МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних та практичних робіт

з дисципліни «Математична логіка та теорія алгоритмів»

для студентів спеціальності 126 –

«Інформаційні системи та технології»

Міністерство освіти і науки України
Вінницький національний технічний університет

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних та практичних робіт
з дисципліни «Математична логіка та теорія алгоритмів»
для студентів спеціальності 126 –
«Інформаційні системи та технології»

Електронне видання
комбінованого (локального та мережного) використання

Вінниця
ВНТУ
2021

Рекомендовано до видання Методичною радою Вінницького національного технічного університету Міністерства освіти і науки України (протокол № 10 від 20.05.2021 р.)

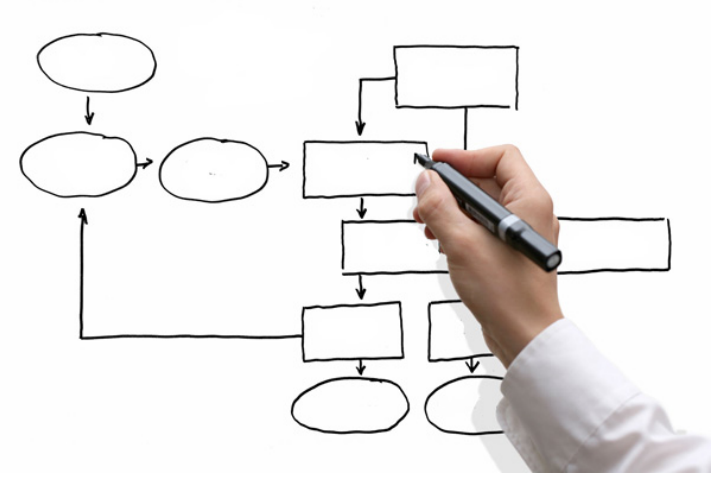
Рецензенти:
П. І. Кулаков, доктор технічних наук, професор ВНТУ
А. Я. Кулик, доктор технічних наук, професор ВНМУ ім. М. Пирогова

Методичні вказівки до виконання лабораторних та практичних робіт з дисципліни «Математична логіка та теорія алгоритмів» для студентів спеціальності 126 – «Інформаційні системи та технології» [Електронний ресурс] / Уклад. Ю. Ю. Іванов. – Вінниця : ВНТУ, 2021. – (PDF 36 с.)

У методичних вказівках наведено завдання на лабораторні та практичні роботи з дисципліни «Математична логіка та теорія алгоритмів», а також рекомендована для їхнього виконання література. Сформовано блоки контрольних питань, які можна використовувати для підготовки до контрольних заходів. Ця дисципліна заснована на матеріалах курсів «Дискретна математика» (Алгоритми, Графи) та «Теорія штучного інтелекту» (Оптимізація, Нейромережі), які необхідно знати студентам для виконання поданих задач.

ЗМІСТ

ЗАХИСТ ТА СТРУКТУРА ЗВІТУ ЛАБОРАТОРНИХ РОБІТ



Студенти виконують *підготовчу роботу* відповідно до теми завдання, щоб зрозуміти теоретичні та практичні аспекти, математичну основу, застосування цих задач у професійній діяльності як тестових полігонів. Вони вивчають конспект лекцій, рекомендовані матеріали (література, аудіо-, відеофайли) тощо. Глибоке вивчення теоретичного матеріалу, набуті навички програмування та вміння використовувати пакети прикладних програм допоможуть успішно виконати поставлені задачі.

За результатами виконаної роботи на аркушах формату А4 оформлюється *звіт*, у якому відображаються всі результати роботи. Він має містити такі *елементи*: титульний аркуш, аналіз літератури та ключового матеріалу, формалізована постановка задачі, розв'язання задачі (числовий приклад) та алгоритми методів, програмне забезпечення, результати роботи, висновки, список використаної літератури. Оформлення звіту необхідно виконувати охайно за *правилами ДСТУ 3008:2015 та ДСТУ ГОСТ 7.1:2006* [1, 2]. Спочатку звіт необхідно подати викладачу, після виконання коригувань студент може захистити роботу. Також потрібно показати роботу програмного забезпечення на ЕОМ та відповісти на контрольні запитання, задані викладачем.

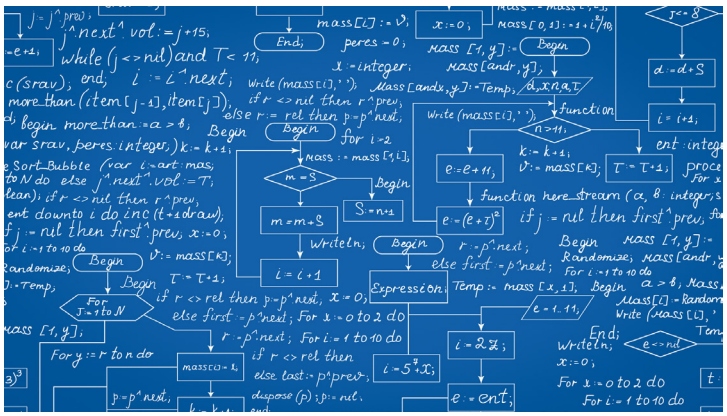
Студент має впевнено та чітко *презентувати* виконану ним роботу, показати її суть, знайомство з ключовою літературою; вміти розв'язати поставлену задачу та визначити методи її вирішення; володіти відповідним термінологічним апаратом; мати прийнятний рівень мовної грамотності, включаючи володіння функціональним стилем викладення матеріалу. Чітке виконання поставлених вимог та використання аргументованих відповідей на запитання викладача з посиланнями на роботи вчених допоможе підвищити оцінку.

КРИТЕРІЇ ОЦІНЮВАННЯ ЗНАНЬ, УМІНЬ ТА НАВИЧОК СТУДЕНТІВ

Рівень компетентності	**За нац. шкалою**	**За шкалою ЕКТС**	**Критерії оцінювання**
IV Високий (творчий) «5»	відмінно «5»	*A*	Виставляється, якщо під час відповіді на питання виявлено всебічні, систематизовані, глибокі знання матеріалу, який виноситься на контроль, уміння вільно виконувати завдання, передбачені програмою, знання основної і додаткової літератури, передбаченої програмою на рівні творчого використання.
III Достатній (конструктивний) «4»	добре «4+»	*B*	Повні знання з питань і задач, що стоять перед студентом. Уміння викладати основні ідеї. Вміння професійно відстоювати свою точку зору. Припускаються несуттєві неточності у викладенні матеріалу та у відповідях.
	добре «4»	*C*	Достатньо повні знання з поставлених питань і задач. Вміння викладати основні ідеї. Здатність самостійно застосовувати вивчений матеріал на рівні стандартних ситуацій, наводити власні приклади на підтвердження власних тверджень. Вміння доводити правильність своїх рішень. Несуттєві неточності у відповідях та деякі нераціональності при програмуванні задач.
II Середній (репродуктивний) «3»	задовільно «3+»	*D*	Студент може відтворити значну частину теоретичного матеріалу, виявляє знання та розуміння основних положень, з допомогою викладача може аналізувати матеріал, робити висновки та розробляти програмні блоки. Пояснення неповні, нелаконічні, не завжди точні. Відповіді на питання неповні, містять неточності, при програмуванні застосовуються не найраціональніші рішення.
	задовільно «3»	*E*	Задовільні знання програмного матеріалу на рівні вищому за початковий. Здатність за допомогою викладача логічно відтворювати значну частину матеріалу. Під час відповіді на запитання виникають труднощі у деяких положеннях, відповіді не повні, програми пишуться нераціонально, не використовуються всі ефективні засоби програмування.
I Низький «2»	незадовільно з можливістю повторного складання «2»	*FX*	Теорією володіє на рівні фрагментів, викладає матеріал уривчасто. Помітна складність в обґрунтуванні рішень. Самостійно, без допомоги викладача, не може сформувати алгоритм рішення задачі. Програми не раціональні та неефективні.
	незадовільно з обов'язковим повторним вивченням «2»	*F*	Теорією володіє на рівні фрагментів, викладає матеріал уривчасто. Не може обґрунтувати рішення задачі, на запитання викладача дає неправильні відповіді. Самостійно не може сформувати алгоритм рішення задачі.

ВСТУП

Щоб зрозуміти алгоритм, потрібно його побачити.
Д. Е. Кнут



Саме слово «*алгоритм*» походить від імені арабського вченого *Абу Абдуллах Мухаммеда ібн Муса аль-Хорезмі*, який близько 825 р. написав працю, де вперше описав придуману в Індії позиційну десяткову систему числення. Він сформулював правила обчислень у новій системі та, імовірно, вперше використав цифру 0 для позначення пропущеної позиції в записі числа (її індійську назву араби перевели як *as-sifr* або *sifr*, звідси такі слова, як «цифра» і «шифр»). У першій половині XII ст. книга аль-Хорезмі в латинському перекладі потрапила в Європу. Перекладач дав їй назву *Algoritmi de numero Indorum* («Алгоритми про індійський рахунок»). По-арабськи книга називалася «*Кітаб аль-джебр валь-мукабала*» («Книга про додавання і віднімання»). З оригінальної назви книги походить також слово «алгебра».

Теорія алгоритмів – наука, яка вивчає загальні властивості і закономірності алгоритмів і різноманітні формальні моделі їх подання, займається проблемою ефективних обчислень. До задач теорії алгоритмів належать формальний доказ алгоритмічної нерозв'язності задач, асимптотический аналіз складності алгоритмів, класифікація алгоритмів відповідно до класів складності, розробка критеріїв для порівняння якості алгоритмів тощо. Загалом, написавши алгоритм, ми маємо запитати себе:

1. *Чи коректний він*?
2. *Який час його роботи* (*залежно від розмірності задачі*)?
3. *Чи існує більш швидкий алгоритм*?

Більшість напрямків сучасної науки, включаючи робототехніку, генну інженерію, біоніку, штучний інтелект тощо застосовують різні алгоритми, які дозволяють виконувати пошук раціональних рішень за прийнятний час роботи в багатовимірному просторі альтернатив, враховуючи нелінійні залежності та дуже великий обсяг даних, що зумовлює високу трудомісткість обчислень [3–8].

У цьому курсі *студентам запропоновано у якості тестових полігонів низку теоретичних* та *практичних* задач з різних науково-практичних галузей. Варто зазначити, що в умовах обмеженого часу досить важко охо-

пити всі теми такого об’ємного курсу. Тому знання, набуті студентами на попередніх курсах та самостійна робота, абсолютно необхідні для чіткого формулювання понять і постановок різних прикладних інтелектуальних задач, їх формалізації і комп'ютеризації, а також для засвоєння і розроблення сучасних інформаційних технологій.

Метою дисципліни є вироблення системи знань, умінь і навичок у студентів як майбутніх фахівців щодо вибору ефективного алгоритму для розв’язання різноманітних науково-практичних задач, пошуку раціональних рішень в багатовимірному просторі альтернатив та прихованих закономірностей у множині даних, реалізації етапів обчислень на комп’ютерній техніці з використанням прикладних пакетів математичного моделювання або розробленим програмним забезпеченням у різних галузях професійної діяльності. Обов’язковим є виконання для кожної задачі числового прикладу (наприклад, у системах математичного моделювання – *MatLab*, *MathCad*, *Mathematica*, *Maple* тощо), підвищує бали розроблення програмного забезпечення з обґрунтуванням вибору технологічної платформи для дослідження ефективності роботи методу розв’язання задачі.

Програмне забезпечення розробляється з використанням принципів структурного, функціонального, об’єктно-орієнтованого програмування. Програмний комплекс має бути гнучким, наприклад, надавати можливість змінювати параметри вхідних даних; включати в себе графічне меню, яке забезпечує можливість введення початкових умов задачі, вибір параметрів та збереження отриманого розв’язку.

Таким чином, студенти мають розширити свої знання за цим курсом та вдосконалити навички програмування, що знадобиться їм у професійній діяльності.

ПРАКТИЧНА РОБОТА

Основне завдання дисципліни полягає у виконанні *самостійного пошуку* студентом (або за допомогою викладача) певної цікавої *науково-дослідницької задачі*, її аналізу, розв'язанню (*теоретично* та *програмно*).

Студенти виконують *підготовчу роботу відповідно до теми завдання*, щоб зрозуміти теоретичні та практичні аспекти, математичну основу, застосування цих задач у професійній діяльності, як тестових полігонів. Вони вивчають потрібні матеріали (література, аудіо-, відеофайли) тощо. Глибоке вивчення теоретичного матеріалу, набуті навички програмування та вміння використовувати пакети прикладних програм допоможуть успішно провести практичне заняття. Обов'язковим є виконання для кожної задачі числового прикладу (наприклад, у системах математичного моделювання – *MatLab, MathCad, Mathematica* тощо) та розроблення програмного забезпечення.

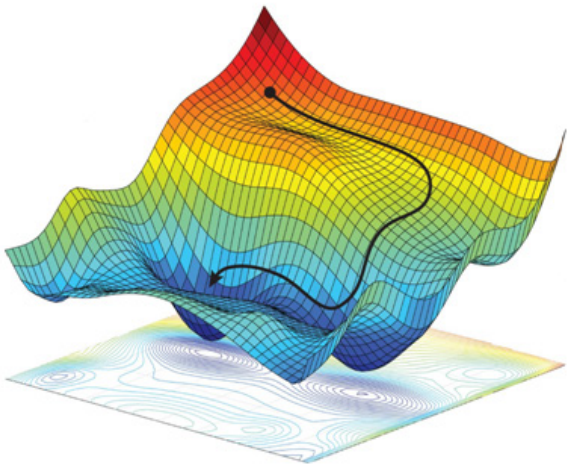
Студент має *оформити звіт* з виконаної роботи, *презентацію* та *виступити* перед аудиторією, яку представляє група студента та викладач дисципліни. Попередньо тема виступу обговорюється з викладачем курсу.

Спершу звіт необхідно подати викладачу, після виконання коригувань студент може захистити роботу. Оформлення звіту необхідно виконувати охайно за правилами *ДСТУ 3008:2015*, *ДСТУ ГОСТ 7.1:2006* [1, 2] для наукових робіт. Він має містити такі *елементи*: титульний аркуш, аналіз літератури та ключового матеріалу, формалізована постановка задачі, розв'язання задачі (числовий приклад) та алгоритми методів, програмне забезпечення, результати роботи, висновки, список використаної літератури.

У *презентації* студент наглядно показує результати своєї роботи. Для отримання *відмінної оцінки* необхідно виступити перед аудиторією, яку представляє група студента та викладач курсу. Студент має впевнено та точно презентувати виконану ним роботу.

Виконання такого завдання допоможе сформувати у студентів вміння використовувати методи та засоби викладання, організовувати навчальний процес, навчитися без сорому виступати перед аудиторією, вміти активізувати пізнавальну діяльність, стимулювали до критичного та творчого мислення, самостійного пошуку й дослідження за цим курсом. Студенти покращать свої знання та вдосконалять навички програмування, що знадобиться їм у професійній діяльності.

ЛАБОРАТОРНА РОБОТА № 1



Тема: Метаевристичні алгоритми розв’язання задач неперервної глобальної оптимізації. Ройовий інтелект.

Мета: навчитися застосовувати метаевристичні алгоритми неперервної глобальної оптимізації та відповідне програмне забезпечення для пошуку розв’язку задачі в багатовимірному просторі альтернатив.

Теоретичні відомості: для детального розуміння матеріалу необхідно проаналізувати джерела [3, 4, 9–38], у яких подано теоретичні основи теорії неперервної глобальної оптимізації [9–14], ройового інтелекту, метаевристик [15–26] та проаналізовано відповідні задачі [22–38].

Завдання

Подати загальну математичну постановку задачі глобальної неперервної оптимізації. Виділити задачі умовної та безумовної оптимізації (*constrained* / *unconstrained global optimization task*). Проаналізувати та теоретично обґрунтувати:

- складність задачі оптимізації (класи *P* та *NP*);
- аналітичний метод пошуку екстремуму функції багатьох змінних;
- пошукові методи безумовної оптимізації (*search techniques*);
- метод умовної оптимізації з використанням штрафів або штрафних функцій (*penalty methods*):
  - летальні функції (*death*);
  - статичні функції (*static*);
  - динамічні (*dynamic*);
  - адаптивні (*adaptive*);
  - проблемно-орієнтовані (*problem-oriented*);
- основи стохастичних алгоритмів оптимізації (метаевристик, популяційних, інтелектуальних алгоритмів, ройового інтелекту) (*swarm intelligence*):
  - стратегії формування початкового положення боїдів (агентів);

- методи генерації чисел для випадкових блукань (*random walks*);
- контроль границь області пошуку (*boundary handling technique*);
- критерії зупинки пошуку (*stopping criterions*);
- мультистартова процедура (*multistart*);
- паралелізація метаевристик (*metaheuristics parallelization*): послідовна, острівна, коеволюційна стратегія;

– теорему Вульперта-МакКріді про відсутність безкоштовних сніданків (*No Free Lunch Theorem*).

Розв'язати такі задачі:

Задача 1. Розробити програму для аналізу ефективності роботи метаевристик, які застосовано для розв'язання задачі глобальної безумовної оптимізації:

1. Розв'язати задачу неперервної глобальної безумовної оптимізації, використавши поняття математичного аналізу. Для цього вибрати вихідні дані з табл. 1.1 за виразом $N = (L + [\text{остання цифра року}]) \bmod 10$, де L – варіант студента за списком.

Таблиця 1.1 – Вихідні умови задачі оптимізації

№	Цільова функція $f(x)$	Початкова точка x^0
1	$y = \sum_{i=1}^{n} (x_i - n)^2 / i^2$	$x_i^0 = 2i,\ i = \overline{1,n}$
2	$y = \sum_{i=1}^{n} (x_i + 7 \cdot i)^2 / i$	$x_i^0 = 0,\ i = \overline{1,n}$
3	$y = \sum_{i=1}^{n} (x_i - i)^4$	$x_i^0 = i/2,\ i = \overline{1,n}$
4	$y = \sum_{i=1}^{n} (x_i + i)^2 / 0{,}1i^2$	$x_i^0 = i,\ i = \overline{1,n}$
5	$y = \sum_{i=1}^{n} (x_i + i)^4 / i$	$x_i^0 = 5i,\ i = \overline{1,n}$
6	$y = \sum_{i=1}^{n} (0{,}1x_i + 2i)^2$	$x_i^0 = 3,\ i = \overline{1,n}$
7	$y = \sum_{i=1}^{n} (i \cdot x_i)^2$	$x_i^0 = 2i,\ i = \overline{1,n}$
8	$y = \sum_{i=1}^{n} (i^2 - x^2)^4$	$x_i^0 = 5i,\ i = \overline{1,n}$
9	$y = 10 \cdot n + \sum_{i=1}^{n} (x_i^2 - 10 \cdot \cos(2\pi x_i))$	$x_i^0 = 3,\ i = \overline{1,n}$
10	$y = -\frac{1}{n} \cdot \sum_{i=1}^{n} (x_i \cdot \sin(\sqrt{\lvert x_i \rvert}))$	$x_i^0 = 10i,\ i = \overline{1,n}$

Побудувати двовимірний та тривимірний графіки функцій, відобразити на них глобальний оптимум.

2. Вибрати 2–3 методи оптимізації з табл. 1.2, використавши 2–3 рази генератор випадкових чисел від 1 до *k*. Навести детальні алгоритми їхньої роботи. Виконати на папері вручну низку ітерацій та показати результати. Перевірити розрахунки аналітичним методом.

Таблиця 1.2 – Стохастичні методи оптимізації

Метаевристики (*k* = 36)		
1–12	13–24	25–36
Алгоритм перемішаних стрибаючих жаб (shuffled frog-leaping)	*Алгоритм оптимізації на основі розповсюдження бур'яну (invasive weed optimization)*	*Алгоритм еволюції розуму (mind evolutionary computation)*
Алгоритм кажанів (bat optimization algorithm)	*Алгоритм світляків (firefly algorithm)*	*Генетичні алгоритми (genetic algorithms)*
Алгоритм рою часток (particle swarm optimization)	*Оптимізація роєм світляків (glowworm swarm optimization)*	*Меметичні алгоритми (memetic algorithms)*
Алгоритм штучної бджолиної колонії (artificial bee colony algorithm)	*Алгоритм «великого вибуху-великого стиснення» («Big Bang – Big Crunch» algorithm)*	*Алгоритм дерев, що ростуть (sapling sowing and growing up algorithm)*
Алгоритм рою бджол (bees algorithm)	*Пошук на основі поведінки косяка штучних риб (fish school search)*	*Гравітаційний пошук (gravitational search)*
Зозулиний пошук (cuckoo search)	*Алгоритм інтелектуальних крапель води (intelligent water drop)*	*Електромагнітний пошук (electro magnetism-like algorithm)*
Стохастичний дифузійний пошук (stochastic diffusion search)	*Алгоритм наслідування поведінки мавп (monkey algorithm)*	*Диференціальна еволюція (differential evolution)*
Мавпячий пошук (monkey search)	*Алгоритм пошуку гармонії (harmony search)*	*Імітація відпалу (simulated annealing)*
Бактеріальна оптимізація (bacterial foraging optimization)	*Алгоритм наслідування поведінки мурашиної колонії (ant colony optimization)*	*Променевий пошук (beam search)*
Оптимізація на основі поведінки котячої зграї (cat swarm optimization algorithm)	*Алгоритм оптимізації на основі поведінки китів (whale optimization algorithm)*	*Алгоритм оптимізації на основі поведінки горобця (sparrow search algorithm)*
Оптимізація на основі поведінки зграї сірих вовків (gray wolf opimizer)	*Алгоритм оптимізації на основі поведінки ворон (crow search algorithm)*	*Оптимизация на основі поведінки беркута (Golden eagle optimizer)*
Алгоритм оптимізації центральною силою (central force optimization)	*Алгоритм штучних імунних систем (artificial immune system algorithm)*	*Оптимізація на основі генерації плазми (plasma generation optimization)*

3. Експерименти для функції з табл. 1.1 провести, використавши $n = \overline{2,30}$ керованих змінних, зафіксувати час роботи програми (табл. 1.3). Необхідно змінювати параметри стохастичних алгоритмів та досліджувати їх вплив на ефективність розв'язання задачі оптимізації. Для вибраного набору найкращих параметрів виконати експерименти не менше s = 10–20 разів для заданої функції певної розмірності n. Зберегти результати запусків для подальшого аналізу. Знайти середнє квадратичне відхилення *RMSE* для кожного запуску за формулою $RMSE = \sqrt{(Y-y)^2 / s}$, де Y – теоретичне або відоме значення екстремуму, а y – результат роботи програми для заданої розмірності задачі n. Відобразити у звіті найгірше, середнє та найкраще значення екстремуму за s запусків програми для кожного n, аналогічні дії виконати для часу виконання програми t. Нанести розподіл точок на графік, на якому по осі ординат нанести значення функції y, а на осі абсцис – кількість змінних n.

Для тестування роботи програми використати набір з трьох класичних тестових функцій із роботи [32] або за посиланнями:

- http://infinity77.net/global_optimization/index.html (*Global Optimization Benchmarks*);
- https://www.sfu.ca/~ssurjano/optimization.html (*Virtual Library of Simulation Experiments: Test Functions and Datasets. Optimization Problems*).

Експерименти проводити, не використовуючи готові пакети та функції оптимізації. Їх можна застосувати для перевірки результатів роботи: https://en.wikipedia.org/wiki/List_of_optimization_software.

Таблиця 1.3 – Результати експерименту для однієї мета евристики

Кількість керованих змінних n	Найкращі параметри алгоритму	Результати обчислень $f(x^*)$ (найбільше, найменше, середнє значення функції)	Час пошуку оптимуму t, сек (найбільший, найменший, середній час)	*RMSE*	Кількість запусків s
2					
...	…	…	…	…	…
n					

4. Повторити пункт 3 для всіх вибраних метаевристик. За допомогою *W*-критерію Уілкоксона або критерію суми рангів (*Wilcoxon rank-sum test*) перевірити однорідність / неоднорідність вибірок (результатів розрахунку значень оптимуму) [33] та порівняти ефективність роботи метаевристик на кожній функції.

Задача 2. Розробити програму для аналізу ефективності роботи метаевристик, які застосовано для розв'язання задачі глобальної умовної оптимізації:

1. Розв’язати практичну задачу неперервної глобальної умовної оптимізації із застосуванням метаевристик, які використані у першій задачі. Вибрати дані з табл. 1.4 за виразом $N = (L + [\text{остання цифра року}]) \bmod p$, де L – варіант студента за списком; p = [10, 12, 15]. Також дослідження ефективності роботи метаевристик можна провести з використанням задач *GO Test Problems* (G1-G13), які можна знайти за посиланням:

- http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page422.htm.

Таблиця 1.4 – Назва задачі умовної оптимізації

№	Назва задачі
1	Задача визначення параметрів стержневої ферми [25] (*three-bar truss design problem*)
2	Задача визначення параметрів зварної балки [25] (*welded beam design problem*)
3	Задача визначення параметрів посудини високого тиску [25] (*pressure vessel design problem*)
4	Задача визначення параметрів пружини розтягу / стиску [25] (*tension* / *compression spring design problem*)
5	Задача визначення параметрів пружини Бельвілля [28] (*Belleville spring design problem*)
6	Задача визначення параметрів консольної балки [30] (*cantiliver design problem*)
7	Задача визначення параметрів редуктора швидкості [29] (*speed reducer design optimization problem*)
8	Задача визначення параметрів процесу алкілювання [27] (*alkylation process design*)
9	Задача визначення параметрів теплообмінника [27] (*heat exchanger design*)
10	Задача визначення параметрів оптимального реактора [27] (*optimal reactor design*)
11	Задача визначення коефіцієнтів швидкостей необоротної хімічної реакції першого порядку [27] (*first order irreversible chain reaction*)
12	Задача термінального управління супутником або задача Майера [26] (*terminal satellite control*, *Mayer task*)
13	Вибір оптимального інвестиційного портфелю (*optimal investment portfolio*): модель Г.М. Марковіца (*Markowitz model*) зі зведенням двокритеріальної задачі оптимізації до однокритеріальної на основі теореми С. Карліна про адитиивну згортку [31]
14	Вибір оптимального інвестиційного портфелю (*optimal investment portfolio*): модель Марковіца (*Markowitz model*) із заміною одного з двох критеріїв задачі обмеженням [31]
15	Вибір оптимального інвестиційного портфелю (*optimal investment portfolio*): модель Марковіца (*Markowitz model*) із заміною двокритеріальної задачі однокритеріальною з дробно-лінійним критерієм (максимум дохідності портфеля на одиницю ризику) [31]

Математичну постановку даних задач можна знайти в мережі Інтернет або джерелах [25–31]. Деякі із цих задач зручно розв'язувати методом штрафів або штрафних функцій [10, 24].

2. Змінюючи параметри метаевристик, дослідити їхній вплив на ефективність розв'язання задачі оптимізації, фіксувати час роботи програми (табл. 1.5). Для вибраного набору найкращих параметрів виконати експерименти не менше s = 10–20 разів. Зберегти результати запусків для подальшого аналізу. Знайти середнє квадратичне відхилення *RMSE* для кожного запуску за формулою $RMSE = \sqrt{(Y-y)^2 / s}$, де Y – теоретичне або найкраще відоме значення екстремуму, а y – результат роботи програми. Відобразити у звіті найгірше, середнє та найкраще значення екстремуму за s запусків програми, аналогічні дії виконати для часу виконання програми t.

Експерименти проводити, не використовуючи готові пакети та функції оптимізації. Але їх можна застосувати для перевірки результатів роботи: https://en.wikipedia.org/wiki/List_of_optimization_software.

Таблиця 1.5 – Результати експерименту для однієї мета евристики

Назва задачі	Найкращі параметри алгоритму	Результати обчислень $f(x^*)$ (найбільше, найменше, середнє значення функції)	Час пошуку оптимуму t, сек (найбільший, найменший, середній час)	*RMSE*	Кількість запусків s
	…	…	…	…	…

3. Повторити пункт 3 для всіх вибраних метаевристик. За допомогою *W*-критерію Уілкоксона або критерію суми рангів (*Wilcoxon rank-sum test*) перевірити однорідність / неоднорідність вибірок (результатів розрахунку значень оптимуму) [33] та порівняти ефективність роботи метаевристик на кожній функції.

4. Порівняти результати роботи вибраних метаевристик з відомими результатами із різних наукових праць. Зробити висновки.

Задача 3. Розробити багатошаровий перцептрон або згорткову нейромережу, які дозволять розв'язати задачу реалізації певної логічної функції від трьох змінних (табл. 1.6) [34–38]. Вихід логічної функції згенерувати випадковим чином. Гіперпараметри нейромережі налаштувати самостійно. Навчання штучної нейронної мережі (*artificial neural network training*) виконати з використанням вибраних метаевристик. Результати їхньої роботи можна порівняти з тими, які отримано класичними алгоритмами градієнтної оптимізації нейромереж, наприклад, методом адаптивної оцінки моментів (*adaptive moment estimation*, *ADAM*).

Таблиця 1.6 – Варіанти завдань

№	Функція
1	Кон’юнкція
2	Диз’юнкція
3	Імплікація
4	Еквівалентність
5	Нерівнозначність
6	Стрілка Пірса
7	Штих Шеффера

Розробити алгоритмічне забезпечення та перевірити результати роботи в програмних середовищах (*MatLab* та *OPTI Toolbox*, *MathCAD, Mathematica, Maple, Python Anaconda,* різноманітних програмних пакетах оптимізаці типу *CPLEX, APMonitor* тощо [60–65]). Отримати рішення заданих задач, представити результати роботи програми, зробити висновки.

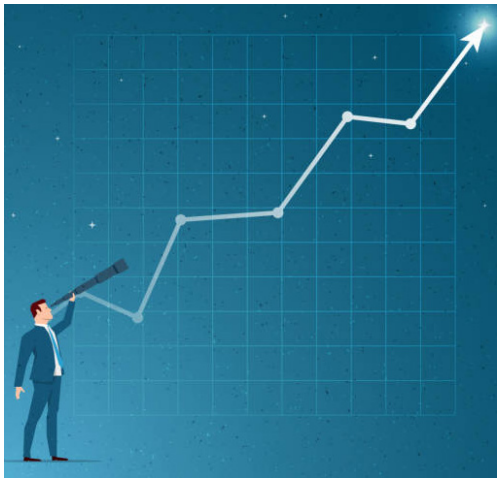
Контрольні запитання

1. Історична ретроспектива. Видатні вчені.
2. Поняття оптимізації. Математична постановка задачі оптимізації.
3. Екстремум функції.
4. Градієнт та антиградієнт.
5. Критерій Сільвестра. Власні значення матриці Гесе.
6. Матриця Гесе. Квадратична форма.
7. Алгоритм ідентифікації оптимуму (математичний аналіз).
8. Методи умовної і безумовної оптимізації.
9. Перетворення задачі умовної оптимізації в безумовну задачу.
10. Задачі умовної оптимізації. Метод штрафних функцій. Види штрафних функцій.
11. Класифікація методів безумовної оптимізації.
12. Особливості пошукових алгоритмів оптимізації багатовимірних функцій. Їх переваги та недоліки.
13. Особливості стохастичної оптимізації. Обчислювальний інтелект. Ройовий інтелект.
14. Метаевристики. Загальна метаевристична схема.
15. Метод Монте-Карло. Парадокс днів народження. Задача про голку Бюффона.
16. Стратегії формування початкового положення агентів: ковдра, дробовик, фокусування, комбінований варіант.
17. Методи контролю границь області пошуку.
18. Інтенсифікація (*intensification*) та диверсифікація (*diversification*) процесу пошуку екстремуму.
19. Проблема балансу між інтенсифікацією та диверсифікацією пошуку.
20. Критерії зупинки метаевристичного алгоритму.
21. Генератори випадкових чисел (*random numbers*).

22. Методи генерації чисел для випадкових блукань: рівномірний розподіл (*uniform distribution*), нормальний розподіл (*normal distribution*), польоти Леві (*Levy distribution*).
23. Алгоритм Мантенья-Янга-Деба для роботи з розподілом Леві.
24. Алгоритм сходження на вершину (*hill climbing*).
25. Кулеметне сходження або падіння (*shotgun hill climbing*).
26. Променевий пошук (*beam search*).
27. Алгоритм модельного загартування або імітації відпалу (*simulated annealing*).
28. Режим охолодження (*cooling*) та гасіння (*quenching*): лінійний закон, схеми Больцмана, Коші, Інгбера, Яо, Люка, Лунді-Месса, Янга-Као-Чанга-Морріса, комбіновані тощо.
29. Перенагрів (*re-heat*). Повторний відпал (*re-annealing*).
30. Алгоритм Лууса-Яакола (*Luus-Jaakola*): редукція та відновлення.
31. Еволюційні алгоритми.
32. Принципи природного відбору Дарвіна, наслідування Менделя та мутації де Фріза.
33. Генетичні (*genetic*) алгоритми.
34. Пропорційний вибір особин (*proportional selection*): колесо рулетки (*roulette wheel selection*), лінійне ранжування (*linear ranking*), сігма-відсікання (*sigma truncation*).
35. Вибір батьківських пар: панміксія, селективний, інбридінг, аутбридінг.
36. Селекція: витіснення, елітизм: (*mu+lambda*)-стратегія, (*mu, lambda*)-стратегія.
37. Меметичні (*memetic*) алгоритми. Гіперевристики.
38. Гібридизація метаевристик.
39. Алгоритм пошуку на основі поведінки амеби (*amoeba algorithm*).
40. Алгоритм оптимізації центральною силою (*central force optimization*).
41. Тасуючий алгоритм стрибаючих жаб (*shuffled frog-leaping algorithm*).
42. Алгоритм оптимізації на основі поведінки зозуль (*cuckoo search*).
43. Алгоритм мавпячого пошуку (*monkey search algorithm*).
44. Мавпячий алгоритм (*monkey algorithm*).
45. Алгоритм оптимізації на основі поведінки світляків (*firefly algorithm*).
46. Алгоритм оптимізації на основі поведінки мурашиної колонії (*ant colony optimization*).
47. Алгоритм штучної бджолиної колонії (*artificial bee colony algorithm*).
48. Алгоритм оптимізації на основі поведінки кажанів (*bat optimization algorithm*).
49. Алгоритм оптимізації на основі поведінки кошачої зграї (*cat swarm optimization*).
49. Алгоритм еволюції розуму (*mind evolutionary computation*).
50. Практичні задачі оптимізації. Навчання штучної нейронної мережі: алгоритм зворотного поширення похибки, нейромережеві оптимізатори.
51. Метаоптимізація метаевристик.

52. Паралелізація метаевристичних схем: мультистарт, острівна модель, коеволюція або ко-алгоритмізація.
53. Як залежить час оптимізації від розмірності задачі?
54. Чи завжди при збільшенні розмірності задачі збільшується час пошуку оптимуму?
55. Які фактори впливають на час розв’язання задачі оптимізації?
56. Практичне застосування метаевристичних методів оптимізації.
57. Як вибрати «найкращий» оптимізатор? Теорема про відсутність безкоштовних сніданків.
58. Особливості розв’язання задачі оптимізації в пакетах математичного моделювання та середовищах програмування.

ЛАБОРАТОРНА РОБОТА № 2



Тема: Метаевристичні алгоритми розв’язання задач дискретної глобальної оптимізації. Ройовий інтелект.

Мета: навчитися застосовувати метаевристичні алгоритми дискретної глобальної оптимізації та відповідне програмне забезпечення для пошуку розв’язку задачі в багатовимірному просторі альтернатив.

Теоретичні відомості: для детального розуміння матеріалу необхідно проаналізувати джерела [3, 4, 9–12, 14–23, 39–54], у яких подано теоретичні основи теорії глобальної оптимізації [10–12], ройового інтелекту, метаевристик [14–23] та проаналізовано відповідні задачі [39–54].

Завдання

Подати загальну математичну постановку задачі дискретної глобальної оптимізації. Проаналізувати та теоретично обґрунтувати:

- складність задачі оптимізації (класи *P* та *NP*);
- основи стохастичних алгоритмів оптимізації (метаевристик, популяційних, інтелектуальних алгоритмів, ройового інтелекту) (*swarm intelligence*):
  - стратегії формування початкового положення боїдів (агентів);
  - методи генерації перестановок (*permutations*);
  - критерії зупинки пошуку (*stopping criterions*);
  - мультистартова процедура (*multistart*);
  - паралелізація метаевристик (*metaheuristics parallelization*): послідовна, острівна, коеволюційна стратегія;
- математичні моделі задач:
  - комівояжера (*travelling salesman task*);
  - про покриття (*cover task*);
  - шідоку (*shidoku*) та судоку (*sudoku*);
  - про доповнення до *N* ферзів (*N-queens completion puzzle*);

- проектування інтерлівера для систем цифрового зв'язку (*interleaver design task*);
- про злодія (*travelling thief task*);
- китайського листоноші (*chinese postman task*);
- нью-йорського підмітальника (*New York sweeper task*);

– теорему Вульперта-МакКріді про відсутність безкоштовних сніданків (*No Free Lunch Theorem*).

Розв'язати наступні задачі:

Задача 1. Задача комівояжера [39, 40, 43, 47]. Тестові набори (кількість вузлових точок, їх координати) взяти з бібліотеки *TSPLIB* Г. Райнельта (http://comopt.ifi.uni-heidelberg.de/software/TSPLIB95/).

Задача 2. Лицарський граф (*knight's graph*) [40–43]. Чи можна шаховим конем, починаючи з довільної клітинки шахової дошки розміром *m* на *n*, обійти всі її поля і повернутись у вихідну точку, побувавши в кожній клітинці один раз? Згенерувати різні варіанти дошок і маршрути коня.

Задача 3. Судоку. Задача № 96 проекту Ейлер, яку можна знайти за посиланням: https://euler.jakumo.org/problems/view/96.html. Протестувати програму на декількох дуже складних судоку з кодовими іменами [49, 50]: *AI Escargot*, *Easter Monster*, *Red Dwarf*, *Fata Morgana*, *Cheese*, *Golden Nagget*, *Platinum Blonde* тощо.

Задача 4. Задача про доповнення до *N* ферзів [51]. Згенерувати позиції *M* ферзів та розмір дошки випадковим чином.

Задача 5. Проектування інтерлівера [52, 53]. Можна проводити пошук набору індексів *N* бінарних символів (перестановки) для кодової послідовності, яка:

– збільшує нормалізовану дисперсію (*normalized dispersion*);

– збільшує розсіювання або спред (*spread*);

– зменшує циклічну кореляційну суму (*cycle correlation sum*);

– збільшує нормалізовану дисперсію або спред на одиницю метрики циклічної кореляційної суми.

Для генерації перестановок можна використати інтерлівери із наукових робіт, випадковий інтерлівер або *S*-випадковий інтерлівер Дівсалара-Поллари.

Задача 6. Мультикритеріальна задача про злодія [54], яка поєднує задачі комівояжера та наповнення рюкзака (*knapsack task*). Тестові дані можна знайти за посиланнями:

– https://cs.adelaide.edu.au/~optlog/research/combinatorial.php;

– https://github.com/markuswagnergithub/TTPexact.

Задача 7. Задача китайського листоноші. Припустимо, що певний район міста обслуговується одним транспортним засобом, який виїжджає зі звалища, збирає сміття зі всіх вулиць району (з контейнерів), а потім повертається до початкового пункту. На поїздку накладене обмеження: потрібно використати мінімальну кількість пального (знайти мінімальний шлях)

та зібрати все сміття (об'ємом кузова для сміття знехувати). Випадковим чином згенерувати низку графів $G_i(V, E)$, який має порядок $V \geq 12$ та розмірність $E \geq 22$. Ребра графа представляють дороги, а вершини – їх перетин. Ваги ребер (відстань) задати випадковим чином.

Для виконання даної роботи застосувати 2–3 методи оптимізації з табл. 2.1 (вибір виконати, використавши 2–3 рази генератор випадкових чисел від 1 до k, де k – кількість методів). Вибір задач здійснити за вказівкою викладача.

Таблиця 2.1 – Стохастичні методи оптимізації

№	Задача	Метод (k = 25)
1	1, 2, 4, 5, 6, 7	*Алгоритм кулеметного сходження або падіння (shotgun hill climbing)*
2	1, 2, 4, 5, 6, 7	*Жадібний рандомізований адаптивний пошук (greedy randomized adaptive search procedure, GRASP)*
3	1, 2, 4	*Алгоритм Куна-Манкреса* (*Kuhn-Munkres algorithm*)
4	1, 2, 3, 4, 5, 6	*Метод гілок та меж (branch and bound) або алгоритм Літтла-Мурті-Суїні-Керолла (Little-Murty-Sweeney-Karel algorithm)*
5	1, 2	*Еластична мережа (elastic net)*
6	1, 2	*Нейронна мережа Кохонена (Kohonen Self-Organizing Maps)*
7	1, 2	*Алгоритм Ліна-Кернігана-Хельсгауна (Lin-Kernighan-Helsgaun, k-opt method)*
8	3	*Алгоритм X Д.Е. Кнута* (*Algorithm X*)
9	1, 2, 3, 4, 5, 6	*Генетичні алгоритми (genetic algorithms)*
10	1, 2, 3, 4, 5, 6	*Меметичні алгоритми (memetic algorithms)*
11	1, 2, 3, 4, 5, 6	*Алгоритм перемішаних стрибаючих жаб (shuffled frog-leaping)*
12	1, 2, 3, 4, 5, 6	*Алгоритм штучної бджолиної колонії бджолиної колонії (artificial bee colony algorithm)*
13	1, 2, 3, 4, 5, 6	*Оптимізація на основі поведінки котячої зграї (cat swarm optimization algorithm)*
14	1, 2, 3, 4, 5, 6	*Оптимізація на основі поведінки зграї сірих вовків (gray wolf opimizer)*
15	1, 2, 3, 4, 5, 6	*Зозулиний пошук (cuckoo search)*
16	1, 2, 3, 4, 5, 6	*Мавпячий пошук (monkey search)*
17	1, 2, 3, 4, 5, 6	*Алгоритм наслідування поведінки мурашиної колонії (ant colony optimization)*
18	1, 2, 3, 4, 5, 6	*Бактеріальна оптимізація (bacterial foraging optimization)*
19	1, 2, 3, 4, 5, 6	*Алгоритм штучних імунних систем (artificial immune system algorithm)*

Продовження таблиці 2.1

20	1, 2, 3, 4, 5, 6	*Імітація відпалу (simulated annealing)*
21	1, 2, 3, 4, 5, 6	*Алгоритм інтелектуальних крапель води (intelligent water drop)*
22	1, 2, 3, 4, 5, 6	*Стохастичний дифузійний пошук (stochastic diffusion search)*
23	1, 2, 3, 4, 5, 6	*Алгоритм еволюції розуму (mind evolutionary computation)*
24	1, 2, 4, 5, 6, 7	*Променевий пошук (beam search)*
25	7	*Комбінований метод Крістофідеса: алгоритм Дейкстри + алгоритм Куна-Манкреса + алгоритм Флері (Christofides algorithm for chinese postman task)*

Навести детальні алгоритми роботи методів оптимізації. Виконати на папері вручну низку ітерацій та показати результати. Необхідно змінювати параметри стохастичних алгоритмів та досліджувати їх вплив на ефективність розв'язання задачі оптимізації. Для вибраного набору найкращих параметрів виконати експерименти не менше s = 10–20 разів. Знайти середнє квадратичне відхилення *RMSE* для кожного запуску за формулою $RMSE = \sqrt{(Y - y)^2 / s}$, де Y – відоме значення екстремуму, а y – результат роботи програми. Відобразити у звіті найгірше, середнє та найкраще значення екстремуму за s запусків програми, аналогічні дії виконати для часу виконання програми t. Порівняти результати обчислень з відомим розв'язком, якщо є такі дані (табл. 2.2).

Таблиця 2.2 – Результати експерименту для однієї мета евристики

Назва задачі	Найкращі параметри алгоритму	Результати обчислень $f(x^*)$ (найбільше, найменше, середнє значення функції)	Час пошуку оптимуму t, сек (найбільший, найменший, середній час)	*RMSE*	Кількість запусків s
…	…	…	…	…	…

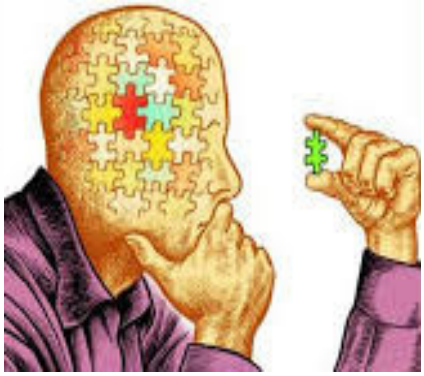
Розробити алгоритмічне забезпечення та перевірити результати роботи в програмних середовищах (*MatLab*, *MathCAD, Mathematica, Maple, Python Anaconda,* різноманітних програмних пакетах оптимізаці типу *CPLEX, Concorde* тощо [60–65]). Отримати рішення заданих задач, представити результати роботи програми, зробити висновки.

Контрольні запитання

1. Історична ретроспектива теорії графів. Видатні вчені.
2. Поняття оптимізації. Математична постановка задачі оптимізації.
3. Екстремум функції.
4. Стіна складності. Поняття *P* та *NP*-складності.
5. Основні поняття теорії графів.
6. Способи представлення графів.
7. Області застосування графових моделей.
8. Математична постановка та історична ретроспектива задач про покриття, судоку, про доповнення до *N* ферзів, проектування інтерлівера для систем цифрового зв'язку, про злодія, китайського листоноші та нью-йорського підмітальника.
9. Гамільтоновий цикл та шлях, їхній зв'язок. Теореми існування графів з такими циклами.
10. Ейлеровий цикл та шлях, їхній зв'язок. Теореми існування графів з такими циклами.
11. Еластична мережа для розв'язання задачі комівояжера.
12. Нейронна мережа Кохонена для розв'язання задачі комівояжера.
13. Алгоритм Куна-Манкреса.
14. Алгоритм Літтла-Мурті-Суїні-Керолла. Метод гілок та меж.
15. Алгоритм Ліна-Кернігана-Хельсгауна (*k-opt*). Приклад 2-opt евристики.
16. До якої задачі зводяться головоломки виду "намалювати, не відриваючи руки"?
17. Задача про кенігсберзькі мости. Алгоритми Флері та Хіергольцера.
18. Комбінований метод Крістофідеса для розв'язання задачі китайського листоноші.
19. Особливості представлення графових моделей у пакетах математичного моделювання та середовищах програмування.
20. Алгоритм Х Д.Е. Кнута для розв'язаня судоку.
21. Практичне застосування даних задач.
22. Метаевристики. Загальна метаевристична схема.
23. Метод Монте-Карло. Парадокс днів народження. Задача про голку Бюффона.
24. Особливості стохастичної оптимізації. Обчислювальний інтелект. Ройовий інтелект.
25. Стратегії формування початкового положення агентів: ковдра, дробовик, фокусування, комбінований варіант.
26. Критерії зупинки метаевристичного алгоритму.
27. Генератори випадкових чисел (*random numbers*).
28. Інтенсифікація (*intensification*) та диверсифікація (*diversification*) процесу пошуку екстремуму.
29. Проблема балансу між інтенсифікацією та диверсифікацією пошуку.
30. Метаоптимізація метаевристик.
31. Гібридизація метаевристик.

32. Паралелізація метаевристичних схем: мультистарт, острівна модель, коеволюція або ко-алгоритмізація.
33. Алгорит кулеметного сходження.
34. Жадібний рандомізований адаптивний пошук.
35. Генетичні алгоритми.
36. Меметичні алгоритми.
37. Алгоритми, які інспіровані живою та неживою природою. Історична ретроспектива. Етапи виконання. Збіжність.
38. Алгоритм перемішаних стрибаючих жаб.
39. Алгоритм штучної бджолиної колонії бджолиної колонії.
40. Оптимізація на основі поведінки котячої зграї.
41. Оптимізація на основі поведінки зграї сірих вовків.
42. Зозулиний пошук.
43. Мавпячий пошук.
44. Алгоритм наслідування поведінки мурашиної колонії.
45. Бактеріальна оптимізація.
46. Алгоритм штучних імунних систем.
47. Імітація відпалу.
48. Алгоритм інтелектуальних крапель води.
49. Стохастичний дифузійний пошук.
50. Алгоритм еволюції розуму.
51. Променевий пошук.
52. Як залежить час оптимізації від розмірності задачі?
53. Які фактори впливають на час розв’язання задачі оптимізації?
54. Теорема про відсутність безкоштовних сніданків.
55. Особливості розв’язання задачі оптимізації в пакетах математичного моделювання та середовищах програмування.

ЛАБОРАТОРНА РОБОТА № 3



Тема: Задача ідентифікації функціональної залежності з використанням нечіткої логіки.

Мета: навчитися застосовувати апарат нечіткої логіки, а також проектувати системи нечіткого логічного виведення з використанням відповідного програмного забезпечення.

Теоретичні відомості: для детального розуміння матеріалу необхідно проаналізувати джерела [3, 9, 15–23, 47, 55–59], у яких подано теоретичні основи навчання нечітких баз знань та проектування відповідних систем [47, 55–59], а також ройового інтелекту для розв'язання задачі оптимізації на етапі параметричної ідентифікації [15–23].

Завдання

Подати загальну математичну постановку задачі навчання нечіткої бази знань, а також виконання етапів структурної та параметричної ідентифікації об'єкта дослідження. Проаналізувати та теоретично обґрунтувати:

- основи аналізу вибірки даних (розбиття на навчальну, валідаційну та тестову вибірки);
- етапи проектування нечіткої бази знань (*fuzzy knowledge base*);
- алгоритми нечіткого логічного висновку (*fuzzy inference*):
  - Мамдані (*Mamdani*);
  - Ларсена (*Larsen*);
  - Такагі-Сугено-Канга (*Takagi-Sugeno-Kang*);
- методи дефаззіфікації (*defuzzification*);
- особливості кривих навчання:
  - залежність нев'язки *RMSE* від часу навчання *t*;
  - залежність *RMSE* від кількості правил *N* в базі знань.
- складність задачі оптимізації (класи *P* та *NP*);
- основи ройового інтелекту (*swarm intelligence*).

Розв'язати такі задачі:

Задача 1. Експериментально встановити залежність якості (точності) нечіткого логічного висновку від кількості правил в базі знань Мамдані або Ларсена. Визначити оптимальну кількість правил [56, 58, 59]:

1. Проектується нечітка база знань, що моделює еталонну залежність $z = f(x, y)$ з табл. 3.1, у якій вказано потужності терм-множин вхідних (x, y) та вихідної (z) змінних нечіткої бази знань. Варіант вибрати за формулою $N = (L + [\text{остання цифра року}]) \bmod 20$, де L – варіант студента.

2. Побудувати тривимірний графік заданої аналітичної залежності.

3. За графічним зображенням залежності $z = f(x, y)$ згенерувати повну базу знань з n нечітких правил.

Таблиця 3.1 – Еталонні залежності

№	Функція	Кількість термів			Діапазон	
		x	y	z	x	y
1	$z = (x^2 + y)/(0{,}1 \cdot e^x + y)$	4	3	5	[-5, 5]	[3, 7]
2	$z = (x^2 - y^3)/(\sqrt{2 \cdot x} + y)$	4	4	5	[0, 4]	[2, 6]
3	$z = \sin(x)/y$	4	3	5	[0, 5]	[1, 10]
4	$z = (x + x \cdot y - \sqrt{y})/(x + y)$	4	4	5	[1, 5]	[0.5, 2]
5	$z = (x^2 + y^3)/(x + y)$	4	3	4	[0, 20]	[6, 10]
6	$z = (x - 4) \cdot (y - 5)^3$	5	4	4	[20, 80]	[3, 12]
7	$z = (x + y^3)/(2 \cdot \sqrt{x \cdot y})$	3	4	5	[2, 7]	[5, 22]
8	$z = x^3 \cdot \sin(0{,}2 \cdot y)/(x + y^2)$	3	4	5	[100, 150]	[50, 125]
9	$z = x \cdot \sqrt{y}$	5	4	4	[2,22]	[3, 16]
10	$z = (x + y)/(\sqrt{x} + y^2 - 1)$	4	5	4	[0.6, 2]	[1, 3]
11	$z = x^4 \cdot \sin(y) - \sqrt{y}$	4	4	5	[3, 6]	[5, 10]
12	$z = 0{,}01 \cdot x^2 \cdot tg(y)$	5	4	4	[- 20, 25]	[-1, 1.3]
13	$z = (x^2 + \sqrt{y})/e^{x \cdot y}$	4	4	5	[-4, -3]	[0, 1]
14	$z = 7 \cdot \sqrt{x} - 6 \cdot x \cdot y$	4	3	5	[30, 50]	[50, 100]
15	$z = \sqrt{x^2 + y^2}/(x + y)$	3	4	5	[2, 4]	[3, 5]
16	$z = (x^2 + \sin(0{,}1 \cdot y))/(\cos(x) + y)$	5	4	4	[1, 5]	[3, 7]
17	$z = \sin(x) \cdot \cos(2 \cdot y)$	4	5	4	[1, 2]	[1, 2]
18	$y = (x - y) \cdot (x^2 + x \cdot y + y^2)$	4	5	4	[0.1, 2]	[0, 1.5]
19	$z = (\sqrt{x} + y) \cdot \sin(x + y)$	4	3	5	[1, 4]	[0.5, 2.5]
20	$y = x \cdot y \cdot \sqrt{x^2 + y^2}$	4	4	5	[5, 20]	[-10, 10]

4. Згенерувати тестову вибірку обсягом 100 точок.

5. Побудувати 3 криві навчання у формі залежності нев'язки *RMSE* від кількості правил нечіткої бази знань $n = \overline{1, n_{\max}}$. Порядок додавання правил у базу знань має бути унікальним для кожної кривої навчання. Провести узагальнення результатів експериментів, побудувавши криві навчання $RMSE_{min}$, $RMSE_{max}$ та $RMSE_{mean}$ від *n*.

Примітка. Недоцільно застосовувати громіздкі конструкції. Краще записати до нечіткої бази знань усі можливі правила та активувати їх за допомогою вагових коефіцієнтів. Нульове значення вагового коефіцієнта відповідає виключенню відповідного правила з бази знань, а одиничне – включенню. Для повного перебору правил нечіткої бази знань можна скористатись функцією, яка визначає можливу кількість комбінацій (або використати трикутник Паскаля). Якщо кількість правил-кандидатів дорівнює $n_{\max}$, тоді можна згенерувати ($2^{n\max} - 1$) унікальних баз знань.

6. Звести отримані нев'язки на один графік та провести апроксимацію (наприклад, квадратичну) для експериментальних даних. Знайти мінімальне значення функції $RMSE = f(n)$.

7. Порівняти точність (*RMSE*), компактність (рівень наповненості правилами) та інтерпретабельність (простота сприйняття бази знань) початкової та найкращої бази знань.

Задача 2. Експериментально встановити залежність якості (точності) ідентифікації від обсягу навчальної вибірки для нечіткої бази знань Такагі-Сугено-Канга [56, 58]:

1. Побудувати тривимірний графік заданої аналітичної залежності. Зобразити експериментальні дані у формі однофакторних залежностей $z = f(x)$ та $z = f(y)$.

2. Сформувати навчальну та тестову вибірки обсягом 100 пар (вхід – вихід). Перевірити їх репрезентативність (однорідність даних).

Примітка. Для перевірки репрезентативності вибірок розрахувати математичне сподівання (*mean*) та дисперсію (*std*) даних за кожною змінною. Якщо результати розрахунків вказують на приблизно однакові значення цих показників для навчальної та тестової вибірок, то можна вважати їх репрезентативними.

3. З навчальної вибірки сформувати 10 підвибірок обсягом $M = \overline{10,100}$ рядків (збільшувати обсяг на 10 рядків). Підвибірка більшого розміру має включати в себе усі елементи підвибірки меншого розміру.

4. Використовуючи тривимірний графік згенерувати правила нечіткої бази знань Такагі-Сугено-Канга. Коефіцієнти в консеквентах правил задати, враховуючи свої спостереження. Кожен терм повинен бути хоча б в одному правилі. Кількість правил в базі знань можна встановити довільно від 4 до *n*, або в 2-3 рази меншою, ніж у найкращої бази знань із першої задачі. Також відобразити у звіті параметри функцій приналежності.

5. Представити структуру нейронечіткої системи з редактору *anfisedit*.

6. Навчити нечітку базу знань Такагі-Сугено-Канга на навчальних вибірках *M* (алгоритм гратчастого розбиття, *grid partition*). Тривалість навчання контролювати за допомогою обчислення помилки на тестовій вибірці, що можна зробити з використанням функції *anfis*, а саме *anfisOptions*. Відобразити у звіті приклад роботи функції *anfis* з командного вікна *MatLab* для вибірки на 100 пар, а також показати отриману залежність *RMSE* від кількості ітерацій *I* на навчальній та тестовій вибірках.

7. Знайти експериментальну графічну залежність *RMSE* на тестовій вибірці від обсягу *M* навчальної вибірки. Порівняти *RMSE* на навчальній та тестовій вибірках.

8. Вивести найкращу нечітку базу знань Такагі-Сугено-Канга. Навести обсяг навчальної вибірки та *RMSE* для неї. Відобразити вид та параметри її функцій приналежності, а також коефіцієнти консеквентів правил.

9. Порівняти точність (*RMSE*), компактність (рівень наповненості правилами) та інтерпретабельність (простота сприйняття бази знань) початкової та найкращої бази знань Такагі-Сугено-Канга, а також найкращої бази знань Мамдані (або Ларсена), яку отримано в першій задачі.

10. Знайти розподіл *RMSE* для не менше *k* наборів навчальних вибірок та побудувати граничні (для $RMSE_{max}$ та $RMSE_{min}$), а також усереднену (для $RMSE_{mean}$) криві навчання.

Примітка. Виконати експерименти, наприклад, $k = 10$ разів.

Задача 3. Задача оптимізації параметрів нечіткої бази знань Мамдані, Ларсена, Такагі-Сугено-Канга: знайти такі параметри функцій приналежності $C = (a, b, c, d, ...)$ та / або ваги оптимальної кількості правил $W = (w_1, ..., w_N)$, $w_i \in [0,1]$, тобто $P = C \cup W$, які забезпечують мінімальне відхилення $RMSE = f(P)$ між експериментальними даними та результатами нечіткого логічного висновку [21].

Примітка. Використати декілька метаевристичних алгоритмів з першої лабораторної роботи (табл. 1.2). Необхідно змінювати параметри алгоритмів та досліджувати їх вплив на ефективність розв'язання задачі оптимізації. Для вибраного набору параметрів виконати експерименти не менше $s = 50$ разів. Визначити найкраще значення *RMSE* для нечіткої бази знань за *s* запусків програми.

Розробити алгоритмічне забезпечення, перевірити результати роботи в програмному середовищі *MatLab* (пакет *Fuzzy Logic toolbox*) [60, 61]. Отримати рішення заданих задач, представити результати роботи програми, зробити висновки.

Контрольні запитання

1. Історична ретроспектива. Видатні вчені. Множини та операції над ними. Діаграми Ейлера-Венна.
2. Типи логік. Прості та складні висловлювання. Логічні операції. Логічні функції.
3. Закони і тотожності алгебри логіки. Еквівалентні перетворення формул алгебри логіки.
4. Наведіть приклади нечітких множин.
5. Наведіть приклади лінгвістичних змінних.
6. Нечітка логіка. Ідеї Л.А. Заде. Функції приналежності.
7. Лінгвістичні змінні. Нечіткі лінгвістичні висловлення.
8. Нечіткі логічні операції.
9. Нечітка арифметика.
10. Різниця між логічним виведенням та нечітким логічним виведенням?
11. Нечіткі бази знань. Наведіть приклад правила з нечіткої бази знань.
12. Машина нечіткого логічного висновку. Базова архітектура систем нечіткого виведення.
13. Основні етапи нечіткого виведення.
14. Методи дефаззіфікації.
15. Етапи проектування нечіткої бази знань (структурний та параметричний).
16. Алгоритми нечіткого логічного висновку Мамдані, Ларсена, Такагі-Сугено-Канга.
17. Ідентифікація функціональної залежності нечіткою базою знань.
18. Математична постановка задачі навчання нечіткої бази знань.
19. Основи аналізу вибірки даних. Недонавчання та перенавчання.
20. Навчальна, валідаційна та тестова вибірки.
21. Репрезентативність вибірок даних.
22. Що таке “плато насичення” кривої навчання ?
23. Задача вибору оптимальної кількості правил для нечіткої бази знань. Її складність.
24. Чому повна нечітка база знань не дозволяє отримати нульову нев’язку ?
25. Чи доцільно застосовувати для розв’язання даної задачі повний перебір комбінацій правил?
26. Залежність тривалості експериментів від кількості правил та довжини вибірки (кількості точок).
27. Залежність точності ідентифікації від обсягу навчальної вибірки для нечіткої бази знань Такагі-Сугено-Канга.
28. Особливості застосування стохастичної оптимізації для пошуку оптимальних параметрів нечіткої бази знань. Ройовий інтелект.
29. Практичне застосування нечіткого логічного виведення. Експертні системи.
30. Особливості проектування систем нечіткого логічного виведення в пакетах математичного моделювання та середовищах програмування.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА

1. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки та техніки. Структура та правила оформлювання / В. Земцева, Ю. Поліщук, Р. Санченко, Л. Шрамко, А. Ямчук. — К. : ДП Український інститут науково-технічної і економічної інформації, 2016. — 31 с.
2. ДСТУ ГОСТ 7.1:2006. Бібліографічний запис, бібліографічний опис. Загальні вимоги та правила складання / О. К. Галевич, І. М. Штогрин. — Львів, 2008. — 20 с.
3. Henderson H. Encyclopedia of Computer Science and Technology / H. Henderson. — New-York, 2003. — 593 p.
4. Дасгупта С. Алгоритмы / С. Дасгупта, Х. Пападимитриу, У. Вазирани. — М. : МЦНМО, 2014. — 320 с.
5. Кормен Т. Х. Алгоритмы: построение и анализ / Кормен Т. Х. — М. : Вильямс, 2007. — 1296 с.
6. Левитин А. В. Алгоритмы: введение в разработку и анализ / А. В. Левитин. — М. : Вильямс, 2006. – 527 с.
7. Скиена С. Алгоритмы. Руководство по разработке / С. Скиена. — СПб. : БХВ-Петербург, 2011. — 720 с.
8. Кнут Д. Э. Искусство программирования (The Art of Computer Programming): в 4-х томах / Кнут Д. Э. — М. : Вильямс, 2001–2013. — 3344 с.
9. Іванов Ю. Ю. Теорія алгоритмів: лекції, алгоритми та задачі [Електронний ресурс]/ Іванов Ю. Ю. — Вінниця, 2021. — 137 с. — Режим доступу : https://iq.vntu.edu.ua/method/read_url.php?id=215210&tbl_num=2&card_id=23169.
10. Пантелеев А. В. Методы оптимизации в примерах и задачах / А. В. Пантелеев, Т. А. Летова. — М. : 2005. — 544 с.
11. Жалдак М. І. Основи теорії і методів оптимізації : навчальний посібник / М. І. Жалдак, Ю. В. Триус. — Черкаси : Брама-Україна, 2005. — 608 с.

12. Mathews J.H. Numerical Methods Using MATLAB / J.H. Mathews, K.D. Fink. — Prentice Hall, 1999. — 336 p.
13. Chapra S.C. Numerical Methods for Engineers / S.C. Chapra, R.P. Canale. — New-York: McGraw-Hill, 2010. — 994 c.
14. Соболь И.М. Метод Монте-Карло / И.М. Соболь. — М.: Наука, 1985. — 240 с.
15. Handbook of Metaheuristics / F. Glover, G.A. Kochenberger and others. — Kluwer Academic Publishers, 2003. — 570 p.
16. Карпенко А. П. Популяционные алгоритмы глобальной поисковой оптимизации. Обзор новых и малоизвестных алгоритмов / А. П. Карпенко // Информационные технологии. — М. : Новые технологии, 2012. — № 7. — 32 с.
17. Субботін С. О. Неітеративні, еволюційні та мультиагентні методи синтезу нечіткологічних і нейромережних моделей / С. О. Субботін, А. О. Олійник, О. О. Олійник. — Запоріжжя : ЗНТУ, 2009. — 375 с.
18. Щербина О. А. Метаэвристические алгоритмы для задач комбинаторной оптимизации / О. А. Щербина // Таврический Вестник Информатики и Математики. — 2014. — Випуск 1 (24). — С. 56–72.
19. Эволюционные методы моделирования и оптимизации сложных систем / Е. С. Семенкин, М. Н. Жукова, В. Г. Жуков, И. А. Панфилов, В. В. Тынченко. — Красноярск : СФУ, 2007. — 310 с.
20. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский. — М. : Горячая линия-Телеком, 2004. — 452 с.
21. Ходашинский И. А. Алгоритмы муравьиной и пчелиной колонии для обучения нечетких систем / И. А. Ходашинский, И. В. Горбунов, П. А. Дудин // Доклады ТУСУРа. — 2009. — № 2 (20). — С. 157–161.
22. Delahaye D. Simulated Annealing: From Basics to Applications / D. Delahaye, S. Chaimatanan, M. Mongeau // International Series in Operations Research & Management Science. — 2018. — 43 p.
23. Yang X.S. Bat Algorithm for Multi-Objective Optimization / X.S. Yang // International Journal of Bio-Inspired Computation. — 2011. — Vol. 3. — P. 267-274.
24. Yeniay O. Penalty Function Methods for Constrained Optimization with Genetic Algorithms / O. Yeniay // Mathematical and computational applications. — V. 10. — №1. — 2005. — P. 45-57.
25. Пантелеев А. В. Методы «роевого» интеллекта в задачах оптимизации параметров технических систем / А. В. Пантелеев, М. Д. Евдокимова. — Научный вестник МГТУ. — М. : МГТУ, 2017. — №2. — С. 6–15.
26. Пантелеев А.В. Решение задачи о переводе спутника между орбитами с помощью гибридного меметического алгоритма / А. В. Пантелеев, В. А. Письменная // Научный вестник МГТУ. — М. : МГТУ, 2014. — № 207. — С. 25–32.

27. Floudas C.A. Encyclopedia of Optimization / C.A. Floudas, P.M. Pardalos and others. — Springer, 2009. — 441 p.
28. Mine Blast Algorithm: A New Population Based Algorithm for Solving Constrained Engineering Optimization Problems / A. Sadollaha, A. Bahreininejada, H. Eskandar, M. Hamdi // Applied Soft Computing. — № 13. — 2013. — P. 2592–2612.
29. Optimization of a Speed Reducer Using Deterministic Techniques / M.H. Lin, J.-F. Tsai, N.-Z. Hu, S.-C. Chang // Mathematical Problems in Engineering. — Article ID 419043. — 2013. — 7 p.
30. Golden Eagle Optimizer: A Nature-Inspired Metaheuristic Algorithm / A. Mohammadi-Balani, M. Dehghan Nayeri, A. Azar, M. Taghizadeh-Yazdi // Computers & Industrial Engineering. — V. 152. — 2021. — 59 p.
31. Портфельная теория Марковица [Электронный ресурс]. — Режим доступа : https://finzz.ru/formirovanie-investicionnogo-portfelya-markovica-v-excel.html.
32. Jamil M. A Literature Survey of Benchmark Functions For Global Optimization Problems / M. Jamil, Xin-She Yang // International Journal of Mathematical Modelling and Numerical Optimization. — 2013. — V. 4. — P. 150–194.
33. Большев Л. Н. Таблицы математической статистики / Л. Н. Большев, Н. В. Смирнов. — М. : Наука, 1983. — 416 с.
34. Люгер Д. Ф. Искусственный интеллект. Стратегии и методы решения сложных проблем / Д. Ф. Люгер. — М. : Вильямс, 2003. — 864 с.
35. Fausett L. Fundamentals of Neural Networks: Architectures, Algorithms, and Applications / L. Fausett. — USA : Prentice-Hall Inc, 1994. — 476 p.
36. Портал искусственного интеллекта [Электронный ресурс] / Нейронные сети. — Режим доступа : http://neuronus.com/.
37. Deep Learning Tutorial. — Montreal (Canada): LISA Lab, University of Montreal, 2015. — 173 p.
38. Бринк Х. Машинное обучение / Х. Бринк, Дж. Ричардс, М. Феверолф. — СПб. : Питер, 2017. — 336 с.
39. Мудров В. И. Задача о коммивояжёре / В. И. Мудров. — М. : Знание, 1969. — 62 с.
40. Кирсанов М.Н. Графы в Maple / М. Н. Кирсанов. — М. : Физматлит, 2007. — 168 с.
41. Касьянов В.Н. Графы в программировании: обработка, визуализация и применение / В. Н. Касьянов, В. А. Евстигнеев. — СПб. : БХВ-Петербург, 2003. — 1104 с.
42. Кристофидес Н. Теория графов. Алгоритмический поход / Кристофидес Н. — М. : Мир, 1978. — 360 с.
43. Ерзин А. И. Задачи маршрутизации / А. И. Ерзин, Ю. А. Кочетов. — Новосибирск: РИЦ НГУ, 2014. — 95 с.

44. Dorigo M. Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem // IEEE Transactions of Evolutionary Computing / M. Dorigo, L.M. Gambardella. — 1997. — V. 1 (1). — P. 53-66.
45. Штовба С. Д. Муравьиные алгоритмы / С. Д. Штовба // Exponenta Pro. Математика в приложениях. — 2003. — № 4. — С. 70–75.
46. Dorigo M. Swarm Intelligence, Ant Algorithms and Ant Colony Optimization // Reader for CEU Summer University Course «Complex System» / M. Dorigo. — Budapest: Central European University, 2001. — P. 1-38.
47. Джонс М. Т. Программирование искусственного интеллекта в приложениях / Джонс М. Т. — М. : ДМК Пресс, 2011. — 312 с.
48. Knuth D. E. Dancing Links [Web resource] / D. E. Knuth. — Stanford University. — 26 p. — Access mode : https://www.ocf.berkeley.edu/~jchu/publicportal/sudoku/0011047.pdf.
49. Mantere T. Solving, Rating and Generating Sudoku Puzzles with GA / T. Mantere, J. Koljonen // IEEE Congress on Evolutionary Computation. — 2007. — P. 1382–1389.
50. Viksten H. Performance and Scalability of Sudoku Solvers / H. Viksten, V. Mattsson. — Stockholm : KTH Computer Science and Communication, 2013. — 54 p.
51. Gent I. P. Complexity of N-Queens Completion / I. P. Gent, C. Jefferson, P. Nightingale // Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence (IJCAI-18). — 2018. — P. 5608–5611.
52. Баринов А. Ю. Перемежение в канальном кодировании: свойства, структура, специфика применения [Электронный ресурс] / А. Ю. Баринов // Журнал радиоэлектроники. — 2019. — № 1. — 32 с. — Режим доступа : http://jre.cplire.ru/jre/jan19/13/text.pdf.
53. Popovski P. Design of Flexible-Length S-Random Interleaver for Turbo Codes / P. Popovski, L. Kocarev, Aleksandar Risteski // IEEE Communications Letters. — V. 8. — № 7. — 2004. — P. 461–463.
54. Bonyadi M. R. The Travelling Thief Problem: the First Step in the Transition From Theoretical Problems to Realistic Problems / M.R. Bonyadi, Z. Michalewicz, L. Barone // IEEE Congress on Evolutionary Computation. — Cancun (Mexico), 2013. — 8 p.
55. Штовба С. Д. Проектирование нечетких систем средствами MATLAB / С. Д. Штовба. — М.: Горячая линия. — Телеком, 2007. — 288 с.
56. Штовба С. Д. Інтелектуальні технології ідентифікації залежностей. Лабораторний практикум : електронний навчальний посібник / С. Д. Штовба, В. В. Мазуренко. — Вінниця : ВНТУ, 2014. — 113 с.
57. Асаи К. Прикладные нечеткие системы : под ред. Т. Тэрано, К. Асаи, М. Сугэно / К. Асаи, Д. Ватада, С. Иваи. — М. : Мир, 1993. — 184 с.

58. Штовба С. Д. Идентификация нелинейных зависимостей с помощью нечеткого логического вывода в системе MATLAB / С. Д. Штовба // Математика в приложениях. — 2003. — №2 (2). — С. 9–15.
59. Штовба С. Д. Дослідження навчання компактних нечітких баз знань типу Мамдані / С. Д. Штовба, В. В. Мазуренко // Штучний інтелект. — Донецьк, 2011. — № 4 — С. 521–529.
60. Ануфриев И. Е. MATLAB 7 / И. Е. Ануфриев, А. Б. Смирнов, Е. Н. Смирнова. — СПб. : БХВ-Петербург, 2005. — 1104 с.
61. MATLAB Programming Fundamentals [Web Resource]. – Massachusetts: MathWorks, 2021. — Access mode : https://www.mathworks.com/help/ pdf_doc/matlab/matlab_prog.pdf. — 1404 p.
62. Лутц М. Программирование на Python : 4-е издание / М. Лутц. — СПб. : Символ-Плюс, 2010. — 1280 с.
63. Центр документации Wolfram Mathematica [Web Resource] / Wolfram Language & System. — Access mode : http://reference.wolfram.com/ language/.
64. Numerical Recipes in C: The Art of Scientific Computing / W. H. Press, S. A. Teukolsky, W. T. Vetterling, B. P. Flannery. — Cambridge: Cambridge University Press, 2002. — 949 p.
65. Макаров Е. Г. Инженерные расчеты в Mathcad 15. Учебный курс / Макаров Е. Г. — СПб. : Питер, 2011. — 400 с.

ОЦІНЮВАННЯ СТУДЕНТАМИ МАТЕРІАЛІВ КУРСУ ТА РІВНЯ ВИКЛАДАННЯ

Після закінчення курсу необхідно дати відповіді на задані запитання, вписавши свої коментарі.

1. Наскільки Ви задоволені змістом дисципліни в цілому? Пропозиції до наповнення змісту курсу та критика.

Коментар

_____
_____
_____
_____

2. Як Ви оцінюєте якість викладання навчального матеріалу викладачем? Що б Ви запропонували змінити в методичному і змістовному плані для вдосконалення викладання цієї дисципліни?

Коментар

_____
_____
_____
_____

3. Що Ви вважаєте найскладнішим під час вивчення курсу (тема, задача, метод тощо)?

Коментар

_____
_____
_____
_____

4. Чи вважаєте Ви, що отримані знання будуть корисними у подальшому під час навчання / роботи? Що було найбільш корисним з точки зору подальшого навчання та / або застосування у практичній діяльності?

Коментар

_____
_____
_____
_____

Електронне навчальне видання
комбінованого використання
Можна використовувати в локальному та мережному режимах

Методичні вказівки до виконання лабораторних та практичних робіт з дисципліни «Математична логіка та теорія алгоритмів» для студентів спеціальності 126 – «Інформаційні системи та технології»

Укладач *Іванов Юрій Юрійович*

Рукопис оформив *Ю. Іванов*

Редактор *О. Ткачук*

Оригінал-макет виготовив *Г. Багдасар'ян*

Підписано до видання 07.07.2021 р.
Гарнітура Times New Roman.
Зам. № Р2021-016.

Видавець та виготовлювач
Вінницький національний технічний університет,
інформаційний редакційно-видавничий центр.
ВНТУ, ГНК, к. 114.
Хмельницьке шосе, 95,
м. Вінниця, 21021.
Тел. (0432) 65-18-06.
press.vntu.edu.ua;
Email: irvc.vntu@gmail.com.
Свідоцтво суб'єкта видавничої справи
серія ДК № 3516 від 01.07.2009 р.