))$. Це відношення не є повним, тобто можуть бути такі пари $(a, \tau(a))$ та $(b, \tau(b))$, що жодна з них не домінує іншу; прикладами таких пар є, скажімо, (40, 8) та (100, 9). Отже, всі ознаки нелінійного відношення строгого порядку перевірено. ■

Твердження 9. Нехай $(a,\tau(a)) \succ (c,\tau(c))$. Тоді для будь-якого натурального m, взаємно простого з a, $(a \cdot m,\tau(a \cdot m)) \succ (c \cdot m,\tau(c \cdot m))$.

Доведення. Маємо $(a,\tau(a)) \succ (c,\tau(c))$, тобто $a < c$ та $\tau(a) \geq \tau(c)$. Помноживши першу нерівність на m, маємо $a \cdot m < c \cdot m$, що є складовою (12) для пар $(a \cdot m, \tau(a \cdot m))$ та $(c \cdot m, \tau(c))$. Помноживши другу нерівність на $\tau(m)$, маємо $\tau(a) \cdot \tau(m) \geq \tau(c) \cdot \tau(m)$. Оскільки m взаємно просте з a, за наслідком твердж. 1, $\tau(a \cdot m) = \tau(a) \cdot \tau(m)$. Оскільки невідомо, чи m взаємно просте з c, за тим самим наслідком маємо $\tau(c \cdot m) \leq \tau(c) \cdot \tau(m)$. Тобто, маємо ланцюжок $\tau(a \cdot m) = \tau(a) \cdot \tau(m) \geq \tau(c) \cdot \tau(m) \geq \tau(c \cdot m)$; лишивши тільки крайні його вирази $\tau(a \cdot m) \geq \tau(c \cdot m)$, отримаємо іншу складову (12) для пар $(a \cdot m, \tau(a \cdot m))$ та $(c \cdot m, \tau(c))$. ■

Примітка до твердження 9. Взаємна простота a і m важлива, без неї твердження неправильне. Наприклад, $(12, 6) \succ (35, 4)$, але якщо домножити 12 та 35 на 16 (не є взаємно простим з 12), вийдуть пари (384, 14) та (560, 20), жодна з них не домінує іншу.

Означення 5. Будемо називати пару $(v, \tau(v))$ ***абсолютно недомінованою***, коли не існує жодної такої пари $(a, \tau(a))$, яка домінувала б пару $(v, \tau(v))$.

Примітка до означення 5. Якщо, враховуючи твердж. 8, вважати, що «$\succ$» з озн. 4 означає «нелінійно більше», то множина абсолютно недомінованих пар з цього озн. 5 є множиною всіх максимумів з множини $\{(n, \tau(n)) \mid n \in \mathbf{N}\}$.

Твердження 10. Абсолютно недомінованими є ті й лише ті пари $(v, \tau(v))$, які утворені з надскладеного (згідно озн. 2) числа v та кількості його дільників.

Доведення. Враховуючи нерівність $a < b$ у (12), для $(a, \tau(a)) \succ (v, \tau(v))$ необхідно (але не достатньо) $a < v$. Якщо v надскладене, жодна пара $(a, \tau(a))$ при $a < v$ не домінує пару $(v, \tau(v))$, бо (1) вимагає $\tau(a) < \tau(v)$, а (12) вимагає $\tau(a) \geq \tau(v)$, що несумісно. Якщо ж

v не надскладене, то $\exists j_{1\le j<v}(\tau(j)\ge\tau(v))$ (бо це заперечення (1)), і можна взяти те $j=j^*$, яке задовольняє квантор існування, і для нього виконуються одночасно $j^*<v$ та $\tau(j^*)\ge\tau(v)$, що за (12) означає $(j^*,\tau(j^*))\succ(v,\tau(v))$, тож $(v,\tau(v))$ не є абсолютно недомінованою. ■

Означення 6. Будемо називати пару $(v,\tau(v))$ ***недомінованою за умови …*** (замість «…» вказується додаткова умова), коли сама пара $(v,\tau(v))$ задовольняє вказану додаткову умову і не існує жодної такої пари $(a,\tau(a))$, для якої теж виконувалася б указана додаткова умова, і яка домінувала б пару $(v,\tau(v))$.

Примітка 1 до означення 6. Дозволяються як ситуація, що існує одна чи кілька пар $(a,\tau(a))$, які домінують $(v,\tau(v))$, але для цих $(a,\tau(a))$ не виконується додаткова умова, так і ситуація, що таких пар не існує. Тому, кожна абсолютно недомінована пара також і недомінована за будь-якої умови, якій задовольняє, а пара, недомінована за деякої умови, може бути абсолютно недомінованою, а може й не бути.

Примітка 2 до означення 6. Додаткова умова не змінює смисл домінування (озн. 4 чи формулу (12)). Змінюється лише множина пар, до яких його застосовують.

Означення 7. Позначимо за $S(N,j)$ (де N — натуральне, j — ціле невід'ємне) множину всіх пар $(v,\tau(v))$, недомінованих за умови «$1\le v\le N$ та в розкладенні v ненульові степені можуть мати лише прості $p_0=2$, $p_1=3$, …, p_j». (Будемо також називати такі умови «умовами для $S(N,j)$».)

Примітка 1 до означення 7. Степені p_0, p_1, …, p_j можуть бути як ненульовими, так і нульовими, як вийде; головне, щоб для p_{j+1}, p_{j+2}, … степені були нульовими.

Примітка 2 до означення 7. З прим. 1 випливає, що коли пара задовольняє умовам для $S(N,j)$, то вона задовольняє також і умовам для $S(N,j+1)$, $S(N,j+2)$, …

Примітка 3 до означення 7. З прим. 2 не випливає, ніби $S(N,0)\subseteq S(N,1)\subseteq\ldots$, бо деякі пари, дозволені в $S(N,j+1)$ і заборонені в $S(N,j)$, можуть домінувати деякі пари з $S(N,j)$. Наприклад: $(8,4)\in S(10,0)$, але $(8,4)\notin S(10,1)$, бо $(6,4)\in S(10,1)$ і $(6,4)\succ(8,4)$.

Твердження 11. Для будь-якого $j\ge0$ та будь-якого $N\ge1$, якщо $(v,\tau(v))\in S(N,j+1)$ і при цьому в розкладенні v показник $k_{j+1}=0$, то $(v,\tau(v))\in S(N,j)$.

Доведення. $(v,\tau(v))\in S(N,j+1)$ означає, що $(v,\tau(v))$ задовольняє умовам для $S(N,j+1)$; раз при цьому $k_{j+1}=0$, то й умовам для $S(N,j)$. Тому, $(v,\tau(v))\notin S(N,j)$ могло би бути, лише якби існувала деяка пара $(a,\tau(a))\succ(v,\tau(v))$, що задовольняла б умови для $S(N,j)$. Але, за прим. 2 до озн. 7, ця $(a,\tau(a))$ задовольняла б також і умови для $S(N,j+1)$. Тоді, враховуючи $(a,\tau(a))\succ(v,\tau(v))$, було б $(v,\tau(v))\notin S(N,j+1)$, що протирічить умові. ■

Твердження 12. Якщо множина S містить лише пари $(a_i,\tau(a_i))$, жодна з яких не домінує жодну іншу пару з S, і впорядкувати (відсортувати) всі пари $(a_0,\tau(a_0))$, $(a_1,\tau(a_1))$, …, $(a_T,\tau(a_T))$ з S, так, щоб виконалося $a_0\le a_1\le\ldots\le a_T$, то автоматично виконаються також $a_0<a_1<\ldots<a_T$ та $\tau(a_0)<\tau(a_1)<\ldots<\tau(a_T)$.

Доведення. $a_0<a_1<\ldots<a_T$ відрізняється від $a_0\le a_1\le\ldots\le a_T$ в точності забороною на повтори однакових значень, а правильність цієї заборони випливає з того, що розглядаємо класичні множини (не мультимножини), а пари $(a_i,\tau(a_i))$ не можуть відрізнятися лише другими елементами, маючи однакові перші, бо a_i визначає $\tau(a_i)$. Ситуація, коли для деяких a_i та a_k (таких, що $(a_i,\tau(a_i))\in S$ та $(a_k,\tau(a_k))\in S$) мають місце водночас $a_i<a_k$ та $\tau(a_i)\ge\tau(a_k)$, неможлива, бо це означало б $(a_i,\tau(a_i))\succ(a_k,\tau(a_k))$, а це заборонено частиною «якщо» цього твердження. Тоді, раз при $a_i<a_k$ можливо лише

$\tau(a_i) < \tau(a_k)$, впорядкування $a_0 < a_1 < \ldots < a_T$ призводитиме до $\tau(a_0) < \tau(a_1) < \ldots < \tau(a_T)$. ■

Примітка до твердження 12. Зокрема, це твердження правильне для всіх $S(N, j)$ при будь-якому $j \geq 0$ та будь-якому $N \geq 1$, бо за озн. 7 кожна $S(N, j)$ якраз і є множиною пар, недомінованих за певних умов. Однак, важливо, що це твердж. 12 виконується не лише для $S(N, j)$, а також і для інших множин пар $(a_i, \tau(a_i))$, в яких жодна пара не домінує жодну іншу пару, навіть якщо нема чітко сформульованої додаткової умови.

Твердження 13. $S(N, 0) = \{(1, 1), (2, 2), (4, 3), \ldots, (2^{\lfloor \log_2 N \rfloor}, \lfloor \log_2 N \rfloor + 1)\}$.

Доведення. В межах умови «розглядаються лише степені двійки» збільшення числ*а* означає збільшення кількості дільників, і для них умова (12) (більша чи рівна кількість дільників у меншого числа) неможлива. А що це вичерпний перелік пар, недомінованих за потрібної для $S(N, 0)$ умови, забезпечується тим, що у множину включили всі пари, де першими елементами є степені двійки вказаного діапазону. ■

Твердж. 13 малокорисне саме по собі, бо мало толку в задачі, обмеженій на самі лише степені двійки. Але воно стає дуже корисним, якщо розглянути його як базу математичної індукції та побудувати й довести відповідний крок індукції.

Твердження 14. Нехай вже відома множина $S(N, j)$, і над нею виконують такі дії:

1. $S_{\text{next}} := S(N, j)$, тобто спочатку нова множина S_{next} складається з усіх пар, що є елементами $S(N, j)$, і лише з них.
2. Для кожної з пар $(a, \tau(a))$, що належать $S(N, j)$, виконати такі дії:
   2.1. перебрати пари $\left(a \cdot p_{j+1}, \tau(a) \cdot 2\right)$, $\left(a \cdot p_{j+1}^2, \tau(a) \cdot 3\right)$, …, $\left(a \cdot p_{j+1}^s, \tau(a) \cdot (s+1)\right)$ (де s можна виразити як $\lfloor \log_{p_{j+1}} (N / a) \rfloor$ чи підібрати циклом `while`), тобто всі пари, де перший елемент не перевищує N і є добутком a на деякий степінь простого p_{j+1}; позначивши ці пари вигляду $\left(a \cdot p_{j+1}^i, \tau(a) \cdot (i+1)\right)$ як $(b, \tau(b))$, для кожної з них виконати такі дії:
   2.1.1. `if` ($(b, \tau(b))$ не домінована жодною з пар, вже наявних у S_{next}) {
   2.1.1.1. включити $(b, \tau(b))$ у S_{next};
   2.1.1.2. `if` ($(b, \tau(b))$ домінує хоча б одну з пар, вже наявних у S_{next})
   2.1.1.2.1. вилучити з S_{next} усі пари, доміновані парою $(b, \tau(b))$;
   }

Тоді утворена цими діями множина S_{next} дорівнює $S(N, j+1)$.

Доведення. Спочатку переконаємося, що ніяка пара, яка не задовольняє умову $S(N, j+1)$, не може потрапити в S_{next}, навіть тимчасово. Всі пари, які потрапляють в S_{next} ініціалізацією в п. 1, задовольняють умову для $S(N, j+1)$, враховуючи прим. 1 до озн. 7. Пари, які потрапляють в S_{next} у п. 2.1.1.1, задовольняють умову для $S(N, j+1)$, бо побудовані з тих, де в розкладенні є лише прості $p_0, p_1, \ldots, p_j$, домноженням на степінь простого p_{j+1}, а умова, що домножений перший елемент $\leq N$, теж забезпечена в п. 2.1.

Переконаємося, що неможлива ситуація, коли деяка пара $(f, \tau(f))$ недомінована за умов для $S(N, j+1)$, але не включена в S_{next}. Випадок, коли таке стається з f, для розкладення якого $k_{j+1} = 0$, неможливий (за твердж. 11, така пара вже є в $S(N, j)$, тож вона копіюється в S_{next} у п. 1). Інакше (для розкладення f маємо $k_{j+1} > 0$), слід дослідити, що саме додається у п. 2.1.1.1. Позначимо степінь з основою p_{j+1}, який входить у f, за r, а частку f / r за g (наприклад, при $f = 1800 = 2^3 \cdot 3^2 \cdot 5^2$, $p_{j+1} = 5$ буде $r = 5^2 = 25$, $g = 2^3 \cdot 3^2 = 72$). Якщо $(g, \tau(g)) \in S(N, j)$, то $(f, \tau(f))$ буде однією з пар $(b, \tau(b))$, які перебиратимуться вкладеними циклами п. 2 та п. 2.1, і в разі успішної перевірки п. 2.1.1 пара $(f, \tau(f))$ буде включена до S_{next} у п. 2.1.1.1 (а в разі неуспішної, її і не слід туди додавати, бо її домінує якась пара (одна чи кілька, це несуттєво), котра, згідно з попереднім абзацем,

задовольняє умову для $S(N, j{+}1)$). Протилежна ситуація $(g, \tau(g)) \notin S(N, j)$ означає, що $(g, \tau(g))$ домінована деякою (щонайменше однією; якщо їх кілька, візьмемо будь-яку одну) парою $(a^*, \tau(a^*))$, належною $S(N, j)$. В такому разі, враховуючи взаємну простоту a^* з r (розкладення a^* не містить ненульових степенів p_{j+1}, тоді як r тільки їх і містить) та твердж. 9, все одно $(r{\cdot}a^*, \tau(r{\cdot}a^*)) \succ (f, \tau(f))$, причому пара $(r{\cdot}a^*, \tau(r{\cdot}a^*))$ буде розглянута у п. 2.1, бо $(a^*, \tau(a^*)) \in S(N, j)$, і, раз $(a^*, \tau(a^*)) \succ (g, \tau(g))$, то $a^* < g$, а отже й $r{\cdot}a^* < r{\cdot}g = f \le N$. Таким чином, ніщо потрібне не буде пропущене й правильність не втратиться, а з точки зору ефективності добре не розглядати безперспективну пару $(f, \tau(f))$. Далі (п. 2.1.1.2.1) вилучаються лише пари, доміновані деякою парою, що задовольняє умову для $S(N, j{+}1)$. Тому, пари, недоміновані за умов для $S(N, j{+}1)$, не будуть вилучені.

Таким чином, S_{next} не може не містити пар, які повинні бути в $S(N, j{+}1)$; лишилося показати, що S_{next} не може містити непотрібних (яких повинно не бути в $S(N, j{+}1)$). Що там не може бути пар, які навіть не задовольняють умову для $S(N, j{+}1)$, вже показано в першому абзаці цього доведення. Лишилося довести, що в S_{next} не може бути водночас і пари $(u, \tau(u))$, і пари $(w, \tau(w))$, які обидві задовольняють умову для $S(N, j{+}1)$, але $(w, \tau(w)) \succ (u, \tau(u))$. За припущенням індукції, в момент ініціалізації $S_{\text{next}} := S(N, j)$ у п. 1 такого не було, і лишається показати, що таке не могло з'явитися при виконанні цих дій. Єдиним (крім ініціалізації у п. 1) місцем, де пари включаються у S_{next}, є п. 2.1.1.1; ситуація, що домінована $(u, \tau(u))$ включається у S_{next}, коли там вже є домінуюча $(w, \tau(w))$, неможлива, бо цього не допустить умова п. 2.1.1; ситуація, що домінуюча $(w, \tau(w))$ включається у S_{next}, коли там вже є домінована $(u, \tau(u))$, тимчасово можлива, але, якщо таке і трапиться у п. 2.1.1.1, то п. 2.1.1.2.1 тут же вилучить $(u, \tau(u))$ з S_{next}, і після завершення п. 2.1.1.2.1 наявність у S_{next} відразу обох пар знов неможлива. ■

Примітка 1 до твердження 14. Для такої побудови $S(N, j{+}1)$ треба **лише** $S(N, j)$, а $S(N, j{-}1)$, $S(N, j{-}2)$, …, $S(N, 0)$ не потрібні (звісно, при $j \ge 1$). Це дозволяє економити пам'ять, починаючи побудову $S(N, j{+}1)$ при $j \ge 1$ з очищення вже не потрібної $S(N, j{-}1)$.

Примітка 2 до твердження 14. Важливо створювати в п. 1 копію множини, а не використовувати одну множину і для циклу п. 2, і для змін у п. 2.1.1.1 та п. 2.1.1.2.1. Хоча б тому, що пари, описані в п. 2.1, правильні, лише якщо a не кратне p_{j+1}.

Примітка 3 до твердження 14. З останнього абзацу доведення випливає, що наприкінці **кожної** ітерації циклу п. 2.1 множина S_{next} містить лише пари, жодна з яких не домінує жодну іншу. □

Теоретично, якщо застосувати твердж. 14 нескінченну кількість разів, вийде недомінованість за умови «задіяні прості p_0, p_1, … (до нескінченності)», тобто абсолютна недомінованість, яка за твердж. 10 є характеристичною властивістю надскладених. Практично, апеляція до нескінченності неконструктивна, але є верхня межа проміжка N, і можна сформулювати та довести твердж. 15 та 16.

Твердження 15. Для будь-якого $j \ge 0$ та будь-якого $N \ge 1$, для кожної пари $(a, \tau(a)) \in S(N, j)$, якщо розкласти перший елемент цієї пари як $a = p_0^{k_0} \cdot p_1^{k_1} \cdot \ldots \cdot p_j^{k_j} \cdot \ldots$, виявиться $k_0 \ge k_1 \ge \ldots \ge k_j \ge k_{j+1} = k_{j+2} = \ldots = 0$.

Доведення. $k_{j+1} = k_{j+2} = \ldots = 0$ слідує з озн. 7. Лишається розібратися зі складовою $k_0 \ge k_1 \ge \ldots \ge k_j \ge 0$ (нетривіальною лише при при $j \ge 1$). Припустимо, ніби є хоч одна пара $(a^*, \tau(a^*)) \in S(N, j)$, така, що в $a^* = p_0^{k_0} \cdot p_1^{k_1} \cdot \ldots \cdot p_j^{k_j}$ має місце $p_u < p_w$ і $k_u < k_w$; тоді можна замінити $p_u^{k_u} \cdot p_w^{k_w}$ на $p_u^{k_w} \cdot p_w^{k_u}$ (лишивши решту степенів незмінними), й отримати таке число a', що $a' < a^*$ (всі множники лишаються додатними, і при цьому $k_w - k_u$ разів замість більшого p_w взято менше p_u), і кількості дільників однакові ($\tau(a^*) = \tau(a')$, бо при обчисленні за (3), обмін місцями множників $(k_w{+}1)$ та $(k_u{+}1)$ не впливає на результат, а решта множників, якщо є, незмінні). Тобто, $(a', \tau(a')) \succ (a^*, \tau(a^*))$. При цьому $a' < N$ (бо

в ланцюжку $a' < a^* \le N$ перша нерівність з'ясована кілька рядків тому, а друга випливає з $(a^*, \tau(a^*)) \in S(N, j)$). Також, $(a^*, \tau(a^*)) \in S(N, j)$ значить, що розкладення a^* не містить ненульових степенів простих, більших p_j, а від заміни, яка перетворює a^* в a', такі степені не з'являться. Тому, $(a', \tau(a'))$ теж задовольняє умови для $S(N, j)$. В поєднанні з $(a', \tau(a')) \succ (a^*, \tau(a^*))$, маємо протиріччя з $(a^*, \tau(a^*)) \in S(N, j)$, і пояснити це протиріччя можна лише хибністю припущення про існування такого a^*. ■

Примітка до твердження 15. Ці твердження й доведення дуже схожі на твердж. 2 і його доведення, але у твердж. 2 йшлося про надскладені, а тут про елементи $S(N, j)$.

Твердження 16. Для кожного скінченного N, існує деяке (залежне від N) скінченне $m_{\max}$, таке, що $S(N, m_{\max}) = S(N, m_{\max}+1) = \ldots = S(N, \infty)$.

Доведення. За алгоритмом побудови $S(N, j+1)$ з твердж. 14, ні за яких N та j $S(N, j+1)$ не може бути власною підмножиною $S(N, j)$: пару можуть вилучити з S_{next} лише в п. 2.1.1.2.1, лише після включення в п. 2.1.1.1 нової домінуючої пари. Однак, з урахуванням твердж. 15, для нової пари має виконуватися $k_0 \ge k_1 \ge \ldots \ge k_j \ge k_{j+1} \ge 1$, а також $p_0^{k_0} \cdot p_1^{k_1} \cdot \ldots \cdot p_j^{k_j} \cdot p_{j+1}^{k_{j+1}} \le N$ (за означенням $S(N, j+1)$). Якщо (аналогічно доведенню насл. 1 твердж. 2 з [6]) перебирати добутки $2 \cdot 3$, потім $2 \cdot 3 \cdot 5$, потім $2 \cdot 3 \cdot 5 \cdot 7$, …, доки добуток не перевищить N, то те значення $j = j^*$, при якому вперше $\prod_{i=0}^{j+1} p_i > N$, можна взяти за $m_{\max}$. Тому що при такому j^* (а також всіх більших) не може бути водночас $k_0 \ge k_1 \ge \ldots \ge k_{j^*} \ge k_{j^*+1} \ge 1$ та $p_0^{k_0} \cdot p_1^{k_1} \cdot \ldots \cdot p_{j^*}^{k_{j^*}} \cdot p_{j^*+1}^{k_{j^*+1}} \le N$ (навіть $p_0^1 \cdot p_1^1 \cdot \ldots \cdot p_{j^*}^1 \cdot p_{j^*+1}^1 > N$, а від збільшення показників добуток може лише збільшитися). ■

Таке $m_{\max}$ є лише оцінкою згори; як правило, відмінності між $S(N, j+1)$ та $S(N, j)$ зникають раніше, бо при досить великих N та j множині $S(N, j)$ належать переважно пари, де кілька початкових $k_0, k_1, \ldots$ більші 1. Теоретично виразити це точніше не здається можливим. Тому, практично доречніше спиратися на наступне твердж. 17; але с*а*ме завдяки твердж. 16 відомо, що цей момент настане, причому досить швидко.

Твердження 17. Коли у процесі знаходження $S(N, 0)$, $S(N, 1)$, … вперше настане ситуація $S(N, j+1) = S(N, j)$, надалі буде $S(N, j) = S(N, j+1) = S(N, j+2) = \ldots = S(N, \infty)$, тож далі цю послідовність можна не будувати.

Доведення. В усіх парах, що належать $S(N, j)$, маємо $k_{j+1} = 0$ (в $S(N, j)$ дійшли до p_j, але не до p_{j+1}); $S(N, j+1) = S(N, j)$ означає, що k_{j+1} ні в якої пари не змінилося, тобто теж дорівнює 0. Отже, $S(N, j+2)$, $S(N, j+3)$, … будуватимуться з пар, де $k_{j+1} = 0$; враховуючи монотонність (твердж. 15), це означає $k_{j+1} = k_{j+2} = \ldots = 0$, тобто нових пар не буде. ■

Примітка 1 до твердження 17. Важливо, щоб однаковими були самі множ*и*ни пар $S(N, j+1)$ та $S(N, j)$, не лише кількості пар у них. Однаковість кількості не виключає ситуації, коли значення пар інші, й це впливає на остаточну відповідь.

Примітка 2 до твердження 17. Теоретично зручно говорити про «перевірку, чи $S(N, j+1) = S(N, j)$», але, раз побудова $S(N, j+1)$ починається з $S_{\text{next}} := S(N, j)$, технічно зручніше мати окрему змінну-прапорець, де відслідковувати, чи була множина змінена.

Твердження 18. Якщо задачу (у постановці № 1) треба розв'язати для кількох (як правило, різних) N_1, N_2, …, N_q, достатньо знайти $N_{\max} = \max(N_1, N_2, \ldots, N_q)$, побудувати один раз множ*и*ни пар $S(N_{\max}, 0)$, $S(N_{\max}, 1)$, …, $S(N_{\max}, j^*)$, $S(N_{\max}, j^*+1)$ (де j^* — те j, при якому вперше досягається $S(N_{\max}, j) = S(N_{\max}, j+1)$), після чого для кожного N_1, N_2, …, N_q знайти в $S(N_{\max}, j^*)$ ту пару, перший елемент якої є найближчим знизу до відповідного N_1, N_2, …, N_q, й це будуть шукані відповіді.

Доведення. $S(N_{\max}, j^*)$ містить усю інформацію, потрібну для знаходження суперскладених з діапазону від 1 до $N_{\max}$, а жодне з чисел $N_1, N_2, \ldots, N_q$ не перевищує $N_{\max}$. ■

Основна ідея технічного подання множин недомінованих пар. Розглянемо впорядковану словникову структуру даних, що підтримує за час $O(\log(size))$ вставку пари (ключ, значення), вилучення пари та пошук пари за ключем (як за точним збігом ключа, так і за запитами «найближчий згори/знизу»). Такими є, зокрема, `map` з C++ STL та `TreeMap` з Java collections. Оскільки пристосування до цієї задачі `map` чи `TreeMap` водночас і нетривіальне, і значно простіше за цілком власну реалізацію спеціалізованої структури даних, а технічні деталі дещо відрізняються навіть для `map` та `TreeMap`, далі наведено опис пристосування до задачі C++ STL `map`. Однак, це лише одна з можливих реалізацій. Більш того, ці ідеї частково спираються на роботу Ф. Препарати [8], написану до появи C++ STL чи Java. Крім того, автор статті дякує М. Рибаку за опис схожої ідеї (в застосуванні до іншої задачі) під час аналізу задач II етапу AUCPC-2007.

Будемо тримати множини недомінованих пар (як-то $S(N, j)$, S_{next} з алгоритму з твердж. 14, інші аналогічні) кожну в своєму примірнику `map`, де ключами є чи*с*ла a, супутніми значеннями кількості їхніх дільників $\tau(a)$.

Твердження 19. Нехай пара $(b, \tau(b))$ подана окремими змінними `long long b` та `int tauB`, а `map`-ом, у якому подана S_{next}, є `map<long long, int> Snext`. Тоді п. 2.1.1 алгоритму з твердж. 14 «`if` пара $(b, \tau(b))$ не домінована жодною з пар, вже наявних у S_{next}», можна реалізувати такими діями (13):

```
auto it = Snext.lower_bound(b);
it--;                                                    (13)
if(it->second < tauB) {… (вкладені підпункти) …}
```

Асимптотична складність цього фрагменту (без вкладених підпунктів) становить $O(\log T)$, де T = `Snext.size()` (тут і в подальших твердженнях).

Доведення. «`Snext.lower_bound(b)`» знаходить ітератор пари, першої (з найменшим ключем) серед тих, де `it->first >= b`; потім, «`it--`» переходить до попередньої пари, ключ якої максимальний серед строго менших b. (Метод `lower_bound` не може вертати значення `Snext.begin()`, бо всі `b` більші 1, а `Snext` містить пару (1, 1), яка кладеться у $S(N, 0)$, копіюється в подальші $S(N, 1)$, $S(N, 2)$, …, й ніколи не вилучається (бо ніщо не домінує її); тому, після `it--` ітератор все ще валідний. Випадок, що `b` перевищить найбільший з наявних у `Snext` ключів і `Snext.lower_bound(b)` поверне значення `Snext.end()`, можливий; але `it--` працює з цим коректно, окремого розгалуження не треба.) Якщо при цьому умова «`it->second < tauB`» хибна, тобто `it->second >= tauB`, то `(it->first, it->second)` $\succ$ `(b, tauB)`, тобто нова пара `(b, tauB)` домінована і її слід пропустити (що буде зроблено, бо `else`-гілки нема).

Ще треба показати, що при істинності «`it->second < tauB`» пару `(b, tauB)` не домінує не лише конкретно пара `(it->first, it->second)`, а й ніяка інша з наявних у `Snext`. Тут важливо, що `Snext` відсортована за зростанням як `first`-ів (бо така властивість C++ STL `map`), так і `second`-ів (за твердж. 12 та прим. 3 до твердж. 14), тож якщо `it->second < tauB`, то пари, ключі яких ще менші, мають ще менші `second`-и, й тому не домінують `(b, tauB)`. Пари, ключі яких більші b, теж не домінують `(b, tauB)`.

Як відомо, кожна з описаних у (13) дій (не враховуючи внутрішність фігурних дужок) працює за $O(\log T)$ дій, де T = `Snext.size()`. ■

Твердження 20. П. 2.1.1.1 алгоритму з твердж. 14 «включити $(b, \tau(b))$ у S_{next}» можна реалізувати очевидним кодом (14):

```
Snext[b] = tauB;                                                    (14)
```

Це, очевидно, має асимптотичну складність $O(\log T)$, де $T =$ `Snext.size()`.

Твердження 21. П. 2.1.1.2 алгоритму з твердж. 14 «`if`$((b, \tau(b))$ домінує хоча б одну з пар, вже наявних у S_{next})» можна реалізувати таким кодом (15):

```
auto it = Snext.upper_bound(b);
if(it != Snext.end() && it->second <= tauB) {… … …}                 (15)
```

При цьому, асимптотична складність (без вкладених підпунктів) становитиме $O(\log T)$.

Доведення. `Snext.upper_bound(b)` шукає пару з мінімальним серед більших b ключем. Істинність умови `if`-а виражає, що така пара існує, причому кіль-ть дільників $\leq$ `tauB`. Тобто, (`b`, `tauB`)$\succ$(`it->first`, `it->second`).

Лишилося перевірити, що за хибності умови `if`-а пара $(b, \tau(b))$ не домінує жодну з пар, вже наявних у S_{next}. Випадок, коли `upper_bound` повертає значення `Snext.end()`, можливий (якщо щойно додана пара (`b`, `tauB`) має найбільший у `Snext` ключ), і в цьому випадку така пара не може домінувати ніяку з пар, вже наявних у S_{next} до (14). Отже, «`it != Snext.end()`» і уникає звернення до невалідного ітератора, і правильно за смислом. Якщо ж причина хибності інша (`it != Snext.end() && it->second > tauB`), то, враховуючи монотонність `second`-ів, раз у найближчої згори пари більше дільників, у подальших їх ще більше, і (`b`, `tauB`) не може домінувати жодну з них. Також, (`b`, `tauB`) не може домінувати й пари, ключі яких менші b.

Виклик `upper_bound` працює за $O(\log T)$, а решта використаних дій за $\Theta(1)$. ■

Твердження 22. П. 2.1.1.2.1 алгоритму з твердж. 14 «вилучити з S_{next} усі пари, домінованої парою $(b, \tau(b))$» можна реалізувати таким кодом (16):

```
while(it != Snext.end()  &&  it->second <= tauB) {
    it = Snext.erase(it); /* вилучає пару ітератора it */
    /* і переставляє it на пару, яка до вилучення була наступною */    (16)
}
```

де початковим значенням `it` є знайдене у (15). Асимптотична оцінка часу виконання цього коду становить гарантовані $O(X \cdot \log T)$, де X — кількість домінованих пар, які будуть вилучені цим циклом, $T =$ `Snext.size()`. Амортизована ж оцінка (смисл поняття див. [5, розд. 17]) при цьому становить $O(\log T)$, тобто не залежить від X.

Доведення. Поки умова у `while` виконується, щоразу має місце аргументація з аналізу фрагмента (15) про те, чому (`b`, `tauB`)$\succ$(`it->first`, `it->second`). Ось цей код і вилучає таку домінована пару. Коли умова перестає виконуватися, той же аналіз фрагмента (15) пояснює, чому `Snext` вже не містить пар, домінованих парою (`b`, `tauB`).

Кожен виклик `Snext.erase(it)` вкладається в $O(\log T)$, тому, X вилучень гарантовано вкладаються у $O(X \cdot \log T)$.

Конкретну пару неможливо вилучити більше одного разу (після вилучення її вже нема в S_{next}), тож сумарна за весь алгоритм кількість вилучень у п. 2.1.1.2.1 менша суми кількостей пар, скопійованих у п. 1, та пар, вставлених у п. 2.1.1.1. У свою чергу, пари, скопійовані в п. 1 при поточній побудові $S(N, j+1)$, не вилучені з S_{next} при попередній побудові $S(N, j)$. Тому, сумарна кількість усіх вилучень відрізняється від сумарної кількості всіх вставок лише на $|S(N, j^*+1)| - |S(N, 0)|$, чим асимптотичний аналіз нехтує у порівнянні, скажімо, з кількістю ітерацій п. 2. Тому, в середньому за весь алгоритм

$X \approx 1$ (як частка кількості всіх вилучень і кількості всіх вставок, які майже однакові). ■

Деякі дрібні оптимізації (14)-(16). Оскільки (15) викликається зразу після (14), можна замінити у (14) `operator[]` на `insert` і оптимізувати (15), замінивши `upper_bound` (складності $O(\log T)$) на ++ (складності $\Theta(1)$), ось так:

```
auto it = (Snext.insert(make_pair(b, tauB)).second)++;
```

(Цей рядок замінює собою і (14), і перший з рядків (15).) Але асимптотика алгоритму в цілому від того не змінюється, а зрозумілість коду погіршується, тому це недоцільно.

Ще, `if`, яким завершується (15), та `while`, яким починається (16), мають однакові умови, тому можна прибрати `if`, бо `while` робить цю перевірку зокрема й на початку циклу. Ця оптимізація більш-менш доречна (але неістотна). □

На жаль, не здається можливим оцінити аналітично кількість $T =$ `Snext.size()`, від якої залежить не лише асимптотична оцінка (13)–(16), а й кількість ітерацій п. 2.

Що ж, раз нема аналітичного вираження цих кількостей, спробуємо дослідити їх експериментально. Реалізацію цього алгоритму можна побачити за посиланням ejudge.ckipo.edu.ua/maxDivsNonDominated1.cpp.

Таблиця 4 — кількості пар у множинах $S(N, 0)$, $S(N, 1)$, … (при значеннях N у межах стандартних цілочисельних типів).

N \ j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	…	19
N \ p_j	2	3	5	7	11	13	17	19	23	29	31	37	41	43	…	71
10^3	10	15	14	15	15											
10^9	30	62	71	66	66	65	65	66	66							
10^{18}	60	156	198	198	194	184	175	171	163	159	156	156	156			
10^{36}	120	389	553	624	627	595	581	537	527	497	489	493	467	441	…	368

Таблиця 5 — експериментально визначені кількісні характеристики множин недомінованих пар (при великих N, з «довгою» арифметикою).

N	10^{20}	10^{50}	10^{100}	10^{200}	10^{500}	10^{1000}
j^* (min, при якому $S(N, j^*) = S(N, j^*+1)$)	12	26	47	84	184	334
p_{j^*}	41	103	223	439	1 103	2 251
$\max\limits_{0 \le j \le j^*+1} \lvert S(N, j) \rvert$	242	1 093	3 511	11 494	56 344	191 555
$\sum_{j=0}^{j^*+1} \lvert S(N, j) \rvert$	2 688	20 703	99 269	476 839	3 733 353	17 553 112
Час (секунд)	0,015	0,156	1,544	15,834	418,316	5 483,411
Час (с) алгоритму 1 з [6]	0,046	32,323	31 647,563	Більший доби, далі не досліжувався		

Перша спроба такого експерименту описана в табл. 4. Рядки цієї таблиці виражають, для якого N будують множ*и*ни недомінованих пар. Стовпчики — номери j простих чисел p_j та відповідних їм $S(N, j)$. Елементи основної частини таблиці означають кількості елементів-пар у відповідній $S(N, j)$. Порожні клітинки виражають, що, згідно твердж. 17, при поточних N та j нема сенсу будувати $S(N, j)$.

Розміри всіх $S(N, j)$ з табл. 4 (навіть для $N = 10^{36}$) досить малі, а виконання майже миттєве. Тому, для вимірювань при ще більших N, алгоритм переписано з використанням довгої арифметики (реалізованої самостійно). Результати див. у табл. 5.

Стовпчики табл. 5 позначають значення N, для яких проведено побудову $S(N, 0)$, $S(N, 1)$, …. Для кожного з цих N окремо, перелік множин $S(N, 0)$, $S(N, 1)$, … будувався «до кінця», тобто доки не наставала ситуація $S(N, j^*) = S(N, j^*+1)$. Рядки табл. 5 виражають, при якому (за номером j^* та значенням p_{j^*}) простому це наставало, якими були максимальна та сумарна кількість елементів-пар у побудованих множинах $S(N, 0)$, $S(N, 1)$, …, $S(N, j^*+1)$, та який час займало обчислення (тактова частота комп'ютера 3,2 ГГц, компілятор g++ (Windows), рівень оптимізацій «–O2», точність вимірювання 1 тік ≈ 0,015 с, усереднене з двох спроб, тобто ті самі, що у [6]). Для повноти порівняння, переписаний (з цією ж реалізацією довгої арифметики) також і алгоритм 1 з [6]. Час виконання нового алгоритму значно менший (при тому, що він знаходить усі надскладені проміжку, а алгоритм з [6] знаходив лише максимальне).

Модифікація алгоритму для постановки № 3а. Аналогічно [6], тут теж досить викреслити K з послідовності простих, і теж час роботи при цьому може або зменшитися, або лишитися незмінним.

Модифікація алгоритму для постановки № 3б. На відміну від [6] (де розгляд складених K потребував великої кількості дивних модифікацій основного алгоритму), тепер істотних модифікацій основного алгоритму менше й вони природніші.

Спочатку варто сформулювати кілька випадків, коли відповідь постановки № 3б можна отримати, розв'язавши задачу в постановці № 1. Ці випадки покривають лише малу частину можливих вхідних даних, але корисні не лише тим, що дозволяють отримати відповідь для цих вхідних даних, а й тим, що дозволяють проводити теоретичні міркування. Твердж. 23–25 очевидні й наводяться без доведень.

Твердження 23. Якщо для деяких N, K задачу розв'язали алгоритмом для постановки № 1 (з тим самим N, проігнорувавши K), і відповідь-значення виявилася не кратною K, то повна відповідь є правильною також і для постановки № 3б.

Твердження 24. Якщо $K > N$, то повна відповідь для постановки № 3б дорівнює відповіді для постановки № 1 (з тим самим N, проігнорувавши K).

Твердження 25. Якщо $K = N$, то повна відповідь для постановки № 3б дорівнює відповіді для постановки № 1 (взявши $N – 1$ замість N, проігнорувавши K).

Твердження 26. Нехай $i' = \max\{ i \mid z_i > 0 \}$ (індекс останнього простого, наявного в розкладенні K у ненульовій степені), j^* є числом, для якого $S(N, j^*) = S(N, j^*+1) = S(N, j^*+2) = \ldots = S(N, \infty)$, і при цьому $j^* < i'$. Тоді відповідь для постановки № 3б дорівнює відповіді для постановки № 1 (з тим самим N, ігноруючи K).

Доведення. З частини «нехай» випливає, що $S(N, \infty) = S(N, j^*)$ при $j^* < i'$; тобто, в усіх парах з $S(N, \infty)$ перший елемент не кратний $p_{i'}$, а отже й не кратний K. Тому, відповідь-значення для постановки № 1 не кратна K, отже застосовне твердж. 23. ■

Означення 8. Позначимо за $R(N, j, K)$ (де N — натуральне число, j — ціле невід'ємне число, K — натуральне число, причому $K \geq 2$) множину всіх пар $(v, \tau(v))$, недомінованих за умови «$1 \leq v \leq N$, в розкладенні v є лише прості $p_0 = 2, p_1 = 3, \ldots, p_j$, та забезпечено, що ні саме v, ні результати його домножень на які б не було степені подальших простих $p_{j+1}, p_{j+2}, \ldots$ не можуть бути кратними K».

Примітка до означення 8. Тут стверджується, що для $(v, \tau(v)) \in R(N, j, K)$ треба $v \leq N$, але для рішення «$(v, \tau(v)) \notin R(N, j, K)$, бо результат домноження v на деякі степені

p_{j+1}, p_{j+2}, … кратний K» неважливо, чи добуток, кратний K, буде меншим, рівним чи більшим N. Наприклад, (4, 3) та (48, 10) однаково належать $S(200, 1)$, але не належать $R(200, 1, 100)$, бо $4 \times 25 = 100 = 1 \times 100$, а $48 \times 25 = 1200 = 12 \times 100$. І тут неважливо, що $100 < 200$, а $1200 > 200$; важливо лише, що і 100, і 1200 кратні 100. □

Озн. 8 може здатися неконструктивним (як перевіряти кратність для відразу всіх результатів домножень v на які б не було степені подальших простих?). Однак, твердж. 27 (очевидне, тому без доведення) та твердж. 28 вирішують цю проблему.

Твердження 27. При натуральних a, b вигляду $a = p_0^{a_0} \cdot p_1^{a_1} \cdot \ldots$ та $b = p_0^{b_0} \cdot p_1^{b_1} \cdot \ldots$, кратність $a \vdots b$ має місце тоді й тільки тоді, коли $\forall i\,(a_i \geq b_i)$.

Твердження 28. Ситуація з озн. 8 «ні саме $v = p_0^{k_0} \cdot p_1^{k_1} \cdot \ldots \cdot p_j^{k_j}$, ні результати його домножень на які б не було степені подальших простих p_{j+1}, p_{j+2}, … не можуть бути кратними $K = p_0^{z_0} \cdot p_1^{z_1} \cdot \ldots$» настає тоді й тільки тоді, коли

$$\exists i^*{}_{0 \leq i^* \leq j}\,(k_{i^*} < z_{i^*}). \tag{17}$$

Доведення. Якщо (17) виконується, то, на які б степені p_{j+1}, p_{j+2}, … не домножали v (тобто, як би не збільшували k_{j+1}, k_{j+2}, …), це не збільшить те k_{i^*} (при $0 \leq i^* \leq j$), при якому $k_{i^*} < z_{*i}$; отже, домножене v не стане кратним K. Якщо ж (17) не виконується, тобто $\forall i\,{}_{0 \leq i \leq j}\,(k_i \geq z_i)$, то в межах $0 \leq i \leq j$ виконується рівно те, що треба згідно твердж. 27, і для кратності $v \vdots K$ лишається забезпечити, щоб $k_i \geq z_i$ виконалося також і для $i = j+1$, $j+2$, …, а це забезпечити дуже легко (наприклад, взяти $k_{j+1} = z_{j+1}$, $k_{j+2} = z_{j+2}$, …). ■

Тепер варто сформулювати й довести твердження 29 та 30, які будуть аналогами тверджень 13 та 14, тільки для $R(N, j, K)$ замість $S(N, j)$.

Твердження 29. $R(N, 0, K)$ складається з пар $\{(1, 1), (2, 2), (4, 3), \ldots, (2^x, x+1)\}$, де $x = \min(\lfloor \log_2 N \rfloor, z_0 - 1)$, а z_0, згідно озн. 3, є показником $p_0 = 2$ в розкладенні K. Зокрема, при $z_0 = 0$ (будь-якому непарному K) та будь-якому N, маємо $R(N, 0, K) = \varnothing$.

Доведення. З одного боку, потрібно задовольнити умову, що v є степенем двійки (містить лише просте $p_0 = 2$), тому $R(N, 0, K)$ не може містити інших пар, крім (згаданих у твердженні 13) пар (1, 1), (2, 2), (4, 3), …, $(2^{\lfloor \log_2 N \rfloor}, \lfloor \log_2 N \rfloor + 1)$. З іншого боку, коли дозволене лиш одне просте p_0, формула (17) з твердж. 28 спрощується до $k_0 < z_0$. Щоб задовольнити обидві вимоги одночасно, слід вибрати мінімум.

Також, при $z_0 = 0$ не існує жодного значення показника k_0, яке було б водночас і цілим невід'ємним (це потрібно для будь-якого розкладення), і меншим z_0. ■

Наслідок з твердження 29. Якщо $K < N$, останньою парою $R(N, 0, K)$ є $(2^{z_0 - 1}, z_0)$.

Доведення. Оскільки $z_0 - 1$ є одним з аргументів мінімума, досить довести, що з $K < N$ випливає $z_0 - 1 \leq \lfloor \log_2 N \rfloor$. Припустимо протилежне, тобто $\lfloor \log_2 N \rfloor < z_0 - 1$. Розглянемо очевидну нерівність $a - 1 < \lfloor a \rfloor$ і підставимо $\log_2 N$ в якості a; вийде $(\log_2 N) - 1 < \lfloor \log_2 N \rfloor$. Тоді з припущення випливає $(\log_2 N) - 1 < z_0 - 1$, тобто $\log_2 N < z_0$, тобто $N < 2^{z_0}$. Оскільки K дорівнює добутку $2^{z_0} = p_0^{z_0}$ на $p_1^{z_1} \cdot p_2^{z_2} \cdot \ldots$, це означає $N < K$, що протирічить $K < N$. Отже, припущення неправильне, й $x = z_0 - 1$. ■

Твердження 30, скорочене формулювання. Щоб побудувати $R(N, j{+}1, K)$, слід, по-перше, застосувати такі ж дії, як у твердж. 14 до $S(N, j)$, але до $R(N, j, K)$; якщо $z_{j+1} > 0$, то, по-друге, слід застосувати також аналогічні дії до $S(N, j)$, але з поправками: при домноженнях пар починати не з множника p_{j+1}, а з $p_{j+1}^0 = 1$, та закінчувати, враховуючи як вимогу «домножене число не перевищує N», так і вимогу «$k_{j+1} < z_{j+1}$».

Твердження 30, детальне формулювання. Нехай вже відомі множини пар $R(N, j, K)$ та $S(N, j)$, і над ними виконують такі дії:

1. $R_{\text{next}} := R(N, j, K)$, тобто спочатку нова множина R_{next} складається з усіх пар, що є елементами $R(N, j, K)$, і лише з них.
2. Для кожної з пар $(a, \tau(a))$, що належать $R(N, j, K)$, виконати такі дії:
   2.1. перебрати пари $\left(a \cdot p_{j+1}, \tau(a) \cdot 2\right)$, $\left(a \cdot p_{j+1}^2, \tau(a) \cdot 3\right)$, …, $\left(a \cdot p_{j+1}^s, \tau(a) \cdot (s+1)\right)$ (де s можна виразити як $\lfloor \log_{p_{j+1}}(N/a) \rfloor$ чи підібрати циклом `while`), тобто всі пари, де перший елемент не перевищує N і є добутком a на деякий степінь простого p_{j+1}; позначивши ці пари вигляду $\left(a \cdot p_{j+1}^i, \tau(a) \cdot (i+1)\right)$ як $(b, \tau(b))$, для кожної з них виконати такі дії:
      2.1.1. `if` ($(b, \tau(b))$ не домінована жодною з пар, вже наявних у R_{next}) {
        2.1.1.1. включити $(b, \tau(b))$ у R_{next};
        2.1.1.2. `if` ($(b, \tau(b))$ домінує хоча б одну з пар, вже наявних у R_{next})
          2.1.1.2.1. вилучити з R_{next} усі пари, доміновані парою $(b, \tau(b))$;
      }
3. Якщо $z_{j+1} > 0$, то:
   3.1. Для кожної з пар $(a, \tau(a))$, що належать $S(N, j)$, виконати такі дії:
      3.1.1. перебрати пари $(a, \tau(a))$, $\left(a \cdot p_{j+1}, \tau(a) \cdot 2\right)$, $\left(a \cdot p_{j+1}^2, \tau(a) \cdot 3\right)$, …, $\left(a \cdot p_{j+1}^s, \tau(a) \cdot (s+1)\right)$ (де s можна підібрати циклом `while` чи виразити як $\min\left(z_{j+1} - 1, \lfloor \log_{p_{j+1}}(N/a) \rfloor\right)$), тобто всі пари, де перший елемент є добутком a на деякий (можливо, 0-й) степінь простого p_{j+1}, і при цьому добуток не перевищує N, а показник не перевищує $z_{j+1} - 1$; позначивши ці пари вигляду $\left(a \cdot p_{j+1}^i, \tau(a) \cdot (i+1)\right)$ як $(b, \tau(b))$, для кожної з них виконати такі дії:
        3.1.1.1. `if` ($(b, \tau(b))$ не домінована жодною з пар, вже наявних у R_{next}) {
          3.1.1.1.1. включити $(b, \tau(b))$ у R_{next};
          3.1.1.1.2. `if` ($(b, \tau(b))$ домінує хоча б одну з пар, вже наявних у R_{next})
            3.1.1.1.2.1. вилучити з R_{next} усі пари, доміновані парою $(b, \tau(b))$;
        }

Тоді утворена цими діями множина R_{next} дорівнює $R(N, j{+}1, K)$.

Ідея доведення. Доведення в цілому аналогічне доведенню твердж. 14 (алгоритм за деяких обставин додає та/або вилучає інші пари, але аргументація, чому додані пари справді слід додати, а вилучені справді слід вилучити, аналогічна.) Головне, що потребує нових міркувань — використання при побудові $R(N, j{+}1, K)$ як $R(N, j, K)$, так і $S(N, j)$, та потреба забезпечити умову «результати домножень першого елемента на які б не було степені подальших простих p_{j+1}, p_{j+2}, … не будуть кратними K». Для всіх пар з $R(N, j, K)$ ця некратність вже забезпечена, тож домноження цих пар на який би не було степінь простого p_{j+1} не можуть її порушити, й можна діяти так само, як у алгоритмі з твердж. 14. При використанні ж пар з $S(N, j)$ некратність забезпечується тим, що

додаткові множники обмежені, щоб забезпечити $k_{j+1} < z_{j+1}$. □

Примітки. Для твердж. 30 дійсні аналоги всіх приміток до твердж. 14. □

При реалізації алгоритму з твердження 30 можна взяти за основу фрагменти коду (13)–(16), але вони потребують деяких правок. Наприклад, в аналізі (13) сказано, що `Snext.upper_bound(b)` не може вертати значення `Snext.begin()`, але тепер, в аналогічному коді для п. 3.1.1.1, при $R(N, j, K) = \varnothing$ `Rnext.upper_bound(b)` може вертати значення `Rnext.begin()`. Також, тепер треба переконатися в правильності обробки ситуації, коли одна й та сама пара (`b`, `tauB`) перевіряється як у п. 2.1.1, так і в п. 3.1.1.1 (в алгоритмі з твердж. 14 основна теорема арифметики про єдність розкладення робила таку ситуацію неможливою). Для скорочення обсягу, такі деталі пропущені. Охочі можуть прочитати код за посиланням ejudge.ckipo.edu.ua/…

Ще слід з'ясувати, до яких пір треба повторювати перетворення $R(N, j, K)$ та $S(N, j)$ у $R(N, j+1, K)$. Використовувати тут незмінне твердж. 17 неправильно, бо деякі з використаних у доведеннях твердж. 15–17 властивості для постановки № 3б не виконуються. Однак, можна сформулювати й довести твердж. 31–33, котрі є правильними для постановки № 3б аналогами тверджень 15–17.

Твердження 31. Для будь-якого $j \geq 0$, будь-якого $N \geq 1$ та будь-якого $K \geq 2$, для кожної пари $(a, \tau(a)) \in R(N, j, K)$, якщо розкласти перший елемент цієї пари як $a = p_0^{k_0} \cdot p_1^{k_1} \cdot \ldots \cdot p_j^{k_j} \cdot \ldots$, а потім викреслити з цього розкладення степені тих простих, для яких $z_i > 0$, то послідовність тих k_i, що залишаться, буде монотонно незростаючою.

Доведення. Самі твердж. 15 і його доведення непридатні, бо від заміни $p_u^{k_u} \cdot p_w^{k_w}$ на $p_u^{k_w} \cdot p_w^{k_u}$ число може стати кратним K, не бувши кратним до заміни. Однак, невикресленими лишаються тільки такі u та w, для яких $z_u = z_w = 0$, а для них ця заміна не впливає на кратність K, тож для них можна повторити міркування доведення твердж. 15. ■

Твердження 32. Для кожного скінченного N та кожного скінченого K, існує деяке (залежне від N та K) скінченне $m_{\max}$, таке, що, починаючи з нього, множ*и*ни недомінованих пар рівні: $R(N, m_{\max}, K) = R(N, m_{\max}+1, K) = \ldots = R(N, \infty, K)$.

Доведення. Проведемо міркування, аналогічні доведенню твердж. 16, але лише для тих простих, де $z_i = 0$ (завдяки твердж. 31 відомо, що серед них має місце монотонність, тому для них міркувати аналогічно можна). ■

Твердження 33. Нехай $i' = \max\{ i \mid z_i > 0 \}$ (індекс останнього простого, наявного в розкладенні K у ненульовій степені). Якщо побудувати послідовність $R(N, 0, K)$, $R(N, 1, K)$, … щонайменше до $R(N, i', K)$ включно, а вже серед $j \geq i'$ почати перевірки, чи $R(N, j+1, K) = R(N, j, K)$, то після першої ж такої рівності буде $R(N, j, K) = R(N, j+1, K) = R(N, j+2, K) = \ldots = R(N, \infty, K)$, тож далі цю послідовність можна не продовжувати.

Доведення. З обмеження $j \geq i'$ випливають такі наслідки: (1) для них $R(N, j+1, K)$ будуються на основі лише $R(N, j, K)$, без використання $S(N, j)$; (2) встановлена у твердж. 31 монотонність тих з k_j, де $z_j = 0$, при $j \geq i'$ стосується всіх підряд таких j. Тому, в цих умовах можна повторити міркування, аналогічні доведенню твердж. 17. ■

Примітки 1–2 до твердження 33. Аналогічні приміткам до твердж. 17.

Примітка 3 до твердження 33. Не можна відкидати вимогу $j \geq i'$ і стверджувати, ніби «як тільки вперше настало $R(N, j+1, K) = R(N, j, K)$, так далі всі множини будуть рівними». Хоча б тому, що при $z_{j+1} > 0$ множина $R(N, j+1, K)$ залежить не лише від

$R(N, j, K)$, а й від $S(N, j)$. Це можна підтвердити й прикладом: $R(100, 0, 35) = R(100, 1, 35) = \varnothing$, але $R(100, 2, 35) = S(100, 1) \neq \varnothing$.

Примітка 4 до твердження 33. Практично доцільно мати **дві** умови завершення алгоритму: одна — на основі цього твердж. 33; інша — на основі поєднання тверджень 17 та 26. Тобто: якщо $S(N, j^*+1) = S(N, j^*)$ (твердж. 17) настає раніше, чим i' (останній з показників $z_i > 0$), то отримати відповідь з $S(N, j^*)$; якщо, навпаки, i' настає раніше, то скористатися поточним твердж. 33.

Примітка 5 до твердження 33. Якщо $j \geq i$' (тобто, побудова $S(N, 0)$, $R(N, 0, K)$, $S(N, 1)$, $R(N, 1, K)$, … вже перейшла через i'), а умова $S(N, j^*+1) = S(N, j^*)$ (з твердження 17) все ще не настала, практично доцільно припиняти побудову $S(N, j^*)$, бо всі подальші $R(N, j+1, K)$ все одно вже не залежатимуть від $S(N, j)$. □

Таблиця 6 — експериментально визначені кількісні характеристики множин недомінованих пар (при великих N, складених K).

	N	10^{20}	10^{50}	10^{100}	10^{200}	10^{500}	10^{1000}
$K = 40 = 2^3 \times 5$	Max $\|S\|, \|R\|$	179	652	2 217	7 724	40 144	139 988
	Sum $\|S\|, \|R\|$	1 845	12 861	62 555	305 569	2 473 648	12 130 863
	Час (с)	0,101	0,226	1,005	9,492	259,265	3 601,578
$K =$ $=829635693357621875=$ $=5^5 \times 17^4 \times 29^3 \times 47^2 \times 59$	Max $\|S\|, \|R\|$	242	1 093	3 511	11 494	56 344	191 555
	Sum $\|S\|, \|R\|$	4 993	32 210	136 503	599 747	4 313 155	19 441 352
	Час (с)	0,108	0,350	2,363	21,902	533,569	6 598,116
$K =$ $=2677277333530800000=$ $=2^7 \times 3^6 \times 5^5 \times 7^4 \times 11^3 \times 13^2 \times 17$	Max $\|S\|, \|R\|$	242	1 093	3 511	11 494	55 033	180 375
	Sum $\|S\|, \|R\|$	3 654	23 985	106 048	473 878	3 582 036	16 601 348
	Час (с)	0,031	0,288	1,793	16,348	405,571	5 259,354
$K =$ $=235920057993400000=$ $=2^6 \times 5^5 \times 11^4 \times 17^3 \times 23^2 \times 31$	Max $\|S\|, \|R\|$	242	1 093	3 511	11 494	56 344	191 555
	Sum $\|S\|, \|R\|$	4 377	27 733	120 255	539 377	3 981 625	18 053 569
	Час (с)	0,101	0,343	2,121	19,406	477,877	5 979,104
$K =$ $=2817320268577887089=$ $=37^4 \times 83^3 \times 127^2 \times 163$	Max $\|S\|, \|R\|$	242	1 093	3 511	11 494	56 344	191 555
	Sum $\|S\|, \|R\|$	3 219	31 879	150 667	654 420	4 716 919	21 068 885
	Час (с)	0,093	0,318	2,363	21,551	539,193	6 907,099
$K = 97033579375 =$ $=5^4 \times 11^3 \times 31^2 \times 997$	Max $\|S\|, \|R\|$	242	1 093	3 511	11 494	56 344	191 555
	Sum $\|S\|, \|R\|$	5 002	39 792	192 637	931 964	7 146 338	29 476 297
	Час (с)	0,101	0,350	3,221	30,980	798,833	9 565,622
$K=321389516527343410=$ $=2\times5\times1979\times1993\times1999\times$ $\times2011\times2027$	Max $\|S\|, \|R\|$	242	1 093	3 511	11 494	56 344	191 555
	Sum $\|S\|, \|R\|$	4 866	38 159	183 760	882 385	6 957 321	32 256 608
	Час (с)	0,015	0,312	2,854	29,452	790,792	10 178,284
$K = 2017 = 2017^1$	Max $\|S\|, \|R\|$	242	1 093	3 511	11 494	56 344	191 555
	Sum $\|S\|, \|R\|$	2 688	20 703	99 269	476 839	3 733 353	17 579 983
	Час (с)	0,008	0,164	1,575	16,208	422,504	5 526,303

Для аналізу того, як усе це вплинуло на час роботи, проведено експерименти, аналогічні тому, підсумки якого наведені в табл. 5, але при деяких K. Вміст табл. 6 аналогічний табл. 5, лише кількісні характеристики множин $S(N, j)$ замінено на характеристики фактично використаних $S(N, j)$ та $R(N, j, K)$ при деяких K. Параметри компіляції та комп'ютер використані ті самі, що для табл. 5. При побудові $R(N, j, K)$ та $S(N, j)$ використані обидві умови завершення циклу (і тверж. 17 та 26, і 33). Прим. 5 до твердж. 33 теж використана. Під «max $|S|, |R|$» мається на увазі максимальна серед кількостей пар деякої реально побудованої $S(N, j)$ чи $R(N, j, K)$, а під «sum $|S|, |R|$» — сумарна кількість пар усіх реально побудованих $S(N, j)$ та $R(N, j, K)$.

Бачимо, що максимальні та сумарні розміри $R(N, j, K)$ та $S(N, j)$, а також час роботи, не сильно відрізняються від відповідних (при тому ж N) розмірів $S(N, j)$, в межах інтуїтивно очікуваного «кілька разів». Випадки, коли вони виявляються навіть меншими, чим у табл. 5, можна пояснити тим, що завдяки прим. 5 до твердження 33

іноді можна не будувати значну частину $S(N, j)$; якщо при цьому $R(N, j, K)$ містять менше, чим $S(N, j)$, пар, то в таких випадках обсяг перебору зменшується.) Природно й те, що більшість величин виявилися мінімальними при $K = 40 = 2^3 \times 5$ (коли і виникли обмеження на степені $p_0 = 2$ та $p_2 = 5$ в $R(N, j, K)$, і дуже швидко припинили обчислювати $S(N, j)$), а максимальними при $K = 321389516527343410 = 2 \times 5 \times 1979 \times 1993 \times 1999 \times \times 2011 \times 2027$ (коли довелося дуже довго обчислювати і $S(N, j)$, і $R(N, j, K)$).

Перспективи продовження досліджень ідей цієї статті. Алгоритм дозволяє додаткові оптимізації, вплив яких ще не досліджений. Зокрема, такі.

1) Для постановки № 1 мало місце твердження 18, за яким можна було один раз побудувати серію множин $S(N_{\max}, 0)$, $S(N_{\max}, 1)$, …, і знаходити відповіді для всіх N_i. Очевидно, для постановки № 3б це так само, коли є різні N_i при однакових K, але коли є різні K_i, ситуація інакша, бо послідовність множин $R(N, 0, K)$, $R(N, 1, K)$, … будується для конкретного K. Однак, варто дослідити умови, за яких такі послідовності множин для різних K_i можна будувати, повторно використовуючи однакові проміжні частини. Наприклад, перша відмінність між $K_1 = 2^3 \times 5^2 \times 17 = 3400$ та $K_2 = 2^3 \times 5^2 \times 19 = 3800$ лише в z_6, відповідному $p_6 = 17$, а $z_0, z_1, \ldots, z_5$ однакові, тому й $R(N, 0, K)$, $R(N, 1, K)$, …, $R(N, 5, K)$ для $K_1 = 3400$ та $K_2 = 3800$ теж однакові (а з $R(N, 6, K)$ в разі великого N починається відмінність). Таким однаковим частинам сприяють також твердження 6 з [6] та його наслідок, які в деяких ситуаціях дозволяють перетворити різні K_i в однакові зменшені K_i'. Як краще організувати такі повторні використання, наскільки помітне вони можуть дати загальне прискорення і наскільки сильно заради них слід жертвувати економією пам'яті з прим. 1 до твердж. 30 — варто дослідити.

2) Твердж. 23–26 та прим. 4 до твердж. 33 згадують випадки, коли відповідь постановки № 3б можна отримати, застосувавши простіший алгоритм для постановки № 1; однак, цей перелік не є ні вичерпним, ні конструктивним. Можна спробувати дослідити глибше, коли варто пробувати розв'язати алгоритмом для постановки № 1, і лише якщо відповідь кратна K, то переходити до основного алгоритму для постановки № 3б, а коли відразу діяти за алгоритмом для постановки № 3б.

3) Було розглянуто реалізацію, де множини недомінованих пар подаються як C++ STL `map`, у якому ключами є ч*и*сла-значення, а `second`-ами — кількості дільників цих чисел. Однак, імовірно, що потрібні алгоритму властивості можна переформулювати й заново довести, організувавши `map` «навпаки», щоб ключами були кількості дільників, а відповідними супутніми значеннями мінімальне з чисел, яке має таку кількість дільників, потім переробити відповідно сам алгоритм. При цьому міркування стануть заплутанішими, та й програма, мабуть, ускладниться. Зате це може дати пришвидшення за рахунок того, що кількості дільників значно менші, чим ч*и*сла, які мають відповідну кількість дільників, а алгоритм робить багато викликів `upper_bound` / `lower_bound` (отже, багато порівнянь ключів). Для проміжків, де ч*и*сла-значення вже потребують довгої арифметики, а кількості дільників ще поміщаються у стандартні типи, слід сподіватися особливо помітного пришвидшення. Там, де кількості дільників теж потреб*у*ють довгої арифметики, слід очікувати меншого, але теж пришвидшення, бо час порівняння у довгій арифметиці залежить від довжин чисел.

Висновки

У цій статті запропоновано принципово інший, оснований на ідеї недомінованих пар, алгоритм вирішення тієї самої задачі, що в [6] — знаходження надскладених чисел, тобто тих, які мають рекордні кількості дільників, а також узагальнення цієї задачі, коли максимум шукається не серед усіх чисел проміжку, а лише серед не кратних деякому K. На відміну від алгоритму з [6], запропонований тут алгоритм не здається

можливим узагальнити на випадок, коли проміжок починається не з 1, а з довільного $M \le N$. Однак, якщо розглядати лише проміжки від 1 до N, запропонований тут алгоритм ефективніший за розібрані в [6], а узагальнення для складених K природніше.

Список використаної літератури:

1. Стаття «Надскладене число» в українській вікіпедії. Режим доступу https://uk.wikipedia.org/wiki/Надскладене_число.
2. Бухштаб, А. А. Теория чисел. / А. А. Бухштаб. — М.: Просвещение, 1966 — 392 с.
3. Волошин, О. Ф. Моделі та методи прийняття рішень. / О. Ф. Волошин, С. О. Мащенко — К.: Видавничо-поліграфічний центр «Київський університет», 2010. — 336 с.
4. Дирихле, П. Г. Л. Лекции по теории чисел. В обработке и с добавлениями Р. Дедекинда. / П. Г. Л Дирихле. — М.: Объединённое научно-техническое издательство НКТП СССР, 1936 — 404 с.
5. Кормен, Т. Вступ до алгоритмів. / Томас Г. Кормен, Чарлз Е. Лейзерсон, Роналд Л. Рівест, Кліфорд Стайн. — К., К.І.С., 2019 — 1288 стор.
6. Порубльов, І. М. Деякі алгоритми пошуку чисел з максимальною кількістю дільників / Вісник Черкаського національного університету, серія «Прикладна математика. Інформатика». Випуск № 1.2020, с. 44–60.
7. Debreu, Gerard. Valuation Equilibrium and Pareto Optimum. / Proceedings of the National Academy of Sciences of the United States of America, Vol. 40, No. 7 (Jul. 15, 1954), pp. 588–592.
8. Preparata F. P., An optimal real-time algorithm for planar convex hulls / Communications of the ACM, Volume 22, Issue 7, 1979.

References:

1. The Article "Supercomposite number". Access mode: https://uk.wikipedia.org/wiki/Надскладене_число [in Ukrainian].
2. Bukhshtab, A.A. (1966) Theory of numbers. – M.: Education. – 392 p. [in Russian]
3. Voloshin, O.F. (2010) Models and methods of Decision-Making. – K.: Kyiv university publishing center. – 336 p. [in Ukrainian]
4. Dirichlet, P.G.L. (1936) Lectures on number theory. In processing and with additions by R. Dedekind. – M.: United Scientific and Technical Publishing House of the NKTP of the USSR. – 404 p. [in Russian]
5. Cormen T., Leiserson C., Rivest R., Stein C. (2019) Introduction to Algorithms. – K.: K. I. C. – 1288 p. [in Ukrainian].
6. Porublyov, I. Some algorithms for searching numbers with maximal quantity of divisors // Bulletin of the Cherkasy Bohdan Khmelnytsky National University. Series "Applied Mathematics. Informatics". Issue #1.2020, pp. 44–60.
7. Debreu, Gerard. Valuation Equilibrium and Pareto Optimum. / Proceedings of the National Academy of Sciences of the United States of America, Vol. 40, No. 7 (Jul. 15, 1954), pp. 588–592.
8. Preparata F. P., An optimal real-time algorithm for planar convex hulls / Communications of the ACM, Volume 22, Issue 7, 1979.

PORUBLYOV Ilya,
Lecturer, Bohdan Khmelnytsky National University of Cherkasy, Ukraine

YET ANOTHER ALGORITHM FOR SEARCH NUMBERS WITH MAXIMAL QUANTITY OF DIVISORS (BASED ON IDEA OF NON-DOMINATED PAIRS SET)

Summary. Introduction. *This article is essentially a continuation of the article [6]. It considers almost the same formulation of the problem (search for number with maximal quantity of divisors among all numbers in given interval, but this time only from 1 to given N, and it's necessary to find minimal of the numbers which have the maximal quantity of divisors). Despite similar formulation, a fundamentally different way of solving is proposed. It's based on the introduced concept of non-dominated pairs, namely: a pair (number, quantity of its divisors) is considered non-dominated iff there are no smaller numbers that have greater or equal quantity of divisors. Similarly to [6], this article considers generalization when maximum quantity of divisors is sought not among all numbers in the interval, but only among non-multiples of some K (natural, greater than or equal to 2, not necessarily prime).*

Purpose. *The purpose of the article is to describe the designed algorithm for solving problem of searching numbers with maximal quantity of divisors, based on sets of non-dominated pairs, and its variations. Similarly to [6], several problem formulation versions are considered:*

1. Given a positive integer N, for all integers in range between 1 and N (both inclusive), find number having maximal (among numbers in the range) quantity of divisors.

3. Given positive integers N and K ($K \geq 2$), for integers in range between 1 and N (both inclusive), but excluding multiples of K, find number having maximal (among the said numbers) quantity of divisors.

Results and conclusion. *After introducing definitions "pair (v, τ(v)) is non-dominated under constraints … (where … is replaced with some constraints) iff it satisfies the constraints and there is no satisfying the constraints pair (w, τ(w)) where $w < v$ and $\tau(w) \geq \tau(v)$" and "S(N, j) is set of all pairs (v, τ(v)), non-dominated under constraints $v \leq N$ and factorization of v contains non-zero degrees for $p_0 = 2$, $p_1 = 3$, …, p_j", an inductive algorithm building S(N, 0), S(N, 1), … is proposed and proved; theoretical and practical estimate for such j* that S(N, j*) = S(N, j*+1) = … = S(N, ∞) are proposed and proved. Algorithm finding high composite numbers using S(N, 0), S(N, 1), … is implemented. Comparing this algorithm with same-goal algorithm from [6], new one found to be much faster to build all highly-composite numbers in range 1..N, than algorithm from [6] to find maximal only.*

Modification for task formulation 3 is based on introducing new series of sets R(N, j, K) which are sets of all pairs (v, τ(v)), non-dominated under constraints $v \leq N$ and factorization of v contains non-zero degrees for $p_0 = 2$, $p_1 = 3$, …, p_j, and (what differs R(N, j, K) from S(N, j)) v can't become a multiple of K after multiplying with any product of any powers of p_{j+1}, p_{j+2}, …. Inductive algorithm building S(N, 0), R(N, 0, K), S(N, 1), R(N, 0, K), … is proposed and proved; theoretical and practical estimate for finishing the process are proposed and proved. This algorithm solves problem in formulation 3, consuming amounts of time and memory, comparable to algorithm for formulation 1.

Keywords: *quantity of divisors, highly composite numbers, non-dominated pairs, ordered dictionary data structure, map, lower_bound, upper_bound.*

Одержано редакцією 18.10.2021 р.
Прийнято до публікації 08.12.2021 р.

УДК 519.248

DOI 10.31651/2076-5886-2021-1-49-57

PACS 07.05.Tp, 62.20.M-, 85.40.Qx, 61.43.Bn, 64.60.De

ПАСІЧНИЙ Микола Олександрович,
кандидат фізико-математичних наук,
доцент, завідувач кафедри фізики,
Черкаський національний університет
імені Богдана Хмельницького
e-mail: pasichnyy@ukr.net
ORCID: 0000-0002-8434-1544

ТАТАРЧУК Євгеній Вікторович,
кандидат фізико-математичних наук,
доцент, доцент кафедри фізики,
Черкаський національний університет
імені Богдана Хмельницького
e-mail: etatar@ukr.net
ORCID: 0000-0001-5885-2079

МОДЕЛЮВАННЯ РОЗМІРНОГО ЕФЕКТУ У ПРОЦЕСІ ВІДМОВ МІКРОЕЛЕКТРОННИХ ПРИСТРОЇВ НА ОСНОВІ ПІДХОДУ ФОРМУВАННЯ ПЕРКОЛЯЦІЙНОГО КЛАСТЕРА

Створено комп'ютерну модель та досліджено процес відмов на основі підходу перколяційного кластера. Встановлено показникові залежності середнього часу до відмови, стандартного відхилення та коефіцієнта асиметрії розподілів часів до відмов від розміру системи.

***Ключові слова**: розмірний ефект, час до відмови, перколяційний кластер.*

Вступ

Компонентна база сучасної мікроелектронної промисловості, основою функціонування якої є використання інтегральних мікросхем на базі напівпровідникових кристалів (чипів), стрімко удосконалюється і ускладняється [1]. Розвиток сучасних електронних пристроїв спрямований на мініатюризацію і зменшення лінійних розмірів елементної бази мікроелектронних компонентів. У процесі експлуатації відбувається деградація складових компонент і в певний момент часу це призводить до виходу пристрою з ладу [2]. Поширеними причинами відмов у роботі мікроелектронних пристроїв є вихід з ладу елементів у місцях електричного з'єднання провідників внаслідок утворення тріщини або пор [3] за рахунок термоміграції, електроміграції, термонапруг та інших явищ [4, 5]. Зменшення розмірів електронних компонент призводить до зростання густини струму через окремі елементи і інтенсифікацію процесів деградації електричного контакту [6]. Разом з цим вимоги щодо надійності експлуатації електронних пристроїв також зростають. З огляду на концепцію мініатюризації елементної бази сучасних електронних пристроїв розуміння загальної поведінки характеристик процесу відмов електронних систем при зменшенні їх розміру є актуальним і важливим з прикладної точки зору.

Метою статті є розробка комп'ютерної моделі і статистичне дослідження поведінки основних характеристик процесу відмов у залежності від розміру системи.

Для дослідження розмірного ефекту здійснено серію комп'ютерних експериментів процесу відмови модельної системи і проведено статистичний аналіз розподілів часів до відмов для встановлення характерних залежностей основних характеристик розподілів для модельних систем різного розміру.

Виклад основного матеріалу

1. Основні положення моделі та методика експерименту

В основу комп'ютерної моделі покладено аналіз можливості проходження електричного струму через двовимірну систему розмірами N * N вузлів у формі квадратної решітки. Основні положення моделі:

- Кожен вузол має 4 найближчих сусідів.
- У початковому стані всі вузли системи заповнені елементами.
- Струм може протікати лише при наявності елементів у двох сусідніх вузлах.
- На кожній ітерації видаляємо елемент з випадково обраного вузла системи.
- Часом відмови вважатимемо відношення кількості ітерації до загальної кількості вузлів системи, коли зникає провідний кластер між лівою і правою границями модельної системи.

На основі описаної моделі проводилася серія комп'ютерних експериментів і за результатами формувався розподіл часів до відмов для модельної системи певного розміру N * N вузлів.

Розглянута модель еквівалентна задачі перколяції вузлів. Утворення перколяційного кластера вузлів видалених елементів модельної системи відповідатиме часу до відмови.

Для комп'ютерної реалізації моделі процесу відмов було використано середовище програмування Embarcadero Delphi 10.4 з ліцензією Community Edition.

2. Результати комп'ютерного моделювання і їх аналіз

На рисунках 1-8 представлені результати проведення серії комп'ютерних експериментів у вигляді розподілів часів до відмов для модельних систем різного

розміру. Для кожного розміру модельної системи було виконано 100000 модельних випробувань. Для аналізу розподілів часів до відмов було використано такі основні характеристики:

– математичне сподівання або середнє значення;
– середнє квадратичне відхилення або Стандартне відхилення – характеризує відхилення значень розподілу від середнього значення;
– коефіцієнт асиметрії (Skewness) – характеризує ступінь асиметрії розподілу відносно середнього значення. Додатне значення коефіцієнта асиметрії вказує на розподіл довша частина якого знаходиться праворуч від середнього значення і навпаки;
– коефіцієнт ексцесу (Kurtosis) — характеризує відносну гостроту розподілу відносно нормального розподілу. Додатний ексцес характеризує відносно гострий розподіл.

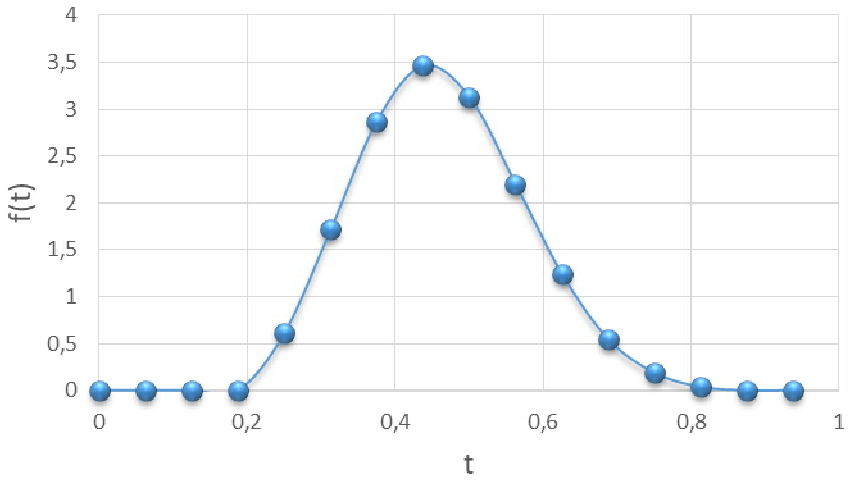


Рис. 1. Розподіл часів до відмов за результатами комп'ютерного моделювання для системи розміром N * N = 4 * 4 вузлів.

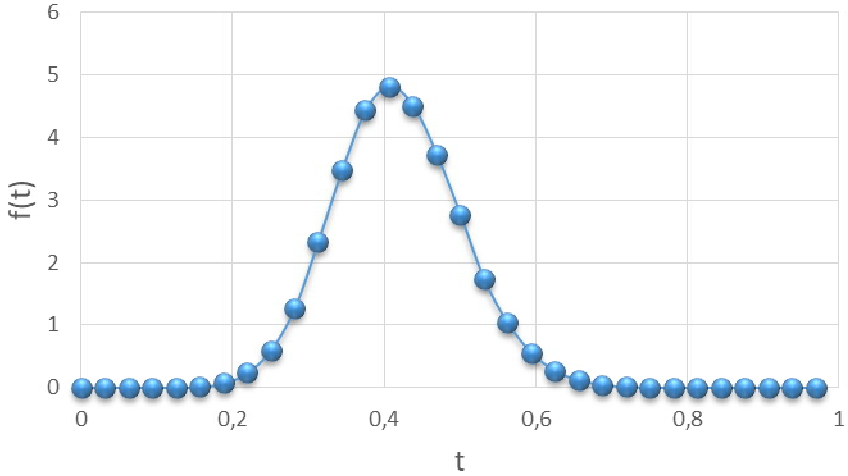


Рис. 2. Розподіл часів до відмов за результатами комп'ютерного моделювання для системи розміром N * N = 8 * 8 вузлів.

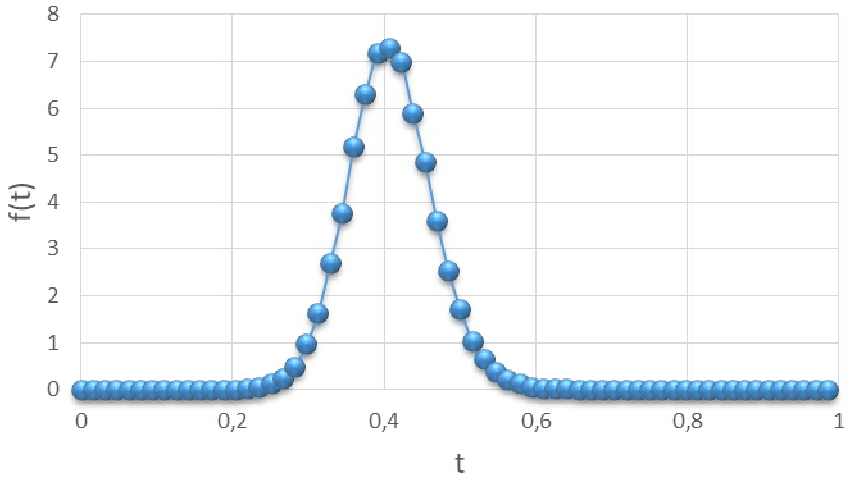


Рис. 3. Розподіл часів до відмов за результатами комп'ютерного моделювання для системи розміром N * N = 16 * 16 вузлів.

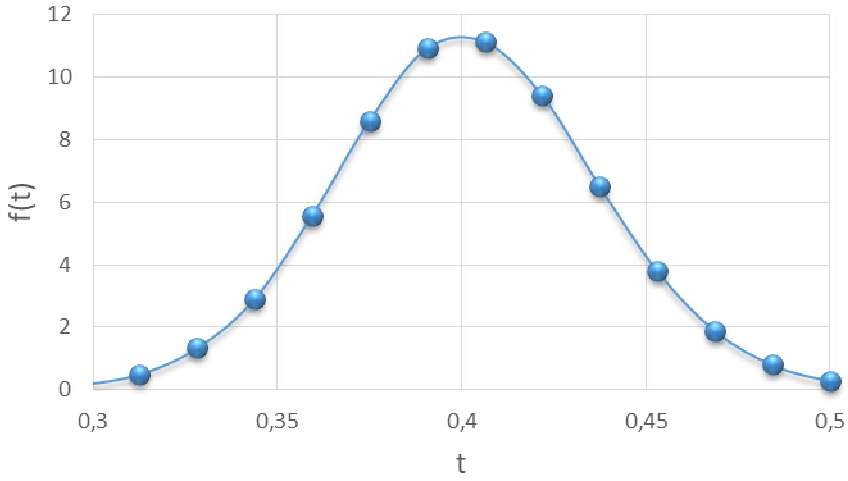


Рис. 4. Розподіл часів до відмов за результатами комп'ютерного моделювання для системи розміром N * N = 32 * 32 вузлів.

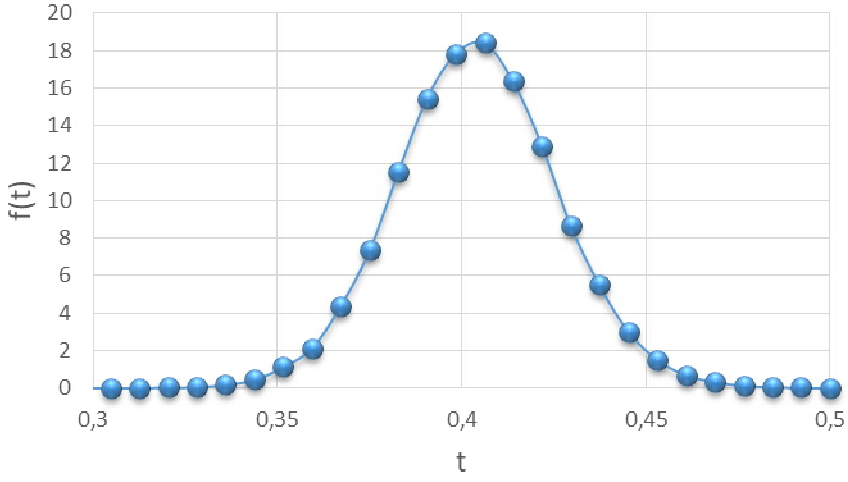


Рис. 5. Розподіл часів до відмов за результатами комп'ютерного моделювання для системи розміром N * N = 64 * 64 вузлів.

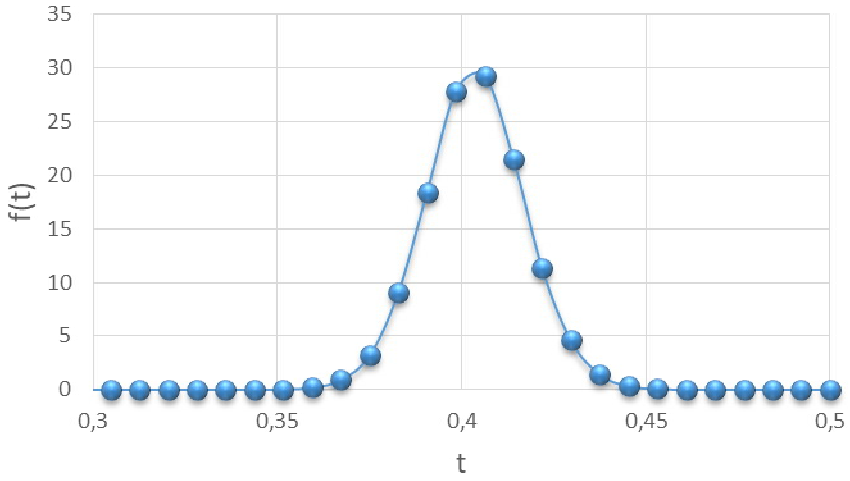


Рис. 6. Розподіл часів до відмов за результатами комп'ютерного моделювання для системи розміром N * N = 128 * 128 вузлів.

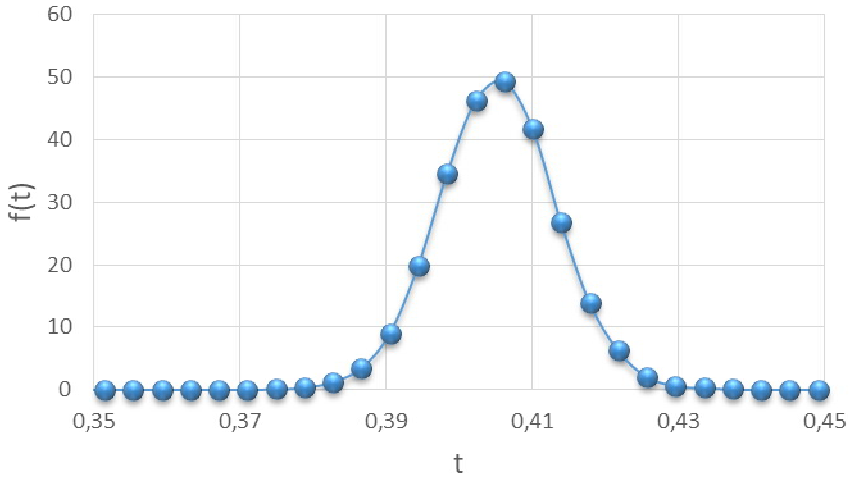


Рис. 7. Розподіл часів до відмов за результатами комп'ютерного моделювання для системи розміром N * N = 256 * 256 вузлів.

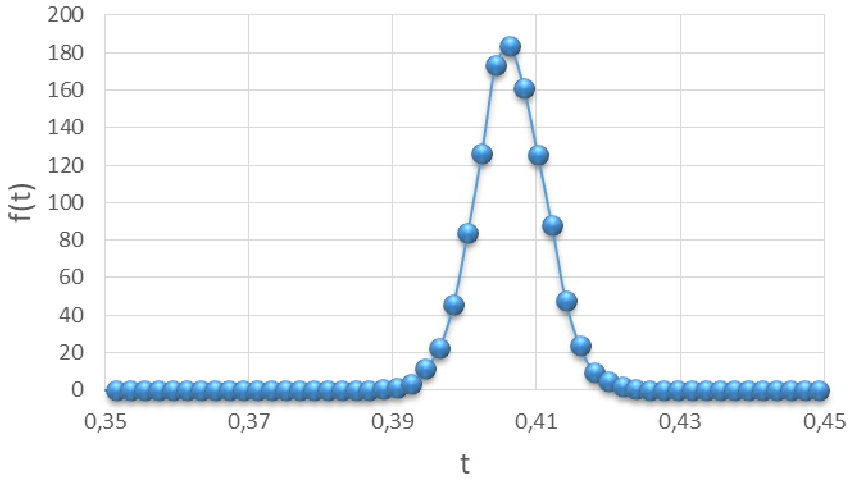


Рис. 8. Розподіл часів до відмов за результатами комп'ютерного моделювання для системи розміром N * N = 512 * 512 вузлів.

На рисунках 9-12 представлено логарифмічні залежності середнього значення, стандартного відхилення, коефіцієнту асиметрії і коефіцієнту ексцесу розподілів часів до відмов від розміру системи. Якщо ми можемо представити залежність параметра $ln\,(Y - Y0)$ від $ln\,(t)$ у вигляді лінійної $y = ax + b$, то це буде означати степеневу залежність відповідного параметру від розміру. Логарифмічні залежності для середнього значення, стандартного відхилення, коефіцієнту асиметрії можна вважати близькими до лінійних, що говорить про степеневі залежності відповідних параметрів розподілів від розміру системи.

Залежність коефіцієнта ексцесу (рис. 12) наблизити простою функціональною залежністю не вдалося.

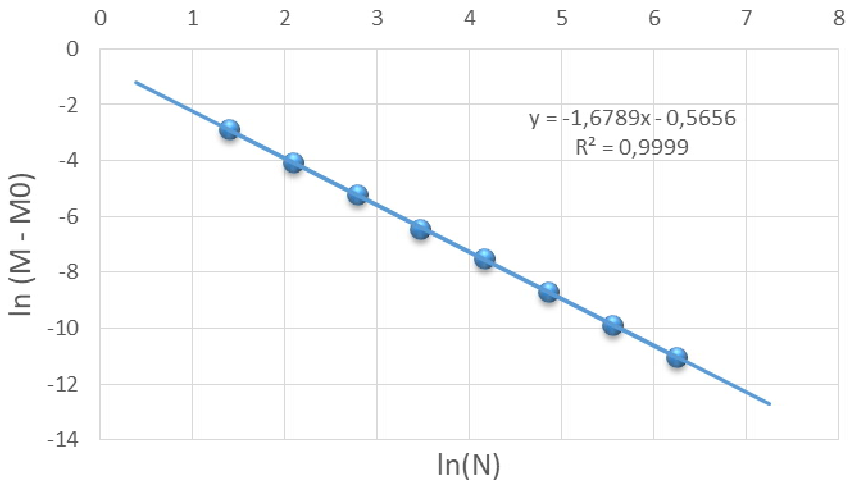


Рис. 9. Залежність логарифма різниці середнього значення розподілу часу до відмови та його асимптотичного значення 0,407225 від логарифма розміру системи.

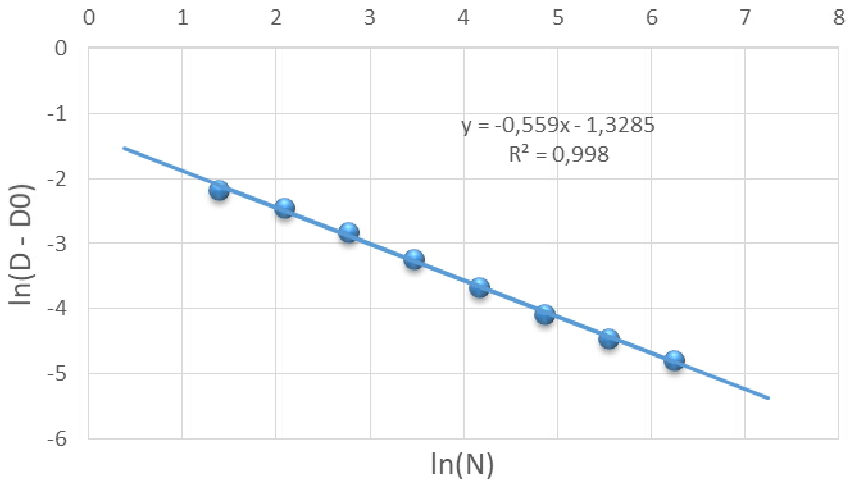


Рис. 10. Залежність логарифма різниці стандартного відхилення розподілу часу до відмови та його асимптотичного значення - 0,0035 від логарифма розміру системи.

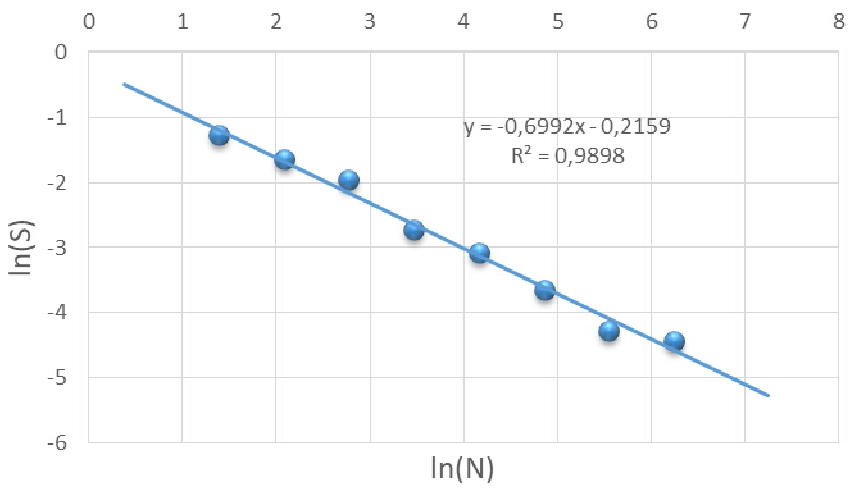


Рис. 11. Залежність логарифма коефіцієнта асиметрії розподілу часу до відмови від логарифма розміру системи.

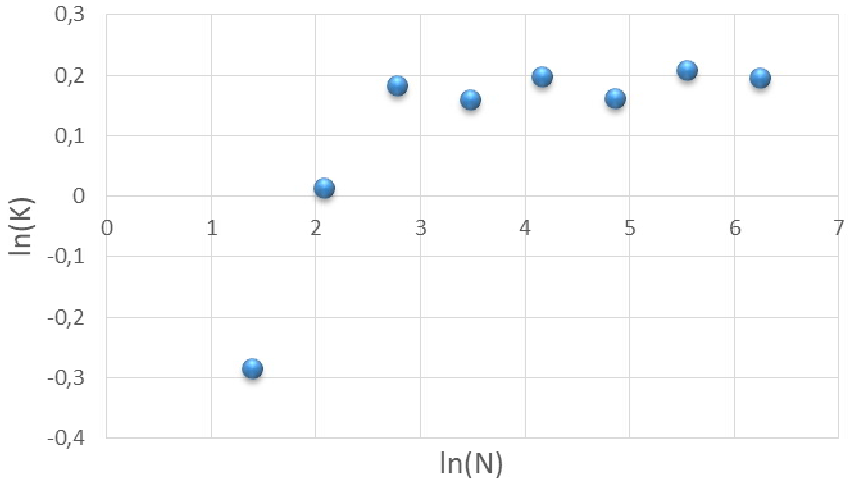


Рис. 12. Залежність логарифма коефіцієнта ексцесу від логарифма розміру системи.

Висновки

У роботі створено двовимірну комп'ютерну модель для симуляції процесу відмов у мікроелектронних системах. На основі розробленої моделі проведено серію комп'ютерних експериментів для аналізу характеристик розподілів часів до відмови у залежності від розміру модельної системи.

Отримано розмірний ефект процесу відмов – характеристики розподілів часів до відмов залежать від розміру системи:

- при зменшенні розміру системи середній час до відмови пристрою зростає.
- стандартне відхилення розподілу часів відмов зростає при зменшенні розміру системи.
- коефіцієнт асиметрії розподілу часів відмов зростає при зменшенні розміру системи.
- встановлено показникові залежності середнього часу до відмови, стандартного відхилення та коефіцієнта асиметрії розподілів часів до відмов від розміру системи.
- коефіцієнт ексцесу зменшується при зменшенні розміру системи.

Подяки

Статтю написано згідно з держбюджетною темою «Сингулярні розв`язки рівнянь математичної фізики в анізотропних і неоднорідних середовищах, моделювання процесів дифузії та абсорбції» (№ державної реєстрації 0119U100421).

Список використаної літератури:

1. Tu, K. N. Electronic thin-film reliability / K. N. Tu. – New York: Cambridge University Press, 2010. – 394 p.
2. Tian, T. A new physical model for life time predictionof Pb-free solder joints in electromigration tests / T. Tian, A. M. Gusak, O. Y. Liashenko, J. K. Han, D. Choi, K. N. Tu // Electronic Components and Technology Conference (ECTC), 2012 IEEE 62nd. – P. 741-746.
3. Liu, H.-C. Interfacial void ripening in Cu[sbnd]Cu joints / H.-C. Liu, A. M. Gusak, K. N. Tu, C. Chen // Materials Characterizationthis. – 2021, - V. 181. – P. 111459.
4. Tu, K. N. A unified model of mean-time-to-failure for electromigration, thermomigration, and stress-migration based on entropy production / K. N. Tu, A. M. Gusak //Journal of Applied Physics. – 2019. – V. 126. – № 7. – P. 075109.
5. Tu, K. N. Mean-Time-To-Failure Equations for Electromigration, Thermomigration, and Stress Migration / K. N. Tu, A. M. Gusak // IEEE Transactions on Components, Packaging and Manufacturing Technology. – 2020. – V. 10(9). – P. 1427–1431.
6. Бобров, О. А. Розмірний ефект розподілу часів до відмови та часів перетворення / О. А. Бобров, М. О. Пасічний, О. Ю. Ляшенко, А. М. Гусак // Вісник Черкаського університету. Серія Фізико-математичні науки. – 2017. – №1. – С. 53-63.

References:

1. Tu, K. N. (2010) *Electronic thin-film reliability*. New York: Cambridge University Press.
2. Tian, T., Gusak, A. M., Liashenko, O. Y., Han, J. K., Choi, D., Tu, K. N. (2012) A new physical model for life time predictionof Pb-free solder joints in electromigration tests. Electronic Components and Technology Conference (ECTC), 2012 IEEE 62nd, 741-746.
3. Liu, H.-C., Gusak A. M., Tu, K. N. Chen, C. (2021) Interfacial void ripening in Cu[sbnd]Cu joints. *Materials Characterizationthis*, 181, 111459.
4. Tu, K. N., Gusak A. M. (2019) A unified model of mean-time-to-failure for electromigration, thermomigration, and stress-migration based on entropy production. *Journal of Applied Physics*, 126(7), 075109.
5. Tu, K. N. Gusak A. M. (2020) Mean-Time-To-Failure Equations for Electromigration, Thermomigration, and Stress Migration. *IEEE Transactions on Components, Packaging and Manufacturing Technology*, 10(9), 1427–1431.
6. Bobrov, O. A., Pasichnyy, M. O., Gusak, A. M. (2017) Rozmirnyj efekt rozpodilu chasiv do vidmovy ta chasiv peretvorennya [Dimensional effect of distribution of times before failure and times of transformation] Visnyk Cherkaskoho universytetu. Seriia “Fizyko-matematychni nauky” – Bulletin of Cherkasy University. Series of Physical and Mathematical Sciences, 1, 53-63.

PASICHNYY Mykola,
Candidate of Physical and Mathematical Sciences, Associate Professor, Head of the Chair of Physics,
The Bohdan Khmelnytsky National University of Cherkasy, Cherkasy, Ukraine,
TATARCHUK Yevhenii,
Candidate of Physical and Mathematical Sciences, Associate Professor, Department of Physics,
The Bohdan Khmelnytsky National University of Cherkasy, Cherkasy, Ukraine,

SIMULATION OF THE SIZE EFFECT IN THE PROCESS OF FAILURES OF MICROELECTRONIC DEVICES ON THE BASIS OF THE APPROACH OF PERCOLATION CLUSTER FORMATION

Summary. Introduction. *A computer model was created and the failure process based on the percolation cluster approach was studied. The indicator dependences of the average time to failure, standard deviation and the asymmetry coefficient of time distributions to failures from the system size are established.*

The component base of the modern microelectronic industry, the basis of which is the use of integrated circuits based on semiconductor crystals (chips), is rapidly improving and becoming more complex. The development of modern electronic devices is aimed at miniaturization and reduction of the linear dimensions of the element base of microelectronic components. During operation, there is a

degradation of components and at some point in time it leads to device failure. Common causes of failures in the operation of microelectronic devices are the failure of elements in the electrical connection of conductors due to the formation of cracks or pores due to thermal migration, electromigration, thermal stress and other phenomena. Reducing the size of electronic components leads to an increase in current density through individual elements and intensification of the processes of degradation of electrical contact. At the same time, the requirements for the reliability of electronic devices are also growing. Given the concept of miniaturization of the element base of modern electronic devices, understanding the general behavior of the characteristics of the failure process of electronic systems while reducing their size is relevant and important from an applied point of view.

Purpose. *The aim of the article is to develop a computer model and statistical study of the behavior of the main characteristics of the failure process depending on the size of the system.*

To study the size effect, a series of computer experiments of the model system failure process was performed and a statistical analysis of time to failures distributions was performed to establish the characteristic dependences of the main characteristics of distributions for model systems of different sizes.

Results. *The computer model is based on the analysis of the possibility of electric current passing through a two-dimensional system the size of N * N sites in the form of a square lattice. The main provisions of the model:*

- *Each site has 4 nearest neighbors.*
- *In the initial state, all sites of the system are filled with elements.*
- *Current can only flow if there are elements in two adjacent sites.*
- *At each iteration, remove the element from a randomly selected system site.*
- *The time to failure will be considered the ratio of the number of iterations to the total number of system sites, when the leading cluster between the left and right boundaries of the model system disappears.*

*Based on the described model, a series of computer experiments was performed and the results formed the distribution of time to failure for a model system of a certain size N * N sites.*

The considered model is equivalent to the problem of percolation of sites. The formation of a percolation cluster of sites of deleted elements of the model system will correspond to the time to failure.

Conclusion*. A two-dimensional computer model was created to simulate the process of failures in microelectronic systems. Based on the developed model, a series of computer experiments were performed to analyze the characteristics of the distributions of time to failure depending on the size of the model system.*

The size effect of the failure process is obtained - the characteristics of time distributions to failures depend on the size of the system:

- *As the system size decreases, the average time to device failure increases.*
- *The standard deviation of the time to failure distribution increases as the system size decreases.*
- *The time to failure distribution asymmetry coefficient (Skewness) increases with decreasing system size.*
- *The dependences of the average time to failure, standard deviation and the asymmetry coefficient of time distributions to failures from the system size are established.*
- *The Kurtosis of time to failure distribution decreases as the size of the system decreases.*

Keywords: *Size effect, Time to failure, Percolation cluster.*

Одержано редакцією 12.10.2021 р.
Прийнято до публікації 24.11.2021 р.

СЕКЦІЯ «ІНФОРМАТИКА»

УДК 004.85:519.6

DOI 10.31651/2076-5886-2021-1-58-68

PACS 02.70.Wz, 07.05.Kf, 07.05.Mh, 07.05.Tp

КОНОНЕНКО Вероніка Вікторівна,
студентка спеціальності «Прикладна математика» Черкаського національного університету імені Богдана Хмельницького

КРАСНОШЛИК Наталія Олександрівна,
кандидат технічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: wlik007@ukr.net
ORCID 0000-0003-4661-6997

ВИКОРИСТАННЯ МЕТОДІВ БУСТІНГУ ДЛЯ ЗАДАЧ МАШИННОГО НАВЧАННЯ

У роботі проведено порівняння методів бустінгу при розв'язанні задач машинного навчання. Розглянуто бустінг над деревами рішень, який являє собою ансамблевий алгоритм і залишається одним з найбільш ефективних і популярних методів машинного навчання. Описано ідею методів градієнтного і адаптивного бустінгу та бібліотек XGBoost і CatBoost. Використано методи градієнтного бустінгу, AdaBoost, XGBoost і CatBoost у середовищі Jupyter Notebook для розв'язання задач класифікації та регресії. Проведено порівняння якості отримуваних прогнозів та часу навчання алгоритмів при розв'язанні задачі медичної діагностики, задачі кредитного скорінгу, задачі прогнозування кількості оренди велосипедів. Встановлено, що найменший час навчання і найкращі результати для задач бінарної класифікації демонструє модель XGBoost. Для задачі регресії кращі результати продемонструвала модель CatBoost, але при цьому час її навчання завжди на порядок більший у порівняні з іншими моделями.

Ключові слова: *машинне навчання, бустінг, градієнтний бустінг, XGBoost, CatBoost.*

Вступ

Машинне навчання є підрозділом досить великої області науки, що вивчає штучний інтелект. Алгоритми, що відносяться до даного напрямку, використовуються при розв'язанні задач, для яких часто складно або неможливо знайти явний алгоритм розв'язання, наприклад: медична діагностика, створення алгоритмів фільтрації реклами і спаму, прогноз погоди, прогнозування економічних і соціальних процесів, детектування об'єктів на фото або відео, розпізнавання тексту, мови та ін..

Серед сучасних методів машинного навчання одними з досить ефективних є методи, які використовують ансамблі алгоритмів. У процесі роботи алгоритми цього класу генерують відразу кілька прогнозів за допомогою кожного окремого алгоритму, що входить в ансамбль. Підсумковий розв'язок задачі одержують шляхом інтеграції набору отриманих розв'язків певним методом, наприклад, методом голосування.

До часто використовуваних ансамблів алгоритмів відносять бустінг, що

обумовлено його досить активним розвитком і високою якістю одержуваних результатів. Бустінг (англ. boosting – покращення) – це процедура послідовного побудови ансамблю алгоритмів машинного навчання, коли кожен наступний алгоритм прагне компенсувати недоліки композиції всіх попередніх алгоритмів.

Мета роботи полягає у застосуванні і порівнянні різних методів бустінгу при розв'язанні задач машинного навчання.

Виклад основного матеріалу

1. Методи бустінгу

Ансамбль алгоритмів – це метод, який використовує кілька алгоритмів з метою отримання кращої ефективності прогнозування, ніж можна було б отримати від кожного алгоритму машинного навчання окремо. В основі таких методів лежить ідея навчання декількох (базових) класифікаторів на одній і тій самій навчальній вибірці та комбінації їх прогнозів для нових тестових даних. Математичним обґрунтування цієї ідеї є теорема Кондорсе про журі присяжних (the Condorcet's jury theorem), що датується XVIII століттям [1-2].

Таким чином, маючи кілька слабких класифікаторів, можна об'єднати їх прогнози і досягти більш високої точності класифікації об'єктів з тестової вибірки. Серед найбільш відомих ансамблевих методів класифікації виділяють: беггінг (bagging), бустінг (boosting), стекінг (stacking).

Бустінг – метод побудови ансамблю алгоритмів, при якому базові алгоритми навчаються послідовно і кожен наступний алгоритм ансамблю застосовується до результатів на виході попереднього.

Таким чином, метод бустінга реалізує послідовну композицію алгоритмів навчання, в якій кожен алгоритм повинен компенсувати помилки, допущені композицією попередніх алгоритмів (див. рис. 1).

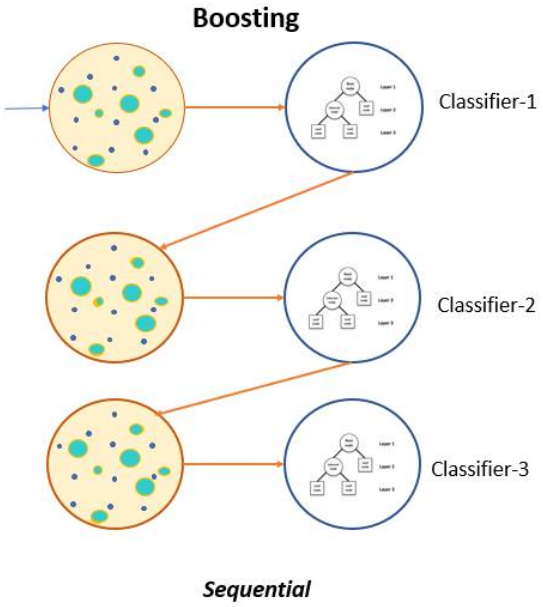


Рис. 1 Ідея методу бустінгу [4]

У основі ідеї бустінга лежить питання: чи може набір «слабких» навчальних алгоритмів дати у підсумку «сильний» алгоритм? Різні алгоритми навчання помиляються на різних прикладах. Тоді перший алгоритм в композиції буде навчатися

на всьому наборі, другий – на тих прикладах, на яких перший припустився помилки, третій – на тих прикладах, де помилилися перший і другий алгоритми, і т. д. При цьому кожному об'єкту вибірки, залежно від величини допущеної на ньому помилки, присвоюється вага, яка впливає на ймовірність вибору об'єкту для навчання наступного алгоритму композиції [3].

На сьогоднішній день бустінг залишається одним з найбільш популярних і широковживаних методів машинного навчання. Основні причини це – простота, універсальність, гнучкість (можливість побудови різних модифікацій), і висока узагальнююча здатність. Головну роль у популяризації бустінгу зіграли різні змагання з машинного навчання, особливо які проводяться на платформі kaggle.com.

Бустінг над деревами рішень вважається одним з найбільш ефективних методів з точки зору якості класифікації. При його застосуванні можна спостерігати сутеве зменшення частоти помилок на тестової вибірці по мірі нарощування композиції. На прикладі бустінга стало зрозуміло, що досить складні композиції алгоритмів демонструють гарну якість, якщо їх правильно налаштовувати. Згодом феномен бустінга отримав теоретичне обґрунтування. Виявилося, що зважене голосування не збільшує ефективну складність алгоритму, а лише згладжує відповіді базових алгоритмів. Кількісні оцінки узагальнюючої здатності бустінга формулюються в термінах відступу. Ефективність бустінга пояснюється тим, що в міру додавання базових алгоритмів збільшуються відступи навчальних об'єктів. Причому бустінг продовжує розмежовувати класи навіть після досягнення безпомилкової класифікації навчальної вибірки [1].

1.1 Адаптивний алгоритм бустінгу AdaBoost

Алгоритм AdaBoost (від Adaptive Boosting) – алгоритм машинного навчання, запропонований Йоав Фройнд (Yoav Freund) і Робертом Шапіро (Robert Schapire). Це мета-алгоритмом, який в процесі навчання будує композицію з базових алгоритмів для покращення їх ефективності. AdaBoost є алгоритмом адаптивного бустінгу в тому розумінні, що кожен наступний класифікатор будується для об'єктів, які погано класифікуються попередніми класифікаторами.

AdaBoost викликає слабкий класифікатор у циклі. Після відповідного виклику оновлюються значення вагів, які визначають важливість тих чи інших об'єктів з навчальної вибірки для класифікації. На кожній ітерації ваги кожного невірно класифікованого об'єкта зростають, таким чином новий класифікатор «фокусує свою увагу» на цих об'єктах [5].

Опишемо базовий алгоритм для задачі побудови бінарного класифікатора. Розглядаємо задачу класифікації на два класи $Y = \{-1, +1\}$. Припустимо, що базові алгоритми $b_1, ..., b_T$ також повертають лише дві відповіді -1 і +1. Вектор вагів об'єктів $\mathrm{W}^l = (\mathrm{w}_1, ..., \mathrm{w}_2)$, навчальна вибірка X^l. Нехай

$$Q(b, W^l) = \sum_{i=1}^{l} w_i \left[y_i \cdot b(x_i) < 0 \right]$$

– стандартний функціонал якості алгоритму класифікації b. Апроксимуємо порогову функцію втрат $[z < 0]$ за допомогою експоненти $E(z) = exp(-z)$.

Кроки алгоритму AdaBoost будуть наступні [5]:

1. Ініціалізація ваги об'єктів: $w_i = 1/i, i = 1, ..., l$;

2. Для всіх $t = 1, ..., T$, поки не виконаний критерій зупинки.

2.1 Знаходимо класифікатор $b_t : X \to \{-1,+1\}$ який мінімізує зважену похибку класифікації

$$b_t = \arg\min_b Q(b, W^l)$$

2.2 Перераховуємо коефіцієнти зваженого голосування для алгоритму класифікації b_t:

$$\alpha_t = \frac{1}{2}\ln\frac{1-Q(b,W^l)}{Q(b,W^l)}$$

2.3 Перерахунок вагів об'єктів: $w_i = w_i \cdot \exp(-\alpha_t \cdot y_i \cdot b_t(x_i)), i = 1,...,l$.

2.4 Нормування вагів об'єктів: $w_0 = \sum_{j=1}^{l} w_j$, $w_i = w_i / w_0$, $i = 1,...,l$.

3. Повертаємо: $a(x) = sign\left(\sum_{i=1}^{T} \alpha_i \cdot b_i(x)\right)$.

Така техніка послідовного навчання нагадує градієнтний спуск, лише замість зміни параметрів одного алгоритму для мінімізації функції втрат, AdaBoost додає моделі в ансамбль, поступово покращуючи його.

1.2 Градієнтний бустінг

Іншим дуже популярним бустінговим алгоритмом, принцип роботи якого дуже схожий на розглянутий AdaBoost, є градієнтний бустінг. Даний алгоритм працює послідовно додаючи до минулих моделей нові так, щоб виправлялися помилки, допущені попередніми предикторами. Метод градієнтного бустінгу відрізняється від адаптивного тим, що, на відміну від AdaBoost, який змінює ваги на кожній ітерації, градієнтний намагається навчати нові моделі за залишковою похибкою попередніх моделей (рухаючись до мінімуму функції втрат).

Градієнтний бустінг починає роботу з побудови початкової моделі і коригує її, крок за кроком створюючи послідовність моделей. Кожна модель у послідовності створюється на основі залишків моделі, яка отримується на попередньому кроці. Залишки моделі по суті використовуються у якості цільової змінної. Тобто розв'язується задача [6]:

$$F = \sum_{i=1}^{m} L\left(y_i, a(x_i) + b_i\right) \to \min \qquad (1)$$

де $a(x_i)$ – початково побудований алгоритм, b_i – наступний алгоритм, який будується і здійснює коригування відповідей $a(x_i)$ до вірних. Таким чином, поліпшується функціонал $\varepsilon_i = y_i - a(x_i)$.

Вираз (1) слід розглядати як мінімізацію функції $F\left(b_1,...,b_m\right)$, отже, необхідно вибрати ефективний метод її мінімізації. У даному випадку функція від багатьох змінних максимально зменшується в напрямку свого антиградієнта:

$$-\left(L'\left(y_1, a(x_1)\right),...,L'\left(y_m, a(x_m)\right)\right) .$$

Отже, відповіді алгоритму b_i можуть бути визначені наступним чином:

$$b_i = -L'\left(y_i, a(x_i)\right), i = 1,...,m .$$

Налаштування алгоритму відбувається на навчальній вибірці:

$$\left(x_i, -L'\left(y_i, a(x_i)\right)\right)_{i=1}^{m} .$$

Звідси визначається і назва методу – градієнтний бустінг. Далі, після побудови відповідей алгоритму b_i, будується третій алгоритм, який коригує зазначену суму. А далі процес продовжується. Це приводить до одного з основних параметрів градієнтного бустінгу – кількості ітерацій бустінгу N. Параметр впливає на точність одержуваних результатів.

Наступним важливим параметром є η або швидкість навчання (learning rate). Суть його полягає у тому, що зміщення аргументу в F відбувається тільки на частину вектору для того щоб зберегти баланс між точністю і швидкістю збіжності алгоритму:

$$a_{t+1}(x) = a_t(x) + \eta \cdot b_t \ ,$$

де даний параметр визначається у межах $\eta \in (0,1]$.

Наразі вважається, що градієнтний бустінг над деревами рішень є один з найбільш універсальних і сильних методів машинного навчання, відомих на сьогоднішній день.

1.3 XGBoost і CatBoost

Екстремальний градієнтний бустінг (XGBoost – eXtreme Gradient Boosting) – це удосконалена реалізація градієнтного бустінгу. Цей алгоритм має високу прогнозну здатність і в десять разів швидше будь-яких інших методів градієнтного бустінгу. Крім того, включає в себе різні регуляризації, що зменшують перенавчання і покращують загальну продуктивність. Тобто, це програмна бібліотека з відкритим кодом, яка використовує системну оптимізацію та удосконалення алгоритму (див. рис. 2).

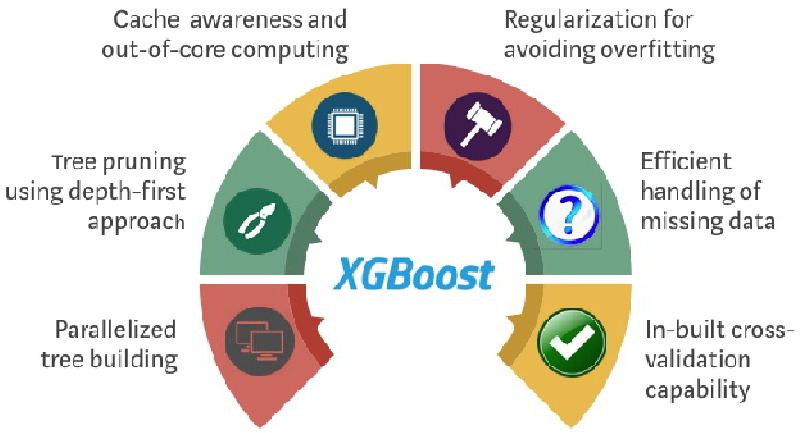


Рис. 2 Як XGBoost оптимізує стандартний градієнтний бустінг [7]

У XGBoost використовується системна оптимізація та покращення алгоритму [7]. Системна оптимізація передбачає наступне:

- *Паралелізація*. У XGBoost побудова дерев ґрунтується на паралелізації. Для покращення часу роботи порядок циклів змінюється: ініціалізація проходить при зчитуванні даних, потім виконується сортування, яке використовує паралельні потоки.
- *Відсікання гілок дерева*. У градієнтному бустінгу критерій зупинки для розбиття дерева залежить від критерію від'ємної втрати у точці розбиття. XGBoost використовує параметр максимальної глибини `max_depth` замість цього критерію і починає зворотне відсікання.

– *Апаратна оптимізація*. Алгоритм був розроблений таким чином, щоб він оптимально використовував апаратні ресурси. Це досягається шляхом створення внутрішніх буферів у кожному потоці для зберігання статистики градієнта.

А покращення алгоритму полягають у наступному:

– *Регуляризація*. Алгоритм штрафує складні моделі, використовуючи як регуляризацію Lasso (L1), так і Ridge-регуляризацію (L2), для того, щоб уникнути перенавчання.
– *Робота з розрідженими даними*. Алгоритм спрощує роботу з розрідженими даними, у процесі навчання заповнюючи пропущені значення в залежності від значення втрат.
– *Метод зважених квантилів.* XGBoost використовує його для того, щоб найбільш ефективно знаходити оптимальні точки поділу у разі роботи зі зваженим набором даних.
– *Крос-валідація*. Алгоритм використовує свій власний метод крос-валідації на кожній ітерації. Тобто, не потрібно окремо програмувати цей пошук і визначати кількість ітерацій бустінгу для кожного запуску.

Ще одним прикладом методу бустінгу є CatBoost.

CatBoost – відкрита програмна бібліотека, розроблена компанією Яндекс, яка реалізує унікальний запатентований алгоритм побудови моделей машинного навчання, що використовує одну з оригінальних схем градієнтного бустінгу. Основне API для роботи з бібліотекою реалізовано для мови Python, також існує реалізація для мови програмування R.

У липні 2017 року компанія Яндекс виклала бібліотеку з алгоритмом CatBoost у вільний доступ з відкритою ліцензією Apache 2.0.

Переваги використання CatBoost наступні [8]:

– CatBoost дозволяє проводити навчання на декількох GPU.

– Бібліотека дозволяє отримати відмінні результати з параметрами за замовчуванням, що скорочує час, необхідний для налаштування гіперпараметрів.

– Забезпечує підвищену точність за рахунок зменшення перенавчання. Навчені моделі CatBoost можна експортувати у Core ML для виведення на пристроях (iOS).

– Реалізовано можливість обробляти пропущені значення.

– Може використовуватися для задач регресії та класифікації.

2. Приклади задач машинного навчання

У якості прикладів задачі бінарної класифікації розглянемо задачу прогнозування у пацієнта хвороби Паркінсона і задачу кредитного скорінгу.

Для прогнозування у пацієнта хвороби Паркінсона скористаємось набором даних «Parkinsons Data Set» [9]. Цей датасет складається з ряду біомедичних вимірювань голосу від 31 людини, 23 з яких мають хворобу Паркінсона. Кожен стовпець у таблиці є певним виміром голосу, а кожен рядок відповідає одному із 195 записів голосу від цих осіб (існує близько шести записів на одного пацієнта) і 23 характеристики голосу.

У якості ще одного прикладу розглянемо задачу визначення кредитної платоспроможності (кредитного скорінгу). Нехай є дані про клієнтів, які звернулися за кредитом. Тут об’єктами є клієнти, а ознаками можуть бути їхня стать, рівень доходу, освіта, інформація про те, є чи немає у них позитивної кредитна історія і т. д. Цільовою ознакою (відповіддю) виступає інформація про те, повернув клієнт кредит в банк чи ні. За цими даними потрібно навчитися передбачати, чи поверне кредит новий клієнт, який звернувся у банк. Таким чином, мова йде про задачу класифікації: потрібно визначити, до якого класу належить клієнт: позитивного (кредит буде повернений) або негативного (кредит не буде повернутий).

Скористаємось набором даних «Credit Approval Data Set» [10]. Цей датасет містить інформацію про 690 клієнтів, з яких 307 виплатили кредит, а 383 – ні. Опис кожного клієнта містить 14 ознак (6 числових і 8 категорійних). Оскільки цей файл стосується кредитних карт, то для захисту конфіденційності даних всі імена та значення ознак замінені деякими символами.

У якості прикладу задачі регресії розглянемо задачу прогнозування кількості велосипедів, які взято на прокат за день.

Будемо працювати з датасетом «Bike Sharing Dataset Data Set» [11], у якому по днях записана календарна інформація та погодні умови, що характеризують автоматизовані пункти прокату велосипедів, а також число орендованих у цей день велосипедів, яке потрібно спрогнозувати.

3. Порівняння методів бустінгу

Для проведення обчислювальних експериментів використано середовище Jupyter Notebook.

Виконаємо попередню обробку даних з датасету «Parkinsons Data Set», фрагмент якого представленого на рис. 3. Для цього необхідно виконати нормування ознак так, щоб кінцеві значення знаходилися в інтервалі від -1 до 1 і розділити вибірку на тренувальну, валідаційну і тестову.

```
In [2]: df = pd.read_csv('parkinsons.data')
        df.head()
```

Out[2]:

	name	MDVP:Fo(Hz)	MDVP:Fhi(Hz)	MDVP:Flo(Hz)	MDVP:Jitter(%)	MDVP:Jitter(Abs)	MDVP:RAP	MDVP:PPQ	Jitter:DDP	MDVP:Shimmer
0	phon_R01_S01_1	119.992	157.302	74.997	0.00784	0.00007	0.00370	0.00554	0.01109	0.04374
1	phon_R01_S01_2	122.400	148.650	113.819	0.00968	0.00008	0.00465	0.00696	0.01394	0.06134
2	phon_R01_S01_3	116.682	131.111	111.555	0.01050	0.00009	0.00544	0.00781	0.01633	0.05233
3	phon_R01_S01_4	116.676	137.871	111.366	0.00997	0.00009	0.00502	0.00698	0.01505	0.05492
4	phon_R01_S01_5	116.014	141.781	110.655	0.01284	0.00011	0.00655	0.00908	0.01966	0.06425

5 rows × 24 columns

Рис. 3 Набір даних «Parkinsons Data Set»

Для оцінки результатів будемо використовувати функцію accuracy_score( ) з бібліотеки sklearn, тобто обрахуємо долю правильних відповідей на тестовій вибірці. Результати застосування різних алгоритмів бустінгу для прогнозування наявності хвороби Паркінсона представлено у табл.1.

Таблиця 1. Бустінг для задачі медичної діагностики

Алгоритм бустінгу	Доля правильних відповідей (accuracy score)	Час навчання
Адаптивний бустінг AdaBoost	0.8813	119 ms
Градієнтний бустінг	0.9573	116 ms
XGBoost	0.8983	37.8 ms
CatBoost	0.9152	3.86 s

Виконаємо попередню обробку даних з датасету «Credit Approval Data Set», фрагмент якого представленого на рис. 4. Для цього видалимо пропущені значення та закодуємо категорійні ознаки.

Застосуємо розглянуті методи бустінгу до задачі кредитного скорінгу, отримані результати представлено у табл. 2.

Таблиця 2. Бустінг для задачі кредитного скорінгу

Алгоритм бустінгу	Доля правильних відповідей (accuracy score)	Час навчання
Адаптивний бустінг AdaBoost	0.8594	76 ms
Градієнтний бустінг	0.8978	69 ms
XGBoost	0.8933	34 ms
CatBoost	0.8910	2.17 s

```
df = pd.read_csv('crx.data', header=None, names=['A1', 'A2', 'A3', 'A4', 'A5', 'A6', 'A7', 'A8',
                 'A9', 'A10', 'A11', 'A12', 'A13', 'A14', 'A15', 'A16'])
df.head()
```

	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13	A14	A15	A16
0	b	30.83	0.000	u	g	w	v	1.25	t	t	1	f	g	00202	0	+
1	a	58.67	4.460	u	g	q	h	3.04	t	t	6	f	g	00043	560	+
2	a	24.50	0.500	u	g	q	h	1.50	t	f	0	f	g	00280	824	+
3	b	27.83	1.540	u	g	w	v	3.75	t	t	5	t	g	00100	3	+
4	b	20.17	5.625	u	g	w	v	1.71	t	f	0	f	s	00120	0	+

Рис. 4 Набір даних «Credit Approval Data Set»

Виконаємо попередню обробку даних з набору даних «Bike Sharing Dataset Data Set» », фрагмент якого представленого на рис. 5. Для цього видалимо стовпчики atemp та windspeed(mph), і виконаємо нормування ознак.

```
df = pd.read_csv('bikes_rent.csv',index_col=False, sep=',')
df.head()
```

	season	yr	mnth	holiday	weekday	workingday	weathersit	temp	atemp	hum	windspeed(mph)	windspeed(ms)	cnt
0	1	0	1	0	6	0	2	14.110847	18.18125	80.5833	10.749882	4.805490	985
1	1	0	1	0	0	0	2	14.902598	17.68695	69.6087	16.652113	7.443949	801
2	1	0	1	0	1	1	1	8.050924	9.47025	43.7273	16.636703	7.437060	1349
3	1	0	1	0	2	1	1	8.200000	10.60610	59.0435	10.739832	4.800998	1562
4	1	0	1	0	3	1	1	9.305237	11.46350	43.6957	12.522300	5.597810	1600

Рис. 5 Набір даних «Bike Sharing Dataset Data Set»

Для оцінки результатів будемо використовувати функцію score( ) з бібліотеки sklearn, тобто коефіцієнт детермінації R^2.

$$R^2 = \frac{S_{reg}}{S_{tot}},$$

де $S_{reg} = \sum_{i=1}^{n} (\hat{y}_i - \overline{y}_i)^2$, $S_{tot} = \sum_{i=1}^{n} (y_i - \overline{y}_i)^2$, $\hat{y}_i$ – прогнозоване значення параметра, $\overline{y}_i$ – середнє арифметичне значення параметру.

Результати застосування різних алгоритмів бустінгу для прогнозування кількості орендованих велосипедів представлено у табл. 3.

Таблиця 3. Бустінг для прогнозування кількості оренди

Алгоритм бустінгу	Коефіцієнт детермінації R^2	Час навчання
Адаптивний бустінг AdaBoost	0.7294	116 ms
Градієнтний бустінг	0.8470	104 ms
XGBoost	0.8483	57 ms
CatBoost	0.8698	2.6 s

Висновки

У роботі описано ідею методів градієнтного і адаптивного бустінгу та бібліотек XGBoost і CatBoost. Було проведено порівняння якості отримуваних прогнозів та часу навчання алгоритмів при розв'язанні задачі медичної діагностики, задачі кредитного скорінгу, задачі прогнозування кількості оренди велосипедів.

Результати обчислювальних експериментів показали, що загалом моделі бустінгу демонструють досить хороші результати як для задач класифікації, так і для задач регресії. Для задач бінарної класифікації кращі результати демонструють моделі, що використовують градієнтний бусттінг. Найменший час навчання і найкращі результати демонструє модель XGBoost, завдяки своїй апаратній і програмній оптимізації при реалізації даної бібліотеки. Модель CatBoost продемонструвала кращі результати для задачі регресії, але при цьому час її навчання завжди на порядок більший у порівняні з іншими моделями.

Список використаної літератури:

1. Кашницкий Ю. С. Ансамблевый метод машинного обучения, основанный на рекомендации классификаторов / Ю. С. Кашницкий, Д. И. Игнатов // Интеллектуальные системы. Теория и приложения, 2015. – Т. 19. №4, С. 37–55.
2. Кривохата А. Г. Застосування ансамблевого навчання в задачах класифікації акустичних даних / Кривохата А. Г., Кудін О. В., Давидовський М. В., Лісняк А. О. // Вісник Запорізького національного університету. Фізико-математичні науки, 2018. – №1. – С. 48-60.
3. Бустинг (Boosting) – Loginom Wiki [Електронний ресурс]. – Режим доступу: https://wiki.loginom.ru/articles/boosting.html.
4. Ensemble Methods in Machine Learning: Bagging Versus Boosting [Електронний ресурс]. – Режим доступу: https://www.pluralsight.com/guides/ensemble-methods:-bagging-versus-boosting.
5. Алгоритм AdaBoost [Електронний ресурс]. – Режим доступу: http://www.machinelearning.ru/wiki/index.php?title=AdaBoost.
6. Пивкин, К.С. Моделирование покупательского спроса на предприятиях розничной торговли на основе методов машинного обучения. [Текст]: дис. ... канд. экон. наук: 08.00.13: Ижевск, 2018. – 220 с.
7. Алгоритм XGBoost: пусть он царствует долго [Електронний ресурс]. – Режим доступу: https://cutt.ly/9nATUGb.
8. Быстрый градиентный бустинг с CatBoost / Блог компании OTUS / Хабр [Електронний ресурс]. – Режим доступу: https://habr.com/ru/company/otus/blog/527554.
9. UCI Machine Learning Repository: Parkinsons Data Set [Електронний ресурс]. – Режим доступу: https://archive.ics.uci.edu/ml/datasets/parkinsons.
10. UCI Machine Learning Repository: Credit Approval Data Set [Електронний ресурс]. – Режим доступу: https://archive.ics.uci.edu/ml/datasets/Credit+Approval.
11. UCI Machine Learning Repository: Bike Sharing Dataset Data Set [Електронний ресурс]. – Режим доступу: http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset.

References:

1. Kashnickij Yu. S. Ansamblevyj metod mashinnogo obucheniya, osnovannyj na rekomendacii klassifikatorov / Yu. S. Kashnickij, D. I. Ignatov // Intellektualnye sistemy. Teoriya i prilozheniya, 2015. – T. 19. №4, C. 37–55 [in Russian].
2. Krivohata A. G. Zastosuvannya ansamblevogo navchannya v zadachah klasifikaciyi akustichnih danih / Krivohata A. G., Kudin O. V., Davidovskij M. V., Lisnyak A. O. // Visnik Zaporizkogo nacionalnogo universitetu. Fiziko-matematichni nauki, 2018. – №1. – S. 48-60 [in Ukraine].

3. Busting (Boosting) – Loginom Wiki. Retrieved from: https://wiki.loginom.ru/articles/boosting.html [in Russian].
4. Ensemble Methods in Machine Learning: Bagging Versus Boosting (2020). – Retrieved from: https://www.pluralsight.com/guides/ensemble-methods:-bagging-versus-boosting.
5. Algoritm AdaBoost (2016). – Retrieved from: http://www.machinelearning.ru/wiki/index.php?title=AdaBoost [in Russian].
6. Pivkin, K.S. (2018) Modelirovanie pokupatelskogo sprosa na predpriyatiyah roznichnoj torgovli na osnove metodov mashinnogo obucheniya: dis. ... kand. ekon. nauk: 08.00.13: Izhevsk [in Russian].
7. Algoritm XGBoost: pust on carstvuet dolgo (2019). – Retrieved from: https://cutt.ly/9nATUGb [in Russian].
8. Bystryj gradientnyj busting s CatBoost / Blog kompanii OTUS / Habr (2020). – Retrieved from: https://habr.com/ru/company/otus/blog/527554 [in Russian].
9. UCI Machine Learning Repository: Parkinsons Data Set. – Retrieved from: https://archive.ics.uci.edu/ml/datasets/parkinsons.
10. UCI Machine Learning Repository: Credit Approval Data Set. – Retrieved from: https://archive.ics.uci.edu/ml/datasets/Credit+Approval.
11. UCI Machine Learning Repository: Bike Sharing Dataset Data. – Retrieved from: http://archive.ics.uci.edu/ml/datasets/Bike+Sharing+Dataset.

KONONENKO Veronika,
student, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
KRASNOSHLYK Natalia,
candidate of Technical Sciences, Associate Professor, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine

USING BOOSTING METHODS FOR MACHINE LEARNING PROBLEMS

Summary. Introduction. *Among the modern methods of machine learning, one of the most effective are the methods used by ensembles of algorithms. Frequently used ensembles of algorithms include boosting. Boosting is a method of building an ensemble of algorithms, in which the basic algorithms are learned sequentially and each subsequent algorithm of the ensemble is applied to the results of the previous one. Boosting on decision trees remains one of the most effective and popular methods of machine learning. The article describes the idea of gradient and adaptive boosting methods, XGBoost and CatBoost libraries, also considers their practical application for machine learning problems. Examples of the binary classification problems are the problem of predicting Parkinson's disease in a patient and the problem of credit scoring. As an example of the regression problem, consider the problem of predicting the number of bicycles rented per day.*

Purpose. *The purpose of this paper is to apply and compare different methods of boosting in solving machine learning problems.*

Results. *Gradient boosting, AdaBoost, XGBoost and CatBoost were used to solve classification and regression problems in the Jupyter Notebook environment. A comparison of the quality of forecasts and the time of learning algorithms in solving the machine learning problems. To assess the quality in the classification problem, the accuracy score on the test dataset was calculated, and in the regression problem - the coefficient of determination.*

The results of computational experiments have shown that, in general, boosting models show quite good results for both classification and regression problems. For binary classification problems, models using gradient boosting show the best results. The XGBoost model demonstrates the shortest training time and the best results, thanks to its hardware and software optimization in the implementation of this library. The CatBoost model has shown better results for the regression problem, but its training time is always an order of magnitude longer than other models.

Conclusion. *This article describes the idea of gradient and adaptive boosting methods, XGBoost and CatBoost libraries. A comparison of the quality of forecasts and the time of learning algorithms in solving the problem of medical diagnostics, the problem of credit scoring, the problem of predicting the number of bicycle rent was compared.*

The XGBoost model has been found to have the lowest learning time and best results for binary classification problems. For the regression problem, the CatBoost model showed the best results, but its learning time is always an order of magnitude longer than other models.

Keywords*: machine learning, boosting, gradient boosting, XGBoost, CatBoost.*

Одержано редакцією 15.04.2021 р.
Прийнято до публікації 10.06.2021 р.

УДК 519.688

DOI 10.31651/2076-5886-2021-1-68-84

PACS 02.60.-x, 02.60.Pn

ХАРЧЕНКО Віталій Вікторович,
студент спеціальності «Прикладна математика» Черкаського національного університету імені Богдана Хмельницького

ДІДКОВСЬКИЙ Руслан Михайлович,
доктор технічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: didkovskyirm@vu.cdu.edu.ua
ORCID 0000-0002-5166-7564

СЕРДЮК Олександр Анатолійович,
кандидат економічних наук, доцент, доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: serdyuk@ukr.net
ORCID 0000-0002-3919-4661

РОЗРОБКА СЕРВЕРНОЇ ЧАСТИНИ СИСТЕМИ ПЕРЕВІРКИ ПРОГРАМНОГО КОДУ НА НАЯВНІСТЬ ПЛАГІАТУ

У роботі описано аналіз передумов розробки та структуру розробленої серверної системи перевірки програмного коду, що є результатом виконання студентами практичних завдань з програмування, на наявність плагіату. Умови розробки системи складено на основі аналізу чотирьох наявних систем пошуку плагіату: MOSS, Codeqiury, Unicheck, CCFinderX. Для кожної з розглянутих систем визначено її переваги та недоліки, на основі чого визначено потрібні властивості розроблюваної системи. Визначено структуру серверної частини та потрібне програмне забезпечення й мову програмування для практичної реалізації. Визначено та наведено алгоритми для токенізації програмного коду та порівняння двох різних фрагментів коду з метою оцінки їх подібності. Реалізована у результаті система може використовуватись для надання API програмам пошуку плагіату у програмному коді.

***Ключові слова**: плагіат, плагіат у програмному коді, система перевірки на наявність плагіату, серверна система.*

Вступ

Запозичення коду серед студентів – це проблема, яка непокоїть багатьох викладачів університету, що викладають дисципліни, пов'язані з вивченням програмування. На жаль, викладачу важко особисто прослідкувати за унікальністю всіх робіт студентів. Окрім того, велику частку робочого часу доводиться витрачати на перевірку робіт, замість того, щоб цей час присвятити, наприклад, підготовці ще якісніших завдань для студентів та ознайомленню з новітніми трендами у

програмуванні. Для полегшення визначення наявності запозичень у програмному коді серед студентів можуть використовуватись системи перевірки робіт на плагіат, але у своїй більшості вони призначені для пошуку плагіату у звичайному тексті, і, відповідно, не пристосовані до аналізу програмного коду, який має свої особливості.

Мета статті – провести аналіз деяких сучасних доступних систем пошуку плагіату, розробити вимоги до власної системи та описати власну розроблену систему оцінки плагіату.

Практичне значення одержаних результатів. Результати, отримані в ході роботи над статтею, можна використовувати у практичних цілях для розробки систем пошуку плагіату у програмному коді і як навчальний матеріал.

Виклад основного матеріалу

1. Плагіат у програмному коді

Швидкий розвиток ІТ-індустрії та розширення її сфери впливу на інші технології, які до не давнього часу здавалися б вільні та незалежні від комп'ютерів, потребують нових кадрових спеціалістів. На фоні появи нових професій, виникла потреба в навчанні спеціалістів пов'язаних з програмуванням. Оскільки кількість студентів, які обрали програмування збільшилась, з'являється тенденція запозичення програмного коду, та видача його як власного авторського продукту. Тому з'являється потреба в якісній боротьбі з плагіатом в програмному продукті. На допомогу викладачам приходять системи виявлення плагіату. Від простих програм, які порівнюють два файли між собою, до автоматичних систем без безпосередньої участі викладача в ролі регулятора.

Плагіат програмного коду – це копіювання або відтворення вихідного коду без письмового дозволу від оригінального творця. Включає мінімальну адаптацію коду або включення фрагментів оригінального коду в свій код. Перетворення оригінального коду на іншу мову програмування можна вважати плагіатом, але тут все залежить від контексту. Використання генераторів коду студентами підпадає під зону ризику, зазвичай, виконання завдання в університетах не потребують використовування сторонніх засобів [27].

Замість того, щоб підозрювати всі роботи на плагіат, можна доручити цю роботу спеціальним системам для визначення плагіату. Розробці подібній системі, як раз і присвячена дана робота.

Плагіат в коді можна визначити як копіювання вихідного коду або окремих його частин без внесення якихось суттєвих змін в його структуру або з незначною його доробкою. Додавання коментарів до існуючого коду, зміни назв змінних, перестановка блоків функцій або класів.

2. Огляд програм для виявлення плагіату

Щоб створити власну системи оцінки програмного коду на плагіат, потрібно дослідити відомі системи для того, щоб мати представлення, як виглядають такі системи, та дослідити їх переваги й недоліки. На основі отриманих результатів спроектувати власний програмний продукт з уникненням типових недоліків. Для цього дослідимо такі відомі системи:

- MOSS [18];
- Codeqiury [5];
- Unicheck [23];
- CCFinderX [4].

Measure Of Software Similarity

MOSS – перша в світі програма для обчислення плагіату створена в 1994 Алексом Ейкеном [18]. Являє собою достатньо надійною програмою, дозволяє якісно перевірити програмний код на ознаки плагіату (рис. 1). Опирається на власну закриту базу даних з програмними кодами. Підтримує багато сучасних мов, включаючи Java, C, C++, Python, JavaScript. Використовує алгоритм відсіву – це алгоритм працює на основі виявлення унікальних особливостей документів, перетворюючи файли в набір хеш-значень, які називаються відбитками [1, 8].

Принцип роботи полягає в тому, що обраний файл порівнюється з власною базою даних. Виявляє подібність у програмах, та надає користувачу перелік подібних файлів до файлу, який перевіряють.

Переваги:

– легкий в роботі та безкоштовний;
– будь-хто може користуватися;
– підтримує багато мов програмування;
– власний ефективний алгоритм.

Недоліки:

– не дозволяє автоматично перевіряти файли на плагіат;
– оцінка плагіату на розсуд того, хто перевіряє;
– не має веб-версії.
– запускається за допомогою скриптів.

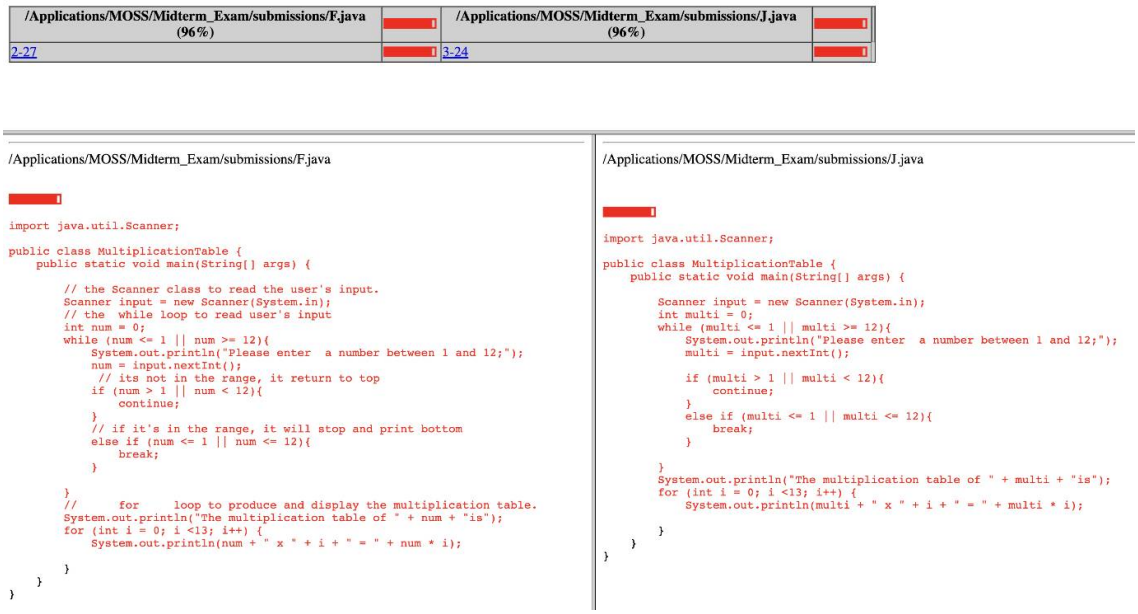


Рис 1. Інтерфейс системи MOSS

Codequiry

Codequiry – одна з найсучасніших системи перевірки плагіату в програмному коді. Детально вираховує унікальність коду, намагається виявити логічні шаблони та унікальний стиль коду (рис. 2). Перетворює програмний код у списки токенів, на другому етапі, шукає подібні кластери токенів у схожих групах [3]. Надає детальний інформацію про плагіат користувачу [5].

Переваги:

– має сучасний веб-інтерфейс, що спрощує роботу з системою;
– має велику базу публічного коду для перевірки на плагіат;

– підтримує велику кількість сучасних мов програмування;
– надає детальну інформацію про свою роботу.

Недоліки:

– платна система, передбачає помісячну підписку;
– демоверсія дозволяє перевірити до 20 файлів;

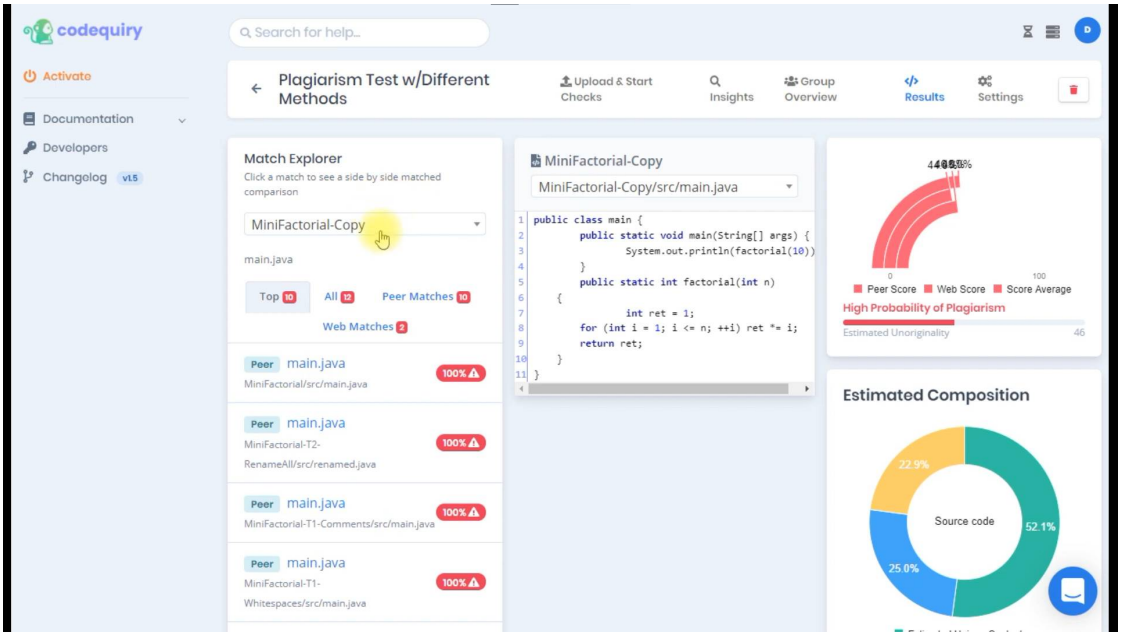


Рис 2. Інтерфейс системи перевірки плагіату Codequiry

Unicheck

Unicheck – широко відома система для перевірки плагіату в текстах. З підвищенням популярності на курси пов'язані з програмуванням, з 2019 року запустили бета-версію системи перевірки плагіату в програмному коді. Використовують власний алгоритм на основі штучного інтелекту [9]. Алгоритм базується на розумінні програмного коду так, як це робить комп'ютер. На момент написання роботи в 2021 році, система знаходиться в бета-тестуванні [23].

Переваги:

– надає детальний звіт про плагіат, відсотки, першоджерело;
– може порівнювати як два файли між собою, так і з базою даних;
– підтримує більшість сучасних мов програмування.

Недоліки:

– знаходиться в тестуванні;
– має ліміт на безкоштовну перевірку файлів, потрібно купувати додаткові спроби тестування.

CCFinderX

CCFinderX – детектор запозичення коду, виявляє клонування коду на різних мовах програмування (рис. 3). Покращена версія попереднього продукту CCFinder, нова архітектура дозволяє використовувати мультизадачність процесора користувача в виявленні плагіату, надає динамічну інформацію про запозичення [4].

Переваги:

– відкритий програмний продукт, користувач може модифікувати його під різні мови програмування;
– надає аналітичні дані про запозичення;

– має графічний інтерфейс.

Недоліки:

– не може працювати з файлами, які написані на декількох мовах програмування;
– складний запуск, необхідні знання роботи з терміналом;
– може порівнювати тільки два файли між собою.

3. Результати дослідів систем з оцінки плагіату

Дослідивши системи представлені вище, виявилось такі характерні ознаки:

– більшість з наведених систем мають сервер, окрім CCFinderX, він працює в локальному режимі з файлами, які користувач йому надав;
– більшість подібних систем мають власні публічні або приватні бази з зразками, з якими відбувається порівнювання, окрім CCFinderX, оскільки він автономний;
– мають власні вдосконалені алгоритми на основі відомих, деякі мають навіть штучний інтелект;
– всі наведені системи мають графічний інтерфейс.

Також виявились і недоліки подібних систем, наприклад:

– деякі екземпляри мають десктопний клієнт, тобто, потребують завантаження з мережі, та додаткових налаштувань, як MOSS та CCFinderX;
– сучасні системи такі, як Unicheck та Codequiry, передбачають користування на платній основі;
– не володіють централізованою системою відображення результатів, тобто викладач не може слідкувати за тими, хто здав завдання.

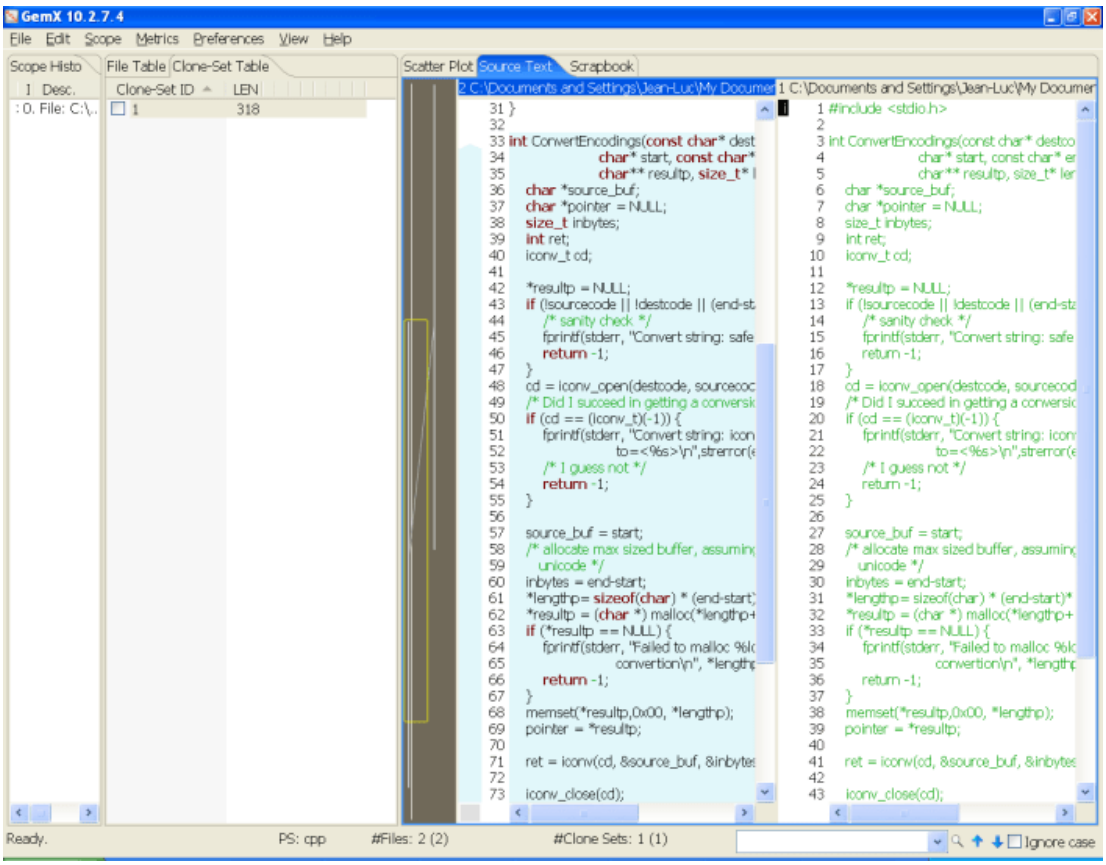


Рис 3. Інтерфейс системи перевірки коду на плагіат CCFinderX.

4. Визначення структури сервера

На основі оцінки вище зазначених систем, було прийняте рішення про створення власної системи оцінки плагіату у програмному коді, у вигляді серверу. Сервер реалізований в ході роботи, має такі властивості:

- можливість реєстрації та авторизації користувача;
- перевірка файлів на плагіат та отримання результату.

У сервері існує два типи ролей для користувачів:

- User – має такі можливості:
  - реєстрація та авторизація;
  - зберігання файлів та перевірка їх на плагіат;
  - перегляд свого акаунта та своїх результатів перевірки файлів.
- Admin – має всі вище зазначені можливості, а також:
  - перегляд всіх користувачів зареєстрованих на сервері;
  - перегляд їх результатів перевірки коду на плагіат.

Щоб досягти результату в ході роботи, будуть використані наступні технології:

- Java 11 – основна мова програмування [13];
- IntelliJ IDEA – середовище розробки з підтримкою мови програмування Java [11];
- Gradle – збирач пакетів [10];
- Spring Boot 2 Framework – контейнер на основі якого розгортається сервер [22];
- REST API – визначення загальної архітектури сервера [21];
- JSON – обмін інформацією з сервером [14];
- OAuth2 – реєстрації та авторизації користувачів на сервері [20];
- JWT – обмін приватною зашифрованою інформацією між користувачем та сервером [15];

5. RESTful API

Application Programing Interface (API) – це набір визначень і протоколів, за допомогою якого відбувається взаємодія між компонентами програмного забезпечення. Цей архітектурний підхід обертається навколо надання програмного інтерфейсу набору послуг для різних програм, які обслуговують різних споживачів [21].

При використанні архітектурного підходу в контексті веб-розробки, API визначається як набір специфікацій, таких як передача повідомлення запиту до ресурсу, за допомогою Hypertext Transfer Protocol (HTTP), визначенням структури запитів-відповідей, з використанням технологій Extensible Markup Language (XML) або JavaScript Object Notation (JSON).

Representational State Transfer (REST) – архітектурний стиль для розробки програмного забезпечення, містить набір архітектурних обмежень, не являє собою протокол чи стандарт. Розробники API можуть реалізовувати цей стиль декількома шляхами.

Коли відбувається запит до сервера чи програмного забезпечення за допомогою RESTful API, клієнт передає інформацію про стан ресурсу, та подає інформацію у одному з форматів через HTTP: JSON, XML, JavaScript-запит, звичайний текст, тощо.

Важливо розуміти дещо про використання HTTP у RESTful API: заголовки та параметри відіграють важливу роль у обміні інформацією між ресурсами, оскільки вони містять важливу інформацію ідентифікатора про метадані запиту, авторизацію, тип контенту, єдиний ідентифікатор ресурсу, кешування, куки, код відповідей, інше. Існують заголовки запитів та відповідей, кожні з них містять власну інформацію про з'єднання HTTP та коди станів.

Найбільш популярний варіант надсилання тіла інформації – це JSON, оскільки він не прив'язаний до певної мови програмування, зручний у використанні, легкий у розумінні як людьми, так і машинами. У своїй основі містить пари — ключ та значення, які використовуються у формуванні тіл запитів та відповідей, у вигляді масивів відповідних пар [14].

Для того, щоб API розглядався як RESTful, має слідувати таким критеріям:

- архітектура, що складається з клієнтів, серверів та ресурсів, із запитами, які використовують протокол HTTP;
- всі комунікації між клієнтом та сервером відбуваються без зберігання стану та сесій, кожен запит та відповідь є відокремленими та незалежним;
- дані кешуються, для полегшення навантаження на сервер;
- стандартизація запитів та відповідей між клієнтом та сервером відповідно;
- багатошарова система з розподіленням обов'язків, ієрархічна структура невидима для клієнтів.

6. JSON Web Token

JSON Web Token (JWT) – це відкритий стандарт (RFC 7519), який визначає компактний та автономний спосіб передачі безпечної інформації між сторонами з використанням стандарту JSON. Інформації яку містить в собі токен можна довіряти, оскільки вона має цифровий підпис. Токен може бути підписаний за допомогою публічного або секретного ключа з використанням шифрувальних алгоритмів [15].

Найбільш поширений спосіб використання JWT – авторизація. Єдиний вхід (Single Sign-in) – це функція, яка широко використовує JWT, що дозволяє один раз ввійти в систему та використовувати токен, допоки не пройде час його існування. Як тільки користувач ввійде в систему, він отримає токен з його особистою інформацією, яку зможе використовувати для доступу до захищених служб та ресурсів системи.

7. Структура JSON Web Token

У компактній формі, токен складається з трьох частин, які шифруються алгоритмом Base64Url та відділені крапкою [15]:

- заголовок – складається з двох частин: тип токену та алгоритм який використовується для кодування його;
- корисне навантаження – інформація для та про користувача, ким виданий токен, кому виданий, тривалість життя токена та інші;
- підпис – вираховується на основі заголовку, корисного навантаження та обраного алгоритму кодування з використанням публічного або приватного ключа.

Переваги використання JWT в порівнянні з підходом використовування сесій такі:

- JWT – не потребує створення сесії та зберігання додаткових даних про сесію. Все що необхідно серверу – це перевірити підпис токена;
- сервер може не займатися створенням токенів, а надати їх іншим ресурсам, наприклад серверу авторизації;
- в токенах можна зберігати додаткову інформацію про користувачів;
- токен надає можливість доступу до різних сервісів, які зможуть прочитати цей токен.

8. Протокол авторизації OAuth2

OAuth2 – відкритий стандарт авторизації, який дозволяє делегувати логіку авторизації перевіреним сервісам, як Google, Amazon, Facebook та іншим, які

виступають в ролі посередника між користувачем та сторонніми сервісам. Дозволяє користувачам отримати доступ до сторонніх сервісів, та ділитися своєю конфіденційною інформацією з ними (рис. 4). Перевага такого способу полягає в тому, що ви не розкриваєте свої облікові дані сторонньому ресурсу [20].

Рис 4. Абстрактний опис протоколу OAuth2

OAuth2 визначає чотири ролі:

- Власник ресурсу – користувач, який авторизує додаток для доступу до свого акаунту;
- Клієнт – додаток, який хоче отримати доступ до акаунту користувача, перед доступом до інформації користувача, додаток має бути авторизований користувачем, а авторизація має буде схвалена сервером авторизації;
- Сервер авторизації – сервер який видає токени клієнту для доступу до інформації користувача;
- Сервер ресурсів – сервер, на якому знаходяться захищені ресурси, до яких клієнт звертається використовуючи токен доступу користувача.

Послідовність роботи протоколу:

- додаток запитує у користувача дозвіл на авторизацію для доступу до серверу ресурсів;
- користувач надає дозвіл додатку на авторизацію;
- додаток запитує авторизаційний токен у сервера авторизації, шляхом надання інформації про себе та дозвіл користувача;
- якщо істинність додатку підтверджена і дозвіл на авторизацію дійсний, то сервер створює токен доступу. Процес авторизації завершений;
- додаток запитує захищений ресурс у серверу ресурсів, надаючи при цьому токен доступу;
- якщо токен дійсний, сервер ресурсів надає ресурс додатку.

Перед тим, як почати використовувати OAuth2, необхідно зареєструвати свій додаток на сервісі. В роботі в якості авторизаційного сервісу використовується Google API. Тому далі буде приклад як зареєструвати додаток в Google. Це робиться в розділі «developer» сайту Google API, де необхідно надати наступну інформацію:

- назва додатку;
- сайт додатку (глобальний або локальний шлях);
- callback URL – це шлях, на який сервіс буде перенаправляти користувача після авторизації або відмови.

Після реєстрації додатку, сервіс створить облікові дані клієнта — ідентифікатор клієнту та секрет клієнта. Ідентифікатор клієнта являє собою публічну доступну строку, яка використовується API сервісом для автентифікації додатку, а також для створення авторизаційних URL для користувачів. Секрет клієнта використовується для автентифікації істинності додатку для API сервісу, коли додаток запитує доступ до акаунту користувача. Секрет повинний бути відомим тільки додатку і API сервісу.

9. Нормалізація та токенізація програмного коду

Щоб зрозуміти як правильно працювати з програмним кодом, потрібно зрозуміти, що він представляє з себе [16].

Програма – це набір команд, інструкцій, даних, призначених для виконання функцій необхідних користувачу, використовуючи для цього апаратне забезпечення операційної системи, для якої вона розроблена. Програми складаються з синтаксичних одиниць, які відповідають правилам конкретної мови програмування: методів, класів та бібліотек, які, у свою чергу, складаються з операторів або інструкцій для розв’язання конкретної задачі (стандарт ISO/IEC 2382-1:1993) [12].

Розглянувши, що таке програма, зрозуміло, що програма – це не просто текст, який можна було б порівняти послівно, вона своєрідну структуру, синтаксис, який залежить від мови програмування, але в більшості, якщо це не вузькоспеціалізована мова під конкретну ситуацію, в них можна легко простежити закономірності, наприклад, створення циклу чи оператора розгалуження. Тобто програма містить спеціальний набір не змінних шаблонів, які відрізняються за синтаксисом, але не за суттю, тому можливо привести до одного стандартизованого виду, що буде зручний для аналізу та подальшого порівняння в алгоритмі. Тому перед цим потрібно провести нормалізацію та токенізацію.

Нормалізація – процес перетворення вихідного коду, таким чином, щоб опустити усі несуттєві частини програмного коду для алгоритму, який буде перевіряти код на плагіат [16]. Включає:

- переведення усіх символів в нижній регістр;
- заміну усіх символів табуляції на пробіли, видалення символів переводу рядка;
- заміну роздільних знаків на пробіли;
- видалення однорядкових та багаторядкових коментарів;
- заміну багатьох пробільних символів одним.

Нормалізація дозволяє уникнути хибного спрацювання або навпаки не спрацювання алгоритму при підрахуванні алгоритму. Наприклад, хибне спрацювання може виникнути, якщо додати коментарі, які збільшать розмір коду, або змінити назву змінної, тому для цього приводимо програмний код до одного уніфікованого виду. Після проведення нормалізації вихідний код готовий для подальшої процедури, перетворення його на список токенів.

Лексема (токен) – послідовність машинних символів вихідного коду програми,

яке має спільне сукупне значення. В мові програмування, частіше використовують слово «токен», тому для зручності, в подальшому будемо використовувати його [26].

Будь-яка мова програмування складається з наступних лексем:

- зарезервовані слова (модифікатори доступу, типи змінних та інше);
- ідентифікатори (назва змінних, методів, класів);
- числові, буквенні, рядкові константи;
- знаки операторів;
- коментарі;
- роздільники.

Процес токенізації буде проходити з використанням спеціальної бібліотеки для парсингу програмного коду – ANTLR4 [2]. Потужний інструмент, використовується для аналізу текстів, створення власних мов програмування, та компіляторів. В випадку системи для оцінки програмного коду на плагіат, дозволяє на 100% правильно перетворити вихідний код в список токенів. Для роботи цієї бібліотеки необхідно мати два файли. В одному файлі містяться всі можливі лексеми для конкретної мови програмування, в іншому містяться типове використання лексем в конструкціях. Алгоритм роботи бібліотеки ANTLR:

- генерація класу, який буде займатися розбором коду на токени;
- генерація синтаксичного аналізатора;
- перевірка вихідного коду, на наявність помилок, з подальшим повідомленням користувачу;
- створює синтаксичне дерево, генерує два об'єкта Visitor та Listener, для обходу дерева.

У нашому випадку нам потрібно отримати з дерева послідовність токенів, для подальшого використання їх в алгоритмі визначення плагіату.

10. Алгоритм Вагнера-Фішера

Головним алгоритмом, який буде виконувати перевірку програмного коду на плагіат – алгоритм Вагнера-Фішера. В основу якого покладене таке поняття як відстань Левенштейна.

Відстань Левенштейна – числове значення, яке дозволяє вирахувати різницю між двома послідовностями символів. Визначається як мінімальна кількість операцій (вставки, видалення, заміни), необхідних для перетворення одної послідовності у іншу [25].

Нехай два рядки S_1 і S_2 – довжиною m і n – рядки відповідно над деяким алфавітом (у нашому випадку множина лексем мови програмування Java), тоді відстань Левенштейна $lev(S_1, S_2)$ можна підрахувати за наступною формулою:

$$lev(i,j) = \begin{cases} 0, & i = 0, j = 0 \\ i, & j = 0, i > 0 \\ j, & i = 0, j > 0 \\ min\begin{cases} lev(i, j-1)+1 \\ lev(i-1, j)+1 \\ lev(i-1, j-1)+m(S_1[i], S_2[i]) \end{cases} & j > 0, i > 0 \end{cases},$$

де $m(a, b)$ дорівнює нулю, якщо $a = b$ і одиниці в інакшому випадку; $min\{a, b, c\}$

повертає найменше з трьох аргументів.

Тут крок по *i* означає видалення (D) із першого рядка, по *j* – вставку (I) у перший рядок, а крок по обох індексах означає заміну символу (R) або відсутність змін (M). Відстань Левенштейна знаходиться у останній комірці матриці *lev (lev(m, n))*.

Приклад розрахунку відстані Левенштейна подано у таблиці 1.

Алгоритм, запропонований Вагнером та Фішером, містить в собі покращення внесені в оригінальний алгоритми і дозволяє вирахувати найкоротшу відстань Левенштейна між двома рядками, з меншими витратами пам'яті за допомогою динамічного обчислення масивів, скоротивши необхідну пам'ять до *min{m, n}* [9]. Тобто, не потрібно тримати в пам'яті матрицю розміром *m·n*, обчислення достатньо мати два вектори довжиною *m* або *n*.

Таблиця 1

Результат роботи алгоритму Левенштейна

		М	**О**	**Р**	**Г**	**Е**	**Н**	**Ш**	**Т**	**Е**	**Р**	**Н**
	0	1	2	3	4	5	6	7	8	9	10	11
Л	1	1	2	3	4	5	6	7	8	9	10	11
Е	2	2	2	3	4	4	5	6	7	8	9	10
В	3	3	3	3	4	5	5	6	7	8	9	10
Е	4	4	4	4	4	4	5	6	7	7	8	9
Н	5	5	5	5	5	5	4	5	6	7	8	8
Ш	6	6	6	6	6	6	5	4	5	6	7	8
Т	7	7	7	7	7	7	6	5	4	5	6	7
Е	8	8	8	8	8	7	7	6	5	4	5	6
Й	9	9	9	9	9	8	8	7	6	5	5	6
Н	10	10	10	10	10	9	8	8	7	6	6	**5**

Псевдокод алгоритму Вагнера-Фішера:

```
function WagnerFischer(char s[1..m],char t[1..n]):
  declare int v0[n+1]
  declare int v1[n+1]
  for i from 0 to n:
    v0[i] := i;
  for i from 0 to m-1:
    v1[0] := i+1;
  for j from 0 to n-1:
    deletionCost := v0[j+1] + 1
    insertionCost := v1[j] + 1
```

```
      if s[i] = t[j]:
      substitutionCost := v0[j]
    else:
      substitutionCost := v0[j] + 1
    v1[j+1] := minimum(deletionCost,
                       insertionCost,
                       substitutionCost)
  swap v0 with v1
return v0[n]
```

З метою отримання результату роботи алгоритму у прийнятному і зрозумілому виді, перетворимо число Левенштейна у відсотки за допомогою наступної формули:

$$PlagiarizedValue = \left\{1 - \frac{LD}{Max(FFLS, SFLS)}\right\} \cdot 100,$$

де

LD (Levenshtein distance) – відстань Левенштейна між двома файлами;
FFLS (First token list size) – довжина першого списку токенів;
SFLS (Second token list size) – довжина другого списку токенів.

11. Структура пакетів сервера

Сервер складається з чотирьох головних пакетів:

- auth – містить в собі логіку реєстрації та авторизації користувачів;
- detector – містить в собі логіку, для перевірки програмного коду на плагіат, нормалізацію, токенізацію та алгоритм Вагнера-Фішера;
- uploadfile – відповідає за можливість завантаження файлів з програмним кодом на сервер;
- plagiarism – відповідає за виведення інформації про плагіат для користувача;
- resources – містить службові файли для роботи серверу.

Пакет auth

Так, як всі шляхи на сервері захищені, і потребують автентифікації, було прийняте рішення додати на сервер логіку реєстрації та авторизації.

В даному пакеті знаходиться логіка, яка відповідає за авторизацію на сервері та реєстрацію користувача на сервері. Реєстрація відбувається двома шляхами: Basic Auth та OAuth2.

Перша, відбувається за специфікацією Basic Auth, тобто, користувач створює власний обліковий запис з використанням логіну та паролю. Цей процес відбувається між користувачем та сервером безпосередньо, вся інформація про користувача зберігається на сервері.

Друга, відбувається за специфікацією OAuth2, тобто між користувачем та сервером, знаходиться третя сторона, яка бере на себе відповідальність за реєстрацію. Користувач ініціює авторизацію, надаючи серверу згоду про авторизацію, в даному випадку компанії Google. За згодою користувача сервер бере необхідні облікові дані про особу в провайдера Google, та оброблює їх. Перевага даного методу реєстрації та авторизації в тому, що на сервері не зберігається пароль користувача, і весь процес проходить між серверами. Також за для безпеки, токен доступу (Access token) на сервері перетворюється на власний токен за стандартом JWT. Навіть якщо зловмисник поцупить цей токен і розкодує, максимум з корисної інформації, він отримає ім’я користувача та його електронний адрес. З даним токеном зловмисник має можливість доступу тільки до сервера перевірки вихідного коду на плагіат і більше нікуди,

відредагувати акаунт він немає можливості, такий функціонал не передбачений на сервері.

Після авторизації, обмін інформацією між клієнтом та сервером відбувається за допомогою закодованого токена стандарту JWT. З нього сервер дістає всю необхідну інформацію, перевіряє права доступу та час життя токену. Якщо все пройшло успішно, користувач має право користуватися захищеними ресурсами серверу. Якщо сервер не зможе прочитати токен, чи час його життя закінчився, користувачу потрібно буде знову авторизуватися в системі.

Пакет detector

В ньому знаходиться вся логіка, яка відповідає за нормалізацію та токенізацію вихідних файлів, а також за їх перевірку на плагіат.

Процес знаходження плагіату відбувається наступним чином:

- відбувається нормалізація та токенізація вхідного файлу;
- поточний вхідний файл у вигляді списку токенів, порівнюється за допомогою алгоритму з списками токенів в базі даних;
- отриманий результат передається користувачу.

Нормалізація та токенізація проходить з використанням бібліотеки ANTLR. На основі двох файлів JavaLexer.g4 та JavaParser.g4, де описані основні граматичні лексеми та правила створення синтаксичних конструкцій мови програмування Java, створюється Lexer та Parser. Лексер в свою чергу за допомогою парсера розбиває файл на токени, які в подальшому використовує алгоритм Вагнера-Фішера. Перевіряючи всі файли з поточним, виявляє найбільший відсоток плагіату та повертає результат. Результат має наступний вигляд:

- plagiarism – відсоток запозичень в коді від 0 до 100;
- lastCheckedTime – дата коли відбувалась перевірка файлу.

Пакет uploadfile

Даний модуль відповідає за завантаження файлу на сервер. Коли користувач завантажив файл на сервер, на сервері копія файлу відправляється в модуль detector, де перетворюється в список токенів та повертається назад в модуль uploadfile, зі свого боку модуль зберігає файл та його токенізований вигляд в базу даних.

Також володіє можливістю віддавати файли на завантаження користувачу. Надає можливість адміністратору переглядати файли, які користувач завантажив.

Пакет plagiarism

Пакет plagiarism відповідає за співпрацю API серверу з модулем детектора. Містить контролер в який користувач надсилає інформацію про файл, який хоче перевірити на плагіат.

Всі результати перевірок зберігаються в базі даних та прикріпляються до відповідного файлу, який користувач хоче перевірити. Дана можливість надає користувачу переглянути пізніше результати роботи, не очікуючи певний час, який може знадобитися на перевірку. Користувач має змогу перевіряти тільки ті файли, які він надав серверу. Файли інших користувачів та їх результати перевірки доступні для перегляду належать тим користувачам, які їх передали до системи, а також адміністратору.

Пакет resources

Містить в собі службові файли для роботи серверу. Наприклад, sql-файли, так звані файли міграції, необхідні при першому запуску сервера в новому місці. В даних

файлах описана структура бази даних, структура таблиць та всі зв'язки між ними.

Також тут знаходяться конфігураційні файли, які містять важливі властивості для роботи сервера, наприклад, шлях до бази даних, список дозволених адрес, які можуть звертатися до сервера, налаштування OAuth2.

Висновки

У статті розглянуто відомі системи для виявлення плагіату в програмному коді, а саме: MOSS, Codeqiury, Unicheck, CCFinderX. Представлено опис переваг та недоліків кожної системи. Кожна з цих систем має унікальну архітектуру та інтерфейс, але всі мають спільний алгоритм роботи:

- приймають вхідний файл на оцінку;
- перетворюють його в список токенів;
- використовують власний алгоритм для оцінки коду на плагіат;
- надають результат користувачу;

Створено власний сервер для оцінки коду на плагіат з урахуванням переваг та недоліків.

У ході роботи було розглянуло та з'ясовано наступне:

- розглянуто, що таке плагіат у програмному коді;
- розглянуто, відомі системи пошуку плагіату та їх реалізації;
- розроблено алгоритм на основі токенізованого представлення;
- реалізовано сервер для визначення плагіату у коді, розроблених мовою програмування Java, яка може виступати в ролі API для додатків.

Список використаної літератури:

1. Aiken A., Schleimer S., Wikerson D. Winnowing: Local Algorithm for Document Fingerprinting. // Proceeding of ACMSIGMOD Int. Conference on Management of Data. San Diego. 2003. P. 76-85. ACM Press. New York, USA. 2003.
2. ANTLR 4, ANother Tool for Language Recognition [Електронний ресурс] – Режим доступу: https://www.antlr.org/
3. Baxter I., Yahin A., Moura L., Anna M.S., Bier L. Clone Detection Using Abstract Syntax Trees. // Proceedings of ICSM. IEEE. 1998.
4. CCFinderX. [Електронний ресурс] – https://github.com/gpoo/ccfinderx
5. Codequiry. [Електронний ресурс] – Режим доступу: https://codequiry.com
6. Faidhi J.A.W., Robinson S.K. An Empirical Approach for Detecting Program Similarity within a University Programming Environment. //Computer and Education. 1987. 11(1). P. 11-19.
7. Heckel P. A. Techique for Isolationg Differences Between File. // Communications of the ACM 21(4). April 1978. P. 264-268.
8. Heinzte N. Scalable Document Fingerprinting. // In 1996 USENIX Workshop of Electronic Commerce, 1996.
9. Huang X., Hardison R.C., Miller W. A Space-efficient Algorithm for Local Similarities. // Computer Applications in the Biosciences 6. 1990. P 373-381.
10. Gradle. [Електронний ресурс] – Режим доступу: https://gradle.org/
11. IntelliJ IDEA Ultimate Edition. [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/ru-ru/idea/
12. ISO/IEC 2382-1:1993 Information Technology – Vocabulary – Part1: Fundamental terms. [Електронний ресурс] – Режим доступу: https://www.iso.org/ru/standard/7229.html
13. Java 11 JDK. Oracle. [Електронний ресурс] – Режим доступу: https://www.oracle.com/java/technologies/downloads/
14. JSON. [Електронний ресурс] – Режим доступу: https://www.json.org/json-en.html
15. JWT. [Електронний ресурс] – Режим доступу: https://jwt.io/
16. Lexical Analysis. [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Lexical_analysis
17. Mishne G., M. de Rijke. Source Code Retrieval using Conceptual Similarity // Proceedings RIAO. Vaucluse. 2004. P. 539-555.
18. MOSS A System for Detecting Software Similarity. [Електронний ресурс] – Режим доступу: https://theory.stanford.edu/~aiken /moss/

19. Prechelt L., Malpohl G., Philippsen M. JPlag: Finding Plagiarism Among a Set of Programs. // Technical Report No. 1/00, University of Karlsruhe, Department of Informatics. March 2000.
20. OAuth2. RFC 6749 [Електронний ресурс] – Режим доступу: https://datatracker.ietf.org/doc/html/rfc6749
21. RESTful API Tutorial. [Електронний ресурс] – Режим доступу: https://restfulapi.net/
22. Spring Boot 2 Framework. [Електронний ресурс] – Режим доступу: https://spring.io/
23. Unicheck. [Електронний ресурс] – Режим доступу: https://unicheck.com/ua/blog/innovative-tool-for-checking-source-code-for-plagiarism
24. Wise M.J. String similarity via greedy string tiling and running Karb-Rabin matching. // Dept. of CS, University of Sidney. December 1993.
25. Відстань Левенштейна. [Електронний ресурс] – Режим доступу: http://uk.wikipedia.org/wiki/Відстань Левенштейна
26. Лексема. [Електронний ресурс] – 2021. – Режим доступу: http://uk.wikipedia.org/wiki/Лексема
27. Плагіат. [Електронний ресурс] – 2021. – Режим доступу: http://uk.wikipedia.org/wiki/Плагіат

References:

1. Aiken A., Schleimer S., Wikerson D. Winnowing: Local Algorithm for Document Fingerprinting. // Proceeding of ACMSIGMOD Int. Conference on Management of Data. San Diego. 2003. P. 76-85. ACM Press. New York, USA. 2003.
2. ANTLR 4, ANother Tool for Language Recognition [Електронний ресурс] – Режим доступу: https://www.antlr.org/
3. Baxter I., Yahin A., Moura L., Anna M.S., Bier L. Clone Detection Using Abstract Syntax Trees. // Proceedings of ICSM. IEEE. 1998.
4. CCFinderX. [Електронний ресурс] – https://github.com/gpoo/ccfinderx
5. Codequiry. [Електронний ресурс] – Режим доступу: https://codequiry.com
6. Faidhi J.A.W., Robinson S.K. An Empirical Approach for Detecting Program Similarity within a University Programming Environment. //Computer and Education. 1987. 11(1). P. 11-19.
7. Heckel P. A. Techique for Isolationg Differences Between File. // Communications of the ACM 21(4). April 1978. P. 264-268.
8. Heinzte N. Scalable Document Fingerprinting. // In 1996 USENIX Workshop of Electronic Commerce, 1996.
9. Huang X., Hardison R.C., Miller W. A Space-efficient Algorithm for Local Similarities. // Computer Applications in the Biosciences 6. 1990. P 373-381.
10. Gradle. [Електронний ресурс] – Режим доступу: https://gradle.org/
11. IntelliJ IDEA Ultimate Edition. [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/ru-ru/idea/
12. ISO/IEC 2382-1:1993 Information Technology – Vocabulary – Part1: Fundamental terms. [Електронний ресурс] – Режим доступу: https://www.iso.org/ru/standard/7229.html
13. Java 11 JDK. Oracle. [Електронний ресурс] – Режим доступу: https://www.oracle.com/java/technologies/downloads/
14. JSON. [Електронний ресурс] – Режим доступу: https://www.json.org/json-en.html
15. JWT. [Електронний ресурс] – Режим доступу: https://jwt.io/
16. Lexical Analysis. [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Lexical_analysis
17. Mishne G., M. de Rijke. Source Code Retrieval using Conceptual Similarity // Proceedings RIAO. Vaucluse. 2004. P. 539-555.
18. MOSS A System for Detecting Software Similarity. [Електронний ресурс] – Режим доступу: https://theory.stanford.edu/~aiken /moss/
19. Prechelt L., Malpohl G., Philippsen M. JPlag: Finding Plagiarism Among a Set of Programs. // Technical Report No. 1/00, University of Karlsruhe, Department of Informatics. March 2000.
20. OAuth2. RFC 6749 [Електронний ресурс] – Режим доступу: https://datatracker.ietf.org/doc/html/rfc6749
21. RESTful API Tutorial. [Електронний ресурс] – Режим доступу: https://restfulapi.net/
22. Spring Boot 2 Framework. [Електронний ресурс] – Режим доступу: https://spring.io/
23. Unicheck. [Електронний ресурс] – Режим доступу: https://unicheck.com/ua/blog/innovative-tool-for-checking-source-code-for-plagiarism
24. Wise M.J. String similarity via greedy string tiling and running Karb-Rabin matching. // Dept. of CS, University of Sidney. December 1993.
25. Levenstein's distance: http://uk.wikipedia.org/wiki/Відстань_Левенштейна. [in Ukrainian]
26. A token: http://uk.wikipedia.org/wiki/Лексема. [in Ukrainian]
27. Plagiarism: http://uk.wikipedia.org/wiki/Плагіат. [in Ukrainian]

KHARCHENKO Vitalii,
Student, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
DIDKOWSKY Ruslan,
Doctor of Technical Sciences, Associate Professor, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
SERDIUK Oleksandr,
Candidate of Economic Sciences, Associate Professor, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine

DEVELOPMENT THE SERVER-SIDE PART OF THE SYSTEM OF CHECKING SOFTWARE CODE FOR THE PRESENCE OF PLAGIARISM

Summary. Introduction. *Borrowing code among students is a problem that worries many university professors who teach disciplines related to the study of programming. Unfortunately, it is difficult for a teacher to personally follow the uniqueness of all students' work. In addition, a large part of the working time has to be spent on checking the work, instead of devoting this time, for example, to preparing even better tasks for students and getting acquainted with the latest trends in programming. To facilitate the detection of borrowings in the program code among students, plagiarism checking systems can be used, but most of them are designed to search for plagiarism in plain text, and, accordingly, are not adapted to the analysis of program code, which has its own characteristics.*

The purpose *of the article is to analyze some modern available plagiarism search systems, to develop requirements for one's own system and to describe one's own developed plagiarism evaluation system.*

Results. *The paper considers 4 programs designed to detect plagiarism: Measure Of Software Similarity (MOSS), Codequiry, Unicheck, CCFinderX. For each of the programs the identified advantages and disadvantages in comparison with other studied programs are given. The comparative analysis revealed the following characteristics of the studied programs: most of these systems have a server; most of these have their own public or private databases with samples to compare with; all systems have their own advanced algorithms based on known ones, some even have artificial intelligence; all of these systems have a graphical interface. The following shortcomings were also identified: some instances have a desktop client, i.e., require network downloads and additional settings, such as MOSS and CCFinderX; modern systems, such as Unicheck and Codequiry, provide for paid use; do not have a centralized system for displaying results, i.e. the teacher can not follow those who passed the task.*

Based on the comparative evaluation of the analyzed systems, a proprietary plagiarism evaluation system was created in the program code in the form of a server, which has the following properties: the ability to register and authorize the user; check files for plagiarism and get the result. There are two types of roles for users on the server: User and Admin. To achieve the result, the following technologies were used: the main programming language - Java 11; development environment - IntelliJ IDEA; Gradle package collector; Spring Boot 2 Framework container; technologies REST API, JSON, OAuth2, JWT. The main algorithm that checks the program code for plagiarism is the Wagner-Fisher algorithm, which is based on such a concept as Levenstein's distance. The article describes Levenstein's algorithm and gives an example of its use to calculate the distance between two lines.

Conclusion. *As a result of the work several known systems of plagiarism detection in the program code are considered, the description of advantages and disadvantages of each system is presented. It is determined that all considered systems have a common algorithm: accept the input file for evaluation; turn it into a list of tokens; use their own algorithm to evaluate the code for plagiarism; provide the result to the user. An algorithm based on a token representation is developed. Implemented its own server for evaluating plagiarism code, taking into account the advantages and disadvantages of the analyzed systems, the structure of which is given in the article. The resulting software product can act as a server that provides an API for applications designed to check for plagiarism in the software code.*

Keywords: *plagiarism, plagiarism in program code, plagiarism check system, server system.*

Одержано редакцією 20.10.2021 р.
Прийнято до публікації 08.12.2021 р.

УДК 378:517

DOI 10.31651/2076-5886-2021-1-84-90

PACS 02.60.-x

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

ВИКОРИСТАННЯ АПАРАТУ АЛГЕБРИ ТА ГЕОМЕТРІЇ У ПРОЦЕСІ РОЗВ'ЯЗАННЯ ПРИКЛАДНИХ ЗАДАЧ ПІД ЧАС НАВЧАННЯ ФАХІВЦІВ У СФЕРІ ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

Вдале поєднання математичного апарату, сучасної інженерії та високих технологій дає змогу реалізувати сміливі ідеї та втілити надсучасні рішення у життя кожного з нас.

У статті показано, що оволодіння апаратом алгебри для фахівців з аналізу даних є вкрай важливим аспектом для успішного вирішення задач прикладного характеру, крім цього, це дозволить у майбутньому перспективно реалізувати себе в кар'єрному плані. В якості підтвердження актуальності обраного напряму дослідження, вказані провідні науковці та їх роботи по тематиці даної статті.

Окреслено основні сучасні напрямки використання математичних дисциплін для розвитку інформаційних систем та технологій. Розроблено алгоритм виконання елементарної задачі теорії матричних ігор та її програмну реалізацію у середовищах Python та Javascript.

Розглянуто розв'язання задачі прикладного характеру, із використанням алгебраїчних підходів, стосовно побудови моделі мережі транспортних потоків та проаналізовано отримані результати.

Ключові слова: *алгебра та геометрія, математична модель, аналіз даних, інформаційні системи, новітні технології.*

Вступ

Стрімкий розвиток інформаційних систем та новітніх технологій вносить динаміку позитивних змін у кожну сферу життєдіяльності людини. Прогресивні інженерні рішення формують нові підходи до автоматизації процесів, наприклад дозволяють комфортно подорожувати по новій місцевості із застосуванням карт, онлайн-перекладачів, дронів тощо. Для успішного функціонування вказаних новітніх технологій передбачається використання сучасного апарату математичних дисциплін [1].

Опанування дисципліни «Алгебра та геометрія» є фундаментальним етапом на шляху до вивчення предметів циклу професійного спрямування спеціальності 126 «Інформаційні системи та технології». Це можна пояснити тим, що більшість задач програмування зводиться до розв'язання систем лінійних рівнянь, побудови матриць та дій над ними, виконання базових операцій над векторами тощо [2].

Крім цього, для більшості ІТ-компаній, рівень математичних знань є індикатором до того, настільки майбутній працівник буде компетентний у своїй роботі.

Разом з цим, безпосереднє вивчення теорії та розв'язання математичних задач, під час навчання, призводить до того, що студент практично втрачає зв'язок між витраченими роками на оволодіння математичних дисциплін та їх значимістю у майбутній кар'єрі.

У зв'язку з цим, виникає необхідність до вивчення питання використання апарату алгебри та геометрії з точки зору практичної реалізації можливих рішень для задач прикладного характеру.

Починаючи із 30-х років XIX століття простежується тенденція до розвитку обчислювальних машин, де ключовим є апарат математичної логіки, який закладає основи існуючих мов програмування, які грунтуються на алгоритмічному виконанні команд.

Обрана тематика дослідження є досить актуальною про що свідчить поява, останнім часом, великої кількості публікацій по даному напрямі, зокрема з рядом нових результатів можна ознайомитися у роботах D.C. Lay, J. Hefferon, D. Poole, І.В. Алєксєєва, К. Бека, В.В. Булдигіна, В.І. Гур'єва, В.В. Корнещука, О.Я. Кучерука, З.О. Сердюк, В.І. Трофименко, Д.Є. Щедролосьєва.

Важливу роль у формуванні світогляду майбутніх програмістів відіграє література видатних сучасників, а саме: Г. Бейтсона, Д. Гілдера, Д. Гофстедтера, П. Діамандіса, С. Котлера, Р. Пенроуза, К. Стейнера та багато інших.

Варто відмітити, що всі існуючі сфери точних наук мають бути зосереджені на розвиток нестандартного мислення, що дозволить програмісту самостійно розробляти унікальні алгоритми та програми без використання шаблонів [3].

Метою статті є реалізація апарату алгебри та геометрії у процесі вирішення прикладних задач при підготовці фахівців у сфері інформаційних систем та технологій.

Виклад основного матеріалу

За допомогою систем лінійних рівнянь можна змоделювати більшу частину задач прикладного характеру, а саме: проектування інженерних споруд; обробку результатів вимірів або опитувань; планування виробничих процесів чи інших технічних завдань; прогнозування наукових експерементів тощо [4]. Набуття вмінь та навичок, під час навчання в університеті, до вирішення такого роду задач, відкриває перспективні кар'єрні можливості перед фахівцями аналізу даних. Для цього, необхідно підтримувати неперервний зв'язок між теоретичним та прикладним застосуванням навчального матеріалу, що дозволить збільшити внутрішню мотивацію до вивчення дисципліни «Алгебра та геометрія». Проілюструємо основні аспекти залучення математичних дисциплін, через призму сучасних напрямів, до розвитку інформаційних систем та технологій.

Аналізуючи ринок IT-індустрії, можна споглядати, що одним із найцікавіших та прогресивних напрямків у сфері інформаційних систем та технологій є розробка комп'ютерних ігор. Значна частина, а саме понад 60% мобільних ігор виготовлені на базі платформи Unity. Останнім часом, простежується тенденція до збільшення попиту на VR технології нового покоління – віртуальні кімнати, простори.

Крім цього, останні досягнення у сфері IT-технологій дозволяють використовувати віртуальні технології для будівництва, облаштування інтер'єрів, кінематографії, автомобілебудування тощо. Якщо досліджувати практичну сторону таких новітніх реалізацій, то це можливо за рахунок використання апарату матричної та векторної алгебри [5, 6].

Використання векторів дає змогу визначати і зберігати у пам'яті такі характеристики об'єкта, як: положення, напрямок, швидкість. Для того щоб ігровий об'єкт або персонаж переміщався необхідно використовувати додавання векторів, а під час стрільби зброї – віднімання векторів. Під час побудови стратегії гри із спецефектами, основна суть зводиться до розрахунку відстані між ними і персонажем

та сумуванні збитків, у такому випадку потрібно розрахувати вектор, який находиться між ними, виконувати операції цілочисельного ділення тощо.

Для прикладу розглянемо квадратну матрицю $M[m_{ij}]$ розмірністю 3×3. Математична постановка задачі полягає у перевірці на цілочисельне ділення відповідних елементів матриці $C[c_i]$ на відповідні елементи матриці $R[r_i]$ (табл. 1.1), де $C[c_i]$– матриця добутку елементів початкової матриці $M[m_{ij}]$ по стовпцях; $R[r_i]$ – матриця добутку елементів початкової матриці $M[m_{ij}]$ по рядках.

Таблиця 1.

$M[m_{ij}]$			$R[r_i]$
1	2	3	6
4	5	6	120
7	8	9	504

$C[c_i]$	$R[r_i]$		$C[c_i]//R[r_i]$
28	6	=	28 // 6=4
80	120		80 // 120=0
162	504		162 // 504=0

$C[c_i]$	28	80	162

Програмну реалізацію задачі (табл. 1) в середовищах Python та Javascript, відповідно, представлено на рис. 1.

```python
def ZeroDiv(M, N):
    R = []
    C = []
    for i in range(N):
        r = 1
        c = 1
        for j in range(N):
            r *= M[i][j]
            c *= M[j][i]
        R.append(r)
        C.append(c)
    z = 0
    try:
        for i in range(N):
            if C[i]//R[i] == 0:
                z += 1
    except:
        return -1
    if z >= 2:
        return 1
    else:
        return 0
if __name__ == "__main__":

    M = [[1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]]

    N = len(M)
    x = ZeroDiv(M, N)
    print(x)
```

```javascript
function ZeroDiv(M, N) {
    let R = []
    let C = []
    for (let i = 0; i < N; i++) {
        r = 1
        c = 1
        for (let j = 0; j < N; j++) {
            r *= M[i][j]
            c *= M[j][i]
        }
        R.push(r)
        C.push(c)
    }
    z = 0
    try {
        for (let i = 0; i < N; i++)
            if (Math.floor(C[i] / R[i]) == 0)
                z += 1
    }
    catch {
        return -1
    }
    if (z >= 2)
        return 1
    else
        return 0
}
let M = [[1, 2, 3],
[4, 5, 6],
[7, 8, 9]]
let N = M.length
let x = ZeroDiv(M, N)
document.write(x)
```

Рис. 1. Програмна реалізація алгоритму в середовищі Python та Javascript

Для забезпечення достовірності оперування функціями будь-якої мови програмування необхідно чітко розуміти алгоритм виконання поставленої задачі та ймовірні шляхи його реалізації [3].

Алгоритм виконання задачі

1. Обчислити добуток елементів матриці $M[m_{ij}]$ по рядках та стовпцях;
2. Зберегти добуток кожного рядка та стовпця у масивах $R[r_i]$ та $C[c_i]$ відповідно;
3. Виконати цілочисельне ділення відповідних елементів масиву $C[c_i]$ на $R[r_i]$;
4. Якщо отримаємо два або більше нулі, то повертається число 1, у протилежному випадку число 0;
5. Якщо знаменник дорівнює нулю, то повертається помилка поділу на нуль число -1.

Беручи до уваги описаний алгоритм та результати представлені у табл. 1., матимемо кінцевим значенням 1, де просторова складність алгоритму буде залежати від розмірності вихідної матриці.

Для обробки та аналізу великих масивів з даними їх необхідно розмістити у багатовимірному просторі та скористатися алгебраїчними і геометричними підходами, описати залежності між ними. Це дозволить виявити існуючі взаємозв'язки, закономірності для візуальної інтерпретації конкретної моделі, яку можна описати в режимі реального часу [4]. Отримані результати такого дослідження відкривають нові перспективи до використання моделей з метою прогнозування та прийняття рішень у вирішенні задач прикладного характеру. Це можуть бути дослідження природних потоків різнопланових величин через мережу, де вводяться до розгляду системи лінійних рівнянь [1]. Прикладами таких досліджень є: 1) контроль структури транспорних потоків у мережі міських вулиць; 2) моніторинг мережі водопроводів; 3) струми, які протікають через електричні кола; 4) аналіз розподілу продукції з виробництв до споживачів через мережу оптової і роздрібної торгівлі тощо. У більшості випадків, виникає необхідність до опрацювання сотень і навіть тисяч змінних та рівнянь.

За своєю структурою мережа являє собою множину точок, які називають вузлами (з'єднаннями), які повністю або частково пов'язані між собою за допомогою прямих (дуг), які називають ланками (гілками). Крім цього, задається напрям потоку для кожної ланки та сумарний потік (швидкість) за рахунок обраної змінної. Під час аналізу потоків у мережі загальноприйнятими є такі припущення: 1) Загальний потік у мережі визначаться, як загальний потік на виході з мережі; 2) загальний потік на вході у вузол дорівнює загальному потоку на виході з вузла.

При постановці задач дослідження потокових мереж виникає необхідність у знаходженні потоку у кожній ланці, при заданій частковій інформації (такої як вхід у сітку) [5].

Зосередимо увагу на вирішенні задачі прикладного характеру стосовно мережі транспортних потоків. На рис. 2. представлений транспортний потік (у транспортних засобах за годину) через декілька центральних вулиць міста з одностороннім рухом у першій половині дня. Необхідно визначити структуру загального потоку для мережі та проаналізувати отримані значення змінних для невідомих потоків у гілках $(x_1, x_2, x_3, x_4, x_5)$.

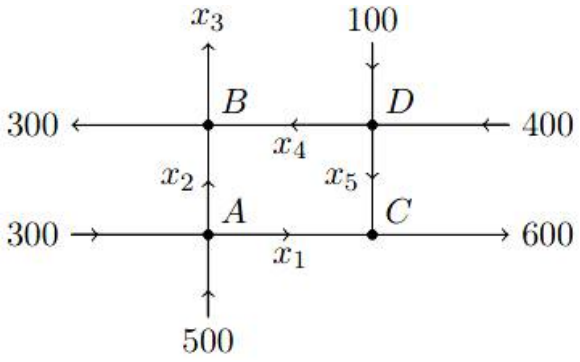


Рис. 2. Транспортний потік вулицями міста

Побудуємо систему лінійних рівнянь, яка описуватиме транспортний потік представлений на рис. 2. та знайдемо її загальний розв'язок. Для цього враховуватимемо, що на кожному перехресті (*A, B, C, D*) «потік у» та «потік з» є рівнозначними величинами, тобто матимемо наступні рівності (1):

$$\begin{aligned} &300+500=x_1+x_2\\ &x_2+x_4=300+x_3\\ &x_1+x_5=600\\ &100+400=x_4+x_5. \end{aligned} \tag{1}$$

Вважається, що загальний потік у сітці та загальний потік виходу з сітки є рівними між собою (500+300+100+400 $=300+x_3+600$), легко бачити, що $x_3=400$. Із врахуванням рівності (1) система лінійних рівнянь матиме вигляд (2).

$$\begin{cases} x_1+x_2=800\\ x_2+x_4-x_3=300\\ x_1+x_5=600\\ x_4+x_5=500\\ x_3=400. \end{cases} \tag{2}$$

Таким чином, загальний потік моделі для сітки можна представити у вигляді (3).

$$\begin{cases} x_1=600-x_5\\ x_2=200+x_5\\ x_3=400\\ x_4=500-x_5\\ x_5-\textit{вільна}. \end{cases} \tag{3}$$

Проаналізуємо отримані результати (3) для вихідної системи лінійних рівнянь (2), із врахуванням вихідної постановки задачі рис. 2. Кожна із представлених змінних – x_1,x_2,x_3,x_4,x_5 не може набувати від'ємного значення, оскільки розглядається випадок транспортного потоку з одностороннім рухом. Очевидно, що це дозволяє встановити

обмеження на допустимі значення змінних, тобто $x_5 \le 500$ і тд. Можуть мати місце і додаткові обмеження, які пов'язані із непередбаченими обставинами (технічним обслуговуванням доріг тощо).

Очевидно, що фахівці сучасного рівня у сфері інформаційних систем та технологій мають досить добре розуміти математичну теорію та орієнтуватися у комп'ютерній реалізації математичних методів до вирішення практичних задач. Для цього існує значна кількість спеціалізованих пакетів програмного забезпечення, таких як Mathcad, Matlab, Mathematica, Maple тощо. Як наслідок, успішне формування вмінь та навичок до використання математичного апарату дозволить програмістам розробляти стабільні утиліти, які задовольнятимуть вимоги користувачів.

Висновки

У статті розглянуто прикладну реалізацію апарату алгебри та геометрії при підготовці фахівців із спеціальності 126 «Інформаційні системи та технології». Запропоновано вдосконалювати існуючі підходи до навчання майбутніх фахівців з аналізу даних, зокрема за рахунок залучення їх до вирішення прикладних задач із можливістю інтерпретувати отримані результати на програмному рівні.

Встановлено, що ключовим фактором успішної підготовки майбутніх програмістів є ґрунтовна математична база, яка дає можливість формувати зрозумілі та логічні рішення під час роботи над різними проектами у подальшій професійній діяльності. В якості підтвердження, вказано основні сучасні напрямки розвитку інформаційних систем та новітніх технологій із використанням математичних дисциплін. Наведено приклад використання апарату матричної та векторної алгебри, розроблено та реалізовано алгоритм елементарної задачі теорії матричних ігор у середовищах Python та Javascript. Розглянуто та проаналізовано використання основних алгебраїчних підходів для побудови моделі мережі транспортних потоків.

Таким чином, володіння апаратом алгебри та геометрії дозволить фахівцю інформаційних систем та технологій генерувати нові ідеї та практично їх реалізовувати, що призведе до вдосконалення існуючих та створення нових напрямків IT- індустрії.

Список використаної літератури:

1. Бахрушин В. Є. Методи аналізу даних : навчальний посібник для студентів / В.Є. Бахрушин. – Запоріжжя : КПУ, 2011. – 268 с.
2. Біла, Н.І. Інформаційні системи та технології в управлінні: методичні вказівки / Н.І. Біла. – Запоріжжя: ЗНТУ, 2014. – 50 с.
3. Івченко, І.Ю. Математичне програмування: Навчальний посібник / І.Ю. Івченко. – К.: Центр учбової літератури, 2007. – 232 с.
4. Tian, Y., Lo, D., Sun, C.: DRONE: predicting priority of reported bugs by multi-factor analysis. In: 2013 IEEE International Conference on Software Maintenance, Sept. 22–28, pp. 200–209. Eindhoven, Netherlands (2021). https://doi.org/10.1109/icsm.2013.31
5. Tian, Y., Lo, D., Xia, X., Sun, C.: Automated prediction of bug report priority using multifactor analysis. Empirical Softw. Eng. 20(5), 1354–1383 (2019). https://doi.org/10.1007/s10664-014-9331-y
6. Yang, X., Du, J., Liu, S., Li, R., Liu, H.: Air pollution source estimation profiling via mobile sensor networks. In: 2020 International Conference on Computer, Information and Telecommunication Systems (CITS), July 6–8. Kunming, China (2020). https://doi.org/10.1109/cits.2016.7546456

References:

1. Bakhrushin, V. (2011). Methods of data analysis: a textbook for students. [in Ukrainian]
2. Bila, N.I. (2014), Informatsiini systemy ta tekhnolohii v upravlinni: metodychni vkazivky [Information Systems and Technologies in Management: Guidelines]. Zaporizhzhia: ZNTU [in Ukrainian].
3. Ivchenko, I. Yu. (2007), Matematichne programuvannya: Navchalnij posibnik [Mathematical Programming: A Tutorial]. Kyiv: Czentr uchbovoyi literaturi [in Ukrainian].

4. Tian, Y., Lo, D., Sun, C. (2021). DRONE: predicting priority of reported bugs by multi-factor analysis. *In: 2013 IEEE International Conference on Software Maintenance*, Sept. 22–28, pp. 200–209. Eindhoven [in Netherlands]. https://doi.org/10.1109/icsm.2013.31
5. Tian, Y., Lo, D., Xia, X., Sun, C. (2019). *Automated prediction of bug report priority using multifactor analysis. Empirical Softw.* Eng. 20(5), 1354–1383. [in Netherlands]. https://doi.org/10.1007/s10664-014-9331-y
6. Yang, X., Du, J., Liu, S., Li, R., Liu, H.(2020). Air pollution source estimation profiling via mobile sensor networks. *In: 2020 International Conference on Computer, Information and Telecommunication Systems (CITS)*, July 6–8. Kunming [in China]. https://doi.org/10.1109/cits.2016.7546456

DZYUBA Viktoriya,
Candidate of Technical Sciences, Lecturer, The Bohdan Khmelnytsky National University of Cherkasy

USING OF ALGEBRA AND GEOMETRY METHODS IN THE PROCESS OF SOLVING APPLIED PROBLEMS DURING THE LEARNING OF INFORMATION SYSTEM AND TECHNOLOGIES SPECIALISTS

Summary. Introduction. *A successful combination of mathematical apparatus, modern engineering and high technology allows us to implement bold ideas and implement state-of-the-art solutions in the life of each of us.*

The article shows that mastering the apparatus of algebra for data analysts is an extremely important aspect for the successful solution of applied problems, in addition, it will allow in the future to realize themselves in a promising career. As a confirmation of the relevance of the chosen direction of research, the leading scientists and their work on the subject of this article are indicated.

The main modern directions of using mathematical disciplines for the development of information systems and technologies are outlined. An algorithm for performing an elementary task of matrix game theory and its software implementation in Python and Javascript environments has been developed.

The solution of the problem of applied nature, using algebraic approaches, to build a model of the network of transport flows is considered and the obtained results are analyzed.

The purpose *of the article is the implementation of the apparatus of algebra and geometry in the process of solving applied problems in the training of specialists in the field of information systems and technologies.*

Results. *The use of basic algebraic approaches to build a model of a network of transport flows is considered and analyzed. It is established that the key factor of successful training of future programmers is a solid mathematical base, which allows to form clear and logical solutions while working on various projects in further professional activities. As a confirmation, the main modern directions of development of information systems and the newest technologies with use of mathematical disciplines are specified.*

An example of using the apparatus of matrix and vector algebra is given, the algorithm of the elementary problem of matrix game theory in Python and Javascript environments is developed and implemented.

Conclusion. *The article considers the applied implementation of the apparatus of algebra and geometry in the training of specialists in the specialty 126 "Information Systems and Technologies". It is proposed to improve the existing approaches to the training of future specialists in data analysis, in particular by involving them in solving applied problems with the ability to interpret the results at the program level.*

Thus, mastering the mathematical apparatus will allow the future programmer to generate new ideas and implement them in practice, which will lead to the improvement of existing and creation of new directions in the IT industry.

Keywords: *algebra and geometry, mathematical model, data analysis, information systems, latest technologies.*

Одержано редакцією 24.09.2021 р.
Прийнято до публікації 24.11.2021 р.

УДК 004.032.26

DOI 10.31651/2076-5886-2021-1-91-112

PACS 07.05.Mh

СЕРДЮК Олександр Анатолійович,
кандидат економічних наук, доцент,
доцент кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: serdyuk@ukr.net
ORCID 0000-0002-3919-4661

ПІСКУН Олександр Варфоломійович,
кандидат технічних наук, доцент,
завідувач кафедри прикладної математики та інформатики Черкаського національного університету імені Богдана Хмельницького
e-mail: piskun@ukr.net
ORCID 0000-0001-5334-6337

ДІХТЯРЕНКО Володимир Анатолійович,
Драбівський НВК «Загальноосвітня школа І-ІІІ ст. ім. С.В. Васильченка-гімназія» Драбівської районної ради

БІЛЕЦЬКА Надія Андріївна,
вчитель математики вищої кваліфікаційної категорії, педагогічне звання «вчитель-методист» Драбівського НВК «загальноосвітня школа І-ІІІ ст. ім. С.В. Васильченка-гімназія» Драбівської районної ради

ПОПЕРЕДНЄ ВИЗНАЧЕННЯ КІЛЬКОСТІ КЛАСТЕРІВ ЗА ДОПОМОГОЮ ШТУЧНИХ НЕЙРОННИХ МЕРЕЖ ТИПУ SOFM

У статті описано авторський алгоритм апріорного визначення кількості кластерів за допомогою самоорганізованих карт ознак Кохонена. У першій частині статті наведено поняття кластеризації та описано використовуваний алгоритм кластеризації k-середніх. У другій частині наведено теоретичні відомості та алгоритм роботи штучної нейронної мережі Кохонена; на прикладах продемонстровано роботу нейронної мережі та проаналізовано отримані результати. У третій частині наведено розроблений алгоритм початкового автоматичного визначення кластерів у системі для алгоритму k-means, отриманого за допомогою SOFM; проаналізовано отримані результати та сформовано подальші напрямки дослідження обраної теми.

Ключові слова: *машинне навчання, кластеризація, k-means, штучні нейронні мережі, SOFM.*

Вступ

Протягом останніх кількох років спостерігається стрімкий розвиток напрямку штучного інтелекту, що називається “машинне навчання”. У основі цього напрямку

лежать методи автономної роботи обчислювальних систем, які дозволяють розв'язувати різноманітні задачі: комп'ютерного бачення, обробки текстів, прийняття рішень тощо. У цьому сенсі важливою підзадачею є скорочення обсягу оброблюваних даних, яких на поточний момент можуть бути тисячі гігабайтів. Для розв'язування таких підзадач, тобто, для скорочення кількості інформативних даних, часто використовуються методи групування даних для визначення їх спільних характеристик: методи кластеризації.

Задача кластеризації полягає у поділі досліджуваної множини об'єктів на групи "схожих" об'єктів, які називаються кластерами. Відповідно, метод розв'язування задачі розбиття множини елементів на кластери називають кластерним аналізом.

Перші публікації з кластерного аналізу з'явились у кінці 30-х років минулого століття, але активний розвиток цих методів і їх широке використання почалося у кінці 60-х – на початку 70-х років. Надалі цей напрямок багатовимірного аналізу інтенсивно розвивався: з'явилися нові методи, модифікації вже відомих алгоритмів, істотно розширилася область застосування кластерного аналізу. Якщо спочатку методи багатовимірної класифікації використовувалися у психології, археології, біології, то зараз вони стали активно застосовуватися у соціології, економіці, статистиці, історичних дослідженнях. Особливо розширилося їх використання у зв'язку з появою і розвитком ЕОМ і, зокрема, персональних комп'ютерів. Це пов'язано, перш за все, з трудомісткістю обробки великих масивів інформації (обробка матриць великих розмірів).

Для наукових досліджень вивчення результатів кластеризації, а саме – з'ясування причин, за якими об'єкти об'єднуються в групи – здатне відкрити нові перспективні напрямки. Традиційним прикладом, який зазвичай наводять для цього випадку, є періодична таблиця елементів. У 1869 р. Дмитро Менделєєв розділив 60 відомих на той час елементів на кластери або періоди. Елементи, що потрапили у одну групу, мали схожі характеристики. Вивчення причин, за якими елементи розбивалися на явно виражені кластери, значною мірою визначило пріоритети наукових досліджень на роки вперед. Але лише через 50 років квантова фізика дала переконливі пояснення періодичної системи.

Мета кластерного аналізу полягає у пошуку структур, які існують у даних, що виражається в утворенні груп схожих між собою об'єктів (це і є кластерами). Водночас дія кластерного аналізу полягає й у структуризації досліджуваних об'єктів. Це означає, що методи кластеризації необхідні для виявлення структури у даних, яку нелегко знайти при візуальному обстеженні або за допомогою експертів.

Однак, методи кластерного аналізу потребують деякої апріорної (початкової) інформації, зокрема, перед проведенням кластерного аналізу для більшості методів вже потрібно знати кількість кластерів, на які необхідно розбити множину об'єктів. Таку інформацію не завжди можливо отримати внаслідок, наприклад, великої розмірності даних, тому дослідники у багатьох випадках обирають кількість кластерів інтуїтивно, що, зрозуміло, не завжди приводить до значущих результатів. Тому перед проведенням кластеризації видається можливим використати які-небудь інші методи, що дозволили б оцінити (принаймні, приблизно) кількість можливих кластерів.

Однією з груп таких методів є використання штучних нейронних мереж [3, 4, 5]. Серед багатьох видів штучних нейронних мереж є такі, що застосовують при автоматичній класифікації множини об'єктів, тобто, якраз їх розподілі по групах, причому, кількість таких груп заранні невідома.

Метою дослідження є визначення можливості використання конкретного виду штучних нейронних мереж – самоорганізованої карти ознак – для попереднього

визначення можливої кількості кластерів для конкретного методу кластеризації – методу k-means.

Завданнями, висунутими у дослідженні, є:

1) розгляд алгоритму кластеризації k-means та його реалізація;
2) розгляд самоорганізованої карти ознак та її реалізація;
3) дослідження можливостей використання самоорганізованої карти ознак для отримання апріорної інформації стосовно кількості кластерів для алгоритму k-means та вироблення рекомендацій по застосуванню самоорганізованих карт ознак при кластеризації даних.

Виклад основного матеріалу

1. Алгоритм кластеризації k-середніх

Слово cluster (“кластер”) перекладається з англійської як згусток, пучок, група. Споріднені поняття, що використовуються в літературі, – клас, таксон, згущення.

Задача кластеризації (англ. clustering) на відміну від інших відомих класів задач машинного навчання передбачає так зване “навчання без учителя” (англ. “unsupervised learning”): її суть у тому, щоб надати досліднику інструмент для автоматичного поділу наявних об’єктів на класи на підставі подібності тих чи інших характеристик (факторів) цих об’єктів. При цьому ні самі класи, ні їх кількість заздалегідь не відомі. Виникає питання: як комп’ютер виконає розбиття на класи, якщо самі ці класи не відомі? Кластери будуються таким чином, щоб характеристики об’єктів одного класу були дуже близькі і при цьому сильно відрізнялися від характеристик об’єктів інших кластерів. Очевидно, що розуміння “подібності” і “відмінності” сильно залежить від предметної області і навіть від конкретної задачі.

Кластеризація може забезпечити краще розуміння даних і спростити їх подальшу обробку. Наприклад, сегментування (поділ на групи) ринку за будь-якими ознаками споживчої поведінки населення дозволяє проводити адресні акції і різні маркетингові заходи, спрямовані на збільшення обсягу продажів.

Іншим прикладом можна навести задачу класифікації технічних рухомих засобів за деяким критерієм, наприклад – кількістю коліс. Всі об’єкти, подібні мотоциклу, потраплять при цьому в одну групу, а всі об’єкти, подібні автомобілю – у іншу. Ці групи потім аналізуються, і від групи подібних мотоциклу засобів відділяється група мопедів чи скутерів як засобів, що мають двигун внутрішнього згорання меншого об’єму. Група мопедів подібна групі мотоциклів, тому ці групи повинні розміститися близько одна від одної і далеко від групи технічних засобів, подібних автомобілю. Алгоритми кластеризації виконують такі операції зі зразками даних.

Значна перевага кластерного аналізу у тому, що він дозволяє здійснювати розбиття множини об’єктів не за одним параметром, а за цілим набором ознак. Наприклад, у випадку автомобілів та мотоциклів однієї ознаки “кількість коліс” – недостатньо, оскільки існують квадроцикли з чотирма колесами, як у автомобіля, але об’ємом двигуна, як у мотоцикла, і тому необхідна ще одна ознака, за якою проводити класифікацію. Крім того, кластерний аналіз, на відміну від більшості математико-статистичних методів, не накладає ніяких обмежень на вид розглядуваних об’єктів і дозволяє розглядати множину вихідних даних практично довільної природи. Кластерний аналіз дозволяє розглядати досить великий обсяг інформації і різко скорочувати, стискати великі масиви інформації, робити їх компактними і наочними за рахунок того, що усередині кластера об’єкти не розрізняються, тобто, розглядаються, як один об’єкт.

Загалом поділ множини об’єктів на кластери повинен відповідати наступним двом вимогам:

– об’єкти усередині одного кластера повинні бути у певному сенсі подібні (наприклад, велосипеди подібні один до одного, так само мотоцикли, автомобілі, тощо);
– кластери, подібні у певному сенсі, повинні розміщуватися близько один від іншого (наприклад, кластер велосипедів буде розміщений ближче до кластера мотоциклів, аніж кластера автомобілів).

На рис. 1 показано розташування на площині даних, які природним чином організовуються у три кластери: точка, що відповідає зразку, потрапляє у певний кластер, якщо вона розміщена близько до точок цього кластера у порівнянні з точками, які належать іншим кластерам. Мірою близькості (або подібності) двох точок зазвичай є квадрат Евклідової відстані між ними, який обчислюється за формулою

$$d = \sum_{i=1}^{n} \left(x_{pi} - x_{qi}\right)^2 ,$$

де d – квадрат Евклідової відстані між точкою p та точкою q;

x_{pi} – i-а координата точки (зразка) p;

x_{qi} – i-а координата точки (зразка) q;

n – значення розмірності.

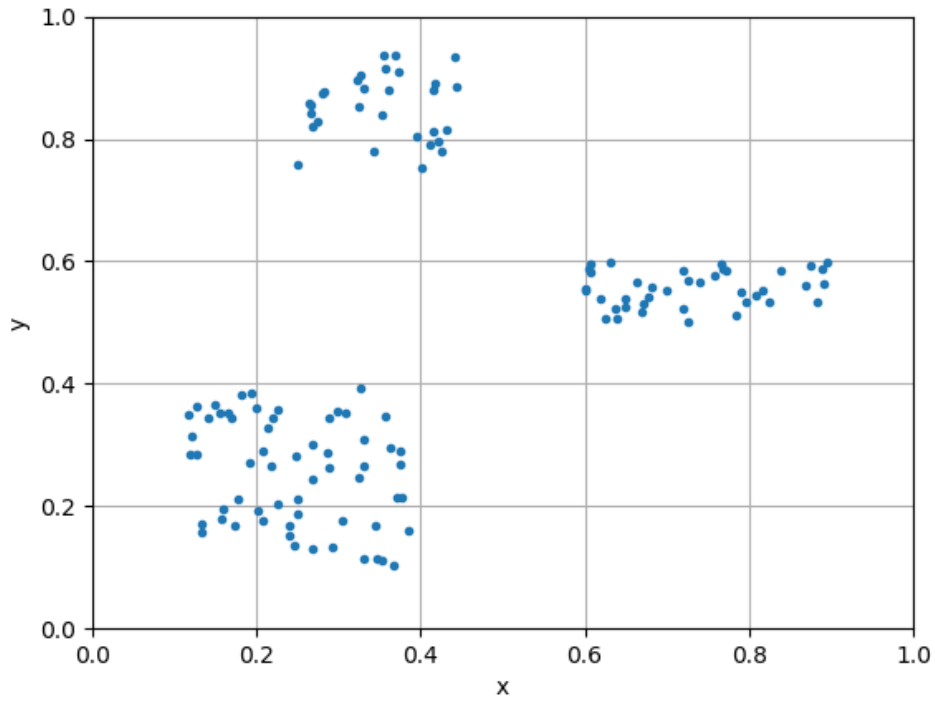


Рис. 1. Дані, що формують три кластери

Якщо для кластера j розглянути вектор $\vec{p}_j$, який визначається центроїдом (точкою, що відповідає усередненій характеристиці розташування усіх зразків у кластері), то для даних на рис. 1 рішення про те, якому з кластерів належить довільний вектор $\vec{x}$ (що зображується на площині у вигляді точки), визначається на основі результату обрахунків:

$$idx(\vec{x}) = min\, d(\vec{p}_j, \vec{x}) \text{ для усіх } j ,$$

де повертається індекс кластера з найменшим квадратом Евклідової відстані до вектора $\vec{x}$. Вектори $\vec{p}_j$ можуть розглядатися як прототипи кластерів, і ці прототипи можуть

служити для представлення ключових ознак кластера. Наприклад, якщо необхідно розділити на групи автомобілі і мотоцикли, цілком зрозуміло, що у якості ознаки можна вибрати кількість коліс. Тому значення елементів, що означають таку кількість у векторах-прототипах двох кластерів, повинні істотно відрізнятися.

Алгоритм кластеризації є статистичною процедурою виділення груп з наявного набору даних. Існує чимало алгоритмів кластеризації найрізноманітніших рівнів складності. Один з найпростіших підходів полягає в тому, щоб припустити існування певної кількості кластерів і довільним чином вибрати координати для кожного з прототипів. Потім кожен вектор з набору даних зв'язується з найближчим до нього прототипом, і новими прототипами стають центроїди усіх векторів, пов'язаних з вихідним зразком. На рис. 2 показаний випадковий вибір прототипів, які до кінця навчання повинні переміститися у центри кластерів, як показано на рис. 3.

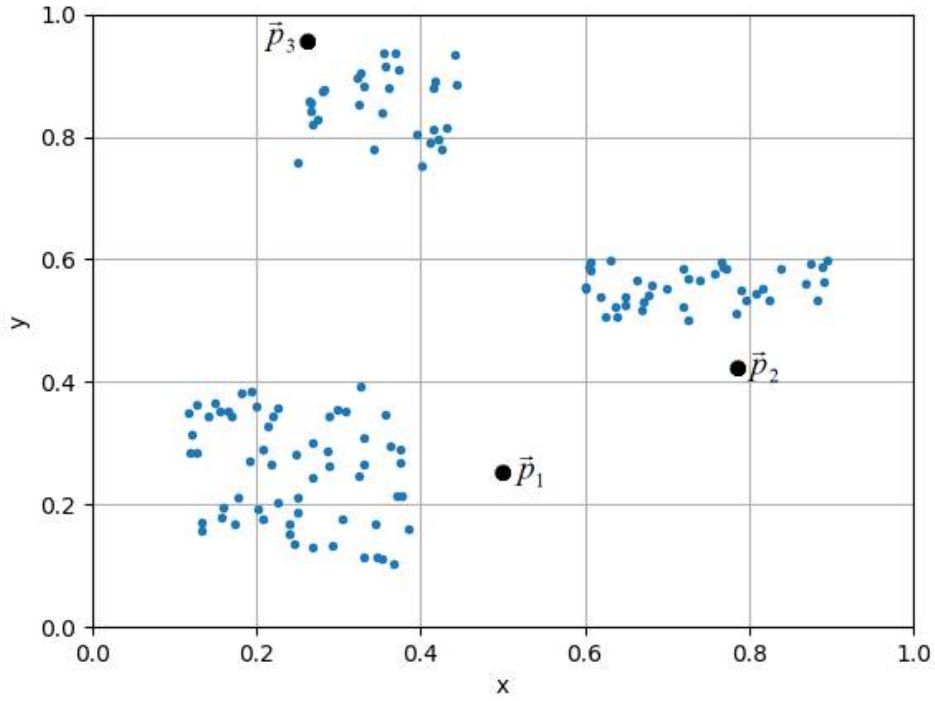


Рис. 2. Три випадкові вектори, що будуть зміщуватись, виступаючи у ролі прототипів для кластерів

Вище наведено ідею одного з найпростіших алгоритмів кластеризації – алгоритму k-means (k-середніх).

У основі алгоритму k-means лежить ітеративний процес стабілізації центроїдів кластерів. Основною характеристикою кластера є його центроїд і уся робота алгоритму спрямована на стабілізацію або – у кращому разі – повне припинення зміни центроїда кластера. Алгоритм k-means будує k кластерів, розміщених на якомога більших відстанях один від одного. Повний опис алгоритму можна знайти у роботі Дж. Хартігана [1]. Нижче подано псевдокод алгоритму.

Алгоритм 1 (алгоритм k-means)
Вхід: множина даних D, кількість кластерів k
Вихід: множина кластерів C, кожний з яких подається точкою центра кластера c_i, та масив *clust*, що ставить у відповідність кожну точку номеру кластера, до якого вона належить

(1) випадковим чином вибрати k точок з множини D
(2) використати вибрані точки у вигляді центрів кластерів C
(3) поки не досягнуто критерія зупинки
  (3.1) перепризначити точки з множини D до найближчого кластера
  (3.2) оновити *clust* таким чином, щоб кожна точка належала кластеру, центр якого знаходиться найближче до неї
  (3.3) оновити C, перерахувавши центри кластерів

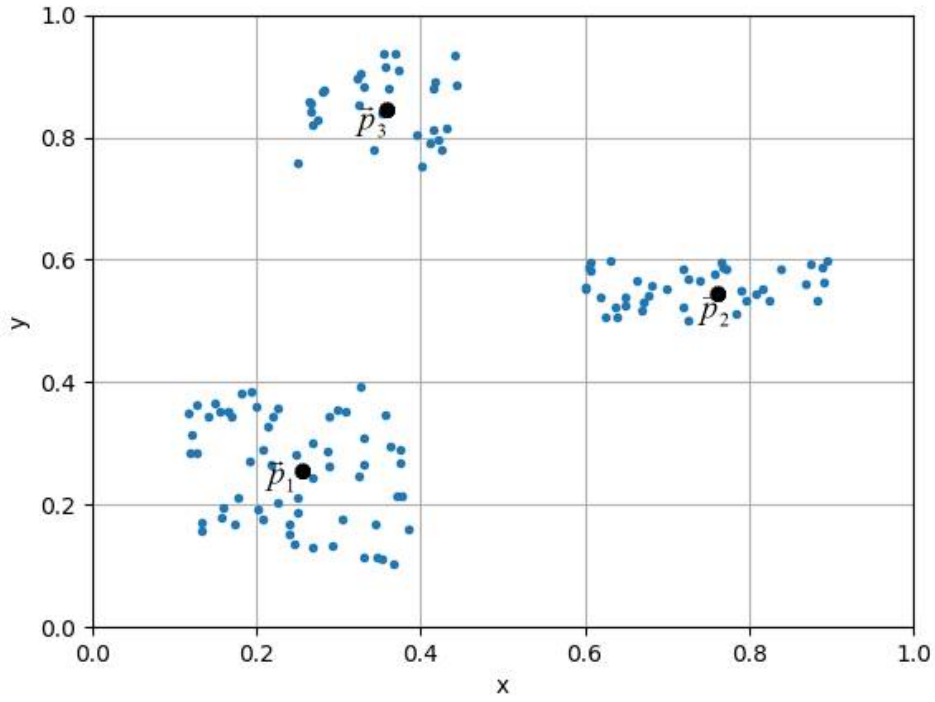


Рис. 3. Кожний з прототипів змістився у точку, що відповідає центроїду кластера

Кластеризація методом k-means починається з вибору k випадково розташованих центроїдів (зразків, що представляють центр кластера). Кожному елементу призначається найближчий центроїд. Після того, як призначення виконано, кожен центр ваги переміщується у точку, яка розраховується, як середнє по усім приписаним до нього елементів. Потім призначення виконується знову. Ця процедура повторюється до тих пір, поки не буде досягнута умова зупинки.

Дія алгоритму така, що він прагне мінімізувати середньоквадратичне відхилення на точках кожного кластера:

$$V = \sum_{i=1}^{k} \sum_{x_j \in S_i} \left(x_j - \mu_i\right)^2 ,$$

де k – кількість кластерів;
S_i – отримані кластери, $i = 1,2,...,k$;
μ_i – центри мас векторів $x_j \in S_i$.

Під час роботи алгоритму на кожній ітерації переобчислюється центр мас для кожного кластера, отриманого на попередньому кроці, потім вектори розбиваються на кластери знову відповідно до того, який з нових центрів виявився ближче до вектора

згідно з обраною метрикою. Алгоритм завершується, коли на якійсь ітерації не відбувається зміни кластерів.

Основний тип задач, які розв'язує алгоритм k-means, – наявність припущень (гіпотез) відносно існування певної кількості кластерів у наборі даних, при цьому вони мають бути різні настільки, наскільки це можливо. Вибір числа k може базуватися на результатах попередніх досліджень, теоретичних міркуваннях або інтуїції.

Умовою зупинки алгоритму може бути одна з наступних:

1) центроїди кластерів більше не змінюються;
2) досягнуто порогового значення помилки;
3) досягнуто певної максимальної (порогової) кількості ітерацій.

Виконання першого критерію є найбажанішим, оскільки його досягнення означає повну стабілізацію структури кластерів. Другий критерій означає, що на черговій ітерації кожний з центроїдів від попереднього положення до наступного змістився не більше ніж на якесь наперед задане значення ε, що називається помилкою. Третій критерій виконується, якщо отримана нестійка структура, що постійно коливається – наприклад, у випадку, коли одна точка по черзі відноситься то до одного, то до іншого кластера.

Перевагами алгоритму k-means є наступні:

- алгоритм простий у використанні;
- висока швидкість роботи алгоритму;
- зрозумілість і прозорість алгоритму.

Однак, алгоритм має й суттєві недоліки:

- висока чутливість до викидів, тобто, окремо розташованих точок;
- повільно працює на великих базах даних;
- не справляється з задачею, коли об'єкт належить до різних кластерів однаково чи не належить жодному з кластерів.

Центроїди можна розглядати як підсумок для досліджуваного набору даних (прототип). Іноді буває зручно представляти кластер кількома прототипами, щоб отримати більш детальну характеристику даних. Щоб розпізнати кластери, які є частинами більшого кластера, необхідно знати положення усіх прототипів один відносно іншого. Однією з проблем застосування алгоритмів кластеризації є вибір оптимальної кількості кластерів. Якщо таку кількість вибрати занадто малою, то можуть бути втрачені деякі важливі характеристики даних, а якщо кластерів виявиться занадто багато, то ми не отримаємо ніякої ефективної підсумкової інформації про дані (може навіть статися, що кожен зразок утворить свій кластер).

Реалізація алгоритму

Алгоритм k-means під час роботи було реалізовано мовою Python.

У програмі реалізовано наступні функції.

Функція `makeInitSet(P)` призначена для створення початкової множини точок, що складається з кількох наборів (кластерів). На вхід функція приймає список `P`, кожен елемент якого є набором даних для формування чергового кластера. Структура одного елементу списку `P` є наступною:

```
(n_i (m_i1, m_i2, ...) (s_i1, s_i2, ...)),
```

де `n_i` – кількість точок;

`s_i1` – розтяг точок вздовж осі абсцис;

`s_i2` – розтяг точок вздовж осі ординат;

`m_i1` – зміщення точок вздовж осі абсцис;

`m_i2` – зміщення точок вздовж осі ординат.

Ідея формування множини точок є наступною. Береться випадкове число x в інтервалі від 0 до 1, на яким потім виконується наступне перетворення:

$$(x \cdot s) + m .$$

Шляхом множення x переноситься з відрізка $[0,1]$ на відрізок $[0,s]$, а за допомогою додавання x переноситься з відрізка $[0,s]$ на відрізок $[m,s+m]$.

Згенерувавши для кожної координати точки різні випадкові числа та задавши різні параметри s та m можна отримати розміщені у різних місцях декартової площини та витягнуті вздовж однієї з осей хмари точок. Повертає функція згенеровану множину точок у вигляді прямокутного масиву координат, де точки розміщені по рядкам.

Наприклад, для набору точок, поданого на рис. 4, було задано наступні параметри:

– для хмари “1”: (50,(0.25,0.65),(0.3,0.4)), тобто, 50 точок, зміщення по x на 0.25, зміщення по y на 0.65, розтяг по x на 0.3, розтяг по y на 0.4;
– для хмари “2”: (50,(0.6,0.5),(0.3,0.3));
– для хмари “3”: (50,(0.1,0.1),(0.5,0.3)).

Функція `makeClusters(X,k)` розбиває множину X на k кластерів, беручи в якості центрів кластерів k випадкових точок. Повертає функція масив центрів кластерів `C` та список (лінійний масив) номерів кластерів, до яких відносяться відповідні точки, `CL`. Для безпосередньо самого розбиття функція використовує функцію `classifyPoints(X,C)`.

Функція `classifyPoints(X,C)` розбиває множину X на кластери, призначаючи кожній точці номер кластера, що відповідає номеру точки з множини `C` (центру кластера), до якої найближче знаходиться дана точка. Повертає функція список (лінійний масив) номерів кластерів, до яких відносяться відповідні точки, `CL`.

Нарешті, функція `kMeans(X,CL,C)` виконує алгоритм k-means для множини точок `X`, заданої початкової множини центрів `C` та початкового розбиття точок на кластери `CL`. Виконується функція або ж кількість ітерацій, що задана в якості четвертого параметра, або ж, якщо параметр не заданий,– поки чергове зміщення центрів кластерів не стане меншим за помилку, задане у функції значення якої 0.001.

Реалізована функція у відповідності з алгоритмом 1, поданим вище.

Роботу реалізованого алгоритму можна прослідкувати на рис. 4-7.

На рис. 4 подано початковий набір точок, згенерованих, як було вказано вище.

На рис. 5 подано початкову конфігурацію центрів кластерів. Для цього було обрано три випадкові точки з поданого набору, які й послужили центрами кластерів. На рисунках центри кластерів виділено маркерами більшого розміру порівняно з і звичайними точками. Як можна бачити, початкові центри кластерів належать вхідному набору 1. Зазначимо, що при різних запусках центри кластерів та саме розбиття точок по кластерам будуть різними, а тому позначення, використані на рисунках – трикутники, круги, хрестики – є умовними, так само, як і кольори.

Після виконання однієї ітерації центри кластерів починають зміщуватись у напрямку реальних кластерів, що наявні в наборі: центр синіх трикутників починає зміщуватись у напрямку множини 3, центр зелених хрестиків – у напрямку множини 2, центр чорних точок – переміщується по множині 1.

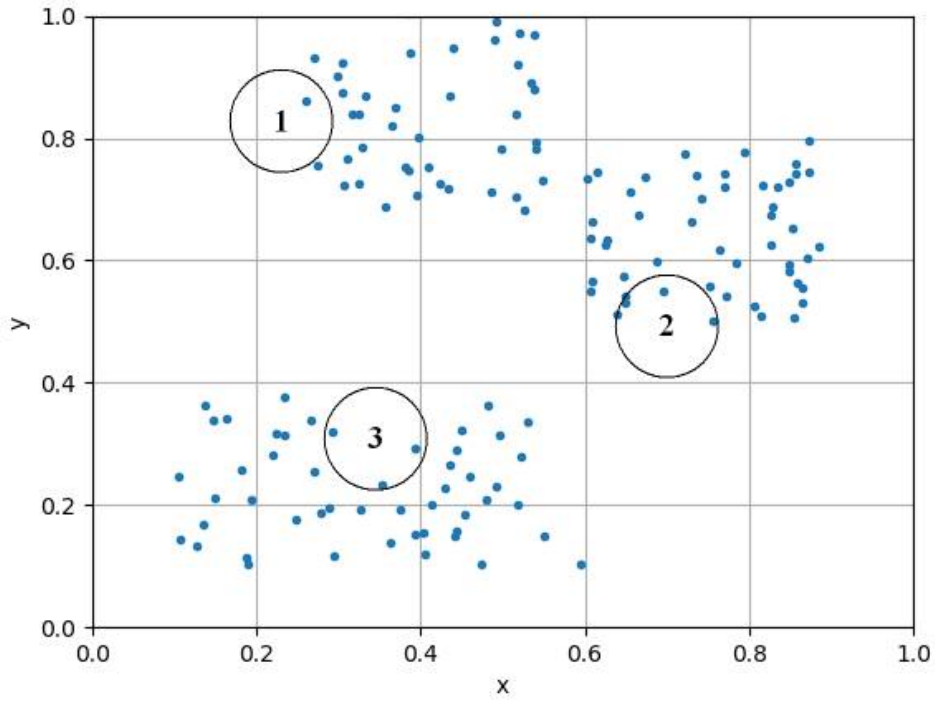


Рис. 4. Початковий набір точок. Точки розміщено так, щоб було чітко видно три кластери (1, 2, 3)

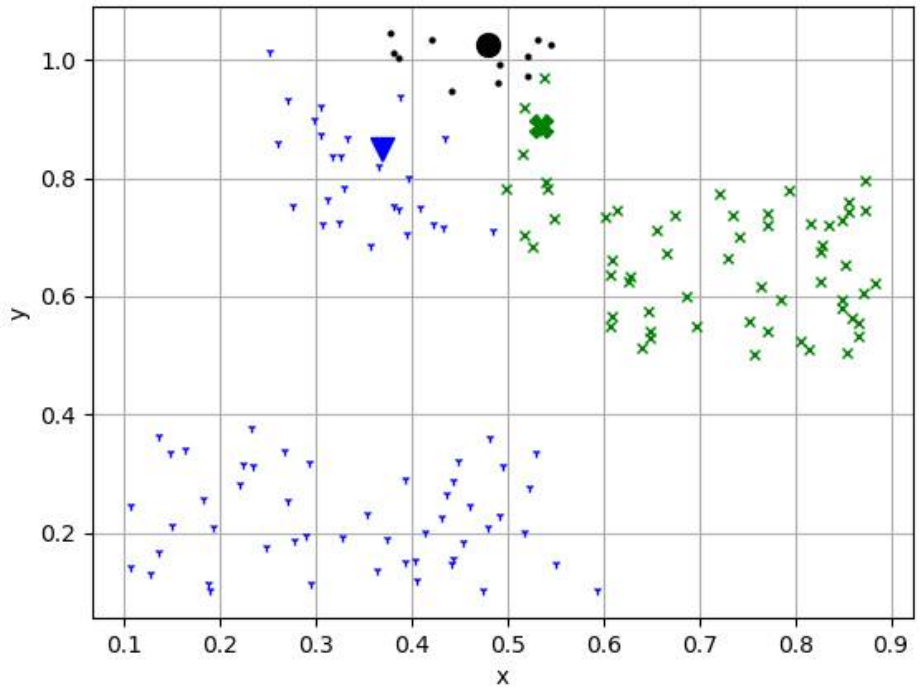


Рис. 5. Випадковим чином вибрані центри кластерів та проведено розподіл точок по кластерам

Нарешті, на рис. 7 подано остаточну – стабільну – конфігурацію, отриману в результаті виконання алгоритму k-means. Як видно, у даному випадку було знайдено правильні кластери, що тепер можна ідентифікувати за їх центрами.

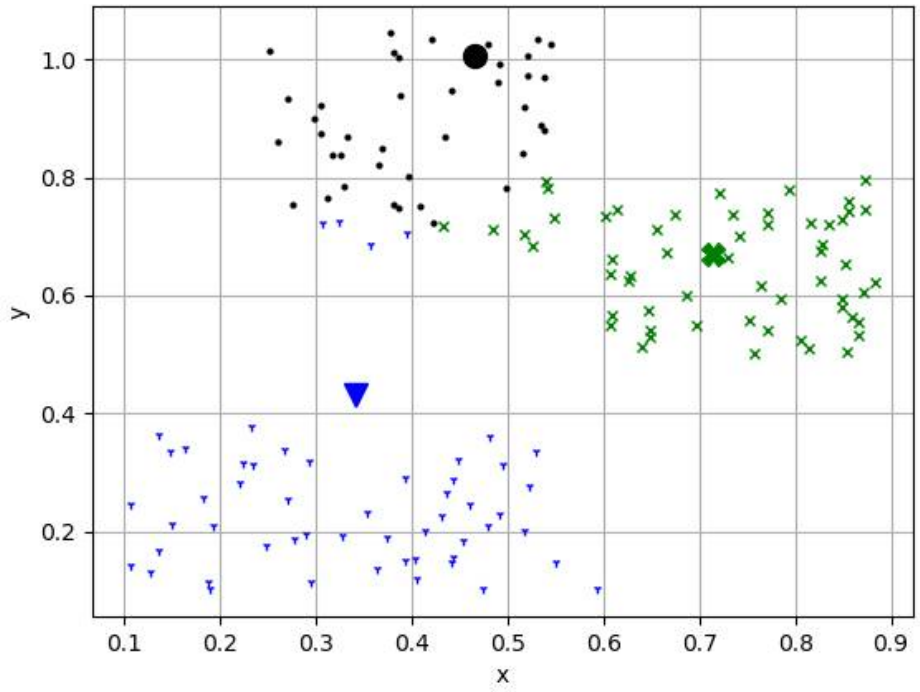


Рис. 6. Виконана одна ітерація алгоритму k-means. Центроїди починають наближатись до центрів кластерів

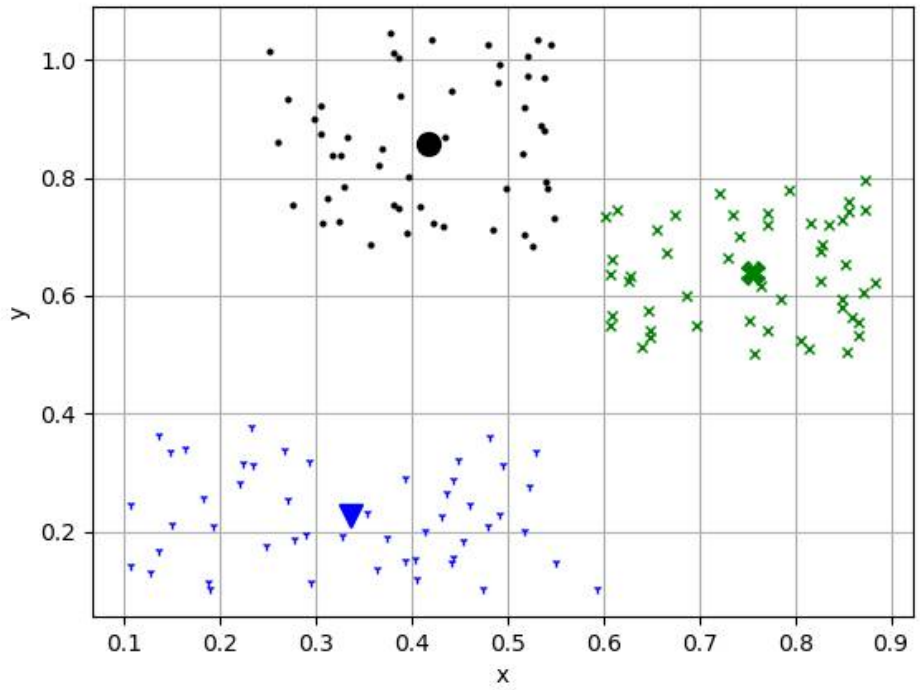


Рис. 7. Закінчено роботу алгоритму k-means. Центри кластерів стабілізувались

Дослідження роботи описаного й реалізованого алгоритму дозволило виявити наступні недоліки.

1. Іноді алгоритм може не знаходити правильні кластери навіть при чітко вираженій структурі набору точок, що залежить від початкового вибору центрів кластерів: при невдалому виборі кластери алгоритм може зупинитись при стабілізації центрів і неправильному визначенні кластерів. Одна з таких конфігурацій зображена на

рис. 8. Як видно з рисунка, виділено неправильні кластери, коли до одного попали множини 1 та 2 з рис. 4 (параметри множин однакові, хоча при різних запусках програми генеруються різні координати точок), а два інші кластери ділять між собою множину 3. Хоча така помилка виникає рідко, вона, все ж, існує. Однак, для того, щоб уникнути цієї помилки, можна виконувати процедуру кластеризації кілька разів, вибираючи, в результаті, кластери, що є однаковими після кількох різних запусків програми.

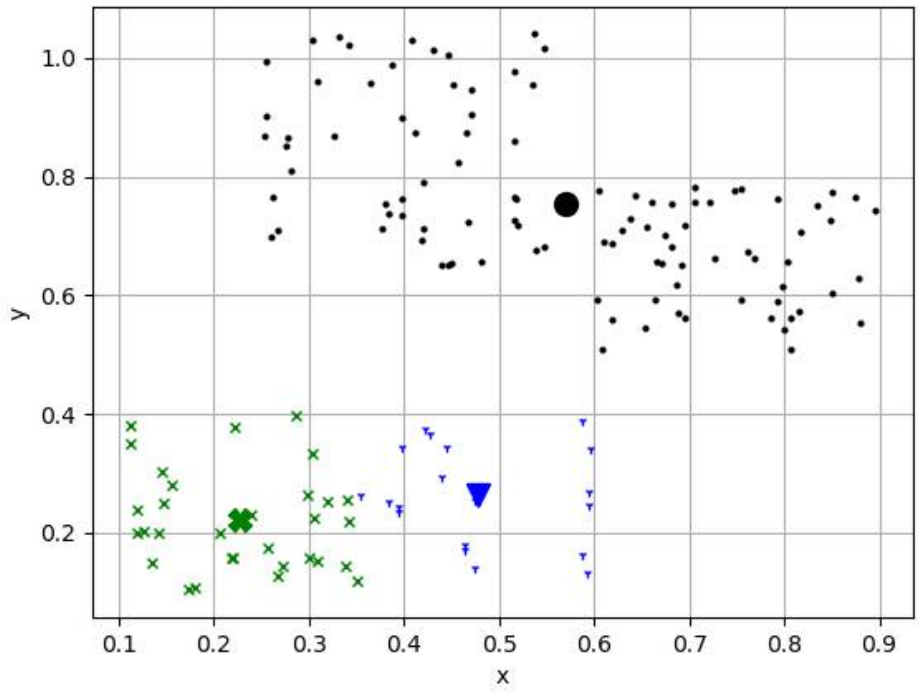


Рис. 8. Стабілізація кластерів при неправильному їх визначенні

2. При вказанні неправильної кількості кластерів алгоритм генерує неправильний результат. Наприклад, на рис. 9 подано результат виконання програми при використанні двох множин з різними параметрами та чотирьох кластерів.

Для того, щоб позбавитись недоліку 2, ми пропонуємо використати один зі способів попередньої оцінки можливої кількості кластерів, використавши результати роботи нейронної мережі з навчанням без учителя, що являє собою самоорганізовану карту ознак – мережу Кохонена.

Отже, у даному розділі розглянуто алгоритм пошуку кластерів k-means, описано результати роботи реалізації алгоритму мовою Python, та після аналізу результатів роботи програми визначено недоліки алгоритму.

2. Самоорганізована карта ознак Кохонена

Самоорганізована карта ознак (мережа SOFM – Self-Organizing Feature Map) [2, 3, 5] має набір вхідних елементів, кількість яких відповідає розмірності навчальних точок, і набір вихідних елементів, які виступають у якості прототипів. Базова архітектура мережі SOFM показана на рис. 10. Наприклад, для даних, показаних на рис. 1, буде необхідною мережа з двома вхідними і принаймні трьома вихідними елементами, що представляють три кластери.

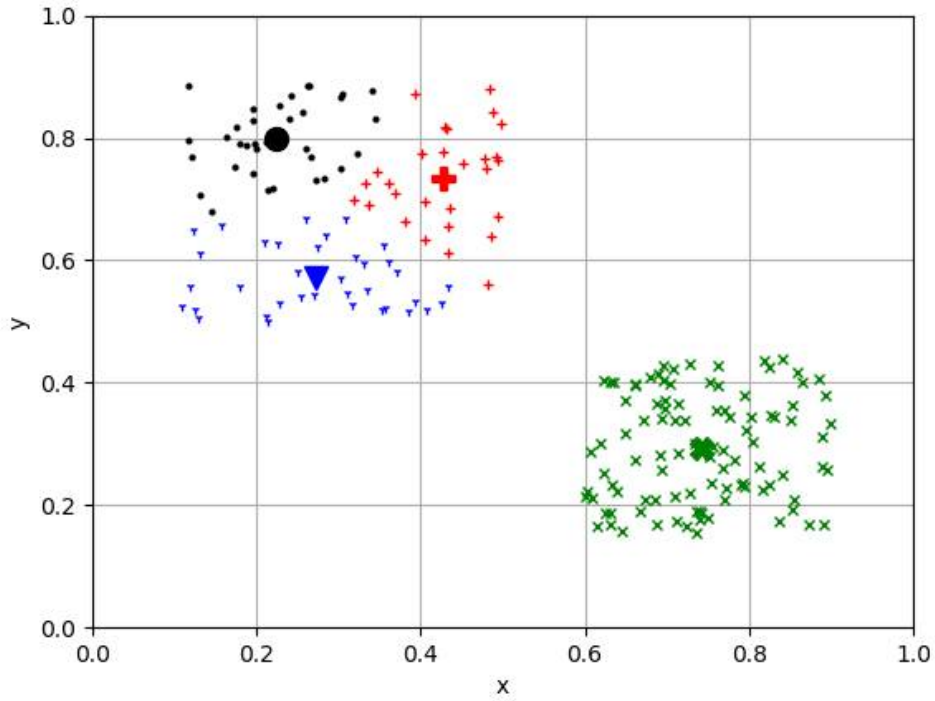


Рис. 9. Утворено 4 кластери при явних двох кластерах. Така конфігурація є результатом неправильно заданої кількості шуканих кластерів для алгоритму k-means

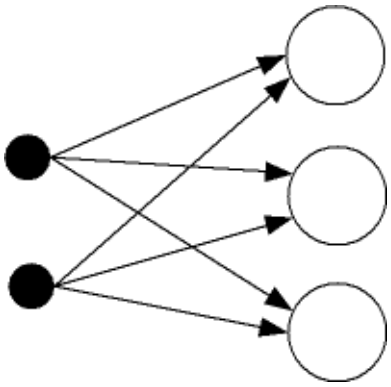


Рис. 10. Мережа має два вхідних і три кластерних елементи, причому кожний елемент вхідного шару пов'язаний з кожним елементом кластерного шару

Вхідні елементи призначені лише для того, щоб розподіляти дані вхідної точки між вихідними елементами мережі. Вихідні елементи називаються кластерними елементами.

Кожний зі зв'язків від вхідного елемента до вихідного має свою вагу, що виражається певним числом. Такі ваги позначаються w_{ij}, де i – номер вихідного елемента мережі, а j – номер вхідного елемента, або координата вхідної точки. Оскільки кількість вхідних елементів відповідає розмірності вхідних точок, а кожен вхідний елемент пов'язаний з усіма кластерними елементами, загальна кількість вагових значень, що впливають на кластерний елемент, теж виявляється рівною розмірності вхідних точок. Часто зручно інтерпретувати вагові значення кластерного елемента як значення координат, що описують позицію кластера у просторі вхідних

даних. На рис. 11 показано, як пов'язуються вагові значення з простором вхідних даних.

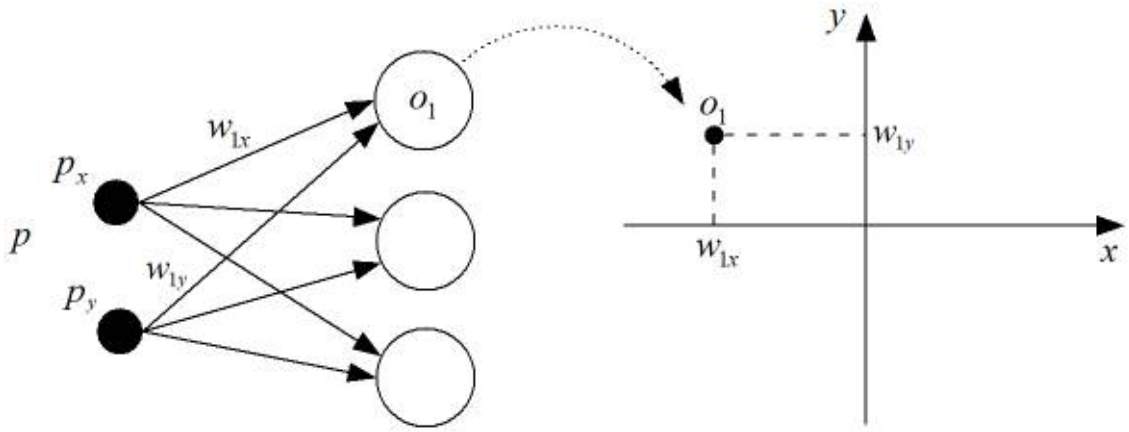


Рис. 11. На схемі показано, як в якості центроїда може виступати кластерний елемент. Вхідний шар використовує значення координат p_x, p_y точки p вхідного простору. У процесі навчання вагові значення коригуються й по закінченні навчання усі кластерні елементи займуть у просторі вхідних даних положення, що відповідає отриманим для них ваговим значенням

Кластерні елементи розміщуються у вигляді одно- або двовимірного масиву, як показано на рис. 12. Під час навчання усі елементи можуть розглядатися як претенденти на нагороду у вигляді навчальних точок. Коли на конкурс виставляється будь-яка навчальна точка, обчислюються відстані від неї до усіх кластерних елементів, і елемент, який знаходиться до даної навчальної точки найближче, оголошується елементом-переможцем.

Для елемента-переможця проводиться коригування вагових значень так, щоб цей кластерний елемент розмістився до навчальної точки ще ближче. Зазвичай коригування вагових значень виконується і для елементів, що знаходяться поруч з елементом-переможцем. Вагові значення елемента оновлюються, якщо він знаходиться усередині кола заданого радіуса з центром у елементі-переможці.

Під час навчання радіус зазвичай поступово зменшується. Норма навчання обмежує величину, на яку кластерний елемент може пересунутися у напрямку до навчального вектора, і, подібно до радіуса, норма навчання теж з часом поступово зменшується. У прикладах, поданих у роботі, розглядається розміщення кластерних елементів у вигляді квадратної сітки, але можуть використовуватися й інші форми, наприклад: лінійне розміщення елементів чи шестикутна сітка. Від топології залежить лише те, які елементи повинні оновлюватися для даного конкретного радіуса.

Зазвичай кількість кластерних елементів вибирають меншою, аніж кількість навчальних зразків, оскільки метою є отримання спрощеної характеристики даних. До кінця навчання кластерні елементи забезпечують «інформаційний звіт» по простору вхідних зразків. Кластерні елементи виступають у ролі карти ознак простору вхідних даних. Наприклад, у результаті кластеризації зображень символів різні області кластерних елементів будуть характеризувати різні символи чи їх комбінації.

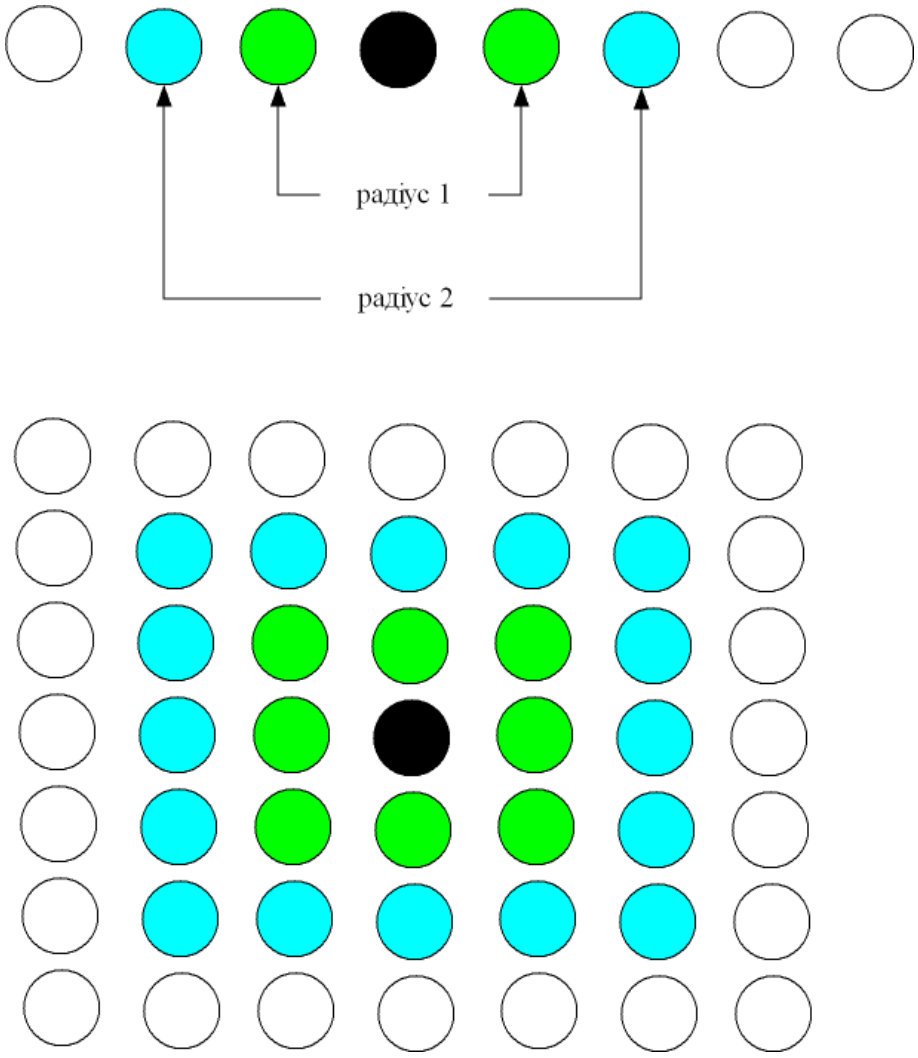


Рис. 12. Угорі кластерні елементи розташовані у один ряд, внизу – у вигляді квадратної сітки. Від розташування залежить, які саме елементи попадуть усередину круга заданого радіуса з центром у елементі-переможці. У випадку радіуса 1 оновлюється чорний та зелені елементи, для радіуса 2 – чорний, зелені та блакитні елементи

Навчання мережі SOFM

У наступному алгоритмі η позначає норму навчання, а n – крок у часі.

Алгоритм 2 (алгоритм навчання мережі SOFM)

Вхід: множина точок D, кількість k кластерних елементів у сітці

Вихід: змінені ваги кластерних елементів, що трактуються як координати розміщення кластерних елементів

(1) ініціалізувати вагові значення випадковими числами

(2) виконувати, поки не виконається критерій зупинки

(2.1) для кожного вхідного вектора

(2.1.1) для кожного кластерного елемента обчислити відстань до навчального вектора:

$$d_j = \sum_i \left(w_{ij} - x_i\right)^2$$

(2.1.2) знайти елемент j, для якого відстань є найменшою

(2.1.3) для елементів з кола заданого радіуса оновити вагові вектори за формулою

$$w_{ij}(n+1) = w_{ij}(n) + \eta(n)(x_i - w_{ij}(n))$$

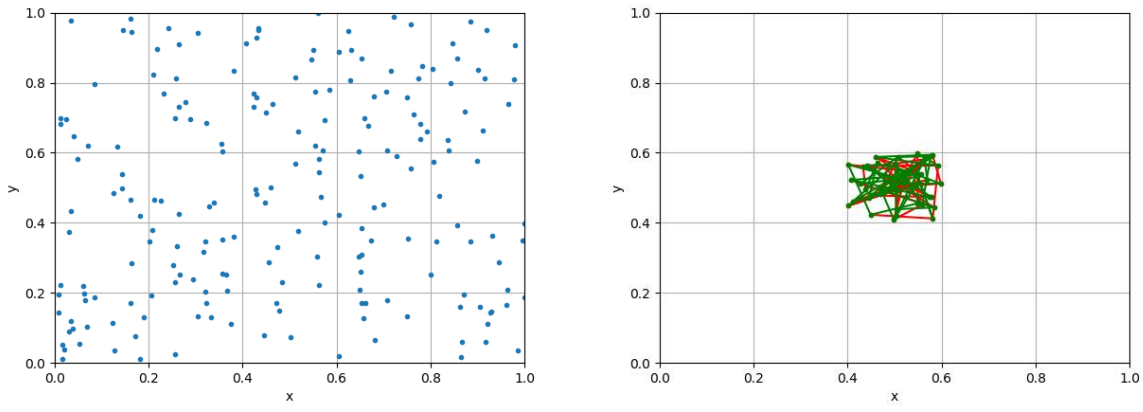
(2.1.4) перевірити, чи потрібне оновлення норми навчання або радіусу

(2.1.5) перевірити критерій зупинки

Навчальні точки вибираються з вхідної множини точок випадковим чином. Умова зупинки виконується тоді, коли величини зміни вагів для усіх кластерних елементів стають дуже малими: за цієї умови навчальні точки повинні потрапляти в одні й ті ж зони карти при переході від однієї епохи (ітерації) до наступної.

Норма навчання з часом змінюється. Вона може, наприклад, спочатку мати значення 0.9, а потім зменшуватися лінійно до деякого фіксованого мінімального значення (наприклад, 0.001), після чого залишатися незмінною. Радіус зазвичай вибирається досить великим, щоб спочатку оновлювалися усі елементи. Радіус теж згодом зменшується, і в кінці, як правило, повинні оновлюватися лише кілька сусідніх з елементом-переможцем елементів, або взагалі лише сам цей елемент. Норма навчання теж може залежати від того, наскільки близько розміщується оновлюваний елемент до елемента-переможця.

Карта ознак проходить два етапи навчання. На першому етапі елементи упорядковуються так, щоб відображати простір вхідних елементів, а на другому етапі відбувається уточнення їх позицій. Як правило, процес представляється візуально шляхом використання двовимірних даних і побудови відповідної поверхні. Наприклад, вхідні вектори вибираються випадковим чином на основі рівномірного розподілу у деякому квадраті, і починається навчання карти. У певні моменти у ході навчання будуються зображення карти шляхом використання відповідності, показаної на рис 11. Елементи з'єднуються лініями, щоб показати їх відносне розташування. Спочатку карта має вигляд сильно "зіжмаканої", але поступово у ході навчання вона розгортається і розправляється. Кінцевим результатом навчання є карта, що покриває увесь вхідний простір і є досить регулярною (тобто, елементи виявляються розподіленими майже рівномірно). Для прикладу було розглянуто карту з топологією квадрату з 49 елементів (7×7), і для 200 точок даних (рис. 13a), взятих з одиничного квадрату, було проведене її навчання, яке починалося з випадкового набору вагових значень, що задають розміщення кластерних елементів у центрі вхідного простору, як показано на рис. 13b.

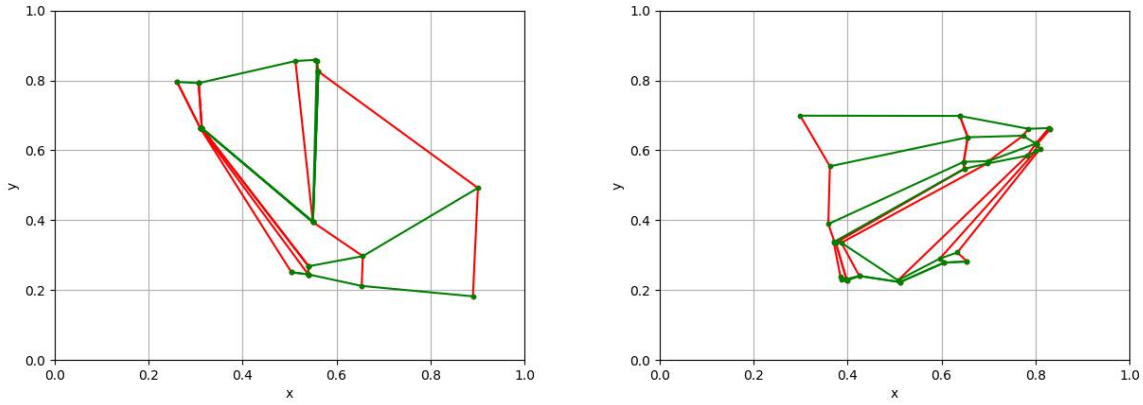


a) b)

Рис. 13. a) початковий набір точок з квадрату $[0,1]\times[0,1]$; b) вагові вектори ініціалізуються випадковими значеннями з відрізку $[0.4,0.6]$

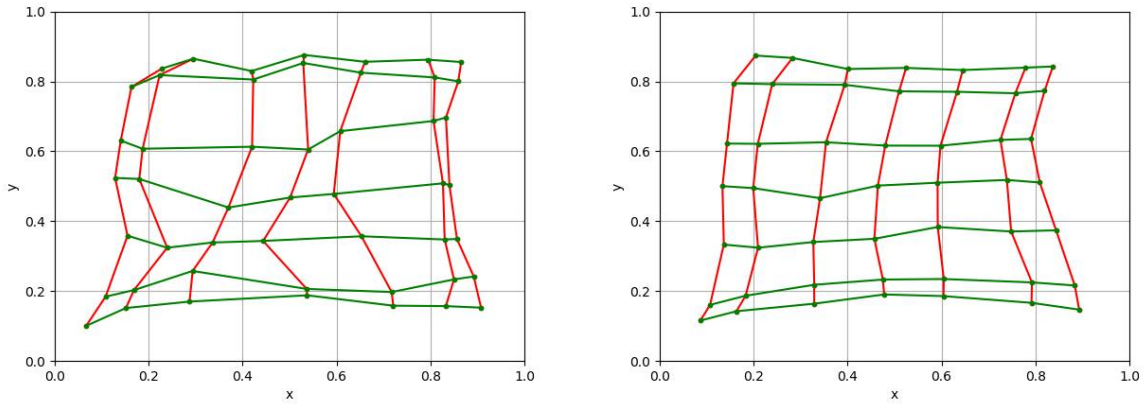
На рис. 14 і 15 ілюструється процес розгортання карти з часом. Як і для інших типів мереж, у даному випадку результат навчання залежить від навчальних даних і вибору параметрів навчання.

Отримана в результаті навчання мережі SOFM карта подана на рис. 15b. При навчанні використовувалось зменшення радіусу на 1 після кожних 400 ітерацій, а зменшення норми навчання – на 0.00035 на кожній ітерації. Всього було проведено 3000 ітерацій, при цьому мінімальне значення радіуса було встановлене в 1, а мінімальне значення коефіцієнта навчання – 0.001.



a) b)

Рис. 14. a) карта вихідних елементів після 500 ітерацій навчання мережі SOFM; b) карта вихідних елементів після 1000 ітерацій навчання мережі SOFM. Елементи, розміщені горизонтально на карті, з'єднані зеленими відрізками, розміщені вертикально на карті, з'єднані червоними відрізками



a) b)

Рис. 15. a) карта вихідних елементів після 2000 ітерацій навчання мережі SOFM; b) карта вихідних елементів після 3000 ітерацій навчання мережі SOFM

Як видно з рис.15b, отримана карта Є не абсолютно рівною. Більше того, в результаті навчання мережі можна отримати навіть "зім'яті" ділянки. Скручування можуть бути результатом неправильного вибору початкових вагових значень, які на початку навчання задають місце розташування кожного кластерного елемента у просторі вхідних даних. Наприклад, якщо на початку навчання усі елементи мають такі ваги, що вони виявляються зміщеними у одному напрямку у просторі вхідних даних, то процес упорядкування може ускладнитися.

У випадку, якщо, наприклад, карта буде лінійним масивом, під час навчання цей лінійний масив скручується так, щоб заповнити простір вхідних даних. Ті ж властивості просторового заповнення можна застосувати й у разі інших форм. Наприклад, якщо набір навчальних даних вибирається рівномірно з трикутника, то можна розмістити елементи квадратної карти так, щоб ці елементи заповнили трикутник.

Реалізація мережі SOFM

Подані у даному розділі рисунки 13-15 створені за допомогою розробленої під час написання програми навчання мережі SOFM.

Для генерації початкових даних використовується функція `makeInitSetSOFM(P,d)`, що приймає в якості аргументів параметри, схожі на параметри функції `makeInitSet(P)`, описаної у попередньому розділі.

Значення вагів зв'язків генеруються за допомогою функції `makeWSOFM2d(nX,nY,limits)`, що приймає в якості параметрів кількість вхідних елементів `nX`, кількість вихідних елементів `nY`, та список коефіцієнтів масштабування значень параметрів, які, зазвичай, повинні знаходитись у квадраті $[0.4,0.6]\times[0.4,0.6]$.

Виконання однієї ітерації навчання проводиться у функції `makeOneIteration2d(X,W,R,etha)`, що приймає в якості параметрів множину точок `X`, набір вагів вихідних елементів мережі `W`, радіус `R` для визначення оновлюваних елементів, та норму навчання `etha`. У функції реалізовані кроки 2.1.1-2.14 алгоритму 2.

Нарешті, додаткова функція `classifySOFMPoints(X,W)` призначена для розподілу точок вхідної множини X по кластерам, заданим ваговими коефіцієнтами (чи, іншими словами, вихідними елементами) `W`. Повертає функція прямокутний масив кількості точок, розміщених у кожному з кластерів. На відміну від алгоритму кластеризації k-means, у функції не визначається належність кожної точки до певного кластеру, оскільки завданням цієї функції є лише визначення кількості точок, що належать до кожного кластеру для подальшого прийняття рішення про кількість наявних кластерів у вхідних даних. Наприклад, для набору точок, зображеного на рис. 13а, результат розподілу точок по кластерам є наступним:

```
[[ 6.  1.  7.  5.  2.  6. 11.]
 [ 2.  5.  1.  3.  4.  2.  4.]
 [ 5.  3.  4.  3.  4.  7.  4.]
 [ 4.  3.  2.  4.  2.  2.  4.]
 [ 5.  3.  5.  3.  3.  6.  5.]
 [ 6.  3.  3.  3.  2.  1.  2.]
 [ 8.  3.  7.  2.  6.  3. 11.]]
```

З наведеного прикладу видно, що до першого (верхній лівий) кластера віднесено 6 точок, сьомого (верхній правий) – 11 точок, останнього (нижній правий) – 11 точок. Приблизно рівномірне розміщення точок у кластерах пояснюється їх приблизно рівномірним розподілом по одиничному квадрату (див. рис. 13а). Однак, ***головним припущенням даної роботи*** є те, що при вираженій кластеризації точок їх розподіл по кластерним елементам карти SOFM буде таким, що дозволить оцінити кількість наявних кластерів.

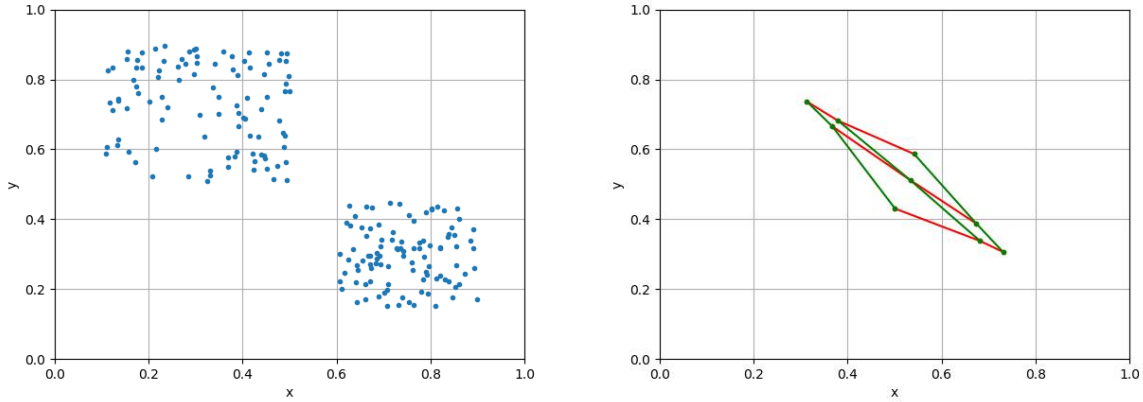
Отже, у даній частині роботи розглянуто алгоритм карти SOFM, описано результати роботи реалізації алгоритму мовою Python, та сформульовано головну гіпотезу роботи.

3. Алгоритм автоматичного визначення кількості кластерів

Для перевірки гіпотези, сформульованої у кінці розділу 2, було об'єднано програми, розроблені у двох попередніх розділах.

У отриманій програмі спочатку генерується набір даних подібно до наведеного на рис. 1, тобто, таких, що мають яскраво виражені кластери. Потім на згенерованих даних проводиться навчання мережі SOFM та аналізується отримана інформація.

У експерименті 1 було згенеровано множину точок, що складається з двох кластерів (рис. 16a). В якості вихідних елементів мережі SOFM було обрано сітку розміром 3×3, що в результаті може давати до 9 кластерів. Отримана після навчання мережі SOFM сітка подана на рис. 16b.



a) b)

Рис. 16. a) вхідна множина точок з двома вираженими кластерами; b) карта вихідних елементів розміром 3×3 після 3000 ітерацій навчання мережі SOFM

Розподіл елементів по кластерах після навчання мережі є наступним:

```
[[ 0. 17. 66.]
 [15.  5. 17.]
 [54. 19.  7.]]
```

У даному випадку видно дві особливості результату експерименту.

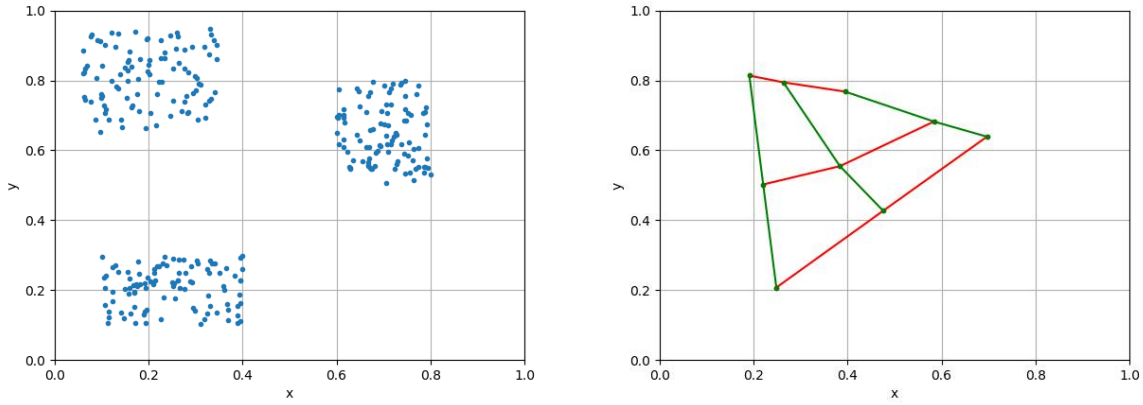
1. Отримано сітку, у якій змінилось топографічне розташування елементів: елементи, що знаходяться у сітці по рядках, розташувались переважно вертикально і навпаки.

2. Отримано 2 кластерні елементи з суттєво більшою кількістю точок – 66 та 54 – порівняно з іншими кластерами. Кількість таких кластерів рівна 2.

У експерименті 2 було згенеровано множину точок, що складається з трьох кластерів (рис. 17a). В якості вихідних елементів мережі SOFM було обрано сітку розміром 3×3, що в результаті знову може давати до 9 кластерів. Отримана після навчання мережі SOFM сітка подана на рис. 17b.

Розподіл елементів по кластерах після навчання мережі є наступним:

```
[[82.  2. 98.]
 [18.  0.  0.]
 [ 5. 34. 61.]]
```



a) b)

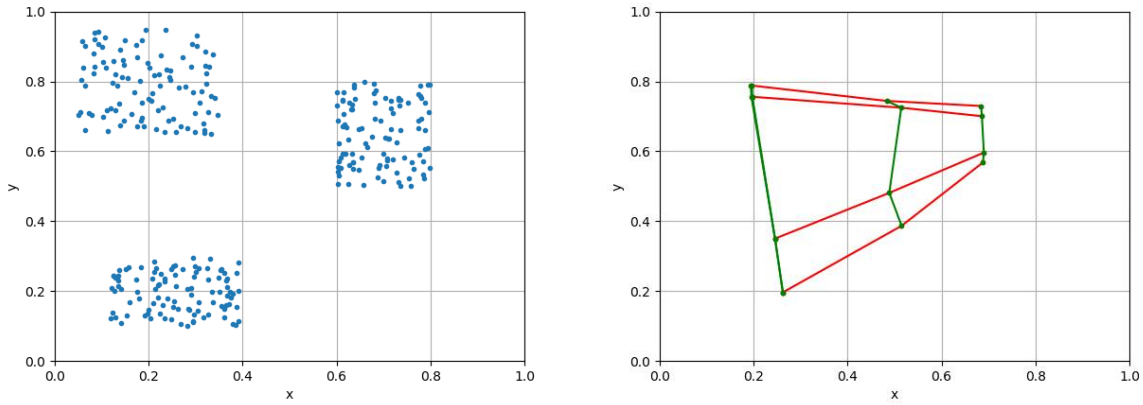
Рис. 17. a) вхідна множина точок з трьома вираженими кластерами; b) карта вихідних елементів розміром 3×3 після 3000 ітерацій навчання мережі SOFM

Результатами експерименту є наступні.

1. Отримано сітку, звужену вправо, де на вихідній множині знаходиться лише один кластер.

2. Отримано 3 кластерні елементи з суттєво більшою кількістю точок – 98, 82, 61 – порівняно з іншими кластерами. Кількість таких кластерів рівна 3.

Нарешті, у експерименті 3 було згенеровано множину точок, що складається з трьох кластерів (рис. 18a), але в якості вихідних елементів мережі SOFM було обрано сітку розміром 4×4, що в результаті може давати до 16 кластерів. Отримана після навчання мережі SOFM сітка подана на рис. 18b.



a) b)

Рис. 18. a) вхідна множина точок з трьома вираженими кластерами; b) карта вихідних елементів розміром 4×4 після 3000 ітерацій навчання мережі SOFM

Розподіл елементів по кластерах після навчання мережі є наступним:

```
[[34.  0. 48. 45.]
 [17.  0.  0.  7.]
 [17.  0. 27. 16.]
 [32.  2. 20. 35.]]
```

Результатами експерименту є наступні.

1. Отримано сітку, звужену вправо, де на вихідній множині знаходиться лише один кластер.

2. Отримано 3 групи кластерних елементів, розділені між собою кластер ними елементами з кількістю точок, менше 10. Першою групою є перший стовпчик сітки: 34, 17, 17, 32. Другою групою є два верхні праві елементи сітки: 48, 45. Третьою групою є 4 елементи у нижньому правому куті: 27, 16, 20, 35.

Таким чином, результати експериментів показують можливість застосування мережі SOFM для оцінки кількості кластерних елементів. Однак, сама оцінка вважається на поточний момент складнішою, аніж просто визначення кількості вихідних елементів мережі SOFM з високими значеннями кількості точок вхідної множини, що їм відповідають.

Таким чином, на основі проведеного дослідження можна сформулювати наступний алгоритм визначення кластерів та проведення кластеризації множини об'єктів.

Алгоритм 3 (кластеризація множини точок з апріорною оцінкою кількості кластерів)
Вхід: множина точок D
Вихід: множина кластерів C, кожний з яких подається точкою центра кластера c_i, та масив *clust*, що ставить у відповідність кожну точку номеру кластера, до якого вона належить

(1) навчити на множині D мережу SOFM
(2) підрахувати кількість елементів множини D, що відповідають кожному вихідному елементу мережі SOFM
(3) провести аналіз розподілу кількості елементів, що відповідають вихідним елементам мережі SOFM та визначити апріорну кількість кластерів k
(4) провести кластеризацію множини D за алгоритмом k-means з використанням кількості кластерів k

У даному випадку недолік алгоритму k-means, що стосується неправильної початкової кількості кластерів, буде дещо згладженим за рахунок попередньої оцінки та отримання більш адекватної кількості кластерів.

Однак, зауважимо, що саме крок (3) алгоритму 3 потребує додаткових досліджень.

Ще одним напрямком подальшого дослідження є збільшення розмірності простору ознак та оцінка результатів роботи алгоритму 3 на множині точок більшої розмірності.

Висновки

Отже, у результаті виконання дослідження:

1. Розглянуто алгоритм пошуку кластерів k-means, описано результати роботи реалізації алгоритму мовою Python, та визначено недоліки алгоритму.

2. Розглянуто та реалізовано мовою Python алгоритм навчання мережі SOFM.

3. Сформульовано та перевірено гіпотезу про можливість застосування мережі SOFM для попередньої оцінки кількості кластерів у заданому наборі точок. Сформовано алгоритм та визначено подальші напрямки дослідження.

Список використаної літератури:

1. Hartigan, J.A. (1975) Clustering Algorithms. John Wiley & Sons, New York, 351 p.
2. Kohonen, T. (1990) The Self-Organizing Map. Proceedings of the IEEE, 78, 1464-1480. http://dx.doi.org/10.1109/5.58325
3. Осовский С. Нейронные сети для обработки информации / Пер. с польского И.Д. Рудинского. – М.: Финансы и статистика, 2002. – 344 с.
4. Рашид Т. Создаем нейронную сеть.: Пер. с англ. – СПб.: ООО "Альфа-книга", 2017. – 272 с.
5. Хайкин С. Нейронные сети: полный курс, 2-е издание.: Пер. с англ. – М.: Издательский дом «Вильямс», 2006. – 1104 с.

References:

1. Hartigan, J.A. (1975) Clustering Algorithms. John Wiley & Sons, New York, 351 p.
2. Kohonen, T. (1990) The Self-Organizing Map. Proceedings of the IEEE, 78, 1464-1480. http://dx.doi.org/10.1109/5.58325
3. Osovsky S. Neural Networks for information processing. – M.: Finance and Statistics, 2002. – 344 c. [in Russian].
4. Rashid T. Making neural networks. – SPb.: "Alpha-book" LTd, 2017. – 272 c. [in Russian].
5. Haikin S. Neural networks, 2nd Ed. – M.: Williams Publishing, 2006. – 1104 c. [in Russian].

SERDIUK Oleksandr,
Candidate of Economic Sciences, Associate Professor, Department of Informatics and Applied Mathematics, The Bohdan Khmelnytsky National University of Cherkasy, Ukraine
PISKUN Oleksandr,
Candidate of Technical Sciences, Associate Professor, Chair of Department of Applied Mathematics and Informatics, Bohdan Khmelnytsky National University of Cherkasy, Ukraine
DIKHTIARENKO Volodymyr,
Drabivskyi NVK "Secondary School of I-III st. named after S.V. Vasylchenko-gymnasium" of Drabiv District Council
BILETSKA Nadiya,
Teacher of Mathematics of the High Qualification Category, Pedagogical Title "Teacher-Methodologist" of Drabivskyi NVK "Secondary School of I-III st. named after S.V. Vasylchenko-gymnasium" of Drabiv District Council

PRELIMINARY DETERMINATION OF THE NUMBER OF CLUSTERS USING SOFM ARTIFICIAL NEURAL NETWORKS

Summary. Introduction. *The task of clustering is to divide the set of objects under study into groups of "similar" objects, called clusters. The purpose of cluster analysis is to find the structures that exist in the data, which is expressed in the formation of groups of similar objects. At the same time, the effect of cluster analysis is to structure the objects under study. This means that clustering techniques are needed to identify a structure in the data that is not easy to find by visual inspection or with the help of experts. However, cluster analysis methods require some a priori information. In particular, before performing a cluster analysis, most methods already need to know the number of clusters. Such information is not always possible due to, for example, the large dimension of the data, so researchers in many cases choose the number of clusters intuitively, which, of course, does not always lead to significant results. Therefore, before clustering, it seems possible to use any other methods that would allow to estimate at least approximately the number of possible clusters.*

Purpose. *The aim of the study is to determine the possibility of using a specific type of artificial neural networks - a self-organized map of features - to pre-determine the number of clusters for a particular method of clustering - the method of k-means. The tasks set in the study are: reviewing of the k-means clustering algorithm and its implementation; reviewing of a self-organized map of signs and its implementation; to analyze the possibilities of using a self-organized feature map to obtain a priori information on the number of clusters for the k-means algorithm and develop recommendations for the use of self-organized feature maps in data clustering.*

Results. *One of the simplest clustering algorithms is k-means, which is based on an iterative process of stabilizing cluster centroids. The main characteristic of the cluster is its centroid and all the work of the algorithm is aimed at stabilization or - at best - complete freezing of changes in the centroid of the cluster. The k-means algorithm builds k clusters placed at as large distances from each*

other as possible. The main type of problem solved by the k-means algorithm is the presence of hypotheses about the existence of a certain number of clusters in the data set, and they should be as different as possible. The choice of the number k can be based on the results of previous research, theoretical considerations or intuition. When specifying the wrong number of clusters, the algorithm generates the wrong result, to avoid which we propose to use one of the methods of preliminary estimation of the possible number of clusters, using the results of neural network with learning without a teacher, which is a self-organizing map - Kohonen network.

The self-organized map of features has a set of input elements, the number of which corresponds to the dimension of the training points, and a set of output elements that act as prototypes. Cluster elements are placed in the form of a one- or two-dimensional array. During the training, all elements can be considered as candidates for the award in the form of training points. For the winning element, the weight values are adjusted so that this cluster element is located even closer to the training point. The weight values of an element are updated if it is inside a circle of a given radius centered in the winning element. During training, the radius usually gradually decreases. In the first stage, the elements are arranged so as to reflect the space of the input elements, and in the second stage, their positions are refined. Typically, the process is represented visually by using two-dimensional data and constructing an appropriate surface.

During our work we made a series of experiments to combine the two above algorithms, using the results of the SOFM network as a priori data for the k-means algorithm. The results of the experiments show the possibility of using the SOFM network to estimate the number of cluster elements. However, the estimate itself is currently considered more complex than simply determining the number of SOFM network outputs with high values of the number of input set points corresponding to them.

***Conclusion.** As a result of the work we have hext conclusions. 1. The algorithm of k-means is reviewed, the results of work of algorithm implementation in Python language are described, and shortcomings of algorithm are defined. 2. The algorithm of SOFM network learning is reviewed and implemented in Python language. 3. The hypothesis about the possibility of using the SOFM network for preliminary estimation of the number of clusters in a given set of points is formulated and tested. An algorithm is formed and further directions of research are determined.*

***Keywords:** machine learning, clustering, k-means, neural networks, SOFM.*

Одержано редакцією 08.02.2021 р.
Прийнято до публікації 10.06.2021 р.

ЗМІСТ

СЕКЦІЯ «ПРИКЛАДНА МАТЕМАТИКА»

СЕКЦІЯ «ІНФОРМАТИКА»

CONTENTS

APPLIED MATHEMATICS SECTION

INFORMATICS SECTION

Наукове видання

ВІСНИК ЧЕРКАСЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ ІМЕНІ БОГДАНА ХМЕЛЬНИЦЬКОГО

Серія
ПРИКЛАДНА МАТЕМАТИКА. ІНФОРМАТИКА

BULLETIN OF THE CHERKASY BOHDAN KHMELNYTSKY NATIONAL UNIVERSITY

APPLIED MATHEMATICS. INFORMATICS

ВИПУСК 1
ISSUE 1

Підп. до друку 20.12.21. Формат 60x84 /8.
Папір офсетний. Друк цифровий.
Гарнітура Times New Roman.
Ум. друк. арк. 13,4. Тираж 100 прим.

Це видання надруковано на папері із деревини, відповідної нормам екологічного лісовикористання

ВИДАВНИЦТВО • ДРУКАРНЯ • СУВЕНІРИ

Видавець ФОП Гордієнко Є.І.
Свідоцтво про внесення суб'єкта видавничої справи до Державного реєстру видавців, виготовників і розповсюджувачів видавничої продукції Серія ДК № 4518 від 04.04.2013 р.
вул. Святотроїцька, 73, м. Черкаси, Україна, 18000, тел. +38 (067) 444-28-94
e-mail: book.druk@gmail.com
@drukcherkasy