

Міністерство освіти і науки України

Чернівецький національний університет імені Юрія Федьковича

Навчально-науковий інститут
фізико-технічних і комп'ютерних наук

КОНСПЕКТ ЛЕКЦІЙ З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ
«КОМП'ЮТЕРНІ СИСТЕМИ ШТУЧНОГО ІНТЕЛЕКТУ»
(електронне видання)

Чернівці
Чернівецький національний університет
2022

*Рекомендовано Вченою радою
навчально-наукового інституту
фізико-технічних та комп'ютерних наук
Чернівецького національного університету імені Юрія Федьковича
(протокол № 8 від 22.09.2022)*

Конспект лекцій з навчальної дисципліни «Комп'ютерні системи штучного інтелекту» / уклад.: С.В. Баловсяк, Х.С. Одайська. Чернівці: ЧНУ, 2022. 175 с. (електронне видання)

У конспекті лекцій викладено систематизовані знання про принципи побудови та функціонування комп'ютерних систем штучного інтелекту, методи представлення знань, системи нечіткої логіки, експертні системи, штучні нейронні мережі, генетичні алгоритми, системи розпізнавання образів.

Для студентів другого (магістерського) рівня вищої освіти спеціальності 123 «Комп'ютерна інженерія» освітньої-професійної програми «Комп'ютерна інженерія» та освітньо-наукової програми «Комп'ютерна інженерія технологій інтернету речей та кіберфізичних систем»; студентів першого (бакалаврського) рівня вищої освіти спеціальності 126 «Інформаційні системи та технології» освітньо-професійної програми «Інформаційні системи та технології».

ЗМІСТ

1. ОСНОВИ ШТУЧНОГО ІНТЕЛЕКТУ	7
1.1. Визначення поняття „штучний інтелект”	7
1.2. Передісторія штучного інтелекту	11
1.3. Історія штучного інтелекту	18
1.4. Основні напрями розвитку штучного інтелекту	28
1.5. Приклади інтелектуальних задач. Тест Тьюринга і фатичний діалог	31
Контрольні запитання до розділу 1	36
2. МЕТОДИ РОЗПІЗНАВАННЯ ОБРАЗІВ	37
2.1. Біологічні аналоги систем розпізнавання образів	37
2.2. Задача розпізнавання образів	45
2.3. Попередня обробка об’єктів розпізнавання	48
2.4. Шаблонні методи розпізнавання образів	51
2.4.1. Кореляційний метод розпізнавання	51
2.4.2. Метод допустимих перетворень	52
2.5. Структурні методи розпізнавання	54
2.6. Розпізнавання у просторі ознак	56
2.6.1. Кластерний аналіз. Гіпотеза компактності	57
2.6.2. Метод потенціалів	59
2.6.3. Байєсівський метод розпізнавання	60
2.6.4. Метод опорного словника	61
2.6.5. Метод зондів	62
Контрольні запитання до розділу 2	64
3. МОДЕЛІ І МЕТОДИ ПРИЙНЯТТЯ РІШЕНЬ	65
3.1. Планування цілеспрямованих дій і прийняття рішень	65
3.2. Складність вирішення оптимізаційних задач	66
3.2.1. Пошук у глибину і у ширину	67
3.2.2. Автоматний спосіб задання алгоритму	68
3.3. Складність алгоритмів	71
3.4. Планування в просторі станів	72
3.4.1. Основні поняття теорії графів	72
3.4.2. Способи задання графів	73
3.4.3. Деревя	74
3.5. Загальна схема алгоритму Дейкстри	75
3.6. Планування в просторі задач	76

3.7. Жадібні алгоритми	78
3.7.1. Градієнтний метод	78
3.7.2. Алгоритм Пріма	78
3.7.3. Алгоритм Крускала	80
3.7.4. Динамічне програмування	81
3.8. Вирішувані інтелектуальних задач	82
3.8.1. Загальний вирішував задач	82
3.8.2. Ігрові задачі як задачі прийняття рішень	82
3.8.3. Позиційні ігри з оптимальними стратегіями	83
3.8.4. Альфа – бета відтинання	85
3.9. Евристичні алгоритми	86
Контрольні запитання до розділу 3	87
4. ШТУЧНІ НЕЙРОННІ МЕРЕЖІ	88
4.1. Біологічні нейронні мережі	88
4.2. Штучний нейрон	89
4.3. Активаційні функції нейронів	90
4.4. Одношарові нейронні мережі	92
4.5. Персептрони і зародження штучних нейромереж	92
4.6. Багатошарові нейронні мережі	93
4.7. Навчання нейронних мереж	94
4.8. Алгоритм зворотного розповсюдження помилки	95
4.9. Мережі зустрічного розповсюдження. Шари Кохонена і Гросберга	97
4.10. Стохастичні методи навчання нейронних мереж	99
4.11. Мережі зі зворотними зв'язками. Мережі Хопфілда	101
4.12. Адаптивна резонансна теорія	103
4.13. Згорткові нейронні мережі	104
4.14. Багатомасштабний аналіз сигналів за допомогою штучних нейронних мереж.....	111
Контрольні запитання до розділу 4	118
5. ГЕНЕТИЧНІ АЛГОРИТМИ	119
5.1. Основи генетичних алгоритмів	119
5.2. Класичний генетичний алгоритм	122
5.3. Програмна реалізація генетичних алгоритмів	124
5.4. Застосування генетичних алгоритмів	131
Контрольні запитання до розділу 5	133
6. КІБЕРНЕТИЧНІ СИСТЕМИ	134
6.1. Поняття кібернетичної системи	134

6.2. Керування кібернетичними системами	134
6.3. Основні напрями досліджень кібернетичних систем	137
7. ЗНАННЯ ЯК ІНФОРМАЦІЙНА ОСНОВА ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ	138
7.1. Методи подання знань	138
7.2. Семантичні мережі	141
7.2.1. Визначення та класифікація семантичних мереж..	141
7.2.2. Застосування семантичних мереж.....	143
7.3. Фрейми	144
7.3.1. Базові поняття фреймів	144
7.3.2. Поповнення первинних описів на основі фреймів.	145
7.3.3. Фрейми та об'єктно-орієнтоване програмування .	147
7.4. Логічні моделі та метод резолюцій	148
7.4.1. Логічні побудови та логічні моделі	148
7.4.2. Вступ до числення предикатів	152
7.4.3. Логічне програмування	152
7.5. Продукційні моделі	153
7.5.1. Характеристика продукційних моделей	154
7.5.2. Продукції та мережі виведення	154
7.5.3. Схема роботи експертної системи на базі продукцій	154
8. НЕЧІТКА ЛОГІКА	158
8.1. Поняття нечіткості та нечіткі функції належності.....	158
8.2. Основні операції над нечіткими множинами	160
8.3. Нечітке логічне виведення	161
8.4. Логічне виведення для недостовірних знань	164
8.5. Нечітке логічне виведення за алгоритмом Мамдані	165
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	173

СПИСОК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

AI – Artificial Intelligence

АПТ – Адаптивна резонансна теорія

CNN – Convolutional neural network

GA – Genetic Algorithms

GPS – General Problem Solver (загальний вирішувач задач)

HMM – Hidden Markov Model (приховані марковські моделі)

HPP – Heuristic Programming Project (проект евристичного програмування)

ГА – Генетичні алгоритми

ЗНМ – Згорткові штучні нейронні мережі

РО – Розпізнавання образів

СМ – Семантична мережа

ШІ – Штучний інтелект

ШНМ – Штучні нейронні мережі

1. ОСНОВИ ШТУЧНОГО ІНТЕЛЕКТУ

Людина – міра всіх речей

Протагор

Я мислю, отже, я існую

Рене Декарт

1.1. Визначення поняття „штучний інтелект”

Здатність мислити є головною властивістю людини – самого складного об’єкта, доступного для нашого сприйняття. Але яка природа мислення? Які процеси відбуваються в нашому організмі, коли ми думаємо, відчуваємо, бачимо, розуміємо? Чи можливо взагалі зрозуміти, як працює наш мозок, і заставити мислити неживу природу? На протязі тисячоліть людина задумувалася над цими питаннями, але до цього часу на них немає остаточної відповіді [1-7].

Термін „інтелект” (*intelligence*) походить від латинського слова „*intellectus*”, яке означає розум, здатність людини до мислення. Відповідно поняття „штучний інтелект” (*Artificial Intelligence, AI*) звичайно пояснюється як властивість автоматичних систем виконувати окремі функції інтелекту людини, наприклад, приймати оптимальні рішення на основі раніше отриманого досвіду і раціонального аналізу зовнішніх дій. У наш час *штучний інтелект* (ШІ) – одна із найновіших областей комп’ютерної науки, яка займається автоматизацією розумної поведінки і створенням апаратно-програмних технічних систем, що в певній мірі відтворюють людський інтелект [1-2]. Комп’ютерні системи ШІ – апаратно-програмна реалізація систем ШІ.

Виникнення штучного інтелекту було закономірним явищем, оскільки засоби ШІ можуть так само розширити можливості природного інтелекту, як механізми і машини – фізичні можливості людини. Виняткова важливість інтелектуальних здібностей для людини проявилася навіть у науковій назві її як виду: „*Homo sapiens*” (*людина розумна*). У даний час тематика ШІ охоплює величезний перелік наукових напрямів, починаючи з таких задач загального характеру, як навчання і сприйняття, і закінчуючи спеціальними задачами, наприклад, гра в шахи, доведення математичних теорем, медична і технічна діагностика, розпізнавання зображень та ін. У ШІ систематизуються і автоматизуються задачі будь-якої сфери інтелектуальної діяльності людини.

Інтелектуальними будемо називати задачі, пов'язані з створенням (або модифікацією) *алгоритму* вирішення класу задач певного типу. Прикладом таких задач є розпізнавання образів, гра в шахи, доведення теорем та ін. Для інтелектуальних задач формальне розбиття процесу пошуку рішення на окремі елементарні кроки часто є дуже складним. Таким чином, можна перефразувати визначення інтелекту як універсальний надалгоритм, який здатний створювати алгоритми вирішення конкретних задач. Виходячи з цього визначення, професія програміста є однією з найбільш інтелектуальних, оскільки продуктом діяльності програміста є алгоритми, реалізовані у вигляді програм.

Оскільки ІІІ знаходиться в стані динамічного розвитку, тому дати формальне визначення понять „інтелект” та „штучний інтелект” є складним завданням [1-7]. Наприклад, визначення „*Інтелект є здатність правильно, в залежності від стану інформаційних входів, реагувати на нову ситуацію*” є неоднозначним, оскільки не визначено поняття „нова ситуація”. Якщо калькулятор ніколи не обчислював суму двох одиниць, то завдання „1+1” є для нього новим. Безумовно, калькулятор виконає правильно операцію додавання (вірно відреагує на нову ситуацію), але чи можна на цій підставі вважати його інтелектуальною системою?

Проаналізувавши визначення ІІІ, можна їх поділити на дві *основні категорії* (табл. 1.1) [1]: засновані на відтворенні *здібностей людини* (ліва половина таблиця) або на *раціональності поведінки* (права половина). Система є *раціональною*, якщо вона „виконує всі дії правильно”, при умові, що система має знання про те, що є правильним. Визначення ІІІ також класифікують за двома *допоміжними категоріями*: заснованими на *процесах мислення* (№ 1-4) та на *поведінці* (№ 5-8). На основі вищевказаних категорій виділяють *чотири напрямки ІІІ* (N 1-4), за кожним з яких проводяться інтенсивні дослідження.

Орієнтуючись на *здібності людини* або на *раціональність поведінки*, виділяють два великі *напрями ІІІ*:

1. *Нейрокібернетика*, яка заснована на апаратно-програмному моделюванню роботи людського мозку; об'єктом досліджень є структура і механізми роботи мозку з використанням психофізіологічних даних; цей напрям також називають *низькорівневим* або *висхідним*.

2. Кібернетика „чорного ящика”, в якій головне – раціональна поведінка системи, а не її будова; основна увага приділяється адекватному моделюванню функцій системи; об’єктом досліджень є системи ШІ; даний напрям ще називають *високорівневим* або *нисхідним*.

Вищевказані напрями ШІ об’єднуються при створенні змішаних *людино-машинних (інтерактивних)* інтелектуальних систем.

Таблиця 1.1

Визначення №1-8 систем штучного інтелекту [1]

№ 1. Думають подібно до людей	№ 2. Думають раціонально
№ 1. „Новий захоплюючий напрям робіт по створенню комп’ютерів, здатних думати, ... розумних машин”	№ 3. „Вивчення розумових здібностей за допомогою обчислювальних моделей”
№ 2. „Автоматизація дій, які ми асоціюємо з людським мисленням, тобто таких дій, як прийняття рішень, навчання...”	№ 4. „Вивчення таких обчислень, які дозволяють відчувати, розмірковувати і діяти”
№ 3. Діють подібно до людей	№ 4. Діють раціонально
№ 5. „Мистецтво створення машин, які виконують функції, що вимагають інтелектуальності при їх виконанні людьми”	№ 7. „Обчислювальний інтелект – це наука про проектування інтелектуальних агентів”
№ 6. „Наука про те, як навчити комп’ютери робити те, в чому люди у даний час їх перевіряють”	№ 8. „Наука, присвячена вивченню інтелектуальної поведінки артефактів (штучних об’єктів)”

Поряд з переліченими двома напрямками моделювання мислення і створення ШІ існує третій, який називається еволюційним програмуванням (моделюванням).

Нейрокібернетичний напрям також називають *імітаційним* або *біонічним*, а напрям кібернетики „чорного ящика” – *прагматичним*. Вибір конкретного напрямку ШІ визначається особливостями задачі. Подібна проблема: наслідувати природу (*біоніка*) чи відтворити тільки функції об’єкта, вирішувалася по-різному у різних областях науки і техніки. Вдалим прикладом застосування *біоніки* є створення

штучних нейронних мереж як спрощеної моделі природних. А в авіації проблему польоту тіл, важчих за повітря, вдалося вирішити тільки за допомогою *прагматичного* підходу, коли брати Райт та інші дослідники перестали імітувати політ птахів і використали нерухоме крило. Тому для вирішення реальних задач варто використовувати як біонічний, так і прагматичний підхід.

Для всіх напрямів ШІ виділяють два основні завдання:

1. Розкриття таємниць людського мислення.
2. Створення технічних систем, які дозволяють замінити людину при вирішенні різноманітних науково-практичних завдань.

Відсутність однозначного визначення поняття „інтелект” ускладнює оцінку інтелектуальності системи, але на інтуїтивному рівні інтелектуальність в основному оцінюється двома методами: *експертних оцінок* і *тестування* [3, С. 8-9]. Ці методи можуть застосовуватися для визначення інтелектуальності як живих істот, так і систем штучного інтелекту.

В методі *експертних оцінок* рішення про ступінь інтелектуальності приймає досить велика група експертів (незалежно або у взаємодії).

У разі використання методу *тестування* пропонується розв’язати набір тестових завдань. Існує значна кількість інтелектуальних тестів для оцінки розумових здібностей людини, що знайшли застосування, наприклад, в психології:

№ 1. *Вставне пропущене число: 36, 30, 24, 18, 6.*

№ 2. *Викресліть зайве слово: соловей, перепілка, стілець, шпак.*

Процес тестування значно ускладнює той факт, що задача може бути вирішена *логічно вірно*, але *неправильно*. Наприклад, у вищенаведеному прикладі № 2 більшість людей вибере відповідь „*стілець*”, оскільки інші слова – це назви птахів. Проте хтось може викреслити слово „*шпак*”, мотивуючи це тим, що це єдине слово, в якому відсутня літера „л”. У таких неоднозначних випадках правильною вважається відповідь, яка є важливішою для практичної діяльності людини.

Для розвитку систем ШІ потрібне краще розуміння природного інтелекту, наприклад за допомогою *когнітології* – міждисциплінарної області ШІ і психології, яка досліджує процеси пізнання. У цій області розумові процеси людини досліджуються двома методами: *інтроспекції* (спроба прослідкувати за ходом власних думок) і *психологічними експериментами*. Основним недоліком інтроспекції є суб’єктивізм, а психологічних експериментів – обмеженість засобів.

1.2. Передісторія штучного інтелекту

Штучний інтелект має багату *передісторію*, вивчення якої дозволяє глибше зрозуміти зміст наукових моделей. Розглянемо основні наукові дисципліни, які внесли суттєвий вклад у формування ШІ (рис. 1.1) [1].



Рис. 1.1. Наукові дисципліни, які мали значний вплив на формування ШІ

Таким чином, значний вплив на формування і розвиток ШІ мали наступні наукові дисципліни:

1. **Філософія** (період з 428 року до н.е. по даний час) досліджує такі питання, важливі для ШІ:

- Чи можуть використовуватися формальні правила для отримання правильних висновків?
- Як такий ідеальний об'єкт, як думка, народжується в такому фізичному об'єкті, як мозок?
- Яке походження знань?
- Яким чином знання ведуть до дій?

Точний перелік законів, які керують *раціональною* (формальною) частиною мислення, вперше сформульований Аристотелем (384-322 роки до н.е.). Він розробив систему *силогізмів*, призначену для проведення правильних міркувань, яка дозволяла виконувати логічні висновки механічно. Класичний приклад силогізму:

Перше твердження: Усі люди смертні.

Друге твердження: Сократ – людина.

Висновок: Сократ – смертний.

Якщо перше і друге твердження в силогізмі істинне, то й висновок буде істинним.

Пізніше Раймунд Луллій (лат. Raymundus Lullius, 1235-1315 рік) висунув ідею, що корисні міркування можна виконувати за допомогою механізмів, наприклад за допомогою запропонованих ним „концептуальних коліс” (рис. 1.2). Запропонована ним система кругів могла повертатися. Кожний круг був поділений на сектори, розфарбовані у різні кольори в позначені латинськими буквами. Повертаючи взаємозв’язані круги можна було отримати різні комбінації символів і кольорів – так звану формулу істини.

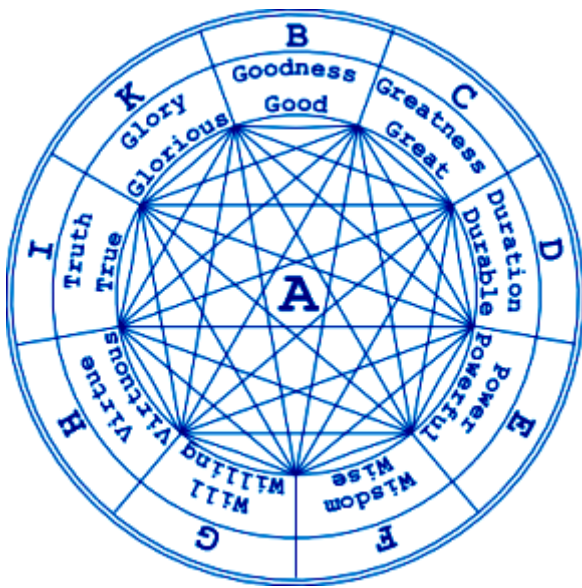


Рис. 1.2. Один з кругів машини Луллія. В пізньому варіанті машина складалася з 14 кругів, символи і кольори яких могли утворювати $18 \cdot 10^{15}$ різних комбінацій. Запити в машину вводилися за допомогою внутрішнього колеса, на якому було написано дев’ять варіантів запитань: Що? Чому? З чого? Скільки? Яким чином? Де? Коли? Яке? Яке з двох?

Машини Луллія могли працювати в різних предметних областях, наприклад складати гороскопи, ставити діагнози хвороб та ін. По суті машина Луллія була першою механічною експертною системою, яка містила базу знань та пристрої вводу/ виводу [1, 2, 4].

Томас Гоббс (1588-1679) вважав, що міркування аналогічні числовим розрахункам. Рене Декарт (1596-1650) розглядав розум як фізичну систему, яка керується фізичними законами. Спосіб отримання знань через відчуття досліджував Френсіс Бекон (1561-1626), який став родоначальником такого наукового напрямку, як *емпіризм*. Девід Юм (1711-1776) запропонував принцип *індукції*, який полягає в отриманні загальних правил на основі дослідження часткових випадків. В роботах Людвіга Вітгенштейна (1889-1951), Бертрана Рассела (1872-1970), Рудольфа Карнапа (1891-1970) розроблено доктрину *логічного позитивізму*. Згідно цій доктрині всі знання можуть бути охарактеризовані за допомогою логічних теорій, пов’язаними з вхідними сенсорними даними.

2. **Математика** (період приблизно з 800 року по даний час) досліджує наступні проблеми, важливі для ІІІ:

- *Які формальні правила формування правильних висновків?*
- *Як виконувати міркування з використанням недостовірної інформації?*

Для перетворення ІІІ у формальну науку був потрібен певний рівень формалізації у трьох фундаментальних областях: *логіка, обчислення і ймовірність*.

Джерела ідей *формальної логіки* були закладені в роботах вчених древньої Греції, але її становлення як математичної дисципліни фактично почалося з праць Джорджа Буля (1815-1864), який детально розробив логіку висловлювань або булеву логіку. Свою мету Д. Буль описував так: „*Дослідити фундаментальні закони таких операцій розуму, якими виконуються міркування; виразити їх на символічній мові і на цій основі створити науку логіки...*” [2].

В 1879 році Готтлоб Фреге (1848-1925) розширив булеву логіку для включення в неї об'єктів і відношень, створивши *логіку першого порядку*, яка в даний час використовується як найбільш фундаментальна система представлення знань. Альфред Тарський (1902-1983) розробив *теорію посилай*, яка показує, як зв'язати логічні об'єкти з об'єктами реального світу.

Дослідження *алгоритмів* як самостійних об'єктів було почато аль-Хорезмі, середньоазіатським математиком IX століття.

Алан Тьюринг (1912-1954) спробував точно охарактеризувати, які функції можуть бути обчислені, розробивши для цього так звану *машину Тьюринга*.

Крім логіки і теорії обчислень, третій за величиною вклад математиків в ІІІ полягав у розробці *теорії ймовірностей*. Значний вклад у цьому напрямку вніс Томас Байєс (1702-1761), який запропонував правило оновлення ймовірностей з врахуванням нових фактів.

3. **Економіка** (період з 1776 року по даний час) знаходить відповіді на такі запитання, пов'язані зі ІІІ:

- *Як потрібно організувати прийняття рішень для максимізації винагороди?*
- *Як діяти в таких умовах, коли інші можуть перешкоджати здійсненню намічених дій?*

Економіка як наука виникла в 1776 році на основі праць шотландського філософа Адама Сміта (1723-1790). Для опису економічних утворень фон Нейманом і Моргенштерном розроблена *теорія ігор*.

4. **Неврологія** (період з 1861 року по даний час) вивчає наступне фундаментальне питання ІІІ: *як виконується обробка інформації у мозку?*

Неврологія – це наука, присвячена вивченню нервової системи, зокрема мозку. Нейрологія розвивається швидкими темпами, але одною з найбільших загадок, яка ще не піддається науковому опису, залишається пояснення того, як саме мозок забезпечує мислення.

Дослідження *афазії* (порушення мови) у пацієнтів з пошкодженням мозку, проведені Полем Брока (1824-1880) в 1861 році, показали існування у мозку локалізованих ділянок, відповідальних за конкретні пізнавальні функції. Наприклад, функції формування мови зосереджені у тій частині лівої півкулі, яка тепер називається зоною Брока. В той час вже було відомо, що мозок складається з нервових клітин або *нейронів*, але тільки в 1873 році Камилло Гольджи (1843-1926) розробив надійний метод, дозволяючий спостерігати окремі нейрони у мозку. Цей метод використав Сантьяго Рамон и Кахал (1852-1934) у своїх піонерських дослідженнях нейронних структур мозку.

Вимірювання активності непошкодженого мозку почалося у 1929 році з винайдення *електроенцефалографа* (ЕЕГ) (рис. 1.3) Гансом Бергером. Розробки в області отримання зображення на основі функціонального магнітного резонансу дозволили неврологам отримувати детальні зображення активності мозку. Це дає можливість проводити виміри характеристик фізіологічних процесів мозку, які пов'язані з пізнавальними процесами. До перспективних методів дослідження мозку відноситься томографія, наприклад магніто-резонансна (Magnetic Resonance Imaging) або ядерна магніто-резонансна томографія (Nuclear Magnetic Resonance Imaging) (рис. 1.4).

Роботи в області неврології дозволяють зробити надзвичайно важливий висновок про те, що сумісна робота нейронів мозку приводить до *мислення, дій і свідомості*, тобто мозок породжує розум. Відкритим залишається питання про те, наскільки повно можна відтворити функції мозку за допомогою комп'ютера.

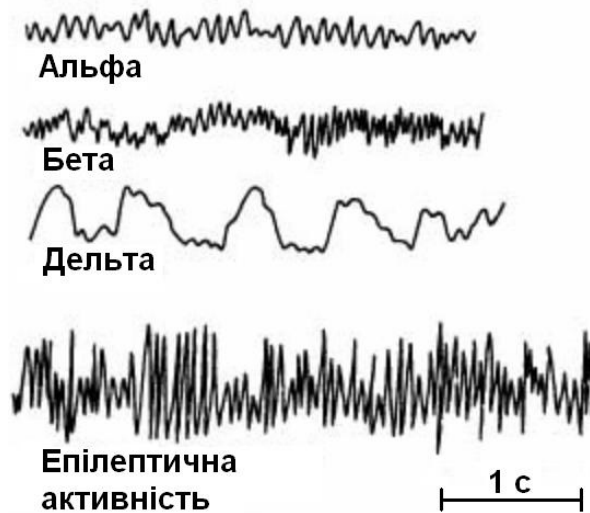


Рис. 1.3. Електроенцефалограми мозку:
альфа – розслаблене неспання (альфа-хвилі),
бета – активне неспання (бета-хвилі),
дельта – сон (дельта-хвилі)



Рис. 1.4. Бічний зріз мозку
при магніто-резонансній
томографії

Мозок і цифровий комп'ютер виконують різні задачі і відрізняються за властивостями. В таблиці 1.2 показано, що у звичайному мозку людини є приблизно в 1000 разів більше нейронів, ніж логічних елементів в процесорі типового персонального комп'ютера. Мікросхеми комп'ютера здатні виконати окрему команду менше ніж за наносекунду, а нейрони працюють в мільйони разів повільніше. Але мозок компенсує цей недолік, оскільки всі його нейрони і синапси діють одночасно, тоді як більшість сучасних комп'ютерів мають один процесор або незначну кількість процесорів.

Таблиця 1.2

Приблизне порівняння фізичних обчислювальних ресурсів
персонального комп'ютера і мозку людини [1]

	Комп'ютер	Мозок
Обчислювальні модулі	Один центральний процесор, 10^8 логічних елементів	10^{11} нейронів
Модулі пам'яті	Оперативна пам'ять на 10^9 бітів Жорсткий диск ємністю 10^{12} бітів	10^{11} нейронів 10^{14} синапсів
Тривалість циклу обробки	10^{-9} секунди	10^{-3} секунди
Пропускна здатність	10^{10} біт/с	10^{14} біт/с

Таким чином, незважаючи на перевагу комп'ютера у швидкості виконання окремих операцій більш ніж у мільйон разів, мозок у порівнянні з ним виконує свої дії приблизно у 100 000 разів швидше. Проте найбільшою перевагою мозку є ефективні, але ще не вивчені принципи обробки інформації, які дозволяють виконувати ще недоступні для комп'ютера операції.

5. Психологія (період з 1879 року по даний час) досліджує таку важливу задачу ІІІ: *як думають і діють люди та тварини?*

Основи наукової психології заклали у своїх працях німецький фізик Герман фон Гельмгольц (1821-1894) і його студент Вільгельм Вундт (1832-1920). Гельмгольц застосував науковий метод для вивчення *зору людини*, чим вніс значний вклад у вивчення фізики і фізіології зору людини.

Прихильники *біхевіористського* руху, очолювані Джоном Уотсоном (1878-1958), відкидали інтроспекцію і наполягали на тому, що можна досліджувати тільки об'єктивні міри сприйняття (або стимули) та відгуки на них.

Згідно поглядам представників *когнітивної психології* мозок розглядається як пристрій обробки інформації. Вважається, що народження *когнітології* як наукового напрямку відбулося на семінарі в Масачусетському технологічному інституті (США) у вересні 1956 року, де свої роботи представили Джордж Міллер, Ноам Хомський, Аллен Ньюелл і Герберт Саймон. У рамках цих робіт було показано, як можна використовувати комп'ютерні моделі для вирішення задач в області психології.

6. Обчислювальна техніка (період з 1940 року по даний час) вирішує питання створення ефективного комп'ютера (апаратної і програмної бази ІІІ).

Механічні обчислювальні пристрої існували задовго до появи електронного комп'ютера. Перша відома *обчислювальна машина* була створена приблизно в 1623 році німецьким вченим Вільгельмом Шиккардом (1592-1635). Більш відомою була обчислювальна машина „Паскаліна”, побудована в 1642 році Блезом Паскалем (1623-1662). Вільгельм Лейбніц (1646-1716) у 1643 році створив механічний пристрій, призначений для виконання операцій над поняттями, а не над числами.

Першим *програмованим пристроєм* був ткацький станок, винайдений у 1805 році Жозефом Жаккардом (1752-1834), в якому використовувалися *перфокарти* для збереження інструкцій, згідно яких ткалися візерунки на тканинах. В 1834 році Чарльз Беббідж (1792-1871) розробив проект механічної „*аналітичної машини*”, яку вважають прообразом сучасних комп'ютерів. Ада Лавлейс, помічниця Ч. Беббіджа і, можливо, перший у світі програміст, описувала аналітичну машину так: „*Можна сказати, що аналітична машина плете алгебраїчні візерунки подібно тому, як станок Жаккарда тче візерунки із квітів і листків...*” [2].

Першим *операційним комп'ютером* був електромеханічний пристрій *Heath Robinson*, створений у 1940 році групою Алана Тьюринга і призначений для розшифрування повідомлень. В 1943 році та ж група розробила ламповий комп'ютер загального призначення *Colossus*. Першим *операційним програмованим комп'ютером* був комп'ютер Z-3, винайдений Конрадом Цузе в Німеччині у 1941 році. Найбільший вплив на розвиток сучасних комп'ютерів мав комп'ютер *ENIAC*, розроблений у 1946 році в Пенсільванському університеті (США) групою, до складу якої входив Джон Мочлі і Джон Екерт.

Під керівництвом С. Лебедева в Інституті електротехніки АН Української РСР у 1951 році створюється комп'ютер „МЭСМ – Малая Электронная Счетная Машина” – не тільки перший комп'ютер в СРСР, але й в континентальній Європі.

Крім найбільш поширених *електронних цифрових* комп'ютерів існують також *аналогові* комп'ютери. В даний час ведуться розробки по створенню *оптичних* комп'ютерів, які використовують замість традиційних електричних сигналів світлові. Інший перспективний напрям передбачає використання молекулярної біології і досліджень дезоксирибонуклеїнової кислоти (ДНК) для створення *біологічного* комп'ютера. Один з найновіших підходів, який здатний привести до принципового підвищення швидкості обчислень, полягає у розробці *квантових* комп'ютерів.

7. Теорія керування і кібернетика (період з 1948 року по даний час) вирішує питання: *яким чином пристрої можуть працювати під своїм власним керуванням?*

Перший *самокерований пристрій* був побудований Ктесибієм із Александрії (приблизно в 250 році до н.е.); це був водяний годинник з регулятором, який підтримував постійний потік води через годинник. До цього вважалося, що тільки живі істоти здатні змінювати свою поведінку у відповідь на зміни в оточуючому середовищі. До інших прикладів саморегульованих систем керування з *оберненим зв'язком* відносяться *регулятор парової машини*, створений Джеймсом Уаттом (1716-1819) і *термостат*, винайдений Корнелісом Дреббелем (1572-1633). Математична теорія стійких систем з оберненим зв'язком була розроблена у ХІХ столітті.

Центральною фігурою у створенні теорії керування як науки був Норберт Вінер (1894-1964). Він вважав, що цілеспрямована поведінка зумовлена дією регуляторного механізму, який намагається мінімізувати „помилку” – різницю між поточним і цільовим станом системи. В популярній книзі Н. Вінера „Cybernetics”, в якій було вперше дано визначення кібернетики як науки, показано також можливість створення машин, які володіють штучним інтелектом.

8. **Лінгвістика** (період з 1957 року по даний час) вивчає питання: *яким чином мова пов'язана з мисленням?*

Видатний внесок у лінгвістику знобив Ноам Хомський, який розробив *теорію мови*, побудовану на *синтаксичних моделях*. Теорія Хомського виявилася достатньо формальною для того, щоб її можна було реалізувати програмно. Лінгвістика і ШІ перетинаються в гібридній області, яка називається *обчислювальною лінгвістикою* або обробкою природної мови. Було виявлено, що проблема розуміння мови виявилася значно складнішою, ніж це здавалося в 1957 році: для розуміння мови потрібно розуміння предмету і контексту мовлення, а не тільки аналіз структури речення.

1.3. Історія штучного інтелекту

Розвиток штучного інтелекту можна умовно розділити на кілька основних етапів [1]:

1. **Поява передумов створення штучного інтелекту** (період з 1943 року по 1955 рік). Класична робота цього періоду і перша наукова робота, що відноситься до ШІ, виконана Уорреном Мак-Каллоком і Уолтером Піттсом. На основі відомих даних про роботу нейронів головного мозку вони створили модель *штучного нейрона*,

модель *штучної нейронної мережі* (ШНМ), а також стверджували, що подібні нейронні мережі здатні навчатися. Дональд Хебб запропонував просте правило модифікації ваг між нейронами при навчанні ШНМ. У 1951 році в Прістонському університеті Марвін Мінський і Дін Едмондс створили перший мережевий комп'ютер на основі нейронної мережі. Пізніше М. Мінський довів важливі теореми, які показують обмеження ШНМ. Серед ранніх робіт, що відносяться до штучного інтелекту, можна виділити роботу Алана Тьюринга (1950 рік) [1], в якій він описав відомий *тест Тьюринга*, принцип *машиного навчання* та *генетичні алгоритми*.

2. ***Народження штучного інтелекту*** (1956 рік). Термін „*штучний інтелект*” запропонував Джон Маккарті після проведення влітку 1956 року двохмісячного семінару в Дартмундському коледжі (США). На цьому семінарі зібралася більшість американських вчених, які займалися теорією автоматів, нейронними мережами і дослідженнями інтелекту. В результаті Дартмундського семінару штучний інтелект оформився як окремий науковий напрям.

Значний вклад у становлення штучного інтелекту внесли Аллен Ньюелл і Герберт Саймон, які продемонстрували евристичну програму „*логік-теоретик*” (*Logic Theorist*), здатну доводити математичні теореми. Щоб зрозуміти механізми творчого мислення, автори евристичного підходу провели експеримент. Була відібрана група студентів, не знайомих з математичною логікою. Кожний студент повинен був довести кілька теорем. При цьому студенти повинні були розмірковувати вголос і записувати свої дії. Після обробки записів програмісти знайшли евристики – способи, якими користувалися студенти при доведенні теорем. На основі цих евристик і була створена програма „*логік-теоретик*”.

3. ***Ранній ентузіазм, великі сподівання*** (період з 1952 року по 1969 рік). Для перших років розвитку ШІ характерна велика кількість успіхів, хоча й скромних. Враховуючи примітивність комп'ютерів того часу, великим успіхом було заставити комп'ютер виконувати дії, подібні до розумних.

Після перших успішних розробок Ньюелла і Саймона було створено програму *загального вирішувача задач* (*General Problem Solver – GPS*). На відміну від програми „*Logic Theorist*”, нова програма з самого початку була призначена для моделювання

процедури вирішення задач людиною (для обмеженого класу головоломок).

Результати успішного використання GPS дозволили сформулювати знамениту *гіпотезу фізичної символічної системи*. Ця гіпотеза стверджує, що будь-яка інтелектуальна система (людина або машина), повинна діяти за принципом маніпулювання структурами даних, які складаються із символів. Пізніше дана гіпотеза була піддана критиці.

Герберт Гелернтер створив програму „*Geometry Theorem Prover*” (програму автоматичного доведення геометричних теорем). Починаючи з 1952 року Артур Самюел написав ряд програм для гри в шашки, які грали на рівні добре підготованого любителя. В ході досліджень А. Самюел заперечив твердження, що комп’ютери здатні виконувати тільки те, чому їх вчили: одна з його програм швидко навчилася грати краще, ніж її творець. Ця програма була продемонстрована по телебаченню (США) і справила сильний вплив на глядачів.

Джон Маккарті, працюючи в Массачусетському технологічному інституті (США), протягом 1958 року зробив три значних вклади у розвиток ШІ. По-перше, він дав визначення нової мови високого рівня *Lisp*, яка стала домінуючою мовою програмування для ШІ. По-друге, для ефективної роботи з обмеженими комп’ютерними ресурсами Д. Маккарті разом із співробітниками розробив *режим розділення часу*. По-третє, описав гіпотетичну програму „*Advice Taker*”, яка вважається першою повною системою ШІ; особливістю програми було використання знань про навколишній світ при пошуку рішень задач.

Бурхливо проводилися дослідження в області ШНМ. Методи навчання Хеба були вдосконалені в роботах Берні Відроу, який називав свої мережі *адалінами*. У 1962 році Френк Розенблат розробив *персептрон* – одношарову нейромережу для розпізнавання зображень.

В кінці 50-х років ХХ століття з’явилася модель лабіринтного пошуку. Згідно цьому підходу рішення інтелектуальної задачі виконувалося шляхом перебору величезної кількості варіантів; сам процес рішення уявлявся як рух по лабіринту. Принциповим недоліком лабіринтного пошуку є обмеження методу повного

перебору при збільшенні складності задачі. В даний час модель лабіринтного пошуку має обмежене використання в ігрових комп'ютерних програмах.

Окремо слід виділити роботи вчених, що працювали в інституті кібернетики НАНУ ім. В.М. Глушкова (м. Київ).

Глушков В.М. виділив основні напрямки досліджень з штучного інтелекту і одним із перших сформулював у термінах теорії автоматів основні поняття штучного інтелекту, такі як "адаптація", "самоорганізація", "самовдосконалення", та ввів їх відносну міру. Саме він подав і реалізував ідею розробки нової формальної системи - алгебри алгоритмів, що дало можливість формалізувати практичні задачі розробки комп'ютерних систем. Плідно працював В.М. Глушков над інтелектуальним комп'ютерним інструментарієм для розв'язання задач економіки. Прогнозний граф, математичні моделі систем, що розвиваються, експертні оцінки –це далеко не повний список його ідей щодо цього.

М.М.Амосов висунув інформаційну гіпотезу про програми психічної діяльності людини. Разом зі своїми учнями він зайнявся моделюванням інтелектуальних функцій мережами автоматів, створив транспортний інтелектуальний робот ТАІР, систему "Соціум".

О.Г.Івахненко розпочав роботу над розпізнавальною самонавчальною системою на базі дворядного персептрона "Альфа". Прихильник біонічного підходу та використання самоорганізації у створенні кібернетичних систем, він розробив метод групового врахування аргументів для розв'язання задач технічної кібернетики, визначив термін "евристична самоорганізація".

4. Зіткнення з реальністю (період з 1966 по 1973 рік). З самого початку дослідники ШІ робили сміливі і багатообіцяючі прогнози. Наприклад, Герберт Саймон у 1957 році прогнозував, що через 10 років комп'ютер стане чемпіоном світу з шахів і доведе важливі математичні теореми. Такі передбачення частково збулися, але тільки через 40 років.

Надлишковий оптимізм дослідників ШІ був зумовлений тим, що перші системи штучного інтелекту демонстрували багатообіцяючу продуктивність, хоча й на простих прикладах. Проте у більшості випадків такі ранні системи не могли вирішити складніші завдання.

Наприклад, в 1957 році в стали інтенсивно розвиватися роботи з *машинного перекладу текстів*; при цьому вважалося, що для збереження точного змісту речень достатньо провести прості *синтаксичні перетворення*, засновані на граматиках відповідних мов. Послівний переклад тексту з використанням електронних словників давав незадовільні результати, оскільки не враховувався зміст речень. Тому в 1966 році інтерес до машинного перекладу текстів значно знизився. Складність перекладу текстів можна пояснити також тим, що вивчення іноземної мови передбачає створення нового світу понять.

Однією з найбільших проблем, яка виникла перед ШІ, був „комбінаторний вибух” – експоненційне збільшення кількості операцій при збільшенні складності задачі. Особливо сильно ця проблема проявилася в задачах розпізнавання образів.

У 1969 році був підданий критиці напрям ШНМ в роботі Мінського і Пейперта, де були показані обмеження перцептронів. Наприклад, ШНМ з двома входами неможливо навчити розпізнаванню такої ситуації, при якій на її входи подаються різні сигнали.

5. *Системи, засновані на знаннях* (період з 1969 по 1980 роки).

Основний підхід до вирішення задач протягом першого десятиліття в області ШІ полягав у механізмі пошуку загального призначення, за допомогою якого намагалися об'єднати в один ланцюжок елементарні етапи формування міркувань. Подібні методи в ШІ отримали назву *слабких*, оскільки погано масштабувалися при збільшенні складності задач.

Альтернативним в порівнянні з слабкими методами став підхід, що передбачає широке використання *знань предметної області*. Одним з перших прикладів такого підходу була програма *Dendral*, яку розробили в Стенфордському університеті Ед Фейгенбаум, Брюс Бьюкенен і Джошуа Леденберг. Ця група займалася проблемою визначення структури молекули на основі інформації, отриманої з мас-спектрометра. Програма *Dendral* працювала за принципом пошуку відомих картин розподілу піків в спектрі мас. Наприклад, пік спектру для маси $m=15$ вказує на присутність метилового фрагменту CH_3 . Значення програми *Dendral* полягало в тому, що це була перша ефективна *експертна система*, заснована на широкому використанні

знань: її здатність вирішувати задачі зумовлена використанням великої кількості правил спеціального призначення.

Керуюсь досвідом, отриманим при розробці програми *Dendral*, Е. Фейгенбаум та інші спеціалісти із Станфордського університету розпочали проект *евристичного програмування* (*Heuristic Programming Project* – HPP), метою якого було перенесення нової методології експертних систем в інші області інтелектуальної діяльності людини.

Наступним кроком була розробка експертної системи *Mycin* для *медичної діагностики*, а саме для діагностики інфекційних захворювань кровоносної системи. Після вводу в систему близько 450 правил *Mycin* ставила діагнози на рівні деяких експертів і краще від недосвідчених лікарів. Система *Mycin* мала наступні особливості: широко використовувалися знання, отримані від експертів; враховувалася степінь невизначеності, якою характеризуються знання в області медицини.

У зв'язку зі збільшенням кількості засобів ШІ, призначених для вирішення проблем реального світу, стали зростати потреби у створенні працездатних схем представлення знань. Було розроблено багато мов для представлення знань і проведення міркувань. Деякі з них були засновані на логіці, наприклад в Європі отримала розповсюдження мова *Prolog* (створена в 1973 році А. Кальмерое), а в США використовувалася мова *Planner*.

В інших мовах, заснованих на висунутій М. Мінським ідеї *фреймів*, був прийнятий структурований підхід, який передбачав збір фактів про конкретні типи об'єктів і подій, а також впорядкування цих типів у вигляді *таксономічної ієрархії*, аналогічної до біологічної *таксономії*.

6. Перетворення штучного інтелекту в індустрію (період з 1980 року по наш час).

Перша успішна комерційна експертна система *R1* з'явилася в компанії DEC (*Digital Equipment Corporation*). Ця програма допомагала складати конфігурації для виконання замовлень на комп'ютерні системи, значну практичну корисність *R1* показала в 1986 році. Сфера використання експертних систем та інших засобів ШІ постійно розширювалася.

В 1981 році в Японії було оголошено про початок проекту створення комп'ютера „*п'ятого покоління*” – 10-річного плану по розробці інтелектуальних комп'ютерів, працюючих під керуванням мови *Prolog*. У відповідь на це в США була сформована корпорація *Microelectronics and Computer Technology Corporation* (МСС) як науково-дослідницький консорціум. В обох випадках ШІ став частиною загального плану, включаючи його використання в проектуванні мікросхем і розробці людино-машинного інтерфейсу. Проте більшість амбіційних планів, поставлених розробниками комп'ютерів п'ятого покоління, досягнуті не були.

7. Повернення до нейронних мереж (період з 1986 року по даний час).

Після спаду інтересу до *нейронних мереж* серед спеціалістів з комп'ютерних наук в кінці 1970-х років у цій області продовжили роботу спеціалісти з інших наукових напрямів. Такі фізики, як Джон Хопфілд, використовували методи із статистичної механіки для аналізу і оптимізації ШНМ. Справжній прорив у напрямку ШНМ стався в середині 1980-х, коли кілька груп одночасно перевідкрили алгоритм навчання ШНМ методом *зворотного розповсюдження помилки*, вперше запропонований в 1969 році Брайсоном і Хо.

ШНМ дали поштовх для розвитку *конекціоністських* (заснованих на зв'язках) моделей інтелектуальних систем, які конкурували з *символьними* моделями. У даний час конекціоністські і символьні підходи є взаємодоповнюючими.

8. Перетворення штучного інтелекту в науку (період з 1987 року по даний час).

У даний час в області ШІ значно частіше зустрічаються роботи, засновані на існуючих теоріях, а не містять принципово нових відкриттів. Твердження, які містяться у цих роботах, засновані на строгих теоремах або надійних експериментальних даних, а не на інтуїції. Такі тенденції, наприклад, спостерігаються в області *розпізнавання мови*. В 1970-ті роки було запропоновано багато архітектур і підходів до розпізнавання, але багато з них не витримали перевірку практикою. В останні роки в області розпізнавання мови домінуючі позиції зайняли підходи, засновані на використанні *прихованих марковських моделей* (*Hidden Markov Model* – НММ). Марковські моделі мають такі переваги: вони засновані на строгій математичній теорії, використовують результати навчання на великому масиві експериментальних даних.

9. Використання інтелектуальних агентів (з 1995 року).

Найбільш відомим прикладом створення повної архітектури агента є робота Аллена Ньюелла, Джона Лерда і Пола Розенблума над проектом SOAR. Важливим прикладом середовища для інтелектуальних агентів є *Інтернет*. Системи ШІ стали настільки розповсюдженими у *Web*, що з'явився термін „*бот*” (скорочення від *робот*). Технології ШІ склали основу багатьох інструментальних засобів Інтернет, таких як машини пошуку, системи створення Web-вузлів та ін. Агентний підхід передбачає, що розумна поведінка формується спільними діями великої кількості відносно простих взаємодіючих напіваавтономних індивідуумів (агентів) [4]. Такі агенти можуть бути нервовими клітинами, індивідуальними особинами біологічного виду або програмами.

В Україні дослідження у сфері ШІ концентрувалися в Інституті кібернетики НАНУ ім. В.М. Глушкова. В 70-80 роках ХХ століття проводилися дослідження в області розпізнавання зображень (В.А.Ковалевський, М.І.Шлезінгер, В.І.Рибак), розпізнавання мови (Т.К.Вінцюк), самонавчання розпізнаючих систем (О.Г.Івахненко), моделювання психіки (М.М.Амосов), теорії автоматів, алгебри алгоритмів, застосування кібернетики в економіці (В.М. Глушков).

Таким чином, у розвитку ШІ можна виділити дев'ять основних етапів (рис. 1.5).

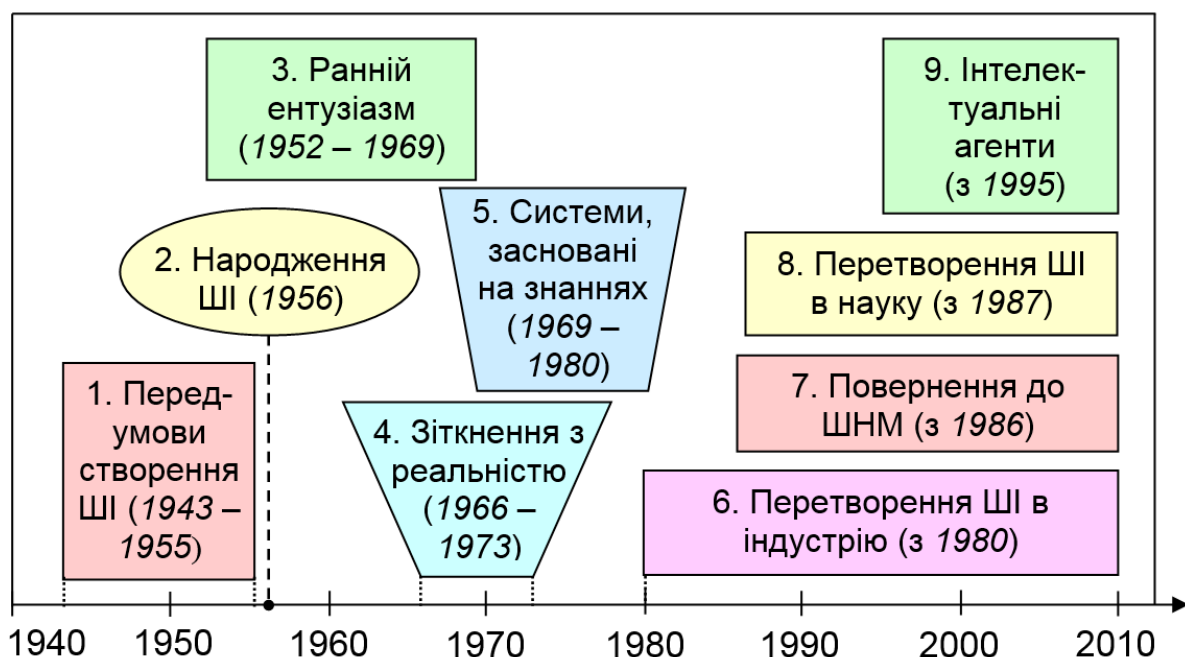


Рис. 1.5. Основні етапи розвитку штучного інтелекту (ШІ)

При цьому історія розвитку окремих напрямів ШІ є досить повчальною (рис. 1.6), коли періоди розвитку і спаду чергуються; розвиток у часі таких напрямів, наприклад ШНМ і машинного перекладу, нагадує меандр (або синусоїду). Тому аналіз історії штучного інтелекту дозволяє не тільки глибше зрозуміти предмет дослідження, але й прогнозувати тенденції розвитку окремих напрямів ШІ.

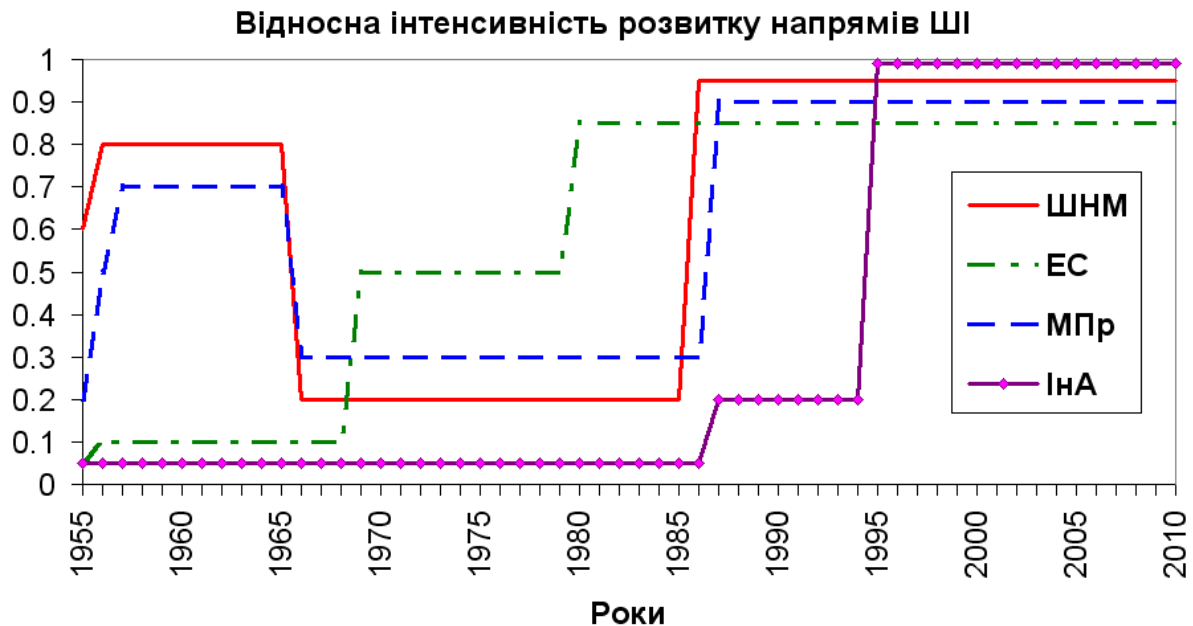


Рис. 1.6. Відносний розвиток основних напрямів штучного інтелекту у часі: ШНМ – штучні нейронні мережі; ЕС – експертні системи; МПр – машинний переклад; Іна – інтелектуальні агенти; 0 – низький рівень; 0.25 – нижче середнього; 0.5 – середній рівень; 0.75 – вище середнього; 1.0 – високий

Історія розвитку ШІ стає ще більш пізнавальною, якщо порівняти розвиток систем ШІ і людини. Одне з перших досліджень у цьому напрямку виконав ще у 1969 році співробітник Каліфорнійського університету (США) Л. Шиклоші, який склав таблицю (табл. 1.3), що здивувала спеціалістів з ШІ.

Ліва половина таблиці Шиклоші описує можливості людини, а права – ШІ. У першому стовпці таблиці наведено середній вік людини, в якому вона здатна повноцінно вирішувати задачі, що вказані у другому стовпці. В третьому стовпці вказано роки створення перших американських програм (четвертий стовпець), які в певній мірі імітували відповідні задачі.

Таблиця Шиклоші

Людина		Штучний інтелект	
Вік, роки	Задача	Рік	Програма
1	Сприйняття навколишнього середовища	1968	Перша програма технічного зору
2	Рухові і маніпуляторні здатності	1968	Проект „око – рука”
3	Мова	1966	Аналізатор мови до 200 слів
		1968	Синтезатор мови
15	Геометрія на площині	1959	Програма, що доводить теореми
15	Алгебраїчні задачі	1964	Програма вирішення алгебраїчних задач
16	Геометричні задачі у просторі	1963	Програма „Аналогія”
17	Програмування для комп'ютера	1963	Програма евристичного програмування
18	Математичний аналіз	1961	Програма для евристичного програмування
19	Числення висловлювань	1959	„Загальний вирішувач задач”
20	Гра в стоклітинні шашки	1959	Програма для гри в шашки
21	Аксіоматична логіка	1956	Програма, що робить висновки у формальній системі

Особливістю таблиці є те, що між віком людини і роками створення програм є явна *обернена залежність*. Те, чим людина оволодіває майже в останню чергу, було запрограмовано для комп'ютерів у першу чергу. Проте ті процедури, які людина починає використовувати для вирішення задач ще у дитячі роки, надзвичайно важко піддаються програмуванню. Тобто для більшості задач ІІІ те, що є простим для людини – складне для комп'ютера, і навпаки.

З часу створення таблиці Шиклоші пройшло багато часу, але виявлена закономірність підтверджується. Успіхи в області програмування шахів та інших комбінаційних ігор стали значними, програмна реалізація різних методів логічного виводу досягла рівня людини. Проте все ще відсутні програми, які могли б з достатньою повнотою аналізувати мовлення, вивчати мови, описувати оточуючий світ у всій його повноті і складності. Це показує, що знайдена закономірність описує об'єктивний факт і вказує на тенденції розвитку систем ІІІ.

1.4. Основні напрями розвитку штучного інтелекту

Сьогодні штучний інтелект – це широка область досліджень і розробок інтелектуальних систем, призначених для роботи в областях діяльності, які погано формалізуються. Системи ШІ, на відміну від жорстко детермінованих комп'ютерних програм, самостійно шукають шляхи вирішення поставленої задачі. При цьому вони можуть змінювати свої параметри і структуру, вдосконалюватися і розвиватися.

В наш час сфера використання штучного інтелекту значно розширилася і сформувалися такі основні напрями ШІ:

1. *Моделі і методи прийняття рішень.* До даного напрямку відноситься планування цілеспрямованих дій і вирішення оптимізаційних задач, зокрема, за допомогою евристичних алгоритмів (систематизованого перебору варіантів). При виборі різних методів вирішення оптимізаційних задач принципове значення має часова оцінка складності алгоритму.

2. *Комп'ютерна лінгвістика.* До даного напрямку відноситься розуміння природної мови, реалізація діалогу між людиною і комп'ютером (фатичного діалогу) і машинний переклад. Перша ідея машинного перекладу полягала у послівному перекладі з однієї мови на іншу. У наш час використовуються складніші структури природно-мовних інтерфейсів, які містять морфологічний аналіз (аналіз слів у тексті), синтаксичний аналіз (аналіз речень, граматики і зв'язків між словами); семантичний аналіз (аналіз змісту кожного речення на основі бази знань програми-перекладача); прагматичний аналіз (аналіз змісту речень залежно від контексту).

3. *Ігри і машинна творчість.* Традиційно в ШІ розглядаються інтелектуальні задачі, які виникають при грі в шашки, шахи, го. В основі цього напрямку лежить лабіринтна модель в поєднанні з евристичними. Крім того, в сучасних ігрових програмах найбільш повно вдалося реалізувати центральні ідеї ШІ: навчання, самонавчання і самоорганізацію. На сьогодні розроблені різні моделі художньої і поетичної творчості, але вони мають більше пізнавальний, ніж практичний інтерес. Найбільший прогрес досягнуто у створенні комп'ютерної музики.

4. *Розпізнавання образів.* До розпізнавання образів в ШІ відноситься широке коло проблем: розпізнавання зображень, звуків, хімічних речовин та ін. Існують системи розпізнавання, засновані на

шаблонних, структурних і ознакових методах, а також нейромережні системи. Розвинуті системи розпізнавання містять бази даних і знань, використовують технології експертних систем, мають здатність до адаптації та навчання. На основі методів розпізнавання зображень сформувалася така наукова дисципліна, як *комп'ютерний зір* (інші назви: машинний або технічний зір). Фундамент комп'ютерного зору поряд з розпізнаванням зображень складає обробка зображень і аналіз сцен; перспективним напрямом є розпізнавання символів і облич.

5. **Штучні нейронні мережі** (нейромережні та нейрокомп'ютерні технології). Цей напрям є альтернативним до систем, заснованих на знаннях. Штучні нейронні мережі в значній мірі запозичують принципи роботи головного мозку людини. Знання в них існують неявно у вигляді ваг синаптичних зв'язків між нейронами. Такі знання не програмуються, а отримуються в процесі навчання (самонавчання). Штучні нейронні мережі використовуються для розпізнавання образів, прогнозування, інтерполяції даних та ін.

6. **Еволюційні та генетичні алгоритми**. Еволюційні алгоритми (еволюційні методи), засновані на принципах природної еволюції, поділяються на три основні напрями: генетичні алгоритми, еволюційні стратегії та еволюційне програмування. Всі напрями еволюційних алгоритмів призначені для вирішення оптимізаційних задач. Для будь-якого з них формується початкова популяція особин, яка піддається селекції та дії генетичних операторів (звичайно схрещування і мутації), що дозволяє знаходити кращі рішення.

7. **Робототехніка**. Роботи – це технічні пристрої, призначені для автоматизації людської праці. Саме слово «робот» з'явилося в ХХ ст., а його автором був чеський письменник Карел Чапек. В наш час в промисловості використовується величезна кількість роботів-маніпуляторів, які працюють згідно з жорсткою схемою керування. На відміну від них інтелектуальні роботи здатні самонавчатися і самоорганізовуватися, адаптуватися до змін навколишнього середовища. Деякі з сучасних роботів створюються у вигляді людиноподібних роботів або планетоходів (рис. 1.8).

8. **Інтелектуальні агенти**. Ставиться задача створення автономної програми – агента, який співпрацює з навколишнім середовищем. Такі програми використовуються, наприклад, для пошуку інформації в Інтернет. Якщо примусити велику кількість «не дуже інтелектуальних» агентів співпрацювати разом, то можна отримати потужніший «мурашиний» інтелект.

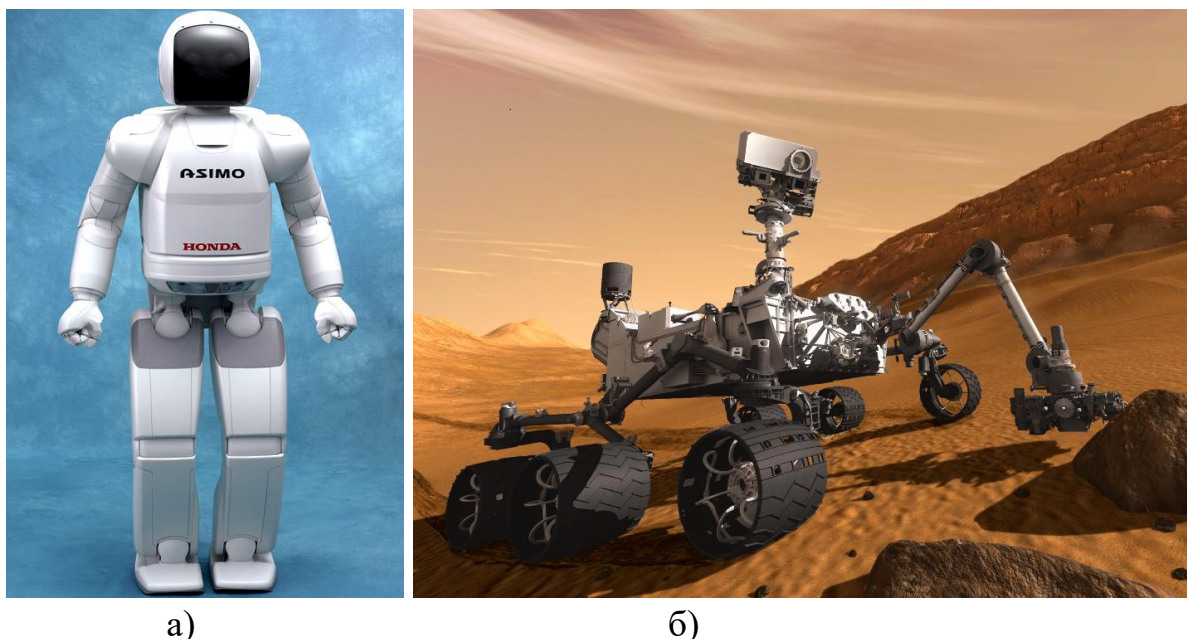


Рис.1.8. Сучасні роботи: а) робот-андроїд Asimo (ріст 130 см, маса 54 кг);
б) робот-марсохід Curiosity (маса 899 кг)

9. Інтелектуальні системи, засновані на знаннях. Експертні системи. Такі системи використовують різні форми представлення знань, виконують формалізацію знань та маніпулювання ними в *базах знань* та *експертних системах*. Моделі знань, що ґрунтуються на вербально-дедуктивній парадигмі, поділяються на чотири класи: семантичні мережі, фрейми, логічні і продукційні моделі. Відомою мовою логічного програмування є Пролог.

10. Обробка недостовірних і нечітких знань. Нечітка логіка створена для формалізації нечітких понять, які зустрічаються в реальних задачах. У нечіткій логіці використовується апарат нечітких (розмитих) множин, які описуються функціями належності і на основі яких виконується нечітке логічне виведення. Завдяки нечіткій логіці можна приймати ефективні рішення навіть в умовах невизначеності.

На даний час питання, чи можливо створити співрозмірний з людським ШІ, залишається відкритим і в цьому напрямі проводяться інтенсивні дослідження. Таким чином, ШІ – перспективна міждисциплінарна область науки, яка швидко розвивається, і в якій можуть проявити свої здібності нові дослідники і першовідкривачі.

1.5. Приклади інтелектуальних задач.

Тест Тьюринга і фатичний діалог

До найпоширеніших інтелектуальних задач, які вирішує людина, відносяться: **розпізнавання образів, мислення та обчислювальні задачі**. В III інтелектуальними вважаються задачі **розпізнавання образів і мислення**, а повністю **обчислювальні задачі** не відносяться до інтелектуальних, оскільки для них існують стандартні методи рішення. Відомий спеціаліст в області III М. Мінський відмітив про задачі, алгоритми вирішення яких вже встановлені, що „...зайвим буде приписувати їм таку містичну властивість, як „інтелектуальність”. Прикладами повністю **обчислювальних задач** є розв’язок системи лінійних алгебраїчних рівнянь, числове інтегрування диференціальних рівнянь та ін. Розглянемо детальніше задачі перших двох типів.

Задача **розпізнавання образів**, зокрема зображень, є життєво необхідною для всіх мислячих істот. Існує два типових завдання **розпізнавання**:

- 1) **ідентифікація** (виділити об’єкт серед інших, наприклад, відділити зображення літери від фону);
- 2) **класичне розпізнавання** (віднести об’єкт до певного класу, наприклад, визначити, до якого символу алфавіту відноситься літера на зображенні).

В моделюванні **мислення**, зокрема при виведенні наслідків з фактів, можна виділити два типи процесів:

- 1) підсвідоме **інтуїтивне** мислення (**образне**);
- 2) **дедуктивні** побудови (**логічне мислення**) (як приклад – дедуктивний метод Шерлока Холмса).

За визначенням **дедукцією** називається перехід від загального до часткового, виведення часткових наслідків із загальних правил. **Дедуктивні** побудови відносно легко можуть бути формалізовані і реалізовані у вигляді програм.

Важливим методом визначення **інтелектуальності** системи є тест, сформульований відомим англійським вченим А. Тьюрингом (**тест Тьюринга**) [3]. За цим тестом визначається момент, починаючи з якого машину можна вважати **інтелектуальною**. Класичне формулювання тесту наступне. Нехай **експерт** за допомогою телефону або подібного віддаленого пристрою спілкується з **об’єктом**, що може бути як людиною, так і машиною. Експерт дає

певні тести-завдання. За результатами відповідей він повинен визначити, з ким має справу – з людиною чи комп'ютером. Якщо він приймає комп'ютер за людину, то комп'ютер може вважатися інтелектуальним. Багато спеціалістів вважали, що подібного тесту цілком достатньо для визначення рівня інтелектуальності комп'ютерної системи. Але виявилося, що це зовсім не так, оскільки тест Тьюринга не передбачає розуміння змісту діалогу.

В основі тесту Тьюринга лежить неявне припущення про те, що необхідною умовою ведення діалогу є розуміння співрозмовника. Але наприкінці 60-х років американський кібернетик Дж. Вейценбаум створив дві програми – ЕЛІЗА і ДОКТОР – для використання в психіатрії. Виявилося, що вони можуть неправильно зорієнтувати експерта в умовах тесту Тьюринга. Тому при побудові таких програм в основу була покладена ідея фатичного діалогу.

Фатичним (від лат. fatuus – нерозумний) називається діалог без розуміння співрозмовника, на основі формального перефразування почутого. Наприклад, на фразу «Я хотів би поїхати у Львів» можна відповісти «Чому Ви хотіли б поїхати у Львів?». У ряді випадків програма може видати заздалегідь заготовлену фразу. Так, на слова експерта „Я знайшов помилку в програмі” можна відповісти „Всі можуть помилятися”.

Такий діалог, звичайно, не має відношення до розуміння співрозмовника, але багато людей, з якими проводили фатичний діалог, вирішили, що мають справу з людиною. Відомий курйозний приклад одного з перших фатичних діалогів. Віце-президент великої американської компанії захотів зв'язатися з відомим системним програмістом Д. Бобровим. Відбулася така телетайпна розмова:

Віце-президент: Чи можу я зараз скористатися комп'ютером?

Бобров: А чому Ви питаєте?

Віце-президент: Може, мені вдасться підписати контракт.

Бобров: Чому Ви в цьому впевнені?

Віце-президент: Мої потенційні покупці ніколи не бачили систему в дії.

Бобров: Це має для вас велике значення?

Віце-президент: Звичайно.

Бобров: Ви маєте в цьому сумнів?

Віце-президент: Що за безглузде зауваження?

Бобров: Що ви маєте на увазі, кажучи про безглузде зауваження?

Віце-президент: Чи можу я зараз скористатись комп'ютером?

Бобров: Розкажіть мені ще про Ваших потенційних покупців.

Віце-президент: Передзвоніть мені 491-1850.

Віце президент був здивований ще більше, коли йому ніхто не зателефонував. Виявилось, що телетайп з'єднував віце-президента не із самим Бобровим, а з програмою, яка реалізувала фатичний діалог.

Звичайно, подібний діалог важко вважати осмисленим, але дати чітке визначення рівня осмисленості також важко.

В основі псевдоінтелектуальних комп'ютерних програм, які здатні підтримувати фатичний діалог із людиною або іншими аналогічними програмами, можуть лежати прості алгоритми зіставлення зі зразками. Наприклад, фатичний діалог із відомою програмою „ДОКТОР” може бути таким:

Доктор: Я Вас слухаю.

Людина: Мене бентежить погода.

Доктор: Як давно Вас бентежить погода?

Людина: Відтоді, як моя сім'я переїхала.

Доктор: Розкажіть про Вашу сім'ю.

Людина: У нас в сім'ї всі захоплюються комп'ютерами.

Доктор: Машини Вас лякають.

Людина: Ні.

Доктор: Будь ласка, не будьте так небагатослівні...

І так далі до безкінечності.

Таким чином, для реалізації діалогу потрібно використати механічне порівняння речень, які вводяться користувачем, зі зразками (шаблонами) речень, що зберігаються програмою (у масиві).

В програмі можна використати один із наступних варіантів порівнянь:

Варіант 1 (повний збіг). Якщо речення, що вводиться, повністю збігається з одним із зразків, то йому відповідає наперед підготована відповідь.

Наприклад:

Людина: Яка система числення використовується в комп'ютерах?

Програма: Використовується двійкова система числення.
або

Людина: Квадрат гіпотенузи дорівнює сумі квадратів катетів.

Програма: Так, але є подібний результат і для непрямокутних трикутників – це теорема косинусів.

Повні збіги, звичайно, трапляються дуже рідко, тому використовуються інші типи порівнянь.

Варіант 2 (використання замінювачів). Типовим є використання замінювачів * і ?. Із замінювачем * зіставляється довільний фрагмент тексту, із замінювачем ? – будь-яке окреме слово.

Наприклад, шаблон (*комп'ютери*) успішно зіставляється з будь-яким реченням, в якому згадується про комп'ютери; шаблон (Я люблю ? яблука) – з такими реченнями, як (Я люблю солодкі яблука) тощо, але не з реченням (Я люблю їсти зелені яблука).

Варіант 3 (надання значень змінним «?» у процесі зіставлення). При цьому можливості програми, що реалізує фатичний діалог, значно розширюються. Вона набуває здатності до генерації відповідей, які залежать від запитань. Так, правило

(Я ? *) → (Що Ви ще ?)

дозволяє на речення (Я люблю яблука) відповісти (Що Ви ще любите?), а на речення (Я ненавиджу дощі) – (Що Ви ще ненавидите?). Безумовно, при використанні українських фраз потрібно стежити за узгодженням суфіксів і закінчень.

Варіант 4 (універсальний зразок). Зі зразком (*) зіставляється будь-яке речення. Звичайно, і відповіді, що відповідають цьому зразкові, повинні бути такими ж універсальними.

Наприклад: (Я Вас не дуже розумію); (Не будьте такими небагатослівними); (Чому це має для Вас значення?).

Варіант 5 (зіставлення більш ніж з одним зразком). Речення може зіставлятися не з одним зразком, а з кількома. Наприклад, якщо в програмі задані підстановки

(*люблю*) → (Що Ви ще любите?)

та

(*комп'ютери*) → (Ви маєте здібності до техніки),

то речення (Я люблю комп'ютери) зіставляється з обома зразками (випадковим чином вибирається один зі зразків).

Прикладом програми, яка реалізує фатичний діалог, є „ELIZA” (програма вільно розповсюджується), створена в середовищі Delphi на основі однойменної програми Дж. Вейценбаума (рис. 1.9).

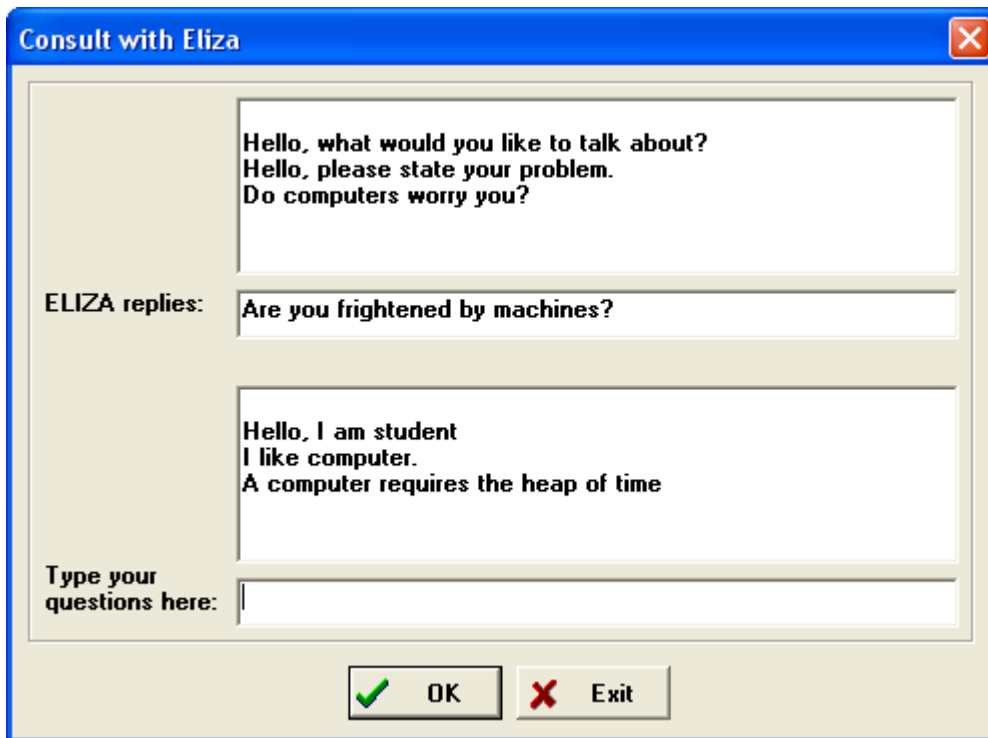


Рис. 1.9. Екранна форма програми „ELIZA” з фатичним діалогом

Фатичний діалог може бути досить досконалим, але системи, які здатні тільки на підтримку такого діалогу, не можна вважати інтелектуальними. Розуміння співрозмовника абсолютно необхідне для нормального діалогу, а для такого розуміння необхідно мати певну суму знань про світ.

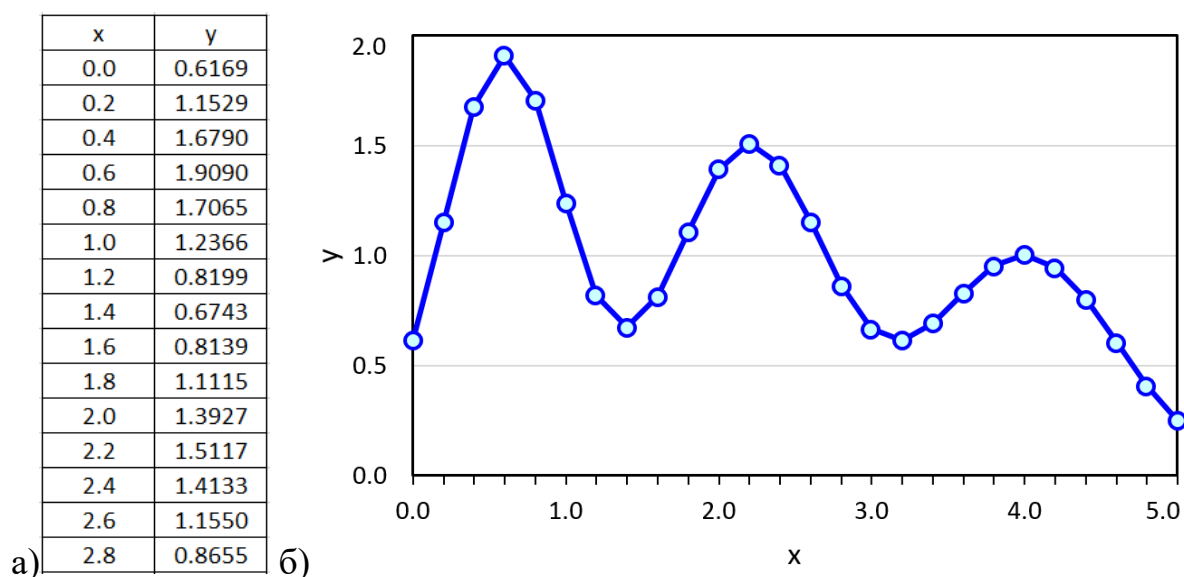
2. МЕТОДИ РОЗПІЗНАВАННЯ ОБРАЗІВ

2.1. Біологічні аналоги систем розпізнавання образів

Розпізнавання образів (РО) – дуже складна наукова і технічна проблема, тому один з методів її вирішення полягає у моделюванні біологічних аналогів систем розпізнавання образів – органів зору людини і тварин (біоніка). Людина отримує інформацію про образи за допомогою органів чуттів: зору, слуху, дотику, смаку та нюху, проте найважливішим каналом отримання інформації є зір [3, 7, 8-14].

Важливо, що проблематика розпізнавання зображень виникає навіть в областях, безпосередньо не пов'язаних з зором. Пояснюється це, ймовірно, тим, що кількість інформації, яка постачається через зір, на три порядки перевищує кількість інформації, яка постачається іншими органами чуттів. Через таку високу пропускну здатність зорового каналу люди намагаються інформацію з більшості джерел не зорової природи також перетворити у видиму форму. Наприклад, людина звичайно краще і швидше аналізує графіки і діаграми, ніж самі дані у табличній формі (рис. 2.1). Про особливості сприйняття зображень людиною американський фізіолог Р. Хабер сказав так: „Картинки залишаються в пам'яті зовсім не у вигляді слів”.

Розпізнавання образів на практиці реалізується людиною та тваринами, але самі принципи ще не розкрито. Малоефективними виявилися спроби розкрити суть розпізнавання через самоаналіз через те, що, напевно, більшість процесів розпізнавання відбувається підсвідомо.



Зображення сприймаються людиною за допомогою системи зору, яка як сенсори застосовує світлочутливі сітківки очей, що містять по 125 млн. рецепторів (рис. 2.2). Фоточутливі сенсори (палички та колбочки) розміщені в сітківці ока нерівномірно: найвища щільність сенсорів у центральній частині ока (максимальна роздільна здатність), а при наближенні до периферії щільність сенсорів зменшується (рис. 2.3). При цьому палички є чутливими до яскравості світла, а колбочки – до кольору.

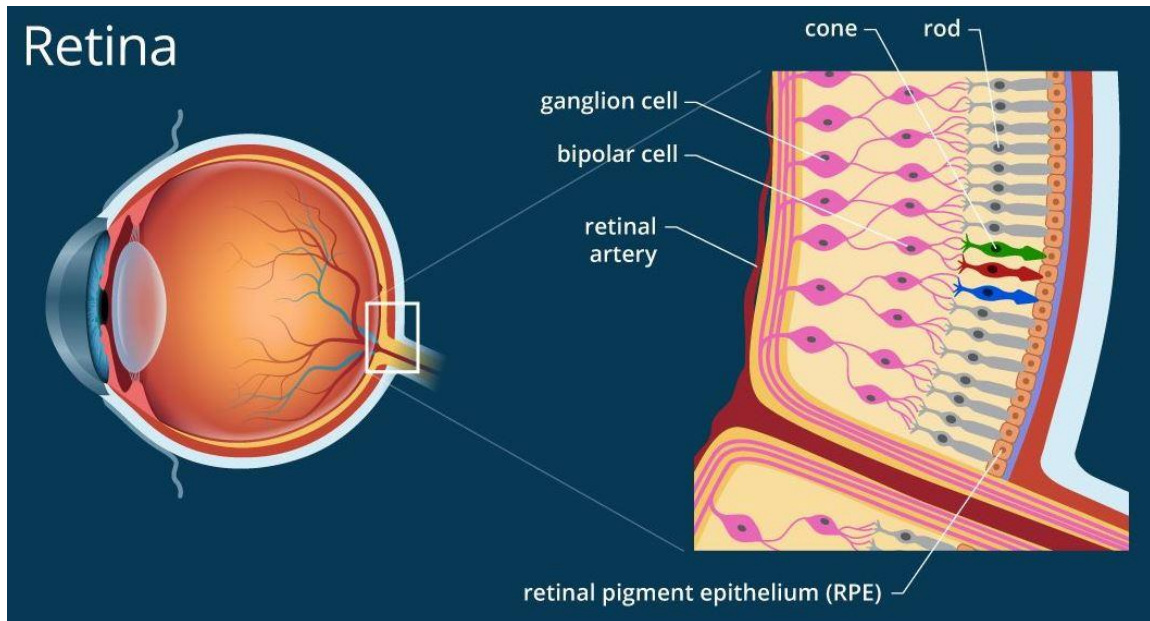


Рис. 2.2. Будова сітківки ока людини; retina – сітківка, cone – колбочка, rod – палочка, ganglion cell – гангліозна клітина, bipolar cell – біполярна клітина; біполярні та гангліозні клітини реагують на перепад яскравості зображення у межах певного околу [1]

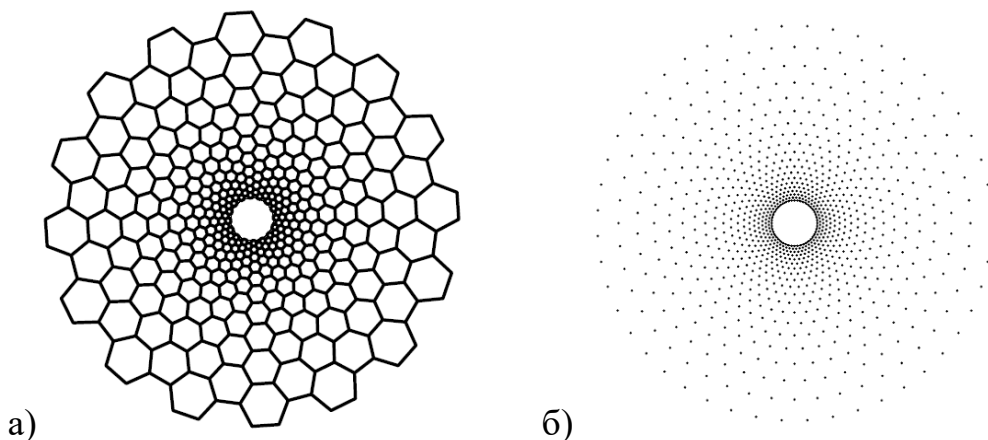


Рис. 2.3. Щільність просторового розміщення фоточутливих сенсорів сітківки ока людини: а) умовні шестикутники відповідають однаковій кількості сенсорів; б) просторове положення центрів шестикутників [13]

Інформація з очей передається в мозок через зоровий нерв, який містить 80 тис. волокон. Особливістю ока як сенсора світла є те, що воно чутливе до рівня освітленості у логарифмічному масштабі, завдяки чому сприймає зображення при зміні освітленості в 10^8 разів [13-14].

Система зору сприймає предмет не миттєво, а у певній послідовності, яку можна розділити на три етапи розпізнавання: 1) з'явився об'єкт; 2) поверхнево оцінити об'єкт; 3) детально оцінити (рис. 2.4).

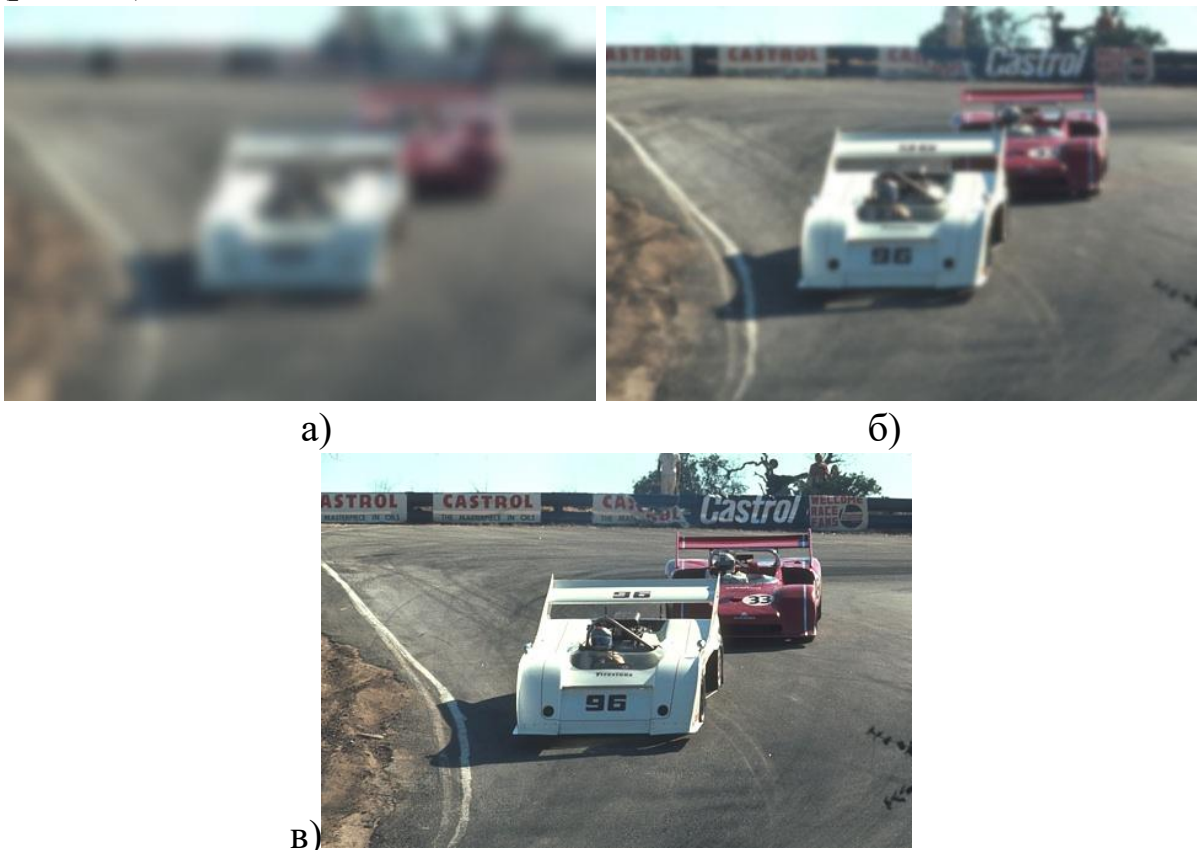


Рис. 2.4. Зміна деталізації зображення у процесі сприйняття: а) мінімальна деталізація; б) середня деталізація; в) максимальна деталізація

Розпізнавання образу для нетренованого спостерігача займає приблизно 0.15 с, а для тренуваного (який готовий до появи об'єкта певного типу) – 0.06 с.

Права та ліва півкулі мозку людини, ймовірно, по різному беруть участь в обробленні зображень [13, 14]. Деякі дослідження показують, що права півкуля мозку в основному відповідає за розпізнавання образів і аналіз форми об'єктів, а ліва – за логічні побудови і мову. При цьому форма об'єктів на зображенні описується положенням частин фігури одна відносно одної.

При сприйнятті зображень система зору людини реагує в першу чергу на контури (перепади яскравості та кольору), а також на змістові центри (області з різкою зміною контуру, наприклад, кути). Завдяки руху очей у кожен момент часу система зору сприймає певний фрагмент зображення.

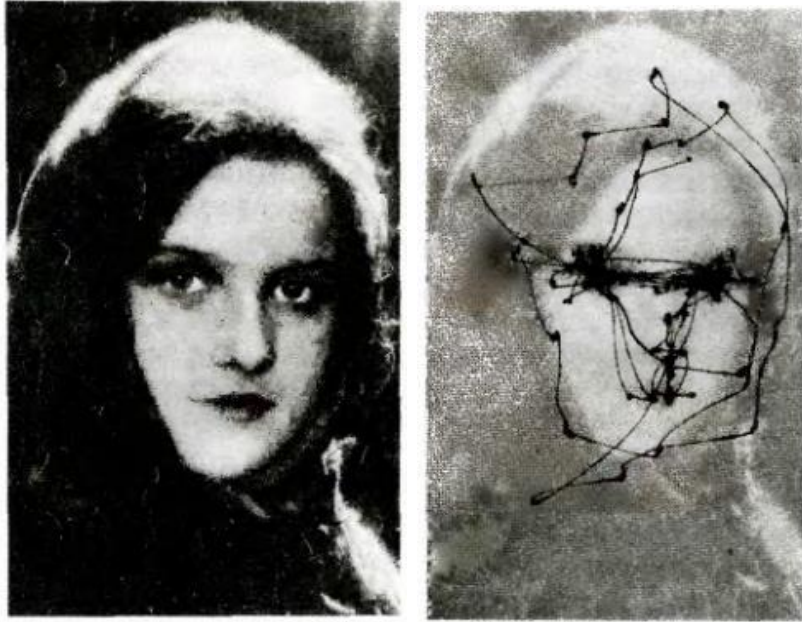


Рис. 2.5. Зміна напрямку зору при розпізнаванні зображення

При сприйнятті зображення система зору людини будує на його основі цілісний образ (гештальт) об'єкту навіть у тому випадку, якщо на зображенні присутня тільки частина об'єкту (рис. 2.6).

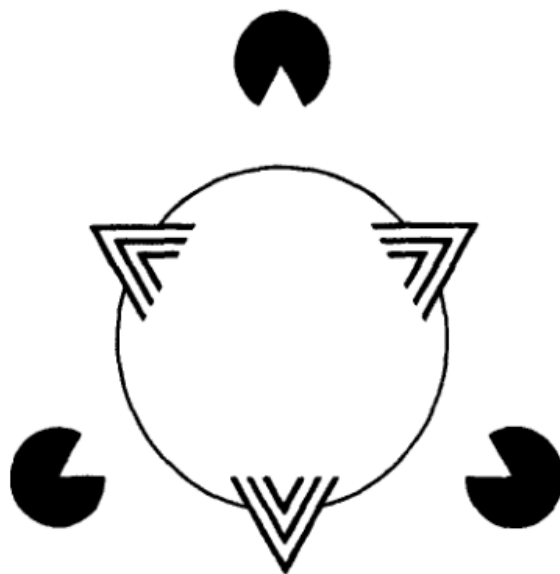


Рис. 2.6. Перекривання чорних та білого трикутників

Система зору людини є недосконалою, для неї характерні різноманітні оптичні ілюзії. Наприклад, ілюзія «два обличчя – ваза», ілюзія «залізничних шляхів» та ін. [13-14] (рис. 2.7, рис. 2.8). Важливо, що при розпізнаванні зображення одночасно сприймається або фігура (об'єкт), або фон (рис. 2.7). Вивчення таких ілюзій може допомогти у вивченні принципів роботи системи зору людини.

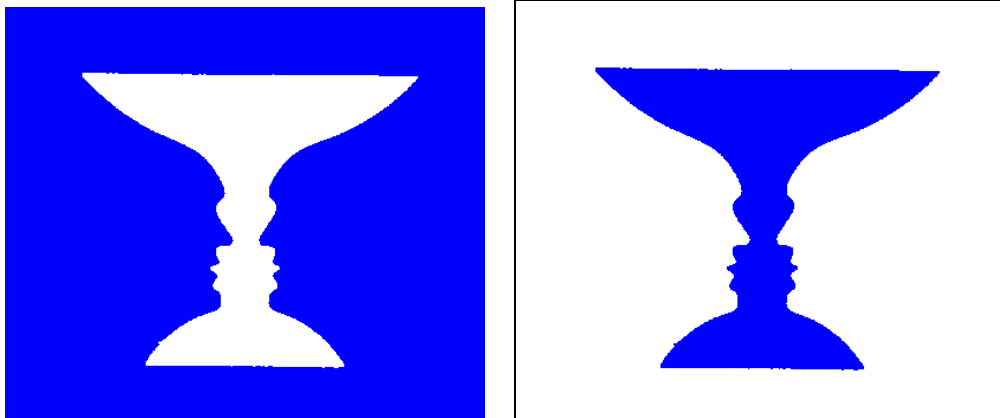


Рис. 2.7. Ілюзія «два обличчя – ваза»

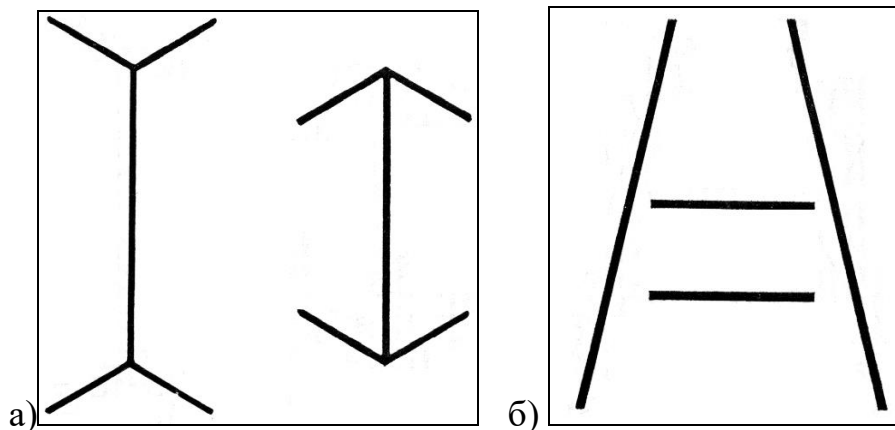


Рис. 2.8. Ілюзії зору людини: а) ілюзія Мюллер-Лайера або ілюзія стріли: права стріла здається коротшою; б) ілюзія Понцо або ілюзія залізничних шляхів: верхня горизонтальна лінія здається довшою

На даний час існує 2 основні гіпотези розпізнавання зображень у системі зору людини [10]:

1. Детекторна полягає в тому, що зображення можна поділити на відносно прості базові елементи (примітиви): лінії, дуги, точки. Для цього в корі мозку є структури, які виділяють базові елементи зображень.
2. Просторово-частотна. Значення сигналу аналізується у певному околі, над яким виконується кусковий (просторово локальний) частотний Фур'є-аналіз. Такий підхід також називається квазіголографічним.

Дослідження показали, що голографічна пам'ять (наприклад, на основі товстої фотоемульсії) і мозок людини мають деякі спільні властивості:

1. Запис кожної порції інформації відбувається не на певній ділянці середовища, що запам'ятовує, розміри якого відповідали б місткості інформаційною пакету, а в ширшій області, що значно перевищує потрібні для цього пакету розміри.
2. Запис і читання інформації відбуваються не поелементно, а відразу у вигляді двовимірних масивів, образів.
3. У одній і тій же області середовища, що запам'ятовує, зберігається множина образів.
4. Характерне однакове реагування на локальні пошкодження пам'яті – не зникнення якоїсь конкретної інформації, що властиве іншим системам, а лише загальне зниження відношення сигнал/шум.

Розрізняють розпізнавання (рис. 2.9) та класифікацію (рис. 2.10) зображень: при розпізнаванні ознаки об'єктів не формалізуються.

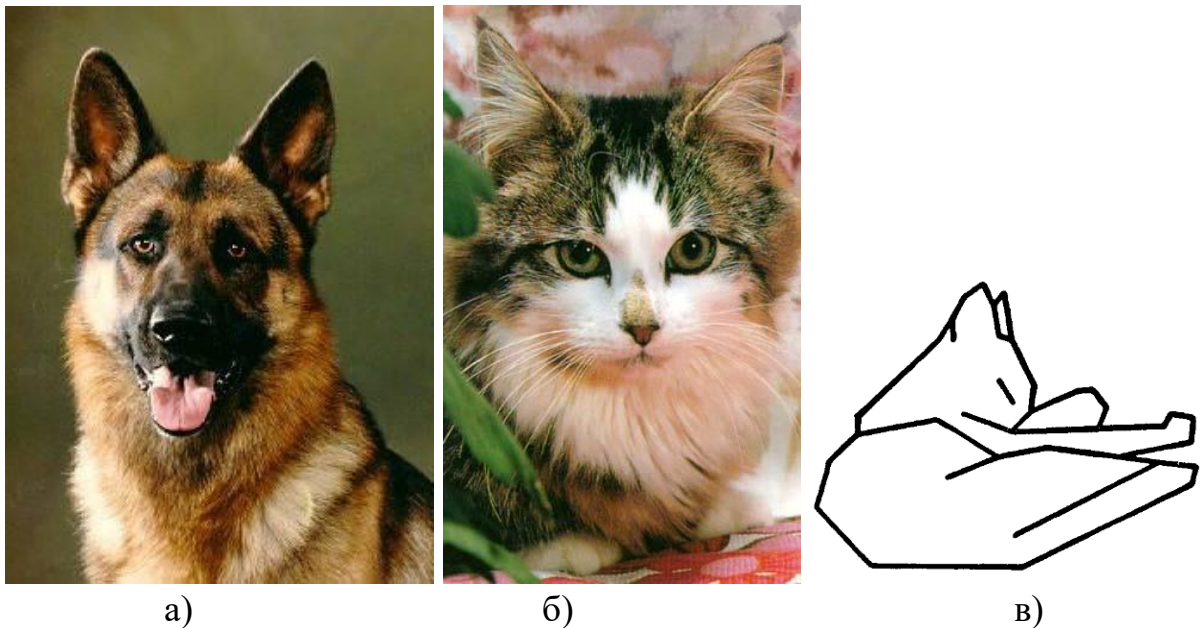


Рис. 2.9. Приклад зображень собаки (а), кішки (б) та стилізованого зображення кішки (в)

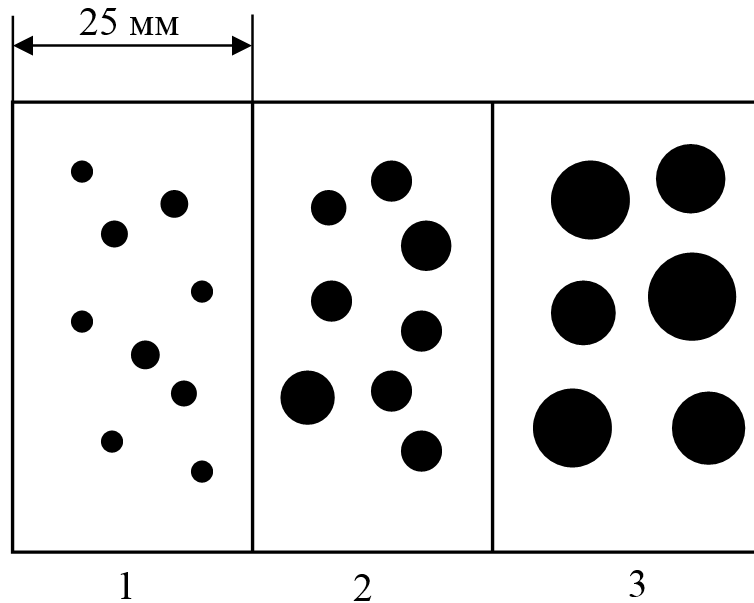


Рис. 2.10. Приклад класифікації зображень кругів з діаметром d :
 1) $d < 3$ мм; 2) $3 \text{ мм} \leq d < 6$ мм; 3) $d > 6$ мм

Перевагою системи зору людини є можливість правильного розпізнавання об'єкта навіть при значній зміні його кольору, розміру, повороту, форми та ін. (рис. 2.11). Система зору людини дозволяє також розпізнавати об'єкти, які за кольором і текстурою практично не відрізняються від фону (рис. 2.12). Тому важливим завданням для систем комп'ютерного зору є розробка таких високоефективних апаратно-програмних засобів розпізнавання образів.



Рис. 2.11. Приклади двох фотографій однієї особи:
 а) у відтінках сірого; б) кольорова

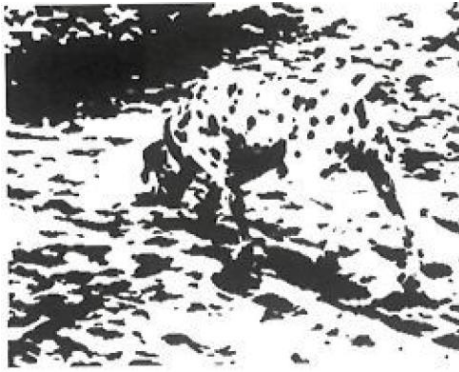


Рис. 2.12. Приклад зображення об'єкта, який за кольором і текстурою майже не відрізняється від фону

Складність комп'ютерного розпізнавання деформованих зображень застосовується в тесті CAPTCHA (англ. «Completely Automated Public Turing test to tell Computers and Humans Apart» – повністю автоматизований публічний тест Тюринга для розрізнення комп'ютерів і людей) для виявлення інтернет-ботів (рис. 2.13).



Рис. 2.13. Приклади зображень, що застосовуються в тесті CAPTCHA

В системі зору людини зображення легко розпізнаються, якщо вони повернуті відносно відомих на кут до 15° (рис. 2.14).

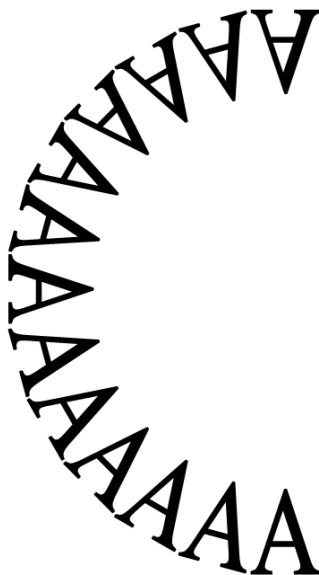


Рис. 2.14. Приклади зображень літер, повернуті на кути, кратні 15°

Розглянуті особливості розпізнавання образів у біологічних системах застосовуються при побудові систем комп'ютерного зору.

2.2. Задача розпізнавання образів

Образи – це об’єкти і явища навколишнього середовища або продукти розумової діяльності людини. Прикладом образів можуть бути: люди (їх зображення, голоси, почерки, пульс); літери; літаки, автомобілі, рослини, тварини; суспільні, економічні явища і процеси.

Ознаки образу – це якісні та кількісні параметри, сукупність яких дозволяє відрізнити один образ від іншого. Ознаки можуть бути суттєвими і несуттєвими, кількісними, якісними, апіорними, апостеріорними. Відомі ознаки до досліду – апіорні, а в результаті досліду отримуються апостеріорні або емпіричні ознаки.

Початковий словник ознак – перелік ознак образу, які отримуються за допомогою сенсорів.

Однотипні образи – група образів з кінцевою кількістю спільних ознак, об’єднані приблизно однаковими функціями (букви, літаки, квіти...) (рис. 2.15, рис. 2.16). Всі класи однотипної групи образів формують її алфавіт.

Клас образів – це підмножина образів з найбільш схожими суттєвими ознаками.

Розпізнавання образів (РО) – процес поділу образів на класи.

Мета РО – розробка системи РО, яка б наближувалася по своїм можливостям до системи зору людини.

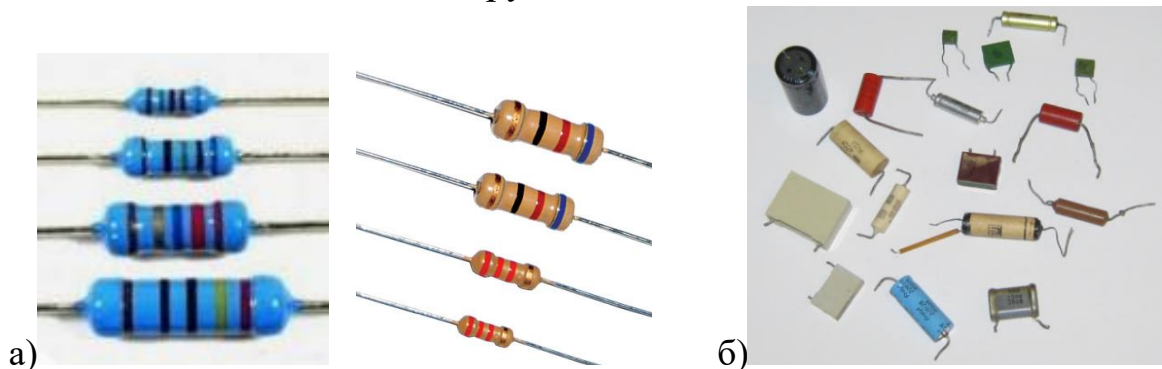


Рис. 2.15. Приклад зображень однотипних образів «радіоелектронні компоненти»: а) резистори; б) конденсатори

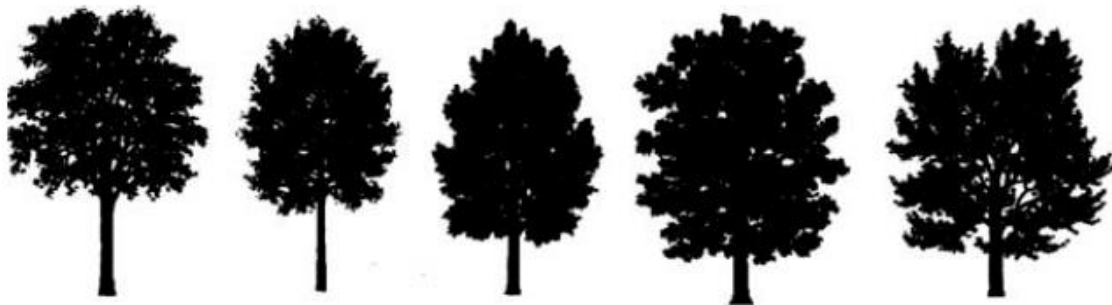


Рис. 2.16. Приклад зображень однотипних образів «форми крон дерев»

Математично задача розпізнавання образів формулюється так. Дано множину об'єктів T_i , $i = 1, \dots, q$ (кількість об'єктів q може бути практично безкінечною). Всі об'єкти поділені на класи (підмножини) C_k , де $k = 1, \dots, m$; кожному класу відповідає один або кілька еталонів. Об'єкти задаються значеннями ознак X_j , $j = 1, \dots, n$; тому $T_i = \{X_1(T_i), \dots, X_n(T_i)\}$.

Задача розпізнавання полягає у знаходженні степені належності об'єкту T_i до кожного з класів C_k . Об'єкт T_i вважається належним до того класу C_k , для якого коефіцієнт (ступінь) належності $K(T_i, C_k)$ максимальний.

Наприклад, розглянемо множину об'єктів T_i „Транспортні засоби”.

Ознаки X_j : кількість коліс, наявність двигуна, вага вантажу.

Класи C_k : Велосипед, мотоцикл, легковий автомобіль, вантажний автомобіль, інший (табл. 2.1).

Таблиця 2.1

Значення ознак X_j для класів C_k

	Ознака	X_1	X_2	X_3
Клас		Кількість коліс	Наявність двигуна (0/1)	Вага вантажу, кг
C_1	Велосипед	2	0	<100
C_2	Мотоцикл	2, 3	1	<200
C_3	Легковий автомобіль	4	1	(>200) AND (<500)
C_4	Вантажний автомобіль	4, 6, 8, 10	1	>1000
C_5	Інший		0	

На основі описаних класів C_k (табл. 2.1) можливе розпізнавання невідомих об'єктів T_i (табл. 2.2).

Таблиця 2.2

Розпізнавання нових транспортних засобів T_1, T_2

	Ознаки	$T_1 (3, 1, 150)$	$T_2 (4, 0, 50)$
Клас		$K(T_1, C_k)$	$K(T_2, C_k)$
C_1	Велосипед	0	2
C_2	Мотоцикл	3	1
C_3	Легковий автомобіль	2	1
C_4	Вантажний автомобіль	1	1
C_5	Інший	0	1

З поняттям „розпізнавання образів” тісно пов’язані задачі класифікації. При цьому виникає задача правильного вибору **ознак** об’єкта.

Типові задачі **класифікації** наступні:

1. Задача ідентифікації (вирізнити конкретний об’єкт серед інших).
2. Віднесення об’єкту до певного класу.
3. Кластерний аналіз (класифікація без вчителя).

Типові задачі **розпізнавання**:

1. Технічна діагностика (виявлення дефектів).
2. Медична діагностика (аналіз кардіограм).
3. Розпізнавання літер (друкований і рукописний текст).
4. Розпізнавання мови.
5. Робототехніка (технічний зір).
6. Охоронні системи (розпізнавання відбитків пальців, облич).

Послідовність **розпізнавання** така:

1. Вибір інформативних для кожного класу C_k .
2. Попередня обробка об’єкту T_i (зменшення шумів, сегментація, ...)
3. Обчислення ознак X_j для об’єкту T_i .
4. Розпізнавання, визначення належності об’єкта T_i до класу C_k .

Режими розпізнавання наступні:

1. Режим навчання (підготовчий режим, обчислення ознак X_j для класів C_k).
2. Робочий режим (розпізнавання): визначення, до якого класу C_k належить об’єкт T_i .

2.3. Попередня обробка об'єктів розпізнавання

Перед розпізнаванням об'єктів (з метою підвищення точності розпізнавання) часто проводять їх нормалізацію, зменшують шум, виділяють найбільш інформативні ознаки, ділять об'єкт на характерні області. Розглянемо основні етапи попередньої обробки об'єктів розпізнавання на прикладі зображень.

Попередня обробка зображень містить такі етапи:

1. Дискретизація (оцифрування), поділ на пікселі з певним кроком.
2. Зменшення шумів і підвищення якості зображення.
3. Сегментація.

Відрізняють 3 випадки представлення **цифрових зображень**:

1. Монохромні.
2. Півтонові (у відтінках сірого).
3. Кольорові.

Сегментація зображень полягає у поділі зображення на складові частини, що мають спільні властивості [13-14]. Існує велика кількість алгоритмів сегментації зображення, але більшість з них можна поділити на дві групи, кожна з яких використовує фундаментальні властивості зображення – подібність та відмінність. Тому існує два основних підходи до сегментації:

- метод пошуку однорідних областей (метод нарощування областей);
- метод виділення меж сегментів (контурів зображення) (метод водорозділів).

Розглянемо сегментацію за методом пошуку однорідних областей, який використовує властивість S_I зображення (колір, текстура, рівень яскравості). Якщо цифрове зображення представлено у відтінках сірого, то математично воно описується матрицею $f(i, k)$, де $i = 1, \dots, M$; $k = 1, \dots, N$; M – висота зображення у пікселях, N – ширина зображення у пікселях. Кольорове зображення описується матрицею $f_{RGB}(i, k, c)$, де $i = 1, \dots, M$; $k = 1, \dots, N$; $c = 1, \dots, 3$; c – номер каналу кольору (червоного, зеленого та синього відповідно).

Нарощування областей полягає в тому, що сусідні елементи з однаковими або близькими рівнями яскравості групують, об'єднуючи їх в однорідні області. При цьому слід уникати помилок невірною визначення сусідніх елементів (наприклад, за рахунок шуму) в залежності від вибраної області фон Неймана (чотирьохзв'язної) або області Мура (восьмизв'язної), які описують оточення кожного пікселя.

Сегментація зображень по параметрам яскравості враховує те, що кожний сегмент зображення відділяється від інших сусідніх. Для сегментації методом порогового розподілу необхідно отримати бінарне зображення $f_B(i, k)$ з напівтонового $f(i, k)$. Для цього встановлюється деяке порогове значення T . Якщо $f(i, k) > T$, то значення $f_B(i, k) = 1$, інакше $f_B(i, k) = 0$. Для більшості цифрових зображень значення яскравості їх пікселів описується 8 бітами даних (256 градацій яскравості від 0 до 255), тому як початкове значення порогу можна вибрати $T = 127$.

На початку сегментації на зображенні вибирається початковий піксель сегмента, для якого $f_B(i, k) = 1$ (сегмент № 1). У процесі сегментації до сегмента додаються сусідні пікселі, для яких також $f_B(i, k) = 1$. Якщо такі пікселі відсутні, то вибирається початковий піксель наступного сегмента і виконується його нарощування. У результаті сегментації всі пікселі зображення з $f_B(i, k) = 1$ поділяються на сегменти, пікселі з $f_B(i, k) = 0$ вважаються фоном.

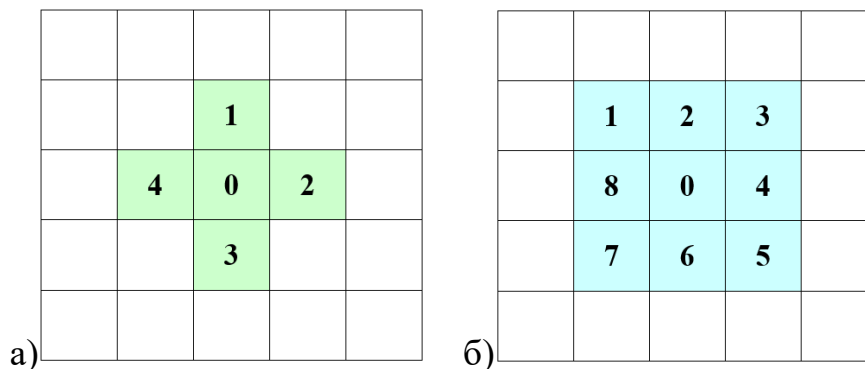


Рис. 2.17. Вигляд чотирьохзв'язної області фон Неймана (а) та чотирьохзв'язної області Мура (б); 0 – початковий піксель

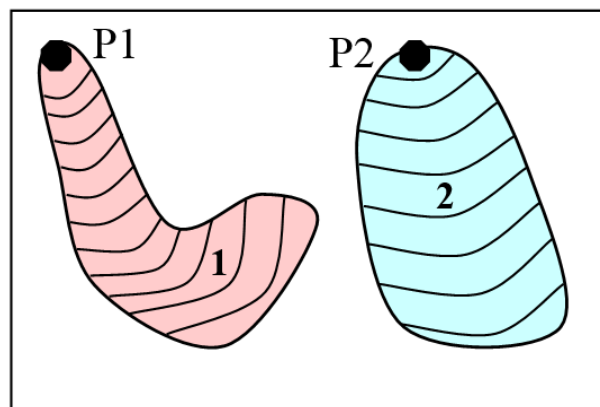


Рис. 2.18. Приклад сегментації зображення методом нарощування областей; P1, P2 – початкові пікселі сегментів 1 і 2 відповідно

Метод водорозділів є ефективним при сегментації зображень з чіткими контурами. На зображенні спочатку виділяються контури, а потім знаходяться межі сегментів. Назва методу походить з географії. Метод водорозділу використовує такий принцип поділу, але замість води ділиться зображення на сегменти (рис. 2.19).

Зображення обробляється у вигляді поверхні рівнів, де значення яскравостей пікселів інтерпретуються як їх висоти. Якщо уявити щоб над такою поверхнею, то вода буде збиратися в двох областях, відмічених як водозбірні басейни.

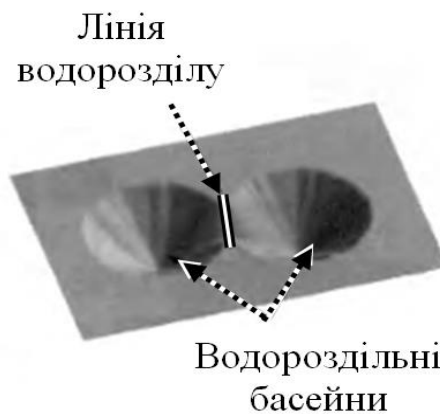


Рис. 2.19. Принцип роботи методу водорозділів

2.4. Шаблонні методи розпізнавання образів

Шаблонні системи перетворюють зображення окремого символу (зразка) в растрове, порівнюють його з усіма шаблонами (еталонами) бази і вибирають шаблон, який найбільше збігається із вхідним зображенням. До шаблонних методів розпізнавання (методів суміщення з еталоном) відносять, зокрема, кореляційний метод розпізнавання та метод допустимих перетворень [10].

2.4.1. Кореляційний метод розпізнавання

Даний метод розпізнавання реалізується програмно в комп'ютерних системах або засобами оптичних систем.

В оптичних системах кореляційний метод розпізнавання реалізується за допомогою фотомасок еталонів. Зображення об'єкта, що розпізнається (наприклад, літери), проектується на еталонні маски. Кожна маска – це трафаретний отвір (або прозора частина) у непрозорому матеріалі. Пучок променів світла освітлює об'єкт (який звичайно має темний колір на білому фоні) і через оптичну систему

направляться через маску на фотодетектор, розміщений за еталонною маскою. Чим більше темний об'єкт збігається з еталоном, тим менше освітлюється фотодетектор. В ідеалі при повному збігові зображення об'єкта з еталоном освітлення фотодетектора мінімальне, і вихідний сигнал дорівнює нулю, що є ознакою розпізнавання образу. Еталонні маски подаються до порівняння по порядку, тому відомо, з яким еталоном збігся об'єкт.

Оптична система РО з фотомаскою має такі недоліки: деформація об'єкта призводить до хибної класифікації; об'єкт повинен бути строго визначених розмірів і знаходитися на визначеній відстані від оптичної системи; потрібне дуже точне центрування схеми.

Програмна реалізація кореляційного методу РО передбачає перетворення зображень об'єктів та еталонів у цифрову форму. Розглянемо випадок розпізнавання зображень у відтінках сірого, коли зображення зразків та еталонів записуються в прямокутні матриці f_S та f_E відповідно (розміром $M \times N$ пікселів). Матриця f_S зображення зразка описується формулою:

$$f_S = \begin{bmatrix} f_S(1,1) & \dots & f_S(1,k) & \dots & f_S(1,N) \\ \dots & \dots & \dots & \dots & \dots \\ f_S(i,1) & \dots & f_S(i,k) & \dots & f_S(i,N) \\ \dots & \dots & \dots & \dots & \dots \\ f_S(M,1) & \dots & f_S(M,k) & \dots & f_S(M,N) \end{bmatrix}. \quad (2.1)$$

Матриця f_E зображення еталону описується аналогічною формулою. Яскравість зображень зразка та еталону нормується у межах від 0 до 1.

При обробленні кольорових зображень для кожного каналу кольору RGB (червоного, зеленого, синього) створюється окрема прямокутна матриця (f_R , f_G та f_B) або створюється тривимірна матриця f_{RGB} для збереження всіх трьох складових кольору. Обробка матриць складових кольору виконується аналогічно, як і матриць яскравості.

Для зображень у відтінках сірого коефіцієнт подібності K_C зображень зразка і еталону обчислюється як скалярний добуток матриць f_S та f_E за формулою

$$K_C = \sum_{i=1}^N \sum_{k=1}^M (f_S(i,k) \cdot f_E(i,k)), \quad (2.2)$$

де N, M – розміри зображення в пікселях.

Коефіцієнт подібності K_C називається також «коефіцієнт кореляції» (за аналогією з подібними обчисленнями в теорії ймовірності).

Задача розпізнавання полягає у знаходженні еталона f_E , який має найбільший коефіцієнт кореляції із зображенням зразка f_S .

Кореляційний метод у програмній реалізації має такі недоліки: метод може застосовуватися, коли зображення одного і того ж класу мають схоже накреслення і сталі розміри; метод чутливий до товщини і контрасту ліній, на результат класифікації впливає зміщення зображення відносно точного положення. Частково такі недоліки можна усунути шляхом використання для одного класу кількох зображень еталонів (наприклад, різного розміру).

2.4.2. Метод допустимих перетворень

У даному методі за рахунок певних перетворень зображення зразка мінімізується різниця між зображеннями. Шляхом математичної обробки компенсуються такі спотворення зображення зразка відносно еталона (рис. 2.20):

- 1) зміщення (центрування);
- 2) масштабування;
- 3) поворот;
- 4) деформація.



Рис. 2.20. Допустимі перетворення зображення символу „А”:

a – зміщення; $б$ – масштабування; $в$ – поворот; $г$ – пошкодження

Для суміщення зображень при наявності спотворень потрібно виконати відповідні допустимі перетворення зображення зразка S (однієї матриці відносно іншої) для знаходження максимального збігу з еталоном E , тобто максимального значення коефіцієнта подібності K_C . Значення коефіцієнта подібності K_C розраховуються, наприклад,

через нормовану середньоарифметичну різницю яскравостей зображень зразка і еталона:

$$K_C = 1 - \Delta = 1 - \frac{1}{NM} \sum_{i=1}^N \sum_{j=1}^M \left(\frac{|S_{ij} - E_{ij}|}{\max} \right), \quad (2.3)$$

де S_{ij} – яскравість пікселя зображення зразка;

E_{ij} – яскравість пікселя зображення еталона;

N, M – розміри зображення;

$\max = 255$ – максимальне значення яскравості зображень;

Δ – нормована середньоарифметична різниця яскравостей зображень зразка і еталона.

Перетворення зображення зразка поділяються глобальні (наприклад, поворот всього зображення на однаковий кут) і локальні (для різних областей зображення перетворення будуть різними).

Зображення належить до того класу (еталону), для якого коефіцієнт подібності максимальний (табл. 2.3).

Існують адаптивні алгоритми суміщення зображення, які виділяють контури об'єктів і центри ваги. В такому випадку використовується суміщення центрів ваги оригіналу й зображення методом найменших квадратів різниці координат центрів ваги.

Таблиця 2.3

Коефіцієнти подібності K_C при порівнянні зображень зразків з еталонами

Класи	Зображення еталонів	Зразки			
		№ 1	№ 2	№ 3	№ 4
					
A		<u>1.0</u>	<u>0.9</u>	0.4	0.3
B		0.4	0.3	<u>1.0</u>	0.25
C		0.3	0.25	0.25	<u>1.0</u>

При повороті зображення зразка відносно еталона для деякого кута повороту збіг зображень буде максимальним (рис. 2.21).

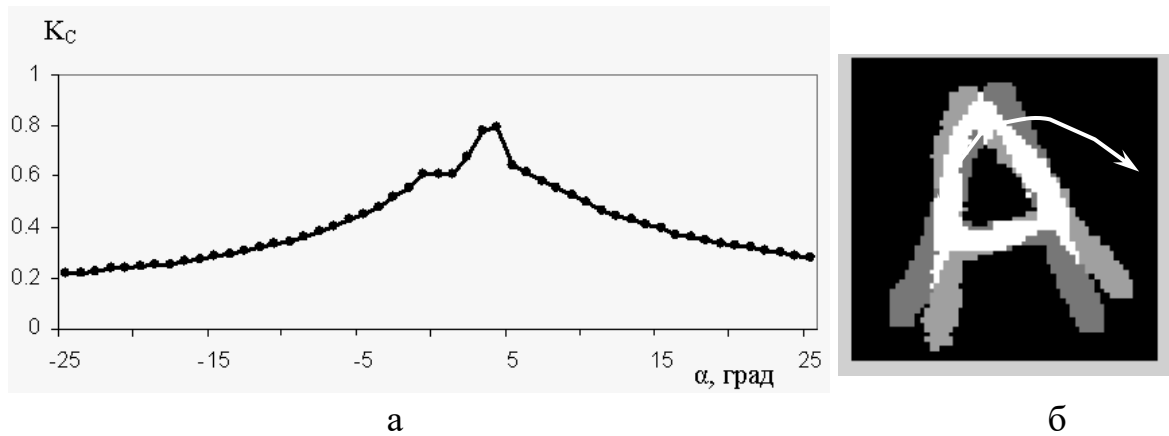


Рис. 2.21. Залежність коефіцієнтів подібності K_c від кута повороту α (а) для зображень (б)

При зсувах зображення зразка відносно еталона отримується матриця коефіцієнтів подібності K_c . У тому випадку, якщо зображення практично повністю збігаються, відрізняючись тільки зсувом, то на матриці коефіцієнтів подібності спостерігається чіткий пік (рис. 2.21 а). Якщо зображення зразка і еталона значно

Для суміщення зображень при наявності спотворень потрібно виконати відповідні допустимі перетворення зображення (однієї матриці відносно іншої) для знаходження максимального співпадання з еталоном (сума квадратів різниць координат відповідних точок зображень мінімальна.).

2.5. Структурні методи розпізнавання

В структурних системах об'єкт описується як граф, вузлами якого є елементи вхідного об'єкта, а дугами – просторові відношення між ними [10]. Такі системи звичайно працюють з векторними зображеннями. Наприклад, в такому представленні буква „Р” – це вертикальний відрізок і дуга.

Послідовність обробки зображень така:

1. Виконується сегментація зображення на складові частини (фрагменти) – відрізки. Виділяються елементарні елементи зображення (атоми), які складають алфавіт символів.
2. Після виділення атомарних елементів починається синтаксичне розпізнавання образів, визначається взаємне розташування елементів (зверху, знизу, сусідні елементи). Множина геометричних елементів створює словник мови. Множина правил будування створює граматику мови.

3. Зображення представляється як послідовність (ланцюг) символів.

4. Розпізнавання образу полягає у порівнянні вхідного ланцюга образу з еталонним.

В синтаксичних методах розпізнавання виконується **класифікація граматик за Хомським**: контекстно-залежна і контекстно-вільна.

Одним з структурних методів розпізнавання є розпізнавання кривих (контурів зображень) **за Фріменом**, в яких для апроксимації кривих використовують одиничні вектори 1..8 (рис. 2.22).

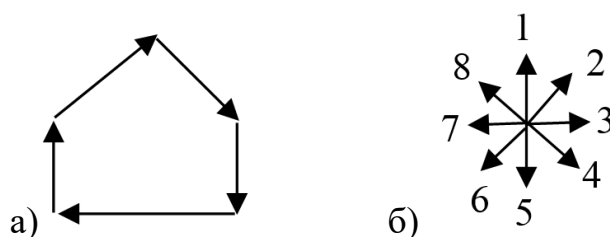


Рис. 2.22. Опис зображення (а) на основі одиничних векторів (б)

Структурний метод розпізнавання передбачає ділення зображення об'єкту на складові частини та встановлення між ними відношень (всередині, вище, нижче, зліва, ...).

Розглянемо квазітопологічний (структурний) метод. Топологія – це наука, яка розглядає властивості графів. Граф складається з точок (вершин) та з'єднуючих їх ребер (ліній). Кожному ребру відповідають дві вершини. Математичний опис графа не залежить від його креслення (рис. 2.23). Тому за допомогою графів можна усунути один з основних недоліків розпізнавання тексту – ускладнення у розпізнаванні деформованого тексту. Пояснюється це тим, що ознаки однакових літер не змінюються при деформації літер.

Хоча ми будемо розглядати не топологічний, а квазітопологічний метод, все ж деякі властивості графів використовуються. При цьому методі встановлюються правила, згідно з якими кожній літері алфавіту призначається якесь число.

Алгоритм опису форми літери:

1. Обхід літери виконується проти руху стрілки годинника по замкненому зовнішньому контуру, починаючи з верхньої правої точки, яка позначається кодом «1».
2. При обході літери позначається цифра коду, яка вказує загальну кількість ребер, що належить вершині графа. На рис. 2.24 вершини графа помічені літерами «а, б, в, г, д».

Згідно з таким алгоритмом кодова комбінація літери «А» має вигляд «1, 3, 1, 3, 3, 1, 3». Цей код отриманий за умови обходу, починаючи з точки «а».

Недоліки квазітопологічного методу: дефекти зображення, переміщення лінії, видалення/ додавання фрагменту утруднюють якісне розпізнавання образу; потрібні додаткові ознаки для розпізнавання, наприклад, літер «І», «Ї», «И», «Й».

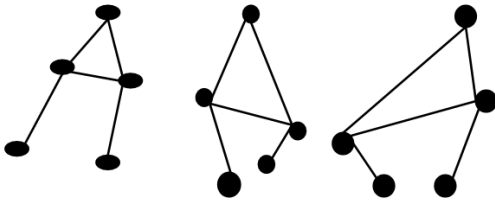


Рис. 2.22. Графи літери «А»

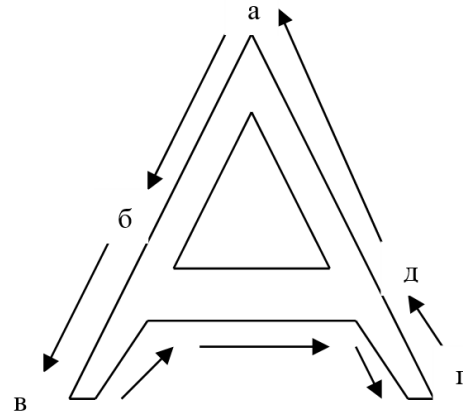


Рис. 2.23. Визначення коду літери «А»

2.6. Розпізнавання у просторі ознак

Об'єкт представляється точкою у просторі, де осями координат є його n ознак [10]. Звичайно використовують кількісні ознаки: діаметр, висота, ширина, площа, периметр (наприклад, геометричні ознаки мікроорганізмів), які утворюють **алфавіт ознак**. Отриманий n -мірний вектор порівнюється з еталонними, й вибирається найближчий з них. Розпізнавання в просторі ознак не відповідає умові цілісності, оскільки при обчисленні ознак втрачається значна частина інформації. Крім того, виникає питання, які саме ознаки обчислювати.

Як міри близькості між об'єктами **a** і **b** використовують зважені евклідові відстані:

$$D(a, b) = \sqrt{\sum_{i=1}^n w_i (a_i - b_i)^2}, \quad (2.4)$$

де a_i, b_i – ознаки об'єктів, w_i – вага ознаки.

2.6.1. Кластерний аналіз. Гіпотеза компактності

Класифікація методом кластерного аналізу базується на **гіпотезі компактності** [10, 15, 16]. Відповідно до цієї гіпотези класу відповідає компактна множина точок (кластер) у деякому класі ознак. Термін „компактний” в даному випадку означає:

1. Число граничних точок мале у порівнянні із загальним числом точок.
2. Будь-які дві внутрішні точки можуть з'єднані пальною лінією так, щоб лінія проходила тільки через точки цієї множини
3. Майже будь-яка внутрішня точка має у достатньо великому околі лише точки цієї множини.

Якщо класи об'єктів не перекриваються, то їх можна розділити лініями на області (2.25).

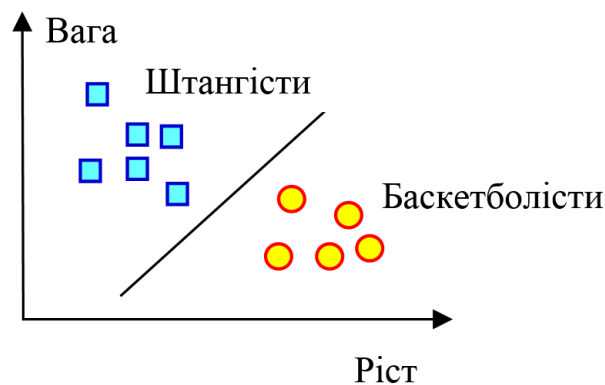


Рис. 2.25. Приклад кластерного аналізу в просторі ознак «ріст», «вага»

Якщо класи частково перекриваються, то вони розділюються приблизно (або потрібно вибрати інші ознаки). За вагою і ростом добре розділяються баскетболісти і штангісти, але погано розділяються футболісти і бігуни. Результатом кластеризації є розбиття об'єктів на групи (кластери або класи).

Формальна постановка задачі кластеризації (або навчання без вчителя) полягає в наступному. Є навчальна вибірка $T = (t_1, \dots, t_M)$ і функція відстані між об'єктами $\rho(t_i, t_k)$, де M – кількість об'єктів. Потрібно розбити вибірку на підмножини, що не перетинаються (кластери), так, щоб кожен кластер складався з об'єктів, близьких по метриці ρ , а об'єкти різних кластерів істотно відрізнялися. При цьому кожному об'єкту t_i приписується мітка (номер) кластера y_j ($j = 1, \dots, N$, N – кількість кластерів)

Алгоритм кластеризації будь-якому об'єкту t_i ставить у відповідність кластер u_j . Кількість кластерів N у деяких випадках відома заздалегідь, проте частіше ставиться задача визначити оптимальне число кластерів, з точки зору того або іншого критерію якості кластеризації.

Вибір відстані $\rho(t_i, t_k)$, між об'єктами t_i та t_k є вузловим моментом дослідження, від нього багато в чому залежить остаточний варіант розбиття об'єктів на класи. Кожен об'єкт t_i описується D ознаками, як позначаються $X_{i,v}$, де $v = 1, \dots, D$. Якщо об'єкти є точками на площині, то кожен об'єкт t_i описується двома ознаками $X_{i,1}$ та $X_{i,2}$ ($D = 2$), які є просторовими декартовими координатами x_1 та x_2 (рис. 2.26).

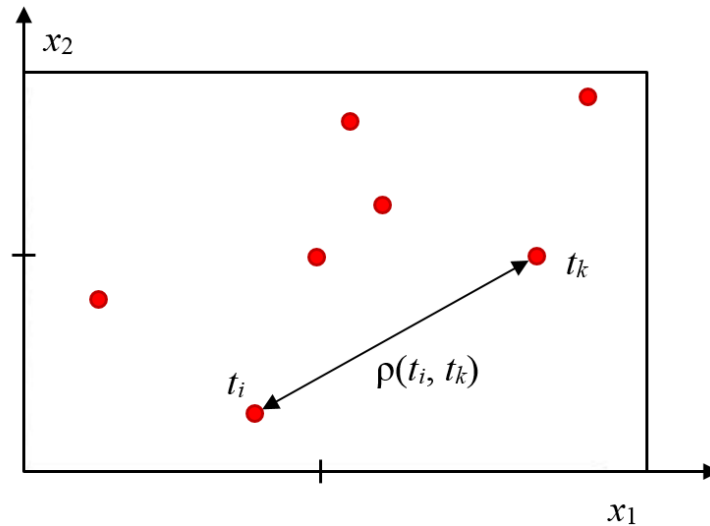


Рис. 2.26. Обчислення відстані між об'єктами t_i та t_k

Існує декілька методів визначення функції відстані ρ :

1. **Відстань Евкліда** є геометричною відстанню в багатовимірному просторі і обчислюється таким чином

$$\rho(t_i, t_k) = \sqrt{\sum_{v=1}^D (X_{i,v} - X_{k,v})^2}. \quad (2.5)$$

Відстань Евкліда як критерій має певні переваги (наприклад, відстань між двома об'єктами не змінюється при введенні в аналіз нового об'єкту, який може виявитися викидом).

2. **Відстань міських кварталів** (манхетенська відстань), яка є середньою різницею по координатах. В більшості випадків ця міра відстані приводить до таких же результатів, як і відстань Евкліда. Проте відзначимо, що для цієї міри вплив окремих великих різниць

(викидів) зменшується (оскільки вони не зводяться в квадрат). Манхетенська відстань обчислюється за формулою

$$\rho(t_i, t_k) = \sqrt{\sum_{v=1}^D |X_{i,v} - X_{k,v}|}. \quad (2.6)$$

3. Відстань Чебишева. Ця відстань може виявитися корисною, коли бажають визначити два об'єкти як «різні», якщо вони розрізняються по якій-небудь одній координаті (яким-небудь одним виміром). Відстань Чебишева обчислюється за формулою

$$\rho(t_i, t_k) = \max(X_{i,v} - X_{k,v}), \quad (2.7)$$

де $v = 1, \dots, D$.

При кластеризації середня внутрішньокластерна відстань має бути якомога менше, а середня міжкластерна відстань має бути якомога більше.

Кластеризація методом k -середніх (англ. k -means clustering) – популярний метод кластеризації, – полягає у впорядкуванні множини об'єктів у порівняно однорідні групи. Мета методу – розділити n спостережень на k кластерів, так щоб кожне спостереження належало до кластера з найближчим до нього середнім значенням.

2.6.2. Метод потенціалів

Досліджувані об'єкти, які потрібно розпізнати, описуються набором електричних зарядів [10]. Потенціал точкового електричного заряду описується формулою

$$P = \frac{k \cdot q}{R^2}, \quad (2.8)$$

де k – постійний коефіцієнт, q – величина заряду, R – відстань від точки до заряду.

Якщо потенціал створюється кількома зарядами, то потенціал у будь-якій точці дорівнює сумі потенціалів всіх зарядів (рис. 2.27); при цьому заряди можуть бути різних знаків – додатні і від'ємні.

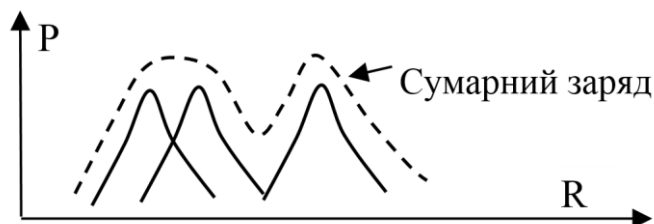


Рис. 2.27. Суперпозиція зарядів

Оскільки в попередній формулі (2.8) для потенціалу є ділення на нуль при $R = 0$, тому на практиці використовується формула

$$P = \frac{q}{1 + \alpha R^2}, \quad (2.9)$$

де коефіцієнт α визначає швидкість зміни потенціалу.

Для двомірного випадку поверхня потенціалу може мати вигляд гірського ландшафту з гірськими хребтами і долинами.

В просторі ознак точки різних класів A і B створюють потенціал певного знаку, клас нової точки C визначається тим, який потенціал в точці C буде більший (рис. 2.28).

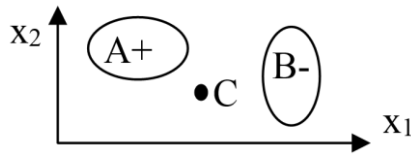


Рис. 2.28. Суперпозиція зарядів A та B і просторі ознак x_1 та x_2

Таким чином, метод потенціалів дозволяє розпізнавати об'єкти довільної кількості класів.

2.6.3. Байєсівський метод розпізнавання

Байєсівський метод розпізнавання належить до імовірнісних. Математично байєсівський метод описується так [10]. Нехай існують класи $K_1 \dots K_m$, які описується вектором ознак Y . З кожним класом пов'язана апіорна ймовірність $P(K_j)$ появи об'єкта j -го класу (частота появи). Умовна ймовірність того, що класу K_i відповідають ознаки Y позначається $P(Y/K_i)$. Тоді можна визначити ймовірність того, що вектору ознак Y відповідає клас K_i .

$$p(K_i / Y) = \frac{p(K_i)p(Y / K_i)}{p(Y)}, \quad (2.10)$$

де спільну для всіх класів величину $p(Y)$ можна скоротити.

Розглянемо приклад використання байєсівського методу, коли потрібно відрізнити стиглі яблука від недостиглих (використовується 2 класи: K_1 = „Стигли” і K_2 = „Недостигли”), $m=2$. Вектор ознак Y =(розмір /малий, великий /, колір /зелений, червоний/). Ймовірність попадання стиглих яблук більша: $p(K_1)=0,7$; $p(K_2)=0,3$.

Якщо яблука стиглі (K_1):

при Y =(великий, червоний) значення $p(Y, K_1)=0,95$;

при Y =(малий, зелений) значення $p(Y, K_1)=0,05$.

Якщо яблука не стиглі (K_2):

при $Y=(\text{великий, червоний})$ значення $p(Y, K_2)=0,05$;

при $Y=(\text{малий, зелений})$ значення $p(Y, K_2)=0,95$.

Якщо спостерігаються ознаки $Y=(\text{великий, червоний})$, то

$$p(K_1/Y)=0,7 \cdot 0,95 = 0,665;$$

$$p(K_2/Y)=0,3 \cdot 0,05 = 0,015.$$

Таким чином, за допомогою байєсівського методу можливо розділити об'єкти різних класів.

2.6.4. Метод опорного словника

Метод опорного словника належить до методів розпізнавання у просторі ознак і містить каталог, у якому знаходяться всі можливі реалізації, які можуть зустрітись при розв'язанні конкретної задачі [10]. Припустимо, що розпізнається слово «ЛИСТ» з 4-х літер. Спочатку виділяється характерна сукупність ознак кожного символу цього слова:

$$X_L=\{x_1, x_2, \dots, x_p\}_L;$$

$$X_H=\{x_1, x_2, \dots, x_p\}_H;$$

$$X_C=\{x_1, x_2, \dots, x_p\}_C;$$

$$X_T=\{x_1, x_2, \dots, x_p\}_T.$$

Потім використовують вирішальні функції, які розроблені для кожного символу українського алфавіту з 33 літер. Вирішальні функції:

$$g_A(X_{1S}), g_B(X_{2S}), \dots, g_Y(X_{32S})$$

приймають максимальне цифрове значення, коли сукупність ознак X_{jS} відноситься до літери, для якої складена конкретна вирішальна функція. Наприклад, $g_A(X_A)$ приймає максимальне значення, а $g_A(X_L)$ приймає мінімальне значення.

Якщо в слові є чотири літери, то система розпізнавання перебирає всі слова з 4-х літер, які є у словнику. Для кожного з таких слів обчислюється сума з 4-х вирішальних функцій (вважаємо, що вирішальні функції літер, з яких складаються слова, відомі для кожного слова). Для всіх цих слів, які є у словнику, отримуються вирішальні функції у вигляді:

$$S_1(АБАТ)=g_A(X_L)+ g_B(X_H)+ g_A(X_C)+ g_T(X_T);$$

.....

$$S_{10}(ЛИСТ)=g_L(X_L)+ g_H(X_H)+ g_C(X_C)+ g_T(X_T);$$

.....

$$S_{50}(ЯЩУР)=g_Я(X_L)+ g_Щ(X_H)+ g_У(X_C)+ g_Р(X_T).$$

В результаті виділяється те слово в словнику, для якого отримана сума має найбільше значення (наприклад, ЛИСТ).

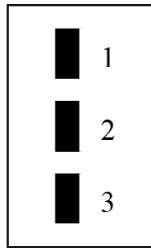
Недоліком розглянутого алгоритму є швидке зростання об'єму розрахунків із зростанням кількості літер, слів та ознак. У результаті словник збільшується і зростає час класифікації об'єкта. Для прискорення роботи системи РО в даному випадку можуть використовуватись вирішальні функції, які враховують статистичні дані по частоті використання слів.

2.6.5. Метод зондів

Метод зондів (розпізнавання у просторі ознак) використовується для розпізнавання цифр та літер, навіть написаних від руки та з деякими відхиленнями у розмірах та стилі написання [10]. Вперше метод зондів був запропонований британським вченим Д.Даймондом у 1958р. і використовувався для обробки чеків на сплату міжнародних телефонних розмов.

Розглянемо принцип використання зондів у припущенні, що наша система РО повинна розпізнавати образи лише чотирьох великих друкованих літер «ЛИСТ», написаних від руки. Можна вважати, що кожний зонд 1, 2, 3 (рис. 2.29) є активним (значення «1»), якщо він перекриває зображення об'єкта, а в іншому випадку – пасивний (значення «0»).

Сигнали від всіх зондів передаються на дешифратор, який і виконує функції класифікації літери (табл. 2.4). Кожному зображенню одного класу (одній літері) відповідає одна комбінація активних зондів. В дійсності зонди для розпізнавання літер всього алфавіту мають складніший вигляд у порівнянні з показаними на рис. 2.29. Пояснюється це тим, що розглядається більша кількість літер і одночасно враховується можливе відхилення форми літери від еталону. Можуть також використовуватись доріжки-зонди, які окреслюють літеру з боків і центрують зонди для точного визначення місця знаходження однозначно (зонди накладаються на центр літери по горизонталі).



ЛИСТ

Рис. 2.29. Використання зондів для літер «ЛИСТ»

Таблиця 2.4

Коди зондів для літер «ЛИСТ»

Літери	Зонди		
	1	2	3
Л	1	0	0
И	0	1	0
С	1	0	1
Т	1	1	1

Існує модифікація методу зондів, яка дозволяє розпізнавати зображення з такими можливими деформаціями: стиснення та розтягування по осях X або Y ; локальні спотворення контуру об'єкта у визначених межах; нахил або поворот об'єкта на деякий кут.

Контрольні запитання до розділу 1

1. Охарактеризуйте поняття фатичного діалогу.
2. У чому полягає тест Тьюринга?
3. Яким чином можна запрограмувати фатичний діалог?
4. Які можливі способи порівнянь речення, що вводиться при фатичному діалозі, із шаблонами?
5. Як використовуються замінювачі „*” і „?” у програмах, що реалізують фатичний діалог?
6. Опишіть послідовність програмної обробки запитання користувача під час фатичного діалогу.

3. МОДЕЛІ І МЕТОДИ ПРИЙНЯТТЯ РІШЕНЬ

3.1. Планування цілеспрямованих дій і прийняття рішень

Оптимізаційні задачі

Інтелектуальна система спочатку **аналізує зовнішню ситуацію**, потім планує **цілеспрямовані дії і приймає рішення** про вибір певної дії [10]. Широкий клас завдань у рамках прийняття рішення можна сформулювати у вигляді класичної **оптимізаційної задачі**: знайти рішення, за яким цільова функція досягає максимуму при заданих обмеження. Оптимізаційні задачі є предметом науки „дослідження операцій”.

Формально оптимізаційна задача описується у вигляді: знайти рішення $x=(x_1, \dots, x_n)$, для якого цільова функція $f(x)$ досягає максимуму (або мінімуму) і задовольняються обмеження $g_i(x) \geq 0$.

Приклад. Мандрівник збирається у дорогу. Він може взяти в **рюкзак** певну кількість x_i предметів різних типів. Нехай є n типів предметів, наявна кількість i -го предмета становить r_i . Кожний предмет має власну цінність c_i та вагу q_i . Потрібно зібрати рюкзак таким чином, щоб сумарна цінність узятих предметів була максимальною, але щоб сумарна вага не перевищувала межі u .

Математично **задача про рюкзак** формулюється так: знайти

$x=(x_1, \dots, x_n)$, для якого $\sum_{i=1}^n c_i x_i \rightarrow \max$ та задовольняються обмеження

$$\sum_{i=1}^n q_i x_i \leq u, \text{ де } x_i - \text{цілі числа, } x_i \geq 0; x_i \leq r_i.$$

Будь-який елемент x , що задовольняє обмеженням $g_i(x) \geq 0$, називається **допустимим рішенням** задачі. Якщо умова максимізації не висувається, то йдеться про **задачу пошуку допустимих рішень**. Якщо обмежень немає, то йдеться про **безумовну оптимізацію**.

Якщо потрібно вирішити задачу **мінімізації**, то вона переводиться до задачі **максимізації** зміною знаку в цільовій функції.

Залежно від вигляду цільової функції та обмежень розрізняють:

- 1) лінійне програмування – цільова функція і обмеження лінійні;
- 2) нелінійне програмування - цільова функція і обмеження нелінійні;
- 3) дискретне програмування – всі x_i є цілими числами (як у задачі про рюкзак).

Розглянемо основні методи вирішення оптимізаційних задач.

Метод повного перебору

Метод **повного перебору** – це очевидний і універсальний метод вирішення оптимізаційних задач, якщо множина допустимих рішень **М обмежена**. Метод є точним, тобто гарантує **оптимальний розв’язок**.

Недолік перебору – для практичних задач кількість варіантів надто велика.

Іноді можна пожертвувати точністю розв’язку і знайти **субоптимальне** рішення за рахунок значного зменшення часу **розрахунку**. В цьому полягає суть евристичних алгоритмів.

Евристичний пошук

Евристичний пошук – процедура **систематизованого перебору** варіантів. Загальна схема методу така:

1. Вибрати певну дію з області можливих дій
2. Здійснити дію, що приведе до зміни ситуації.
3. Оцінити нову ситуацію.
4. Якщо досягнуто успіху – кінець, інакше повернутися на крок 1.

Здійснити дію можна реально (стратегія спроб і помилок) або уявно (планування). Одним з важливих варіантів евристичного пошуку є метод послідовних поліпшень.

3.2. Складність вирішення оптимізаційних задач

Розглянемо складність рішення оптимізаційних задач на прикладі задачі про **виконуваність булевого виразу**, яка формулюється так: для будь-якого виразу від n змінних знайти хоча б один набір значень $(x_1, \dots, x_n)$, для якого вираз приймає значення 1 [10]. Найпростіша схема евристичного пошуку така: надаємо одне з можливих значень (0/1) першій, другій і т.д. змінним. Після встановлення значень всіх змінних перевіряється значення виразу: 1 – задача вирішена, не 1 – повернутися на початок.

Таку задачу можна розглядати як задачу пошуку на певному **дереві перебору**. Кожна вершина цього дерева відповідає певному набору встановлених значень $x_1, \dots, x_k$. Ми можемо встановити змінну x_{k+1} в 0 або в 1, тобто маємо вибір з двох дій. Тоді кожній з цих дій відповідає одна з двох дуг, які йдуть від цієї вершини. Тобто вершина $(x_1, \dots, x_k)$ має двох нащадків $(x_1, \dots, x_k, 0)$ і $(x_1, \dots, x_k, 1)$.

При $n=3$ дерево перебору /можливостей/ матиме вигляд (рис. 3.1).

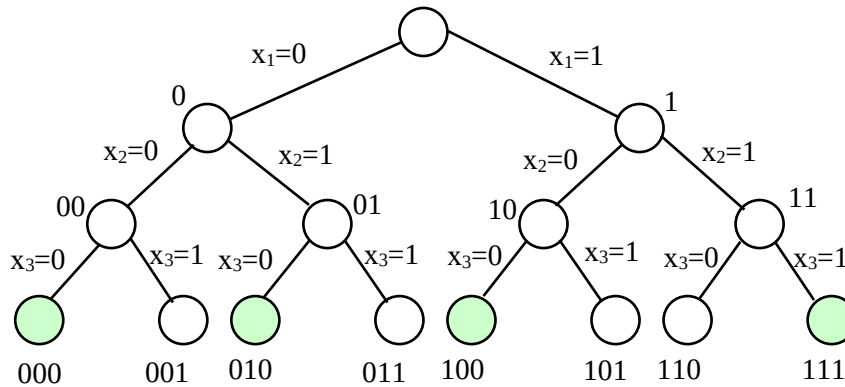


Рис. 3.1. Дерево перебору (вершини – значення змінних, дуги – рішення), зафарбовані вершини відповідають виразу $(x_1x_2) = x_3$

З рис. 3.1 видно, що вирішення задачі зводиться до перебору листів (гілок) дерева з пошуком набору, що відповідає умові задачі. Для виразу $(x_1x_2) = x_3$ такими наборами будуть 000, 010, 100, 111.

Якщо $n=3$, дерево має $2^3=8$ листів. Тобто число варіантів експоненційно залежить від n : $2^n = \exp(n \ln 2)$.

Не кожний перебірний алгоритм є **експоненційним** (алгоритм пошуку максимального елемента в масиві – лінійний).

Із збільшенням розмірності задачі будь-який **поліноміальний** алгоритм стає ефективнішим за **експоненційний**.

3.2.1. Пошук у глибину і у ширину

Є дві основні стратегії пошуку потрібної вершини на дереві [10]:

- 1) пошук у **ширину**;
- 2) пошук у **глибину** (перебір з поверненнями, **бектрекінг**).

Процедура пошуку у **ширину** передбачає аналіз на кожному кроці нащадків усіх вершин (**паралельна** перевірка всіх можливих альтернатив).

Процедура пошуку в **глибину** передбачає першочерговий аналіз нащадків тих вершин, що були проаналізовані останніми. Всі альтернативи аналізуються **послідовно**; аналіз деякої альтернативи завершується лише тоді, коли встановлюється, приводить вона до успіху чи ні. Якщо ж альтернатива зумовлює невдачу, відбувається повернення і розгляд інших альтернатив.

На рис. 3.2 показана послідовність перегляду вершин при пошуку в ширину, на рис. 3.3 – пошуку в глибину.

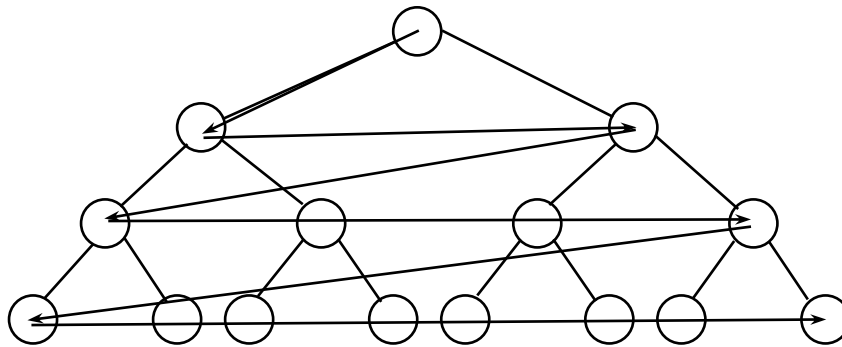


Рис. 3.2. Схема пошуку в ширину

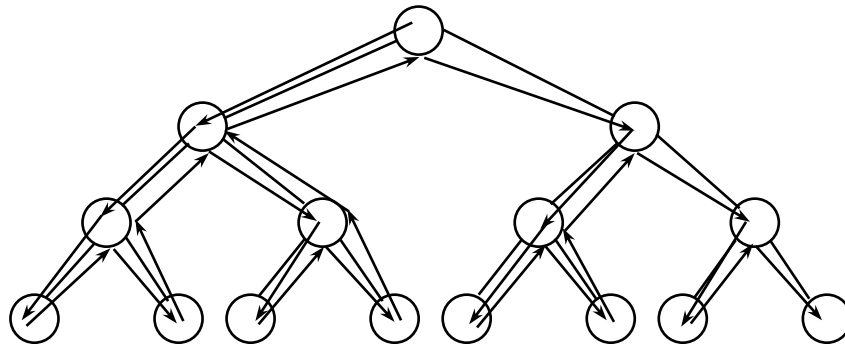


Рис. 3.3. Схема пошуку в глибину

Пошук в **глибину** (більш поширений) дозволяє зекономити **пам'ять**, оскільки при його реалізації не потрібно запам'ятовувати все дерево, а лише вершини, що мають відношення до поточної альтернативи.

3.2.2. Автоматний спосіб задання алгоритму

В комп'ютері вхідна послідовність бітів перетворюється у вихідну за допомогою **логічних схем** [10]. Логічні схеми поділяються на **комбінаційні схеми** і схеми з **пам'яттю (скінченні автомати)**.

У **комбінаційних схемах** значення вихідних змінних залежать лише від стану вхідних і не залежать від поточного стану схеми. Будь-яка комбінаційна схема реалізує булеву (0/1) функцію від своїх вхідних змінних.

Скінчений автомат є перетворювачем, вихід якого залежить від входу, але й від поточного стану автомата. При цьому кількість

вхідних і вихідних змінних, їх значень є скінченою. Поточний стан автомата – в пам'яті.

Найуніверсальнішою моделлю комп'ютерних обчислень є машина **Тьюринга**.

Автомати мають пам'ять у вигляді стрічки і пристрій для читання з неї. Стрічка розбита на квадрати з символами. Автомат розглядає символи по черзі, виконує обчислення і розв'язує задачу.

Формально скінчений автомат описується так:

$$FA = \langle Q, A, b, q_0, F \rangle$$

де $Q = (q_1, q_2, \dots, q_n)$ - скінченна множина станів керування; $A = (a_1, a_2, \dots, a_m)$ - вхідний алфавіт; $b: Q^*A \rightarrow Q$ - функція переходів; q_0 - початковий стан; $F \subseteq U$ - множина заключних станів керування.

Приклад. Потрібно побудувати скінчений автомат який допускає тільки послідовності 0/1, при цьому в послідовності розпізнавання кількість **одиниць** повинна бути **парною**, а **нулів** – **непарною**.

Згідно з умовою задачі A складатиметься з символів 0, 1, #, де # позначатиме довільний символ, відмінний від 0 і 1. Тобто, $A = \langle 0, 1, \# \rangle$.

Множину станів виберемо так: $Q = (q_1, q_2, \dots, q_5)$,

де q_0 - початковий стан;

q_1 - стан помилки;

q_2 - кількість **нулів непарна**, а **одиниць** – **парна**;

q_3 – кількість нулів і одиниць непарна;

q_4 – кількість нулів і одиниць парна;

q_5 – кількість нулів парна, а одиниць – непарна;

Згідно з умовою задачі $F = \langle q_2 \rangle$

Функції переходів визначаються так:

$q_0\# \rightarrow q_1$	$q_00 \rightarrow q_2$	$q_30 \rightarrow q_5$
$q_1 \rightarrow q_1$	$q_01 \rightarrow q_5$	$q_31 \rightarrow q_2$
$q_2 \rightarrow q_1$	$q_10 \rightarrow q_1$	$q_40 \rightarrow q_2$
$q_3 \rightarrow q_1$	$q_11 \rightarrow q_1$	$q_41 \rightarrow q_5$
$q_4 \rightarrow q_1$	$q_20 \rightarrow q_4$	$q_50 \rightarrow q_3$
$q_5 \rightarrow q_1$	$q_21 \rightarrow q_3$	$q_51 \rightarrow q_4$

Автомати зручно задавати таблично (рис. 3.4):

	#	0	1	←
q ₀	q ₁	q ₂	q ₅	
q ₁	q ₁	q ₁	q ₁	
q ₂	q ₁	q ₄	q ₃	
q ₃	q ₁	q ₅	q ₂	
q ₄	q ₁	q ₂	q ₅	
q ₅	q ₁	q ₃	q ₄	

Рис. 3.4. Табличне задання автомата

Рядки таблиці позначаються станами множини Q , а стовпці – символами вхідного алфавіту. Символ $\leftarrow$ позначає заключні допустимі стани.

Графічний опис автомата полягає у побудові графа (рис. 3.5), де вершини q_i і q_j з'єднуються дугою a якщо виконується команда $q_i a \rightarrow q_j$. Заключні допустимі вершини позначаються подвійним колом.

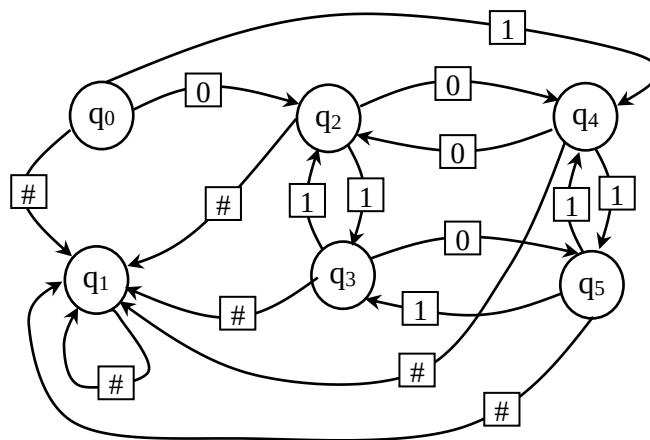


Рис. 3.5. Графічне задання автомата

Спеціальним типом **автомата** є **машина Тьюринга**. Універсальна машина Тьюринга може обчислити будь-який інтуїтивний алгоритм.

Універсальним обчислювальним пристроєм називатимемо пристрій, на якому можна промодельовати роботу машини Тьюринга.

Машини Тьюринга дозволяють обчислювати **тільки рекурсивні** функції. **Частково** рекурсивні функції – визначені не при всіх значеннях аргумента.

Теза Тьюринга: Клас функцій, частково обчислюваних за Тьюрингом, збігається з класом частково рекурсивних функцій.

Машина Тьюринга дуже незручна для програмування через послідовний доступ до комірок пам'яті (стрічка), неструктурованість записів на стрічці (потрібно використовувати розподільники), відсутні основні арифметичні операції та ін.

3.3. Складність алгоритмів

Під складністю алгоритму розуміється часова оцінка його роботи або ємність необхідної пам'яті [10].

Апріорний аналіз алгоритму виконується теоретично, а **тестування** передбачає практичну перевірку.

Розмірність задачі (вхідних даних) – це число (ціле), яке характеризує розмір вхідних даних задачі (розмірність одновимірних, двовимірних масивів, ...).

Часова складність алгоритму – час вирішення проблеми у залежності від розмірності задачі.

Асимптотична часова складність – поведінка часової складності при збільшенні розмірності задачі.

Задачі з великою часовою складністю неможливо вирішити за допомогою сучасних комп'ютерів за допустимий час.

При аналізі алгоритмів використовують такі позначення. Запис $f(n)=O(g(n))$ означає: функція має порядок $g(n)$ тоді, існують дві додатні константи c і n_0 , що $f(n) \leq c[g(n)]$ для всіх $n > n_0$.

Звичайно $f(n)=O(g(n))$ визначає час обчислень; $\lim_{n \rightarrow \infty} \frac{f(n)}{g(n)} \rightarrow 1$.

Теорема. Якщо $A(n)=a_m n^m + \dots + a_1 n + a_0$ є поліномом ступеня m , тоді $A(n)=O(n^m)$.

Нехай є два алгоритми з часовою оцінкою обчислень $O(n)$ і $O(n^2)$. Час виконання першого алгоритму – $c_1 n$, а другого – $c_2 n^2$ (c – константи). Тоді при $n < c_1/c_2$ перевагу має перший алгоритм, а при $n > c_1/c_2$ – другий.

При великих n значення констант несуттєві.

Найбільш типові часові оцінки алгоритмів: $O(1)$ - константна, $O(\log n)$ - логарифмічна, $O(n)$ - лінійна, $O(n^m)$ - поліноміальна, $O(2^n)$ - експоненційна. Для досить великих n :

$$O(1) < O(\log n) < O(n) < O(n \log n) < O(n^2) < O(n^3) < \underline{O(2^n)}$$

$\Omega(g(n))$ – мінімальна часова оцінка.

Задачі класу P і NP

Якщо розмірність задачі n , то **P** задача може бути розв'язана за поліноміальний час $O(n^m)$.

Задача **NP** може бути вирішена за поліноміальний час для **недетермінованої** машини Тьюринга (така машина породжує на кожному кроці певну кількість нових машин).

Більшість дискретних і комбінаторних задач можна вирішити шляхом перебору. Проте перебірні методи мають експоненційну складність.

Існує проблема – знайти метод вирішення експоненційних задач за поліноміальний час.

3.4. Планування в просторі станів

3.4.1. Основні поняття теорії графів

При **плануванні в просторі станів** заданим вважається деякий набір станів (ситуацій). Відомі дії, які може здійснювати система та які визначають перехід з одного стану в інший [10].

Графом станів задачі називається орієнтований граф, вершини якого відповідають можливим станам предметної області, а дуги – методам переходу від стану до стану.

Дуги можуть мати мітки, які інтерпретуються як вартість або довжина відповідного переходу. Тоді вирішення задачі являє собою пошук шляху від початкового стану до цільового; при цьому типовою вимогою є оптимізація даного рішення (пошуку найкоротшого шляху).

Класичний приклад планування в просторі **станів** – задача **комівояжера**.

Граф – сукупність вершин і дуг. Для комп'ютерного опису графу використовуються: матриця інцидентності, матриця суміжності та списки суміжності.

Поширена структура даних в програмуванні – лінійні списки.

Лінійний список (рис. 3.6) – скінченна послідовність однотипних елементів (певної довжини). Списки бувають однозв'язні (лінійні), двозв'язні (циклічні) і т.п. Лінійний список **L**, що складається з $l_1, l_2, \dots, l_n$ елементів, записується у вигляді $L = \langle l_1, l_2, \dots, l_n \rangle$

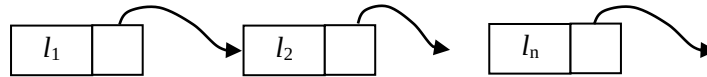


Рис. 3.6. Графічне зображення списку

Існує два основних методи задання списків: **послідовний** (масиви) і **зв'язний** (динамічні змінні).

Стек – список, в якому занесення і видалення елементів можна проводити тільки через один початковий елемент (вершину стеку). Стек працює за принципом: останній зайшов – перший вийшов (Last In First Out - LIFO)

Чергою називають список, в якому всі вставки здійснюються в кінець списку (змінна *rear*), а всі видалення відбуваються з голови (початку) списку (змінна *front*). Черга працює за принципом: перший зайшов – перший вийшов (First In First Out - FIFO).

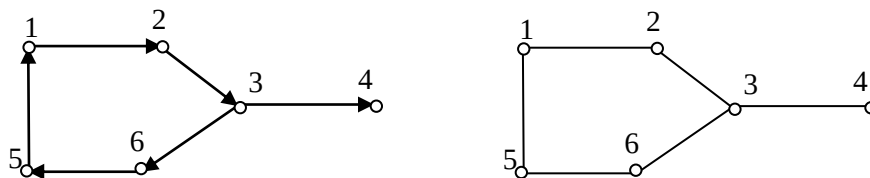
Поняття граф ввів у 1736 році Ейлер. Граф **G** містить дві множини $G = (V, E)$, де **V** - множина вершин (вузлів $1 \dots n$), **E** - множина ребер. Коли пари вершин **впорядковані** – то граф **орієнтований**, інакше – неорієнтований (рис. 3.7). Впорядкована пара позначається $\langle i, j \rangle$, неупорядкована – (i, j) . У **зважених** графах кожне ребро має вагу.

Ступенем вершини називається число суміжних вершин. В орієнтованому графі виділяють **півступінь входу** (число вхідних ребер) та **півступінь виходу** (число вихідних ребер).

Шляхом називається послідовність вершин між двома вершинами **p** і **q**.

Циклом називається шлях, у якого початкова і кінцева вершини збігаються.

Граф називається **зв'язним**, якщо для довільної пари вершин існує між ними шлях.



а)

б)

Рис. 3.7. Зображення орієнтованого (а) і неорієнтованого (б) графів.

3.4.2. Способи задання графів

Існує **послідовний і зв'язний** спосіб задання графів.

Послідовна форма використовує квадратну таблицю ($\text{Graf}(n,n)$, n – вершин), яку називають **матрицею суміжності** (рис. 3.8). Якщо є зв'язок між вершинами, то $\text{Graf}(I,j)=1$, інакше $\text{Graf}(I,j)=0$.

	1	2	3	4	5	6
а) 1	0	1	0	0	1	0
2	1	0	1	0	0	0
3	0	1	0	1	0	1
4	0	0	1	0	0	0
5	1	0	0	0	0	1
6	0	0	1	0	1	0

	1	2	3	4	5	6
б) 1	0	1	0	0	0	0
2	0	0	1	0	0	0
3	0	0	0	1	0	1
4	0	0	0	0	0	0
5	1	0	0	0	0	0
6	0	0	0	0	1	0

Рис. 3.8. Матриці суміжності графів з рис. 3.7:

а) – неорієнтованого, б) – орієнтованого

Матриця **інцидентності** відображає зв'язок n вершин за допомогою m дуг, розмір матриці інцидентності $(n*m)$, але вона менш зручна, ніж матриця суміжності.

Інша форма задання графу – **список зв'язності**. Граф з n вершинами буде задаватися списками – по одному для кожної вершини. Список для вершини i містить вершини, суміжні з i .

3.4.3. Дерева

Дерево – це орієнтований ациклічний граф, для якого виконуються такі умови:

- 1) існує одна вершина, в яку не входить жодне ребро (корінь);
- 2) у будь-яку вершину, крім кореня, входить лише одне ребро;
- 3) з кореня можна знайти унікальний шлях до кожної вершини дерева.

Орієнтований граф з кількох вершин – **ліс**. **Бінарне дерево** – кожна вершина має два нащадки. При **пошуку** певного вузла у дереві використовують пряме і зворотне проходження вузлів.

Розрізняють такі способи зберігання дерев: **послідовне** і **зв'язне**. **Послідовне** зберігання ділиться на рівневі і діжкове:

- **Рівневе** – для кожного рівня дерева послідовно вказуються його вузли.
- **Дужкове** – дужками вказуються нащадки даного вузла.

Зв'язне зберігання – у вигляді списків, кожен список містить вказівними на нащадків.

В деревах використовується пошук у **глибину** і **ширину**. Пошук в **глибину** – послідовно відвідуються всі **нові** вершини –нащадки (якщо вершина була відвідана, то вона перестає бути новою). Пошук у **ширину** – перевіряються суміжні вершини одного рівня, потім – перехід на нижчий рівень.

Довільне дерево $\langle V, T \rangle$ неорієнтованого графу $G = \langle V, E \rangle$ називається **остовим**.

Ейлерові шляхи

Для вирішення багатьох прикладних проблем використовуються **ейлерові шляхи**. Ейлеревими називають шляхи, які проходять через кожне ребро графу **один** раз. Якщо початковий вузол є кінцевим, то такий граф називається ейлеревим циклом.

Задача про Кенігсбергські мости.

Теорема. У графі існує ейлерів шлях тільки тоді, коли граф зв'язний і містить не більше двох вершин непарного ступеня.

Знаходження найкоротших шляхів у графі

Потрібно знайти мінімальний шлях (контур) у навантаженому графі, де $d(u, v)$ - відстань між вершинами u і v , якщо не існує шляху, то $d(u, v) = \infty$.

3.5. Загальна схема алгоритму Дейкстри

Алгоритм **Дейкстри** [10] призначений для пошуку найкоротшого шляху у графі (рис. 3.9). Введемо такі позначення:

s - початкова вершина;

q - цільова вершина;

c(i, j) - довжина ребра між вершинами **i** та **j**;

g(n) - довжина найкоротшого шляху між від **початкової** вершини до **n-ї**.

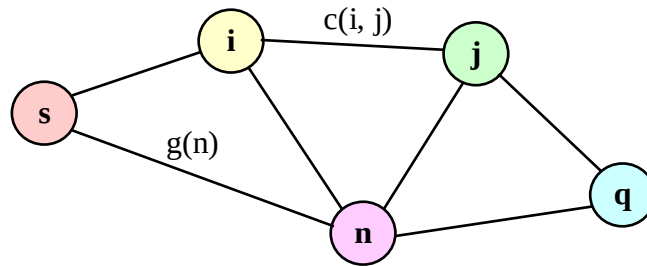


Рис. 3.9. Приклад графу

Загальна схема пошуку найкоротшого шляху:

1. Сформувати граф пошуку G , який спочатку складається з початкової вершини s . Прокласти $g(s)=0$.
2. Для вершин i графу G знайти суміжні n вершин (крім початкової), додати їх до графу G . Для кожної вершини n обчислити довжину шляху $g(n)=g(i)+c(i, n)$ для всіх вершин i , вибрати мінімальне $g(n)$ та зберегти відповідний йому шлях. Якщо суміжні вершини n відсутні і цільова вершина q не досягнута, то вихід – потрібного шляху не існує.
3. Якщо граф G містить всі вершини початкового графу, то знайдений шлях до цільової вершини q буде найкоротшим; завершити алгоритм.
4. Повернутися на крок 2.

3.6. Планування в просторі задач

І-АБО граф

Планування в просторі задач полягає в розбитті задачі на **підзадачі**, поки під задача не зведеться до **елементарної** (для якої існує готовий алгоритм розв'язку).

Для планування в просторі задач традиційно використовують **І-АБО графи**.

І-АБО графом називають **орієнтований** граф, вершини якого відповідають **задачам**, а дуги – **відношенням** між задачами (І - АБО).

І-АБО граfi фактично є фрагментами мереж виведення продукційних систем. Розглянемо продукційну систему

$A \rightarrow C$

$B \rightarrow C$

$D, C \rightarrow G$

Для такої системи існує І-АБО граф (рис. 3.10)

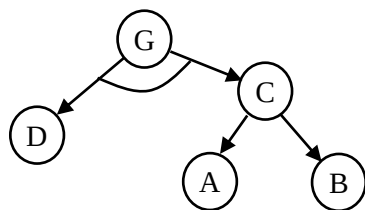


Рис. 3.10. І-АБО граф

Вершина **C** пов'язана з вершинами **A** і **B** відношенням **АБО** (достатньо виконання однієї підзадачі), а вершина **G** пов'язана з вершинами **D** і **C** відношенням **I** (необхідне виконання всіх підзадач).

Метод „Поділяй і пануй”

Нехай потрібно вирішити задачу для елементів масиву початкових даних від **p** до **q**.

Метод „поділяй і пануй”, або поділ на підзадачі, можна записати у вигляді рекурсивної процедури **SubTask**.

Булева функція **Small(p, q)** визначає, чи не зведена під задача до елементарної. Якщо задача елементарна, то знаходиться її розв'язок за допомогою функції **G(p,q)**.

Якщо задача не елементарна, то вона ділиться на дві частини функцією **Divide(p,q)**. Процедура **Combine** рекурсивно викликає на виконання процедуру **SubTask**, але для меншої розмірності вхідних даних. Процес рекурсивно продовжується до тих пір, поки всі задачі не будуть зведені до елементарних.

```

procedure SubTask(p, q:integer);
  Var m:integer;
Begin
  if Small(p, q) then G(p,q);
  else
  begin
    m:=Divide(p,q); { p <= m < q }
    Combine (SubTask(p,m), SubTask(m+1,q));
  end;
End;
  
```

3.7. Жадібні алгоритми

Види жадібних алгоритмів

Жадібні алгоритми відносяться до оптимізаційних і полягають в послідовному локальному поліпшенні розв'язку задачі. Тобто на кожному кроці цільова функція не повинна зменшуватися [10].

Принцип дії жадібних алгоритмів полягає у послідовному переборі варіантів для кожного елементу розв'язку задачі. З усіх локальних варіантів вибирається найкращий і відбувається перехід до наступного вузла. Після обходу всіх вузлів відбувається перехід до наступного кроку.

3.7.1. Градієнтний метод

Одним з прикладів жадібних алгоритмів є градієнтний метод. Він використовується, якщо допустима множина розв'язку $D = R^n$ (n - розмірний евклідовий простір), а цільова функція диференційована. Градієнтом функції у точці $x = (x_1 \dots x_n)$ називається вектор

$$\nabla f(x) = \left(\frac{\partial f(x)}{\partial x_1}, \dots, \frac{\partial f(x)}{\partial x_n} \right). \quad (3.1)$$

Градієнт задає напрямок найшвидшого зростання функції. Тоді у градієнтному методі $x^{(k+1)} = x^{(k)} - \lambda \nabla f(x^{(k)})$, де k – номер ітерації, λ – величина кроку в напрямі антиградієнта.

Аналогія градієнтного методу – знайти найвищу точку на горбистій місцевості; при цьому досягається **локальний**, а не глобальний екстремум.

3.7.2. Алгоритм Пріма

Для графа $G = \langle V, E \rangle$ задані ваги його ребер. Остовим деревом графу G є неорієнтоване дерево $T = \langle V, E' \rangle$, яке містить всі вершини графу. Вартість остового дерева визначається як вартість його ребер.

Алгоритм Пріма полягає в знаходженні остового дерева мінімальної вартості. Остове дерево будують крок за кроком, додаючи до нього по одному ребру за певним оптимізаційним критерієм.

Побудова дерева починається з ребер, що мають мінімальну вартість. На черговому кроці вибирається ребро, що приводить до мінімального збільшення суми вартостей ребер дерева й не приводить

до появи циклу. Процес побудови закінчується, якщо дерево **T** матиме всі вершини графу.

Приклад використання алгоритму Пріма для графу **G** (рис. 3.11, табл. 3.1):

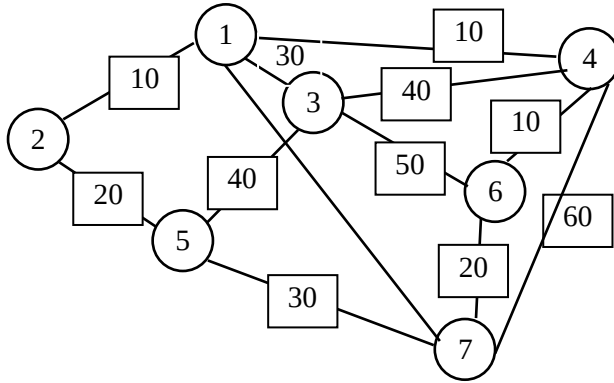


Рис. 3.11. Приклад зваженого графу

Таблиця 3.1

Послідовність побудови остового дерева

Канди-дати	Ребра	Вар-тість	Остове дерево
(1, 2)	(1, 2)	10	
(2, 5) (1, 4)	(1, 4)	10	
(2, 5) (4, 6)	(4, 6)	10	
(2, 5) (6, 7)	(2, 5)	20	
(6, 7)	(6, 7)		
(1, 3) (5, 7)	(1, 3)	30	

3.7.3. Алгоритм Крускала

Алгоритм Крускала відрізняється від алгоритма Пріма тим, що одночасно будується не одне остове дерево, а кілька дерев одночасно (ліс). По мір зростання окремих дерев вони об'єднуються в спільне остове дерево. Нові вершини можуть приєднуватися тільки до активних вершин (голови купи).

Розглянемо роботу алгоритму Крускала до графу (рис. 3.12, табл. 3.2, рис. 3.13):

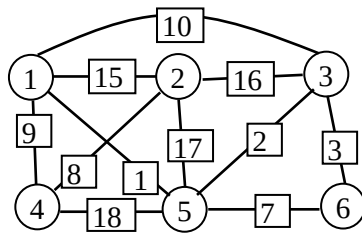


Рис. 3.12. Приклад зваженого графу

Таблиця 3.2

Послідовність кроків побудови остового дерева

Голова купи	Остовий ліс	Дія	Результуюче дерево
(V1, V5)	{V1, V5}	додати	{V1,V5}, {V2}, {V3}, {V4}, {V6}
(V3, V5)	{(V1, V5) (V3, V5)}	додати	{V1, V3, V5}, {V2}, {V4}, {V6}
(V3, V6)	{(V1, V5) (V3, V5) (V3, V6)}	додати	{V1, V3, V5, V6}, {V2}, {V4}
(V5, V6)	{(V1, V5) (V3, V5) (V3, V6)}	залишити	{V1, V3, V5, V6}, {V2}, {V4}
(V2, V4)	{(V1,V5) (V3,V5) (V3,V6), (V2,V4)}	додати	{V1, V3, V5, V6}, {V2, V4}
(V1, V4)	{(V1, V5) (V3, V5) (V3, V6) (V2,V4) (V1, V4)}	додати	{V1, V3, V5, V6, V2, V4}

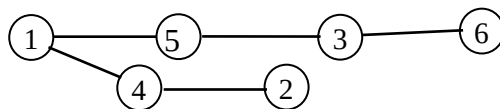


Рис. 3.13. Результуюче остове дерево

3.7.4. Динамічне програмування

Динамічне програмування – це алгоритмічний метод, коли рішення задачі є результатом певної послідовності розв’язків. Метод побудований на послідовному аналізі варіантів. На кожному кроці відкидаються розв’язки, серед яких не може бути оптимального, і таким чином множина конкурентоспроможних варіантів стискується (до одного або кількох варіантів); наприклад, в задачі про рюкзак потрібно послідовно знайти кількість одиниць кожного предмета.

У динамічному програмуванні обробляється одночасно послідовність рішень, серед яких повинне бути оптимальне. Послідовність рішень поступово звужується, для чого використовуються **рекурентні** рівняння. У рекурентних відношеннях є два підходи до пошуку варіантів розв’язку: прямий і зворотний.

Числа Фібоначчі

Розглянемо техніку динамічного програмування на класичному прикладі побудови послідовності чисел Фібоначчі. **N**-не число Фібоначчі рекурсивно визначається так:

$$F(N) = \begin{cases} 1 & \text{if } N = 0 \\ 1 & \text{if } N = 1. \\ F(N-1) + F(N-2) & \text{if } N > 1 \end{cases} \quad (3.2)$$

Потрібно написати програму, яка обчислює **N**-не число Фібоначчі. Цю задачу можна вирішити двома способами: за допомогою рекурсії або за допомогою динамічного програмування.

Рекурсивний спосіб можна реалізувати функцією:

Function Fib(n:integer):integer;

Begin

if n=0 then fib:=1;

else if n=1 then fib:=1;

else fib:=fib(n-1)+fib(n-2);

End;

Недоліком рекурсії є обчислення чисел Фібоначчі, менших **N**, по кілька разів. В динамічному програмуванні для обчислення чисел Фібоначчі використовують дві змінні, які зберігають значення двох останніх чисел. Наступне число визначається через два попередніх.

Динамічне програмування – такий спосіб побудови алгоритмів, коли рішення задачі можна розбити на підзадачі. Розв’язок наступної задачі – суперпозиція попередніх розв’язків. Рекурсивна схема може

використовуватися для **бектрекінгу** – пошуку у графі з поверненням назад. Задача знаходження гамільтонових циклів має експоненційну складність: знайти, шлях, що проходить по одному разу через всі вершини графа.

3.8. Вирішувані інтелектуальних задач

Вирішувач інтелектуальних задач – штучна система, яка сприймає формалізований опис деякої задачі і на основі його розробляє план рішення.

3.8.1. Загальний вирішувач задач

В кінці 50-х років і на початку 60-х розвивалася програма „Загальний вирішувач задач” (GPS). В рамках цієї програми створено програму „Логік-теоретик” (А.Ньюелл, Г.Саймон, Дж. Шоу) для доведення теорем математичної логіки (генерація гіпотези й перевірка її істинності).

Схема прийняття рішення наступна:

1. Проаналізувати поточну ситуацію.
2. Порівняти ситуацію з бажаною; якщо відмінностей немає – кінець роботи.
3. З’ясувати, які оператори можна застосувати для зменшення існуючої різниці.
4. Послідовно застосовувати знайдені оператори.
5. Повернутися на крок 1.

Метод виявився не достатньо універсальним – його не вдалося використати для гри в шахи.

3.8.2. Ігрові задачі як задачі прийняття рішень

Ігрові задачі – планування цілеспрямованих дій і прийняття рішень. Характерна особливість ігрових задач – наявність **суперника**.

Теорія ігор – математична дисципліна, яка вивчає аспекти поведінки учасників конфліктних ситуацій та має на меті виробити оптимальні для кожного з них стратегії (засоби досягнення цілей). Конфліктною називають ситуацію, коли гравці мають різні цілі (різні функції виграшу).

Розглянемо найпростіший тип ігор – одно крокові антагоністичні ігри.

Розглянемо основи теорії **однокрокових ігор**. Нехай грають два гравця **A** і **B**, які мають такі можливості:

- Гравець **A** може вибрати одну з **m** стратегій α_i , $i=1\dots m$.
- Гравець **B** може вибрати одну з **n** стратегій β_j , $j=1\dots n$.

Обидва гравці здійснюють свій вибір одночасно, після чого гравець **B** платить гравцю **A** суму r_{ij} , яка залежить від обраних стратегій. Оскільки виграш одного гравця дорівнює програшу іншого, сума виграшів двох гравців дорівнює нулю і така гра називається грою з нульовою сумою або антагоністичною.

Одно крокові антагоністичні ігри характеризуються **матрицею виграшів**, елементи якої дорівнюють виграшам одного з гравців (**A**).

$$\begin{matrix} & \beta_1 & \beta_2 \\ \alpha_1 & \begin{pmatrix} 1 & -1 \end{pmatrix} \\ \alpha_2 & \begin{pmatrix} -1 & 1 \end{pmatrix} \\ \alpha_3 & \begin{pmatrix} 1 & -1 \end{pmatrix} \\ \alpha_4 & \begin{pmatrix} -1 & 1 \end{pmatrix} \end{matrix} \quad (3.3)$$

Однокрокові ігри можна розглядати задачі з прийняттям рішень в умовах невизначеності.

3.8.3. Позиційні ігри з оптимальними стратегіями

Досить дослідженими є позиційні ігри двох осіб: шахи, шашки, „хрестики-нулики”... . За правилами гри кожен гравець по черзі робить свій хід. Ці ігри є **детермінованими** і з повною інформацією, відносяться до скінченних антагоністичних ігор.

Базою теорії ігор вважається результат **Льюїса і Райфа**: для кожного гравця у будь-якій позиції існує оптимальна стратегія, тобто алгоритм визначення чергового ходу, що призведе до оптимального результату.

Ключовим поняттям в теорії позиційних ігор є **дерево гри** (дерево аналізу). Воно будується таким чином:

1. Кожна вершина дерева відповідає окремій позиції.
2. Корінь дерева відповідає позиції, що аналізується.
3. Листки дерева відповідають завершальним позиціям з відомою функцією виграшу.
4. Дуги відповідають ходам.

Суперники традиційно носять імена α і β . За домовленістю вважається, що правило вибору ходу в початковій позиції належить гравцеві α . Вершини, в яких право ходу належить α , називають **α -вершини**; відповідно називають **β -вершини**.

За домовленістю стандартний хід в шахах, шашках і т.д. вважається **півходом**, а **хід** складається з ходу гравця і відповіді суперника.

Дерево гри може бути **експліцитним** (задається явно) й **імпліцитним** (породжується по мірі необхідності).

Для пошуку по дереву використовують стратегії в **ширину** та **глибину**.

Процедура, яка дозволяє знайти оптимальну стратегію, називається мінімаксною. Мінімаксна процедура є процедурою повного перебору, яка задається правилами:

1. Мінімаксна оцінка завершальних позицій дорівнює функції виграшу.
2. Мінімаксна оцінка альфа-вершини дорівнює максимуму мінімакських оцінок безпосередніх наступників.
3. Мінімаксна оцінка бета-вершини дорівнює мінімуму мінімакських оцінок безпосередніх наступників.

Розглянемо дерево гри (рис. 3.14); на основі такого дерева виділяються ходи для оптимальної гри суперників.

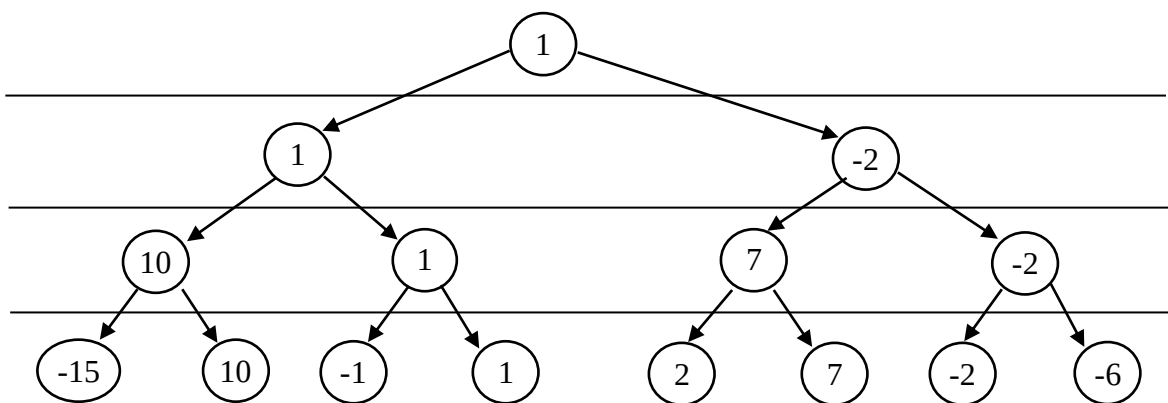


Рис. 3.14. Аналіз дерева гри на основі мінімаксної процедури.

Кількість можливих варіантів ходів є надзвичайно великою. Наприклад (для гри в шахи), якщо на кожному кроці можна зробити 30 ходів, а партія складається з 80 півходів, то кількість позицій 30^{80} .

Обмеження глибини перебору

Для обмеження перебору звичайно зменшують глибину перебору **d**. Горизонтом для даної позиції називають множину позицій для глибини **d**. На горизонті отримуються **абсолютно завершальні** (оцінка позиції визначена) та **відносно завершальні** позиції. Для відносно завершальних позицій вводяться **статичні оціночні функції**, які оцінюють отриману ситуації без врахування динаміки. Наприклад, для шахів враховується кількість та вид фігур, їх розміщення та ін.

В шашках статична оцінка може бути обчислена як функція **$6k+4m+u$** , де **k** – перевага в дамках, **m** – у простих шашках, **u** – перевага в рухливості.

3.8.4. Альфа – бета відтинання

Альфа-бета відтинання є одним зі способів скорочення перебору. **Відтинанням** у деякій вершині називається припинення аналізу цієї вершини разом з усіма її наступниками.

Базова ідея відтинань наступна: якщо деякий хід досягає **максимально можливого виграшу**, то інші його альтернативи не розглядаються. З кожною вершиною зв'язується **попередня оцінка**, яка змінюється у ході аналізу її наступників. Перед початком аналізу попередня оцінка **альфа**-вершини дорівнює $-\infty$, а **бета**-вершини $+\infty$. Після завершення аналізу всіх наступників попередня оцінка набуває остаточного значення, тобто стає **остаточною** оцінкою.

Попередня оцінка альфа-вершини дорівнює максимуму з остаточних оцінок безпосередніх наступників.

Попередня оцінка бета-вершини дорівнює мінімуму з остаточних оцінок безпосередніх наступників.

Правила відтинань такі:

1. Відтинання у альфа-вершині відбувається, коли попередня оцінка цієї вершини стає не меншою, ніж попередня оцінка будь-якого бета-попередника.
2. Відтинання у бета-вершині відбувається, коли попередня оцінка цієї вершини стає не більшою, ніж попередня оцінка будь-якого альфа-попередника.

Якщо глибина перебору **d**, а **b** – коефіцієнт розгалуження, то при повному переборі потрібно розглядати варіантів **b^d** . Альфа-бета відтинання потребує приблизно **$4b^{d/2}$** позицій.

Огляд сучасних шахових програм

Програми Hiarch, Rebel, Mchess показують якість гри на рівні гросмейстерів.

Спеціалізований шаховий комп'ютер RS/6000 в 1997 році виграв у Каспарова з рахунком 3,5 на 2,5.

Прогрес шахових програм досягнуто за рахунок:

1. Зростання швидкості обчислень.
2. Використання дебютних та ендшпільних бібліотек.
3. Використання бібліотек найкращих ходів.
4. Методу форсованих варіантів (перевіряються форсовані варіанти нижче глибини перебору – шахи і взяття фігур).

3.9. Евристичні алгоритми

Евристики використовуються як засіб зменшення перебору, описуються правилами субоптимальної поведінки в певних ситуаціях. Прислів'я – приклади евристик (сім разів відміряй, а потім відріж).

Евристика є довільним правилом, стратегією, методом, хитрістю, спрощенням або іншим засобом суттєвого зменшення, обмеження обсягу пошуку розв'язку.

Прикладом використання евристик є задача розфарбовування графу. В 19 ст. постало питання, якою мінімальною кількістю фарб можна розфарбувати топологічну карту. Тоді з'явилася ідея розглядати карту як планарний граф, і тому виникла проблема розфарбовування графу мінімальною кількістю фарб. Спочатку розглядалися лише планарні, а потім – й довільні графи. До задачі розфарбовування графу можна звести задачі класифікації об'єктів, розподілу ресурсів та ін.

Планарний граф – це граф, який можна зобразити на площині так, що різним площинам відповідають різні точки площини; жодні два ребра не мають спільних точок, крім інцидентних їм обох вершин.

Нехай $G=(V, E)$ – неорієнтований граф, для якого є n вершин. **Розфарбовуванням** графу називатимемо відображення f множини V на множину C таку, що $C=c_1 \dots c_m$. Елементи множини C називатимемо **фарбами**.

Вірним розфарбовуванням графу називатимемо таке, при якому дві суміжні вершини розфарбовані різними кольорами. Вірне розфарбовування розбиває множини V на m хроматичних класів.

Хроматичним числом $\chi(G)$ називають мінімальне число фарб. Називатимемо граф k -розфарбовуванням, якщо $\chi(G) \leq k$, та хроматичним, якщо $\chi(G) = k$.

З розфарбовуванням графу пов'язано дві задачі: 1) знайти хроматичне число графу $\chi(G)$; 2) знайти хоча б одне вірне розфарбовування.

Для планарного графу доведено гіпотезу, що його можна розфарбувати не більше ніж 4-ма фарбами.

Існують точні та евристичні методи розфарбовування. Точні методи мають експоненційну складність. Для графу G кількість фарб $\chi(G) \leq k \leq n$.

Теорема Кенінга. Граф є біхроматичним (2-х хроматичним), тоді й тільки тоді, коли він не містить непарних простих циклів. З теореми випливає, що будь-яке дерево є біхроматичним.

При розфарбовуванні графу звичайно послідовно перебираються вершини, при цьому кожна вершина розфарбовується відповідним кольором.

Контрольні запитання до розділу 3

1. У чому полягає задача розпізнавання образів? Що таке клас образів?
2. Сформулюйте математичну постановку задачі розпізнавання образів.
3. За яким принципом вибираються ознаки образу?
4. Якими ознаками описуються зображення фігур травлення кристалів?
5. Наведіть приклад розпізнавання образів за набором ознак.
6. У чому полягає попередня обробка зображень перед їх розпізнаванням?
7. Перелічіть основні методи розпізнавання образів.
8. У чому полягає принцип шаблонних методів розпізнавання?
9. Поясніть принцип кореляційного методу розпізнавання.
10. У чому полягає метод допустимих перетворень? Для розпізнавання яких зображень він може бути застосований?
11. Як обчислюється коефіцієнт подібності зображень?

12. Як в розробленій програмі виконується зсув і поворот зображення зразка?
13. Як впливають межі зсуву і повороту зображення зразка на точність розпізнавання?
14. Наведіть приклади геометричних перетворень зображення.
15. Яким чином за коефіцієнтом подібності зображень можна визначити, до якого класу символів відноситься зображення зразка?

4. ШТУЧНІ НЕЙРОННІ МЕРЕЖІ

Про роботу мозку в даний час відомо дуже мало, тому штучні **нейронні мережі** (neural networks) є надзвичайно спрощеною моделлю біологічних нейронних мереж [3, 17-23]. Особливістю **нейромереж** (neuronet) є те, що вони **навчаються**, а не програмуються.

4.1. Біологічні нейронні мережі

Нервова система людини побудована з елементів (нейронів), має приголомшуючу складність. Близько 10^{11} нейронів беруть участь в приблизно 10^{15} передаючих зв'язках, що мають довжину метр і більше. Кожен нейрон володіє багатьма якостями, спільними з іншими елементами тіла, але його унікальною здатністю є прийом, обробка і передача електрохімічних сигналів по нервових шляхах, які утворюють комунікаційну систему мозку (рис. 4.1).

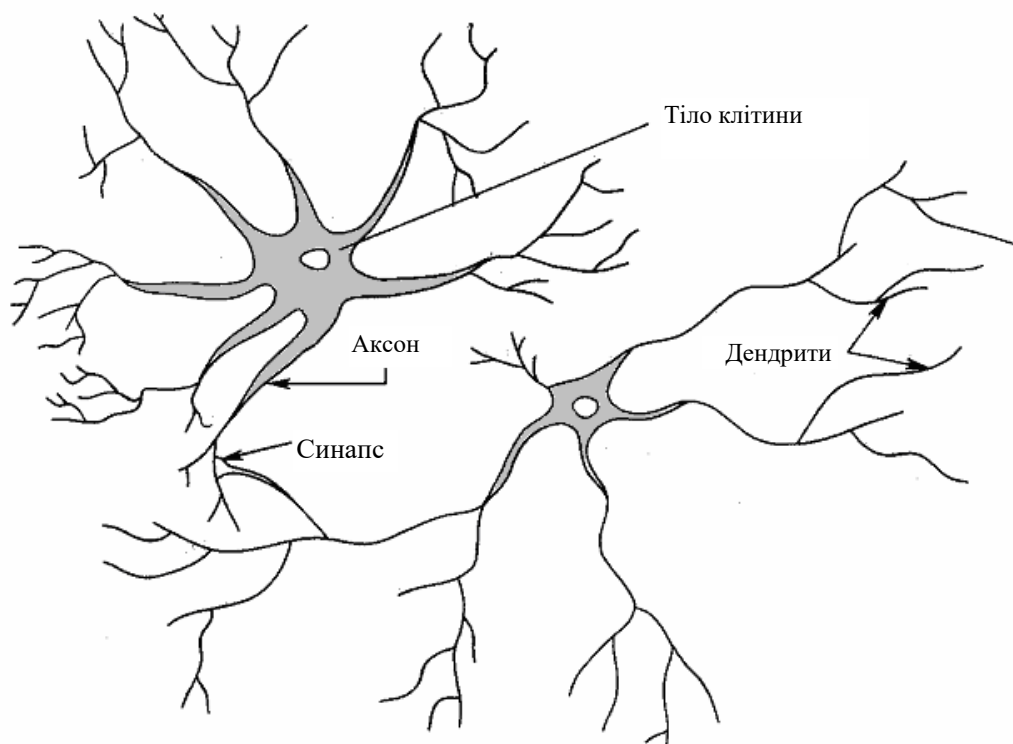


Рис. 4.1. Біологічний нейрон

На рис. 4.1 показана структура пари типових біологічних нейронів. **Дендрити** (входи нейрона) йдуть від тіла нервової клітини до інших нейронів, де вони приймають сигнали в точках з'єднання (**синапсах**). Прийняті синапсом вхідні сигнали підводяться до тіла

нейрона. Тут вони підсумовуються, причому одні входи стимулюють активізацію нейрона, а інші – зниження його активності. Коли сумарна активність (збудження) нейрона перевищує деякий поріг, нейрон переходить в активний стан, посилаючи по **аксону** (виходу нейрона) сигнал іншим нейронам. У цієї основної функціональної схеми багато спрощень і виключень, проте більшість штучних нейронних мереж моделює лише ці прості властивості.

4.2. Штучний нейрон

Основними компонентами нейромережі є **нейрони** /neurons/ (елементи, вузли), які з'єднані зв'язками. Сигнали передаються по **зваженим зв'язкам** (connection), з кожним з яких пов'язаний **ваговий коефіцієнт** (weighting coefficient) або вага.

Використовуються програмні та апаратні моделі штучних нейронних мереж (ШНМ), найбільш поширені моделі – програмні. Використання ШНМ – для розпізнавання образів, прогнозування, створення асоціативної пам'яті.

Штучний нейрон імітує в першому наближенні властивості біологічного нейрона. На вхід штучного нейрона поступає множина сигналів, які є виходами інших нейронів. Кожен вхід множиться на відповідну вагу, аналогічну його синаптичній силі, і всі виходи підсумовуються, визначаючи рівень активації нейрона. На рис. 4.2 представлена модель, що реалізує цю ідею. Хоча мережеві парадигми досить різноманітні, в основі майже всіх їх лежить ця конфігурація. Тут множина вхідних сигналів, позначених $x_1, x_2, \dots, x_n$, поступає на штучний нейрон. Ці вхідні сигнали, в сукупності позначаються вектором X , відповідають сигналам, що приходять в синапси біологічного нейрона. Кожен сигнал множиться на відповідну вагу $w_1, w_2, \dots, w_n$, і поступає сумуючий блок, позначений Σ . Кожна вага відповідає «силі» одного біологічного синаптичного зв'язку (множина ваг в сукупності позначається вектором W). Сумуючий блок, який відповідає тілу біологічного нейрона, складає зважені входи алгебраїчно, створюючи вихід, який ми називатимемо **NET**. У векторних позначеннях це може бути компактно записано таким чином

$$NET = X \cdot W. \quad (4.1)$$

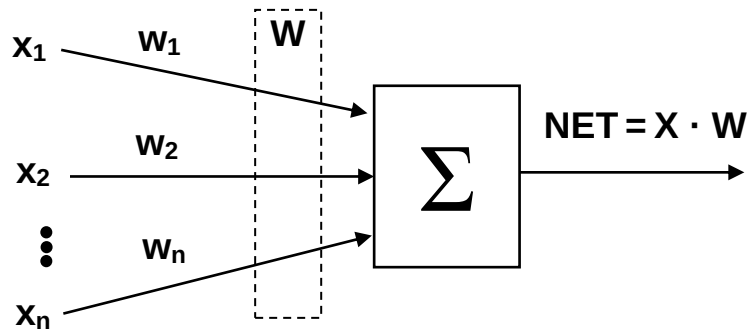


Рис. 4.2. Штучний нейрон

4.3. Активаційні функції нейронів

Сигнал NET далі, як правило, перетворюється активаційною функцією F і дає вихідний нейронний сигнал $OUT = F(NET)$. Активаційна функція $F(NET)$ може бути такою [3]:

1. Пороговою бінарною функцією (рис. 4.3, а)

$$OUT = \begin{cases} 1, & NET \geq T \\ 0, & NET < T \end{cases}, \quad (4.2)$$

де T – деяка постійна порогова величина, або ж функція, що точніше моделює нелінійну передавальну характеристику біологічного нейрона.

2. Лінійною обмеженою функцією (рис. 4.3, б)

$$OUT = \begin{cases} 1, & NET \geq T \\ NET, & 0 \leq NET < T \\ 0, & NET < 0 \end{cases}. \quad (4.3)$$

3. Функцією гіперболічного тангенса (рис. 4.3, в)

$$OUT = th(C \cdot NET) = \frac{\exp(C \cdot NET) - \exp(-C \cdot NET)}{\exp(C \cdot NET) + \exp(-C \cdot NET)}, \quad (4.4)$$

де $C > 0$ – коефіцієнт ширини сигмоїди по осі абсцис (звичайно $C=1$).

4. Сигмоїдною (S-подібною) або логістичною функцією (рис. 4.3, г)

$$OUT = \frac{1}{1 + e^{-C \cdot NET}}. \quad (4.5)$$

З виразу для сигмоїда очевидно, що вихідне значення нейрона лежить у діапазоні $[0, 1]$. Популярність сигмоїдної функції зумовлюють такі її властивості:

- здатність підсилювати слабкі сигнали сильніше, ніж великі, і опиратися „насиченню” від потужних сигналів;

- монотонність і диференційованість на всій осі абсцис;
- простий вираз для похідної

$$F'(NET) = C \cdot F(NET) \cdot (1 - F(NET)), \quad (4.6)$$

що дає можливість використовувати широкий спектр оптимізаційних алгоритмів.

У штучному нейроні з активаційною функцією (рис. 4.4) блок, позначений F , приймає сигнал NET і видає сигнал OUT . Якщо блок F звужує діапазон зміни величини NET так, що при будь-яких значеннях NET значення OUT належать деякому кінцевому інтервалу, то F називається «стискаючою» функцією. Як «стискаюча» функція часто використовується сигмоїдна функція.

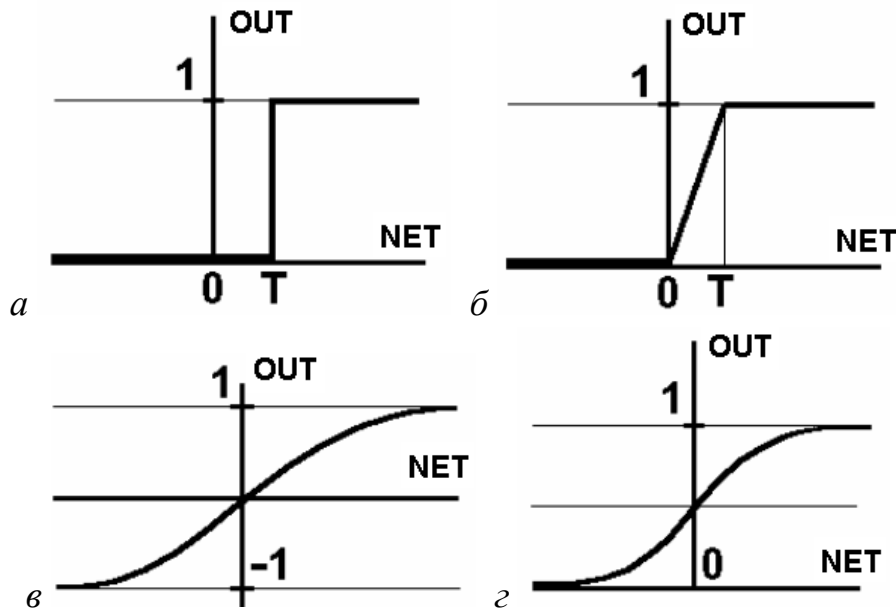


Рис. 4.3. Вид функцій активації: a – порогова бінарна функція; $б$ – лінійна обмежена функція; $в$ – гіперболічний тангенс; $г$ – сигмоїд

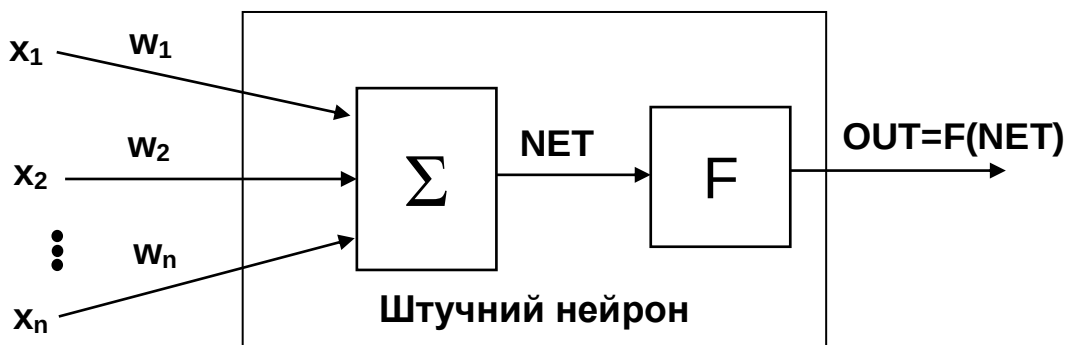


Рис. 4.4. Штучний нейрон з активаційною функцією F

4.4. Одношарові нейронні мережі

Хоча один нейрон і здатний виконувати прості процедури розпізнавання, ефективність нейронних обчислень залежить від з'єднань нейронів у мережі (рис. 4.5). Зазначимо, що вершини-круги зліва служать лише для розподілу вхідних сигналів і тому не вважатимуться шаром. Кожен елемент із множини входів X окремою вагою сполучений із кожним штучним нейроном.

Зручно вважати вагу елементами матриці W . Матриця має n рядків і m стовпців, де n – кількість входів, а m – кількість нейронів. Наприклад, $w_{3,2}$ – це вага, що пов'язує третій вхід із другим нейроном. Таким чином, обчислення вихідного вектора Y , компонентами якого є виходи OUT нейронів, зводиться до матричного множення $Y = X \cdot W$, де Y і X – вектори-рядки.

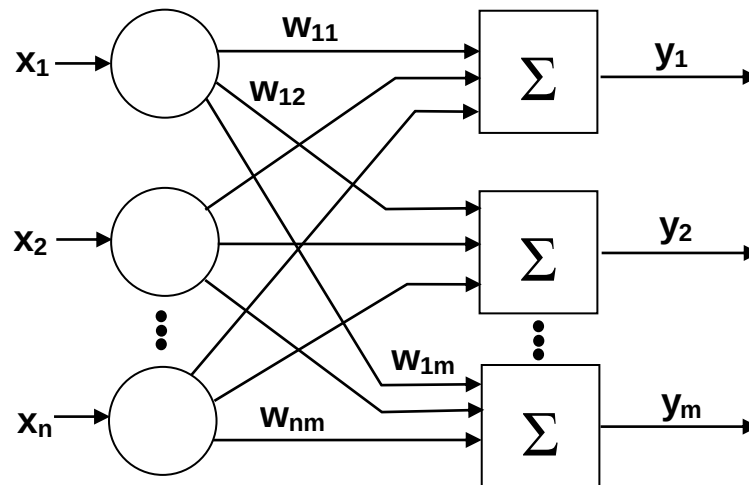


Рис. 4.5. Одношарова ШНМ

4.5. Персептрони і зародження штучних нейромереж

Персептрони складаються з одного шару штучних нейронів (рис. 4.6), сполучених за допомогою вагових коефіцієнтів із множиною входів (рис. 4.7).

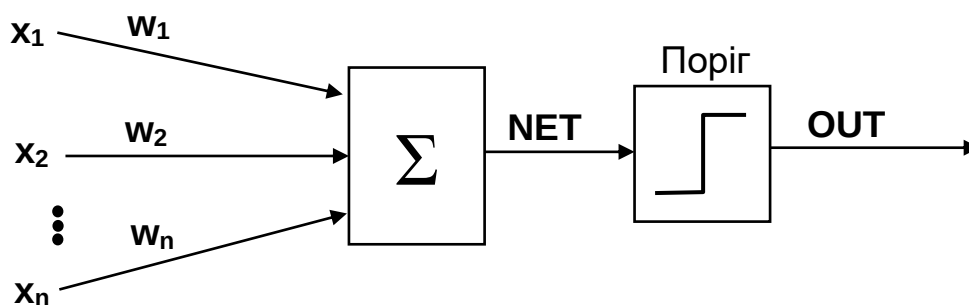


Рис. 4.6. Нейрон персептрона

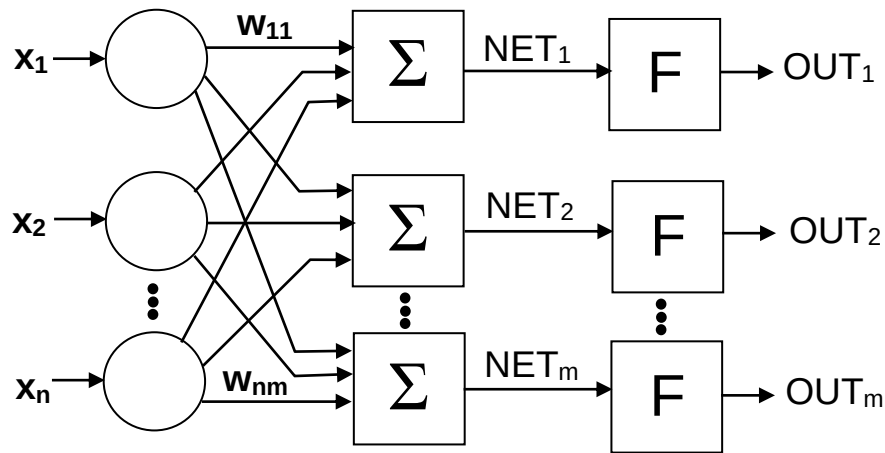


Рис. 4.7. Персептрон

Перші персептрони були створені Ф. Розенблатом у 60-х роках і викликали великий інтерес. Первинна ейфорія змінилася розчаруванням, коли виявилося, що персептрони не здатні навчитися розв'язанню ряду простих задач. Проте відкриття методів навчання багатошарових ШНМ сприяло відродженню інтересу до ШНМ.

4.6. Багатошарові нейронні мережі

Багатошарові мережі (рис. 4.7) володіють значно більшими можливостями, ніж одношарові. Проте багатошарові мережі можуть привести до збільшення обчислювальної потужності в порівнянні з одношаровими лише в тому випадку, якщо активаційна функція між шарами буде *нелінійною* (оскільки множення матриць асоціативне, то $X \cdot (W_1 \cdot W_2)$ еквівалентно $(X \cdot W_1) \cdot W_2$).

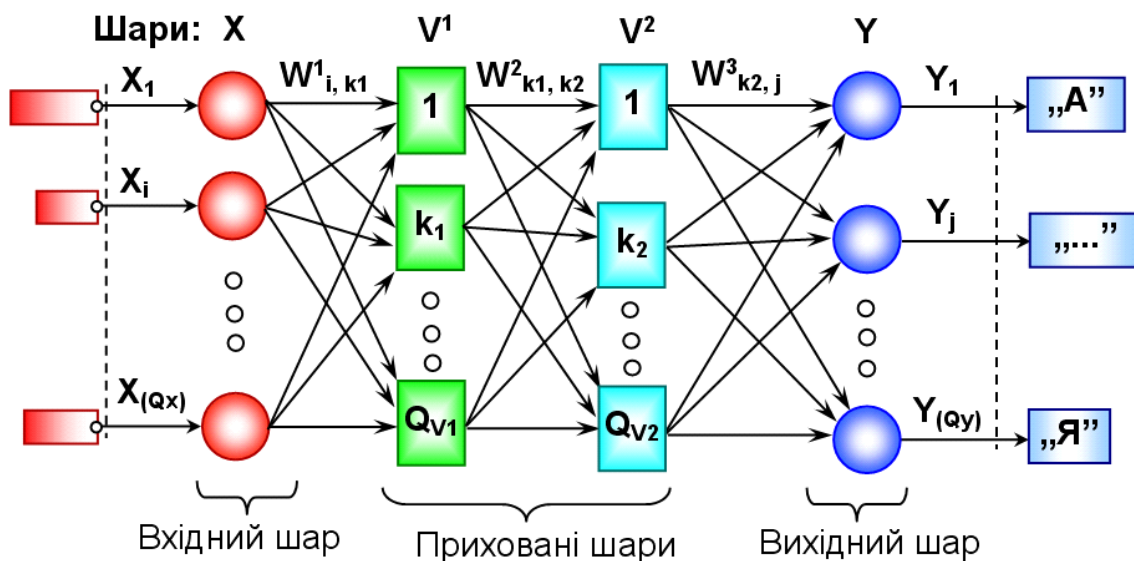


Рис. 4.8. Багатошарова ШНМ

4.7. Навчання нейронних мереж

Мережа навчається, щоб для деякої множини входів X давати бажану множину виходів Y . Навчання здійснюється шляхом послідовного пред'явлення вхідних векторів з одночасним налагодженням ваг відповідно до певної процедури. В процесі навчання ваги мережі поступово стають такими, щоб кожен вхідний вектор виробляв потрібний вихідний вектор. Розрізняють алгоритми навчання з учителем і без учителя.

Навчання з вчителем передбачає, що для кожного вхідного вектора X існує цільовий вектор Y^T , що є необхідним виходом. Разом вони називаються навчальною парою. Звичайно мережа навчається для деякої кількості таких навчальних пар (навчальної множини). В ході навчання зчитується вхідний вектор X , обчислюється вихід мережі Y і порівнюється з відповідним цільовим вектором Y^T , різниця $D \sim Y^T - Y$ за допомогою зворотного зв'язку подається в мережу і змінюються ваги W відповідно до алгоритму, прагнучого мінімізувати помилку ϵ . Зчитування векторів навчальної множини й налагодження ваг виконується доти, поки сумарна помилка для всієї навчальної множини не досягне заданого низького рівня.

Навчання без вчителя. Незважаючи на численні прикладні досягнення, навчання з вчителем критикувалося за свою біологічну неправдоподібність. Важко уявити навчальний механізм у мозку, який би порівнював бажані й дійсні значення виходів, виконуючи корекцію за допомогою зворотного зв'язку. Якщо допустити подібний механізм в мозку, то звідки тоді виникають бажані виходи? Навчання без учителя – набагато правдоподібніша модель навчання в біологічній системі. Розвинена Кохоненом і багатьма іншими, вона не потребує цільового вектора для виходів і, отже, не вимагає порівняння з ідеальними відповідями. Навчальна множина складається лише з вхідних векторів. Навчальний алгоритм налагоджує вагу мережі так, щоб виходили узгоджені вихідні вектори, тобто щоб пред'явлення досить близьких вхідних векторів давало однакові виходи.

4.8. Алгоритм зворотного розповсюдження помилки

Зворотне розповсюдження помилки (**backpropagation**) означає, що сигнали помилки з виходу мережі використовуються для корекції ваг попередніх шарів. Розглянемо алгоритм навчання на прикладі 3-шарової ШНМ (рис. 4.8), структура якої така:

1. *Вхідний шар* X ; стани його елементів записані у векторі X_i , де $i = 1 \dots Q_X$. Розмір навчальної множини (кількість векторів) на вході X дорівнює Q_N , номер вектора $n = 1 \dots Q_N$.

2. *Приховані шари*

Шар V^1 (рівень $L = 1$): вектор V_{k1}^1 , де $k_1 = 1 \dots Q_{V1}$; матриця ваг $W_{i,k1}^1$; різниця векторів D_{k1}^1 ;

Шар V^2 ($L = 2$): вектор V_{k2}^2 , де $k_2 = 1 \dots Q_{V2}$; матриця ваг $W_{k1,k2}^2$; різниця D_{k2}^2 ;

3. *Вихідний шар* ($L = 3$): стани його елементів записані у векторі Y_j ; $j = 1 \dots Q_Y$; матриця ваг $W_{k2,j}^3$; різниця D_j^3 ; істинний вихід (true) описується вектором Y_j^T , $j = 1 \dots Q_Y$. Тобто Y_j^T – істинне значення для елемента j , Y_j – його реальних вихід.

ШНМ навчається за таким алгоритмом (рис. 4.9).

1. *Ініціалізація*. Початкові вагові коефіцієнти W встановлюються рівними малим випадковим значенням з діапазону $[-\Delta W, \dots \Delta W]$ (наприклад, $\Delta W = 0.3$):

$$W_{i,k1}^1 = (Rnd - 0,5) \cdot 2 \cdot \Delta W, \quad i = 1 \dots q_X, \quad k_1 = 1 \dots q_{V1}. \quad (4.7)$$

У вагових матрицях рядки відповідають елементам, від яких йдуть зв'язки, а стовпці – до яких йдуть зв'язки.

2. *Нормалізація* (масштабування) значень усіх векторів X , Y^T (для кожного типу окремо) в діапазон $(MinN; MaxN)$, наприклад $MinN = 0.01$; $MaxN = 0.99$.

$$X_i^n = MinN + \frac{(X_i - X_{\min}) \cdot (MaxN - MinN)}{(X_{\max} - X_{\min})}. \quad (4.8)$$

3. *Пряме розповсюдження* (Direct) полягає у знаходженні вихідного вектора Y на основі вхідного X за такими формулами:

$$\text{Шар 1: } Net_{k1} = \sum_{i=1}^{q_X} X_i \cdot W_{i,k1}^1, \quad V_{k1}^1 = f\left(\frac{1}{1 + \exp(-Net_{k1})}\right), \quad \text{де } k_1 = 1 \dots q_{V1}.$$

$$\text{Шар 2: } Net_{k2} = \sum_{k1=1}^{q_{V1}} V_{k1}^1 \cdot W_{k1,k2}^2, \quad V_{k2}^2 = f\left(\frac{1}{1 + \exp(-Net_{k2})}\right), \quad \text{де } k_2 = 1 \dots q_{V2}.$$

$$\text{Шар 3: } Net_j = \sum_{k2=1}^{q_{V2}} V_{k2}^2 \cdot W_{k2,j}^3, \quad Y_j = f\left(\frac{1}{1 + \exp(-Net_j)}\right), \quad \text{де } j = 1 \dots q_Y.$$

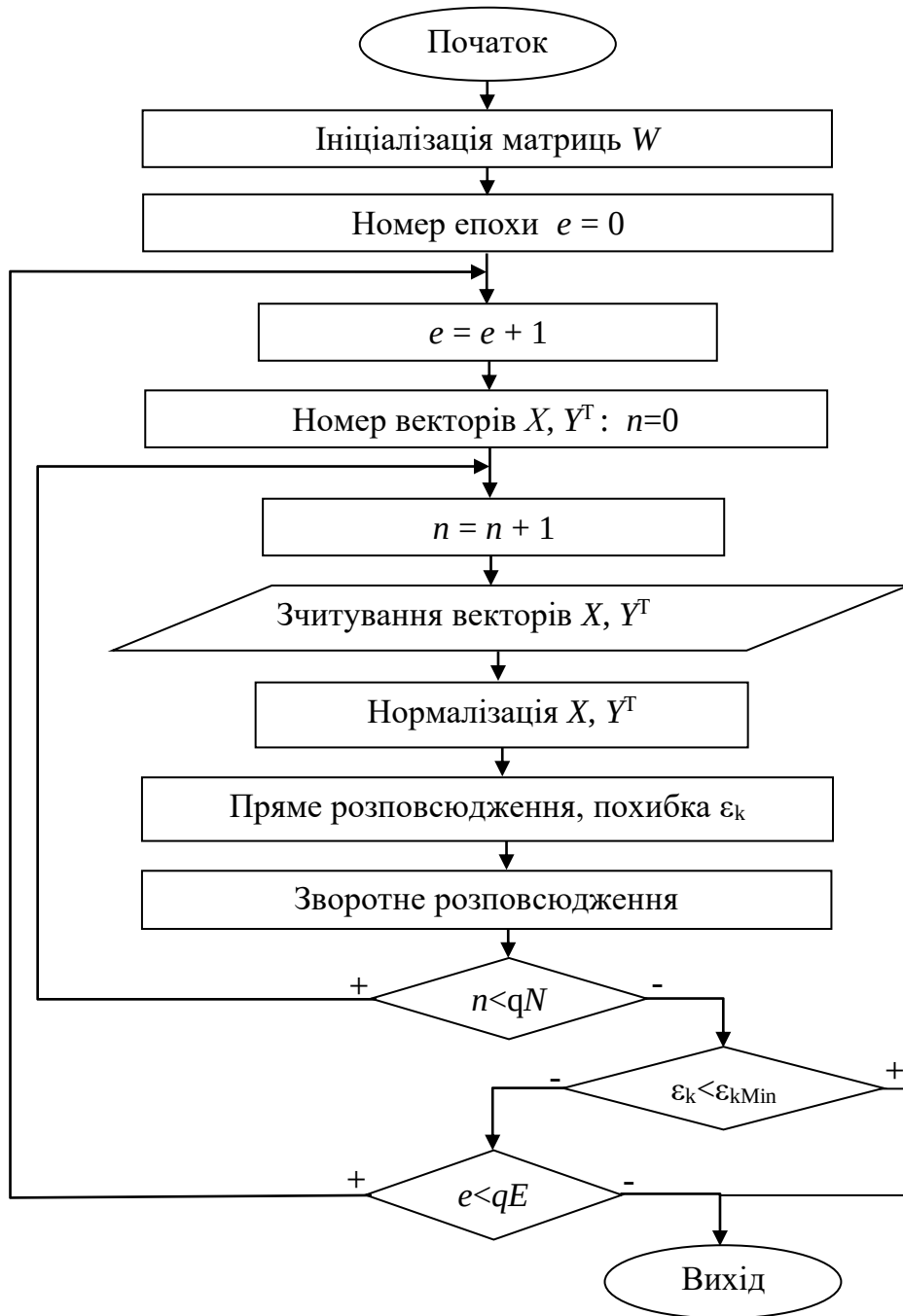


Рис. 4.9. Алгоритм навчання нейромережі зі зворотним розповсюдженням помилки

У результаті прямого розповсюдження можна обчислити помилки навчання мережі (для всіх векторів навчальної множини).

Лінійна помилка

$$\varepsilon = \sum_{n=1}^{qN} \sum_{j=1}^{qY} |Y_{j,n} - Y_{j,n}^T|, \quad (4.9)$$

Середня квадратична помилка (mse – Mean squared error)

$$\varepsilon_k = \frac{1}{qN} \frac{1}{qY} \sum_{n=1}^{qN} \sum_{j=1}^{qY} (Y_{j,n} - Y_{j,n}^T)^2, \quad (4.10)$$

Середня відносна помилка (0 – 100%)

$$\varepsilon_{\sigma} = \frac{1}{qN \cdot qY} \sum_{n=1}^{qN} \sum_{j=1}^{qY} \left(\frac{|Y_{j,n} - Y_{j,n}^T|}{|Y_{j,n}^T| + 1} \right) \cdot 100\%. \quad (4.11)$$

Якщо значення сумарної квадратичної помилки ε_k менше від заданого, то процес навчання мережі завершується.

4. *Зворотне розповсюдження помилки (backpropagation)* полягає в корекції вагових коефіцієнтів через сигнал різниці D .

Шар 3:

$$D_j^3 = Y_j \cdot (1 - Y_j) (Y_j^t - Y_j), \quad W_{k2,j}^{3(e)} = W_{k2,j}^{3(e-1)} + \eta_Y \cdot D_j^3 \cdot V_{k2}^2, \quad (4.12)$$

де $j = 1..qY$, e – номер епохи (*epochs*); оскільки як активаційна функція використовується сигмоїдна, тому різниця векторів $(Y^T - Y)$ множиться на похідну від сигмоїдної функції: $Y(1 - Y)$.

Шар 2:

$$D_{k2}^2 = \sum_j^{qY} V_{k2}^2 \cdot (1 - V_{k2}^2) \cdot (D_j^3 \cdot W_{k2,j}^3), \quad W_{k1,k2}^{2(e)} = W_{k1,k2}^{2(e-1)} + \eta_{L2} \cdot D_{k2}^2 \cdot V_{k1}^1, \quad (4.13)$$

де $k_2 = 1..qV_2$.

Шар 1:

$$D_{k1}^1 = \sum_{k2}^{qV_2} V_{k1}^1 \cdot (1 - V_{k1}^1) \cdot (D_{k2}^2 \cdot W_{k1,k2}^2), \quad W_{i,k1}^{1(e)} = W_{i,k1}^{1(e-1)} + \eta_{L1} \cdot D_{k1}^1 \cdot X_i, \quad (4.14)$$

де $k_1 = 1..qV_1$, η_Y , η_{L2} , η_{L1} – норми навчання (наприклад, 0.5).

З метою контролю значень матриці W визначаються: *Min* – мінімальне значення; *Max* – максимальне; *Ms* – математичне сподівання; *Sigma* – середньоквадратичне відхилення.

4.9. Мережі зустрічного розповсюдження. Шари Кохонена і Гросберга

Основною перевагою НМ зустрічного розповсюдження є порівняно малий час навчання (приблизно в 100 разів менше, ніж для зворотного розповсюдження помилки) [3]. В зустрічному

розповсюдженні об'єднані два відомі алгоритми: карта **Кохонена** і зірка **Гроссберга**.

Мережа зустрічного розповсюдження функціонує подібно до довідкового бюро. В процесі навчання вхідні вектори асоціюються з відповідними вихідними векторами. Ці вектори можуть бути двійковими або неперервними. Коли мережа навчена, обробка вхідного вектора приводить до необхідного вихідного вектора. Узагальнююча здатність мережі дозволяє одержувати правильний вихід навіть при пошкоджені вхідного вектора. Це дозволяє використовувати дану мережу для розпізнавання і відновлення образів. Структура ШНМ зустрічного розповсюдження наступна (рис. 4.10).

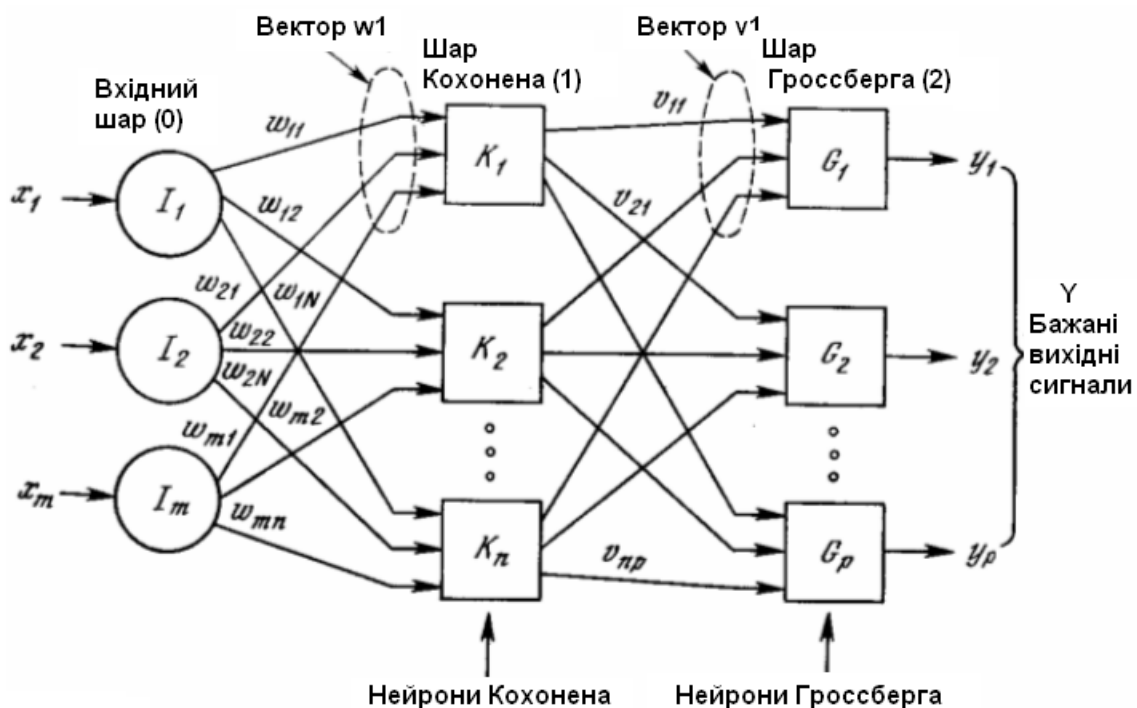


Рис. 4.10. ШНМ зустрічного розповсюдження без зворотних зв'язків

Нейрони шару 0 (круги) служать лише точками розгалуження і не виконують обчислень. Кожен нейрон шару 0 сполучений з кожним нейроном шару 1 (шару **Кохонена**) окремою вагою w_{mn} . Ця вага в цілому розглядається як матриця ваг $\mathbf{W}$. Аналогічно, кожен нейрон в шарі **Кохонена** (шарі 1) сполучений з кожним нейроном в шарі **Гроссберга** (шарі 2) вагою v_{np} . Ця вага утворює матрицю ваг $\mathbf{V}$. Зустрічне розповсюдження функціонує в двох режимах: у нормальному режимі, при якому приймається вхідний вектор $\mathbf{X}$ і

видається вихідний вектор Y , і в режимі навчання, при якому подається вхідний вектор і ваги коректуються, щоб дати необхідний вихідний вектор.

Нормальне функціонування ШНМ зустрічного розповсюдження полягає в наступному. **Шари Кохонена.** У своїй простій формі шар Кохонена функціонує за принципом «переможець забирає все», тобто для даного вхідного вектора один і лише один нейрон Кохонена видає на виході логічну одиницю, всі інші видають нуль. Переможцем є нейрон з максимальним значенням вихідного сигналу $NET_j = \sum_i x_i w_{ij}$.

Вихід нейронів шару Гроссберга є сумою $NET_j = \sum_i x_i w_{ij}$.

Навчання ШНМ помагає в наступному. Шар Кохонена класифікує вхідні вектори в групи схожих. Це досягається за допомогою такого налагодження ваг шару Кохонена, що близькі вхідні вектори активують один і той же нейрон даного шару. Задачею шару Гроссберга є отримання необхідних виходів. Навчання шару Кохонена є самонавчанням, що виконується без вчителя.

4.10. Стохастичні методи навчання нейронних мереж

Стохастичні методи корисні як для навчання штучних нейронних мереж, так і для отримання виходу від вже навченої мережі [3]. Стохастичні методи навчання приносять велику користь, дозволяючи виключати локальні мінімуми в процесі навчання. Проте з ними також зв'язаний ряд проблем. Штучна нейронна мережа навчається за допомогою деякого процесу, що модифікує її вагу. Якщо навчання успішне, то пред'явлення мережі множини вхідних сигналів приводить до появи бажаної множини вихідних сигналів. Є два класи повчальних методів: детерміністичний і стохастичний.

Детерміністичний метод навчання крок за кроком здійснює процедуру корекції ваг мережі, засновану на використанні їх поточних значень, а також величин входів, фактичних виходів і бажаних виходів. Зворотне розповсюдження помилки є прикладом подібного підходу.

Стохастичні методи навчання виконують псевдовипадкові зміни величин ваг, зберігаючи ті зміни, які ведуть до поліпшень. Для навчання мережі може бути використана наступна процедура:

1. Вибрати вагу випадковим чином і змінити її на невелику випадкову величину. Пред'явити множину входів і обчислити виходи.

2. Порівняти виходи з бажаними виходами і обчислити величину різниці між ними. Загальноприйнятий метод полягає в знаходженні різниці між фактичним і бажаним виходами для кожного елементу навчальної пари. Метою навчання є мінімізація цієї різниці (цільової функції).

3. Якщо зміна ваги допомагає (зменшує цільову функцію), то зберегти її, інакше повернутися до первинного значення ваги.

Повторювати кроки з 1 до 3 до тих пір, поки мережа не буде навчена достатньою мірою.

Однією з найбільших проблем при навчанні НМ є локальні мінімуми (рис. 4.11).

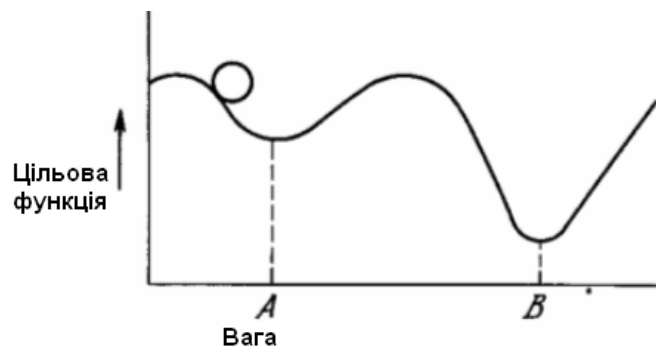


Рис. 4.11. Проблема локальних мінімумів

Для вирішення проблеми локальних мінімумів використовується наступний метод. Штучні нейронні мережі навчаються спочатку грубим налагодженням ваг, а потім більш точним. На першому етапі робляться великі випадкові корекції із збереженням тільки тих змін ваг, які зменшують цільову функцію. Потім середній розмір кроку поступово зменшується, і глобальний мінімум досягається. Це нагадує відпал металу, тому для опису такої методики використовують термін «**імітація відпалу**». У металі, нагрітому до температури, що перевищує його точку плавлення, атоми знаходяться в сильному безладному русі. Як і у всіх фізичних системах, атоми прагнуть до стану мінімуму енергії (єдиному кристалу в даному випадку), але при високих температурах енергія атомних рухів перешкоджає цьому. В процесі поступового

охладження металу виникають все більш низькоенергетичні стани, поки не буде досягнуто глобальний мінімум. В процесі відпалу розподіл енергетичних рівнів описується наступним співвідношенням (розподілом Больцмана):

$$P(e) = \exp(-e/kT). \quad (4.15)$$

де $P(e)$ – ймовірність того, що система знаходиться в стані з енергією e ; k – постійна Больцмана; T – температура за шкалою Кельвіна.

При високих температурах $P(e)$ наближається до одиниці для всіх енергетичних станів. У міру зменшення температури ймовірність високоенергетичних станів зменшується в порівнянні з низькоенергетичними. У випадку НМ $P(e)$ – ймовірність того, що буде збережена зміна ваги, яка приводить до збільшення цільової функції.

4.11. Мережі зі зворотними зв'язками. Мережі Хопфілда

Одними із ШНМ зі зворотними зв'язками є мережі **Хопфілда** [3]. Мережі зі зворотними зв'язками мають шляхи, що передають сигнали від виходів до входів, тому відгук таких мереж є динамічним, тобто після зчитування нового входу обчислюється вихід i , передаючись по мережі зворотного зв'язку, модифікує вхід. Потім вихід повторно обчислюється, і процес повторюється знову і знову. Для стійкої мережі послідовні ітерації приводять до все менших змін виходу, поки вихід не стає постійним. Для багатьох мереж процес ніколи не закінчується, такі мережі називають нестійкими. Нестійкі мережі володіють цікавими властивостями і вивчалися як приклад хаотичних систем.

Розглянемо мережу зі зворотними зв'язками, яка містить 1 шар (рис. 4.12). В першій роботі Хопфілда функція F була просто пороговою відносно порогу T . Вона обчислюється наступним чином:

$$NET_j = \sum_{i \neq j} w_{ij} OUT_i + IN_j,$$

$$OUT = 1, \text{ якщо } NET_j > T_j,$$

$$OUT = 0, \text{ якщо } NET_j < T_j,$$

$$OUT \text{ не змінюються, якщо } NET_j = T_j,$$

Мережа з зворотними зв'язками є **стійкою**, якщо її матриця симетрична й має нулі на головній діагоналі, тобто якщо $w_{ij} = w_{ji}$ й $w_{ii} = 0$ для всіх i .

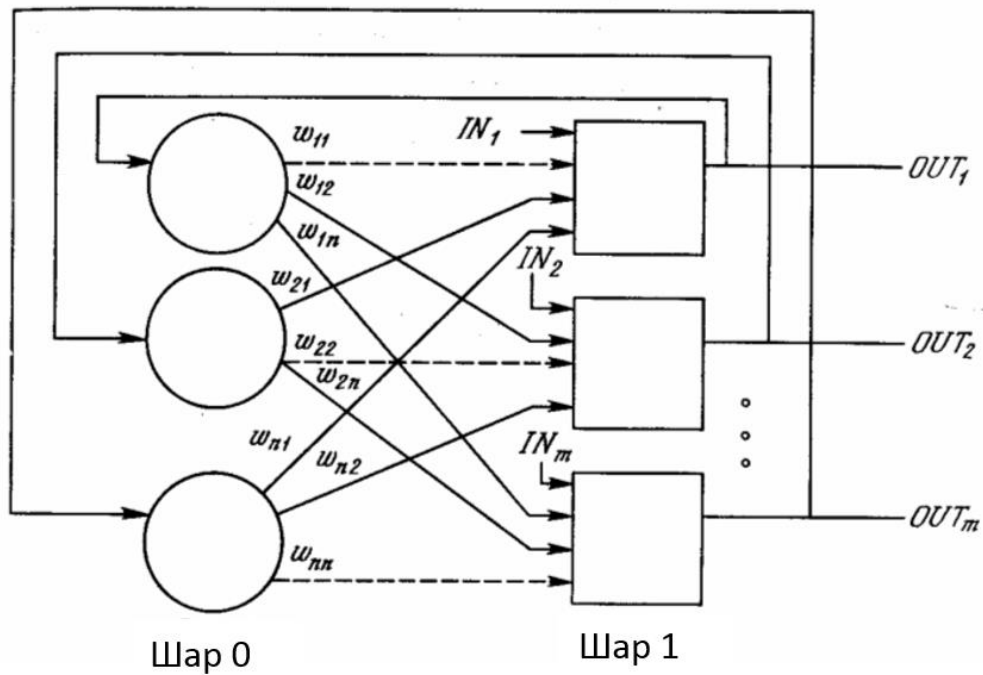


Рис. 4.12. Одношарова ШНМ зі зворотними зв'язками; пунктирні лінії позначають нульові ваги.

Людська пам'ять **асоціативна**, тобто деякий спогад може породжувати велику пов'язану з ним область. Наприклад, декілька музичних тактів можуть викликати цілу гамму спогадів, включаючи пейзажі, звуки і запахи. Навпаки, звичайна комп'ютерна пам'ять адресується локально.

Мережа із зворотним зв'язком формує асоціативну пам'ять. Подібно людській пам'яті по заданій частині потрібної інформації вся інформація витягується з «пам'яті». Щоб організувати асоціативну пам'ять за допомогою мережі із зворотними зв'язками, вага повинна вибиратися так, щоб утворювати енергетичні мінімуми в потрібних вершинах одиничного гіперкуба.

Хопфілд розробив асоціативну пам'ять з безперервними виходами, що змінюються в межах від +1 до -1, відповідних двійковим значенням 0 і 1. Інформація, що запам'ятовується, кодується двійковими векторами і зберігається у вагах згідно наступній формулі:

$$w_{ij} = \sum_{d=1 \dots m} (OUT_{i,d} OUT_{j,d}), \quad (4.16)$$

де m – число вихідних векторів, що запам'ятовуються; d – номер вихідного вектора, що запам'ятовується; $OUT_{i,j}$ – i -компонента вихідного вектора, що запам'ятовується.

4.12. Адаптивна резонансна теорія

Мозок людини виконує важку задачу обробки безперервного потоку сенсорної інформації, одержуваної з навколишнього світу [3]. З потоку тривіальної інформації він повинен виділити життєво важливу інформацію, обробити її і, можливо, зареєструвати в довготривалій пам'яті. Розуміння процесу людської пам'яті є серйозною проблемою; нові образи запам'ятовуються в такій формі, що раніше запам'ятовані не модифікуються і не забуваються. Це створює дилему: яким чином пам'ять залишається пластичною, здатною до сприйняття нових образів, і в той же час зберігає стабільність, що гарантує, що образи не знищуються і не руйнуватимуться в процесі функціонування?

Традиційні штучні нейронні мережі виявилися не в змозі розв'язати **проблему стабільності-пластичності**. Дуже часто навчання новому образу знищує або змінює результати попереднього навчання. В деяких випадках це не істотно. Якщо є тільки фіксований набір навчальних векторів, то вони можуть пред'являтися при навчанні циклічно. У мережах із зворотним розповсюдженням, наприклад, навчальні вектори подаються на вхід мережі послідовно до тих пір, поки мережа не навчиться всьому вхідному набору. Проте, якщо повністю навчена мережа повинна запам'ятати новий повчальний вектор, він може змінити вагу настільки, що потрібно повне перенавчання мережі.

У реальній ситуації мережа піддаватиметься діям, що постійно змінюються; вона може ніколи не побачити один і той же навчальний вектор двічі. При таких обставинах мережа часто не навчатиметься; вона безперервно змінюватиме свою вагу, не досягаючи задовільних результатів. Більш того, в роботі є приклади мережі, в якій тільки чотири повчальні вектори, що пред'являються циклічно, примушують ваги мережі змінюватися безперервно, ніколи не сходяться. Така нестабільність є одним з головних чинників, що примусили Гроссберга і його співробітників досліджувати радикально відмінні конфігурації. **Адаптивна резонансна теорія (АРТ)** є одним з результатів дослідження цієї проблеми.

Мережею АРТ є векторний класифікатор. Вхідний вектор класифікується залежно від того, на якій з множини образів, раніше запам'ятованих, він схожий. Своє класифікаційне рішення мережа

АРТ виражає у формі збудження одного з нейронів шару, що розпізнає. Якщо вхідний вектор не відповідає жодному з образів, що запам'ятовані, то створюється нова категорія. Якщо визначено, що вхідний вектор схожий на один із запам'ятованих векторів (для певного критерію схожості), то запам'ятований вектор буде змінюватиметься під впливом нового вхідного вектора так, щоб стати більш схожим на цей вхідний вектор.

Таким чином, розв'язується дилема стабільності-пластичності. Новий образ може створювати **додаткові класифікаційні категорії**, проте новий вхідний образ не може примусити змінитися існуючу пам'ять (рис. 4.13).

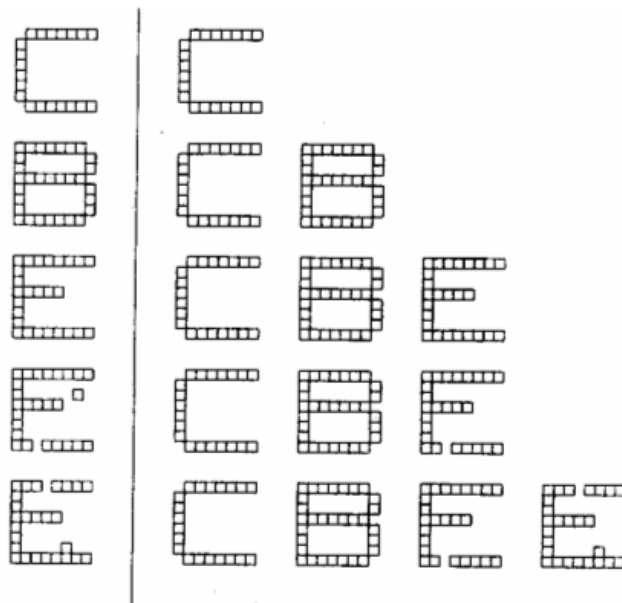


Рис. 4.13. Процес навчання АРТ; лівий стовпець зображень символів – вхідні зображення

АРТ є ефективною при навчанні ШНМ новим образам.

4.13. Згорткові нейронні мережі

Згорткові ШНМ (ЗНМ) (convolutional neural network, CNN) призначені для розпізнавання зображення і належать до технологій глибокого навчання (deep learning), які використовують багат шарову обробку сигналів для великої кількості прикладів [18-21]. ЗНМ – спеціальна архітектура нейромереж, яка запропонована Я. Лекуном в 1988 році і орієнтована на ефективне розпізнавання зображень. В ЗНМ використовуються деякі особливості зорової кори мозку людини, в якій були відкриті так звані прості клітини, що

реагували на прямі лінії під різними кутами, і складні клітини, реакція яких пов'язана з активацією певної множини простих клітин. Назва ЗНМ зумовлена тим, що в них широко використовується операція згортки.

ЗНМ привернули загальну увагу в 2012 році, коли за їх допомогою вдалося зменшити відсоток помилок класифікації в конкурсі ImageNet з 26% до 15%, що стало справжнім проривом. Сьогодні глибинне навчання з використанням ЗНМ лежить в основі послуг багатьох компаній: Facebook використовує нейронні мережі для алгоритмів автоматичного встановлення тегів, Google – для пошуку фотографій користувача, Amazon – для формування рекомендацій товарів, Pinterest – для персоналізації домашньої сторінки користувача. Проте класичним і найбільш популярним варіантом використання ЗНМ є розпізнавання зображень.

Згорткові ШНМ складаються зі згорткових (конволюційних) шарів (англ. convolution layers) та субдискретизуючих шарів (англ. subsampling layers) або (англ. pooling layers) – шарів підвибірки (рис. 4.14). Згорткові шари та шари підвибірки чергуються.

Структура мережі — однонапралена (без зворотних зв'язків), принципово багатoshарова. Для навчання ЗНМ використовуються стандартні методи навчання, в основному метод зворотного поширення помилки.

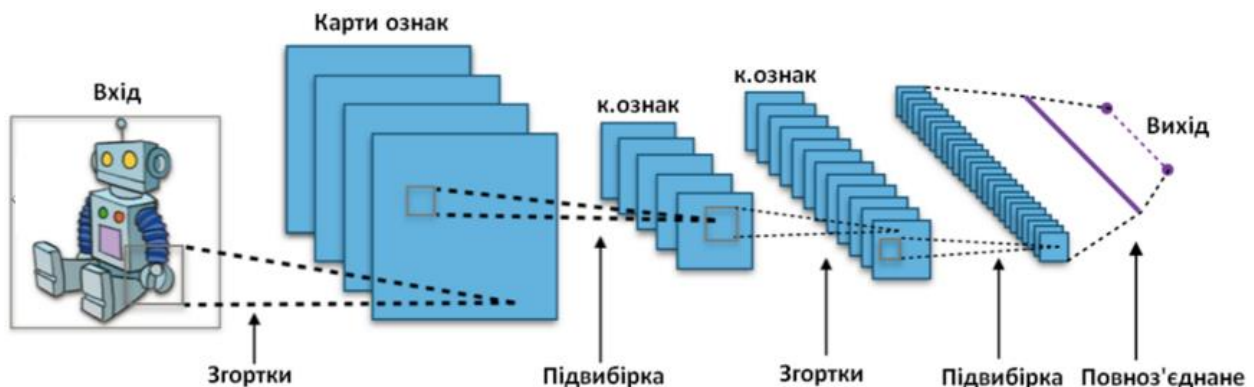


Рис. 4.14. Типова архітектура ЗНМ

Призначення шарів згортки полягає у згортці області попереднього шару з ядром фільтра.

Перший шар ЗНМ завжди згортковий. Згортка виконується в просторовій області над входною матрицею f_1 (розміром $M_1 \times N_1$ елементів) з ядром фільтра w розміром $M_w \times N_w$ елементів

(звичайно $M_w = N_w$), у результаті чого отримується вихідна матриця g_1 (розміром $M_2 \times N_2$ елементи) за формулою [11]:

$$g_1(i, k) = \sum_{m=1}^{M_w} \sum_{n=1}^{N_w} f_1(i - m + m_c + M_{w2}, k - n + m_c + N_{w2}) \cdot w(m, n), \quad (4.17)$$

де $i = 1, \dots, M_C, k = 1, \dots, N_C$;

$M_w = 2 \cdot M_{w2} + 1$ – висота ядра фільтра;

$N_w = 2 \cdot N_{w2} + 1$ – ширина ядра фільтра;

M_{w2}, N_{w2} – цілі частини від половини розмірів ядра фільтра,

$m_c = (M_{w2} + 1)$ – центр ядра фільтра за висотою;

$n_c = (N_{w2} + 1)$ – центр ядра фільтра за шириною.

Сума елементів низькочастотного ядра w дорівнює 1. Операція згортки зображення f_1 з ядром w спрощено записується у вигляді

$$g_1 = f_1 * w. \quad (4.18)$$

Таким чином, один елемент вихідної матриці g_1 формується на основі $M_w \times N_w$ елементів початкової матриці f_1 , які покриваються ядром фільтра w (така область початкового зображення також називається рецептивним полем або полем сприйняття) (рис. 4.15).

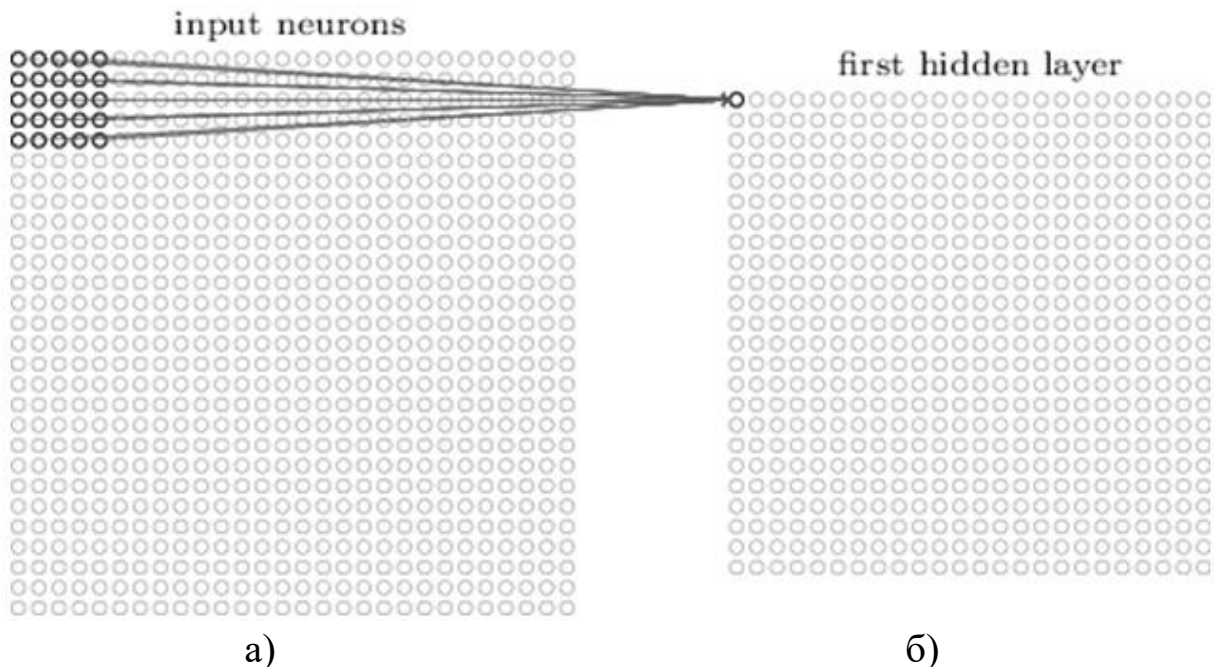


Рис. 4.15. Візуалізація згортки (конволюції) вхідної матриці f_1 (розміром 28×28 елементів) (а) з ядром фільтра розміром 5×5 елементів, у результаті чого отримується вихідна матриця g_1 (карта ознак, розміром 24×24 елементи) (б)

Матрицю w , яка описує ядро фільтра, називають також матрицею ваг.

З метою усунення крайових ефектів вихідна матриця g_1 має розмір, на $2 \cdot M_{w2}$ елементів менший за розмір початкової матриці f_1 за висотою та на $2 \cdot N_{w2}$ елементів менший за шириною, тобто:

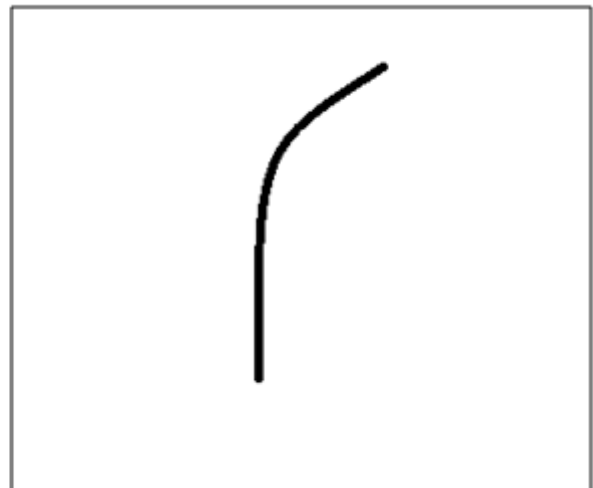
$$M_C = M_1 - 2 \cdot M_{w2}, N_C = N_1 - 2 \cdot N_{w2}. \quad (4.19)$$

При виборі таких розмірів матриць в результаті згортки ядро фільтра не виходить за межі початкового зображення і тому не виникають спотворення.

Вищеописана згортка (4.17) в ЗНМ розглядається як ідентифікатор певної властивості об'єкта на зображенні. В першу чергу це використовується при детектуванні відрізків кривих та фігур простої форми (наприклад, кіл, квадратів та ін.) (рис. 4.16). Тобто, якщо ядро w фільтра накладається на область початкового зображення, розподіл яскравості якої подібний до розподілу яскравості ядра, то результат згортки для такої області буде максимальний. Наприклад, якщо ядро w (рис. 4.16) має вигляд Г-подібної кривої, то таке ядро буде детектором таких кривих на зображенні (рис. 4.16б).

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

а)



б)

Рис. 4.16. Значення ядра фільтра w (а) та крива на зображенні (б), для детектування якої призначений фільтр

Проте форма реальних зображень є дуже складною, тому для ідентифікації (детектування) властивостей об'єктів на зображеннях застосовуються найрізноманітніші ядра фільтрів (рис. 4.17).

Наприклад, для розглянутої множини ядер фільтрів (рис. 4.16) верхня половина ядер використовується для детектування на зображенні смуг певного напрямку, а нижня частина – для детектування локальних областей певного кольору (у випадку оброблення кольорових зображень).

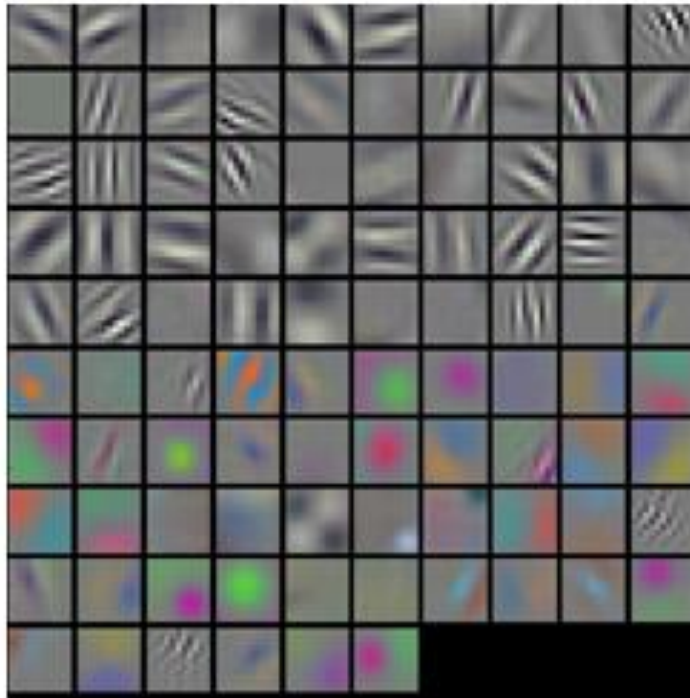


Рис. 4.16. Приклад ядер фільтрів w , які використовуються у ЗНМ

Послідовність оброблення сигналів у класичній ЗНМ описується такою схемою (рис. 4.17), Input – вхідне зображення, Conv – згорткові шари, ReLU – шари підвибірки, Pool – вихід шару, Fully Connected – повнозв’язний шар. В описаній схемі ReLU (rectified linear unit) – блок лінійної ректифікації, який у більшості випадків виконує перетворення $\max(x, 0)$. Використання саме шару ReLU забезпечує високу швидкість навчання ЗНМ.

Input -> Conv -> ReLU -> Conv -> ReLU -> Pool -> ReLU -> Conv -> ReLU -> Pool -> Fully Connected

Рис. 4.16. Послідовність оброблення сигналів у ЗНМ

Принцип роботи ЗНМ, призначеної для розпізнавання облич, полягає в наступному. Перший згортковий шар використовується для детектування простих ознак обличчя, таких як фрагменти контурів

різного напрямку та області зображення з різним кольором та яскравістю. Другий згортковий шар детектує складніші ознаки (ознаки вищого рівня) як поєднання фрагментів контуру (наприклад, кола детектуються як множина дуг). На кожному наступному шарі зростає складність об'єктів, які він здатен розпізнавати; на передостанніх шарах можливе детектування змістовних частин обличчя (очей, рота, підборіддя та ін.), а в останньому шарі детектується саме обличчя. Проте обробка зображень в ЗНМ не повністю збігається з обробкою зображень в системі зору людини, тому як змістовні ознаки обличчя ЗНМ може виділяти не тільки звичні для людини ознаки (око, рот, ніс), але й довільні їх поєднання. Крім цього, чим більший номер шару ЗНМ, тим більшу область зображення він обробляє (більше поле сприйняття).

Шари підвибірки ЗНМ призначені для зменшення розмірності сигналів (звичайно у 2 рази за шириною і висотою). У більшості випадків функціонування шару підвибірки полягає у заміні прямокутної області карти ознак (виходу згорткового шару) на їх максимальне значення (рис. 4.17). Використання шару підвибірки приводить не тільки до прискорення обчислень, але й до того, що неймережа стає більш стійкою (інваріантною) до зміни масштабу чи зсуву об'єктів на зображеннях.

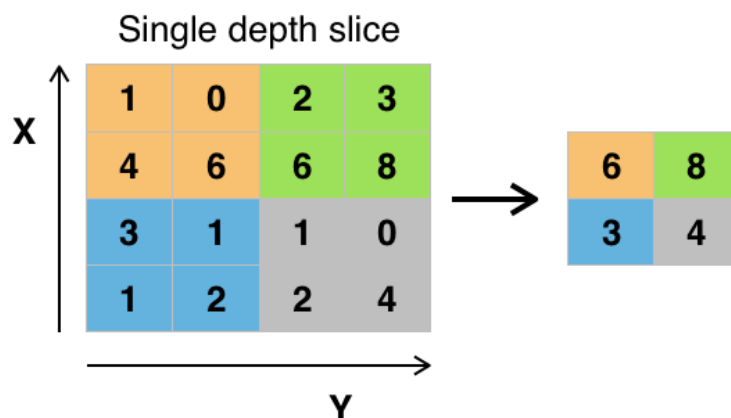


Рис. 4.17. Принцип роботи шару підвибірки з вибором максимального значення для заданих областей

Останнім шаром ЗНМ є повнозв'язний, виходами якого є номери класів образів, тобто номери осіб, обличчя яких розпізнаються (кожен клас, крім номеру, може мати назву – наприклад, ім'я, особи). Виходи ЗНМ нормуються в діапазонів від 0

до 1, тому їх значення можна розглядати як ймовірність того, то вхідне зображення належить до певного класу облич. Наприклад, якщо вихід ЗНМ – $[0.1 \ 0.8 \ 0.1]$, то це означає, що вхідне зображення з ймовірністю 10% належить класу №2, а з ймовірністю 80% – класу №1. Повнозв'язний шар ЗНМ практично такий самий, як і у багатошарових персептронів, а для його навчання застосовується метод зворотного поширення помилки.

Наприклад, ЗНМ, призначену для розпізнавання зображень символів, можна описати такою структурою (рис. 4.18).

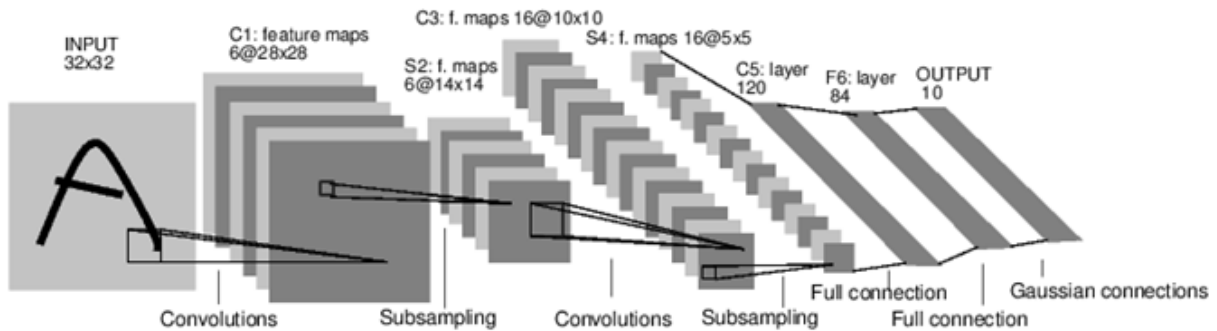


Рис. 4.18. Структура ЗНМ, призначеної для розпізнавання зображень

Завдяки використанню згорткового шару для навчання ЗНМ потрібна відносно мала кількість параметрів. Наприклад, якщо початкове зображення має розмір 100×100 пікселів (у відтінках сірого), а згортковий шар використовує фільтри з ядром 3×3 пікселів з виходом на 6 карт ознак (на 6 каналів), тоді при навчанні нейромережі потрібно налаштувати $9 \times 6 = 54$ параметри, що значно менше, ніж для повнозв'язної ШНМ.

Загальна схема ЗНМ наступна (рис. 4.19). Початкове зображення f обробляється у відтінках сірого (яскравість f нормується до 1). ЗНМ містить $(Q_L - 1)$ згорткових (convolution) шарів і $(Q_L - 1)$ шарів підвибірки (subsampling), а також повнозв'язний шар з номером Q_L (наприклад, $Q_L = 6$). На виході повнозв'язного шару формується вихідний вектор Y і виконується розпізнавання обличчя. Кожен згортковий шар з номером L містить $Q_m(L)$ карт розміром $M_1(L) \times N_1(L)$ елементів, а значення його елементів зберігаються в масиві $c(L, m_p, i_1, k_1)$, де $m_p = 1, \dots, Q_m(L)$, $i_1 = 1, \dots, M_1(L)$, $k_1 = 1, \dots, N_1(L)$. Значення згорткового шару обчислюються в результаті згортки попереднього шару з ядром згортки $w(L)$ розміром $M_w(L) \times N_w(L)$ елементів.

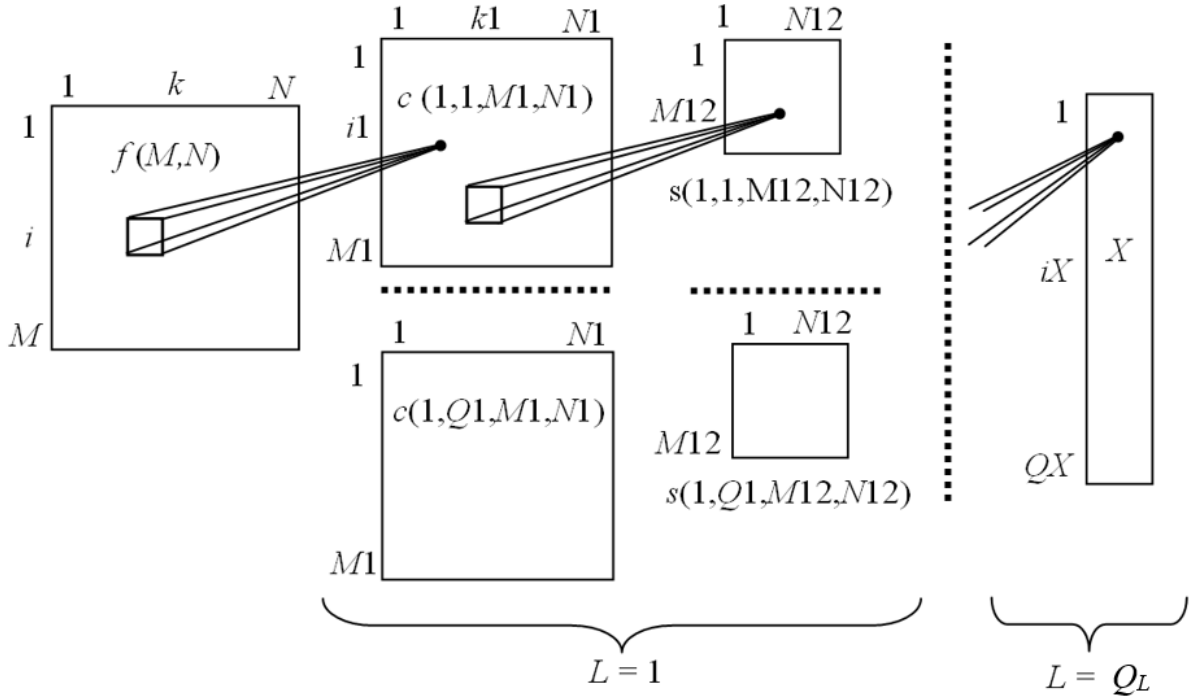


Рис. 4.19. Структура Q_L шарів згорткової нейронної мережі

Шар підвибірки з номером L містить $Q_m(L)$ карт розміром $M_2(L) \times N_2(L)$ елементів, а значення його елементів зберігаються в масиві $s(L, m_p, i_1, k_1)$, де $m_p = 1, \dots, Q_m(L)$, $i_2 = 1, \dots, M_2(L)$, $k_2 = 1, \dots, N_2(L)$. На входи повнозв'язного шару з номером (Q_L) подається вектор $X(i_X)$, де $i_X = 1, \dots, Q_X$.

Навчання повнозв'язного шару ЗНМ виконується методом зворотного поширення помилки на основі навчальної вибірки.

4.14. Багатомасштабний аналіз сигналів за допомогою штучних нейронних мереж

Використання класичного методу навчання ШНМ (зворотного розповсюдження помилки) потребує значного часу навчання, тому для підвищення швидкості навчання розроблено багатомасштабне оброблення вхідних векторів X та алгоритм багатомасштабного навчання ШНМ (рис. 4.20) [24]. Згідно алгоритму всі вхідні вектори навчальної вибірки записуються в масив X_0 , а всі істинні вектори – у масив Y^T . ШНМ навчається послідовно для Q_{SD} масштабів вхідного вектора, а кожному масштабу з номером s_d відповідає масштабний коефіцієнт

$$S_C = 2^{(Q_{SD}-s_d)}, \quad (4.20)$$

відповідно до якого обчислюється розмір вхідного вектора

$$Q_X = [Q_{X0} / S_C], \quad (4.21)$$

де Q_{X0} – максимальний розмір вхідного вектора (для $s_d = Q_{SD}$).

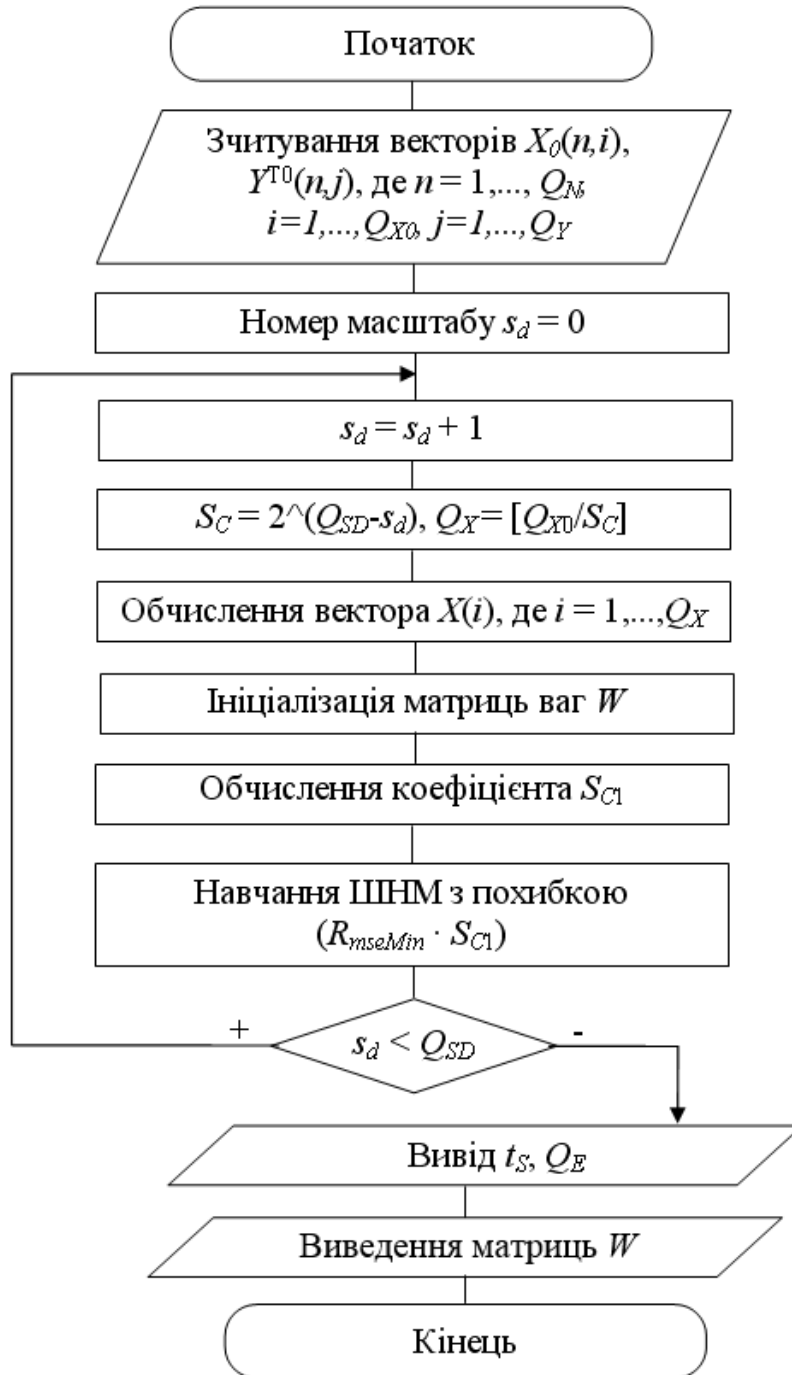


Рис. 4.20. Схема алгоритму багатомасштабного навчання ШНМ

Для кожного масштабу з номером s_d також обчислюються значення додаткового рівня – вхідного вектора X (розміром Q_X елементів) на основі початкового вектора з масиву X_0 (розміром Q_{X0}

елементів) шляхом лінійної інтерполяції. Таким чином, на початку навчання ШНМ ($s_d = 1$) кількість нейронів у шарі X мінімальна (рис. 4.21, а), а кожного наступного масштабу кількість нейронів Q_X подвоюється (використовується динамічне додавання нейронів у вхідний шар X) (рис. 4.21, б).

Завдяки цьому для масштабу $s_d = 1$ одному нейрону вхідного шару відповідає S_C точок вхідного сигналу (їх середнє значення), а для масштабу Q_{SD} одному нейрону вхідного шару відповідає одна точка вхідного сигналу (рис. 4.22). Завдяки вищеписаному масштабуванню вхідних векторів виконується спочатку грубе налаштування ваг нейромережі, а потім – точне налаштування ваг, що потенційно дозволяє підвищити швидкість навчання ШНМ.

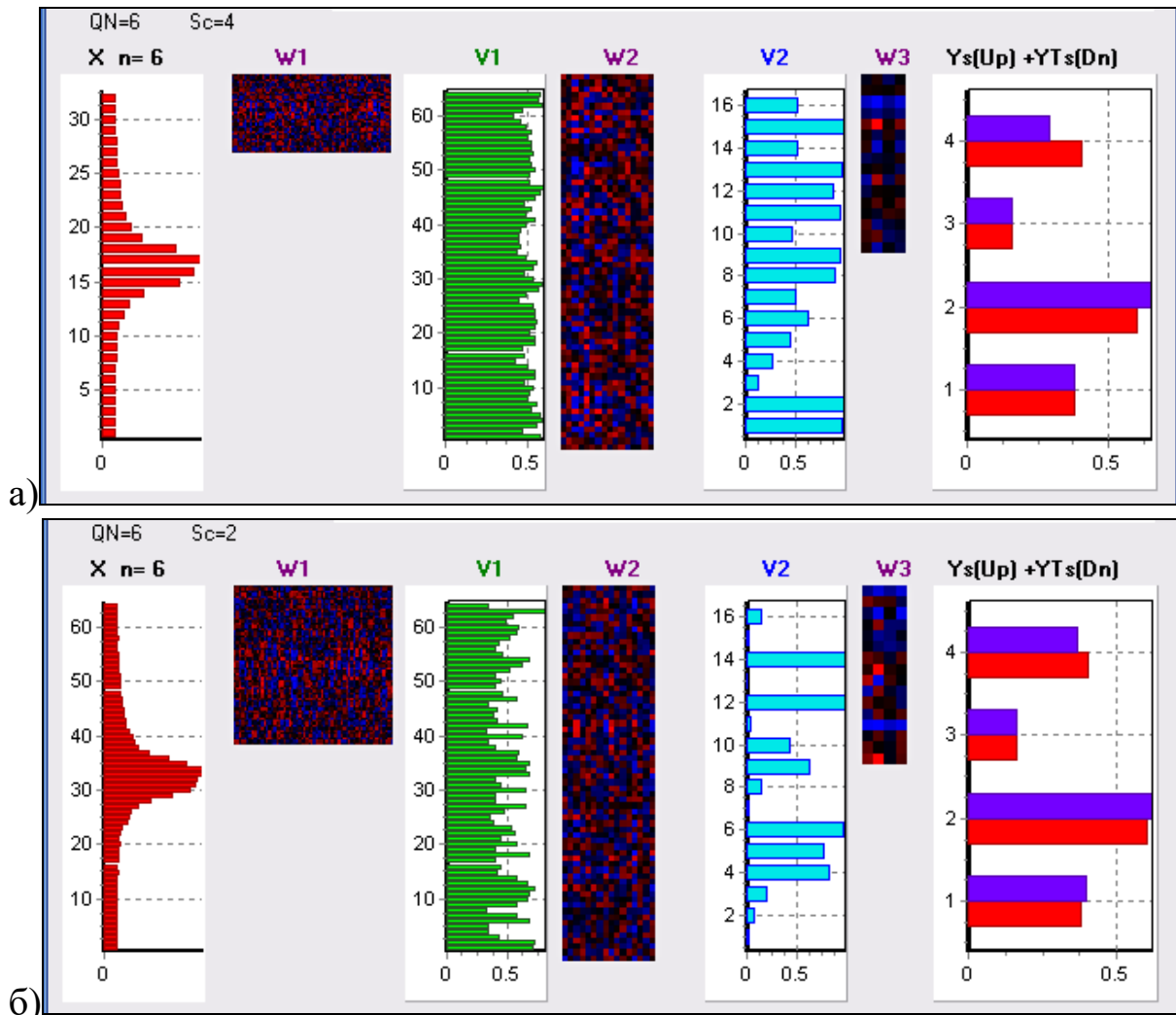


Рис. 4.21. Фрагменти головної форми програми моделювання ШНМ з результатами навчання при багатомасштабному обробленні вхідних векторів X : а) масштаб $(1/S_C) = 1/4$, $Q_X = 32$; б) масштаб $(1/S_C) = 1/2$, $Q_X = 64$

Відповідно до зміни розміру вектора X змінюється розмір матриці ваг W^1 для першого шару, а її ініціалізації виконується так. Для номеру масштабу $s_d = 1$ початкова матриця ваг заповнюється випадковими значеннями, а для наступних масштабів ($s_d > 1$) початкові ваги W^1 обчислюються через налаштовані ваги для попереднього масштабу за формулою

$$W_{i,k1}^{1(s_d)} = W_{i1,k1}^{1(s_d-1)} \cdot k_W, \quad (4.22)$$

де $i = 1, \dots, Q_X$; $k_1 = 1, \dots, Q_{V1}$; $i_1 = [i/2]$;

Q_X – кількість нейронів шару X для номеру масштабу s_d ;

k_W – коефіцієнт масштабу вагових коефіцієнтів (наприклад, $k_W = 1.5$).

Після вищеприписаного обчислення початкових ваг (для $s_d > 1$) до матриці W^1 додаються малі випадкові значення, проте у випадку багатомасштабного навчання ШНМ максимальна швидкість навчання забезпечується при $\Delta_W = 0.7$ (рис. 4.23).

У запропонованому багатомасштабному алгоритмі навчання ШНМ змінюється розмір вхідного вектора X і відповідно матриці вагових коефіцієнтів W^1 , однак можлива зміна також кількості нейронів у прихованих шарах V^1 та V^2 , та, відповідно, і розмірів матриць вагових коефіцієнтів W^1 та W^2 . Оскільки при переході до наступного масштабу вектора X його розмір подвоюється, то розміри векторів ШНМ зручно вибирати кратними степеню 2.

Наближене значення коефіцієнту k_W можна вибрати з умови, що при зміні масштабу вхідного вектора виходи суматорів нейронів першого шару не повинні змінитися, тобто $NET_{k1}^{s_d} = NET_{k1}^{s_d-1}$, де $k_1 = 1, \dots, Q_{V1}$. Згідно з (4.22) виходи суматорів нейронів дорівнюють:

$$NET_{k1}^{s_d} = \sum_{i=1}^{Q_X} X_i \cdot W_{i,k1}^{1(s_d)} = k_W \sum_{i=1}^{Q_X} X_i \cdot W_{i1,k1}^{1(s_d-1)}, \quad (4.23)$$

$$NET_{k1}^{s_d-1} = \sum_{i2=1}^{Q_{X1}} X_{i2}^{s_d-1} \cdot W_{i2,k1}^{1(s_d-1)} = \sum_{i2=1}^{Q_{X1}} \frac{X_{2 \cdot i2-1} + X_{2 \cdot i2}}{2} \cdot W_{i2,k1}^{1(s_d-1)}, \quad (4.24)$$

$$NET_{k1}^{s_d-1} = \frac{1}{2} \sum_{i=1}^{Q_X} X_i \cdot W_{i1,k1}^{1(s_d-1)}, \quad (4.25)$$

де $k_1 = 1, \dots, Q_{V1}$; $i_1 = [i/2]$; $Q_{X1} = [Q_X/2]$.

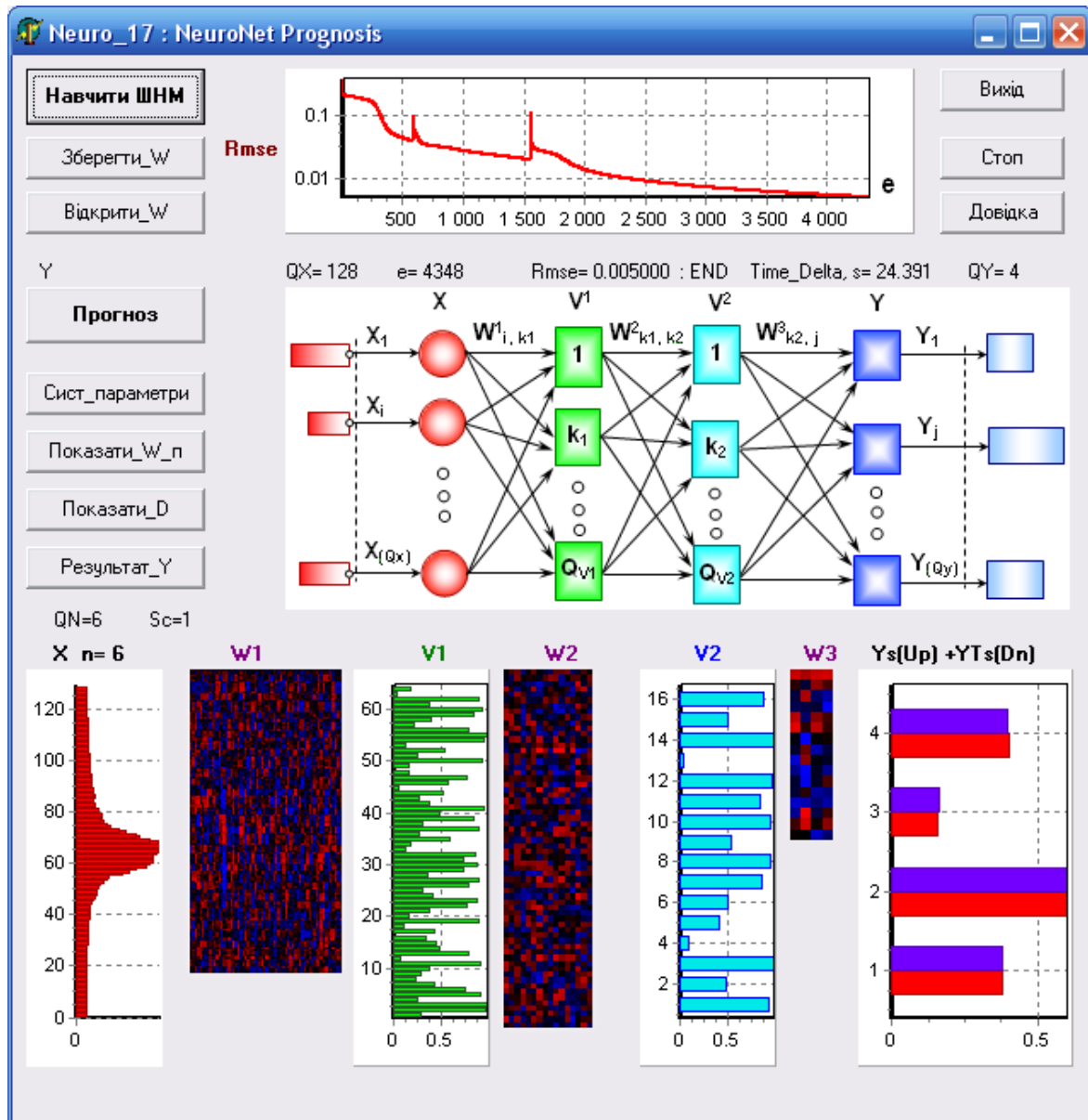


Рисунок 4.22. Головна форма програми моделювання ШНМ з результатами навчання при багатомасштабному обробленні вхідних векторів X ; масштаб $(1/S_C) = 1$, $Q_X = 128$

З умови рівності виразів (4.23) та (4.25) отримаємо, що значення коефіцієнту $k_W = 0.5$, проте таке k_W забезпечує малі значення векторів різниці D (на виходах шарів ШНМ) при зміні масштабу вхідного вектора, що приводить до повільної зміни ваг нейромережі. Тому з метою мінімізації часу навчання ШНМ вибрано значення коефіцієнту $k_W = 1.5$, завдяки чому при зміні масштабу X значення векторів різниці D збільшуються (збільшенню D відповідають характерні піки для похибки навчання R_{mse} – рис. 4.22), після чого швидкість навчання нейромережі значно зростає.

Отже, для початкових номерів масштабу s_d нейромережа навчається на сигналах спрощеної форми (з меншою кількістю точок Q_x) з відносно великою похибкою, а на останньому рівні s_d використовуються сигнали максимальної розмірності і досягається мінімальна похибка навчання. Такий принцип оброблення інформації, коли спочатку використовується спрощена й узагальнена модель об'єкта, яка поступово ускладнюється і деталізується, особливо ефективний при навчанні ШНМ з великими розмірами навчальних множин. Варто зауважити, що подібний принцип поступової деталізації використовується в зоровій системі людини та в деяких системах технічного зору [13-14].

Контрольні запитання до розділу 3

1. Що таке штучний нейрон?
2. Опишіть спрощену будову та принцип дії біологічного нейрона.
3. У чому полягає роль вагових коефіцієнтів?
4. Перерахуйте та опишіть основні активаційні функції нейронів, наведіть відповідні формули і графіки.
5. Чому як активаційна часто використовується сигмоїдна функція?
6. Зобразіть структуру одношарової нейромережі, поясніть принцип її функціонування.
7. Що таке „персептрон”?
8. У чому полягає головний недолік персептрона?
9. Зобразіть структуру багатошарової нейромережі, поясніть принцип її функціонування.
10. Чому для ефективної роботи багатошарової нейромережі активаційна функція повинна бути нелінійною?
11. Опишіть основні способи навчання нейромереж, поясніть їх недоліки та переваги.
12. У чому полягає концепція Хебба, яка використовується при навчанні НМ?
13. Опишіть алгоритм навчання нейромережі зі зворотним розповсюдженням помилки.
14. Як формуються матриці вагових коефіцієнтів НМ?

15. Для чого потрібно проводити нормалізацію значень вхідних і вихідних векторів НМ?
16. Опишіть математичну модель прямого розповсюдження сигналу на прикладі 3-шарової нейромережі.
17. Як математично описуються помилки навчання нейромережі?
18. Опишіть математичну модель зворотного розповсюдження помилки на прикладі 3-шарової нейромережі.
19. Чому нейромережі ефективні при розпізнаванні зображень?

5. ГЕНЕТИЧНІ АЛГОРИТМИ

5.1. Основи генетичних алгоритмів

Генетичні алгоритми (*Genetic Algorithms*) є складовою еволюційних методів [3-5, 26], оскільки виникли у результаті спроб копіювання еволюції живих організмів. Генетичні алгоритми (ГА) – це процедури пошуку, засновані на механізмах природного відбору і спадкування; в ГА використовується еволюційний принцип виживання найбільш пристосованих особин. Генетичні алгоритми мають певні переваги у порівнянні з традиційними методами оптимізації, оскільки поєднують кращі властивості градієнтних методів та методів повного перебору. Порівняно з градієнтними методами ГА забезпечують кращий результат, а порівняно з методами повного перебору ГА мають значно вищу швидкодію. Тому ГА широко використовуються в системах штучного інтелекту при вирішенні різноманітних оптимізаційних задач.

За допомогою генетичних алгоритмів задача вирішується наступним чином. Уявімо собі штучний світ, населений кількістю N істот (особин), причому генетичний код кожної істоти – це деякий розв’язок нашої задачі. Разом всі особини утворюють популяцію P_0 . Для простоти вважається, що генетичний код кожної особини (генотип) записується у вигляді однієї хромосоми (*Chromosome*) X (для людини генотип містить 46 хромосом). В ГА хромосома X особини є впорядкованою послідовністю з M генів, тобто хромосома $X = \{G_1, G_2, \dots, G_M\}$ – числовий вектор, який описує рішення задачі. Генотип всіх особин утворює популяцію хромосом $P = \{X_1, X_2, \dots, X_N\}$.

В класичному ГА кожний ген записується у вигляді набору R бітів, тому вся хромосома записується набором двійкових чисел (алелів) довжиною $L = M \cdot R$ (подібно до молекули ДНК). Кожен ген кодує певну властивість організму (в людини більше 60 тис. генів).

Особина вважається тим більш пристосованою, чим краще рішення задачі вона забезпечує (більше значення цільової функції). В ГА цільова функція також називається функцією пристосованості або фітнес-функцією (*fitness function*) F . Тоді задача оптимізації цільової функції зводиться до пошуку найбільш пристосованої особини. У віртуальному світі ГА буде існувати N істот (особин), покоління яких змінюють одне одного. Тепер, якщо ввести в дію природний відбір і генетичне наслідування, то одержаний світ підкорятиметься законам еволюції.

Розглянемо принципи роботи генетичних алгоритмів на прикладі класичної задачі комівояжера (*TSP - travelling salesman problem*). Задача формулюється так: комівояжеру вимагається об'їхати M міст, побувавши в кожному один раз, і повернутися в початкову точку. Цільовою функцією є довжина шляху S , яка повинна мінімізуватися. Виявляється, що вже для $M = 30$ міст пошук оптимального шляху традиційними методами (наприклад, повного перебору) є складною задачею (потрібно перебрати $30! \approx 10^{32}$ маршрутів, що нереально). Тому це спонукало розвиток нових методів оптимізації, зокрема генетичних алгоритмів.

У випадку задачі комівояжера кожний маршрут (фенотип) описується рядком з M десяткових чисел, тобто хромосома X особини є впорядкованою послідовністю з M генів, де на j -му місці стоїть номер j -го по порядку обходу міста. Якщо на початку рішення задачі комівояжера ми маємо випадковий набір маршрутів (особин), то в процесі еволюції виживуть тільки маршрути з мінімальною довжиною (кращим набором генів) (рис. 5.1).

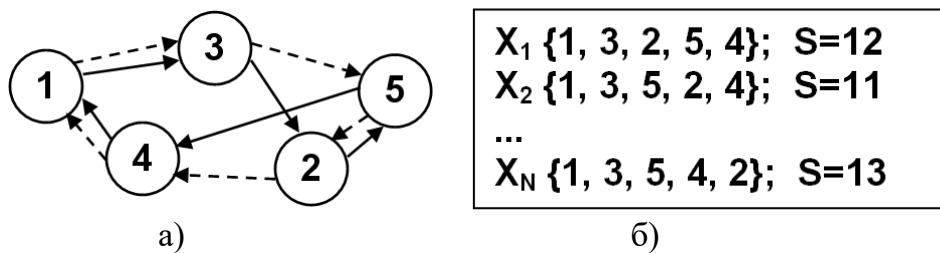


Рис. 5.1. Вирішення задачі комівояжера за допомогою ГА: а) фенотип (маршрути); б) генотип (набір хромосом популяції); S – довжина шляху

В ГА часто використовується стратегія елітизму, яка полягає у збереженні кращого з індивідумів поточного покоління.

Основні відмінності ГА від традиційних задач оптимізації такі:

- 1) ГА обробляють не значення параметрів самої задачі, а їх закодовану форму;
- 2) виконують пошук рішення виходячи не з однієї точки, а з деякої популяції;
- 3) використовують тільки цільову функцію, а не її похідні або іншу додаткову інформацію;
- 4) використовують ймовірнісні, а не детерміновані правила відбору.

5.2. Класичний генетичний алгоритм

Класичний ГА складається з наступних кроків (рис. 5.2):

1. Ініціалізація, або вибір початкової популяції хромосом.
2. Оцінювання пристосованості хромосом в популяції.
3. Перевірка умови закінчення алгоритму.
4. Селекція хромосом.
5. Використання генетичних операторів.
6. Формування нової популяції.
7. Вибір найкращої хромосоми.

Розглянемо ГА більш детально:

1. *Ініціалізація* – створення початкової популяції з N хромосом (початкової множини рішень). У даній роботі для ініціалізації використано стратегію «дробовика», яка полягає у генерації досить великої випадкової підмножини рішень.

2. *Оцінювання пристосованості хромосом* у популяції полягає у розрахунку функції пристосованості для кожної хромосоми $F(X_i)$. Приймається, що функція пристосованості завжди набуває невід’ємні значення. Звичайно знаходиться максимум цієї функції.

3. *Перевірка умови закінчення алгоритму*. Алгоритм може бути закінчений, якщо його виконання не приводить до поліпшення отриманого результату, або через певний проміжок часу.

4. *Селекція хромосом* полягає у виборі на основі функції пристосованості тих хромосом, які будуть брати участь у створення нащадків, тобто нового покоління. Такий вибір виконується згідно з принципом природного відбору, за яким найбільші шанси на створення потомства мають хромосоми з найвищими значеннями функції пристосованості. До найбільш поширених методів селекції відноситься метод рулетки, ранговий та турнірний методи.

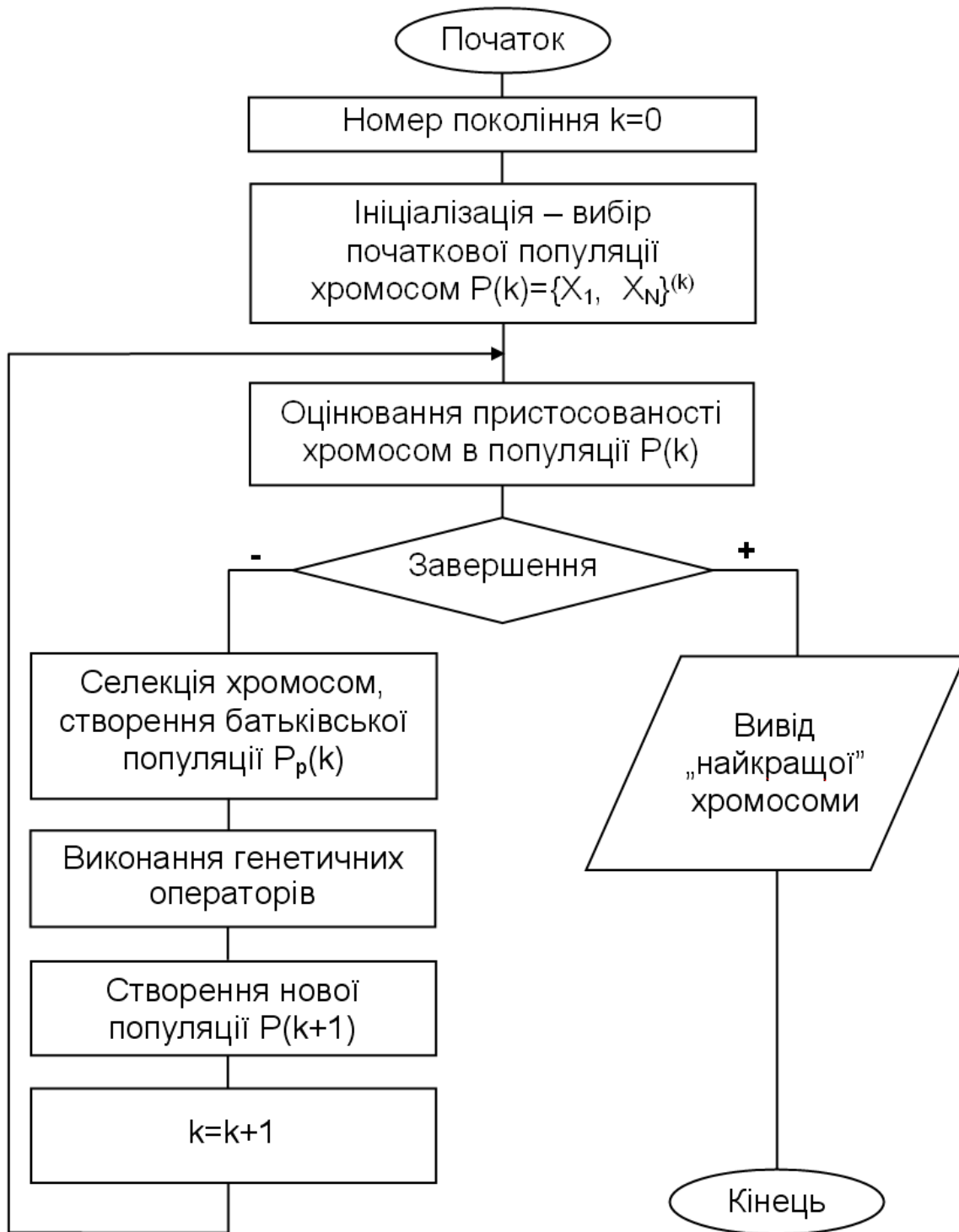


Рис. 5.2. Схема класичного генетичного алгоритму

В *методі рулетки* кожній хромосомі X_i ставиться у відповідність сектор колеса рулетки, величина якого пропорційна до функції пристосованості F даної хромосоми. Ймовірність селекції хромосоми X_i , де $i = 1 \dots N$, дорівнює:

$$p_S(x_i) = F(x_i) / \sum_{j=1}^N F(x_j). \quad (5.1)$$

Отже, ймовірність селекції в методі рулетки найбільша для хромосом із великим значенням функції пристосованості. У результаті селекції формується батьківська популяція з чисельністю N , в яку особини з великим значенням $p_S(X_i)$ можуть увійти кілька разів, а з малим значенням $p_S(X_i)$ – жодного.

При *ранговій селекції* особини популяції впорядковуються за значенням їх функції пристосованості F (за спаданням), де кожній хромосомі X ставиться у відповідність її номер i у списку (ранг). Ймовірність вибору хромосоми у батьківську популяцію в такому випадку визначається формулою

$$p_S(x_i) = \begin{cases} 1/\mu, & 1 \leq i \leq \mu \\ 0, & \mu < i \leq N \end{cases}, \quad (5.2)$$

де $\mu \leq N$ – параметр методу, в найпростішому випадку $\mu = N/2$.

При *турнірному відборі* з популяції, яка містить N хромосом, випадковим чином вибирається t хромосом, краща з яких (переможець) записується у масив батьківських хромосом. Розмір туру $2 \leq t < N$ звичайно вибирається рівним 2 (парний турнір). Далі цей процес повторюється, поки кількість батьківських хромосом не стане рівною N .

5. *Використання генетичних операторів* до відібраних у результаті селекції батьківських хромосом приводить до створення популяції нащадків. У класичному ГА використовують два основних генетичних оператори: оператор схрещування (*crossover*) і оператор мутації (*mutation*). Як і в живих організмів, імовірність мутації звичайно дуже мала ($p_m < 0.1$). В ГА мутація може виконуватися на популяції батьків перед схрещуванням або на популяції нащадків, утворених у результаті схрещування.

Оператор схрещування. На першому етапі схрещування вибираються пари хромосом із батьківської популяції. Операції схрещування полягають в обміні фрагментами ланцюжків між двома батьківськими хромосомами. Далі для кожної пари вибирається позиція гена (локус) у хромосомі, який визначає точку схрещування. Якщо хромосома містить L двійкових чисел, то точка схрещування L_K вибирається випадковим чином з інтервалу $[1, L]$. В результаті схрещування утворюються два нащадки:

1) нащадок, хромосома якого на позиціях від 1 до L_K складається з генів першого предка, а позиції від $L_K + 1$ до L – з генів другого предка;

2) нащадок, хромосома якого на позиціях від 1 до L_K складається з генів другого предка, а позиції від $L_K + 1$ до L – з генів першого предка (рис. 5.3).

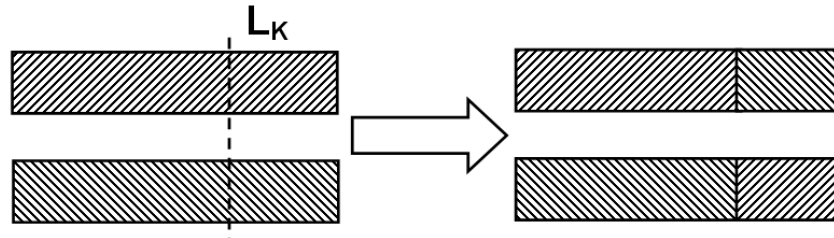


Рис. 5.3. Умовна схема схрещування

Наприклад, якщо хромосоми (1, 2, 3, 4, 5) і (0, 0, 0, 0, 0) розрізати між третім і четвертим генами і обміняти їх частини, то утворяться нащадки (1, 2, 3, 0, 0) і (0, 0, 0, 4, 5).

Оператор *мутації* з ймовірністю p_m змінює значення гена на певну числову величину (амплітуду мутації A_m).

6. *Формування нової популяції*. Хромосоми, отримані у результаті дії генетичних операторів на популяцію предків (покоління k), утворюють нову популяцію (покоління $k+1$).

7. *Вибір найкращої хромосоми*. Найкращою вважається хромосома з максимальним значенням функції пристосованості.

5.3. Програмна реалізація генетичних алгоритмів

Розглянемо програмну реалізацію генетичних алгоритмів на прикладі програми „Gen_11_4”, яка призначена для моделювання еволюції форми дерева за допомогою ГА (рис. 5.2).

На головній формі програми (рис. 5.4) моделюється процес еволюції форми дерева та встановлюються його параметри. Еволюція починається при натисненні кнопки „Старт”, а зупиняється кнопкою „Стоп”. В процесі еволюції на формі програми відображаються: номер покоління хромосом k , номер хромосоми i та номер найкращої хромосоми в популяції i_Max . На формі показується найкраще значення функції пристосованості (або фітнес-функції) F для першого покоління F_0 та результуюче значення F .

В процесі еволюції можна змінювати режим відображення даних на формі програми за допомогою перемикачів області „Показати”. Значення коефіцієнтів форми дерева можна показати, якщо встановити перемикач „Коефіцієнти”. Ввімкнений перемикач „Еліту” відображає найкращу хромосому в кожній популяції, а перемикач „Еволюцію” показує всі хромосоми. Перемикачі „k” та „i” показують стан програми для кожного покоління k і номеру хромосоми i .

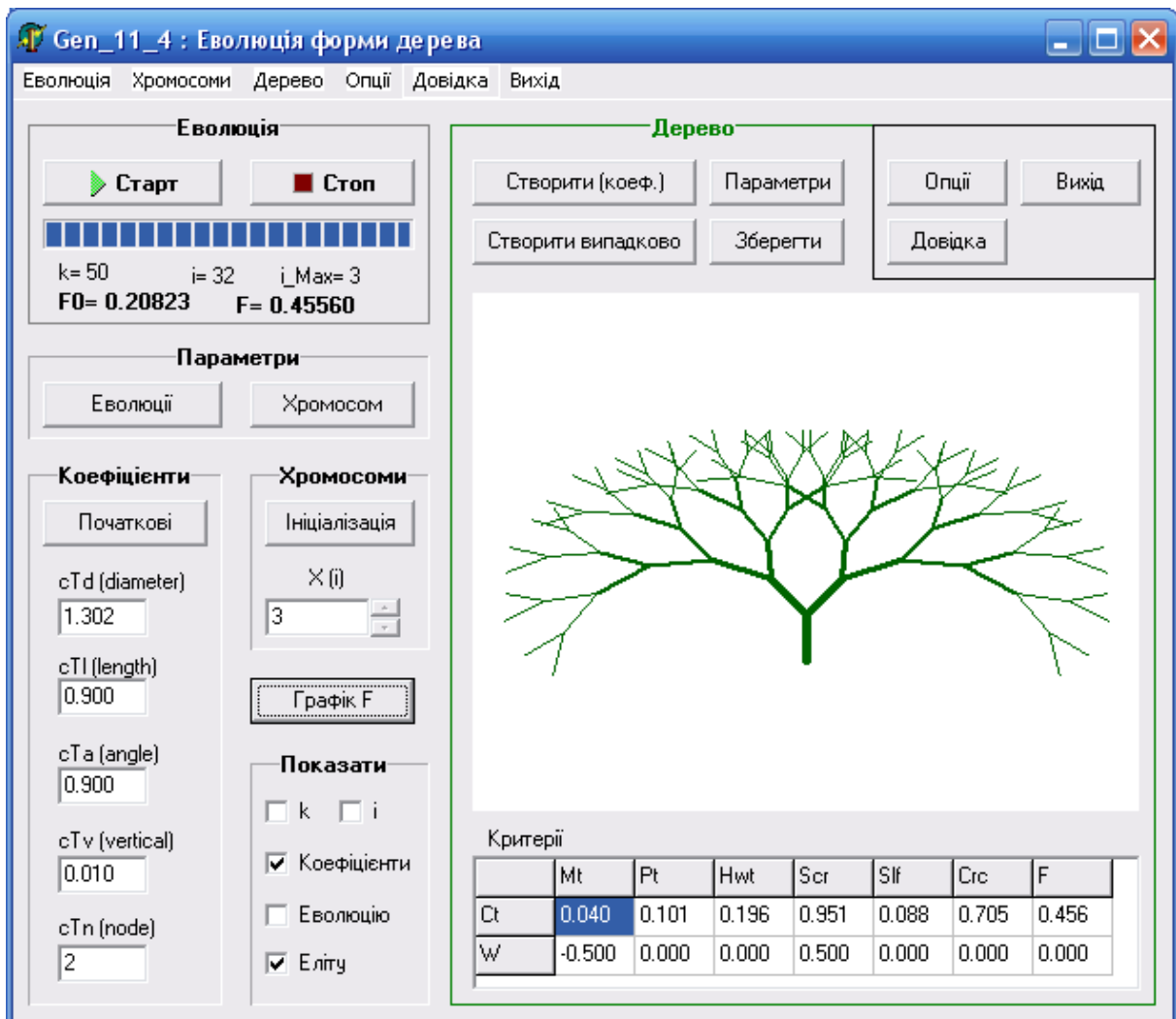


Рис. 5.4. Головна форма програми „Gen_11_4”: результат еволюції

У програмі використано модифіковані генетичні алгоритми, які в якості значень генів використовують дійсні числа. Для даної задачі моделювання еволюції форми дерева вибрано наступний формат хромосоми: $X_i (G_1, G_j, \dots, G_M)$, де $M = 5$, а гени G_j описують коефіцієнти форми дерева (cTd , cTl , cTa , cTv , cTn) (рис. 5.5a). Тобто форма дерева розраховується на основі наступних коефіцієнтів:

1. cTd – коефіцієнт діаметра гілок (*Coefficient Tree Diameter*), який дорівнює відносному збільшенню діаметра гілок на попередньому ярусі дерева відносно наступного (рис. 5.5б,в); стовбур дерева вважається першим ярусом, а листя – останнім.
2. cTl – коефіцієнт довжини гілок (*Coefficient Tree Length*), який дорівнює відносному зменшенню довжини гілок на наступному ярусі дерева відносно попереднього (рис. 5.5г,д).
3. cTa – коефіцієнт кута гілок (*Coefficient Tree Angle*), який дорівнює відносному зменшенню кута між розгалуженнями гілок на наступному ярусі дерева відносно попереднього (рис. 5.5е,є).
4. cTv – коефіцієнт вертикальності гілок (*Coefficient Tree Vertical*); при значенні коефіцієнта 0.5 довжина гілки не залежить від її нахилу; при $cTv < 0.5$ довжина гілки тим більша, чим більше відхилення гілки від вертикалі; при $cTv > 0.5$ довжина гілки тим більша, чим менше відхилення гілки від вертикалі (рис. 5.5ж,з).
5. cTn – коефіцієнт розгалуження (*Coefficient Tree Node* дорівнює кількості гілок, на які розгалужуються гілка попереднього рівня, коефіцієнт приймає значення 2 або 3 (рис. 5.5а,и).

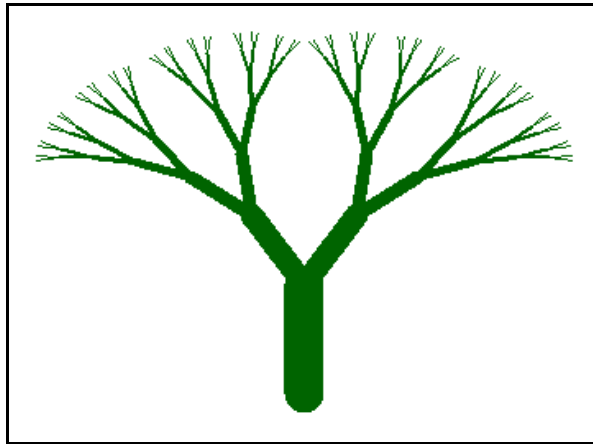
Оскільки хромосома складається з коефіцієнтів форми дерева, то формат хромосоми $X_i(cTd, cTl, cTa, cTv, cTn)$. Значення коефіцієнтів форми дерева за замовчуванням встановлюються кнопкою «Початкові».

Ініціалізація значень хромосом виконується кнопкою „Ініціалізація”, після чого значення генів можна переглядати, встановивши номер хромосоми X_i . Ініціалізація означає створення популяції хромосом $P = \{X_1, X_i, \dots, X_N\}$, які описують форму дерева.

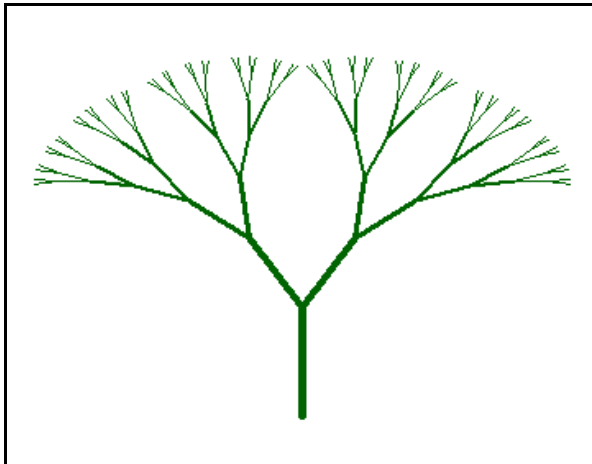
Створення і збереження дерева, перегляд його параметрів виконується кнопками в області „Дерево”. Кнопка „Створити (коэф.)” створює дерево на основі коефіцієнтів, які зчитуються з форми. Кнопка „Створити випадково” створює дерево на основі коефіцієнтів, які вибираються випадково з допустимого діапазону для генів хромосом. Кнопка „Параметри” відкриває форму з параметрами дерева, де встановлюється максимальна кількість ярусів дерева qL ($qL \leq 8$). Кнопка „Зберегти” зберігає зображення дерева у файл.

На основі форми дерева обчислюються значення 6-ти критеріїв St , які служать для розрахунку функції пристосованості F . При цьому

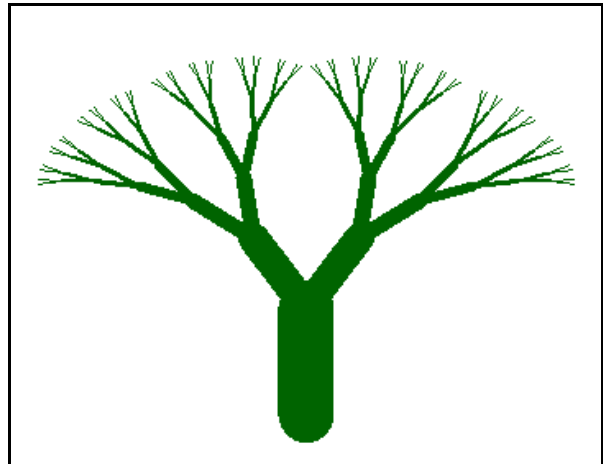
кожний критерій має свою вагу W , яку можна встановлювати в таблиці на формі (рис. 5.4).



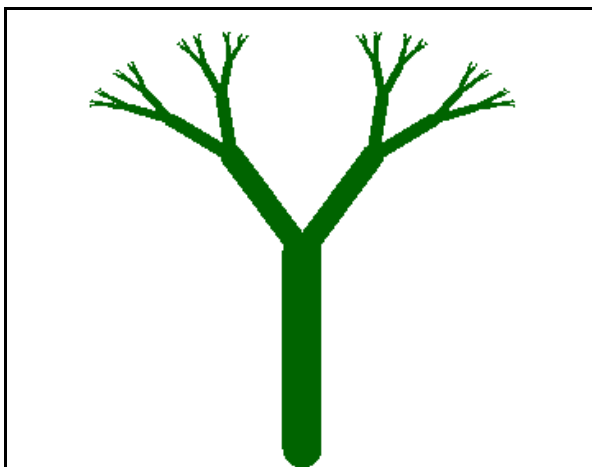
а) X_1 (1.70, 0.75, 0.75, 0.50, 2) – коефіцієнти форми дерева за замовчуванням



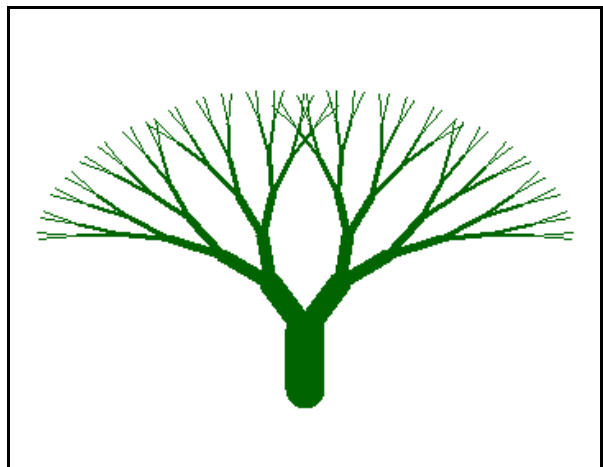
б) X_2 (1.30, 0.75, 0.75, 0.50, 2)



в) X_3 (1.80, 0.75, 0.75, 0.50, 2)

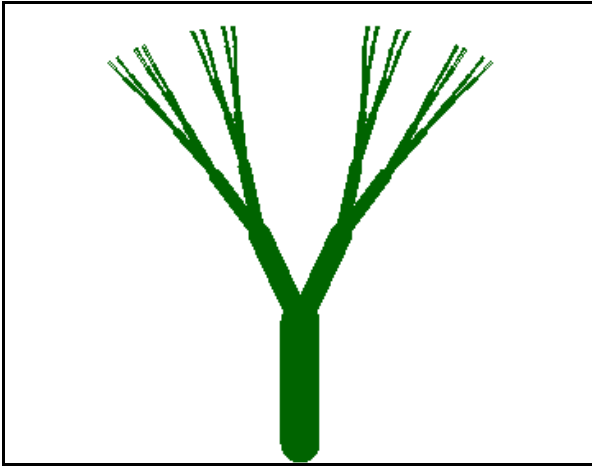


г) X_4 (1.70, 0.55, 0.75, 0.50, 2)

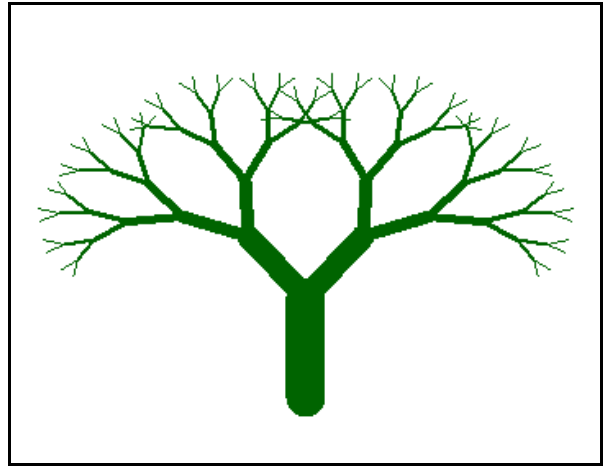


д) X_5 (1.70, 0.90, 0.75, 0.50, 2)

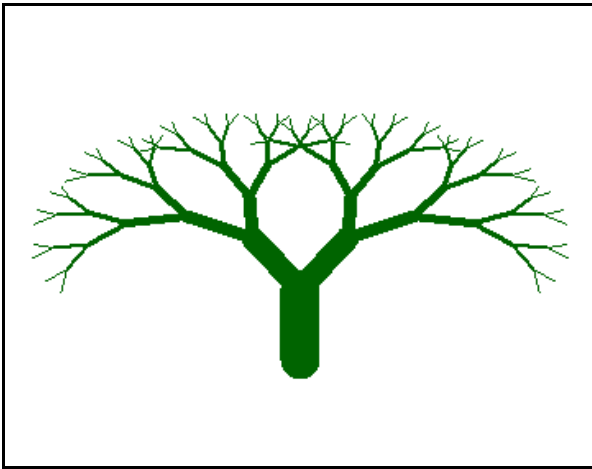
Рис. 5.5. Залежність форми дерева (фенотипу) від значень генів хромосоми (генотипу) X_i (сTд, сTl, сТа, сTv, сTn)



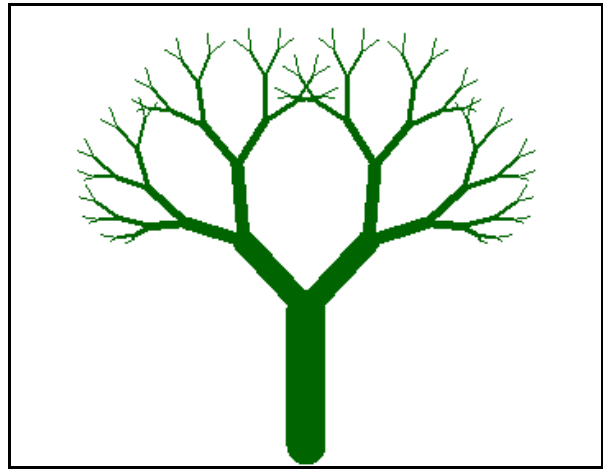
е) $X_6 (1.70, 0.75, 0.50, 0.50, 2)$



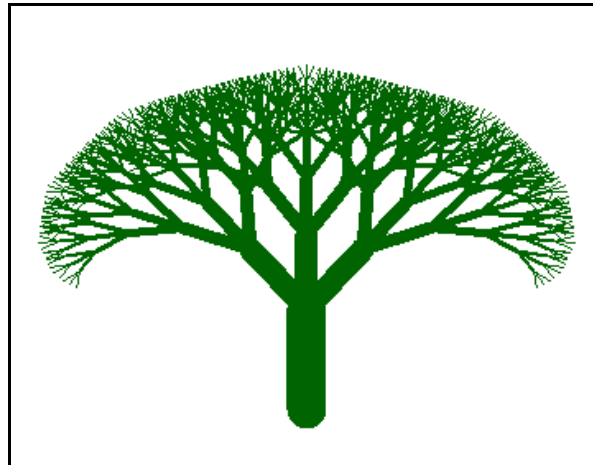
е) $X_7 (1.70, 0.75, 0.90, 0.50, 2)$



ж) $X_8 (1.70, 0.75, 0.90, 0.01, 2)$



з) $X_9 (1.70, 0.75, 0.90, 0.99, 2)$



и) $X_{10} (1.70, 0.75, 0.90, 0.50, 3)$

Продовження рис. 5.5.

Характеристики форми дерева описуються наступними критеріями:

1. Mt – маса дерева (*Mass of tree*).
2. Pt – максимальний тиск на гілки дерева (*Pressure of tree /Max/*).
3. Hwt – відношення висоти дерева до його ширини (*Height/ width of tree*).
4. Scr – площа поверхні крони дерева (*Surface of crown of tree*).
5. Slf – площа освітленої поверхні листя, тобто гілок останнього ярусу (*Surface of leaf of tree*).
6. Crc – відношення центру ваги крони до висоти дерева (*Crown center of tree*).

На формі „Параметри еволюції” для кожного гена встановлюється нижня та верхня межі (рис. 5.6).

З форми „Параметри хромосом” зчитуються параметри ГА: кількість поколінь Q_k , кількість хромосом N ($N \leq 256$), максимальне значення фітнес-функції F_Max , ймовірність мутації P_M , амплітуда мутації A_M (рис. 5.7, рис. 5.8). Значення генів хромосоми відображаються у табличному вигляді, а також як зображення $Image_X$.

Parameter	Min Value	Max Value
☒ cTd	1.3	1.8
☒ cTl	0.5	0.9
☒ cTa	0.5	0.9
☒ cTv	0.01	0.99
☒ cTn	2	3

Рис. 5.6. Форма „Параметри еволюції”: мінімальні та максимальні значення генів

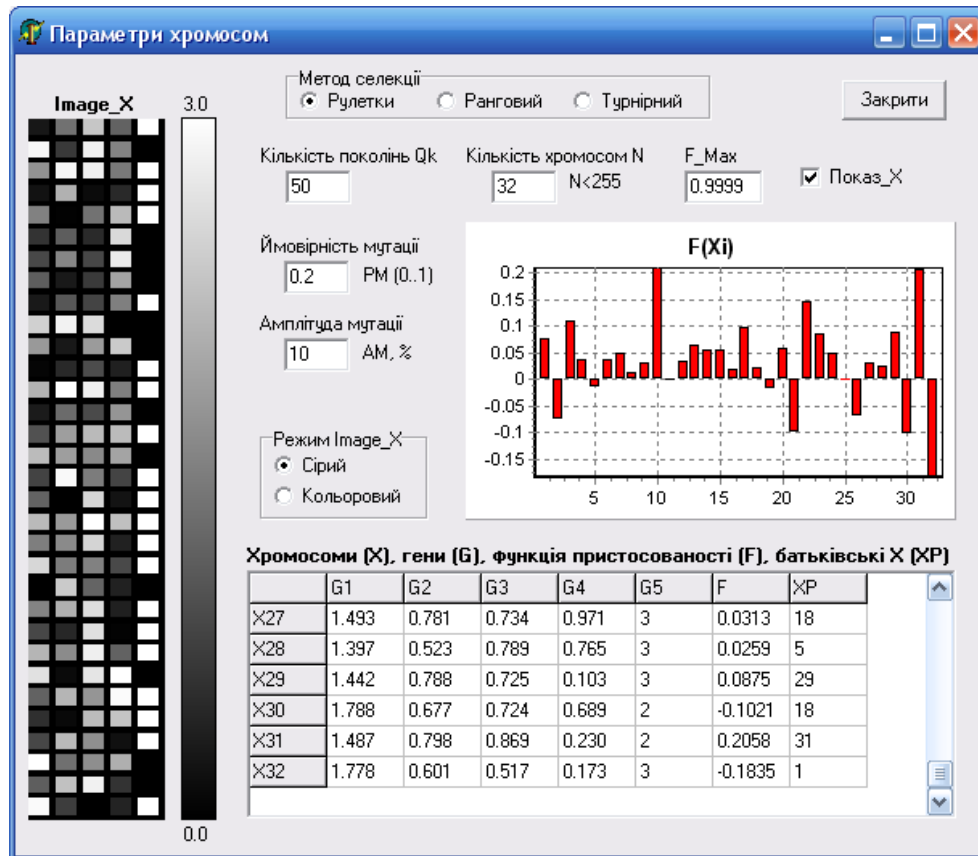


Рис. 5.7. Форма „Параметри хромосом”: стан хромосом після ініціалізації

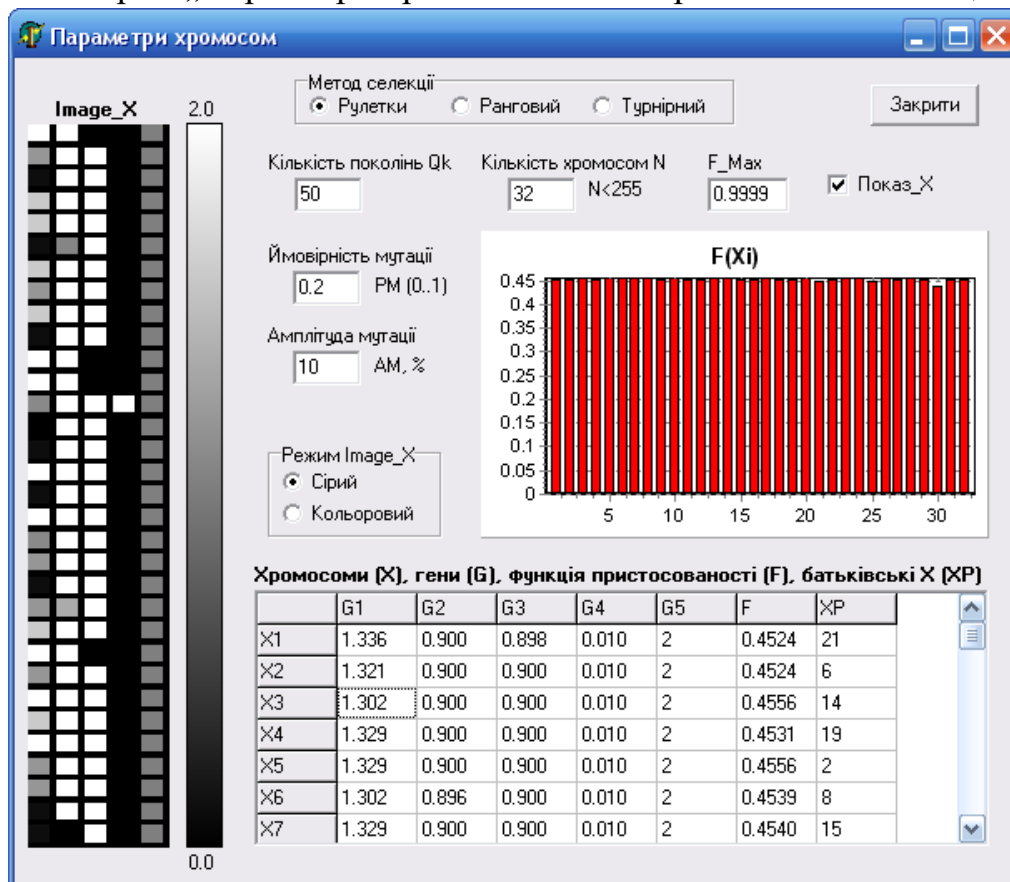


Рис. 5.8. Форма „Параметри хромосом”: стан хромосом після завершення еволюції

Після встановлення ваг W критеріїв, які описують форму дерева, розпочинається процес еволюції, тобто оптимізації форми дерева згідно з вибраними критеріями St . В процесі еволюції відбираються кращі форми дерев, тому функція пристосованості хромосом F зростає протягом певної кількості поколінь k (рис. 5.9). В результаті отримується хромосома з максимальним значенням F , яка відповідає формі дерева, близькій до оптимальної (рис. 1.4, рис. 5.8).

На формі „*Onції*” встановлюється мова інтерфейсу, а також час затримки $Time_Sleep_Tree$ (в мс) при відображенні дерева на формі в процесі еволюції. За рахунок часу затримки можна візуально спостерігати еволюцію форми дерева.

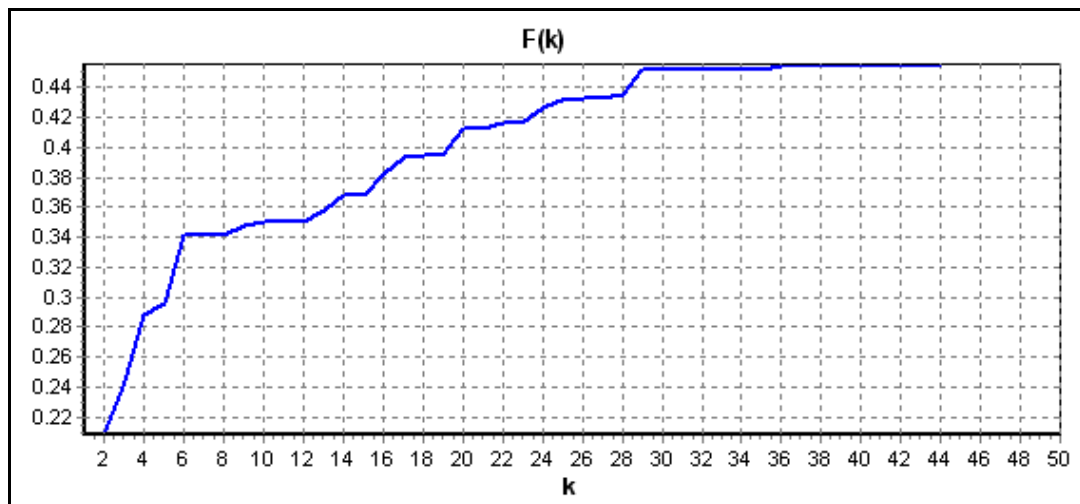


Рис. 5.9. Фрагмент форми „Графік $F(k)$ ”: функція пристосованості F зростає

5.4. Застосування генетичних алгоритмів

Розглянемо використання ГА на прикладі задачі комівояжера для 20 міст. Як особини (індивідууми) розглядатимемо маршрути обходу. Інформацію про маршрут можна записати у вигляді однієї хромосоми - вектора довжини 20, де в першій позиції стоїть номер першого міста на шляху проходження, потім – номер другого міста і т.д. Перше утруднення виникає, коли ми намагаємося визначити мутації для таких хромосом – стандартна операція, що змінює тільки одну позицію вектора, недопустима, оскільки приводить до некоректного маршруту. Але можна визначити мутацію як перестановку значень двох випадково вибраних генів. При такому перетворенні шлях проходження міняється тільки в двох містах.

По умові задачі в даних хромосомах кожен ген (номер міста) повинен зустрічатися тільки один раз. Такий різновид хромосом називається хромосоми "з унікальними генами" і часто використовується в комбінаторних задачах. Стандартна операція схрещування для цього типу хромосом знову ж таки некоректна, тому тут використовується складніша схема двоточкового схрещування.

В процесі еволюції довжина найкоротшого маршруту в популяції монотонно убиває до деякого моменту, а потім стабілізується. Знайдений маршрут, ймовірно, не є найоптимальнішим, але близький до нього по довжині - як правило, генетичні алгоритми "помиляються" не більше ніж на 5-10%. Цей недолік компенсується для комбінаторних задач відносно високою швидкістю роботи. На практиці генетичні алгоритми нерідко використовують спільно з іншими методами, які дозволяють підвищити їх точність.

Для багатьох задач метод перебору потребує дуже багато часу, а градієнтний метод не гарантує знаходження глобального екстремуму (рис. 5.10). Проте, комбінуючи переборний і градієнтний методи, можна сподіватися одержати хоча б наближене рішення, точність якого зростатиме при збільшенні часу розрахунку. Генетичний алгоритм є саме таким комбінованим методом. Механізми схрещування і мутації в якомусь значенні реалізують переборну частину методу, а відбір кращих рішень – градієнтний спуск. Така комбінація дозволяє забезпечити стійко хорошу ефективність генетичного пошуку для будь-яких типів задач (рис. 5.11).

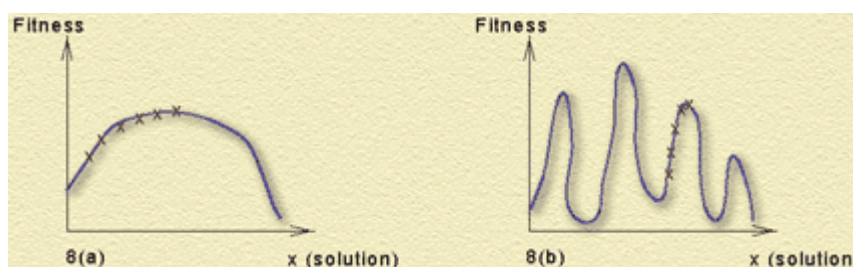


Рис. 5.10. Знаходження оптимального рішення (а) і локального максимуму (b) методом ітерацій.

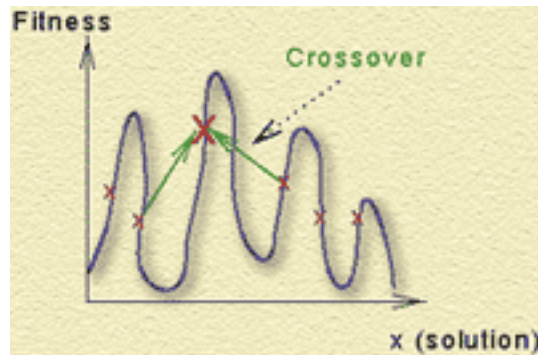


Рис. 5.11. Генетический пошук.

Генетичні алгоритми використовуються не тільки в задачах комбінаторного типу (як задача комівояжера), але і в тих випадках, коли підбирані параметри можуть бути будь-якими дійсними числами. Така, наприклад, задача оптимального розподілу інвестицій.

Генетичні алгоритми є евристичним підходом, тобто показує добрі результати на практиці, але погано піддається теоретичному дослідженню і обґрунтуванню. Ця популярність викликана, мабуть, винятковою красою підходу і його близькістю до природного механізму.

Контрольні запитання до розділу 5

1. Що таке генетичні алгоритми?
2. Опишіть походження генетичних алгоритмів.
3. Яка область використання генетичних алгоритмів?
4. Яким чином кодуються умови задачі в генетичних алгоритмах?
5. Які значення мають терміни „популяція”, „хромосома”, „ген” у генетичних алгоритмах?
6. У чому різниця між генотипом і фенотипом?
7. Опишіть класичний генетичний алгоритм.
8. Які є види селекції хромосом?
9. У чому полягає операція схрещування?
10. У чому полягає операція мутації?
11. Яким чином у генетичних алгоритмах формується нове покоління?
12. Які переваги та недоліки генетичних алгоритмів?

13. Яка структура хромосом використовується в генетичних алгоритмах?
14. Як виконується ініціалізація і селекція хромосом?
15. Як виконується схрещування і мутація хромосом?

6. КІБЕРНЕТИЧНІ СИСТЕМИ

6.1. Поняття кібернетичної системи

Кібернетика – це наука про керування і зв'язок [10]. Значний внесок у розвиток кібернетики зробив Норберт Вінер.

Кібернетична система є сукупністю об'єктів, що обробляють інформацію. Систему будемо розглядати як сукупність порівняно простих елементів, що зв'язані між собою в єдине ціле для виконання спільних функцій. Система має властивості, які не має сума її частин (автомобіль, рослина, комп'ютер). Система є підсистемою великої надсистеми та містить підсистеми (утворюється ієрархія).

Для системи як перетворювача інформації можна записати вираз

$$y_j = f_j(x_1..x_n, a_1..a_q), j = 1..m, \quad (6.1)$$

де y – вихідні сигнали, x – вхідні, a – внутрішні параметри.

Використовуються такі способи класифікації кібернетичних систем.

I. За ступенем визначеності функціонування:

- 1) детерміновані;
- 2) ймовірнісні.

II. За ступенем складності:

- 1) прості – можливо математично описати;
- 2) складні – математично описати дуже складно;
- 3) дуже складні – не існує задовільного математичного опису.

6.2. Керування кібернетичними системами

Керування означає переведення керованого об'єкту у заданий стан при мінімальних витратах. Розрізняють такі види керування:

- 1) точне керування – за математичною моделлю (наприклад, ПД-регулятор);
- 2) наближені методи;
- 3) дуже складні системи (живий організм, суспільство) – ситуаційне керування.

Для керування системами використовують розімкнений і замкнений контури керування (рис. 6.1).

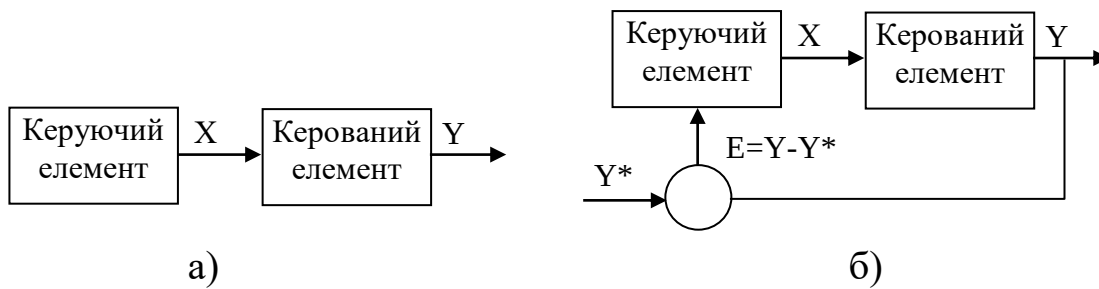


Рис. 6.1. Конттури керування: а) розімкнений, б) замкнений

Інформаційні виходи задаються значеннями інформаційних входів та станом системи (рис. 6.2).



Рис. 6.2. Інтелектуальна система як „чорний ящик”

Для адаптації системи використовують зворотний зв'язок (ЗЗ) (негативний – у підсилювачах для зменшення шумів, позитивний – у генераторах імпульсів/ інфляція).

Приклад негативного ЗЗ – система керування променем лазера при скануванні доріжки компакт-диску.

Стабільність системи забезпечується завдяки гомеостазу – підтримці умов нормального функціонування системи.

Для вирішення кібернетичних задач використовується **алгоритмічний і декларативний підхід**:

1. Алгоритмічний – імперативний (як зробити, конкретні інструкції).

2. Декларативний – що зробити (наприклад, зробити звіт).

В кібернетичних системах існує проблема поповнення первинних інструкцій, оскільки потрібно сприймати нову інформацію за змістом, на основі моделі світу (наприклад, людина правильно розуміє інструкції «обов'язково одягніть на голову каску», «обов'язково візьміть на руки собаку», що є складним для кібернетичних систем).

В кібернетичних системах формалізовано поняття алгоритмічності та декларативності з врахуванням таких позначень:

q – первинні інструкції, первинний опис ситуації;

S – фактори, що визначають поточний стан виконавців, містять множини S' і S'' ;

S' – контрольовані фактори;

S'' – неконтрольовані;

Z – знання виконавця;

r_A – робочий алгоритм, що формується на алгоритмічній основі;

r_D – робочий алгоритм на декларативній основі, програма не може бути явно сформованою.

У такому випадку компіляція описується виразом:

$$r_A = f(q, S').$$

Квазіалгоритм (наприклад, купити взуття) описується виразом:

$$r_D = h(q, S', S'', Z).$$

За означенням **алгоритм** – чітка недвозначна послідовність інструкцій, що зрозуміла виконавцеві і обов'язково приводить до гарантованого результату за скінчений час.

Квазіалгоритм застосовуються для опису неформалізованих задач, тому для квазіалгоритму результат не гарантований.

Існує 4 джерела квазіалгоритмічності:

- 1) випадкові чинники (погода);
- 2) неточність роботи виконавці (невірно визначити межі відрізків);
- 3) нечіткість формулювань (додати за смаком);
- 4) свобода волі – високоорганізована система має власні переконання й може не виконати наказ.

Розрізняють інтелектуальні системи бувають із зовнішнім керуванням, самокеруванням та комбіновані (марсохід).

Інтелектуальна система – самокерована кібернетична система, що має певну суму знань про світ і здатна на основі безпосереднього сприйняття і аналізу ситуацій планувати дії на досягнення мети, а також поповнювати знання.

6.3. Основні напрями досліджень кібернетичних систем

- 1) **біологічний** – моделювання психофізичних властивостей мозку, **когнітивна психологія** (імітація польоту птахів – махаюче крило);
- 2) **прагматичний** – орієнтація на отримання результату, **символьний** та **конекціоністський** підходи (практична зручність, використання нерухомого крила в літаках).

Соціальні наслідки реалізації систем штучного інтелекту описано в **трьох законах робототехніки Айзека Азімова**:

- 1) робот не може завдати шкоди людині;
- 2) робот не може виконати наказ людини, якщо він суперечить пункту 1;
- 3) робот може забезпечити свою безпеку, якщо це не суперечить пунктам 1 і 2.

7. ЗНАННЯ ЯК ІНФОРМАЦІЙНА ОСНОВА ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ

7.1. Методи подання знань

Знання – це систематизована інформація. Основним підходом до подання знань є **вербально-дедуктивний (або словесно-логічний)**, який полягає в наступному [10]:

- 1) будь-яка інформація задається вербально (набір тверджень і фактів);
- 2) основний механізм отримання знань – дедукція.

Проте, не всі знання є вербальними – є також **образне** мислення (що таке вода або температура відомо всім, але описати досить складно).

При **вербально-дедуктивному** визначенні знаннями інтелектуальної (І) системи називається трійка

$\langle F, R, P \rangle$,

де **F** – сукупність явних фактів;

R – сукупність правил виведення,

P – сукупність процедур, що визначають використання правил виведення.

База знань (БЗ) є сукупністю знань системи, містить **екстенсіональну** (сукупність фактів) та **інтенсіональну** частину (сукупність правил виведення). Наприклад:

Явні факти

F1. Заєць їсть траву

F2. Вовк їсть м'ясо

F3. Заєць є м'ясом

Правила виведення

R1. Якщо А їсть м'ясо, то А є хижаком

R2. Якщо А їсть В, то В є жертвою

З такої БЗ отримується **наслідок**, що заєць є жертвою.

Бази знань застосовуються в **експертних системах (ЕС)**, оскільки основою експертної системи може бути база знань.

Експертними системами називаються інтелектуальні програмні засоби, здатні у ході діалогу з людиною одержувати, накопичувати та коригувати знання у певній предметній області, виводити нові знання, розв'язувати практичні задачі [26]. ЕС акумулюють знання експертів

– провідних спеціалістів у даній галузі. Призначення ЕС – консультації для фахівців низької і середньої кваліфікації (медична і технічна діагностика, економічний аналіз).

Створюють ЕС фахівці 2 категорій:

- 1) експерт;
- 2) інженер знань – формалізує знання експерта для вводу їх у БЗ.

На даний час у багатьох сферах спостерігається перехід від використання баз даних до БЗ.

Між **інформаційними одиницями** утворюються певні зв'язки.

Опис, що утворює інформаційну одиницю – абстракція реальної сутності. Розрізняють такі зв'язки: часові, просторові відношення (знизу, зліва), причинно-наслідкові відношення. Системи на таких відношеннях утворюють псевдофізичні логіки (просторова, часова логіки), на основі яких можна робити висновки (наприклад, лекція з дисципліни «Комп'ютерні системи штучного інтелекту» почалася о 11:10, а відоеконференція почалася о 12:00. Яка подія почалася раніше?).

Основними зв'язками між інформаційними одиницями є узагальнення і агрегація (табл. 7.1).

Таблиця 7.1

Зв'язки між інформаційними одиницями

Агрегація	Узагальнення
Задає будову об'єкта (комп'ютер складається з процесора, материнської плати...)	Задає ієрархію класів
Відношення „Має”, 1) „Ціле – частина”, HAS_PART 2) „Є частиною”, IS_PART	Відношення „Є” 1) клас – підклас (наслідуються всі властивості) 2) екземпляр – клас (наслідуються ті властивості, що належать всім екземплярам класу)
Множина – закритий характер, різнотипні елементи	Клас – відкритий характер, однотипні елементи
Птах – екземпляр	Птах – клас
Ієрархія об'єктів (відношення елемент – множина)	Ієрархія класів (відношення клас – підклас)
Відношення екземпляр – клас	Спільне (об'єднує)

В БЗ існує **проблема винятків**, наприклад:

1. Всі птахи літають.
2. Пінгвіни – птахи.
3. Пінгвіни не літають.

У випадку винятків логіка БЗ стає **немонотонною** – додавання нового твердження змінює висновок. При виникненні **суперечностей** успадковується властивість від найближчого попередника в ієрархії класів.

Знання відрізняються від даних за такими ознаками:

- 1) внутрішня інтерпретованість – кожна інформаційна одиниця повинна мати унікальне ім'я (для пошуку).
- 2) структурованість – частина / ціле.
- 3) зв'язність: можна додавати, віднімати, змінювати зв'язки (причинно-наслідкові, просторові).
- 4) семантична метрика – близькість за змістом.
- 5) активність – виконання програм повинно визначатись станом БЗ.

Модель знань – це фіксована система формалізмів (понять і правил), згідно яких інтелектуальна система сприймає знання та здійснює операції над ними.

Моделі знань, що ґрунтуються на вербально-дедуктивній парадигмі, поділяються на 4 класи:

- 1) семантичні мережі;
- 2) фрейми;
- 3) логічні моделі;
- 4) продукційні моделі.

У БЗ можна виділити відносно незалежні фрагменти (області):

- 1) зміни в одній області приводять до малих змін у іншій;
- 2) задача може бути розбита на підзадачі.

В ЕС виділяються такі області знань:

- 1) предметна область знань, знання про систему;
- 2) область мови діалогу;
- 3) область системи, знання про власні можливості;
- 4) область користувача (доступ, рівень);
- 5) область діалогу (мета).

БЗ є сукупністю простих тверджень (речень природної мови), тобто концептуальних одиниць (факти + правила виведення).

Факт може описуватись **предикатом**, тобто логічною функцією, що залежить від тієї чи іншої кількості змінних і може приймати одне з 2 можливих значень: істинно / хибно, наприклад:

Твердження	Предикат „Плюс”
$7 + 5 = 12$	Плюс(7, 5, 12) = істинно
$4 + 1 = 9$	Плюс(4, 1, 9) = хибно

Складну концептуальну одиницю можна представити у вигляді бінарних предикатів (що залежать від двох змінних). Наприклад, концептуальна одиниця „Студент Петренко отримав 5 на іспиті зі штучного інтелекту” містить такі предикати:

Є (Петренко, Студент)

Є (Ісп_ІІІ, Іспит)

Ісп_ІІІ(Петренко, 5)

Об’єкти можна описувати **тріадами „об’єкт – атрибут – значення”** (або об’єкт – властивість - значення), наприклад, тріада: «Стіна – колір – червоний».

7.2. Семантичні мережі

7.2.1. Визначення та класифікація семантичних мереж

Семантичні мережі (СМ) – спосіб графічного подання знань, з наголосом на зв’язках між різними інформаційними одиницями. Інформація про певне поняття формується навколо відповідного вузла [10].

Семантичну мережу можна уявити у вигляді графу, вершини якого визначають об’єкти предметної області, а дуги відповідають зв’язкам між ними.

Формально сем. мережа визначається як набір

$$\langle I, C_1, C_2, \dots, C_n, \Gamma \rangle$$

де I – множина інформаційних одиниць, $C_1, C_2, \dots, C_n$ – типи зв’язків між інформаційними одиницями, Γ – відображення, що задає зв’язки між інформаційними одиницями.

Історично першою моделлю семантичної мережі була модель Квілліана.

Наприклад, твердження „Студент Петренко отримав 5 на іспиті зі штучного інтелекту” можна зобразити такою семантичною мережею (рис. 7.1).

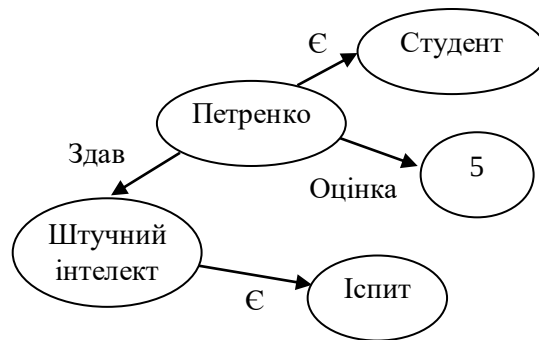


Рис. 7.1. Семантична мережа «Студент Петренко»

Розрізняють класифікуючі, функціональні СМ та сценарії.

- 1) **класифікуючі** СМ задають відношення ієрархії між інформаційними одиницями.
- 2) **функціональні** СМ характеризуються функціональними відношеннями, що задають обчислення одних інформаційних одиниць через інші.
- 3) у **сценаріях** використовується відношення „причина – наслідок”.

Є ряд свідчень, що знання в людській пам'яті зберігаються у вигляді структур, що нагадують семантичні мережі (школа Жана Піаже). Існує гіпотеза, що людина перетворює видимі об'єкти у семантичні мережі (**внутрішня мова**), а при малюванні виконує обернене перетворення – **реконструкцію** об'єкта.

Семантичні мережі можна описати **трирівневою архітектурою** (рис. 7.2). Повний БЗ відповідає семантична мережа, окремій концептуальній одиниці – концептуальний граф.



Рис. 7.2. Трирівнева архітектура семантичних мереж

7.2.2. Застосування семантичних мереж

Семантичні мережі підтримують **асиміляцію** нових знань. Для сприйняття інтелектуальною системою нової інформації потрібно приєднати концептуальний граф нового твердження до семантичної мережі.

Асиміляція знань – переклад знань на внутрішню мову системи:

- 1) **Повна** – нова інформація сприймається як істинна;
- 2) **Модальна** – інформація додається з примітками (джерело, умови).

Наприклад, до асиміляції «Байт містить 8 біт», після асиміляції: «У довіднику 1 вказано, що Байт містить 8 біт».

Сприйняття **нових знань** можна розбити на 2 етапи:

- 1) формування внутрішнього опису нової інформації у вигляді концептуальних графів;
- 2) під'єднання сформованих концептуальних графів (фрагментів) до існуючої семантичної мережі.

Семантичні мережі можливо відображати з вигляді графів (рис. 7.3) або **бінарних фактів** (об'єкт – атрибут – значення), наприклад:

Петренко – Є – Студент

Петренко – Здав – ШІ

Петренко – Оцінка – 5

ШІ – Є – Іспит

Найбільший недолік такого подання – складність зв'язків „один – до – багатьох”, „багато – до - багатьох” (наприклад, якщо Петренко здав ще СКБД, то до якого предмету відноситься оцінка); з метою уточнення можна поділити граф на підграфи (за тематикою).

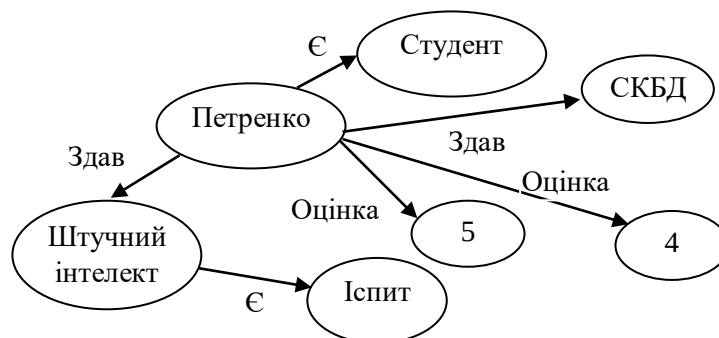


Рис. 7.3. Граф семантичної мережі

На семантичних мережах виконується **логічне виведення** (рис. 7.4).

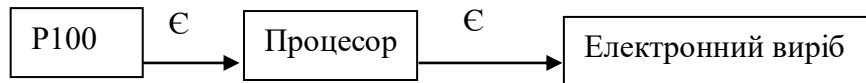


Рис. 7.4. Приклад логічного виведення

При суперечностях успадковуються властивості найближчого попередника (рис. 7.5) – наприклад, для пінгвіну, який не літає (на відміну від більшості птахів).

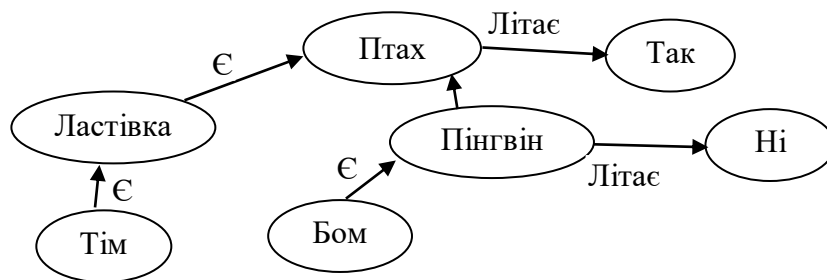


Рис. 7.5. Граф семантичної мережі, в якій при логічному виведенні виникають суперечності

7.3. Фрейми

7.3.1. Базові поняття фреймів

В 1975 році М.Мінський запропонував гіпотезу, що знання людини групуються у модулі (фрейми) [10].

Фрейм – це структура даних, призначена для опису типових ситуацій і типових задач, мінімально можливий опис основних характеристик об'єкта. Фрейм певного поняття може утворюватися з всіх бінарних фактів та пов'язаних з даним поняттями. У загальному випадку фрейм записується у вигляді:

Ім'я_Фрейму ((Атрибут_1, Значення_1), .. (Атрибут_i, Значення_i))

Структури даних, що описують окремі атрибути фрейму, називаються **слотами**. Наприклад, для графічного інтерфейсу програми **фреймом** є компонент Caption, **слотом** – його властивість Font (та інші властивості – Color, Width, ...).

В фреймах реалізується **конкретизація, ієрархія та наслідування фреймів**. Фрейм описує групу (клас) однотипних об'єктів з однаковими характеристиками. Конкретні об'єкти є екземплярами Фрейму.

Опис **екземплярів** виконується на основі **конкретизації** фреймів. Наприклад, існує така ієрархія фреймів (рис. 7.6):

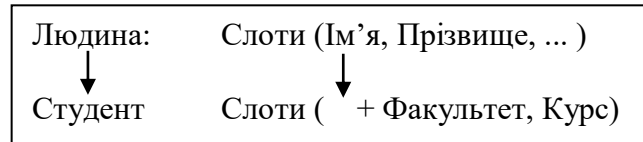


Рис. 7.6. Приклад ієрархії фреймів

7.3.2. Поповнення первинних описів на основі фреймів

Фрейми використовуються для опису ситуацій та для зіставлення з структурами, записаними у пам'яті (розпізнавання поняття). Поповнення первинного опису фрейму виконується так:

- 1) створюється екземпляр фрейму;
- 2) заповнюються слоти;
- 3) уточнюється фрейм.

Наприклад, фрейм «Заняття_в_школі» можна записати у вигляді:

ЗАНЯТТЯ_В_ШКОЛІ ((Вид_Роботи, Навчання), (Початок, 9.00), (Кількість_уроків, _), (Закінчення, Початок + Кількість_уроків*0.45 + 0.10*(Кількість_Уроків – 1), (Коли, _), (Хто, _), (Де, Вул. Головна, 8)).

У такому фреймі «Закінчення» є обчислюваним атрибутом. Якщо на вхід системи подати речення „Вчора Петро був у школі”, то виконується така послідовність дій:

- 1) за алгоритмом речення зіставляється з фреймом **ЗАНЯТТЯ_В_ШКОЛІ** (наприклад, за ключовими словами);

- 2) створюється екземпляр фрейму

Заняття_Петра (ЗАНЯТТЯ_В_ШКОЛІ) ((Вид_Роботи, Навчання), (Початок, 9.00), (Кількість_уроків, _), (Закінчення, Початок + Кількість_уроків*0.45 + 0.10*(Кількість_Уроків – 1), (Коли, Вчора), (Хто, Петро), (Де, Вул. Головна, 8)).

Частина слотів заповнюється з базового фрейму, а частина – з введеного тексту (частина може бути не заповненою). Після цього система може відповісти на питання, наприклад, „Де вчора був Петро?”. Якщо даних для відповіді не достатньо – виконується процедура уточнення (заповнення слотів).

Для виконання **операцій** фрейми містять **приєднані процедури** (їх можна розглядати як окремі слоти):

- 1) процедури зовнішнього виклику (з іншого фрейму);
- 2) демон – процедура, що автоматично активізується при виконанні певної умови (у тому числі слотові демони).

Фреймові структури можна зберігати в пам’яті по-різному, існує декілька мов опису знань на основі фреймів: FRL (Frame Representation Language), KRL (Knowledge Representation Language), (розвитком якого є HPRL) та ін.

У типовому фреймі зберігається:

- 1) ім’я і загальний опис фрейму;
- 2) інформація про батьківський фрейм;
- 3) інформація про слоти (імена, значення, режими успадкування, ...).

Для обробки знань використовують:

- 1) **формалізми для структуризації знань**: семантичні мережі та фрейми;
- 2) **процедури дедуктивного виведення**: логічні та продукційні моделі.

Фрейми і семантичні мережі мають багато спільного (вузли семантичної мережі можна розглядати як фрейми/слоти, а між слотами фрейму можна вводити різноманітні зв’язки). Проте, в семантичних мережах основна увага приділяється зв’язкам між інформаційними одиницями, а у фреймах – відношенням узагальнення та агрегації.

Мережі подібностей і відмінностей використовують для пошуку фреймів, які найкраще описують дане поняття.

Сценарій – фреймоподібна структура даних, яка визначає послідовність дій для певного процесу (для семантичних мереж – вказуються причинно-наслідкові зв’язки).

Наприклад, сценарій „Відвідування_Ресторану”:

Відвідування_Ресторану

(Мета: Приймання їжі,

Ролі: Відвідувач, Офіціант;

Змінні: Назва ресторану, Місце, Чекання, Їжа, Сума;

Кількість сцен: 4

Сцена_1 (Зайняття місця):

Відвідувач заходить до „Назва_ресторану”

Відвідувач вибирає „Місце” (умова: місце вільне)

.....

7.3.3. Фрейми та об’єктно-орієнтоване програмування

Можна вважати, що **об’єктна** модель є конкретизацією більш абстрактної **фреймової** моделі, в якій використовуються такі поняття:

Об’єкт (екземпляр класу) – абстракція, що характеризується станом, поведінкою та ідентифікованістю.

Стан – сукупність властивостей (статичні / динамічні).

Поведінка – реакція на події (приводить до зміни станів та передачі повідомлень).

Ідентифікованість – властивість, що відрізняє об’єкт від інших.

Об’єкти та класи мають такі найважливіші властивості:

1. **Абстрагування** – опис найважливіших характеристик деякої сутності.
2. **Інкапсуляція** – внутрішня будова залишається у межах класу.
3. **Модульність.**
4. **Ієрархія** класів/ об’єктів.

Використовуються такі об’єктно-орієнтовані мови і середовища: Delphi, C++, Java, Python та ін.

Властивості фреймів використовуються в мові UML (Unified Modeling Language – уніфікована мова моделювання), яка є мовою семантичного моделювання предметних областей та подальшого проектування відповідних програмних засобів. В основі мови UML – виділення класів, опис їх характеристик та взаємозв’язків. У 1997 р. мова UML була прийнята консорціумом **OMG** (Object Management Group) як стандарт.

Основою опису предметної області в UML є **діаграма класів** (може бути реалізована в об'єктно-орієнтованих мовах). Відношення між класами наступні:

1. **Асоціація** – описує структурні зв'язки між класами.
2. **Узагальнення** – відношення „Є” (клас / підклас)
3. **Залежність** – вплив однієї сутності на іншу.

7.4. Логічні моделі та метод резолюцій

7.4.1. Логічні побудови та логічні моделі

Для практичних завдань важливо так формалізувати логічні побудови, щоб вони виконувалися автоматично (за допомогою комп'ютерних програм). Найкраще піддається автоматизації дедуктивний спосіб виведення [10].

Розглянемо дві задачі автоматизації дедуктивних побудов:

- 1) **задача виведення наслідків**: знайти всі вірно побудовані формули, які можна логічним шляхом вивести з аксіом на основі правил виведення (знайти всі наслідки з фактів);
- 2) **задача перевірки правильності твердження (автоматичного доведення теорем)**.

Автоматизація дедуктивних побудов використовується в експертних системах.

Механізм автоматичного доведення теорем покладений в основу **логічного програмування**. Програма розглядається як набір логічних формул разом з теоремою, що має бути доведена. Найвідоміший представник логічного програмування – мова **Пролог**. У той же час **людську** логіку можна вважати інтелектуальною моделлю з **нечіткою** структурою.

Логічна модель – формальна система, що задається множинами $\langle T, P, A, B \rangle$,

де **T** – множина базових елементів (алфавіт);

P – множина синтаксичних правил;

A – множина аксіом (формули, які приймаються за істинні);

B – множина правил виведення.

У типовому випадку логічна модель задання знань базується на формалізмах логіки **предикатів першого порядку**.

7.4.2. Вступ до числення предикатів

Предикатом є логічна функція (вірність або хибність певного твердження, наприклад, „X більше 2”). Логіка предикатів застосовує математичний апарат формалізації людського мислення.

Числення предикатів використовує:

- 1) константи (константні **терми**): $C_1, C_2, \dots$
- 2) змінні (змінні **терми**): $x_1, x_2, \dots$
- 3) функціональні літери: $f_1, f_2, \dots$
- 4) предикатні літери: $p_1, p_2, \dots$
- 5) логічні символи: $\rightarrow, \leftrightarrow, \vee, \&, \sim, \exists, \forall$.

Елементарне твердження складається з предиката і зв'язаних з ним термів. Складні твердження будуються з елементарних за допомогою логічних зв'язок: „і” (and, $\&$), „або” (or, $\vee$), „ні” (not, $\sim$), **імплікації** „ $\rightarrow$ ” (якщо ..., тоді ...)

Якщо твердження позначити через **a** і **b**, то між ними можливі наступні зв'язки (табл. 7.2).

Таблиця 7.2

Логічні зв'язки між твердженнями

Зв'язок	Запис	Інтерпретація
Заперечення	$\sim a$	не є a
Кон'юнкція	$a \& b$	a та b
Диз'юнкція	$a \vee b$	a або b
Імплікація	$a \rightarrow b$	a впливає з b
Тотожність (еквівалентність)	$a \leftrightarrow b$	a еквівалентне b

Приклад імплікації: Клен (X) $\rightarrow$ Дерево (X)

Для імплікації та тотожності справедливі твердження:

- 1) $a \rightarrow b$ дорівнює $(\sim a) \vee b$
- 2) $a \leftrightarrow b$ дорівнює $(a \& b) \vee (\sim a \& \sim b)$
- 3) $a \leftrightarrow b$ дорівнює $(a \rightarrow b) \& (b \rightarrow a)$

З кожним **предикатом** може бути пов'язаний **квантор** – елемент, який визначає, за яких умов предикат може перетворюватися на істинне висловлювання.

Розрізняють квантор **узагальнення** ($\forall$) і квантор **існування** ($\exists$).

Якщо u - змінна, r - твердження, то запис

“ $\forall u \ r$ ” означає „ r справджується для всіх u ” (для будь-якого u , all);

“ $\exists u \ r$ ” означає „існує u , для якого r справджується”

(існує хоча б одне u для якого ..., exists);

Наприклад: $\text{All}(X, \text{Ясен}(X) \rightarrow \text{Дерево}(X))$

Елементарним об'єктом, який має значення „істина”, є **атомарна формула (літерал)**:

$P(t_1, t_2, \dots, t_n)$, де P – предикат, t – терми, n - арність предикату.

Хибна формула: „ $\square$ ”.

Тавтології (тотожньо істинні формули): $\sim A \leftrightarrow A$.

Позитивний літерал не стоїть під знаком заперечення, а **негативний** – під знаком заперечення.

Диз'юнктом (фразою теорії) називається диз'юнкція деякої кількості атомарних формул.

Часто використовують фрази спеціального типу – **фрази Хорна** (хорнівські диз'юнкти) – диз'юнкція довільної кількості атомарних формул, з яких позитивною є не більше ніж одна ($\sim A \vee \sim B \vee \sim C \vee D$).

Фрази Хорна по суті є **імплікаціями**, оскільки за правилами Моргана ($\sim A \vee \sim B \vee \sim C \vee D$) еквівалентне $(A \vee B \vee C) \rightarrow D$.

Для практичного застосування у логічних виразах потрібно позбутися кванторів узагальнення та існування. Для усунення **кванторів існування** використовують **константи і функції Сколема**. Наприклад:

- 1) У формулі $\exists x: P(x)$ вкажемо предметну константу C (константу Сколема), для якої твердження $P(C)$ є істинним; таким чином перейдемо від $\exists x: P(x)$ до $P(C)$.
- 2) У формулі $\forall y \exists x: P(x, y)$, вкажемо деякий елемент C , для якого твердження $P(C, y)$ є істинним, тому можна перейти від формули $\forall y \exists x: P(x, y)$ до формули $\forall y P(h(y), y)$, де $h(y)$ – функція Сколема.

Квантори узагальнення просто видаляються (вважається, що якщо фраза залежить від деяких змінних, то всі вони зв'язані оператором узагальнення); формула $\forall x \forall y \forall z P(x, y, z)$ замінюється на $P(x, y, z)$.

Використовується фразова форма запису логічних формул, для спрощення логічних формул використовують рівності:

- 1) $\sim(\mathbf{a\&b})$ означає $(\sim\mathbf{a}) \vee (\sim\mathbf{b})$;
- 2) $\sim\mathbf{Exists(u,r)}$ означає $\mathbf{All(u,\sim r)}$;
- 3) $\sim\mathbf{All(u,r)}$ означає $\mathbf{Exists(u,\sim r)}$.

У результаті перетворень отримуються хорнівські фрази, наприклад $(\sim\mathbf{A} \vee \sim\mathbf{B} \vee \sim\mathbf{C} \vee \mathbf{D})$, яка еквівалентна $\mathbf{D} \leftarrow (\mathbf{A} \vee \mathbf{B} \vee \mathbf{C} \vee)$, а в мовах логічного програмування записується у вигляді $\mathbf{D:- A, B, C}$.

На основі логічних моделей можлива побудова теорії певної області знань, яка включає дослідження структури цієї області і вибір позначень для її елементів. Потім будується множина вірно **побудованих формул (ВПФ)** для опису цієї структури. Множина ВПФ є **теорією** цієї області знань, в якій кожна ВПФ – **аксіома**.

Для побудови бази знань певної предметної області використовують наступні принципи:

1. Визначаються предметні змінні.
2. База знань описується як кон'юнкція деяких тверджень (фактів або правил виведення).
3. Факти задаються позитивним або негативним літералом, що не містить змінних.
4. Правила виведення задаються фразами Хорна.
5. Квантори існування усуваються шляхом введення констант і функцій Сколема.
6. Квантори узагальнення усуваються механічно.
7. Після побудови формалізованої бази знань здійснюється логічне виведення.

Наприклад, є неформальний опис: „Всі студенти люблять відвідувати лекції. Деякі студенти люблять морозиво. Іваненко є студент, але не любить морозива”.

Введемо такі предикати і предметні константи:

$\mathbf{L(x,y)}$ – $\mathbf{x}$ любить $\mathbf{y}$.

$\mathbf{S(x)}$ – $\mathbf{x}$ є студентом.

$\mathbf{A}$ – предметна константа „відвідування лекцій”.

$\mathbf{I}$ – предметна константа „Іваненко”

$\mathbf{Q}$ – предметна константа „Морозиво”.

Отримаємо базу знань:

$$\forall x: (\sim S(x) \vee L(x,A))$$

$$\exists x: (S(x) \& L(x,Q))$$

$$S(I)$$

$$\sim L(I,Q)$$

Усунення кванторів і розподіл кон'юнкцій приводить до такої бази знань

$$S(x) \vee L(x,Q)$$

$$S(c)$$

$$L(c,Q)$$

$$S(I), \sim L(I,Q),$$

де C – константа Сколема.

7.4.3. Логічне програмування

Логічне програмування реалізується, зокрема, мовою Пролог. У логічному програмуванні для автоматичного доведення певного твердження (ВПФ) використовується метод від **супротивного**: береться інверсія ВПФ і додається до бази знань; якщо при цьому виникає протиріччя ($A \& \sim A$), то ВПФ вважається вірною.

Фрази бази знань можуть бути спрощені методом **резолюцій**. Для цього знаходиться дві фрази, які утворюють **контрарну пару**. В контрарній парі одна фраза містить певну атомарну **формулу без заперечення**, а інша фраза – ту ж формулу із **запереченням**.

Наприклад, фрази $(\sim P \vee M)$ і $(P \vee L)$ утворюють контрарну пару, результатом якої (резольвентою) є фраза $(M \vee L)$.

Розглянемо приклад застосування методу резолюцій. Дано набір фраз: „вовк їсть м'ясо”, „тварини, які їдять м'ясо, є хижаками”.

Потрібно довести, що „вовк є хижаком”

Введемо **предикати**:

$$G(x,y) \Rightarrow \text{„}x \text{ їсть } y\text{”}, \quad Q(x) \Rightarrow \text{„}x \text{ є хижак”}$$

Предметні **константи** такі: W – вовк, M – м'ясо.

Отримаємо базу знань: $G(W,M)$;

$$\sim G(x,M) \vee Q(x)$$

Додамо до БЗ заперечення: $G(W,M)$;

$$\sim G(x,M) \vee Q(x)$$

$$\sim Q(W)$$

Резолюція двох останніх записів дає $\sim G(W,M)$, тобто виникає протиріччя

7.5. Продукційні моделі

7.5.1. Характеристика продукційних моделей

Продукційною називається модель знань, в якій база знань складається із сукупності продукцій, тобто правил „**Якщо А, тоді В**”.

Термін „**продукція**” введений американським математиком Е. Постом. Як і логічні, продукційні моделі базуються на дедуктивному виведенні, але є менш формалізованими і простими [10]. Приклади продукцій для зоологічного парку: „Якщо тварина є птахом, вона не літає, вона плаває, має чорно-білий колір – то це пінгвін”.

Продукції описують **імплікації** (з істиності **А** впливає істинність **В**). Приклад для системи керування технологічним процесом: „Якщо температура перевищує 200 градусів, то ввімкнути систему охолодження”. Такі продукції задають **умовні операції** (якщо справедлива умова **А**, то виконати дію **В**).

Продукційна система – це база знань разом з механізмом керування продукціями. Продукційна модель передбачає подання знань у вигляді сукупності продукцій. Продукція визначається виразом:

$$(i) Q; P; A \Rightarrow B; N,$$

де **(i)** – ім'я продукції, **Q** – сфера застосування продукції (предметна область), **P** – умова застосування продукції, **$A \Rightarrow B$** ядро продукції, **N** – постумова продукції, постулює виконання дій після реалізації ядра.

Приклад продукції:

„(БС-29) Банківська сфера; клієнт має рахунок у банку на достатню суму; якщо клієнт бажає зняти певну суму грошей, то він повинен написати ордер; внести зміни до бази даних”.

Продукційні моделі можна поєднувати з **семантичними мережами**, де знання описуються семантичними мережами, а операції над знаннями задаються як продукції.

Переваги продукційних моделей такі:

1. Більшість людських знань можна описати у вигляді продукцій.
2. Більшість продукцій є взаємно незалежними.
3. Продукційні моделі можуть реалізувати будь-які алгоритми.
4. Продукції можуть працювати паралельно.

Основний **недолік** продукцій – складність перевірки правильності функціонування.

7.5.2. Продукції та мережі виведення

Системи продукцій можна зображати у вигляді графів – **мереж виведення**. Мережі виведення будуть так. Вершини графу відповідають умовам і діям, що входять до лівих і правих частин продукцій. Якщо є продукція $A \rightarrow B$, то від вершини A до вершини B йде орієнтована дуга. Якщо є продукція $A, B \rightarrow C$, до дуги (AC) та (BC) зв'язані відношенням „I” (кон'юнкції).

Наприклад, для продукційної системи, що описується графом (рис. 7.7), існують такі продукції:

$A \rightarrow C$

$B \rightarrow C$

$B, F \rightarrow L$

$F \rightarrow Q$

$D, C \rightarrow G$

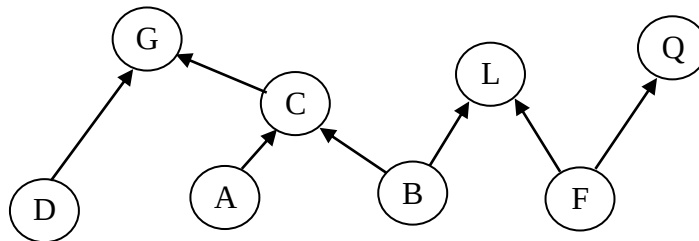


Рис. 7.7. Граф продукційної системи

Мережа виведення забезпечує поділ задачі на підзадачі і є структурою, близькою до **I-АБО-графів**.

7.5.3. Схема роботи експертної системи на базі продукцій

Експертна система на базі продукцій має таку структуру (рис. 7.8).

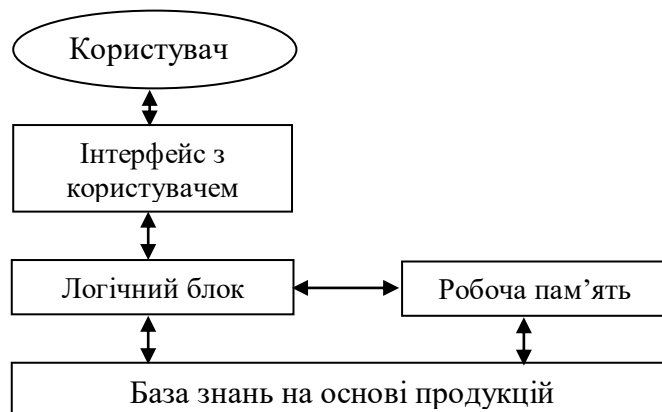


Рис. 7.8. Структура експертної системи на базі продукцій

До робочої пам'яті системи заносяться факти від користувача, а також його запити. Логічний блок зіставляє цю інформацію з правилами із бази знань, і використовує ті правила, що можуть бути зіставлені зі вмістом робочої пам'яті.

Наприклад, продукції $A \rightarrow B$ і $B \rightarrow C$ виконуються, якщо стає відомо про істинність A .

В експертних системах виконується пряме та зворотне виведення.

Пряме виведення на основі правил передбачає аналіз:

- 1) усіх наслідків з фактів;
- 2) наслідків з наслідків і т.д.

Процес продовжується, поки не буде встановлено істинність або хибність запиту користувача. Нехай є продукції $A \rightarrow B$ та $B \rightarrow C$; користувач оголосив про істинність факту A і спитав про істинність факту C . Ланцюг прямого виведення матиме вигляд: з A виводиться B , з B виводиться C , отже C – істинне. При прямому виведенні перебирається багато безперспективних продукцій.

При **зворотному виведенні** логічний блок починає роботу з запиту користувача, тобто з твердження, яке перевіряється. Розглядається одне з правил, на основі яких можна вивести дане твердження, після чого перевіряється істинність лівої частини цього правила. Процес повторюється, поки не дійде до фактів, які вважаються істинними.

У нашому випадку ланцюжок зворотного розповсюдження матиме вигляд: „ C виводиться з B , B виводиться з A , A істинне, отже і C істинне”.

Зворотне виведення, як і пряме, є стратегією перебору (вимагає повторень у випадку невдалої спроби). В загальному для зворотного виведення потрібно виконати менше продукцій, ніж для прямого. Зворотне виведення тісно пов'язане з прологівським механізмом виконання програм.

Продукції поділяються на детерміновані й не детерміновані.

У **детермінованих** продукціях при виконанні лівої частини завжди виконується й права.

У **не детермінованих** продукціях права частина при виконанні лівої виконується з певною ймовірністю.

Є два режими керування виконанням продукцій:

- 1) режим **негайного виконання** (виконується перша вірна продукція);
- 2) режим формування **конфліктного набору** або „фронту готових продукцій” (списку продукцій, готових до виконання). Далі відбувається вибір однієї продукції зі списку.

При виконанні конфліктних наборів виникають проблеми вибору продукцій. Наприклад, нехай існують продукції, які готові до виконання і утворюють конфліктний набір:

$$1) A \rightarrow B$$

$$2) A \rightarrow C,$$

де A – істинне.

Якщо ядра продукцій є **імплікаціями**, то продукцій на система є чисто декларативною і особливих проблем не виникає. Взаємний вплив між продукціями відсутній, тобто порядок виконання продукцій впливає тільки на час роботи системи. Якщо ж ядра продукцій є **процедурами**, то в результаті виконання однієї продукції може змінитися стан іншої продукції.

Наприклад, після виконання першої продукції A може перестати бути істинними, відповідно продукція 2 не виконається. Якщо ж першою виконається продукція 2, то результат може бути зовсім інший.

Існують різноманітні стратегії вирішення конфліктів, **централізоване** і **децентралізоване** керування продукціями (наприклад, механізм „класної дошки”). Класна дошка – спільна область пам’яті для всіх продукцій, на якій кожна продукція може знайти інформацію про свій запуск.

Розглянемо основні **стратегії вирішення конфліктів** у продукційних системах. Для повного аналізу дій системи потрібно перебрати всі можливі варіанти її роботи. Це вимагає експоненційного зростання об’ємів обчислень при збільшенні числа продукцій, тому стратегій вирішення конфліктів зводяться до зменшення кількості варіантів перебору і по суті є евристичними. Розглянемо деякі з них:

1. Принцип „стосу книг”. Ґрунтується на ідеї, що найкориснішою є та продукція, яка використовується найчастіше (у стосі книг зверху звичайно опиняються книги, що користуються найбільшим попитом). Стратегія є ефективною, якщо продукції відносно незалежні між собою.

2. Принцип найдовшої умови. З фронту готових продукцій вибирається та, для якої справедлива найдовша умова (вважається, що спеціалізовані правила корисніші за загальні).

Наприклад, з двох продукцій:

1) Якщо А – птах, то А – літає;

2) Якщо А птах і А – пінгвін, то А не літає.

за принципом найдовшої умови вибирається 2. Принцип найдовшої умови ефективний при аналізі винятків.

3. Принцип метапродукцій. Ґрунтується на введенні до системи мета продукцій, тобто правил використання продукцій. Типовими мета продукціями є правила, що визначають дії при надходженні до фронту готових продукцій нових.

4. Принцип „класної дошки”. Конфлікти вирішуються на основі обміну інформацією з використанням класної дошки.

5. Принцип вибору за пріоритетом (важливістю). Пріоритети продукцій можуть бути різними для різних ситуацій, тобто статичними і динамічними.

8. НЕЧІТКА ЛОГІКА

8.1. Поняття нечіткості та нечіткі функції належності

Для природної мови характерна **нечіткість**. Наприклад: маленьке число, висока температура, тощо. Прилад невизначеності – означення **купи**, коли можливі псевдоіндуктивні логічні висновки:

- 1) один камінець не утворює купу;
- 2) додавання одного камінця до «не купи» не перетворює її на купу;
- 3) отже, будь-яка кількість камінців не утворює купу.

Для формалізації нечітких понять запропоновано апарат **нечітких** (розмитих) множин (**fuzzy sets**) [10, 27].

Функція належності є основною характеристикою нечіткої множини. Якщо деяке поняття не можна окреслити точно, то його можна описати наближено. Наприклад, людина з ростом 135 см зовсім не висока, 170 см – більш-менш висока, 185 см – досить висока. Тобто для кожного росту можна задати **ступінь** того, наскільки такий ріст відповідає **поняттю** висока людина.

Зі **звичайними множинами** тісно пов'язане поняття **характеристичної функції**. **Характеристичною функцією** $\chi(x)$ називається функція, значення якої вказують, чи належить елемент x множині A : $\chi_A(x)=1$, якщо $x \in A$ і $\chi_A(x)=0$, якщо $x \notin A$.

Професор Л.Заде у 1965 р. запропонував розширити поняття характеристичної функції. Він ввів поняття **функції нечіткої належності** $\mu(x)$ елемента до множини, дозволивши цій функції набувати будь-якого значення від 0 до 1. Чим більшим є значення функції належності, тим більшою мірою належить елемент до множини.

Якщо $\mu_A(x)=1$, то елемент x чітко належить множині A , якщо ж $\mu_A(x)=0$, то елемент x чітко не належить A . При $0 < \mu_A(x) < 1$ елемент належить множині нечітко. Чим більшим є значення функції належності, тим більшою мірою елемент належить множині.

Означення

Нехай E – довільна непуста множина. **Нечіткою підмножиною** A множини E називається множина пар

$$A = \{(x, \mu_A(x))\},$$

де $x \in E, \mu_A(x) \in [0, 1]$.

Функція $\mu_A: E \rightarrow [0,1]$ називається **функцією належності** нечіткої підмножини **A**, а **E** – **базовою множиною** (базовою шкалою).

Значення $\mu_A(x)$, визначене для кожного x , називається ступенем належності елемента до множини.

Наприклад, формалізуємо поняття „**високий зріст**”. Для цього слід визначити функцію належності для відповідної нечіткої множини. Базова множина **E** – множина невід’ємних цілих чисел. Функцію належності (рис. 8.1) можна задати аналітичною формулою

$$\mu_A(x) = \begin{cases} 0; & x < 1,5; \\ 2x - 3; & 1,5 \leq x \leq 2; \\ 1; & x > 2 \end{cases} \quad (8.1)$$

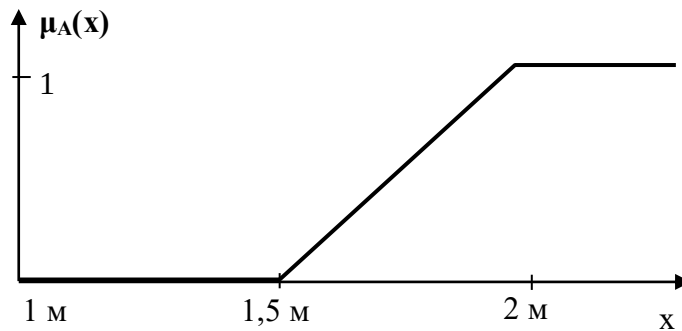


Рис. 8.1. Функція належності поняття „високий зріст”

У такому випадку на запитання „Чи є високою людина зі зростом 185 см?” може бути відповідь: „Ступінь належності 0,7”, що означає „досить висока”.

Цю саму нечітку множину можна задати явно, перелічивши деякі ступені належності:

$$A = \{ (1,5; 0), (1,6; 0,2), (1,7; 0,4), (1,8; 0,6), (1,9; 0,8), (2,1) \}$$

Для нечітких множин є такі особливості:

1. **Функція належності не може бути визначена однозначно**, оскільки вона є **суб’єктивною** (кожна людина може запропонувати свої варіанти).
2. **Самі значення функції належності можна розглядати як розмиті**. Наприклад, дехто оцінив зріст 185 см як високий зі ступенем належності 0,8; але ступінь належності може приймати також значення 0,78 або 0,82. Тому можна говорити про **інтервали**

оцінок ступенів належності. Якщо ступінь належності кожного елемента оцінюється інтервалом, то говорять про **нечіткі множини другого роду** з функцією належності $\mu_A : E \rightarrow [0,1]^{[0,1]}$. Такі міркування можна продовжити й далі. Проте на практиці звичайно користуються нечіткими множинами першого роду (з конкретними значеннями ступенів належності).

3. **Незважаючи на те, що функція належності є суб'єктивною оцінкою, її не можна вибрати довільно.** У нашому випадку функція належності, яка описує поняття „високий зріст”, повинна бути неспадною: неможливо, щоб хтось оцінював зріст 165 см більшим ступенем належності, ніж зріст 195 см.

8.2. Основні операції над нечіткими множинами

Розглянемо основні операції над нечіткими множинами:

1. **Рівність.** Дві нечіткі підмножини базової E називаються рівними між собою, якщо $\forall x \in E : \mu_A(x) = \mu_B(x)$; позначається рівність $A=B$.
2. **Включення.** A є підмножиною B ($A \subseteq B$), якщо $\forall x \in E : \mu_A(x) \leq \mu_B(x)$.
3. **Перетин.** Перетином двох нечітких множин A і B називається нечітка множина $C = A \cap B$ з функцією належності $\mu_C(x) = \min(\mu_A(x), \mu_B(x))$.
4. **Об'єднання.** Об'єднанням двох нечітких множин A і B називається нечітка множина $C = A \cup B$ з функцією належності $\mu_C(x) = \max(\mu_A(x), \mu_B(x))$.
5. **Доповнення.** Нечітка множина B є доповненням нечіткої множини A , якщо $\mu_B(x) = 1 - \mu_A(x)$.

Для нечітких множин виконуються всі основні властивості звичайних чітких множин, крім двох: об'єднання $A \cup \bar{A}$ не обов'язково дорівнює базовій множині; перетин $A \cap \bar{A}$ не обов'язково дорівнює порожній множині.

8.3. Нечітке логічне виведення

Розглянемо типову задачу логічного виведення в умовах невизначеності:

Дано (нечітке) продукцій не правило „Якщо А, то В”.

Спостерігається A' (А в певній мірі).

Яким повинно бути В?

Формальніше задані носії U і V та їх нечіткі підмножини A і B .
Задано або елемент $u \in U$, або множину $A' \subseteq U$. Потрібно визначити елемент $v \in V$, що являє висновок – результат застосування нечіткого правила.

Приклад. Дано нечітке продукцій не правило:

„Якщо студент багато працює в бібліотеці, то він отримає високу оцінку”.

Як множину U можна розглядати множину чисел, яка визначає кількість годин на тиждень (0..30), які студент проводить в бібліотеці:
 $U = \{0, 3, 6, 9, 12, 18, 21, 27\}$.

Як множину V оцінок (рейтингів) за 100-бальною шкалою розглянемо діапазон чисел від 59 до 100: **$V = \{59, 72, 84, 91, 96, 100\}$.**

Задамо функції належності для нечітких множин A („багато працює в бібліотеці”) та B („високий рейтинг”) таким чином:

$$A = \{(3, 0); (6, 0.1); (9, 0.4); (12, 0.6); (18, 0.8); (21, 1); (27, 1)\}$$

$$B = \{(59, 0); (72, 0.2); (84, 0.4); (91, 0.7); (96, 0.9); (100, 1)\}$$

Нехай явно задана кількість годин x , які студент працює в бібліотеці. Знаючи x , можна знайти ступінь належності α елемента до множини й навпаки. Тоді для отримання висновку можна застосувати метод простої підстановки нечіткого значення, відповідно до якого V вибирається з умови:

$$\mu_B(V) = \alpha. \quad (8.2)$$

Нехай у нашому прикладі дано, що студент працює в бібліотеці 9 годин. Ступінь належності дорівнює 0.4 і система робить висновок, що такий студент повинен отримати оцінку 84 (добре). Якщо у множені V відсутнє значення з таким ступенем належності, то береться найближче значення або виконується інтерполяція.

Для отримання чітких значень (на основі нечітких) застосовується, зокрема, **метод центру тяжіння композиції максимум-мінімум.**

У деяких випадках A' задається не як конкретне значення, а як нечітка множина (наприклад, якщо у вище приведеному прикладі відомо, що „студент проводить у бібліотеці середню кількість часу”), тому використовуються **відношення нечіткої імплікації**.

Нечітким відношенням між множинами A та B називається нечітка підмножина їх декартового добутку. Тобто, якщо

$$A = \{(u_i, \mu_A(u_i)), u_i \in U\}, B = \{(v_j, \mu_B(v_j)), v_j \in V\}, \quad (8.3)$$

то відношення $A \mathbf{R} B$ визначається як множина пар

$$ARB = \{(u_i, v_j), \mu_R(u_i, v_j), u_i \in U, v_j \in V\}. \quad (8.4)$$

Відношення нечіткої імплікації $A \rightarrow B$ можна виводити по різному. Часто використовується формула **min-імплікації**:

$$\mu_{A \rightarrow B}^{(\min)}(u_i, v_j) = \min(\mu_A(u_i), \mu_B(v_j)). \quad (8.5)$$

Для задання імплікації задають також й інші формули:

1) нечітке розширення класичної імплікації:

$$\mu_{A \rightarrow B}^{(KI)}(u_i, v_j) = \min(\mu_B(v_j), 1 - \mu_A(u_i)), \quad (8.6)$$

2) нечітка імплікація Лукашевича:

$$\mu_{A \rightarrow B}^{(Luc)}(u_i, v_j) = \min(1, 1 - \mu_A(u_i) + \mu_B(v_j)). \quad (8.7)$$

За допомогою імплікації можна отримати множину B' – нечітку множину висновків, які відповідають множині A' . Це є результатом композиції множини A' й нечіткої імплікації:

$$B' = A' \bullet (A \rightarrow B), \quad (8.8)$$

де $\bullet$ – знак композиції, що обчислюється за формулою

$$\mu_{B'}(v_j) = \max_i \cdot \min(\mu_{A'}(u_i), \mu_{A \rightarrow B}(u_i, v_j)). \quad (8.9)$$

Проте, отримати лише множину B' недостатньо, треба ще знайти конкретну числову відповідь (провести **дефазифікацію**). Найчастіше за числову відповідь береться **центр тяжіння** знайденої нечіткої множини, який обчислюється за формулою

$$v^* = \frac{\sum v_j \mu_{B'}(v_j)}{\sum \mu_{B'}(v_j)}. \quad (8.10)$$

Увесь описаний метод нечіткого логічного виведення часто називають методом **центру тяжіння композиції максимум-мінімум**.

Повернемося до попереднього прикладу.

Нехай задано функції належності для нечітких множин **A** („багато працює в бібліотеці”) та **B** („високий рейтинг”) таким чином:

$$A = \{(3, 0); (6, 0.1); (9, 0.4); (12, 0.6); (18, 0.8); (21, 1); (27, 1)\}$$

$$B = \{(59, 0); (72, 0.2); (84, 0.4); (91, 0.7); (96, 0.9); (100, 1)\}$$

Дано, що „студент працює в бібліотеці середню кількість часу”.

Задамо нечітку множину **A'** (середня кількість часу):

$$A' = \{(3, 0); (6, 0.2); (9, 0.7); (12, 1); (18, 0.6); (21, 0.2); (27, 0)\}$$

Обчислимо відношення імплікації нечітких множин **A** та **B** (табл. 8.1):

Таблиця 8.1

Відношення імплікації нечітких множин **A** та **B**

		59	72	84	91	96	100
	A/B	0	0.2	0.4	0.7	0.9	1
3	0	0	0	0	0	0	0
6	0.1	0	0.1	0.1	0.1	0.1	0.1
9	0.4	0	0.2	0.4	0.4	0.4	0.4
12	0.6	0	0.2	0.4	0.6	0.6	0.6
18	0.8	0	0.2	0.4	0.7	0.8	0.8
21	1	0	0.2	0.4	0.7	0.9	1
27	1	0	0.2	0.4	0.7	0.9	1

Знайдемо композицію множини **A'** та щойно знайденого відношення **A**→**B**. Результатом буде нечітка множина **B'** (табл. 8.2):

Таблиця 8.2

Композиція множин

		59	72	84	91	96	100
	A/B	0	0.2	0.4	0.7	0.9	1
3	0	0	0	0	0	0	0
6	0.2	0	0.1	0.1	0.1	0.1	0.1
9	0.7	0	0.2	0.4	0.4	0.4	0.4
12	1.0	0	0.2	0.4	0.6	0.6	0.6
18	0.6	0	0.2	0.4	0.6	0.6	0.6
21	0.2	0	0.2	0.2	0.2	0.2	0.2
27	0	0	0	0	0	0	0
	B'	0	0.2	0.4	0.6	0.6	0.6

Далі проведемо дефадзифікацію отриманої множини $\mathbf{B}'$ (за формулою 8.10)

$$v^* = (72 \cdot 0.2 + 84 \cdot 0.4 + 91 \cdot 0.6 + 96 \cdot 0.6 + 100 \cdot 0.6) / (0.2 + 0.4 + 0.6 + 0.6 + 0.6) = 220.2 / 2.4 = 91.75 \approx 92.$$

Отже, на основі проведеного розрахунку із застосуванням нечіткого композиційного правила виведення можна дійти висновку, що студент повинен отримати оцінку 92 бали.

8.4. Логічне виведення для недостовірних знань

Неточне виведення є логічним виведенням в умовах неточності (недостовірності) знань. Розглянемо деякі визначення **теорії ймовірностей**: $p(A)$ – ймовірність події A , $0 \leq p(A) \leq 1$, неможлива подія $p(A)=0$, достовірна подія $p(A)=1$.

Ймовірність події A визначається відношенням кількості сприятливих наслідків до їх загальної кількості (множини елементарних наслідків Ω). Наприклад, якщо кидається кубик, на кожній грані якого є числа від 1 до 6, то ймовірність випадання парного $p(A) = 3/6 = 1/2$, ($\Omega = \{1,2,3,4,5,6\}$; $\Omega_A = \{2,4,6\}$).

Протилежна подія: $p(\bar{A}) = 1 - p(A)$.

Частотне визначення ймовірності полягає в наступному. Нехай проведено n експериментів, $K_A(n)$ – кількість випадків, коли подія A відбулася. Тоді:

$$p(A) = \lim_{n \rightarrow \infty} \frac{K_A(n)}{n}. \quad (8.11)$$

$p(A \cup B)$ – ймовірність, що буде подія A або B .

$p(A \cap B)$ – ймовірність події A і B .

$$p(A \cup B) = p(A) + p(B) - p(A \cap B).$$

Для несумісних подій: $p(A \cup B) = p(A) + p(B)$.

Умовна ймовірність: $p(A|B)$ – ймовірність появи події A за умови, що B відбулася.

Події незалежні, якщо $p(A|B) = p(A)$.

Нехай $H_1 \dots H_n$ – повна група подій, де $\sum p(H) = 1$.

Тоді справедлива **формула повної ймовірності**

$$p(A) = \sum_{i=1}^n p(H_i) \cdot p(A|H_i). \quad (8.12)$$

8.5. Нечітке логічне виведення за алгоритмом Мамдані

Нечітке логічне виведення засноване на апараті нечітких множин [14-21]. Нечітким виведенням називається апроксимація залежностей між величинами за допомогою нечітких правил (Якщо – тоді) та нечітких логічних операцій. Сукупність нечітких правил (Якщо – тоді) утворюють нечітку базу знань, яка описує певну предметну область. Частина правила «якщо» називається антецедентом або посилюючою, а частина правила «тоді» – консеквентом або висновком. На даний час застосовуються різноманітні бази знань, наприклад класифікаційна база знань, база знань Мамдані, база знань Сугено, база знань Ларсена, база знань Цукамото та ін. Розглянемо детальніше базу знань Мамдані.

В базі знань Мамдані умови (антецеденти) і висновки (консеквенти) задано нечіткими множинами. Таку базу знань можна трактувати як розбиття простору ознак на зони з нечіткими межами, в кожній з яких функція належності приймає нечітке значення.

Наведемо для прикладу фрагмент нечіткої бази знань Мамдані про залежність рівня безпомилковості (y) людини-оператора від тривалості роботи (x_1) та напруженості роботи (x_2), яка містить такі правила [15]:

- 1) якщо x_1 = Мала, тоді y = Низький;
- 2) якщо x_1 = Середня та x_2 = Середня, тоді y = Високий;
- 3) якщо x_1 = Велика та x_2 = Низька, тоді y = Низький;
- 4) якщо x_1 = Велика та x_2 = Велика, тоді y = Низький;
- 5) якщо x_1 = Велика та x_2 = Середня, тоді y = Середня.

Тривалість роботи (x_1) є носієм нечітких множин T_1 (Мала тривалість роботи), T_2 (Середня тривалість роботи) і T_3 (Велика тривалість роботи), відповідно такі нечіткі множини описуються функціями належності $\mu_{T_1}(x_1)$, $\mu_{T_2}(x_1)$ та $\mu_{T_3}(x_1)$. Лінгвістична змінна «Тривалість роботи», яка визначається змінною x_1 , має терми «Мала», «Середня» і «Велика».

Напруженість роботи (x_2) є носієм нечітких множин H_1 (Низька напруженість роботи), H_2 (Середня напруженість роботи) і H_3 (Велика напруженість роботи), відповідно такі нечіткі множини описуються функціями належності $\mu_{H_1}(x_2)$, $\mu_{H_2}(x_2)$ та $\mu_{H_3}(x_2)$.

Рівень безпомилковості (y) є носієм нечітких множин B_1 (Низький рівень безпомилковості), B_2 (Середній рівень безпомилковості) і B_3 (Високий рівень безпомилковості), відповідно такі нечіткі множини описуються функціями належності $\mu_{B_1}(y)$, $\mu_{B_2}(y)$ та $\mu_{B_3}(y)$.

Нечітке логічне виведення працює так. На входи системи подаються відомі значення вхідних змінних x_1 та x_2 , на основі яких визначаються значення функції належності $\mu_T(x_1)$ та $\mu_H(x_2)$ для відповідних термів. Далі на основі правил нечіткої бази знань обчислюються значення функції належності $\mu_B(y)$ за формулою

$$\mu_B(y) = \min(\mu_T(x_1), \mu_H(x_2)). \quad (8.13)$$

Після цього виконується композиція нечітких множин (B_1 , B_2 , B_3), у результаті чого обчислюється нечітка множина B_C , для якої

$$\mu_{B_C}(y) = \max(\mu_{B_1}(y), \mu_{B_2}(y), \mu_{B_3}(y)). \quad (8.14)$$

На основі нечіткої множини B_C обчислюється її чітке значення методом центра тяжіння.

Опишемо за допомогою нечітких множин рівень вуличного (зовнішнього) освітлення для системи керування освітленням «розумного будинку». Рівень зовнішнього освітлення позначимо змінною L_K , яка описує освітленість поверхні об'єктів в умовних одиницях (від 0 до 100). При цьому значенню $L_K = 0$ відповідає освітлення 0.3 лк (у люксах) для повного Місяця, значенню $L_K = 45$ відповідає освітлення 100 лк для дуже похмурого дня, значенню $L_K = 100$ відповідає освітлення 10 000 лк для ясного дня (в тіні). Рівень зовнішнього освітлення отримується за допомогою фоточутливих сенсорів і нормується у діапазоні від 0 до 100. Введемо нечіткі множини Z_1 (Зовнішнє освітлення низьке), Z_2 (Зовнішнє освітлення середнє), Z_3 (Зовнішнє освітлення високе). Тоді нечіткі множини (Z_1 , Z_2 , Z_3) опишемо лінійними або трикутними функціями належності $\mu_{Z_1}(L_K)$, $\mu_{Z_2}(L_K)$ та $\mu_{Z_3}(L_K)$ (рис. 8.2).

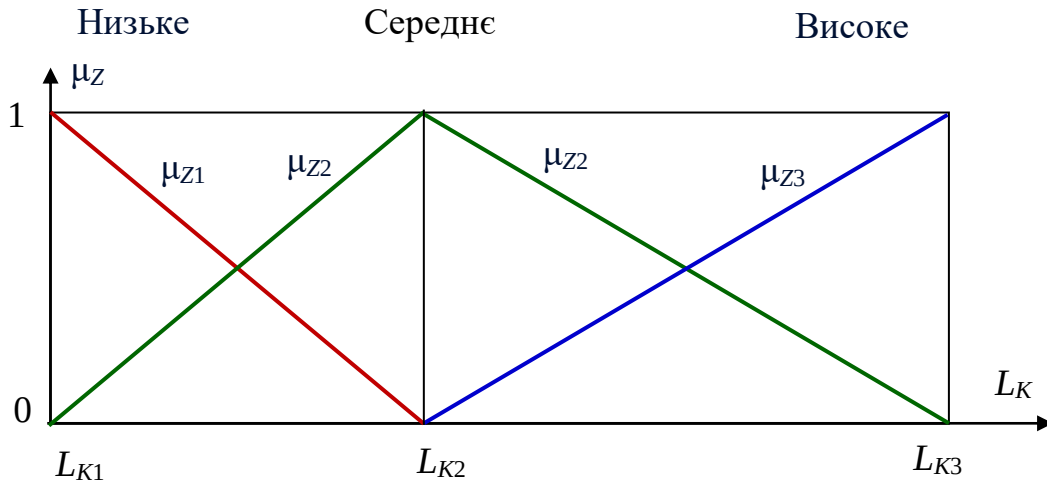


Рис. 8.2. Функції належності нечітких множин Z_1 , Z_2 , Z_3 , які описують низьке, середнє і високе зовнішнє освітлення відповідно; L_K – рівень зовнішнього освітлення; $L_{K1} = 0$, $L_{K2} = 45$, $L_{K3} = 100$

Змінній L_K відповідає логістична змінна «Рівень зовнішнього освітлення», яка має три терми «Низьке», «Середнє» і «Високе». Згідно з описаною моделлю (рис. 8.2) функції належності нечітких множин Z_1 , Z_2 , Z_3 описуються формулами:

$$\mu_{Z1}(L_K) = \begin{cases} 1; L_K < L_{K1} \\ -\frac{(L_K - L_{K1})}{L_{K2} - L_{K1}} + 1; L_{K1} \leq L_K \leq L_{K2} \\ 0, L_K > L_{K2} \end{cases}, \quad (8.14)$$

$$\mu_{Z2}(L_K) = \begin{cases} 0, L_K < L_{K1} \\ \frac{(L_K - L_{K1})}{(L_{K2} - L_{K1})}, L_{K1} \leq L_K \leq L_{K2} \\ -\frac{(L_K - L_{K2})}{(L_{K3} - L_{K2})} + 1, L_{K2} \leq L_K \leq L_{K3} \\ 0, L_K > L_{K3} \end{cases}, \quad (8.15)$$

$$\mu_{Z3}(L_K) = \begin{cases} 0; L_K < L_{K2} \\ \frac{(L_K - L_{K2})}{L_{K3} - L_{K2}}; L_{K2} \leq L_K \leq L_{K3} \\ 1, L_K > L_{K3} \end{cases}. \quad (8.16)$$

Опишемо за допомогою нечітких множин рівень руху у приміщеннях «розумного будинку», обчислений на основі кадрів відеопотоку. Рівень руху позначимо змінною L_M , який описує рівень (інтенсивність) руху осіб в умовних одиницях (від 0 до 100). При цьому значенню $L_M = 0$ відповідає відсутність руху, значенню $L_M = 50$

відповідає середній рівень руху (в приміщення іноді заходить одна особа), значенню $L_M = 100$ відповідає високий рівень руху (кілька осіб часто заходять у приміщення). Введемо нечіткі множини M_1 (Рівень руху низький), M_2 (Рівень руху середній), M_3 (Рівень руху високий). Тоді нечіткі множини (M_1 , M_2 , M_3) опишемо лінійними або трикутними функціями належності $\mu_{M_1}(L_M)$, $\mu_{M_2}(L_M)$ та $\mu_{M_3}(L_M)$ (рис. 8.3).

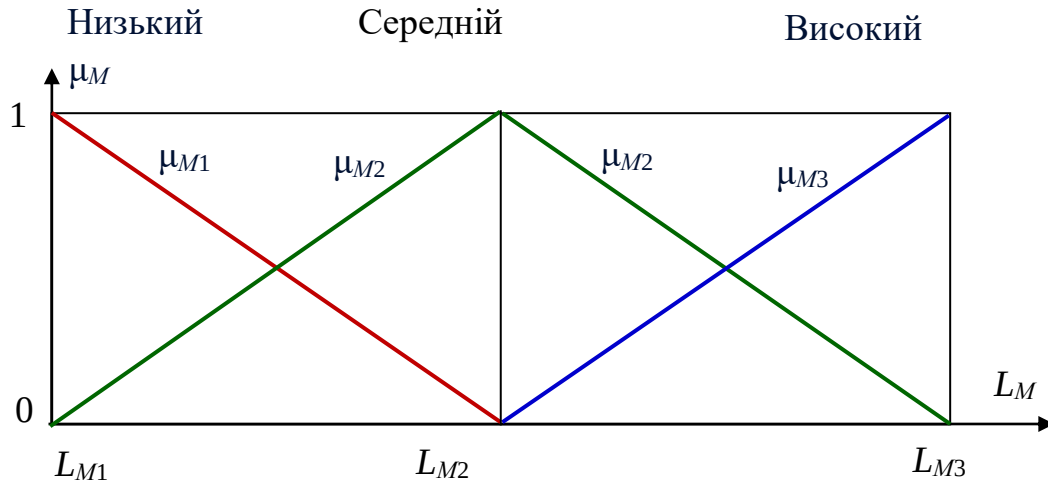


Рис. 8.3. Функції належності нечітких множин M_1 , M_2 , M_3 , які описують низький, середній і високий рівень руху в приміщеннях відповідно; L_M – рівень руху в приміщеннях;
 $L_{M1} = 0$, $L_{M2} = 50$, $L_{M3} = 100$

Змінній L_M відповідає логістична змінна «Рівень руху в приміщеннях», яка має три терми «Низький», «Середній» і «Високий». Згідно з описаною моделлю (рис. 8.3) функції належності нечітких множин M_1 , M_2 , M_3 описуються формулами:

$$\mu_{M_1}(L_M) = \begin{cases} 1; L_M < L_{M1} \\ -\frac{(L_M - L_{M1})}{L_{M2} - L_{M1}} + 1; L_{M1} \leq L_M \leq L_{M2} \\ 0, L_M > L_{M2} \end{cases}, \quad (8.17)$$

$$\mu_{M_2}(L_M) = \begin{cases} 0, L_M < L_{M1} \\ \frac{(L_M - L_{M1})}{(L_{M2} - L_{M1})}, L_{M1} \leq L_M \leq L_{M2} \\ -\frac{(L_M - L_{M2})}{(L_{M3} - L_{M2})} + 1, L_{M2} \leq L_M \leq L_{M3} \\ 0, L_M > L_{M3} \end{cases}, \quad (8.18)$$

$$\mu_{M3}(L_M) = \begin{cases} 0; L_M < L_{M2} \\ \frac{(L_M - L_{M2})}{L_{M3} - L_{M2}}; L_{M2} \leq L_M \leq L_{M3} \\ 1, L_M > L_{M3} \end{cases} \quad (8.19)$$

Опишемо за допомогою нечітких множин потужність освітлювальних пристроїв у приміщенні «розумного будинку». Потужність освітлювальних пристроїв позначимо змінною L_P (у Вт). При цьому значенню $L_P = L_{P1} = 0$ Вт відповідає вимкнення освітлювальних пристроїв, значенню $L_P = L_{P2}$ (наприклад, $L_{P2} = 15$ Вт) відповідає нижча середньої потужність освітлювальних приладів, значенню $L_P = L_{P3}$ (наприклад, $L_{P3} = 35$ Вт) відповідає середня потужність освітлювальних приладів, значенню $L_P = L_{P4}$ (наприклад, $L_{P4} = 50$ Вт) відповідає вища середньої потужність освітлювальних приладів, значенню $L_P = L_{P5}$ (наприклад, $L_{P5} = 70$ Вт) відповідає висока потужність освітлювальних приладів. Введемо нечіткі множини $P1$ (Потужність низька), $P2$ (Потужність нижча середньої), $P3$ (Потужність середня), $P4$ (Потужність вища середньої), $P5$ (Потужність висока). Тоді нечіткі множини ($P1, P2, P3, P4, P5$) опишемо лінійними або трикутними функціями належності $\mu_{P1}(L_P)$, $\mu_{P2}(L_P)$, $\mu_{P3}(L_P)$, $\mu_{P4}(L_P)$ та $\mu_{P5}(L_P)$ (рис. 8.4).

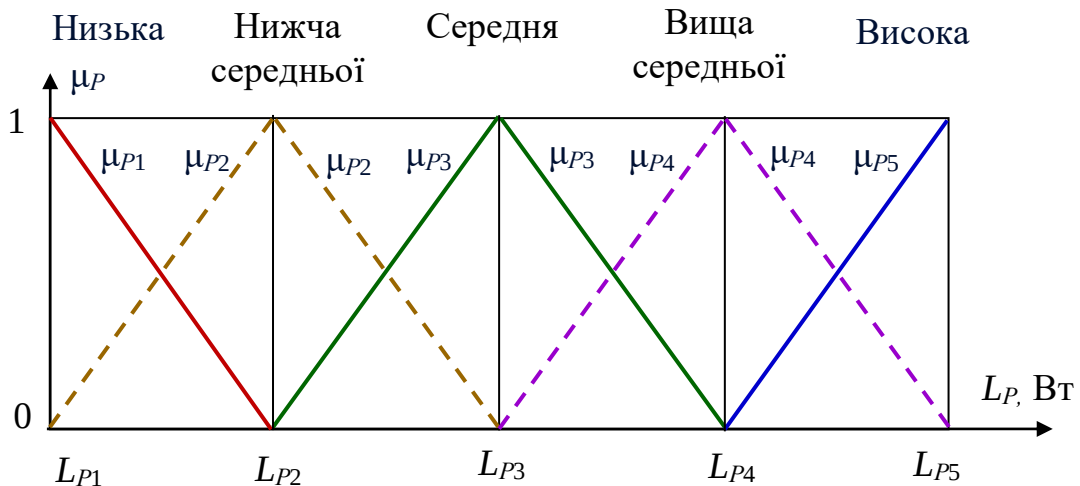


Рис. 8.4. Функції належності нечітких множин $P1, P2, P3, P4, P5$, які описують низьку, нищу середньої, середню, вищу середньої і високу потужність освітлювальних приладів у приміщеннях відповідно; L_P – потужність освітлювальних приладів у приміщеннях; наприклад,

$$L_{P1} = 0 \text{ Вт}, L_{P2} = 15 \text{ Вт}, L_{P3} = 35 \text{ Вт}, L_{P4} = 50 \text{ Вт}, L_{P5} = 70 \text{ Вт}$$

Змінній L_P відповідає логістична змінна «Потужність освітлювальних приладів», яка має 5 термів «Низька», «Нижча середньої», «Середня», «Вища середньої» і «Висока».

Згідно з описаною моделлю (рис. 8.4) функції належності нечітких множин P_1, P_2, P_3, P_4, P_5 описуються формулами:

$$\mu_{P_1}(L_P) = \begin{cases} 1; L_P < L_{P1} \\ -\frac{(L_P - L_{P1})}{L_{P2} - L_{P1}} + 1; L_{P1} \leq L_P \leq L_{P2}, \\ 0, L_P > L_{P2} \end{cases} \quad (8.20)$$

$$\mu_{P_2}(L_P) = \begin{cases} 0, L_P < L_{P1} \\ \frac{(L_P - L_{P1})}{(L_{P2} - L_{P1})}, L_{P1} \leq L_P \leq L_{P2} \\ -\frac{(L_P - L_{P2})}{(L_{P3} - L_{P2})} + 1, L_{P2} \leq L_P \leq L_{P3} \\ 0, L_P > L_{P3} \end{cases}, \quad (8.21)$$

$$\mu_{P_3}(L_P) = \begin{cases} 0, L_P < L_{P2} \\ \frac{(L_P - L_{P2})}{(L_{P3} - L_{P2})}, L_{P2} \leq L_P \leq L_{P3} \\ -\frac{(L_P - L_{P3})}{(L_{P4} - L_{P3})} + 1, L_{P3} \leq L_P \leq L_{P4} \\ 0, L_P > L_{P4} \end{cases}, \quad (8.22)$$

$$\mu_{P_4}(L_P) = \begin{cases} 0, L_P < L_{P3} \\ \frac{(L_P - L_{P3})}{(L_{P4} - L_{P3})}, L_{P3} \leq L_P \leq L_{P4} \\ -\frac{(L_P - L_{P4})}{(L_{P5} - L_{P4})} + 1, L_{P4} \leq L_P \leq L_{P5} \\ 0, L_P > L_{P5} \end{cases}, \quad (8.23)$$

$$\mu_{P_5}(L_P) = \begin{cases} 0; L_P < L_{P4} \\ \frac{(L_P - L_{P4})}{L_{P5} - L_{P4}}; L_{P4} \leq L_P \leq L_{P5}. \\ 1, L_P > L_{P5} \end{cases} \quad (8.24)$$

На основі вищенаведених нечітких множин (рис. 8.2 – рис. 8.4) створено нечітку базу знань Мамдані про залежність потужності освітлювальних приладів у приміщенні «розумного будинку» L_P (рис. 8.4) від рівня зовнішнього освітлення L_K (рис. 8.2) та рівня руху в приміщеннях L_M (рис. 8.3), яка містить такі правила:

- 1) якщо $L_M = \text{«Низький»}$ та $L_K = \text{«Низький»}$, тоді $L_P = \text{«Низька»}$;
- 2) якщо $L_M = \text{«Низький»}$ та $L_K = \text{«Середній»}$, тоді $L_P = \text{«Низька»}$;
- 3) якщо $L_M = \text{«Низький»}$ та $L_K = \text{«Високий»}$, тоді $L_P = \text{«Низька»}$;
- 4) якщо $L_M = \text{«Середній»}$ та $L_K = \text{«Низький»}$, тоді $L_P = \text{«Вища середньої»}$;
- 5) якщо $L_M = \text{«Середній»}$ та $L_K = \text{«Середній»}$, тоді $L_P = \text{«Середня»}$;
- 6) якщо $L_M = \text{«Середній»}$ та $L_K = \text{«Високий»}$, тоді $L_P = \text{«Нижча середньої»}$;
- 7) якщо $L_M = \text{«Високий»}$ та $L_K = \text{«Низький»}$, тоді $L_P = \text{«Висока»}$;
- 8) якщо $L_M = \text{«Високий»}$ та $L_K = \text{«Середній»}$, тоді $L_P = \text{«Висока»}$;
- 9) якщо $L_M = \text{«Високий»}$ та $L_K = \text{«Високий»}$, тоді $L_P = \text{«Висока»}$.

На основі правил №1-№9 вищеописаної нечіткої бази знань обчислюються значення функції належності $\mu_P(L_P)$ за формулою

$$\mu_P(L_P) = \min(\mu_Z(L_K), \mu_M(L_M)). \quad (8.25)$$

Після цього виконується композиція нечітких множин (P_1, P_2, P_3, P_4, P_5), у результаті чого обчислюється нечітка множина P_C , для якої

$$\mu_{P_C}(L_P) = \max(\mu_{P_1}(L_P), \dots, \mu_{P_5}(L_P)). \quad (8.26)$$

На основі нечіткої множини P_C обчислюється її чітке значення L_{PC} методом центра тяжіння (2.2) за формулою

$$L_{PC} = \frac{\sum L_P \cdot \mu_{P_C}(L_P)}{\sum \mu_{P_C}(L_P)}. \quad (8.27)$$

Система Matlab дозволяє працювати з нечіткими множинами як програмно, так і за допомогою редактора `fis` (рис. 8.5). Для завантаження `fis`-редактор потрібно ввести в командному рядку слово «fuzzy». За допомогою відповідних бібліотек забезпечується робота з нечіткими множинами також на мові Python.

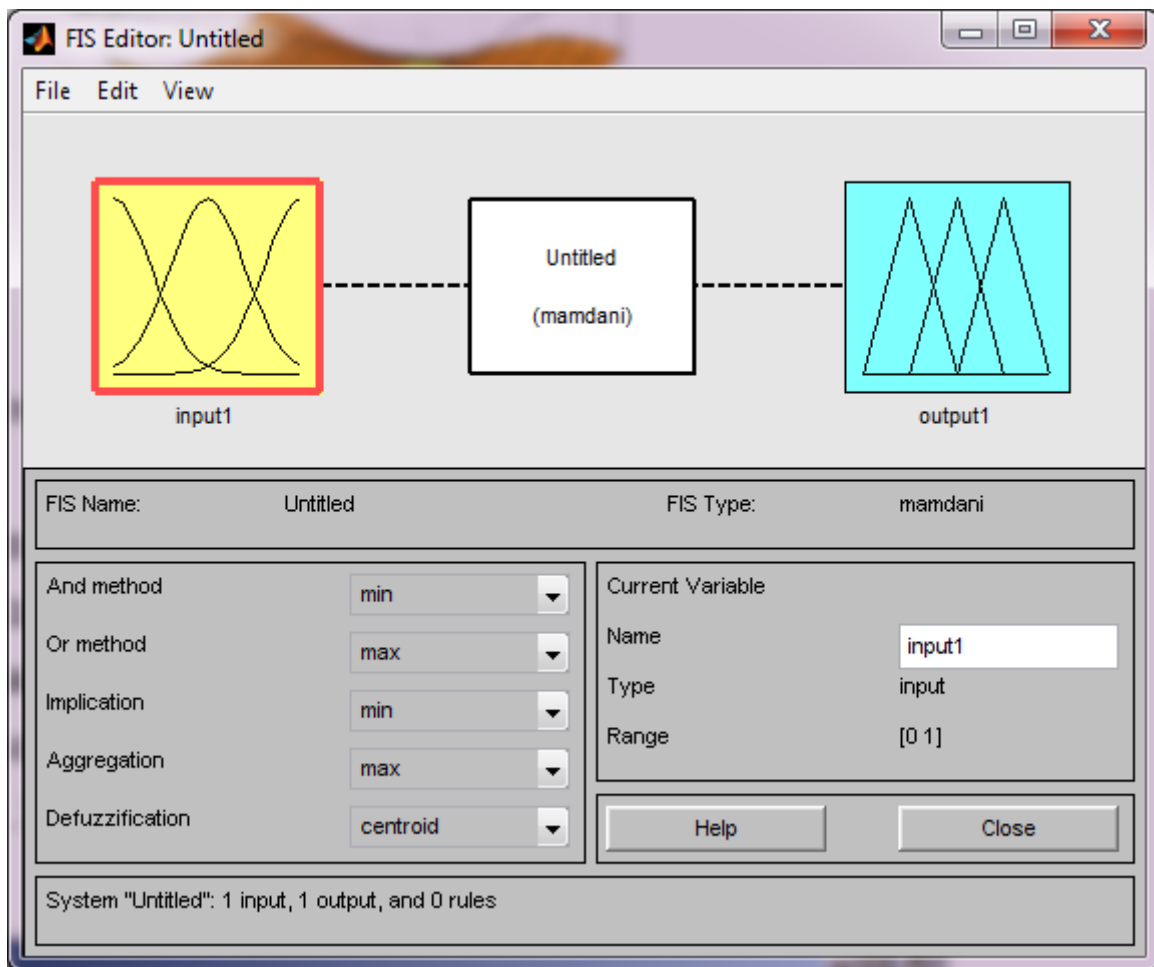


Рис. 8.5. Вікно редактора fis, призначеного для роботи з нечіткими множинами в системі Matlab

При нечіткому логічному виведенні як вхідні дані можуть використовуватися сигнали з сенсорів, а виходами можуть бути сигнали керування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Artificial Intelligence. A Modern Approach : Fourth Edition / Ed. S. Russell, P. Norvig. – Pearson Education, 2021. – 1069 p. – <http://aima.cs.berkeley.edu>.
2. Luger G.F. Artificial intelligence: structures and strategies for complex problem solving: 6th ed / G.F. Luger. – Pearson Education, 2009. – 754 p. – http://www.uoitc.edu.iq/images/documents/informatics-institute/exam_materials/artificial%20intelligence%20structures%20and%20strategies%20for%20%20complex%20problem%20solving.pdf.
3. Глибовець М.М. Штучний інтелект : підруч. [для студ. вищ. навч. закл.] / М.М. Глибовець, О.В. Олецький. – К. : КМ Академія, 2002. – 336 с.
4. Зайченко Ю.П. Основи проектування інтелектуальних систем. Навчальний посібник / Ю.П. Зайченко. – К.: Слово, 2004. – 352 с.
5. Литвин В.В. Інтелектуальні системи : підручник / В.В. Литвин, В.В. Пасічник, Ю.В. Яцишен. – Львів: Новий світ, 2009. – 405 с.
6. Савченко А.С. Методи та системи штучного інтелекту: Навч. посібник / А.С. Савченко, О. О. Синельников. – К. : НАУ, 2017. – 190 с.
7. Шаховська Н. Б. Системи штучного інтелекту: навчальний посібник / Н. Б. Шаховська, Р. М. Камінський, О. Б. Вовк. – Львів : Видавництво Львівської політехніки, 2018. – 392 с.
8. Intelligent Imaging and Analysis / Ed. DaeEun Kim, Dosik Hwang. – Switzerland, Basel: MDPI, 2020. – 492 p. URL: <https://mdpi.com/books/pdfview/book/2059>. DOI: 10.3390/books978-3-03921-921-6.
9. Zhu L. Towards Image Classification with Machine Learning Methodologies for Smartphones / L. Zhu, P. Spachos // Machine Learning and Knowledge Extraction. – 2019, No. 1(4). – P. 1039-1057.
10. Кутковецький В.Я. Розпізнавання образів: Навчальний посібник / В.Я. Кутковецький. – Миколаїв: Вид-во МДГУ ім. П.Могили, 2017. – 420 с.
11. Штовба С.Д. Інтелектуальні технології ідентифікації залежностей. Лабораторний практикум : електронний навчальний посібник / С.Д. Штовба, В.В. Мазуренко. – Вінниця : ВНТУ, 2014. – 113 с.

12. Gonzalez R. Digital image processing / R. Gonzalez, R. Woods. - 4th edition, Pearson/ Prentice Hall, NY, 2018. - 1192 p.
13. Krigg S. Computer Vision Metrics. Survey, Taxonomy, and Analysis. – Spredd Open, 2014. – 498 p.
14. Russ J.C. The Image Processing. Handbook [Sixth Edition]. – Taylor & Francis Group, LLC, 2011. – 853 p.
15. Сергеев-Горчинський О.О. Інтелектуальний аналіз даних: Комп'ютерний практикум: навч. посібник / О.О. Сергеев-Горчинський, Г.В. Іщенко. – Київ : КПІ ім. Ігоря Сікорського, 2018. – 73 с.
16. Черняк О.І. Інтелектуальний аналіз даних: Підручник / О.І. Черняк, П.В.Захарченко. - Київський національний університет ім. Т. Шевченка. –К. : Знання, 2014. – 599 с.
17. Balovsyak S. Analysis of X-Ray Moiré Images Using Artificial Neural Networks / S.Balovsyak, I. Fodchuk, Kh.Odaiska, Yu. Roman, E.Zaitseva // IntelITSIS 2022: 3rd International Workshop on Intelligent Information Technologies and Systems of Information Security, March 23–25, 2022. – Khmelnytskyi, Ukraine, CEUR Workshop Proceedings, 2022. – P. 187-197. – <http://ceur-ws.org/Vol-3156/paper12.pdf>
18. Chollet F. Xception: Deep Learning with Depthwise Separable Convolutions. [Електронний ресурс]. URL: https://openaccess.thecvf.com/content_cvpr_2017/papers/Chollet_Xception_Deep_Learning_CVPR_2017_paper.pdf.
19. Geron A. Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow /A. Geron. – O'Reilly Media, Inc., 2019. – 510 p.
20. Navin Kumar Manaswi. Deep Learning with Applications Using Python. – Apress, 2018. – 219 p.
21. Березький О.М. Класифікація гістологічних та цитологічних зображень на основі згорткових нейронних мереж / О.М. Березький, О.Й. Піцун, А.Р. Боднар, Т.М. Долинюк // Штучний інтелект. – 2017. – № 1. – С. 29-37.
22. Руденко О.Г. Штучні нейронні мережі. Навч. посібник / О.Г. Руденко, Є.В. Боданський . – Харків: СНІТ, 2006. – 404 с.
23. Субботін С. О. Нейронні мережі : теорія та практика: навч. посіб. / С. О. Субботін. – Житомир : Вид. О. О. Євенок, 2020. – 184 с. URL: http://eir.zntu.edu.ua/bitstream/123456789/6800/1/Subbotin_Neural.pdf.

24. Баловсяк С.В. Багаторівневі методи оброблення електронно-дифракційних та Х-променевих сигналів у комп'ютеризованих інформаційно-вимірювальних системах : дис. докт. техн. наук: 05.13.05 – Комп'ютерні системи та компоненти. : захищ. 7.06.2019. затв. 15.10.2019 / Сергій Васильович Баловсяк. – Вінниця, 2019. – 452 с.
25. Баловсяк С. В. Суміщення зображень об'єктів з використанням генетичних та градієнтних алгоритмів / С.В. Баловсяк, І.М. Фодчук // Комп'ютинг. – 2013. – Т. 12, Вип. 2. – С. 160-169.
26. Локажук В.М. Інтелектуальне діагностування мікропроцесорних пристроїв та систем: Навч. посібник для вузів / В.М. Локажук, О.В. Поморова, А.О. Домінов. – Хмельницький – Київ: Такі справи. – 2001. – 286 с.
27. Hooda D.S. Fuzzy Logic Models. An Introduction / D.S. Hooda, Vivek Raich. – U.K., Oxford: Alpha Science International Ltd, 2017. – 408 p.