МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ МІСЬКОГО ГОСПОДАРСТВА імені О. М. БЕКЕТОВА

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до виконання практичних робіт
із навчальної дисципліни

«ІНТЕЛЕКТУАЛЬНІ УПРАВЛЯЮЧІ СИСТЕМИ І ТЕХНОЛОГІЇ»

(для здобувачів другого (магістерського) рівня вищої освіти денної, заочної і дистанційної форм навчання зі спеціальності 122 – Комп'ютерні науки)

Харків
ХНУМГ ім. О. М. Бекетова
2025

Методичні рекомендації до виконання практичних робіт із навчальної дисципліни «Інтелектуальні управляючі системи і технології» (для здобувачів другого (магістерського) рівня вищої освіти денної, заочної і дистанційної форм навчання зі спеціальності 122 – Комп'ютерні науки) / Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова ; уклад. Н. Д. Сізова. – Харків : ХНУМГ ім. О. М. Бекетова, 2025. – 69 с.

Укладач д-р фіз.-мат. наук, проф. Н. Д. Сізова

Рецензент

М. Ю. **Карпенко**, кандидат технічних наук, доцент кафедри комп'ютерних наук та інформаційних технологій Харківського національного університету міського господарства імені О. М. Бекетова

Рекомендовано кафедрою комп'ютерних наук та інформаційних технологій, протокол № 1 від 29 серпня 2024 р.

ЗМІСТ

ВСТУП

Сучасні технології автоматизації та управління стрімко розвиваються, що зумовлює необхідність використання інтелектуальних систем у різних сферах діяльності. У зв'язку зі швидким розвитком технологій, автоматизації та штучного інтелекту зростає потреба у висококваліфікованих спеціалістах, здатних проєктувати, аналізувати та впроваджувати інтелектуальні управляючі системи в різних галузях.

Курс «Інтелектуальні управляючі системи і технології» орієнтований на вивчення сучасних методів управління, алгоритмів машинного навчання, адаптивних та нейромережевих систем управління, автоматизованого ухвалення рішень у складних динамічних системах, а також їхнього практичного застосування в промисловості, транспорті, медицині та інших сферах.

Мета вивчення курсу − формування у студентів глибоких теоретичних знань і практичних навичок, необхідних для розробки інтелектуальних управляючих систем, що здатні самостійно ухвалювати рішення, адаптуватися до змінюваних умов та оптимізувати технологічні процеси.

Основні завдання курсу:

- ознайомлення з принципами побудови та функціонування інтелектуальних управляючих систем;
- вивчення алгоритмів штучного інтелекту та їхнього використання в системах управління;
- аналіз методів адаптивного та нейромережевого управління;
- розгляд сучасних технологій автоматизації та їхнє впровадження у промислові процеси;
- практичне застосування отриманих знань у розробці реальних систем управління.

Практичні роботи, що входять до складу курсу, мають на меті формування у студентів компетентностей, пов'язаних із розробкою, аналізом та впровадженням інтелектуальних управляючих систем. Вони дозволяють отримати практичні навички роботи з алгоритмами машинного навчання, нечіткої логіки, нейромережевих технологій, експертних систем та методів оптимізації. У процесі виконання практичних завдань студенти ознайомляться з програмними засобами для моделювання та реалізації інтелектуальних алгоритмів, а також з їхнім застосуванням у реальних управлінських завданнях. Це сприятиме розвитку професійних навичок, необхідних для вирішення сучасних інженерних проблем у сфері автоматизованих систем управління.

Практичні роботи є невід'ємною складовою навчального процесу, що допомагає глибше зрозуміти принципи побудови інтелектуальних управляючих систем і технологій та підготуватися до їхнього ефективного використання у професійній діяльності. Це видання призначено для вивчення практичних питань з курсу «Інтелектуальні управляючі системи і технології» призначено для здобувачів другого (магістерського) рівня вищої освіти денної, заочної і дистанційної форм навчання зі спеціальності 122 – Комп'ютерні науки.

ПРАКТИЧНА РОБОТА № 1

СИСТЕМНИЙ АНАЛІЗ ІНТЕЛЕКТУАЛЬНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Мета – зробити аналіз інтелектуальної інформаційної системи і етапи створення для її функціонування.

Завдання – обрати відповідно до варіанта одну систему і навести етапи її функціонування.

Варіанти для виконання практичної роботи

1. Система оцінки ризиків порушення конфіденційності інформації.
2. Система проєктування комп'ютерної мережі підприємства.
3. Система захисту даних на основі рольової моделі.
4. Система моніторингу бездротової локальної мережі.
5. Система захисту від несанкціонованого доступу до інформаційно-телекомунікаційних систем.
6. Оцінка ризику інформаційної безпеки.
7. Система захисту розумного будинку.
8. Система тестування знань студентів.
9. Система захисту комерційної інформації.
10. Система оцінки рейтингу закладу вищої освіти.
11. Система поставки ресурсів.
12. Система оцінки ринкової вартості нерухомості.
13. Система календарного планування виконання робіт.
14. Система оцінки рейтингу викладача.
15. Система оцінки програмного продукту.
16. Система відбору персоналу.
17. Система захисту програмного забезпечення
18. Система управління доступом користувача до інформаційних систем.
19. Система ухвалення рішень оцінки конкурентоспроможності продукції підприємства.
20. Система ухвалення рішень оцінки основних напрямків розвитку будівельного підприємства.

Загальні відомості

Системний аналіз є ключовим підходом до проєктування, розробки та впровадження інтелектуальних інформаційних систем (далі – ІІС). Він забезпечує всебічний підхід до вивчення складних систем та дозволяє визначити їх структуру, функції та взаємозв'язки [1].

Системний аналіз ІІС передбачає використання таких етапів:

1. Визначення цілей системи.
2. Моделювання процесів.
3. Аналіз даних.
4. Розробка архітектури.
5. Аналіз алгоритмів.
6. Оптимізація процесів.
7. Впровадження технологій штучного інтелекту.
8. Оцінка ризиків.
9. Системний моніторинг і адаптація.

Елементи ІІС − це інформаційні об’єкти зберігання (інформаційним елементом), логічно однорідні одиниці інформації, для зберігання яких достатньо одного запису, наприклад, інформаційні об’єкти зберігання для БД системи.

Для проведення функціонального аналізу необхідно дати відповідь на такі питання:

- визначення підсистем системи;
- навколишнє середовище;
- ціль (мета) та призначення системи і підсистем;
- входи, ресурси, затрати;
- вихід, результати, прибуток;
- рівень робот, програми, підпрограми;
- виконавці, ОПР (особа, що ухвалює рішення) і керівники;
- варіанти системи;
- критерій для оцінки досягнення цілей;
- моделі дослідження системи;
- тип системи;
- властивості системи.

Приклад 1. Об’єктом аналізу є автомобіль.

Мета – забезпечити нормальне функціонування автомобіля.

1. Система загалом повна система і має такі підсистеми:
   - S_1 – система виконавця (водій, водійський склад);
   - S_2 – система об’єктів перевезень (грузи, пасажири);
   - S_3 – система живлення (автозаправні станції);
   - S_4 – система забезпечення і обслуговування (станції технічного обслуговування автомобілів);
   - S_5 – система доріг.
2. Повна система – автомобіль як сукупність функціональних підсистем.

Навколишнє середовище – це системи S_1 – S_5 , а також S_6 − природне середовище, S_7 – система навчання водіїв, S_8 – економічна система (автозаводи, торгівельні організації), S_9 – технологічна система тощо.

3. Ціль (мета) та призначення системи і підсистем.

Призначення автомобіля – перевезення пасажирів і вантажів.

Призначення підсистем випливає із назв.

Набір умов і обмежень:

- тип вантажу (наприклад, будівельні матеріали);
- маса вантажу (наприклад, 3,5 тонн);
- відстань (наприклад, 60–80 км);
- час доставки (наприклад, не більше 1–1,5 години);
- характеристика місцевості (наприклад, місто чи його найближчі відстані);
- збереження вантажу (наприклад, втрати не більше 0,1%) тощо.

4. Входи, ресурси, затрати. Вхід – це об'єкт перевезення (вантаж, пасажир). Ресурси – це паливно-мастильні матеріали, а також гроші, час і зусилля на перевезення. Затрати визначаються як витрата ресурсів на перевезення після досягнення мети (наприклад, витрати бензину –29 л, витрати грошей –20 у. о., витрати часу (трудовитрати) – 3 год, витрати зусиль – 4 000 Ккал) тощо.

5. Вихід, результати, прибуток. Вихід – це об'єкт перевезення (вантаж, пасажир), який доставлено, а також економія грошей, часу, зусиль за рахунок перевезення. Прибуток – це кількісна оцінка результатів у певних одиницях. Наприклад, економія грошей – 30 у. о., економія часу – 1 година, економія зусиль – 4 000 Ккал. Результат і прибуток оцінюються відносно цілей системи більш високого рівня (технологічний процес, виконання проєкту, виконання замовлення) у вигляді впливу на зменшення простоїв, забезпечення неперервності технічного циклу, зменшення рекламацій і штрафних санкцій.

6. Рівень робот, програми, підпрограми. Можливі такі види робіт:

- перевезення вантажів різного призначення (твердих, сипучих тощо);
- робота за графіком;
- термінове постачання вантажу;
- перевезення вантажу на великі відстані тощо.

7. Виконавці, ОПР і керівники. Виконавець – це водій (водійський склад); ОПР – диспетчер, начальник ділянки робіт; керівник – начальник робіт, проєкту, для яких виконуються перевезення.

8. Варіанти системи можна створити, якщо використати умови й обмеження з пункту 3. Для наведеного прикладу для дослідження підходять, наприклад, ГАЗ 53 А, КамАЗ, ЗІЛ тощо.

9. Критерій для оцінки досягнення цілей містить функціональні, техніко-економічні, ергономічні показники, наприклад, вантажопідйомність, швидкість, потужність двигуна, а також надійність, економічність, експлуатаційні витрати, комфорт, зручність в управлінні, зручність в обслуговуванні тощо.

10. Моделі ухвалення рішень.

11. Тип системи: технічна, відносно-закрита, статична.

12. Властивості системи. Автомобіль має властивість ієрархічної впорядкованості, оскілки може бути розкладений на підсистеми; має властивість централізації, оскільки його центром є двигун; має властивість інерційності,

оскільки у нього є кінцевий час розгону і гальмування; автомобіль є адаптивним середовищем, оскільки він зберігає свою функцію у разі збурювальних впливів середовища, наприклад, у разі зміни кваліфікації водія, якості палива, обслуговування, дороги, зміни погодних умов тощо.

Приклад 2. Об'єктом аналізу є комп'ютер. Мета – забезпечити нормальне функціонування комп'ютера.

1. Система в цілому повна система і має підсистеми:

- S_1 – система виконавця (оператор, користувач);
- S_2 – система завдань, що вирішуються (вхідна інформація, джерела вхідного впливу);
- S_3 – система живлення (електрична мережа);
- S_4 – система забезпечення і обслуговування (система програмного забезпечення, інформаційні мережі, служби налагодження і ремонту).

Повна система – комп'ютер як сукупність функціональних підсистем.

Врахувати призначення комп'ютера зберігати і обробляти інформацію.

2. Навколишнє середовище – це системи $S_1 – S_5$, а також S_6 – природне середовище, S_7 – система навчання, S_8 – економічна система (фірми розробники, торгівельні організації), S_9 – технологічна система тощо.

3. Ціль (мета) и призначення системи і підсистем. Призначення комп'ютера – зберігання та обробка інформації. Призначення підсистем випливає із їхніх назв. Набір умов і обмежень (комп'ютер, як і інші технічні системи, є багатоцільовою системою):

- тип завдань, що вирішується (наприклад, редагування тексту);
- вид тексту (наприклад, наукова стаття);
- обсяг тексту (наприклад, 10 сторінок);
- складність тексту (наприклад, наявність таблиць, формул, рисунків);
- час редагування (наприклад, не більше 1 год.);
- остаточна форма подання тексту (наприклад, тип і розмір шрифту, параметри сторінок тощо).

4. Входи, ресурси, затрати. Входом є вхідна інформація про задачу, яка розглядається. До ресурсів відносяться електроенергія, інформація, а також гроші час і зусилля на розв'язання задачі. Витрати – це кількісна оцінка ресурсів, наприклад, кількість інформації –1 Мбайт, електроенергії за добу 0,5 Квт · год, гроші (обслуговування, апаратурне і програмне забезпечення, заробітна платня) – 20 у. о., зусилля – 2 000 Ккал, годин – 1 год.

5. Вихід, результати, прибуток. Вихід – це результат розв'язання задачі (інформація про розв'язок), наприклад,

відредагований текст, схема, результат обчислень тощо.

6. Прибуток – це кількісна оцінка результатів у певних одиницях. Наприклад, економія грошей – 30 у. о., економія часу – 1 година, економія зусиль – 4 000 Ккал. Результат і прибуток оцінюються відносно цілей системи більш високого рівня (технологічний процес, виконання проєкту, виконання замовлення) у вигляді впливу на зменшення працевтрат, підвищення ефективності управління, скорочення часу на обробку інформації тощо.

7. Виконавці, ОПР і керівники. Виконавець – оператор; ОПР – кваліфікований користувач, постановник завдання, керівник підрозділу, у якому використовується комп'ютер, для яких виконуються перевезення.

8. Варіанти системи можна створити, якщо використати умови й обмеження з пункту 3 і визначаються конфігурацією комп'ютера. Для наведеного прикладу для дослідження підходять, наприклад, типу АТ, ХТ тощо.

9. Критерій або міри ефективності. Для комп'ютера до функціональних критеріїв відносяться: швидкість, об'єм пам'яті, універсальність, стабільність і тощо до техніко-економічних – економічність, надійність, витрати на експлуатацію, до ергономічним – безпека, простота у обслуговуванні, зручність, дизайн.

10. Моделі ухвалення рішень.

11. Тип системи: технічна, відносно-закрита, статична.

12. Властивості системи. Система є ієрархічно упорядкованою, нецентралізованою (розподіленою), адаптивною, інерційною.

Контрольні запитання

1. Дайте визначення системи.
2. Опишіть навколишнє середовище системи.
3. Охарактеризуйте ціль (мету) і призначення системи і підсистем.
4. У чому суть входів, ресурсів, затрат; виходів, результатів, прибутку?
5. Що таке рівень робіт, програм, підпрограм?
6. У чому роль виконавців, ОПР (особа, що ухвалює рішення) і керівників?
7. Як обрати варіанти системи?

ПРАКТИЧНА РОБОТА № 2

СКЛАДНІ СИСТЕМИ. ОСНОВНІ ПРИНЦИПИ ОБ'ЄКТНО-ОРІЄНТОВАНОГО ПІДХОДУ

Мета − засвоїти основні ознаки складних систем та основні принципи об'єктно-орієнтованого підходу.

Завдання

1. Провести аналіз складності системи за п'ятьма ознаками складних систем.

2. Зробити аналіз наявності трьох принципів об'єктно-орієнтованої парадигми: інкапсуляція, успадкування, поліморфізм.

Варіанти для виконання практичної роботи обрати із варіантів практичної роботи № 1.

Загальні відомості

Об’єктно-орієнтована методологія аналізу та проєктування інформаційних систем складається з п’яти головних кроків [1,2]:

1. Визначення предметної області.
2. Визначення об’єктів області.
3. Визначення структури об’єктів шляхом створення відношень «складається з» і «є».
4. Визначення атрибутів об’єктів.
5. Визначення сервісу об’єктів (методів поведінки) і взаємодій шляхом посилання повідомлень між об’єктами.

Статична структура інформаційної системи описується в термінах об’єктів і встановлених зв’язків між ними.

Динамічна структура відображає поводження системи й описується в умовах обміну повідомленнями між об’єктами. Кожен об’єкт системи має своє власне поводження, що моделює поводження об’єкта реального світу.

Концептуальною основою об’єктно-орієнтованого підходу є об’єктна модель. Основними положеннями її побудови є:

- абстрагування (abstraction);
- інкапсуляція (encapsulation);
- модульність (modularity);
- ієрархія (hierarchy).

Абстрагування − це виділення істотних характеристик деякого об’єкта, які відрізняють його від усіх інших видів об’єктів і, таким чином, чітко визначають його концептуальні кордони щодо подальшого розгляду та аналізу. Абстрагування концентрує увагу на зовнішніх особливостях об’єкта і дозволяє відокремити найістотніші особливості його поводження від деталей їхньої реалізації. Вибір правильного набору абстракцій для заданої предметної області становить головне завдання об’єктно-орієнтованого проєктування.

Інкапсуляція – це процес **відокремлення** один від одного окремих елементів об’єкта, що визначають його пристрій і поводження. Інкапсуляція слугує для того, щоб ізолювати інтерфейс об’єкта, що відображає його зовнішнє поводження, від внутрішньої реалізації об’єкта. Об’єктний підхід передбачає, що власні ресурси, якими можуть маніпулювати тільки методи цього об’єкта, сховані від зовнішнього середовища.

Абстрагування й інкапсуляція є взаємодоповнювальними операціями: абстрагування фокусує увагу на зовнішніх особливостях об’єкта, а інкапсуляція (чи, інакше, обмеження доступу) не дозволяє об’єктам-користувачам розрізняти внутрішню будову об’єкта.

Приклад. Припустимо, управління банку постановило, що якщо клієнт має кредитний рахунок, цей кредит може бути використаний як овердрафт на його рахунку «до запитання». (Овердрафт – короткостроковий кредит (до року) в межах встановленого ліміту, що дозволяє здійснювати розрахунки, коли коштів на поточному рахунку недостатньо).

У неінкапсульованій системі ми починаємо з вузьконаправленого аналізу змін, які необхідно внести до системи. Ми зазвичай не знаємо, де в системі знаходяться всі звернення до функції зняття з рахунку, тому доводиться шукати їх скрізь. Після того, як вони знайдені, ми повинні внести до них деякі зміни, щоб реалізувати нові вимоги; якщо ми працюємо ретельно, то, ймовірно, зможемо виявити близько 80 % випадків використання цієї функції.

В інкапсульованій системі не потрібно проводити такий детальний аналіз.

Ієрархія − це ранжирування або упорядкування системи абстракцій, розташування їх по рівнях. Прикладами ієрархії класів є просте і множинне спадкування (один клас використовує структурну або функціональну частину відповідно одного або декількох інших класів).

Основними поняттями об'єктно-орієнтованого підходу є об'єкт і клас (рис. 2.1, 2.2).

Експерт	Ім'я класу
ПІБ	Поля даних (властивості)
Звання	
Вчений ступінь	
Місце роботи	
Встановити рейтинг	Методи (функції операції)
Визначити допуск	

Рисунок 2.1− Приклад опису класу [1]

Об'єкт (рис. 2.2) визначається як реальність (tangible entity) − предмет чи явище, що мають чітко визначене поводження. Структура і поводження схожих об'єктів визначають загальний для них клас. Терміни «екземпляр класу» і «об'єкт» є еквівалентними. Стан об'єкта характеризується переліком всіх можливих (статичних) властивостей цього об'єкта і поточними значеннями (динамічними) кожної з цих властивостей.

Експерт	Ім'я об'єкта
Петренко Юрій Іванович	Поля даних (властивості)
Професор	
Доктор технічних наук	
ХНУБА	
Встановити рейтинг	Методи (функції операції)
Визначити допуск	

Рисунок 2.2 − Приклад опису об'єкта

Об'єкти: студент Петров, національний університет будівництва та архітектури, Вища математика, Теорія прийняття рішень.

Клас – це безліч об'єктів, пов'язаних спільністю структури і поведінки. Будь-який об'єкт є екземпляром класу. Визначення класів і об'єктів − одна з найскладніших завдань об'єктно-орієнтованого проєктування. Клас є моделлю ще не існуючої сутності (об'єкта). Клас має в складі поля даних та методи.

Два об'єкти − Вища математика та Теорія прийняття рішень належать одному і тому же класу − ДИСЦИПЛІНА. Назва дисципліни, семестр, обсяг годин, форма контролю – це його атрибути, що належать класу. Значення атрибутів належить об'єкту. Наприклад: Вища математика. 7 семестр, 100 годин, іспит. Всі об'єкти одного і того ж класу характеризуються однаковими наборами атрибутів.

Об'єднання об'єктів в класи дозволяє ввести в задачу абстракцію і розглянути її в загальнішій постановці. Клас має ім'я , яке відноситься до всіх об'єктів цього класу. Крім того, у класі вводяться імена атрибутів, які визначені для об'єктів. У цьому сенсі опис класу аналогічний опису типу структури (записи); при цьому кожен об'єкт має той же сенс, що і екземпляр структури (змінна або константа відповідного типу).

Поля даних (атрибути). Параметри (властивості) об'єкта, що задають його стан. Фізично поля становлять значення (змінні, константи), оголошені як такі, що «належать класу».

Методи. Процедури і функції, що пов'язані з класом. Вони визначають дії, які можна виконувати над об'єктом такого типу і які сам об'єкт може виконувати.

Об'єктно-орієнтоване проектування полягає в описі структури та поводження проєктованої системи, тобто фактично у відповіді на два основних питання:

1. З яких частин складається система?
2. У чому полягає відповідальність кожної з частин?

Виділення частин проводиться так, щоб кожна мала мінімальний за обсягом і точно певний набір виконуваних функцій (обов'язків) і при цьому взаємодіяла з іншими частинами якомога менше.

П'ять ознак складної системи:

1. Складні системи часто є ієрархічними та складаються з взаємозалежних підсистем, які також можуть бути розділені на підсистеми, і т. д., аж до «найнижчого рівня». Архітектура системи складається як з компонентів, так і з (ієрархічних) відношень цих компонентів.
2. Вибір, які компоненти в цій системі вважаються елементарними, є відносно довільним та в великій мірі покладається на дослідника. Нижній рівень для одного користувача може виявитись достатньо високим для іншого.
3. Внутрішньокомпонентний зв'язок зазвичай є сильнішим, ніж зв'язок між компонентами. Це дозволяє відділяти «високочастотні» взаємодії всередині компонент від «низькочастотної» динаміки взаємодії між компонентами.
4. Ієрархічні системи зазвичай складаються з небагатьох типів підсистем, по-різному скомбінованих та організованих.
5. Будь-яка працююча складна система є результатом розвитку більш простої системи, що працювала раніше.

Складна система, спроєктована «з нуля», ніколи не запрацює. Варто починати з працюючої простішої системи.

Приклади складних систем:

– «астрономічний» всесвіт;
– літак;
– диспетчерська система управління повітряним рухом;
– ринкова економіка.

Основні принципи об’єктно-орієнтованої парадигми

Успадкування. В об’єктно-орієнтованих системах успадкування є механізм, що дозволяє створювати нові об’єкти, ґрунтуючись на тих, що вже існують. Породжуваний (child) об’єкт-нащадок успадковує властивості батьківського (parent) об’єкта, що його породжує.

Приклад. У банківській системі успадкування можна застосовувати для роботи з різними типами рахунків. Наш гіпотетичний банк обслуговує чотири типи рахунків: до запитання (checking), ощадний (savings), кредитний (credit) і депозитний сертифікат. Ці різні типи рахунків мають схожі властивості, до яких відносяться номер рахунку, ставка відсотка і власник. Отже, можна створити батьківський об’єкт account (рахунок) із загальними характеристиками всіх рахунків (рис. 2.3).

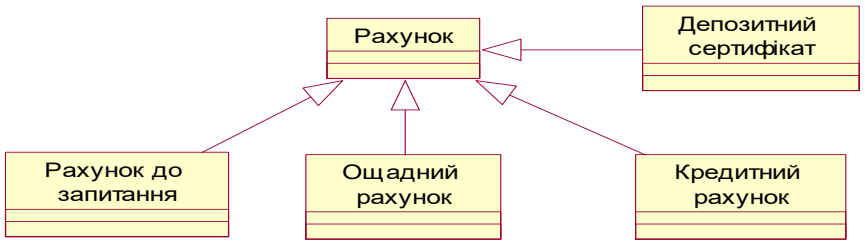


Рисунок 2.3 – Успадкування різних типів рахунків від «рахунку взагалі» [1]

Поліморфізм. Поліморфізм означає наявність багатьох форм або реалізацій конкретної функціональності. У термінах об’єктно-орієнтованих систем це означає, що конкретні функціональні можливості можуть мати багато реалізацій. Ще спроба визначення: це наявність одного інтерфейсу для різних варіантів спорідненої функціональності.

Припустимо, що розробляється система малювання графіків. Коли користувач хоче щось накреслити – чи то лінія або прямокутник; система повинна виконати команду «Draw» (намалювати). Наша система включає багато типів різних форм, кожна з яких містить поводження, що дозволяє їй намалювати себе. За допомогою поліморфізму система визначає, яку фігуру вона повинна намалювати.

Без поліморфізму код функції «draw()» виглядає приблизно так:

```
Function Shape.DrawMe()
{
        CASE Shape.Type
                Case “Circle”
                        Shape.drawCircle();
                Case “Rectangle”
                        Shape.drawRectangle ();
                Case “Line”
                        Shape.drawLine ();
        END CASE
}
```

Реалізовано три інтерфейси для малювання. За використання поліморфізму код для малювання є зверненням до функції «DrawMe()» викреслюваного об’єкта, наприклад:

```
Function draw()

{

        Shape.drawMe();

}
```

Зараз маємо лаконічніший інтерфейс. При цьому кожна форма (круг, лінія, прямокутник і так далі) повинна містити власну функцію «DrawMe()» для її малювання.

Контрольні запитання

1. Дайте визначення складної системи.
2. Що означає інкапсуляція?
3. Опишіть механізм успадкування.
4. Що означає поліморфізм?
5. У чому суть основних принципів об’єктно-орієнтованого підходу до складних систем?
6. Назвіть признаки, за якими визначаються складні системи.

ПРАКТИЧНА РОБОТА № 3
МОДУЛЬНА СТРУКТУРА ПРЕДМЕТНОГО СЕРЕДОВИЩА

Мета роботи – ознайомитися із модульною структурою предметного середовища.

Завдання − для заданого опису предметного середовища побудувати модульну структуру системи та виконати її поділ на підсистеми.

Варіанти для виконання практичної роботи обрати із варіантів практичної роботи № 1.

Загальні відомості

Процес моделювання [3] починається з етапу вивчення предметного середовища.

Приклад. Система «Служба зайнятості в межах ЗВО» призначена для того, щоб допомогти студенту влаштуватися на роботу вже в процесі його навчання у ЗВО. Подавши заяву в систему, студент стає її клієнтом і починає обслуговуватися протягом усього навчання. Заява становить анкету. Система пропонує професійні (засновані на досліджуваних предметах) психологічні тестування, що проводяться раз у семестр, а за підсумками успішності формуються експертні оцінки. На основі зібраної інформації складається резюме, яке надсилається всім організаціям, що мають необхідні вакансії.

Основним призначенням системи є автоматизація введення й зберігання звітних даних по студентах, складання характеристик і резюме, пошуку вакансій у фірмах. Система дозволяє змінювати, доповнювати, вести пошук і перегляд інформації про студентів, накладати обмеження доступу, зберігати списки студентів, що закінчили навчання, у вигляді архіву, контролювати видачу студентові завдань на курсові роботи й проєкти, зв’язувати ЗВО із фірмами, зацікавленими в пошуку співробітників.

Також ця система може бути використана для складання окремих списків груп, для друку залікових відомостей, повної бази даних і статистики.

Система складається із чотирьох підсистем (п/с):

1. Підсистема контролю успішності студентів.
2. Підсистема професійних і психологічних тестів.
3. Підсистема обробки запитів та визначення категорій користувачів.
4. Підсистема експертних оцінок.

Модульну систему **підсистем** наведено на рисунках 3.1−3.3, модульна структура **системи** подається на рисунку 3.4.

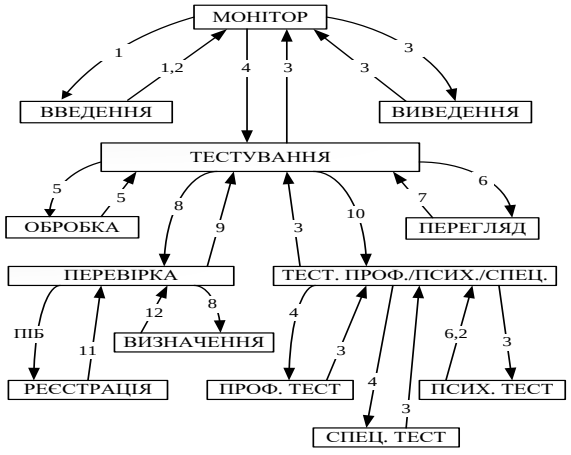


Рисунок 3.1 − Модульна структура підсистеми професійних і психологічних тестів [3]

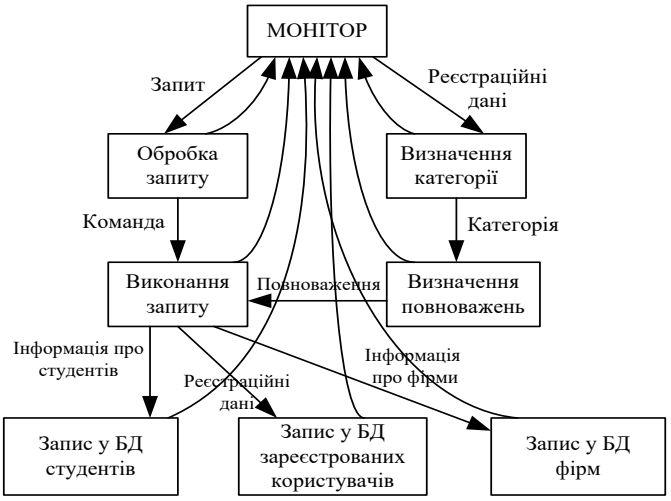


Рисунок 3.2 − Модульна структура підсистеми обробки запитів та визначення категорій користувачів [3]

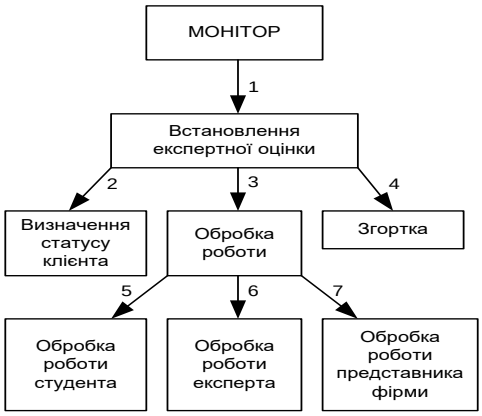


Рисунок 3.3 − Підсистема експертних оцінок [3]

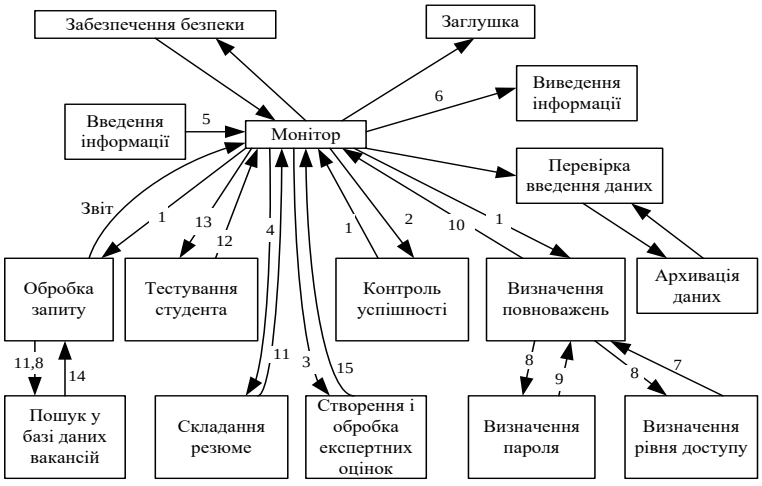


Рисунок 3.4 − Модульна структура системи [3]

Контрольні запитання

1. Поясніть модульну структуру системи.
2. Перелічіть основні інформаційні об’єкти системи.
3. Опишіть порядок занесення даних у БД.
4. Перелічіть функціональні характеристики системи.
5. Назвіть категорії користувачів системи та їхні повноваження.
6. Опишіть критерії оцінок.

ПРАКТИЧНА РОБОТА № 4
МЕТОДОЛОГІЯ ПРОЄКТУВАННЯ ІНТЕЛЕКТУАЛЬНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Мета – вивчення особливостей методів проєктування інформаційної системи.

Завдання – провести за своїм варіантом аналіз робочих процесів і фаз створення інформаційної системи, специфікацію вимог, створити функціональну модель.

Варіанти для виконання практичної роботи обрати із варіантів практичної роботи № 1.

Загальні відомості

Всі сучасні технології аналізу і проєктування ІІС ґрунтуються на методології моделювання предметної області, які поділяються на структурну та об’єктно-орієнтовану. Принципова відмінність між структурною та об’єктно-орієнтованою методологією проєктування інформаційних систем полягає в підході до декомпозиції системи. Об’єктно-орієнтований підхід базується на об’єктній декомпозиції.

Статична структура інформаційної системи описується в термінах об’єктів і встановлених зв’язків між ними.

Динамічна структура відображає поводження системи й описується в умовах обміну повідомленнями між об’єктами. Кожен об’єкт системи має своє власне поводження, що моделює поводження об’єкта реального світу.

Практичні роботи з моделювання інтелектуальних інформаційних систем охоплюють різні аспекти аналізу, проєктування та реалізації систем [4].

Наведемо список таких аспектів.

Аналіз вимог − це збір, документування та аналіз вимог до інформаційної системи. Приклад: створити специфікацію вимог для системи обліку клієнтів у CRM. Інструменти: текстові редактори, таблиці, діаграми BPMN.

Побудова діаграм бізнес-процесів – це вміння моделювати бізнес-процеси організації. Приклад: змоделювати процес обробки замовлення на

складі за допомогою діаграми BPMN. Інструменти: Bizagi Modeler, Microsoft Visio, Lucidchart.

Моделювання даних − це розробка логічної та фізичної моделі бази даних.

Приклад створити ER-діаграму для системи обліку студентів у навчальному закладі. Інструменти: MySQL Workbench, ERwin, dbdiagram.io.

Проєктування архітектури інформаційної системи − це визначення структури основних компонентів системи та їхньої взаємодії. Приклад: спроєктувати архітектуру вебзастосунку для онлайн-магазину. Інструменти: UML-діаграми компонентів, Draw.io.

Побудова UML-діаграм – це моделювання окремих аспектів інформаційної системи. Приклади: побудувати діаграму варіантів використання (Use Case) для системи електронного документообігу. Розробити діаграму класів для системи управління бібліотекою. Побудувати діаграму активностей для процесу реєстрації користувачів. Інструменти: StarUML, Lucidchart, Visual Paradigm.

Моделювання взаємодії користувачів із системою – це розробка прототипів інтерфейсів. Приклад: створити інтерфейс системи онлайн-бронювання квитків. Інструменти: Figma, Adobe XD, Axure.

Розробка сценаріїв тестування − це проєктування тестових сценаріїв для перевірки роботи системи. Приклад: створити тестові випадки для функціоналу реєстрації користувача. Інструменти: TestRail, Excel, JIRA.

Моделювання динаміки системи – це розробка динамічних моделей для аналізу роботи системи. Приклад: змоделювати взаємодію об’єктів у системі за допомогою діаграми послідовностей (Sequence Diagram). Інструменти: UML-редактори, Draw.io.

Імітаційне моделювання – це аналіз поводження системи за допомогою імітаційних моделей. Приклад: створити модель роботи супермаркету для оцінки завантаженості кас. Інструменти: AnyLogic, Simulink, Arena.

Проєктування логіки системи – це реалізація логіки бізнес-правил і процедур. Приклад : створити логіку обробки замовлень у системі за допомогою BPMN-діаграми. Інструменти: Camunda, Signavio.

Оптимізація інформаційної системи – це визначення вузьких місць та запропонування шляхів вдосконалення системи. Приклад: оптимізувати процес доставки товарів у логістичній системі. Інструменти: аналіз діаграм, математичні моделі.

Програмна реалізація моделі – це реалізація прототипу інформаційної системи. Приклад : створити просту систему управління складом із базою даних. Інструменти: Python, SQL, JavaScript.

Уніфікованим процесом розробки інформаційних систем є **Rational unified process (RUP)** – ітеративний процес розроблення інформаційної системи, який надає план ефективного створення та розгортання систем (рис. 4.1, 4.2).

Моделі розробляються в ході робочих процесів, кожен із яких характеризується:

1) співробітник (worker) – діяльність, що доручена одній або кільком особам, визначає потрібні для цієї дії якості та здібності. Співробітникові відповідає опис:

- обов'язків;
- очікуваної поведінки;

2) вид діяльності – частина роботи (операція), яку виконує співробітник під час робочого процесу;

3) артефакт (artifact) – істотна частина інформації, яка створюється, змінюється або використовується співробітником під час розроблення ПП. Артефактом може бути модель, її елемент або документ;

4) модель ВВ (use-case model) (вв – використання варіантів) – повний набір акторів, ВВ системи та взаємозв'язків між ними, зокрема прототип інтерфейсу, опис архітектури ПЗ, сценаріїв ВВ (**Use Case (сценарій користування)** – це перелік дій, сценарій, за яким користувач взаємодіє з додатком або програмою для виконання будь-якої дії та досягнення конкретної мети).

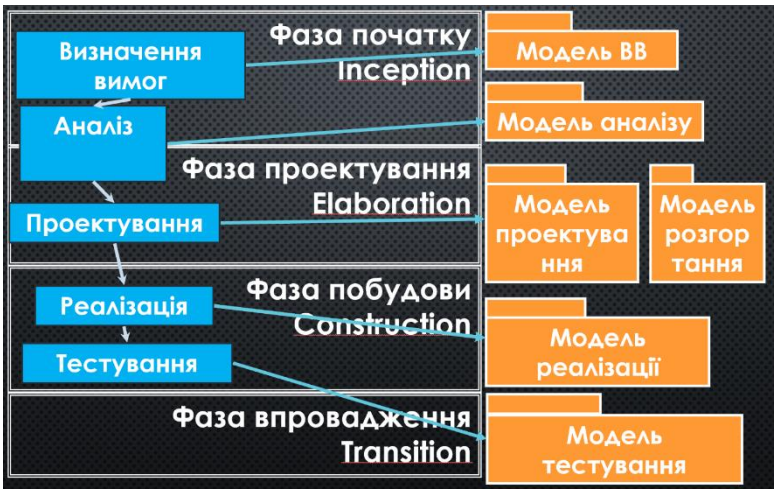


Рисунок 4.1− Етапи послідовного моделювання з використанням RUP [4]

Робочі процеси і фази створення інформаційної системи.

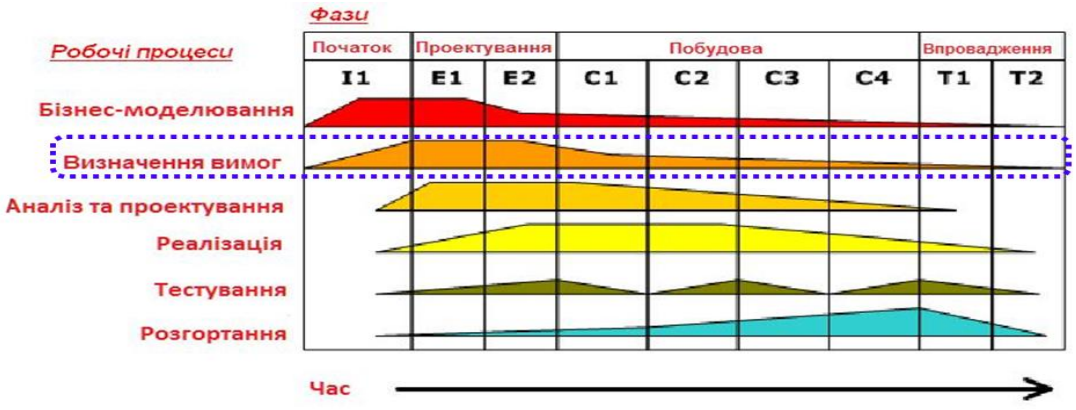


Рисунок 4.2 − Робочі процеси і фази створення інформаційної системи [4]

Робочий процес визначення вимог подано на рисунку 4.3

Рисунок 4.3 − Процес визначення вимог [4]

Ідентифікація ВВ у визначенні вимог

Розробка Use Cases (UC) − варіантів використання (ВВ):

- всі функції визначаються ВВ;
- нефункціональні вимоги об’єднуються в один ВВ;
- інші нефункціональні вимоги, спільні для кількох ВВ, заносять у спеціальні вимоги.

Use case – *безліч сценаріїв, об’єднаних за деяким критерієм, які описують послідовності виконуваних системою дій, що надають значущий для деякого актора результат.*

Інші артефакти у визначенні вимог

Опис архітектури – архітектурна форма моделі ВВ:

- включає критичні для функціонування системи ВВ та ті, що впливають на організаційну структуру системи;
- є основою для визначення пріоритетів та послідовності розробки.

Архітектура – це організаційна структура та відповідна поведінка системи.

Архітектура може бути рекурсивно розкладена на частини, які взаємодіють через інтерфейси, відносини, які з’єднують частини, і обмеження на їхню збірку. Частини системи, які взаємодіють через інтерфейси, включають класи, компоненти і підсистеми.

Більшість існуючих методів об’єктно-орієнтованого аналізу і проєктування (ООАП) включають як **мову моделювання**, так і **опис процесу моделювання**.

Мова моделювання − це нотація (сукупність умовних знаків у системі правил для опису), яка використовується методом для опису проєктів.

Нотація становить сукупність графічних об’єктів, які використовуються в моделях. Вона є синтаксисом мови моделювання. Наприклад, нотація діаграми

класів визначає, як подаються такі елементи і поняття, як клас, асоціація і множинність.

Процес − це опис кроків, які необхідно виконати під час розроблення проєкту.

Моделювати складну систему необхідно з кількох різних точок зору, кожен раз беручи до уваги один аспект, що моделюється, і абстрагуючись від інших.

Мовою, яка об'єднала сильні сторони відомих методів і забезпечила найкращу підтримку моделювання, стала уніфікована мова моделювання UML (Unified Modeling Language) [1,4].

Моделювання в UML змістовно поділяється **на** моделювання використання, моделювання структури та моделювання поводження системи, зв'язок між ними (рис. 4.4) поданий у вигляді діаграми діяльності.

Моделювання структури покликане відповідати (з різним ступенем деталізації) на питання «З чого складається система?».

Моделювання поведінки покликане відповідати на питання «Як працює система?».

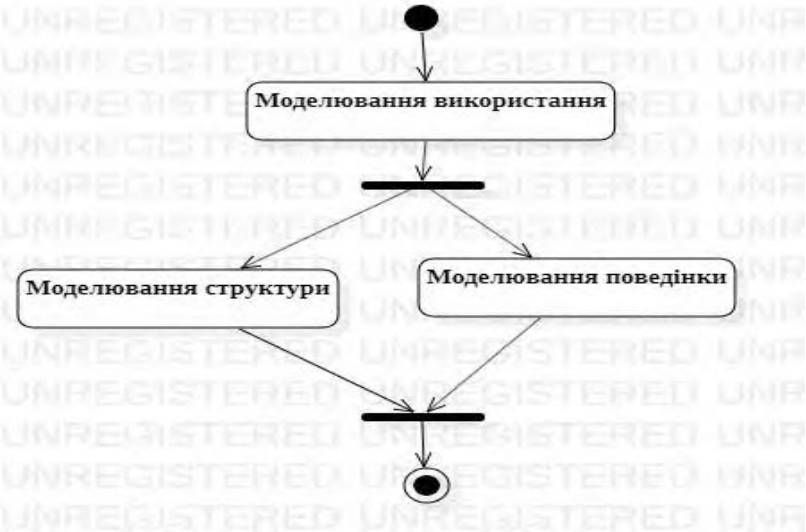


Рисунок 4.4 − Діаграма діяльності процесу моделювання в UML [4]

Розглянемо три типи призначення моделі:

- концептуальні моделі;
- моделі проєктування;
- моделі реалізації.

Концептуальна модель (conceptual model). Формулювання змістовного і внутрішнього подання, що **поєднує концепцію користувача й розробника моделі.** Це абстрактна модель, що визначає структуру модельованої системи, властивості її елементів і причинно-наслідкові зв'язки, властиві системі, що є суттєві для досягнення мети моделювання.

Основні елементи концептуальної моделі:

- умови функціонування об’єкта, що визначені характером взаємодії між об’єктом і його оточенням, а також між елементами об’єкта;
- мета дослідження об’єкта та напрямок покращення його функціонування;
- можливості керування об’єктом, визначення складу керованих змінних об’єкта.

Концептуальною основою об’єктно-орієнтованого підходу є діаграма класів. Існують різні точки зору на побудову діаграм класів залежно від цілей їхнього застосування.

Концептуальна точка зору − діаграма класів описує модель предметної області, у ній присутні тільки класи прикладних об’єктів.

Точка зору проєктування (специфікації) − діаграма класів застосовується під час проєктування інформаційних систем. Деталізує концептуальні рішення для можливостей побудови системи і здійснення процесу програмування.

Точка зору реалізації − діаграма класів містить класи, використовувані безпосередньо в програмному коді (під час використання об’єктно-орієнтованих мов програмування).

Діаграма (diagram) − графічне подання сукупності елементів моделі у формі зв’язного графа, вершинам і ребрам (дугам) якого приписується визначена семантика.

Під час моделювання систем, незалежно від предметної області, UML передбачає можливість побудови діаграм різних типів, що відображають найбільш важливі аспекти побудови системи.

Сукупність побудованих у такий спосіб діаграм є самодостатньою в тому сенсі, що в них міститься вся інформація, яка необхідна для реалізації проєкту складної системи (рис. 4.5).

Рисунок 4.5 − Структура діаграм UML [4]

Структурні діаграми. В UML існують структурні діаграми для візуалізації, специфікації, конструювання та документування статичних аспектів системи, які складають її основу.

Статичні аспекти інформаційних систем відображають наявність та розташування класів, інтерфейсів, кооперацій, компонентів, вузлів та інших сутностей.

Приклад. Створити систему забезпечення онлайн-купівлі квитків на розважальні заходи. Для покупки клієнт повинен надати інформацію про своє прізвище, ім'я та актуальний email. Оплачені квитки повинні надсилатись на адресу email, вказану у замовленні. Клієнт може роздрукувати оплачені квитки.

Реєстрація в системі потрібна, якщо клієнт хоче отримувати персоналізовані пропозиції.

Оплата виконується через онлайн-сервіс платіжної системи.

Етапи створення системи

1. Ідентифікація акторів.

Актори:

- *клієнт (Новий та Зареєстрований)* – особа, яка використовує вебсайт для здійснення покупки квитків в інтернеті;
- *платіжна система* – автоматизована система виконання онлайн-платежів.

2. Ідентифікація ВВ.

Варіанти використання:

- *пошук заходів* – клієнт шукає квитки за запитом про місто проведення, діапазоном дат та/або типом заходу;
- *придбання квитків* – для відібраних у UC «Пошук заходів» клієнт обирає захід та кількість квитків і здійснює покупку, оплачені квитки надсилаються клієнту на вказаний у замовленні email;
- *реєстрація клієнта* – новий клієнт реєструється на сайті (дані для реєстрації – логін, пароль, контактні дані: прізвище, ім'я, актуальний email);
- *аутентифікація* – перевірка даних зареєстрованого клієнта.

3. Опис архітектури:

а) критичні для функціонування системи ВВ:

- *UC «Придбання квитків»;*
- *UC «Пошук заходів»;*

б) ВВ, що впливають на структуру системи:

- *UC «Реєстрація клієнта»;*
- *UC «Аутентифікація».*

Контрольні запитання

1. На чому ґрунтуються сучасні технології аналізу і проєктування ІІС ?
2. Яка різниця між структурною та об'єктно-орієнтованою методологією проєктування інформаційних систем?
3. Назвіть принципи ООП.
4. Назвіть основні аспекти аналізу, проєктування та реалізації систем.

ПРАКТИЧНА РОБОТА № 5
СЕМАНТИЧНІ МЕРЕЖІ ДЛЯ ПОДАННЯ ЗНАНЬ В ІНТЕЛЕКТУАЛЬНИХ СИСТЕМАХ УПРАВЛІННЯ

Мета роботи − навчитися створювати та застосовувати семантичні мережі для подання знань в інтелектуальних системах.

Завдання − побудувати семантичну модель заданого об'єкта; реалізувати програму з використанням семантичної моделі.

Загальні відомості

Проблема подання знань є однією з центральних під час побудови системи штучного інтелекту [2, 4].

Знання − це закономірності наочної області (принципи, закони, зв'язки), отримані в результаті практичної діяльності і професійного досвіду, які дозволяють фахівцям ставити і вирішувати завдання в певній галузі.

Всі знання в системі штучного інтелекту зберігаються в базі знань.

Об'єктивній оцінці ситуації відповідають певні моделі знань, які умовно поділяються на концептуальні та емпіричні.

Семантична мережа − один із способів подання знань. Спочатку семантична мережу розроблялася як модель уявлення довгострокової пам'яті у психології, але згодом стала одним із способів подання знань у експертних системах.

Семантика − спільні відносини між символами та об'єктами цих символів.

Семантичні мережі є однорідним класом моделей подання знань. Часто загальною основою для віднесення схеми подання знань до семантичної мережі є те, що вона подана у вигляді орієнтованого графа, вершини якого відповідають об'єктам (поняттям, сутностям) предметної області, а дуги – відносинам (зв'язкам) між ними. І вузли, і дуги зазвичай мають мітки (імена). Імена вершин і дуг зазвичай збігаються з іменами відповідних об'єктів та відносин предметної області.

Об'єкти предметної області, що відображаються в семантичній мережі, можна умовно поділити на три групи: узагальнені, індивідуальні (конкретні) та агрегатні.

Узагальнений об'єкт відповідає деякій абстракції збірної реального об'єкта, процесу або явища предметної області. Узагальнені об'єкти фактично представляють певні класи предметної області.

Індивідуальний об'єкт є одиничним представником (примірником) класу, виділеного певним чином.

Агрегат − складний об'єкт, утворений з інших об'єктів, що розглядаються як його складові.

Введена класифікація об’єктів є відносною. Залежно від розв’язуваного завдання той самий об’єкт може розглядатися як узагальнений чи індивідуальний, як агрегатний чи неагрегатний.

Типи зв’язків між об’єктами семантичних мереж можуть бути будь-якими. Але найчастіше використовуються такі основні зв’язки (відносини): «рід-вид», «є представником», «є частиною». Наявність зв’язку «рід-вид» між узагальненими об’єктами А і означає, що поняття А є більш загальним, ніж поняття В. Будь-який об’єкт, що відображається поняттям В, відображається і поняттям А, але не навпаки.

Вперше семантичні мережі з’явилися у сфері машинної лінгвістики як засіб аналізу змісту (семантики) природної мови. У семантичній мережі М.Р. Куілліана вершини відповідають обумовленому поняттю і забезпечуються вказівниками на інші слова, що розкривають це поняття.

База знань організована у вигляді сторінок (площин), на кожній з який представляється граф, що визначає одне слово.

Отже, семантичні мережі − це графічні моделі у вигляді графів, вершинам (вузлам) яких відповідають деякі поняття (об’єкти, події, процеси), а зв’язкам (тобто дугам графа) − відносини між цими поняттями. Вузли мережі зображуються колами, прямокутниками, а зв’язки між ними лініями зі стрілками (рис. 5.3).

Вершини − це об’єкти, дуги − це відносини. Семантична модель **не** розкриває сама по собі, як здійснюється подання знань. Тому семантична мережа розглядається як метод подання знань та структурування знань.

Різноманіття відносин, що використовуються у семантичних мережах, підрозділяються на такі групи:

- лінгвістичні відносини, що також підрозділяються на відмінкові (агент, об’єкт, інструмент, час, місце), дієслівні (спосіб, час, вид, число, стан) і атрибутивні (колір, розмір, форма і т. п.);
- логічні відносини (кон’юнкція, диз’юнкція, заперечення, імплікація);
- теоретико-множинні відносини містять у собі відносини типу «множина-підмножина» («рід-вид», «клас-підклас»), «ціле-частина», «елемент множини» і ін.;
- квантифіковані відносини, що підрозділяються на логічні квантори спільності й існування, нечіткі квантори (багато, трохи, часто тощо).

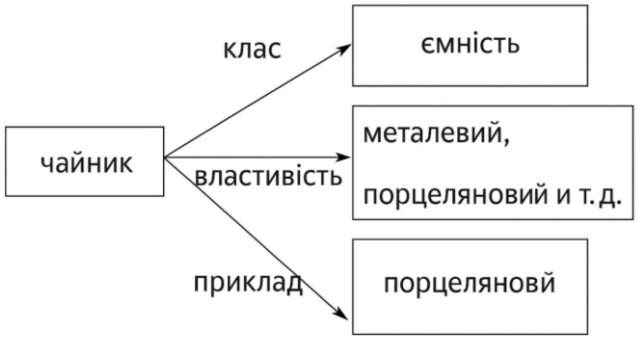


Рисунок 5.1 – Приклад найпростішого зразка семантичної мережі

Таким чином, розглянуті раніше відносини, («рід-вид», «є представником», «є частиною», відмінкові відносини) далеко не вичерпують усього набору відносин, що застосовуються у семантичних мережах. Проте вони утворюють основу для побудови прикладних баз знань.

Особливе місце при цьому займають теоретико-множинні відносини, що володіють транзитивними властивостями.

Способи опису семантичних мереж і логічний вивід

Серед методів опису семантичних мереж найчастіше використовуються концептуальні графи Дж. Сува і блокові структури Х. Хендрикса. Тому розглянемо опис семантичних мереж як концептуальних графів.

Вершинами концептуального графа є об'єкти (поняття, сутності) предметної галузі чи концептуальні відносини. Ребра концептуального графа з'єднують поняття-вершини-поняття та відносини-вершини-напрямки. У той самий час ребра можуть з концепції вершини і закінчуватися у бік вершин відносин, і навпаки. Вершини «куля» та «червона» відповідають вершинам понять, а вершина «колір» відповідає бінарному відношенню. Щоб розрізняти ці типи вершин концептуальних граф, вони зображуються прямокутником і еліпсом відповідно. Однією з переваг поділу відносин на незалежні вершини концептуального графа є спрощення уявлення пов'язаних відносин. Такий зв'язок зображується *n*-ребрами-вершинами-версусами (рис. 5.2).

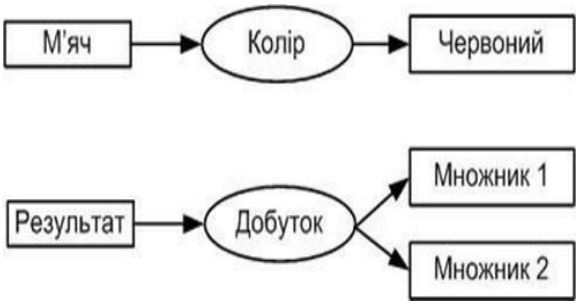


Рисунок 5.2 − Приклади концептуальних графів

Зазвичай кожний концептуальний граф фіксує одне речення. Тоді база знань буде подаватись у вигляді сукупності таких графів. Наприклад, верхній граф на рисунку 5.3 відповідає реченню «М'яч має червоний колір». Граф, що зображений на рисунку 5.4, відображає більш складне речення: «Іван закріпив деталь стільця клеєм».

Рисунок 5.3 – Приклад концептуального графа пропозиції

Цей граф використовує відмінкові відносини дієслова «закріпити» і демонструє можливості концептуальних графів для подання пропозицій природної мови.

Кожне поняття-вершина-поняття концептуального графа може мати мітку типу. Тип позначає клас приналежності вершини. На концептуальному графі мітку типу вершини відокремлюють двокрапкою від конкретного імені вершини.

Крім числового маркеру, на концептуальних графах можуть застосовуватися узагальнені маркери, які позначаються знаком «*».

На концептуальних графах вводяться операції, що дозволяють виконувати їхнє перетворення. Нові графи виходять за допомогою чотирьох операцій: копіювання, спеціалізації, об'єднання і спрощення.

Концептуальні графи можуть містити вершини, що є пропозиціями. Концептуальний граф дозволяє природним чином виражати кон'юнкцію. Складніше з іншими логічними операціями: диз'юнкцією, логічним запереченням, кванторами.

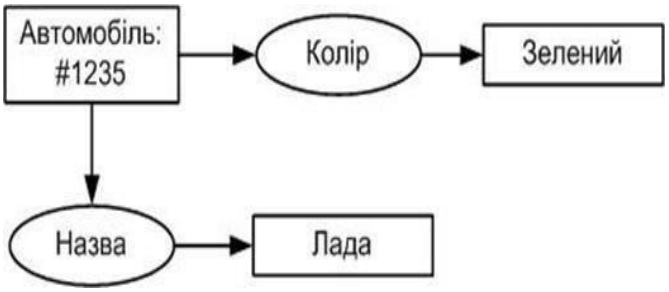


Рисунок 5.4 − Концептуальний граф із числовим маркером

Приклад. Створити семантичну мережу «Комп'ютер складається з процесора, пам'яті, пристрою вводу-виводу інформації та пристрою управління. Процесор зчитує дані з пам'яті і передає отримані результати для запису в пам'ять. Дані також надходять в пам'ять з пристрою вводу-виводу інформації та можуть зчитуватися з пам'яті за допомогою цього пристрою. Робота процесора, пам'яті і пристрою вводу-виводу керується за допомогою пристрою управління».

Представимо інформацію, що міститься в цьому тексті, за допомогою семантичної мережі (рис. 5.5). Як вузли цієї мережі виступають поняття «Комп'ютер», «Процесор», «Пам'ять», «Пристрій вводу-виводу», «Пристрій управління». Зв'язки (відносини) між цими поняттями називаються: «має», «записує дані», «зчитує дані», «управляє».

Можна стверджувати, що дана семантична мережа визначає деякий клас об’єктів («Комп’ютер»), названий структурою. Ця структура характеризується чотирма атрибутами («Процесор», «Пам’ять», «Пристрій вводу-виводу», «Пристрій управління»). Конкретний об’єкт (наприклад, «Персональний комп’ютер HP Compaq nx9010») належить цій структурі, володіючи всіма перерахованими атрибутами.

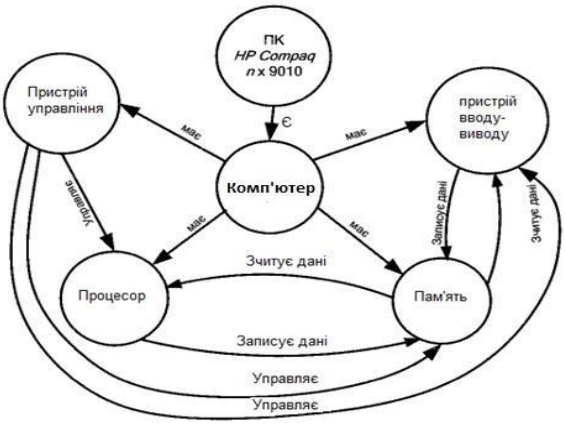


Рисунок 5.5 − Семантична мережа «Комп’ютер» [4]

Для опису (подання в ЕОМ) семантичної мережі може використовуватися квадратна матриця (рис. 5.6) такого вигляду, де використані скорочення: ПВВ − пристрій вводу-виводу; ПУ − пристрій управління.

	1. Комп’ютер	2. Процесор	3. Пам’ять	4. ПВВ	5. ПУ
1. Комп’ютер		Має	Має	Має	Має
2. Процесор			Записує дані		
3. Пам'ять		Зчитує дані		Зчитує дані	
4.ПВВ			Записує дані		
5.ПУ			Управляє	Управляє	Управляє

Рисунок 5.6 − Матриця для опису семантичної мережі

На перетині і-го рядка і j-го стовпчика цієї матриці записується ім’я відповідного ставлення, якщо з і-го вузла в j-й вузол виходить лінія (зв’язок). У нашому прикладі (не рахуючи виділеного об’єкта «HP Compaq пх9010») ми маємо п’ять рядків і п’ять стовпців (за числом понять – вузлів мережі). Загальна кількість не порожніх клітин матриці дорівнює 11, тобто числу відмічених зв’язків мережі.

Кожен вузол мережі може володіти власною структурою, тобто може бути поданий у вигляді власної семантичної мережі, яка розкриває суть цього поняття. Наприклад, поняття «Процесор» може бути визначене як безліч взаємодіючих між собою компонент «Суматор», «Регістри операндів», «Регістри зсуву» тощо. У цьому випадку можна говорити про ієрархічну семантичну мережу, яка передбачає розбиття розглянутої мережі на ряд підмереж.

Переваги семантичних мереж:

- наочність (не випадковим є виникнення терміна «семантична візуалізація» інформації);
- багатство засобів відображення різних відношень між базовими поняттями розглянутої предметної області;
- можливість створення правил бази знань.

Приклад виконання. Створити семантичну мережу для розпізнавання типу комп'ютера. Дуги: країна − виробник, стандартна конфігурація, область застосування, використане програмне забезпечення.

Використовуючи відповідні дуги, побудувати семантичну мережу, що стосується розпізнавання типу комп'ютера. Дуги: країна виробник, стандартна конфігурація, область застосування, використане програмне забезпечення.

Розроблена семантична мережа наведена на рисунках 5.7−5.10:

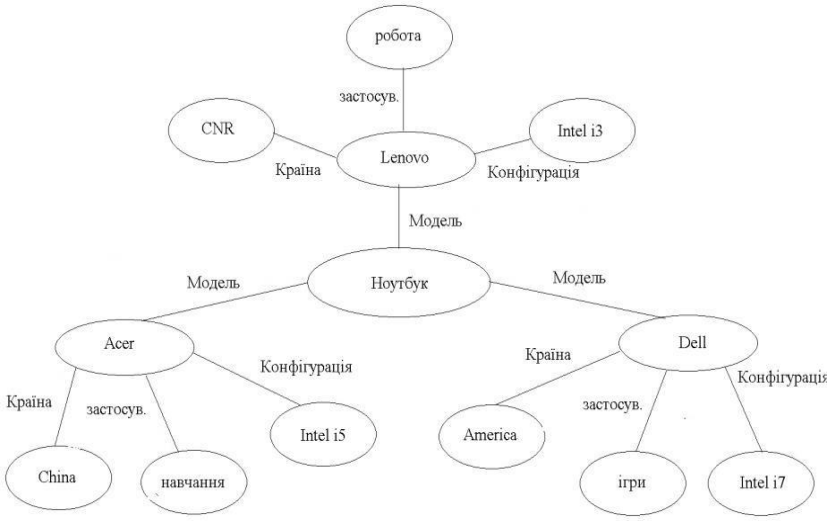


Рисунок 5.7 − Семантична мережа

Програма в MatLab:

```
>> SN=SNnew;
>> SN=SNaddORnode(SN,'China','America','CNR',
'education','games','work','Intel i5','Intel i7','Intel i3') SN = node:
{'China' 'America' 'CNR' 'education' 'games' 'work' 'Intel i5' 'Intel
i7' 'Intel i3'} relation: {9x9 cell}
nodetype: [1 1 1 1 1 1 1 1 1]
>> SN=SNaddANDnode(SN,'Model','Країна',
'застосув.','кофігурація','Acer','Dell','Lenovo') SN = node: {1x16
cell}
relation: {16x16 cell}
nodetype: [1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0]
>> SN=SNaddrelation(SN,'Model','has','Країна');
>> SN=SNaddrelation(SN,'Model','has','застосув.');
>> SN=SNaddrelation(SN,'Model','has','кофігурація');
>> SN=SNaddrelation(SN,'Країна','type','China');
>> SN=SNaddrelation(SN,'Країна','type','America');
>> SN=SNaddrelation(SN,'Країна','type','CNR');
>> SN=SNaddrelation(SN,'застосув.','type','education');
>> SN=SNaddrelation(SN,'застосув.','type','games');
>> SN=SNaddrelation(SN,'застосув.','type','work');
>> SN=SNaddrelation(SN,'кофігурація','type','Intel i5');
>> SN=SNaddrelation(SN,'кофігурація','type','Intel i7');
>> SN=SNaddrelation(SN,'кофігурація','type','Intel i3');
>> SN=SNaddrelation(SN,'China','include','Acer');
>> SN=SNaddrelation(SN,'America','include','Dell');
>> SN=SNaddrelation(SN,'CNR','include','Lenovo');
>> SN=SNaddrelation(SN,'education','include','Acer');
>> SN=SNaddrelation(SN,'games','include','Dell');
>> SN=SNaddrelation(SN,'work','include','Lenovo');
>> SN=SNaddrelation(SN,'Intel i5','include','Acer');
>> SN=SNaddrelation(SN,'Intel i7','include','Dell');
>> SN=SNaddrelation(SN,'Intel i3','include','Lenovo');
>> SNplot(SN, 'hierarchy');
>> SNplot(SN, 'circle');
```

Рисунок 5.8 – Код програми

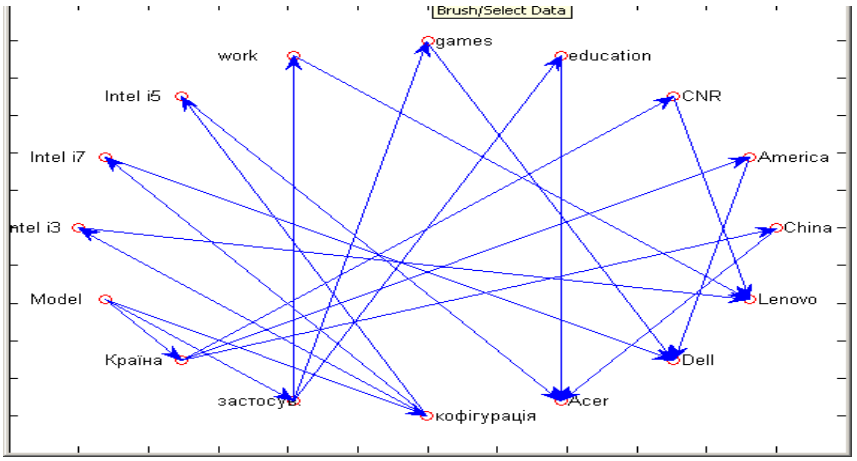


Рисунок 5.9 − Семантична мережа кругової структури, виконана в MatLab [6]

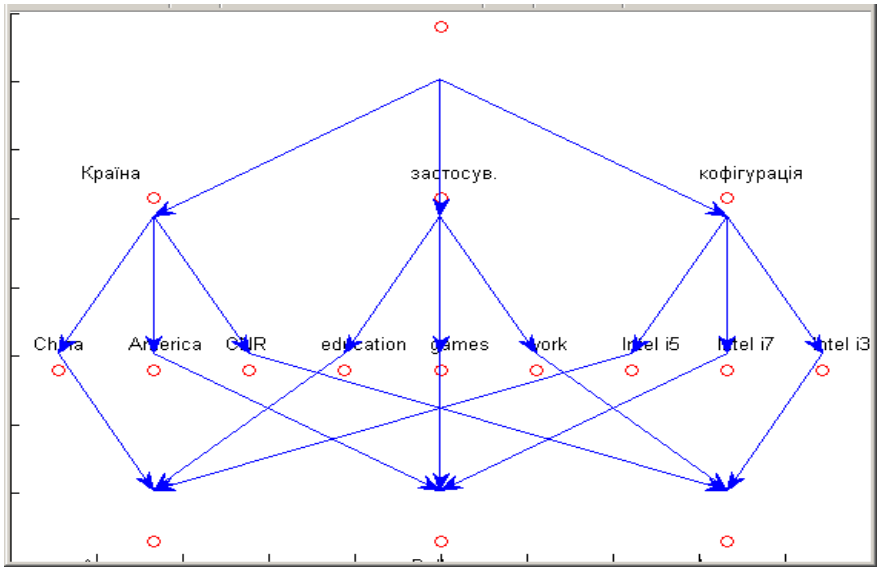


Рисунок 5. 10 − Семантична мережа ієрархічної структури, виконана в MatLab [6]

Контрольні запитання

1. Що таке семантична мережа і для чого її застосовують?
2. У чому полягає ідея створення семантичної мережі?
3. Як представляються дані в семантичній мережі?
4. Чи існують обмеження на число зв'язків елементів, властивостей і складність при побудові семантичної мережі?
5. Які стосунки запропоновані як оператори відносин для групування вершин?

ПРАКТИЧНА РОБОТА № 6
ВИКОРИСТАННЯ ФРЕЙМІВ ДЛЯ ФОРМУВАННЯ БАЗИ ЗНАНЬ В ІНТЕЛЕКТУАЛЬНИХ СИСТЕМАХ УПРАВЛІННЯ

Мета роботи − навчитися використовувати фрейми для подання знань в інтелектуальних системах управління.

Завдання − використовуючи фреймову модель подання знань, реалізувати структуру відносин відповідно до варіанта завдання.

Варіанти завдань

1. Екзамен з дисципліни за семестр у викладача при складових: семестр, екзамен, викладач, оцінка, студент, отримувати.
2. Відомість при складових: дисципліна, студент, іспит, семестр, викладач, оцінка.
3. Конференція з комерційних питань при складових: дата, місце проведення, тема, мета, доповідачі.
4. Отримання оцінки при складових: викладач, студент, оцінка, отримувати.
5. Використання виробу при складових: організація, розробка технологічного рішення, дослідження «фізичного ефекту», методи створення виробу.
6. Інформаційна структура бази даних для технічних приладів при складових: фізичні ефекти, технічні рішення, вироби, об'єкт поставки виробу, стенди, нормативи.
7. Класифікація продукту при складових: назва, область застосування, спосіб зберігання, спосіб транспортування.
8. Аудиторія (опис) при складових: місткість, призначення, що складають, місцезнаходження.
9. Отримання оцінки при складових: викладач, студент, оцінка, отримувати.
10. Використання виробу при складових: організація, розробка технологічного рішення, дослідження «фізичного ефекту», методи створення виробу.

Загальні відомості

Фреймові моделі були вперше запропоновані одним з патріархів штучного інтелекту, професором Массачусетського технологічного інституту (США) Марвіном Мінскі [4].

У загальному випадку фрейм може бути поданий у вигляді структури, показаної на рисунку 6 .1. Опис фрейму такий

$$< Z,(Y_1,T_1,A_1),...,(Y_n,T_n,A_n) >,$$

де Z – ім'я фрейму: Y_i – ім'я слота; T_i − значення слота; A_i – ім'я приєднаної процедури (необов'язковий параметр).

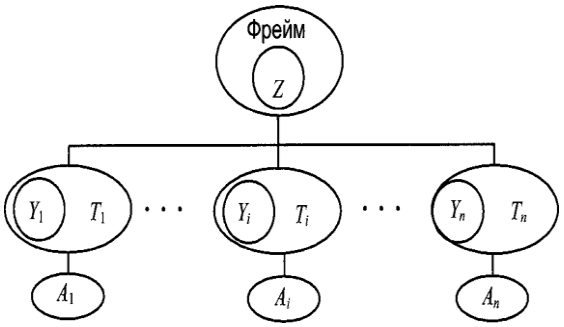


Рисунок 6.1 − Структура фрейму [4]

Слоти – це деякі незаповнені структурні елементи фрейму, заповнення яких призводить до того, що цей фрейм ставиться у відповідність ситуації, що розглядається (явищу, об’єкту). Фрейм з незаповненими слотами називається фреймом-прототипом (або протофреймом). Фрейм із заповненими слотами називається фреймом-прикладом (або фреймом-екземпляром). Як дані фрейм може містити звернення до процедур (так звані приєднані процедури).

Існують різні способи заповнення значень слотів у фреймі-екземплярі:

- за замовчуванням від фрейму-прототипу, що зберігається в базі знань;
- через успадкування властивостей фрейму;
- через приєднану процедуру;
- явно з діалогу з користувачем;
- з бази даних.

На рисунку 6.2 наведено приклад фрейму «Персональний комп’ютер».

Ім'я фрейма	Ім'я слота	Значення слота
Персональний комп'ютер *HP Compaq n×9010*	Процесор	*Intel Pentium* 4 2,66 ГГц
	Монітор	*TFT* 15,1» *XGA*
	Оперативна пам'ять	256 МБ
	Жорсткий диск	40 ГБ
	Модем	56 К (*V*.92)
	Операційна система	*Microsoft Windows XP Home*

Рисунок 6.2 − Фрейм «Персональний комп’ютер» [4]

Важливою особливістю фреймів є їхня вкладеність, тобто значення слота, який також може бути набором слотів нижчого рівня тощо («Принцип матрьошки»). Подібно до семантичних мереж, каркасні моделі подання знань дозволяють описати найважливіші зв’язки між атрибутами об’єктів і реалізувати швидкий висновок на основі принципу успадкування. Таким чином, рамкові моделі виступають потужним і ефективним засобом організації знань при збереженні інформації про структуру досліджуваних об’єктів.

Фреймворки є одним із поширених формалізмів для подання знань в експертних системах. Фрейм можна уявити як конструкцію, що складається з набору комірок – прорізів. Кожен слот складається з імені та пов'язаних значень. Значеннями можуть бути дані, процедури, посилання на інші кадри або порожні. Ця конструкція дуже зручна для моделювання аналогій, опису областей із загальними зв'язками понять тощо.

Будь-який кадр складається з кількох компонентів, назви та вміст яких описані нижче:

1. Назва фрейму. Це ідентифікатор, призначений кадру. Фреймворк повинен мати унікальне ім'я в цій системі фреймворків.

2. Назва слота. Це ідентифікатор, призначений слоту. Слот повинен мати унікальне ім'я в межах кадру, до якого він належить. Зазвичай назва слота не має семантичного значення і є лише ідентифікатором цього слота.

3. Покажчики успадкування. Ці покажчики стосуються тільки фреймових систем ієрархічного типу, засновані на відносинах «абстрактне – конкретне» вони показують, яку інформацію про атрибути слотів у фреймі верхнього рівня успадковують слоти з такими ж іменами у фреймі нижнього рівня. Типові покажчики успадкування: Unique (U – унікальний), Same (S: такий саме), Range (R: встановлення шраниь), Override (O: ігнорувати) тощо. U показує, що фрейм може мати слоти з різними значеннями: S – всі слоти повинні мати однакові значення, R – значення слотів фрейму нижнього рівня повинні знаходитися в межах, зазначених значеннями слотів фрейму верхнього рівня, O – за відсутності вказівки значення слота фрейму верхнього рівня стає значенням слота фрейму нижнього рівня, але в разі визначення нового значення слотів фреймів нижніх рівнів вказуються як значення слотів.

4. Тип даних. Тут вказується, що слот має чисельне значення, або служить покажчиком іншого фрейму. До типів даних відносяться: FRAME (покажчик), INTEGER (цілий), REAL (дійсний), BOOL (булевий), LISP (приєднана процедура), TEXT (текст), LIST (список), TABLE (таблиця), EXPRESSION (вираз) та ін.

5. Значення слота. Пункт вводу значення слота. Значення слота повинне збігатися із зазначеним типом даних цього слота, крім того повинна виконуватися умова успадкування.

6. Демон. Тут дається визначення демонів типу IF-NEEDED, IF-ADDED, IF-REMOVED і т.д. Демоном називається процедура, що автоматично запускається при виконанні деякої умови. Демони запускаються при зверненні до відповідного слота. Крім того, демон є різновидом приєднаної процедури.

7. Приєднана процедура. Як значення слота можна використовувати програму процедурного типу. Коли ми говоримо, що в моделях представлення знань фреймами об'єднуються процедурні та декларативні знання, то вважаємо демони і приєднані процедури процедурними знаннями.

Особливістю ієрархічної структури є те, що інформація про атрибути фрейму на верхньому рівні спільно використовується всіма фреймами нижніх рівнів, пов'язаних із ним.

Контрольні запитання

1. Що становить фрейм, його складові?
2. Що таке слот і з яких частин він складається?
3. Для чого слугують ім’я фрейму та ім’я слота?
4. Для чого слугують покажчики успадкування?
5. Для чого слугують вказівка типу даних, демон?
6. Для чого слугують приєднана процедура і значення слота?

ПРАКТИЧНА РОБОТА № 7
НЕЙРОННІ МЕРЕЖІ І МАШИННЕ НАВЧАННЯ

Мета – моделювання нейронної мережі з використанням машинного навчання.

Завдання:

1. Зібрати дані, про динаміку будь-якого процесу. Провести аналіз і підготовку даних. На основі зібраних даних побудувати моделі регресії, провести їхній порівняльний аналіз і зробити висновки. Побудувати нейронну мережу (НМ) про прогнозування обраного показника якості, зробити висновки про якість моделі.

2. Зібрати дані, що підлягають класифікації (наприклад, про види перехідних процесів). На їхній основі побудувати моделі класифікації, провести їхній порівняльний аналіз і зробити висновки про доцільність застосування. Побудувати НМ, зробити висновки про якість моделі.

3. Зібрати дані, що підлягають кластеризації (наприклад, дані з попереднього завдання без класів). Побудувати відповідні моделі, провести їхній порівняльний аналіз і зробити висновки про доцільність застосування. Побудувати НМ, зробити висновки про якість моделі.

Загальні відомості

Машинне навчання (*Machine learning, ML*) – група методів у сфері штучного інтелекту, набір алгоритмів, які застосовують для створення машини, яка вчиться на власному досвіді. Як навчання машина обробляє величезні масиви вхідних даних і знаходить у них закономірності. За вже звичним словом «алгоритм» ховається складна робота з його налаштування та навчання. Алгоритми адаптивно покращують свою ефективність, оскільки збільшуються дані, доступні для навчання. Отже, чим більше даних, тим точнішою буде наша модель і ефективнішим навчання [6].

Інструменти *«Data science»* і *«Machine learning»* багато в чому перетинаються, але все ж вони різні і кожен зі своїми завданнями (рис. 7.1).

Рисунок 7.1 − Складові штучного інтелекту [7]

Штучний інтелект (*Artificial intelligence, AI*) – це різні технологічні та наукові рішення і методи, які допомагають зробити програми подібні до інтелекту людини. *Artificial intelligence* містить безліч інструментів, алгоритмів і систем, серед яких також всі складові *Data science* і *Machine learning*.

Data science – наука про методи аналізу даних і вилучення з них цінної інформації, знань. Вона перетинається з такими областями, як машинне навчання і наука про мислення (*Cognitive Science*), а також із технологіями для роботи з великими даними (*Big Data*). Результатом роботи *Data science* є проаналізовані дані і знаходження правильного підходу для подальшої обробки, сортування, вибірки, пошуку даних.

Машинне навчання або *Machine learning* – один з розділів *AI*, алгоритми, що дозволяють комп'ютеру робити висновки на підставі даних, не підкоряючись жорстко заданим правилам. Отже, машина може знайти закономірність у складних і багатопараметричних завданнях (які мозок людини не може вирішити), таким чином знаходячи більш точні відповіді. Як результат – правильне прогнозування.

Нейронна мережа (НМ) за допомогою штучних нейронів моделює роботу людського мозку (нейронів), що вирішує певне завдання, самонавчається з урахуванням попереднього досвіду (рис. 7.2). І з кожним разом робить все менше помилок. Неймережі є одним із видів машинного навчання, а не окремим інструментом.

Мета машинного навчання – частково або навіть повністю автоматизувати розв'язання різних складних аналітичних задач.

Тому насамперед машинне навчання покликане давати максимально точні прогнози на підставі вхідних даних, щоб людина могла приймати правильні рішення в своїй роботі. У результаті навчання машина може передбачати результат, запам'ятовувати його, за необхідності відтворювати, вибирати кращий з декількох варіантів.

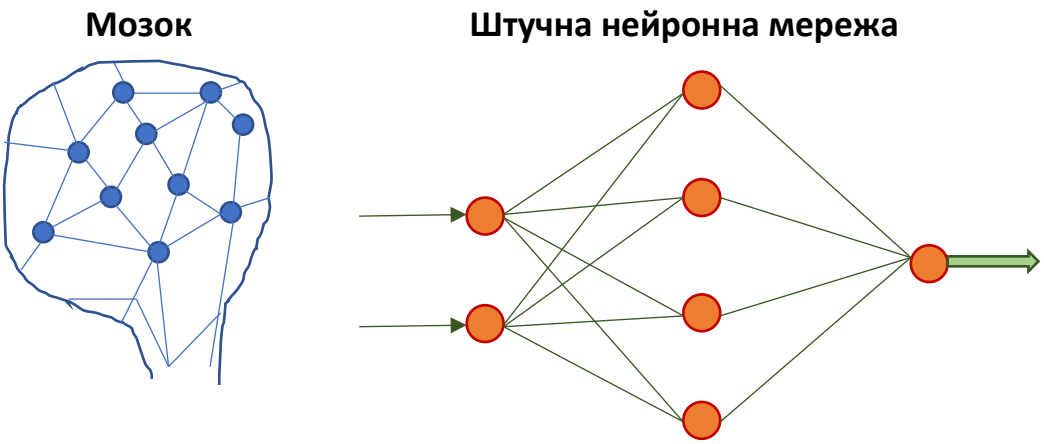


Рисунок 7.2 − Нейронні мережі [7]

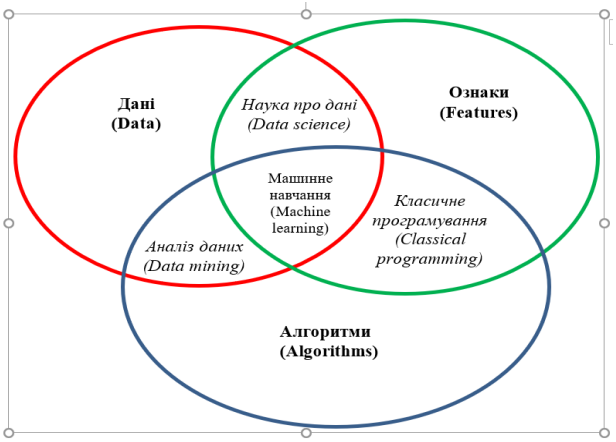


Рисунок 7.3 − Складові машинного навчання[7]

Машинне навчання має три основні складові:

а) дані – базова інформація, в яку входять будь-які вибірки даних, роботі з якими потрібно навчити систему;

б) ознаки – які саме характеристики і властивості повинна відстежувати система в результаті навчання;

в) алгоритм – вибір методу для вирішення поставленого завдання.

Види машинного навчання

Загалом розрізняють два типи машинного навчання: навчання по прецедентах (або індуктивне навчання) і дедуктивне навчання. Оскільки останнє прийнято відносити до сфери експертних систем, то терміни «машинне навчання» і «навчання по прецедентах» можна вважати синонімами. Бази знань, що лежать в основі експертних систем, важко узгоджувати з реляційною моделлю даних, тому промислові СУБД неможливо ефективно використовувати для наповнення баз знань експертних систем.

Навчання по прецедентах також поділяють на три основних типи: контрольоване навчання, або навчання з учителем (*supervised learning*),

неконтрольоване навчання (*unsupervised learning*), або навчання без учителя, і навчання з підкріпленням (*reinforcement learning*).

Крім названих, розробляються й інші методи навчання: активне, багатозадачне, різноманітне, трансферне тощо. Особливо успішно розвивається в останні роки «глибоке навчання», за використання якого можуть успішно поєднуватися алгоритми навчання з учителем і без вчителя.

За типом застосовуваних алгоритмів можна виділити два види:

1) класичне навчання – відомі і добре вивчені алгоритми навчання, розроблені переважно більше 50-ти років тому для статистичних бюро. Підходить насамперед під завдання роботи з даними: класифікація, кластеризація, регресія тощо. Застосовують для прогнозування, сегментації і так далі;

2) нейронні мережі і глибоке навчання – найбільш сучасний підхід до машинного навчання. Нейронні мережі застосовуються там, де потрібні розпізнавання або генерація зображень і відео, складні алгоритми управління або нелінійні об'єкти управління і подібні складні завдання.

Кілька підходів можна об'єднати і тоді вийдуть ансамблі моделей.

Нейронні мережі і глибоке навчання

Для машинного навчання використовують різні технології та алгоритми. Зокрема, можуть застосовуватися дискримінантний аналіз, байєсовські класифікатори та багато інших математичних методів. Наприкінці XX століття все більше уваги почали приділяти штучним нейронним мережам (далі – ШНМ). Черговий вибух інтересу до них почався в 1986 році, після істотного розвитку так званого «Методу зворотного поширення помилки», який з успіхом застосували під час навчання нейронної мережі.

ШНМ є системою з'єднаних і взаємодіючих між собою штучних нейронів, виконаних на основі порівняно простих процесорів. Кожен процесор ШНМ періодично отримує сигнали від одних процесорів (або від сенсорів, або від інших джерел сигналів) і періодично посилає сигнали іншим процесорам. Всі разом ці прості процесори, з'єднані в мережу, здатні вирішувати доволі складні завдання (рис. 7.4).

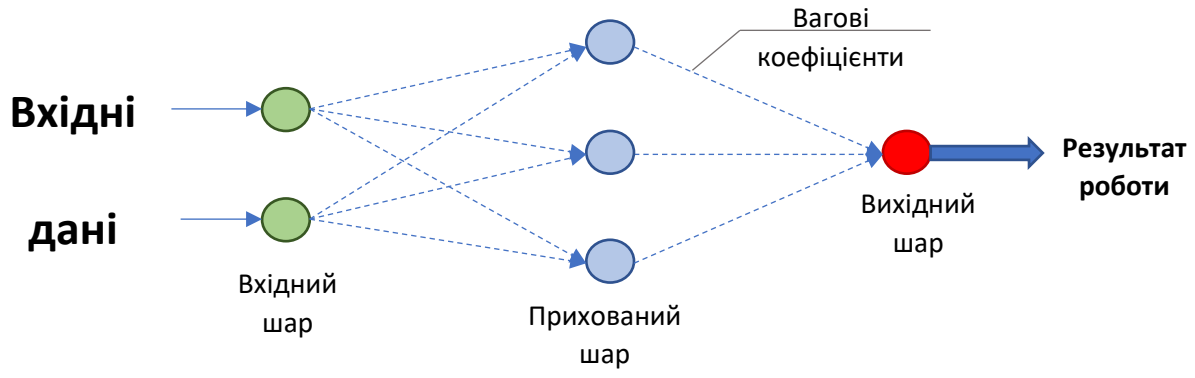


Рисунок 7.4 – Схема штучної нейронної мережі [7]

Найчастіше нейрони розташовуються в мережі за рівнями (їх ще називають шарами). Нейрони першого рівня – це зазвичай вхідні. Вони отримують дані ззовні (наприклад, від сенсорів системи розпізнавання осіб) і після їхньої обробки передають імпульси через синапси нейронів на наступному рівні. Нейрони на другому рівні (його називають прихованим, оскільки він безпосередньо не пов'язаний ні з входом, ні з виходом ШНМ) обробляють отримані імпульси і передають їх нейронам на вихідному рівні. Оскільки мова йде про імітацію нейронів, то кожен процесор вхідного рівня пов'язаний з декількома процесорами прихованого рівня, кожен з яких також пов'язаний з декількома процесорами рівня вихідного. Така архітектура найпростішої ШНМ, яка здатна до навчання і може знаходити прості взаємозв'язки в даних.

Архітектура штучних нейронних мереж

Хоча один нейрон і здатний виконувати найпростіші процедури розпізнавання, сила нейронних обчислень виникає під час з'єднання нейронів у мережах. Штучна нейронна мережа (ШНМ) – це мережа з кінцевим числом шарів з однотипних елементів – штучних нейронів із різними типами зв'язків між шарами. При цьому число нейронів у шарах вибирається виходячи з необхідності забезпечення заданої якості виконання завдання, а число шарів нейронів – якомога меншим для скорочення часу розв'язання задачі.

За архітектурою зв'язків ШНМ можуть бути згруповані в два класи: мережі прямого поширення (у яких графи не мають петель) – одношарові або багатошарові; і рекурентні мережі (або мережі зі зворотними зв'язками): змагальні або мережа Кохонена, мережа Хопфілда, мережа Хеммінга та ін.

На рисунку 9.7 подано структурні схеми одношарової і двошарової ШНМ прямого поширення, де нейрони подані одним кружечком, що містить суматор і активаційний блок, і пов'язані між собою інформаційними каналами. Кожній стрілці відповідає своя синаптична вага. Багатошарові ШНМ містять внутрішні приховані шари.

У мереж, наведених на рисунку 7.4, немає зворотного зв'язку, тобто з'єднань, що йдуть від виходів деякого шару до входів цього ж шару або попередніх шарів. Цей спеціальний клас мереж, званих мережами без зворотних зв'язків або мережами прямого поширення, становить інтерес і широко використовується.

Вибір структури НМ здійснюється відповідно до особливостей і складності завдання. Здебільшого оптимальний варіант виходить на основі інтуїтивного підбору. Єдина жорстка вимога, що висувається архітектурою до елементів мережі, це відповідність розмірності вектора вхідних сигналів мережі числу її входів.

НМ представляє адаптивну систему, життєвий цикл якої складається з двох незалежних фаз – навчання і роботи мережі. Нейронні мережі **не** програмуються в звичному сенсі цього слова, вони навчаються. Можливість навчання – одне з головних переваг нейронних мереж перед традиційними алгоритмами. Технічно

навчання полягає в знаходженні коефіцієнтів зв'язків між нейронами. У процесі навчання нейронна мережа здатна виявляти складні залежності між вхідними і вихідними даними, а також виконувати узагальнення. Це означає, що, в разі успішного навчання, мережа зможе надати правильний результат на підставі даних, які були відсутні в навчальній вибірці.

Контрольні запитання

1. Що таке машинне навчання?
2. Які основні складові машинного навчання?
3. Які є види машинного навчання?
4. Назвіть основні типи задач машинного навчання.
5. У чому відмінність класифікації від кластеризації?
6. Які алгоритми використовуються в задачі регресії?
7. Що необхідно врахувати під час прогнозування?
8. За яким принципом будуються штучні нейронні мережі?
9. Що дає зворотній зв'язок в архітектурі ШНМ?
10. У чому полягає навчання ШНМ?
11. Які варіанти використання ШНМ застосовуються в АСК?

ПРАКТИЧНА РОБОТА № 8
ЕВОЛЮЦІЙНЕ МОДЕЛЮВАННЯ ТА ГЕНЕТИЧНИЙ АЛГОРИТМ В ЗАДАЧАХ ШТУЧНОГО ІНТЕЛЕКТУ

Мета роботи – знайомство з моделюванням та генетичним алгоритмом у задачах штучного інтелекту.

Завдання:

1. Розглянути алгоритм задачі комівояжера с використанням генетичних алгоритмів (файл додається). Побудувати блок-схему.

2. Створити алгоритм оптимізації нетрадиційної задачі методом генетичного алгоритму.

Загальні відомості

Генети́чний алгори́тм (*genetic algorithm*) (далі – ГА) – це еволюційний алгоритм пошуку, що використовується для розв'язання задач оптимізації і моделювання шляхом послідовного **підбору, комбінування і варіації** шуканих параметрів із використанням механізмів, що нагадують біологічну еволюцію [7].

Особливістю генетичного алгоритму є акцент на використання оператора «схрещення», який виконує операцію **рекомбінацію р**ішень-кандидатів, роль якої аналогічна ролі схрещення в живій природі.

Більшість задач, які розв'язують за допомогою генетичних алгоритмів, мають **один** критерій оптимізації, який використовують як функцію

пристосованості. Багатокритерійна оптимізація також ґрунтується на відшукуванні розв'язку, що одночасно оптимізує більше, ніж одну функцію. У цьому випадку шукають певний компроміс, яким і виступає розв'язок.

Структура генетичного алгоритму

Задача кодується так, щоб її вирішення могло бути подано у вигляді **масиву подібного до інформації складу хромосоми**. Цей масив часто називають саме так «хромосома».

Випадково у масиві створюється деяка кількість початкових елементів «осіб», або початкова популяція.

Особи оцінюються з використанням **функції пристосування**, в результаті якої кожній особі присвоюється певне значення пристосованості, яке визначає можливість виживання особи.

Після цього з використанням отриманих значень пристосованості вибираються **особи, допущені до схрещення** (*селекція*).

До осіб застосовується «генетичні оператори» (здебільшого це **оператор схрещення** (crossover) і оператор мутації (mutation), створюючи таким чином наступне покоління осіб.

Особи наступного покоління також оцінюються застосуванням генетичних операторів і виконується селекція і мутація.

Так моделюється еволюційний процес, що продовжується декілька життєвих циклів (*поколінь*), поки не буде виконано критерій зупинки алгоритму.

Таким критерієм може бути:

- знаходження глобального, або надоптимального рішення;
- вичерпання числа поколінь, що відпущені на еволюцію;
- вичерпання часу, відпущеного на еволюцію.

Генетичні алгоритми можуть використатися для пошуку рішень в дуже великих просторах пошуку.

Можна виділити такі етапи генетичного алгоритму:

1. Створення початкової популяції:

а) обчислення функції пристосованості для осіб популяції (оцінювання);
б) повторювання до виконання критерію зупинки алгоритму;
в) вибір індивідів із поточної популяції (селекція) ;
г) схрещення або/та мутація;
д) обчислення функції пристосовуваності для всіх осіб;
е) формування нового покоління.

Наслідком першого кроку є популяція H, що налічує N осіб.

2. Відбір.

На етапі відбору необхідно із всієї популяції вибрати її певну долю, яка залишиться в «живих» на цьому етапі популяції. Є декілька способів провести відбір. Ймовірність виживання особи **h** повинна залежати від значення її пристосованості Fitness (h). Сама ж доля відібраних **s** зазвичай є параметром генетичного алгоритму, і її просто задають заздалегідь.

Внаслідок відбору із N осіб популяції H повинні залишитись sN осіб, які ввійдуть в наступну популяцію H'. Решта осіб «загине».

3. Розмноження.

Розмноження в генетичних алгоритмах зазвичай статеве – щоб «народити» нащадка, необхідно декілька батьків, зазвичай потрібна участь двох. Розмноження в різних алгоритмах описується по різному – воно, залежить від формату осіб. Головна вимога до розмноження – щоб нащадок чи нащадки мали можливість успадкувати риси всіх батьків, «змішавши» їх якимось достатньо розумним чином.

Розмноження або оператор рекомбінації застосовують відразу ж після оператора відбору батьків для отримання нових особин-нащадків. Сенс рекомбінації полягає в тому, що створені нащадки повинні **наслідувати** генну інформацію від обох батьків. Особи для розмноження зазвичай вибираються із всієї популяції **H**, а не із тих, що вижили на першому кроці (хоча останній варіант теж має право на існування). Справа в тому, що головна проблема генетичних алгоритмів – **нестача різноманітності** (diversity) в особах. Достатньо швидко виділяється єдиний генотип, який становить локальний максимум і згодом всі елементи популяції програють йому в відборі, і вся популяція «забивається» копіями цієї особи. Існують різні способи боротьби із таким небажаним ефектом; один з них – вибір для розмноження не з самих «пристосованих», а взагалі зі всіх осіб.

4. Мутації.

До мутацій відноситься все те ж, що і до розмноження: **є** деяка доля мутантів **m**, що є параметром генетичного алгоритму, і на кроці мутацій необхідно вибрати **mN** осіб, а згодом змінити їх згідно з заздалегідь заданими операціями мутації.

Пояснення до виконання роботи

Завдання № 1 Задачі комівояжера з використанням генетичних алгоритмів

Задача комівояжера (Travelling salesman problem, TSP) – одна з найвідоміших задач комбінаторної оптимізації, що полягає у знаходженні самого вигідного маршруту, що проходить через зазначені міста хоча б по одному разу з наступним поверненням у початкове місто. В умовах задачі вказуються критерій вигідності маршруту (найкоротший, найдешевший, сукупний критерій тощо) і відповідні матриці відстаней, вартості і т. п. Вказується зазвичай, що маршрут повинен проходити через кожне місто тільки один раз.

Розв’язок задачі комівояжера (TSP) за допомогою генетичного алгоритму (ГА) передбачає такі складові.

Мета задачі комівояжера – знайти найкоротшу відстань між N різними містами. Шлях, по якому продавець повинен пройти, називається туром.

Перевірка всіх варіантів проходу по N містах буде N!. Для розрахунку 30 турів по місту доведеться виміряти загальну відстань в $2{,}65 \times 10^{32}$ різних турів. Припускаючи близько трильйона операцій в секунду, це займе

252.333.390.232.297 років. Додавання ще одного міста призведе до збільшення кількості розрахунків на 31!. Очевидно, що такий підхід неприпустимий.

Щоб знайти розв'язок за набагато коротший термін може бути використаний **генетичний алгоритм**. Хоча він не може знайти краще рішення, він може стати практично ідеальним рішенням для розрахунку 100 турів по місту менше ніж за хвилину.

Основні кроки у розв'язанні задачі комівояжера за допомогою генетичного алгоритму.

1. Створити групу з багатьох випадкових турів у початковій популяції. Цей алгоритм використовує **жадібні** обчислення для формування початкової популяції, віддаючи перевагу зв'язкам міст, що знаходяться близько один до одного.
2. Вибрати з двох кращих батьків (**найкоротших турів**) у популяції і скомбінувати їх, **зробити двох нових** нащадків. Сподіваючись, що ці нащадки будуть краще, ніж будь-який з батьків.
3. З невеликою ймовірністю нащадки **піддаються мутації**. Це робиться, щоб запобігти ідентичності всіх турів у популяції.
4. Кращі нащадки (короткі тури) **замінюють собою** самі довжелезні тури в популяції. Чисельність особин у популяції залишається незмінною.
5. Процес ітераційно повторюється до досягнення бажаної мети.

Як видно з назви, генетичні алгоритми імітують природу і еволюцію з використанням принципів виживання найбільш пристосованих особин.

Для вирішення завдання комівояжера з використанням генетичного алгоритму виникають **основні проблеми**: питання кодування туру і алгоритм кросовера (**схрещення)**, який використовується, щоб об'єднати двох батьків для отримання нащадків.

У стандартному генетичному алгоритмі кросовер забезпечується випадковим вибором позиції в батьківській послідовності і обміном частин батьків. У цьому прикладі, точка перетину – між 3-м і 4-м пунктом списку.

Батько 1	FABECGD
Батько 2	DEACGBF
1 Дитина	FABCGBF
2 Дитина	DEAECGD

Складність завдання комівояжера полягає в тому, що відвідати кожне місто в турі можна лише **один раз**. Якщо літери в наведеному вище прикладі представляють міста, тури-нащадки, створені за цим кросовером, будуть вважатися недійсними. 1-й нащадок їде в місто F і B два рази, і ніколи не заїде в міста D або E.

Кодування не може бути просто списком міст у порядку, як вони були відвідані. Для цього створені інші методи кодування. Хоча ці методи не будуть створювати недійсні тури, вони не беруть до уваги той факт, що тур «ABCDEFG»

такий же, як «GFEDCBA». Щоб належно вирішити цю проблему, кросовер алгоритму повинен бути складніше.

Це рішення зберігає посилання в обох напрямках для кожного туру. У наведеному вище прикладі про тури, новий батько 1 буде збережений як:

Місто	Перша вершина	Друга вершина
A	F	B
B	A	E
C	E	G
D	G	F
E	B	C
F	D	A
G	C	D

Кросовер операції набагато складніше, ніж поєднання двох рядків. Кросовер буде використовувати всі посилання, які є в обох батьків, і місце цих зв'язків в обох нащадках:

- для Нащадка 1, вибір посилання буде у виборі серед Батька 2, а потім Батька 1;
- для нащадка 2, вибір складається з Батька 2 і Батька 1 з іншим набором посилань.

Для будь-якого нащадка є ймовірність того, що посилання може створити недійсними тур, де замість одного шляху в турі буде створено багато розірваних турів. Ці посилання повинні бути відкинуті. Для заповнення решти відсутніх ланок, міста вибираються випадково. Оскільки кросовер **не** є цілком випадковим, цей кросовер буде називатися жадібним.

Зрештою, ГА зробить так, що всі рішення будуть **ідентичними.** Це не є ідеальним рішенням. Якщо всі особини в ГА стануть ідентичними, ГА не зможе **покращити** рішення. Є два способи обійти це.

Перший полягає у використанні дуже великих початкових розмірів популяції, так, що ГА буде потрібно багато часу, щоб все особини стали ідентичними.

Другий метод – мутація, коли випадкові нащадки мутують, створюючи новий унікальний тур.

Цей генетичний алгоритм також використовує жадібні обчислення при розрахунку початкової популяції. Посилання на міста в первісному турі **не є** повністю випадковими. ГА воліє робити посилання між містами, які близькі один до одного. Це робиться на **кожному кроці**, інакше це зробило б всіх особин ідентичними.

Встановлено 6 параметрів для управління роботою генетичного алгоритму:

- **чисельність населення (Population Size)** – чисельність популяції початкового числа випадкових турів, які створюються, коли алгоритм починає роботу. Велика кількість особин вимагає більше часу для знаходження

результату. Менша кількість особин збільшується ймовірність того, що кожен тур до певної міри буде заміщати іншого і бути ідентичним. Це збільшує ймовірність того, що краще рішення не буде знайдено;

- **район / розмір групи (Neighborhood/Group Size)** – кожне покоління, це кількість турів, випадково обраних з популяції. 2 кращих тури будуть батьками, гірші 2 тури будуть замінені їх нащадками. Для великого розміру групи зростає ймовірність того, що як батькии будуть обрані дійсно хороші тури, але це також призводить до того, що багато турів ніколи не будуть використовуватися як батьки. Великий розмір групи приводить до швидшої роботи алгоритму, але він не може знайти найкраще рішення;
- **мутація % (Mutation %)** – відсоток, що кожний нащадок після кросовера буде піддаватися мутації. Якщо до туру застосовано мутацію, тоді одне місто випадково пересувається з однієї точки в турі на інше;
- **# довколишніх міст (# Nearby Cities)** – як частина жадібної початкової популяції, генетичний алгоритм під час формування початкових турів надає перевагу зв'язці міст, які знаходяться близько один до одного. Під час створення початкової популяції – це число міст, які вважаються близькими;
- **поряд міста коефіцієнти % (Nearby City Odds)** – це відсоткова ймовірність, що будь-яке місто в випадковому турі в початковій популяції зволіють використовувати навколишні, а не абсолютно випадкові міста. Якщо генетичний алгоритм вибирає для використання навколишні міста, тобто в рівній мірі випадковість, що це буде одне з міст, порівняно з попереднім параметром;
- **максимум поколінь (Maximum Generations)** – скільки разів повторюється кросовер до завершення алгоритму.

Інші варіанти, які можуть бути налаштовані є (примітка: дані доступні тільки в завантажуваній версії):

- **випадкове ядро (Random Seed) –** це ядро для генератора випадкових чисел. Маючи фіксоване, а не випадкове ядро, можна повторити попередні результати з тими ж значеннями інших параметрів. Це корисно для пошуку помилок в алгоритмі;
- **перелік Міст (City List)** – завантаження файлу, що дозволяє імпортувати списки з міста XML-файлів. Під час налагодження проблем корисно мати можливість запускати алгоритм з тими ж точними параметрами.

Початкові значення параметра:

Параметр	Початкове значення
Розмір популяції	10,000
Розмір групи	5
Мутації	3 %
Кількість сусідніх вершин	5
Коефіцієнти сусідніх міст	90 %

Завдання № 2

Створити математичну модель та реалізувати демонстраційну програму оптимізації функції двох змінних зі складною областю визначення методом генетичного алгоритму з такими функціями:

1. Введення параметрів генетичного алгоритму (розмір початкової популяції, кількість мутацій, кількість пар для кросовера).

2. Введення системи обмежень для створення не опуклої області визначення. Відобразити область визначення в тривимірній системі координат.

3. Створення початкової популяції та відображення її екземплярів у тривимірній системі координат.

Для поточного кроку регенерації показати:

- значення цільової функції кожного екземпляра популяції;
- екземпляри поточної вибірки;
- нові отримані екземпляри;
- провести операції кросовера та мутації, показати нові екземпляри популяції;
- провести селекцію, вилучити неконкурентних екземплярів;
- на кожному кроці регенерації показати у впорядкованому порядку числові значення екземплярів популяції та їхню оцінку та відобразити на зображенні та у числовому вигляді розв'язок задачі оптимізації.

Примітка. Для кращої демонстрації процесів, що відбуваються у генетичному алгоритмі, рекомендовано вибрати функцію, яка у області визначення має декілька точок локального мінімуму.

Наприклад, $z = (x-2)^2 + (y-3)^2 + 4$.

Контрольні запитання

1. Дайте визначення генетичного алгоритму.
2. Опишіть структуру генетичного алгоритму.
3. Назвіть етапи генетичного алгоритму.
4. Опишіть основні кроки у розв'язанні задачі комівояжера за допомогою генетичного алгоритму.

ПРАКТИЧНА РОБОТА № 9
НЕЧІТКІ МНОЖИНИ В ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЯХ

Мета роботи – вивчити методи розв'язування задач нечіткого математичного програмування.

Завдання для підготовки до виконання роботи

Розв'язати задачу нечіткої оптимізації, використовуючи методи багатокритеріальної оптимізації.

Таблиця 9.1 – Варіанти завдань

Номер варіанта	Система	Номер варіанта	Система
1	2	3	4
1	$f(x_1, x_2)=2x_1 + 5x_2 \to max;$ $x_1 + x_2 \geq\approx 4,$ $4x_1 + 6x_2 \leq\approx 24,$ $3x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$	16	$f(x_1, x_2)=x_1 - 5x_2 \to min;$ $x_1 + x_2 \geq\approx 4,$ $4x_1 + 6x_2 \leq\approx 24,$ $3x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$
2	$f(x_1, x_2)=x_1 - x_2 \to max;$ $x_1 + 2x_2 \geq\approx 2,$ $2x_1 + 5x_2 \leq\approx 10,$ $12x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$	17	$f(x_1, x_2)=2x_1 + 3x_2 \to min;$ $x_1 + 2x_2 \geq\approx 2,$ $2x_1 + 5x_2 \leq\approx 10,$ $12x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$
3	$f(x_1, x_2)=2x_1 + 5x_2 \to max;$ $2x_1 + 3x_2 \geq\approx 6,$ $x_1 + 6x_2 \leq\approx 18,$ $2x_1 + 3x_2 \leq\approx 12,$ $x_1, x_2 \geq 0.$	18	$f(x_1, x_2)=2x_1 + x_2 \to max;$ $x_1 + 2x_2 \geq\approx 2,$ $2x_1 + 5x_2 \leq\approx 10,$ $12x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$
4	$f(x_1, x_2)=2x_1 + 5x_2 \to max;$ $x_1 + x_2 \geq\approx 4,$ $4x_1 + 6x_2 \leq\approx 24,$ $3x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$	19	$f(x_1, x_2)=2x_1 + x_2 \to min;$ $3x_1 + x_2 \geq\approx 4,$ $x_1 - 6x_2 \leq\approx 24,$ $3x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$
5	$f(x_1, x_2)=10x_1 - x_2 \to max;$ $x_1 + 2x_2 \geq\approx 2,$ $2x_1 + 5x_2 \leq\approx 10,$ $12x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$	20	$f(x_1, x_2)=12x_1 - 3x_2 \to max;$ $2x_1 + 3x_2 \geq\approx 6,$ $2x_1 + 6x_2 \leq\approx 18,$ $2x_1 + x_2 \leq\approx 12,$ $x_1, x_2 \geq 0.$
7	$f(x_1, x_2)=x_1 - x_2 \to max;$ $x_1 + 2x_2 \geq\approx 2,$ $2x_1 + 5x_2 \leq\approx 10,$ $12x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$	22	$f(x_1, x_2)=2x_1 + 3x_2 \to max;$ $-3x_1 + 2x_2 \geq\approx 2,$ $2x_1 + 5x_2 \leq\approx 10,$ $2x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$

Продовження таблиці 9.1

1	2	3	4
8	$f(x_1, x_2)=2x_1 + 3x_2 \to max;$ $5x_1 + 2x_2 \geq\approx 2,$ $2x_1 + 5x_2 \leq\approx 10,$ $3x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$	23	$f(x_1, x_2)=7x_1 - 5x_2 \to max;$ $2x_1 + x_2 \geq\approx 2,$ $4x_1 + 6x_2 \leq\approx 24,$ $3x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$
9	$f(x_1, x_2)=2x_1 + x_2 \to min;$ $3x_1 + x_2 \geq\approx 4,$ $x_1 - 6x_2 \leq\approx 24,$ $3x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$	24	$f(x_1, x_2)=10x_1 - x_2 \to max;$ $x_1 + 2x_2 \geq\approx 2,$ $2x_1 + 5x_2 \leq\approx 10,$ $12x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$
10	$f(x_1, x_2)=12x_1 - 3x_2 \to max;$ $2x_1 + 3x_2 \geq\approx 6,$ $2x_1 + 6x_2 \leq\approx 18,$ $2x_1 + x_2 \leq\approx 12,$ $x_1, x_2 \geq 0.$	25	$(x_1, x_2)=2x_1 + x_2 \to min;$ $3x_1 + x_2 \geq\approx 4,$ $x_1 - 6x_2 \leq\approx 24,$ $3x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$
11	$f(x_1, x_2)=10x_1 - x_2 \to max;$ $x_1 + 2x_2 \geq\approx 2,$ $2x_1 + 5x_2 \leq\approx 10,$ $12x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$	26	$f(x_1, x_2)=2x_1 + 3x_2 \to min;$ $x_1 + 2x_2 \geq\approx 2,$ $2x_1 + 5x_2 \leq\approx 10,$ $12x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$
2	$f(x_1, x_2)=x_1 - 5x_2 \to min;$ $x_1 + 4x_2 \geq\approx 4,$ $4x_1 + 6x_2 \leq\approx 24,$ $3x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$	27	$f(x_1, x_2)=2x_1 - 5x_2 \to max;$ $x_1 + x_2 \geq\approx 4,$ $2x_1 + 5x_2 \leq\approx 24,$ $12x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0$
13	$f(x_1, x_2)=2x_1 + x_2 \to min;$ $3x_1 + x_2 \geq\approx 4,$ $x_1 - 6x_2 \leq\approx 24,$ $3x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$	28	$f(x_1, x_2)=7x_1 - 5x_2 \to max;$ $2x_1 + x_2 \geq\approx 2,$ $4x_1 + 6x_2 \leq\approx 24,$ $3x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$

Закінчення таблиці 9.1

1	2	3	4
14	$f(x_1, x_2)=12x_1 - 3x_2 \to max;$ $2x_1 + 3x_2 \geq\approx 6,$ $2x_1 + 6x_2 \leq\approx 18,$ $2x_1 + x_2 \leq\approx 12,$ $x_1, x_2 \geq 0.$	29	$f(x_1, x_2)=2x_1 + x_2 \to min;$ $3x_1 + x_2 \geq\approx 4,$ $x_1 - 6x_2 \leq\approx 24,$ $3x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$
15	$f(x_1, x_2)=10x_1 - x_2 \to max;$ $x_1 + 2x_2 \geq\approx 2,$ $2x_1 + 5x_2 \leq\approx 10,$ $12x_1 + 8x_2 \leq\approx 24,$ $x_1, x_2 \geq 0.$	30	$f(x_1, x_2)=12x_1 - 3x_2 \to max;$ $2x_1 + 3x_2 \geq\approx 6,$ $2x_1 + 6x_2 \leq\approx 18,$ $2x_1 + x_2 \leq\approx 12,$ $x_1, x_2 \geq 0.$

Загальні відомості

Стандартна задача математичного програмування зазвичай полягає в максимізації (або мінімізації) заданої функції на цій множині допустимих альтернатив, яку описано системою нерівностей [8]. Наприклад,

$$\begin{gathered} f(x) \to \max, \\ \varphi_i(x) \leq 0, \quad i = 1, \ldots, m, \\ x \in X, \end{gathered} \tag{9.1}$$

де X – задана множина альтернатив, $f : X \to R^1$ й $\varphi_i : X \to R^1$, $i = 1,..., m$ – задані функції.

Моделюючи в такій формі реальні задачі, дослідник часто може мати у своєму розпорядженні лише нечіткі описи функцій f і φ або їхніх параметрів, нечітко може бути описана й множина альтернатив X. Нечіткий опис ситуації ухвалення рішень може, наприклад, відображати неадекватність інформації про цю ситуацію або бути формою наближеного опису, достатнього для розв'язування певної задачі.

Форми нечіткого опису інформації можуть бути різними, звідси й походить відмінність у математичних постановках відповідних задач *нечіткого математичного програмування* (НМП) з використанням поняття нечітких множин.

Нечітка множина (Fuzzy Set) − розширення класичної (чіткої) множини, де елемент може належати до множини не лише повністю (1) або не належати зовсім (0), а й мати певний ступінь належності у діапазоні [0,1][0,1]. Нечітка множина – це математична концепція, що дозволяє елементам належати множині з певним ступенем. Формально нечітка множина задається через функцію належності μA(x)\mu_A(x), де: μA(x)∈[0,1]\mu_A(x) \in [0,1].

Розглянемо задачу математичного програмування із нечіткими обмеженнями.

На відміну від стандартної задачі (9.1), їй характерні «пом'якшені» обмеження, тобто припускається можливість деякого їхнього спрощення.

Крім того, замість максимізації функції $f(x)$, можна ставити за мету досягнення її певного фіксованого значення, причому різним відхиленням $f(x)$ від цієї величини належить приписувати різні ступені допустимості (наприклад, чим більше відхилення, тим меншим буде ступінь його допустимості). Нечітку задачу при цьому можна записати так:

$$f(x) \gtrsim z_0,$$
$$\varphi(x) \lesssim 0,$$
$$x \in X,$$

тут символ **~** означає нечіткість поданої нерівності.

Опишемо підходи до розв'язання цієї задачі.

1. *Зведення до задачі нечітко визначеної мети.*

Множини мети й обмежень, використовуючи такі функції належності, становлять такі співвідношення:

$$\mu_G(x) = \begin{cases} 0, & f(x) \le z_0 - a, \\ \mu(x,a), & z_0 - a < f(x) < z_0, \\ 1, & f(x) \ge z_0, \end{cases}$$

$$\mu_C(x) = \begin{cases} 0, & \varphi(x) \ge b, \\ \nu(x,b), & 0 < \varphi(x) < b, \\ 1, & \varphi(x) \le 0, \end{cases}$$

де $\mu(x, a)$ та $\nu(x, b)$ задані функції, які описують міру виконання відповідних нерівностей з погляду ОПР та з урахуванням умов конкретної задачі ухвалення рішень, а саме $\mu : X \to [0, 1]$ та $\nu : X \to [0, 1]$.

Отже, вихідну задачу буде сформульовано у вигляді задачі досягнення нечітко визначеної мети, до якої можна застосувати підхід Белмана – Заде.

2. *Розкладання нечіткої множини допустимих альтернатив на множини рівня*. Цей підхід полягає в тому, що вихідну задачу нечіткого математичного програмування формулюють у вигляді сукупності звичайних задач максимізації функції на всіляких множинах рівня множини допустимих альтернатив. Якщо альтернатива $x_0 \in X$ є розв'язком задачі: $\varphi(x) \to max$ на множині рівня λ, то природно вважати, що її ступінь належності до нечіткої множини розв'язків задачі не менший від числа λ. Отже, перебравши всілякі значення λ, ми одержимо функцію належності нечіткого розв'язку.

Для побудови функції належності нечіткого розв'язку необхідно кожній альтернативі $x \in X$ приписати ступінь належності до цієї множини. Зробимо це таким чином: ступенем належності альтернативи x_0 до нечіткої множини розв'язків будемо вважати максимальне (точніше верхню границю) з чисел λ, для яких відповідна множина $N(\lambda)$ містить альтернативу x_0.

Розв’язком задачі НМП будемо називати нечітку підмножину μ_D, яка описується такою функцією належності:

$$\mu^1(x) = \sup_{\lambda > x \epsilon N(\lambda)} \lambda$$

Зауважимо, що коли функція належності неперервна, то застосування цього підходу вимагає розгляду нескінченної кількості задач, оскільки нескінченною буде кількість множин рівня. Однак для практичного використання буде достатньо розглянути скінченну множину задач для множин рівня, визначених експертами або ОПР.

3. *Зведення до багатокритеріальної задачі*. Для цього підходу характерно те, що із самого початку явно враховується прагнення ОПР при виборі альтернативи отримати якомога більші значення як максимизованої функції, так і функції належності нечіткої множини допустимих альтернатив.

З цією метою у розв’язок задачі включають лише ті альтернативи, які в задачах багатокритеріальної оптимізації називаються ефективними за Парето [1], тобто розв’язують задачу такого вигляду:

$$\varphi(x) \to max,$$
$$\mu_C(x) \to \max,$$
$$x_0 \epsilon X.$$

Приклад. Розв’язати таку задачу нечіткого математичного програмування:

$$f(x_1, x_2) = 2x_1 + 5x_2 \to max,$$
$$2x_1 + 3x_2 \geq 6,$$
$$x_1 + 6x_2 \leq 18,$$
$$2x_1 + 3x_2 \leq\approx 12,$$
$$x_1, x_2 \geq 0.$$

Порядок виконання роботи

1. Будемо вважати, що нечіткі обмеження описуються такою функцією належності:

$$\mu(x) = 0, \text{ якщо } 2x_1 + 3x_2 \geq 14,$$
$$\mu(x) = 0{,}5 \text{ якщо } 13 \leq 2x_1 + 3x_2 \leq 14,$$
$$\mu(x) = 0{,}7 \text{ якщо } 12 \leq 2x_1 + 3x_2 \leq 13,$$
$$\mu(x) = 1 \text{ якщо } 2x_1 + 3x_2 \leq 12.$$

2. Використовується метод розкладання на множини рівня. Враховуючи вигляд функції належності обмежень, зробіть висновок, що необхідно розв’язувати задачі на таких множинах рівня: $\lambda_1 = 1$; $\lambda_2 = 0{,}7$; та $\lambda_3 = 0{,}5$.

Якщо $\lambda_1 = 1$, то задача набуває такого вигляду:

$$f(x_1, x_2)=2x_1 + 5x_2 \to max;$$

$$2x_1 + 3x_2 \geq\approx 6,$$

$$x_1 + 6x_2 \leq\approx 18,$$

$$2x_1 + 3x_2 \leq\approx 12,$$

$$x_1, x_2 \geq 0.$$ Введіть тут рівняння.

Коли $\lambda_2 = 0{,}7$, то маємо таку задачу:

$$f(x_1, x_2)=2x_1 + 5x_2 \to max;$$

$$2x_1 + 3x_2 \geq\approx 6,$$

$$x_1 + 6x_2 \leq\approx 18,$$

$$2x_1 + 3x_2 \leq\approx 13,$$

$$x_1, x_2 \geq 0.$$

І якщо $\lambda_3 = 0{,}5$, то задачу буде записано в такому вигляді:

$$f(x_1, x_2)=2x_1 + 5x_2 \to max;$$
$$2x_1 + 3x_2 \geq\approx 6,$$
$$x_1 + 6x_2 \leq\approx 18,$$
$$2x_1 + 3x_2 \leq\approx 14,$$
$$x_1, x_2 \geq 0.$$

Оскільки вихідна задача була лінійною, у результаті розкладання ми також отримали задачі лінійного програмування і для їхнього розв'язування можна застосовувати, наприклад, симплекс-метод, або, оскільки кожна з них має лише дві змінні, розв'язати їх графічно.

З цією метою зобразимо на координатній площині множини рівня нечіткої множини допустимих альтернатив (рис. 9.1).

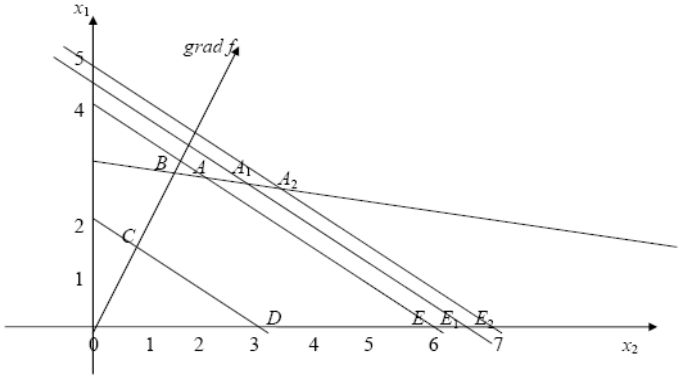


Рисунок 9.1 – Графічна інтерпретація розв'язування задачі нечіткого математичного програмування

Багатокутник *ABCDE* відповідає множині рівня $\lambda_1 = 1$; багатокутник A_1DCDE_1 – множині рівня: $\lambda_2 = 0{,}7$; а A_2CDE_2 – множині рівня: $\lambda_3 = 0{,}5$.

Розв'язком задачі на множині рівня 1 буде точка *A*. Знайдемо її координати з такої системи рівнянь:

$$\begin{cases} x_1 + 6x_2 = 18, \\ 2x_1 + 3x_2 = 12. \end{cases}$$

Отримуємо такий результат: $x_1 = 2$, $x_2 = 2\frac{2}{3}$. Значення цільової функції в цій точці $f(A) = 17\frac{1}{3}$.

Аналогічно знаходимо розв'язки задач на множинах рівня 0,7 та 0,5. Це точки $A1$ (вона має координати: $x_1 = 2\frac{2}{3}$, $x_2 = 2\frac{5}{9}$) та A_2 (її координати: $x_1 = 3\frac{1}{3}$, $x_2 = 2\frac{4}{9}$).

Цим розв'язкам відповідають такі цільові функції: $f(A_1) = 18\frac{1}{9}$, $f(A_2) = 18\frac{2}{3}$. Об'єднавши результати в таблицю, маємо нечіткий розв'язок задачі (табл. 9.2).

Таблиця 9.2 – Результати розв'язування задачі

x_1	x_2	f	
2	$2\frac{2}{3}$	$17\frac{1}{3}$	1
$2\frac{2}{3}$	$2\frac{5}{9}$	$18\frac{1}{9}$	0,7
$3\frac{1}{3}$	$2\frac{4}{9}$	$18\frac{2}{3}$	0,5

Контрольні запитання

1. Сформулюйте загальну постановку задачі нечіткого математичного програмування.
2. Як класифікують задачі нечіткого математичного програмування?
3. Які підходи застосовують до розв'язування задач НМП?
4. У чому полягає сутність методу зведення до задачі досягнення нечітко визначеної мети під час розв'язання задач НМП?
5. У чому полягає сутність методу розкладання на множини рівня при розв'язувані задач НМП?
6. Розкрийте суть методу модальних значень у застосуванні до задач НМП?
7. Опишіть застосування методу зведення до багатокритеріальної задачі при розв'язуванні задач НМП?

ПРАКТИЧНА РОБОТА № 10
ФОРМУВАННЯ СИСТЕМ ІЗ НЕЧІТКОЮ ЛОГІКОЮ В СЕРЕДОВИЩІ MATLAB FUZZY LOGIC TOOLBOX

Мета − ознайомитися з принципом формування систем із нечіткою логікою в середовищі Matlab Fuzzy Logic Toolbox.

Завдання − спроєктувати систему ухвалення рішень нечіткої логіки, обравши вхідні і вихідні параметри використанням відповідних функцій приналежності вхідних даних та набором правил для налаштування нечітких висновків.

Таблиця 10.1 – Варіанти до виконання роботи

Номер варіанта	Система	Вхідні параметри	Вихідні параметри
1	2	3	4
1, 7,14, 28	Антиблокувальна система гальм (ABS)	швидкість обертання кожного колеса (датчики швидкості); тиск у гальмівній системі; кут повороту керма; прискорення / уповільнення автомобіля; дорожні умови (наприклад, датчики зчеплення)	регулювання тиску в гальмівній системі; попереджувальні сигнали для водія активація / деактивація ABS
2, 8,15, 29	Круїз-контроль	поточна швидкість автомобіля; встановлена бажана швидкість; кут нахилу дороги (залежно від датчиків); відстань до інших автомобілів (якщо адаптивний круїз-контроль)	регулювання подачі пального; контроль дросельної заслінки; можливе гальмування для підтримки дистанції
3, 9,16, 30	Розумна пральна машина	тип тканини; рівень забруднення; об’єм та вага білизни; температура води; наявність мийних засобів	тривалість циклу прання; витрати води та електроенергії; швидкість віджиму; рекомендації користувачеві (наприклад, про необхідність чищення барабана)

Продовження таблиці 10.1

1	2	3	4
4, 10,17, 27	Медична діагностика	симптоми пацієнта; дані аналізів (кров, сеча, рентген, тощо); історія хвороби; параметри життєдіяльності (пульс, тиск, температура, рівень кисню)	попередній діагноз; рекомендації щодо лікування; рівень ризику захворювань; необхідність додаткових обстежень
5, 11,18, 26	Кондиціонери та клімат-контроль	температура всередині приміщення / автомобіля; температура зовнішнього середовища; рівень вологості; бажана температура, встановлена користувачем; наявність людей у приміщенні (для розумних систем)	регулювання швидкості вентилятора; вмикання / вимикання компресора; контроль вологості повітря;
6, 12,19, 25	Оптимізація світлофорів залежно від трафіку	інтенсивність руху на перехрестях (камери, датчики); час доби; наявність пішоходів; аварійні ситуації; пріоритетний транспорт (швидка, пожежна, громадський транспорт)	тривалість сигналів світлофора; зміна режиму роботи світлофорів; пріоритетне пропускання спецтранспорту
7, 13,21, 24	Оцінка кредитоспроможності клієнтів на основі нечітких факторів	доходи клієнта; кредитна історія; наявність поточних кредитів; соціальні фактори (тип зайнятості, освіта); наявність майна або застави	рівень кредитного ризику; максимальна сума кредиту; відсоткова ставка; імовірність схвалення заявки

Загальні відомості

Під час розроблення інтелектуальних систем знання про конкретну предметну область, для якої створюється система, не часто бувають повними й абсолютно достовірними. Навіть кількісні дані, отримані шляхом достатньо точних експериментів, мають статистичні оцінки вірогідності, надійності, значущості тощо. Інформація, якою заповнюються експертні системи, отримується у результаті опитування експертів, думки яких є суб’єктивними і можуть розходитися. Поряд із кількісними характеристиками в базах знань інтелектуальних систем повинні зберігатися якісні показники, евристичні правила, текстові знання і т. д. При обробці знань із застосуванням твердих механізмів формальної логіки виникає протиріччя між нечіткими знаннями і чіткими методами логічного виведення. Розв’язати це протиріччя можна шляхом

подолання нечіткості знань (коли це можливо) або використанням спеціальних методів подання й обробки нечітких знань.

Системи нечіткої логіки **(Fuzzy Logic)** широко використовуються там, де класична булева логіка не справляється через невизначеність або неточність даних. Це логічний підхід, який використовує поняття нечітких множин для моделювання ситуацій, де істина може бути не лише «так» або «ні», а приймати **проміжні значення**. Нечітка логіка використовує нечіткі множини для обробки невизначених і розмитих даних. Нечітка логіка є узагальненням класичної двозначної логіки, де логічні вислови можуть мати **ступінь істинності** від 0 до 1 і використовуються **нечіткі оператори** (НЕ, І, АБО) з нечіткими числами.

У середовищі Matlab одним з пакетів прикладних інженерних і математичних програм є Fuzzy Logic Toolbox для проєктування і дослідження систем на нечіткій логіці (рис. 10.1).

Для виклику вікна для роботи необхідно в командному вікні ввести функцію «fuzzy».

У візуальній формі в цьому інтерфейсі представлено входи і виходи системи нечіткої логіки, у центрі якого наведено процесор нечітких правил.

Кожну вхідну та вихідну змінну можна редагувати, а саме визначати функції приналежності.

За допомогою пункту «Edit – Add Variable» можна додати більше входів та виходів.

Функція «Edit – Membership Function» – викликає редактор функції приналежності.

Функція «Edit – Rules» – викликає редактор правил нечіткого виводу. Пункт меню View містить такі операції:

- Rules – викликає програму перегляду правил нечіткого виводу;
- Surface – викликає програму перегляду поверхні нечіткого виводу.

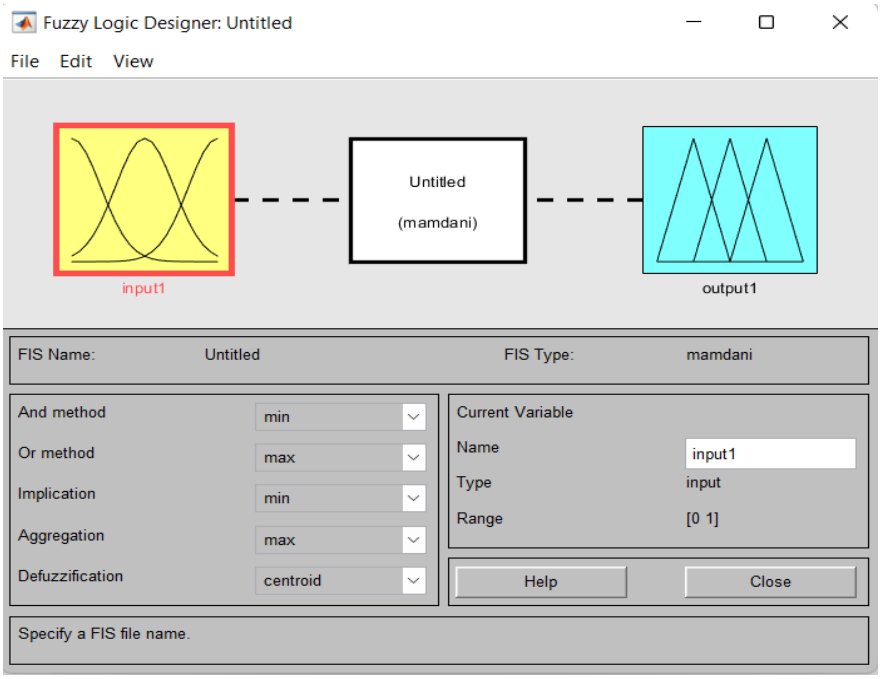


Рисунок 10.1 – Fuzzy Logic Toolbox [5]

У лівому нижньому кутку робочого інтерфейсу редактору FIS маємо 5 вікон:

And method – дозволяє виконати один з таких методів для логічної кон'юнкції в умовах нечітких правил:

- Min – метод мінімального значення;
- Prod – метод алгебраїчного добутку;
- Custom – метод, визначений користувачем.

Or method – дозволяє виконати один з таких методів для логічної диз'юнкції в умовах нечітких правил:

- Max – метод максимального значення;
- Probor – метод алгебраїчної суми;
- Custom – метод, визначений користувачем.

Implication method – дозволяє виконати один з таких методів для логічного заключення в кожному з нечітких правил:

- Min – метод мінімального значення;
- Prod – метод алгебраїчного добутку;
- Custom – метод, визначений користувачем.

Aggregation method – дозволяє виконати один з таких методів для агрегування значень функції приналежності кожної з вихідних змінних при заключенні нечітких правил:

- Max – метод максимального значення;
- Sum – метод граничної суми;
- Probor – метод алгебраїчної суми;
- Custom – метод, визначений користувачем.

Defuzzification method – дозволяє виконати один з таких методів для деффазифікації вихідних змінних в системі нечіткого виводу типу Мамдані:

- Centroid – метод центру ваги для дискретної множини значень функції приналежності;
- Bisector – метод центра площини;
- Mom – метод середнього максимуму, як середнє арифметичне лівого та правого модальних значень;
- Som – метод найменшого (лівого) модального значення;
- Lom – метод найбільшого (правого) модального значення;
- Custom – метод, визначений користувачем.

Приклад. Задано дві вхідні змінні нечіткої системи контролю температури води в крані – похибка температури (temp) і похибка її витрат (flow) (рис. 10.2). Кожен вхід описується трьома функціями приналежності (рис. 10.3). Двома виходами нечіткої системи є швидкість, з якою клапани холодної та гарячої води відкриваються або закриваються (cold та hot, відповідно). Кожен вихід має п'ять входів.

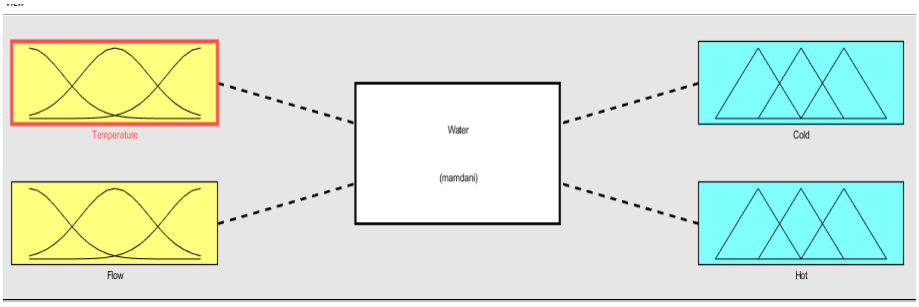
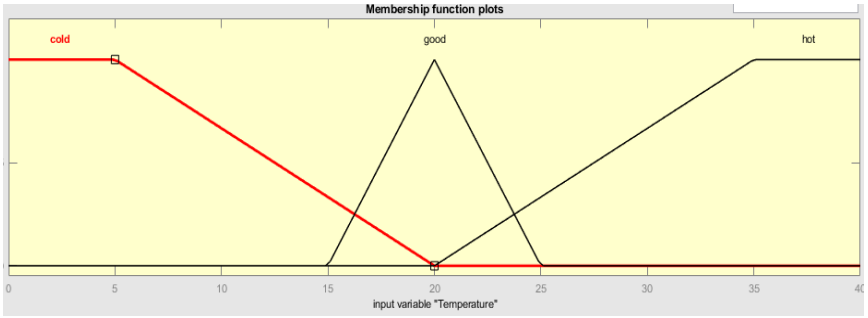
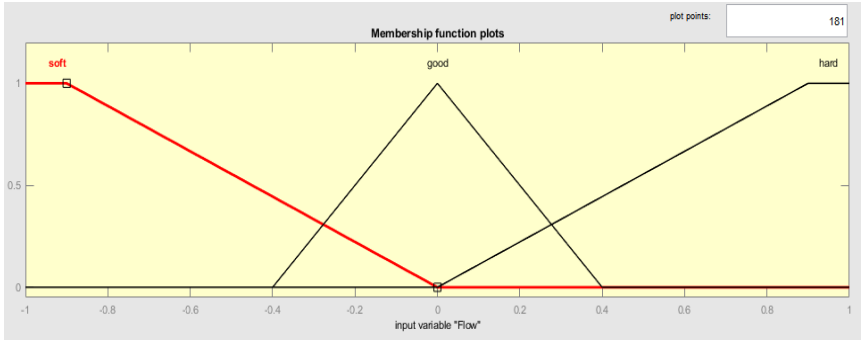


Рисунок 10.2 – Загальна структура системи нечіткої логіки регулювання температури води [5]



а



б

Рисунок 10.3 – Функції приналежності вхідних змінних: ((а) – температури, (б) – швидкості потоку води) [5]

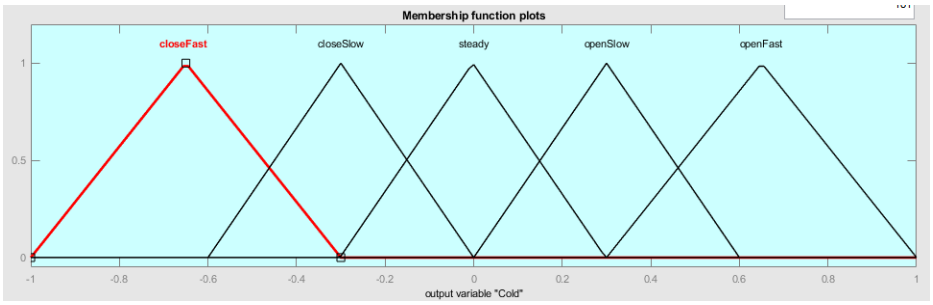


Рисунок 10.4 – Функції приналежності вихідних змінних [5]

Нечітка система має дев’ять правил для регулювання клапанів гарячої та холодної води на основі похибок витрат та температури (рис. 10.5). Правила регулюють загальну швидкість потоку на основі похибки потоку та регулюють

відносні швидкості гарячого та холодного потоку на основі температурної похибки.

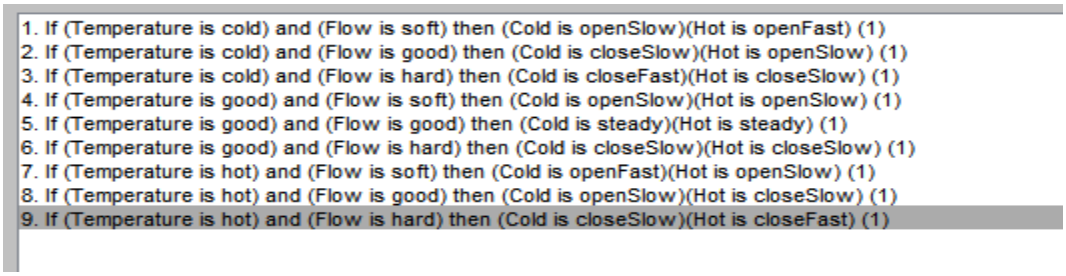
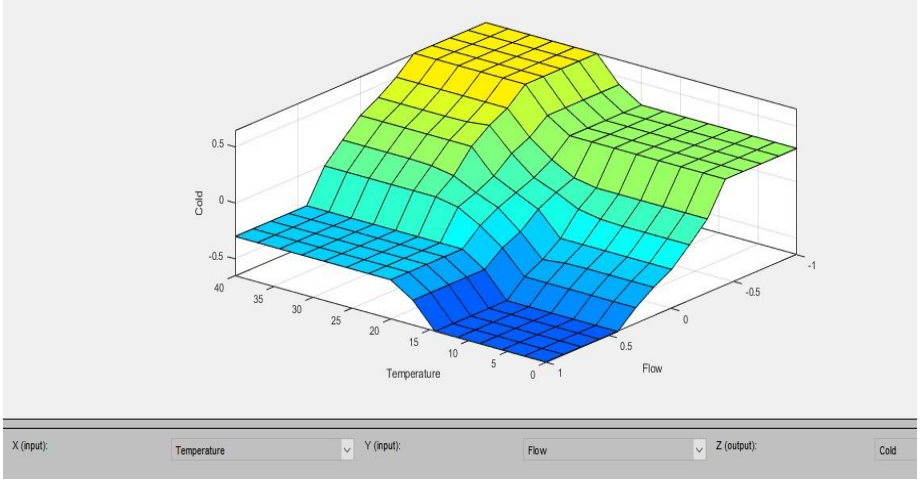
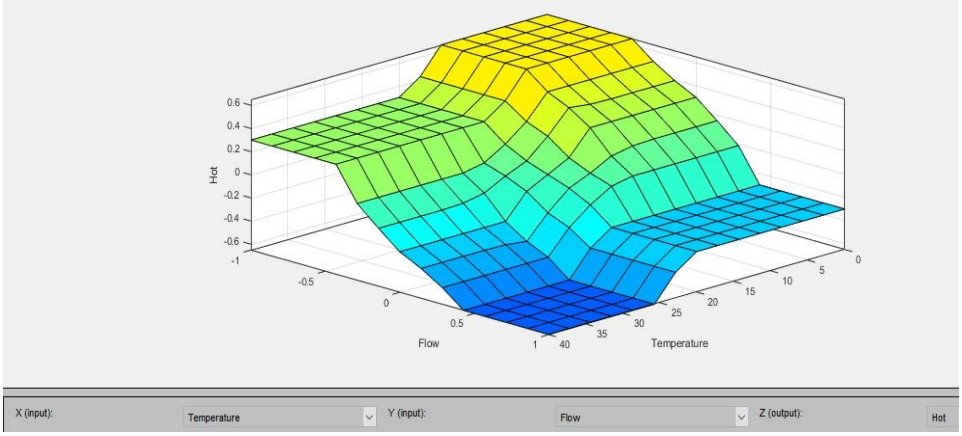


Рисунок 10.5 – Правила для визначення вихідних змінних системи нечіткої логіки

Результат роботи системи для кожної вихідної змінної можна оцінити за допомогою відповідного графіка розподілу вхідних даних. Для прикладу контролю температури води в крані для потоку холодної та гарячої води відповідний розподіл подано на рисунку 10.6.



а



б

Рисунок 10.6 – Графік розподілу вхідних даних як результат роботи системи нечіткої логіки – для регулювання води [5]:
((а) – холодної та (б) – гарячої)

Так можна побачити, що у нижньому куті при максимальній силі потоку холодної води (синій колір на графіку), її потреба у крані мінімальна, а у верхньому куті при інтенсивній подачі гарячої води, потреба у холодній для виконання балансу температур максимальна (жовтий колір на графіку). В центрі графіку спостерігаємо баланс температур.

Порядок виконання роботи

1. Задати вхідні параметри системи.
2. Задати функції приналежності кожного вхідного параметра з відповідними пороговими значеннями.
3. Задати вихідні параметри системи.
4. Записати відповідні правила для ухвалення рішень системи нечіткої логіки.
5. Отримати результати роботи системи для кожного виходу у вигляді графіку розподілу вхідних параметрів та зробити висновки.

Контрольні запитання

1. Опишіть принципом формування систем із нечіткою логікою в середовищі Matlab Fuzzy Logic Toolbox.
2. Як визначаються правила вхідних і вихідних змінних системи нечіткої логіки?
3. Як задати функції приналежності для кожного вхідного параметра з відповідними пороговими значеннями?
4. Сформулюйте правила для ухвалення рішень системи нечіткої логіки.

ПРАКТИЧНА РОБОТА № 11
РОЗПІЗНАВАННЯ ОБЛИЧЧЯ

Мета роботи − навчитися створювати програми для розпізнавання обличчя мовою Python за допомогою бібліотеки OpenCV.

Завдання − виконати ідентифікацію довільної особи.

Загальні відомості

Автоматична ідентифікація особистості людини по її зображенню на фотографії або в відеопотоці має широке комерційне та наукове застосування. Ця тематика з’явилася на початку 1980-х років, однак її бурхливий розвиток почалося в 1990-х, після створення нових технологій у сфері обробки зображень та обчислювальних машин. Ця технологія становить особливий інтерес у зв’язку з тим, що може відбуватися безконтактно.

На сьогодні збільшився інтерес до різних методів ідентифікації осіб. Для вирішення цього завдання необхідно виконати два етапи: виявити особу на фотографії, виділити і розпізнати її [6].

При всьому різноманітті різних алгоритмів і методів розпізнавання зображень, типовий метод розпізнавання складається з трьох компонентів:

- перетворення вихідного зображення в початкове уявлення (може містити як попередню обробку, так і математичні перетворення, наприклад обчислення головних компонент);
- виділення ключових характеристик (наприклад береться перші n головних компонент або коефіцієнтів дискретного косинусного перетворення);
- механізм класифікації (моделювання): кластерна модель, метрика, нейронна мережа тощо.

Крім цього, побудова методу розпізнавання спирається на апріорну інформацію про предметну область (у цьому випадку – характеристики особи людини), і коригується експериментальної інформацією, що з'являється по ходу розробки методу.

Налаштування програмного середовища. Для початку необхідно завантажити та інсталювати IDE PyChar. Після чого необхідно створити новий проєкт. Далі необхідно інсталювати бібліотеку opencv. Для цього у терміналі програми вводимо команду:

```
pip install opencv-python
```

Далі у вікні програми імпортуємо цю бібліотеку командою:

```
import cv2
```

Враховуючи те, що opencv – безкоштовна бібліотека з відкритим кодом, то багато класифікаторів можна знайти на github. Переходимо за посиланням: URL: https://github.com/opencv/opencv та відкриваємо папку data, там же обираємо вже існуючі класифікатори.

Для нашого завдання оберіть з папки haarcascade класифікатор визначення обличчя:

haarcascade_frontalface_default.xml.

Далі необхідно завантажити цей файл та перенести його до поточного проєкту за шляхом:

```
Lib\site-packages\cv2\data.
```

Після чого необхідно створити об'єкт у Python (у дужках прописано шлях до обраного файлу):

```
face_cascade_db =
cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_frontalface_  efault.xml")
```

Далі необхідно завантажити до проєкту будь-яке фото з обличчям у поточну директорію проєкту.

Після чого необхідно створити об’єкт для цього зображення:

```
img = cv2.imread("img.jpg")
```

Для того щоб наше зображення правильно розпізнало існуючим класифікатором, необхідно його перетворити у відтінки сірого:

```
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY).
```

Виведемо це зображення на екран. Для цього введемо команду « imshow» (назва вікна, назва зображення) та затримку cv2.waitKey():

```
cv2.imshow('rez', img_gray)
cv2.waitKey()
```

Та запускаємо проєкт на виконання (ПКМ – Run). У результаті ми побачимо зображення у сірих відтінках (рис. 14.1).

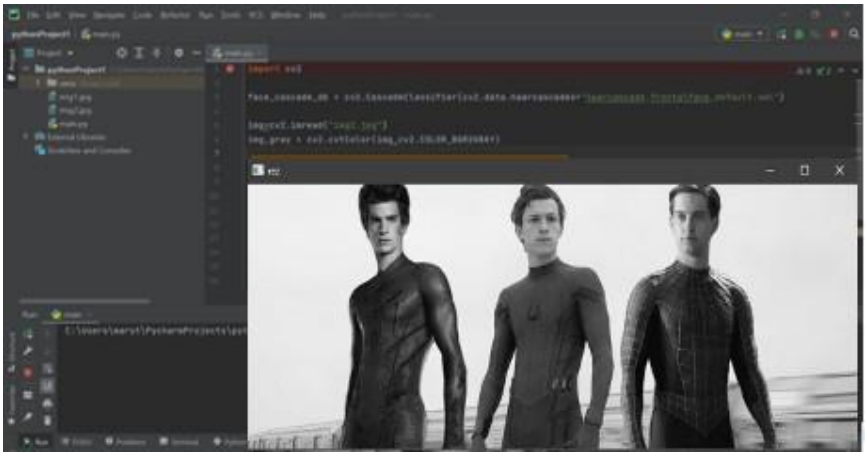


Рисунок 14.1 – Виведення зображення у сірих відтінках [6]

Далі створимо об’єкт faces, у який і запишемо розпізнані обличчя за допомогою класифікатора:

```
faces = face_cascade_db.detectMultiScale(img_gray, 1.1, 19)
```

За допомогою об’єкта face ми отримуємо перелік знайдених координат на цьому зображенні.

Далі виведемо квадрат на знайдених обличчях та не чорно-біле, а кольорове, зображення. Повний код програми наведено у лістингу 14.1.

Лістинг 14.1 – Код програми для розпізнавання обличь на фотографії

```
import cv2 face_cascade_db =
cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_fronta
lface_default.xml")

img = cv2.imread("family.jpg")
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

faces = face_cascade_db.detectMultiScale(img_gray, 1.1, 19) for (x, y, w, h) in
faces:
cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

cv2.imshow('rez', img) cv2.waitKey()
```

Після чого запустимо проєкт та у результаті отримаємо програму, яка розпізнає обличчя на фотографії (рис. 14.2).

Рисунок 14.2 – Результат розпізнавання обличчя на фотографії [6]

Розпізнавання обличчя з відеопотоку. Необхідно написати програму, яка здатна розпізнавати обличчя з потоку вебкамери.

Для цього створимо новий файл у проекті. Як джерело інформації введемо поток з вебкамери:

```
cap = cv2.VideoCapture(0)
```

Далі у циклі отримаємо зображення з вебкамери:

```
while True:
success, img = cap.read()
img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

```
faces = face_cascade_db.detectMultiScale(img_gray, 1.1, 19) for
(x, y, w, h) in faces:
      cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)
```

Функція для виходу з циклу після натискання кнопки «q»:

```
  cv2.imshow('rez', img)
if cv2.waitKey() & 0xff == ord('q'):
break
```

Повний код програми наведено у лістингу 4.2.

Лістинг 14.2 – Код програми для розпізнавання облич з потоку вебкамери

```
  import cv2
  face_cascade_db =
  cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_fronta
  lface_default.xml")

  cap = cv2.VideoCapture(0)
  while True:
      success, img = cap.read()
      img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

      faces = face_cascade_db.detectMultiScale(img_gray, 1.1, 19) for
      (x, y, w, h) in faces:
          cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

      cv2.imshow('rez', img)
      if cv2.waitKey() & 0xff == ord('q'):
          break

  cap.release()
  cv2.destroyAllWindows
  ()
```

У результаті, почергово будуть відображатись кадри з вебкамери з обведеним обличчям на ньому у разі успіху (рис. 14.3).

Контрольні запитання

1. Опишіть програмні продукти, які існують для розпізнавання облич.
2. Опишіть бібліотеки Python, які призначені для розпізнавання.
3. Назвіть бібліотеки Python для роботи з нейронними мережами.
4. Які особливості мають згорткові нейронні мережі?

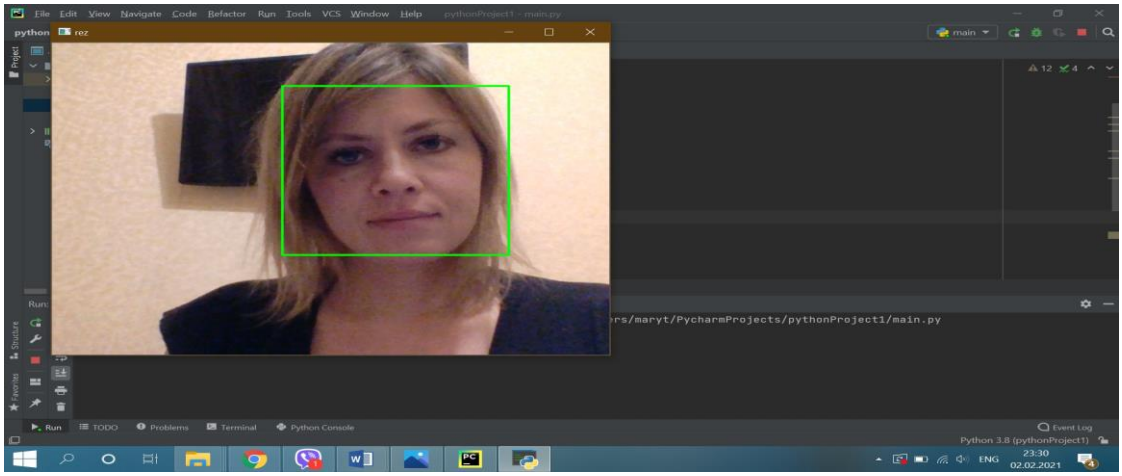


Рисунок 14.3 – Результат розпізнавання обличчя з потоку вебкамери [6]

Як бачимо з результатів, програмою зчитано інформацію з потоку веб-камери, розпізнано на ній обличчя та обведено його зеленою рамкою.

ЗАВДАННЯ ДЛЯ ВИКОНАННЯ САМОСТІЙНИХ РОБІТ

САМОСТІЙНА РОБОТА № 1
АНАЛІЗ ТЕКСТІВ, ЗГЕНЕРОВАНИХ ШТУЧНИМ ІНТЕЛЕКТОМ. ЗАБЕЗПЕЧЕННЯ ПРИНЦИПІВ АКАДЕМІЧНОЇ ДОБРОЧЕСНОСТІ

Мета – отримати практичні навички аналізу текстів на предмет генерації їх штучним інтелектом.

Завдання:

1. Ознайомитися з методами, які дозволять визначити текст, написаний штучним інтелектом чи людиною.
2. За допомогою ШІ (наприклад, ChatGPT) згенерувати текст до 1 500 слів англійською та українською мовою.
3. Скористатися для автоматичного аналізу тексту безкоштовними онлайн-сервісами. Результати аналізу англомовного та україномовного тексту відобразити окремо.
4. Відобразити в текстовому файлі результати аналізу.

САМОСТІЙНА РОБОТА № 2
БАГАТОШАРОВА НЕЙРОННА МЕРЕЖА ПРЯМОГО ПОШИРЕННЯ

Мета − набути практичних навичок використання системи комп'ютерної математики Matlab для розв'язання задач прогнозування за допомогою багатошарових штучних нейронних мереж на базі алгоритму зворотного поширення похибки.

Завдання

1. У робочій області GUI-інтерфейсу NNTool СКМ Matlab створити двошарову нейронну мережу прямого поширення з сигмоїдальною активаційною функцією 'logsig' з двома входами, де n_1 – кількість нейронів першого шару нейронів, а n_2 – кількість нейронів другого шару. Навчити штучну нейронну мережу обчислювати значення функції на основі методу зворотного поширення помилки.

2. Обчислити 10 значень функції по заданих значеннях аргументів та їхніх варіаціях з кроком ±1. За допомогою функції «*newff*» створити двошарову нейронну мережу прямого поширення для мережі з трьома входами і трьома виходами. Перший шар має m_1 нейронів, а другий – m_2 нейронів. Для інших параметрів встановити такі значення:

– параметр net.trainParam.goal = 1e-5;

– параметр epochs підібрати так, щоб процес завершувався за умовою MSE<Goal;

– параметр BTF приймає значення: 'traingd', 'traingdm', 'traingda';

– параметр Tfi приймає значення: tansig, logsig, purelin;

– інші параметри за замовчуванням.

Навчити нейронну мережу обчислювати значення функції на базі алгоритму зворотного поширення помилки.

САМОСТІЙНА РОБОТА № 3
НЕЧІТКІ ВІДНОШЕННЯ ПЕРЕВАГИ

Мета роботи – вивчити властивості нечітких відношень переваги та методів ухвалення рішень за нечіткими відношеннями переваги.

Завдання для підготовки до виконання роботи

1. Визначте раціональний вибір альтернативи з певної множини $X = \{ x_1 , x_2 , x_3 , x_4 , x_5\}$ на основі поданих відношень переваги R_1 та R_2, значущість яких, на думку ОПР, дорівнює відповідно λ_1 і λ_2.

2. Розв'яжіть задачу вибору.

3. Проведіть аналіз отриманих результатів.

САМОСТІЙНА РОБОТА № 4
АНАЛІЗ ІСНУЮЧИХ ПІДХОДІВ ДО СТВОРЕННЯ ІНТЕЛЕКТУАЛЬНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

Мета роботи:

− ознайомитись існуючими методами та алгоритмами інтелектуальних систем;

– визначити коло задач, для вирішення яких доцільним є використання новітніх технологій;

– визначити сферу використання та налаштування програмної реалізації інтелектуальних систем для аналізу обраної економічної системи;

– набути навичок зі збору та попередньої обробки даних;

– навчитися інтерпретувати отримані результати та застосовувати засоби для їхнього зручного подання;

– навчитися проєктувати нові моделі обираючи доцільні налаштування.

Завдання – зробити аналіз питань, які наведені у переліку з мети роботи, підібрати відповідні джерела інформації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Вдовиченко І. Н. Інтелектуальні системи : навч. посіб. / І. Н. Вдовиченко, В. Б. Хоцкіна. – Кривий Ріг : Державний університет, економіки і технологій, 2023. – 187 с. – Існує електрон. версія. (Режим доступу: https://dspace.duet.edu.ua/items/2ef5e726-4706-47c2-8499-d7dfeafa5a43, вільний).

2. Сізова Н. Д. Інтелектуальні управляючі системи і технології [Електрон. ресурс] : конспект лекцій для здобувачів другого (магістерського) рівня вищої освіти денної, заочної і дистанційної форм навчання зі спеціальності 122 – Комп'ютерні науки / Н. Д. Сізова ; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Електрон. текст. дані. – Харків : ХНУМГ ім. О. М. Бекетова, 2024. – 65 с. – Режим доступу: https://eprints.kname.edu.ua/65611/1/2024%20112%D0%9B%20%D0%A1%D0%86%D0%97%D0%9E%D0%92%D0%90%20%28%D0%9A%D0%9E%D0%9D%D0%A1%D0%9F%D0%95%D0%9A%D0%A2%20%D0%86%D0%A3%D0%A1%D0%A2%29_3%20%D1%82%D1%80%D0%B0%D0%B2%D0%BD%D1%8F.pdf, вільний (дата звернення: 04.04.2025). – Назва з екрана.

3. Принципи проектування відкритих розподілених систем : Лабораторний практикум / Уклад. І. Е. Райчев. – Київ : НАУ, 2007. – 80 с. – Існує електрон. версія. (Режим доступу: https://er.nau.edu.ua/items/5d60ce66-6e65-4601-b24b-100409834208, вільний).

4. Методичні вказівки до виконання лабораторних робіт з навчальної дисципліни «Інтелектуальні системи управління та пристрої» (Частина 1. Подання знань в інтелектуальних системах управління) для здобувачів вищої освіти другого (магістерського) рівня / С. Є. Стець. – Рівне : НУВГП, 2024. – 30 с. – Існує електрон. версія. (Режим доступу: https://ep3.nuwm.edu.ua/15668/, вільний).

5. Методичні вказівки до лабораторних робіт з дисципліни «Основи теорії інтелектуальних систем» для студентів спеціальності 123 «Комп'ютерна інженерія» за освітньою програмою «Комп'ютерні системи та мережі» денної форми навчання / уклад. М. Ю. Тягунова. – Запоріжжя : НУ «Запорізька політехніка», 2022. – 58 с. – Існує електрон. версія. (Режим доступу: https://eir.zp.edu.ua/items/4a414c4d-8885-4f34-94b1-1c99ff7fceca, вільний).

6. Кононюк А. Ю. Нейроні мережі і генетичні алгоритми / А. Ю. Кононюк. – Київ : Корнійчук, 2008. – 446 с. – Існує електрон. версія. (Режим доступу: https://ep3.nuwm.edu.ua/2252/, вільний).

7. Кондратенко Ю. П. Нечіткі множини та нечітка логіка. Методичні рекомендації та вказівки для виконання лабораторних робіт студентами спеціальності 122 «Комп'ютерні науки» / Ю. П. Кондратенко, Г. В. Кондратенко, Є. В. Сіденко ; під ред. д-р техн. наук, професор Ю. П. Кондратенка. – Миколаїв : ЧНУ ім. Петра Могили, 2019. – 36 с. (Методична серія ; вип. 267). – Існує електрон. версія. (Режим доступу: https://dspace.chmnu.edu.ua/jspui/bitstream/123456789/308/1/%D0%9A%D0%BE%D0%BD%D0%B4%D1%80%D0%B0%D1%82%D0%B5%D0%BD%D0%BA%D0%BE%20%D0%AE.%20%D0%9F.%20%D0%9D%D0%B5%D1%87%D1%96%D1%82%D0%BA%D1%96%20%D0%BC%D0%BD%D0%BE%D0%B6%D0%B8%D0%BD%D0%B8.%20%D0%92%D0%B8%D0%BF.%20267.pdf, вільний).

Електронне навчальне видання

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ

до виконання практичних робіт
із навчальної дисципліни

«ІНТЕЛЕКТУАЛЬНІ УПРАВЛЯЮЧІ СИСТЕМИ І ТЕХНОЛОГІЇ»

(для здобувачів другого (магістерського) рівня вищої освіти денної, заочної і дистанційної форм навчання зі спеціальності 122 – Комп'ютерні науки)

Укладач **СІЗОВА** Наталія Дмитріївна

Відповідальний за випуск *М. В. Новожилова*
Редактор *О. В. Михаленко*
Комп'ютерне верстання *Н. Д. Сізова*

План 2025, поз. 308М

Підп. до друку 08.07.2025. Формат 60 × 84/16.

Ум. друк. арк. 4,1.

Видавець і виготовлювач:
Харківський національний університет
міського господарства імені О. М. Бекетова,
вул. Чорноглазівська (Маршала Бажанова), 17, Харків, 61002.
Електронна адреса: office@kname.edu.ua
Свідоцтво суб'єкта видавничої справи:
ДК № 5328 від 11.04.2017.