Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут»

В. О. Апостолюк, О. С. Апостолюк

Інтелектуальні системи керування

Конспект лекцій

Затверджено Методичною радою НТУУ «КПІ»

Київ
НТУУ «КПІ»
2008

УДК 004:89 (075.8)
ББК 32.813я73
А76

Гриф надано Методичною радою НТУУ «КПІ»
(Протокол № 8 від 19.04.2007 р.)

Рецензент *Л. М. Рижков,* д-р техн. наук, проф.,
Національний технічний університет України
«Київський політехнічний інститут»

Відповідальний редактор *П. М. Бондар,* канд. техн. наук, проф.,
Національний технічний університет України
«Київський політехнічний інститут»

Апостолюк В. О.

А76 Інтелектуальні системи керування: конспект лекцій [Текст] / В. О. Апостолюк, О. С. Апостолюк. – К.: НТУУ «КПІ», 2008. – 88 с. – Бібліогр.: с. 84 –85. – 50 пр.

Викладено найбільш важливі аспекти сучасних технологій створення інтелектуальних систем керування, зокрема генетичні алгоритми для вирішення завдань глобальної багатопараметричної оптимізації, теорія нейронних мереж, теорія нечітких множин, методи синтезу нечітких регуляторів, а також методи застосування нейронних мереж та нечітких регуляторів у системах керування.

Для студентів спеціальності «Прилади та системи керування літальними апаратами».

УДК 004:89 (075.8)
ББК 32.813я73

ЗМІСТ

1. УВЕДЕННЯ ДО ІНТЕЛЕКТУАЛЬНИХ СИСТЕМ КЕРУВАННЯ

1.1. Загальні принципи алгоритмізації та побудови систем керування

Об’єктом керування (ОК) називають частину навколишнього світу, на стан якого можна впливати цілеспрямовано, тобто керувати нею. Під керуванням розуміють процес організації цілеспрямованої дії на ОК, що переводить ОК у потрібний стан.

Керування реалізує система керування (СК). Під СК розуміють всі необхідні алгоритми оброблення інформації і засоби їх реалізації, об’єднані для досягнення мети керування.

У загальному випадку складний ОК може містити низку функціонально підпорядкованих підсистем. Ієрархія їх підпорядкування зумовлює декомпозицію вихідних цілей і завдань керування на рекурсивну послідовність вкладених складових. У кінцевому підсумку такий поділ припускає багаторівневу організацію СК. До того ж структура СК складним об’єктом зазвичай містить стратегічний, тактичний та виконавчий (привідний) рівні (див. рис. 1.1).

Кожен рівень у наведеній ієрархії виконує певні функції:

Стратегічний Аналіз глобальної ситуації, прийняття рішень про подальші дії, гарантування безпеки, вибір цілей для тактичного рівня.

Тактичний Аналіз локальної ситуації, прокладання маршруту з урахуванням перешкод, орієнтування на маршруті, формування цілей керування для виконавчого рівня на основі закладених алгоритмів.

Виконавчий Реалізацію цілей керування, що надходять від тактичного рівня, із забезпеченням належної якості.

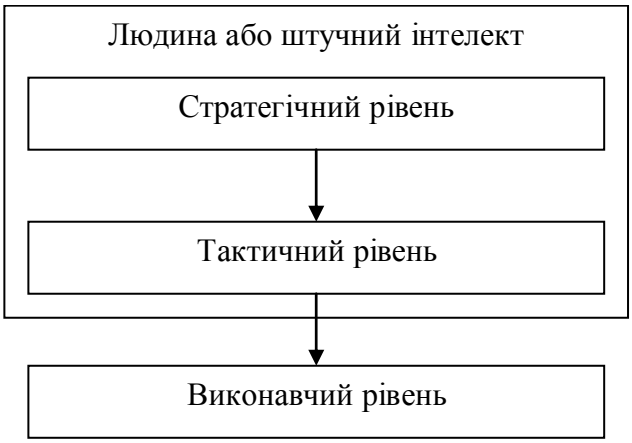


Рис. 1.1. Узагальнена ієрархія систем керування

Виконання цих функцій забезпечують різними методами і алгоритмами. У традиційних системах розв'язування задач стратегічного і тактичного рівнів покладають на людину-оператора.

Розглянемо детальніше структуру виконавчого рівня СК (рис. 1.2).

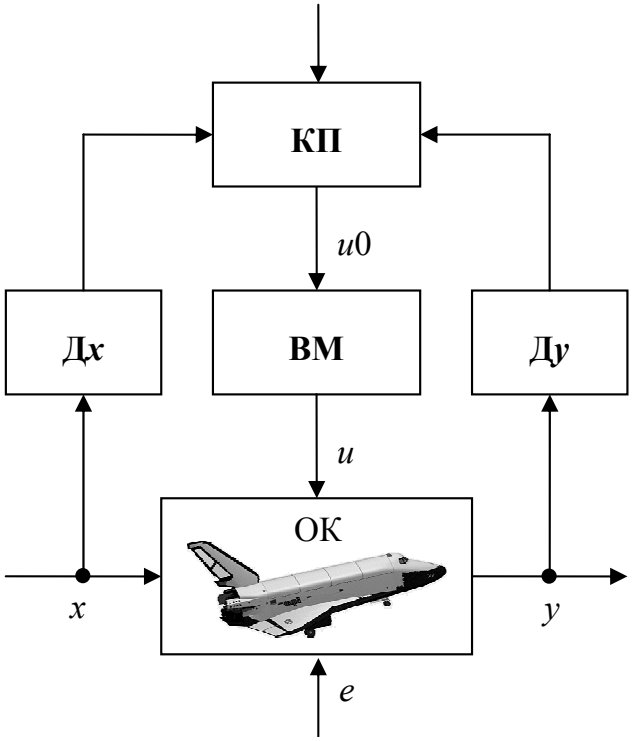


Рис. 1.2. Структура систем керування:
КП – керувальний пристрій; ВМ – виконавчий механізм; ОК – об'єкт керування; *x* – стан середовища; *y* – стан ОК; *e* – неспостережувані збурення; *u*0 – керувальний сигнал; *u* – керувальний вплив; Д*x* і Д*y* – датчики стану середовища і об'єкта керування відповідно

Для цілеспрямованого функціонування керуювального пристрою йому, окрім інформації з датчиків Д*х* і Д*у*, потрібно також задати мету керування, яка надходить з тактичного рівня, і алгоритм керування, тобто спосіб досягнення заданої мети.

1.2. Труднощі побудови систем керування

Труднощі побудови СК визначаються складністю ОК. Наведемо основні ознаки складного ОК.

Брак математичного опису. Немає алгоритму обчислення стану *у* об'єкта за спостереженнями його входів (керованого *и* і некерованого) та спостережуваного стану середовища *х*.

Недетермінованість (стохастичність). Зумовлена переважно складністю об'єкта і пов'язаними з цим численними другорядними процесами, якими нехтують під час побудови математичної моделі.

Нестаціонарність. Виявляється у дрейфі характеристик об'єкта.

1.3. Показники якості керування

Якість керування, що забезпечується системами регулювання, визначається поведінкою системи як в усталеному режимі, так і під час перехідного процесу (див. рис. 1.3).

Основні показники перехідного процесу:

Перерегулювання

$$\text{у} = \frac{y_{\max} - y_0}{y_0} \cdot 100,$$

де σ вимірюють у відсотках.

Тривалість перехідного процесу T_0 – час з моменту виникнення стрибка задавального впливу, після якого відхилення регульованої величини не перевищує заданого припустимого значення Δ*е* (рис. 1.3).

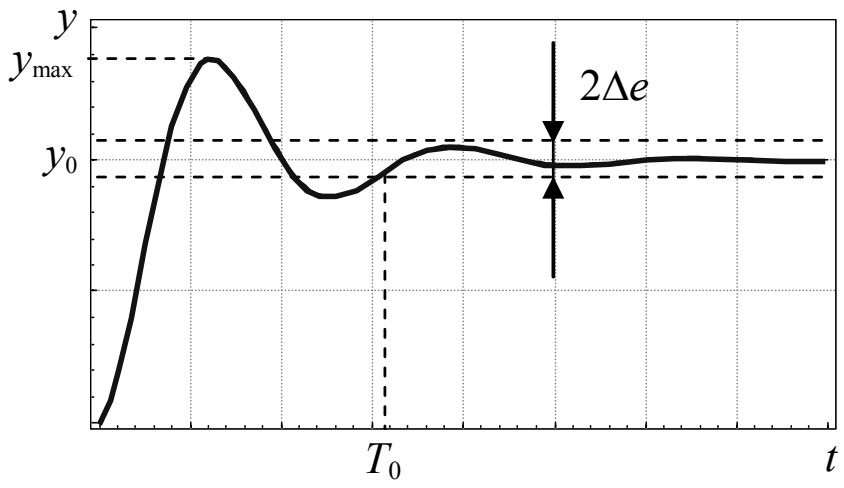


Рис. 1.3. Характеристики перехідного процесу

Якість керування в усталеному режимі визначається інтегральними оцінками, які дають змогу характеризувати перехідний процес у системі одним числом. На практиці найбільшого поширення набули такі інтегральні оцінки.

Квадратична інтегральна оцінка:

$$I = \int_0^{+\infty} e(t)^2 \, dt,$$

де $e(t) = x(t) - y(t)$ – похибка регулювання.

Поліпшена квадратична інтегральна оцінка:

$$I = \int_0^{+\infty} \left(e(t)^2 + T^2 \left[\frac{d}{dt} e(t) \right]^2 \right) dt.$$

Перша із вказаних квадратичних оцінок тим менша, чим перехідний процес в розглядуваній системі ближчий до задавального впливу, а друга – чим він ближчий до перехідного процесу на виході інерційної ланки зі сталою часу T.

1.4. Типові математичні моделі об’єктів керування

На практиці точна модель ОК зазвичай невідома. Водночас для оцінних розрахунків корисно мати модель, що хоча б наближено описувала властивості об’єкта.

Досвід показує, що в першому наближенні математичний опис багатьох промислових об’єктів, наприклад електричних та електромеханічних, можна подати у вигляді структури – моделі Гаммерштейна (рис. 1.4).

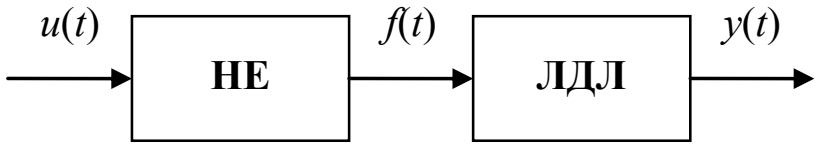


Рис. 1.4. Типова структура об’єкта керування:
НЕ – безінерційний нелінійний елемент; ЛДЛ – лінійна динамічна ланка

Під НЕ розуміють нелінійність типу «*зона нечутливості з обмеженням*» (рис. 1.5).

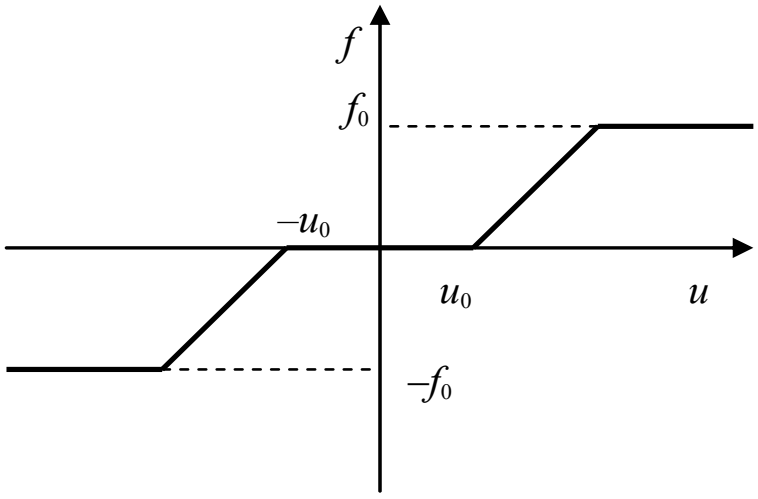


Рис. 1.5. Характеристика нелінійного елемента

Передатна функція ЛДЛ зазвичай має вигляд

$$W(p)=\frac{Y(p)}{F(p)}=\frac{ke^{-p\tau}}{(1+pT_1)(1+pT_2)},$$

де $F(p)$ і $Y(p)$ – зображення за Лапласом (за нульових початкових умов) вхідного та вихідного сигналів ЛДЛ відповідно для об'єктів із самовирівнюванням

$$W(p)=\frac{Y(p)}{F(p)}=\frac{ke^{-p\tau}}{(1+pT)p},$$

та для об'єктів без самовирівнювання.

1.5. Традиційні лінійні регулятори

Розглянемо замкнену (зі зворотним зв'язком) одноконтурну систему автоматичного регулювання (рис. 1.6).

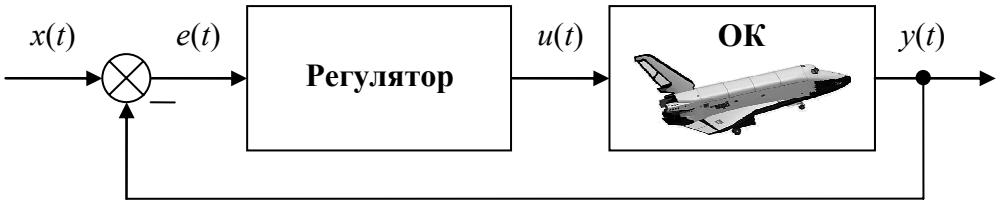


Рис. 1.6. Структура системи автоматичного регулювання

Лінійні регулятори являють собою динамічні ланки, що описані лінійними диференціальними рівняннями. Найпоширенішими лінійними регуляторами є ПІД-регулятори, які мають такий математичний опис:

$$u(t)=K_P e(t)+K_I\int_0^t e(t)dt+K_D\frac{d}{dt}e(t),$$

де $u(t)$ – вихідний сигнал регулятора; $e(t)$ – його вхідний сигнал (похибка регулювання); K_P, K_I, K_D – коефіцієнти пропорційної, інтегральної та диференціальної складових відповідно.

Нехай ОК описано лінійною ланкою з передатною функцією (1.1), параметри об'єкта сталі, при цьому параметр τ досить малий. Передатна функція замкненої системи має відповідати передатній фу-

нкції ланки запізнювання зі сталою τ. При цьому регулятор буде описано рівнянням (1.2) з такими параметрами:

$$K_P = \frac{T_1 + T_2}{\tau k},\ K_I = \frac{1}{\tau k},\ K_P = \frac{T_1 T_2}{\tau k}.$$

Зауважимо, що системи з ПІД-регуляторами перевершують за своїми характеристиками системи з ПІ-регуляторами, однак ПІ-регулятори менш чутливі до зміни параметрів об'єкта і простіше настроюванні.

1.6. Інтелектуальні системи керування

Інтелектуальні СК – це СК, здатні до «розуміння» і навчання щодо ОК, збурень, зовнішнього середовища та умов роботи.

Основна відмінність інтелектуальних систем полягає в *наявності механізму системного оброблення знань*. Головна архітектурна особливість, яка відрізняє інтелектуальні СК від традиційних, – це *механізм отримання, зберігання і оброблення знань* для реалізації функцій керування.

В основу створення інтелектуальних СК покладено два узагальнені принципи:

– керування на основі аналізу зовнішніх даних, ситуацій та подій (ситуаційне керування);

– використання сучасних інформаційних технологій оброблення знань.

Розрізняють декілька сучасних інформаційних технологій, що дозволяють створювати інтелектуальні СК:

– експертні системи;

– штучні нейронні мережі (*artificial neural networks*);

– нечітка логіка (*fuzzy logic*);

– еволюційні методи і генетичні алгоритми (*genetic algorithms*).

В основу концепції інтелектуальності покладено:

– уміння працювати з формалізованими знаннями людини (експертні системи, нечітка логіка);

– властиві людині способи навчання і мислення (нейронні мережі, генетичні алгоритми).

Структурно інтелектуальні СК містять додаткові блоки, які виконують системне опрацювання знань на основі цих інформаційних технологій. Такі блоки можна виконувати або як надбудову над звичайним регулятором, налагоджуючи належним чином його параметри, або безпосередньо включатися у контур керування.

1.7. Застосування інтелектуальних систем керування

Найбільш суттєві причини поширення інтелектуальних СК такі:

– особливі якості інтелектуальних СК, зокрема мала чутливість до зміни параметрів ОК;

– те, що синтез інтелектуальної СК із застосуванням сучасних засобів апаратної та програмної підтримки часто простіший, ніж традиційних.

Є випадки, коли застосування інтелектуальних СК виправдане і дає кращий результат:

– системи регулювання, для яких модель ОК визначена лише якісно або її немає взагалі;

– як надбудова над традиційними системами для надання їм адаптивних властивостей;

– відтворення дій людини-оператора;

– системи організаційного керування верхніх (стратегічного і тактичного) рівнів.

Сфера ефективного застосування традиційних, нейромережевих та нечітких СК щодо ОК, показані на рис. 1.7.

Застосування гібридного підходу (поєднання традиційних методів керування, нечіткої логіки та нейронних мереж) дозволяє створювати СК, ефективні в усьому спектрі ситуацій, і тому межі різних підходів, показаних на рис. 1.7, вельми умовні.

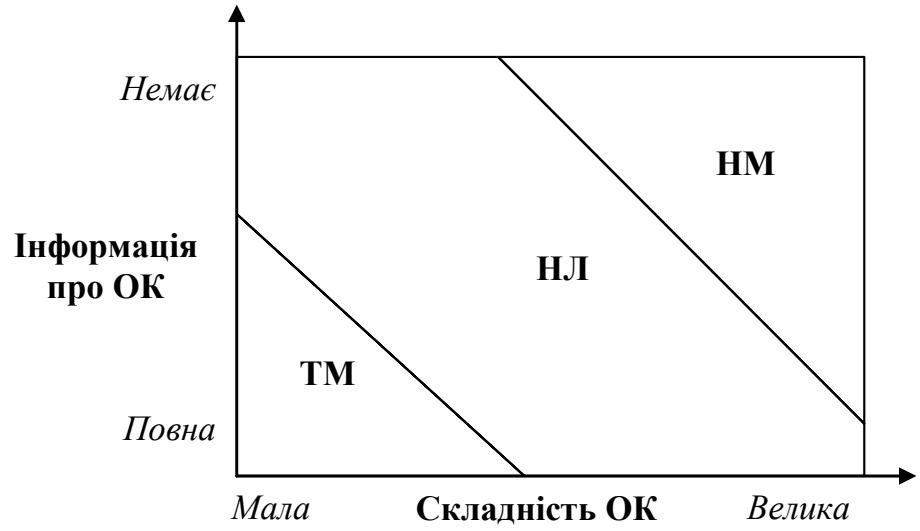


Рис. 1.7. Сфера ефективного застосування різних систем керування:
ТМ – системи, які використовують традиційні методи керування;
НЛ – системи керування з нечіткою логікою; НМ – системи керування
на нейронних мережах

2. ГЕНЕТИЧНІ АЛГОРИТМИ

2.1. Вступ та історична довідка

Поширення та популярності генетичні алгоритми (ГА) набули завдяки роботам Дж. Холланда на початку 70-х років минулого століття. Однак генетичними вони стали називатися пізніше, а 1975 року Холланд називав їх репродуктивними планами (*reproductive plan*).

Генетичні алгоритми базуються на теоретичних досягненнях синтетичної теорії еволюції, що враховує мікробіологічні механізми успадковування ознак у природних і штучних популяціях організмів, а також на нагромадженому людському досвіді у селекції тварин і рослин.

Методологія ГА ґрунтується на гіпотезі селекції, яка у загальному вигляді може бути визначена так: *чим вища пристосованість особини, тим вища ймовірність того, що у потомстві, отриманому за її участі, ознаки, що визначають пристосованість, будуть виражені ще сильніше*.

Попри свою біологічну термінологію, ГА перш за все – це потужний засіб розв’язування задач *глобальної оптимізації*. Інакше кажучи, ГА переважно застосовують для знахождення *глобальних* екстремумів функцій *багатьох* змінних. Від інших методів оптимізації ГА відрізняються таким:

– колективним пошуком екстремуму за допомогою популяції особин;

– обробляють не значення параметрів самої задачі, а їх закодовану форму;

– використовують тільки цільову функцію, а не її похідні чи будь-яку іншу додаткову інформацію;

– застосовують імовірнісні, а не детерміновані правила вибору і модифікації параметрів задачі.

Розглянемо ці аспекти детальніше.

2.2. Подання параметрів оптимізації

Нехай маємо деяку цільову функцію багатьох змінних $f(x_1, x_2, \ldots, x_n)$, для якої потрібно знайти глобальний екстремум (мінімум або максимум). Подамо незалежні змінні у вигляді ланцюжка *двійкових чисел* 1 і 0 (ланцюжка *бітів*), виділяючи для кожного параметра m бітів (рис. 2.1).

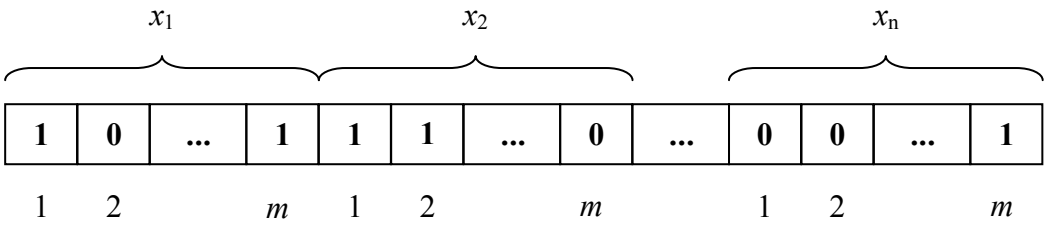


Рис. 2.1. Структура «хромосоми»

Такий ланцюжок бітів, що кодує параметри оптимізації, називають *хромосомою*, а окремі біти в ній – *генами*.

Хромосоми генеруються випадково, послідовним заповненням розрядів (генів), одразу в двійковому вигляді, і всі подальші зміни в популяції зачіпають спочатку генетичний рівень, а тільки потім наслідки цих змін аналізують у параметрах оптимізації, і ніколи навпаки.

У принципі, для декодування генетичної інформації із двійкової форми до десяткової підходить будь-який двійково-десятковий код, але найкращі результати забезпечує подання хромосом із використанням *коду Грея* (див. табл. 2.1).

Таблиця 2.1

Декодування параметрів із різних двійкових кодів

Код Грея	Двійковий код	Десяткове значення
0000	0000	0
0001	0001	1
0011	0010	2
0010	0011	3

Продовження табл. 2.1

Код Грея	Двійковий код	Десяткове значення
0110	0100	4
0111	0101	5
0101	0110	6
0100	0111	7
1100	1000	8
1101	1001	9
1111	1010	10
1110	1011	11
1010	1100	12
1011	1101	13
1001	1110	14
1000	1111	15

Від коду Грея переходимо до двійково-десяткового коду, а від нього – до натуральних цілих чисел. Відношення отриманого числа до максимального числа, доступного для кодування такою кількістю розрядів (в табл. 2.1 це число 15) і дає шукане значення зсуву змінної відносно лівої межі a_i допустимого діапазону її зміни, нормованого на ширину діапазону:

$$x_i = a_i + (b_i - a_i)\frac{d_i}{2^m - 1}, \quad x_i \in [a_i, b_i],$$

де d_i – ціле десяткове число, отримане переведенням коду з двійкового в десяткове подання; a_i – нижня межа допустимих значень для параметра x_i, а b_i – його верхня межа.

На відміну від звичайного двійково-десяткового подання код Грея гарантує, що дві сусідні вершини гіперкуба, на якому здійснюється пошук, котрі належать одному ребру, завжди декодуються в дві найближчі точки простору дійсних чисел, розміщених на відстані в одну величину точності їх дискретного подання.

Альтернативою двійковому кодуванню можна вважати метод *логарифмічного кодування*, який застововується для зменшення довжини хромосоми. Його використовують здебільшого в задачах багатовимірної оптимізації з великими просторами пошуку розв'язків.

Під час логарифмічного кодування перший біт (α) закодованої послідовності – це біт знака показникової функції, другий біт (β) – біт знака степеня цієї функції, а решта бітів (*bin*) – це значення самого степеня:

$$x = (-1)^{\text{б}} e^{(-1)^{\text{в}}[bin]_{10}},$$

де $[bin]_{10}$ – десяткове значення числа, закодованого у вигляді двійкової послідовності *bin*.

2.3. Генетичні оператори

У процесі функціонування ГА хромосоми зазнають трьох основних видів модифікації (операторів): *схрещування* (*cross-over*), *мутації* (*mutation*) та *інверсії* (*inversion*).

Схрещування. Під час схрещування в двох обраних хромосомах випадково обирається точка схрещування і потім відбувається обмін частин хромосом-батьків з утворенням двох нащадків (рис. 2.2).

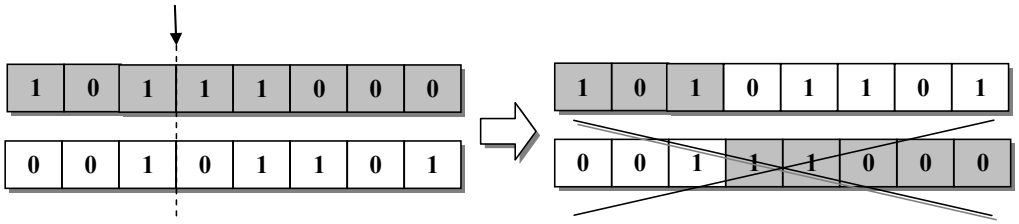


Рис. 2.2. Дія оператора схрещування

Далі з двох утворених нащадків випадково обирається тільки один. Усі генетичні оператори застосовуються із заздалегідь вибраною імовірністю (в діапазоні від 0 до 1). Типова ймовірність застосування оператора схрещування на якому здійснюється пошук – 0,9.

Мутація. У процесі мутації випадково обраний ген змінюється на протилежний (рис. 2.3). Типова ймовірність застосування мутації – 0,1.

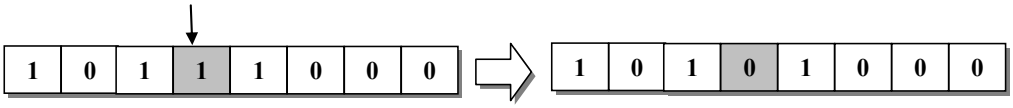


Рис. 2.3. Дія оператора мутації

Інверсія. Оператор інверсії змінює послідовність генів у хромосомі відносно випадково вибраної точки (рис. 2.4).

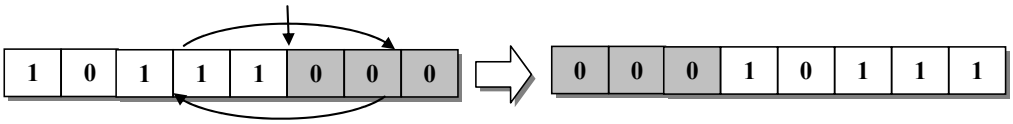


Рис. 2.4. Дія оператора інверсії

Зазвичай оператор інверсії застосовують з імовірністю 0,01.

2.4. Репродуктивний план Холланда

Основний (класичний) генетичний алгоритм, відомий також як репродуктивный план Холланда, складається з таких кроків:

Крок 1. **Ініціалізація** – формування початкової популяції (набору хромосом, множини потенційних розв'язків) випадковим чином.

Крок 2. **Оцінювання пристосованості** – розрахунок функцій пристосованості для кожної особини (хромосоми) в популяції.

Крок 3. **Перевірка умови зупинення алгоритму** – зупинення алгоритму після досягнення очікуваного оптимального значення, можливо із заданою точністю (залежно від його конкретного застосування).

Крок 4. **Селекція хромосом** – вибір за розрахованими значеннями функції пристосованості тих хромосом, які будуть брати участь у створенні нащадків для наступного покоління популяції.

Крок 5. **Застосування операторів** – застосування генетичних операторів схрещування, мутації та інверсі до вибраних на попередньому кроці батьківських хромосом.

Крок 6. **Створення нового покоління** – циклічне повторення кроків 4 і 5, доки популяція наступного покоління не заповниться вся.

Крок 7. **Вибір найкращої хромосоми** – обрання розв'язком задачі хромосоми з найкращим значенням функції пристосованості, коли умова зупинення еволюції виконана.

Блок-схему репродуктивного плану Холланда показано на рис. 2.5. Розглянемо тепер докладніше найбільш значущі елементи наведеного алгоритму.

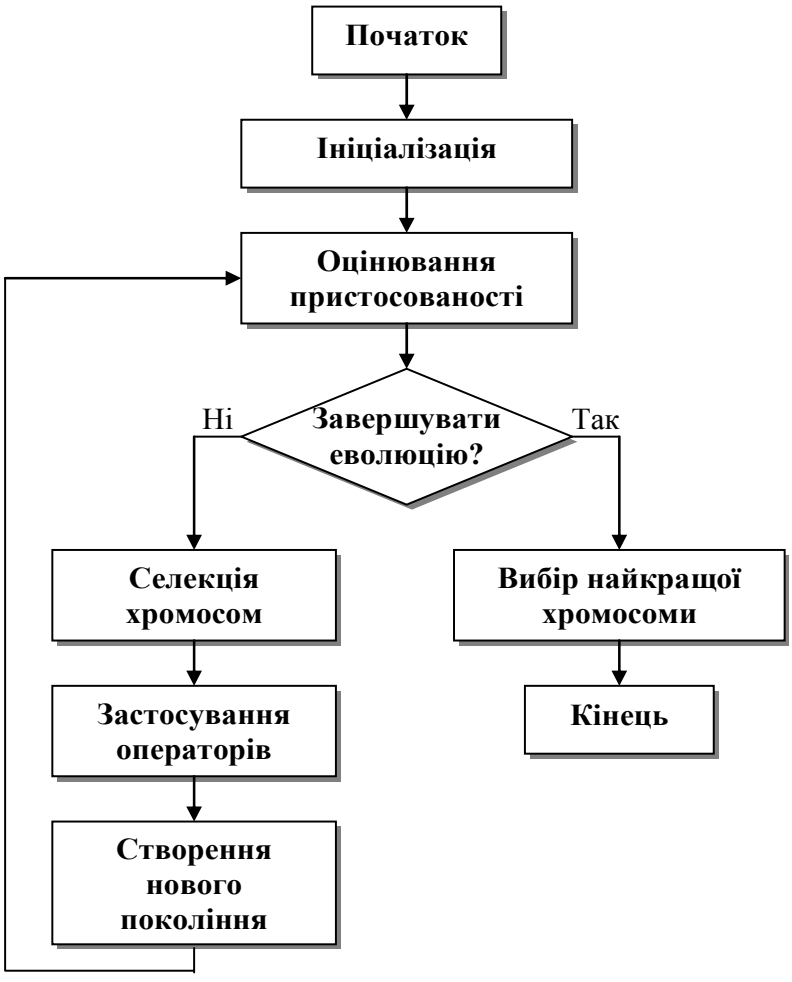


Рис. 2.5. Блок-схема генетичного алгоритму

2.5. Функція пристосованості

Оцінювання пристосованості хромосом у популяції полягає в розраховування *функції пристосованості* (*fitness function*). Чим більше значення цієї функції, тим вища «якість» хромосоми, а отже, і відповідного їй розв'язку задачі. Форма функції пристосованості залежить від характеру розв'язуваної задачі. Припустимо, що для розв'язання вихідної задачі треба максимізувати цю функцію. Якщо вихідна форма функції пристосованості не задовольняє цих умов, то виконують відповідне перетворення. Наприклад, задачу мінімізації функції можна легко звести до задачі максимізації оберненням цільової функції.

Аргументами для функції пристосованості стають значення параметрів оптимізації, отримані з певної хромосоми декодуванням.

2.6. Селекція батьківських хромосом

За розрахованими значеннями функції пристосованості з поточної популяції вибираються батьківські хромосоми для подальшого схрещування, продукування нащадків і формування нового покоління. Такий вибір роблять відповідно до принципу природного добору, за яким найбільші шанси на участь у створенні нових особин мають хромосоми з *найбільшими* значеннями функції пристосованості.

Є різні методи селекції. Найбільш популярним вважається *метод рулетки* (*roulette wheel selection*), який свою назву отримав за аналогією з відомою азартною грою. Кожній хромосомі відповідатиме сектор колеса рулетки, розмір якого задають пропорційним значенню функції пристосованості цієї хромосоми:

$$S_i = \frac{p_i}{\sum_{j=1}^{N} p_j},$$

де S_i – розмір «сектора»; N – кількість хромосом (особин) у популяції; p_i – нормоване за такою формулою значення функції пристосованості.

$$p_i = \frac{f_i - f_{\min}}{f_{\max} - f_{\min}}, \qquad (2.1)$$

де f_i – ненормоване значення, отримане в результаті обчислення функції пристосованості; $f_{\min}$ і $f_{\max}$ – відповідно мінімальне і максимальне значення функції пристосованості посеред її значень, обчислених для всіх хромосом у популяції.

Тож, чим більше значення функції пристосованості, тим більший сектор на колесі рулетки. Як видно з виразу (2.1) сума величин усіх секторів дорівнює одиниці (рис. 2.6).

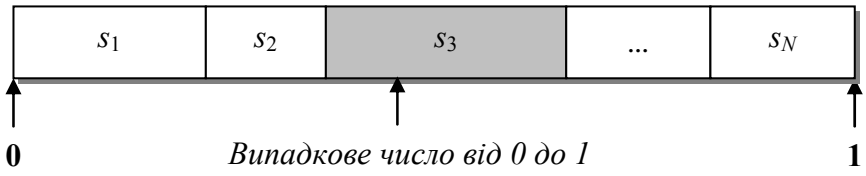


Рис. 2.6. Розподіл «секторів рулетки» в діапазоні від 0 до 1 ($... < s_2 < s_N < s_1 < s_3 < ...$)

Після цього генерується випадкове число в діапазоні від 0 до 1 і перевіряється, в який сектор воно потрапило. «Успішна» хромосома (на рис. 2.6 – це хромосома 3) і вибирається як батьківська для формування наступного покоління.

Слід зазначити, що у деяких модифікаціях ГА процес вибору батьківських хромосом не завжди випадковий. Зокрема, за підходу, котрий називають *елітизм* або *елітарна стратегія*, одна або декілька хромосом з найвищим значенням функції пристосованості одразу переносяться в наступне покоління без жодних модифікацій, тобто без застосування генетичних операторів.

2.7. Критерії зупинення еволюції

Визначення умови зупинення ГА залежить від його конкретного застосування. В оптимізаційних задачах, якщо відоме максимальне (або мінімальне) значення функції пристосованості, то зупинення

алгоритму може відбутися після досягнення очікуваного оптимального значення заданої точності. Алгоритм може також зупинятися у випадку, коли його виконання не сприяє поліпшенню вже досягнутого значення. Алгоритм може бути зупинено після закінчення визначеного часу виконання або після виконання заданої кількості операцій.

Окрім цього, критерієм зупинення може бути факт попадання певної кількості хромосом популяції у наперед заданий окіл їх усередненого значення. У процесі еволюції (ітерацій алгоритма) дедалі більше хромосом набуває ознак оптимального розв'язку задачі, що означає деяку їх «схожість». Це означає, що щільність хромосом в околі оптимуму буде збільшуватись (рис. 2.7).

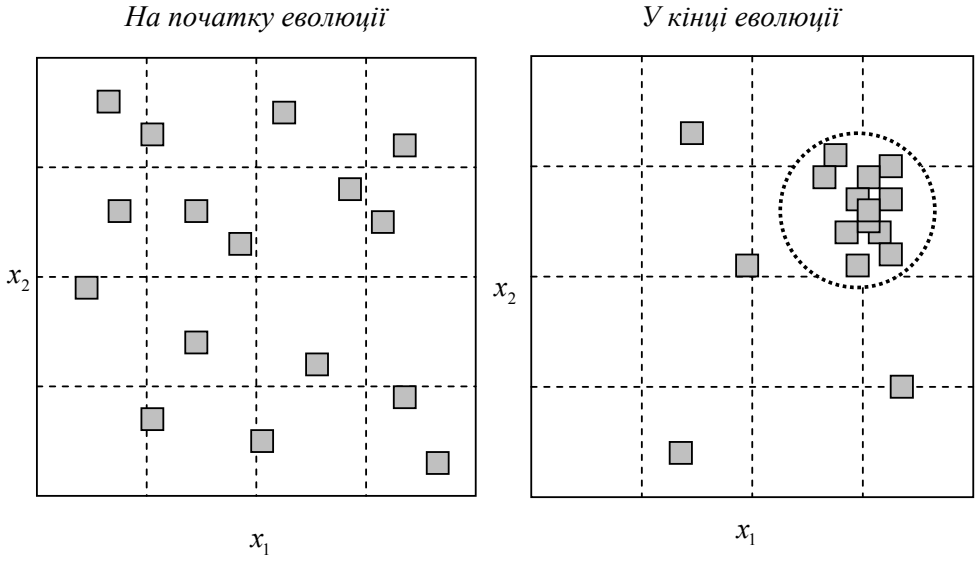


Рис. 2.7. Зміна стану популяції в процесі еволюції

Оцінюючи радіус розкиду параметрів від їх середнього значення, можна зупинити алгоритм, коли цей радіус стане достатньо малим. Середнє значення за кожним з параметрів оптимізації розраховують як

$$c_j = \frac{1}{N}\sum_{i=1}^{N} x_{ij},$$

де N – кількість хромосом у популяції; x_{ij} – значення (після декодування) j-го параметра в i-й хромосомі.

Тоді відстань від *i* хромосоми до центру буде

$$R_i = \sqrt{\sum_{j=1}^{n} (x_{ij} - c_j)^2} \le R_0. \qquad (2.2)$$

Коли відстань (2.2) стане меншою від прийнятого R_0 для наперед заданої кількості хромосом, алгоритм зупиняється.

2.8. Генетичні алгоритми для багатокритерійної оптимізації

Більшість задач, які розв'язують за допомогою генетичних алгоритмів, мають один критерій оптимізації, який використовують як функцію пристосованості. У свою чергу, багатокритерійна оптимізація ґрунтується на відшукуванні розв'язку, що одночасно оптимізує більше, ніж одну функцію. В цьому випадку шукають певний компроміс, яким і виступає розв'язок.

Є декілька класичних методів, що стосуються багатокритерійної оптимізації. Один з них – це метод *зваженої функції* (*method of objective weighting*). Відповідно до цього методу функції f_i, що оптимізуються, взяті із вагами w_i, утворюють єдину функцію:

$$F(\overline{x}) = \sum_{i=1}^{N_f} w_i f_i(\overline{x}),$$

де N_f – кількість функцій, що оптимізуються, $w_i \in [0,1]$ і $\sum_{i=1}^{N_f} w_i = 1$.

Ще один підхід до багатокритерійної оптимізації пов'язаний із розділенням популяції на підгрупи однакового розміру (*sub-populations*), кожна з яких «відповідає» за одну функцію, яку оптимізують. Селекцію виконують автономно для кожної функції, однак операцію схрещування виконують без урахування меж підгруп.

3. БІОЛОГІЧНІ ТА ШТУЧНІ НЕЙРОННІ МЕРЕЖІ

3.1. Біологічний нейрон

Попри розбіжності у будові, всі нейрони проводять інформацію однаково. Нейрони отримують сигнали сильно розгалуженими відростками (*дендритами*), а посилають сигнали по нерозгалуженим відросткам (*аксонам*). Інформація передається по аксонах (рис. 3.1) у вигляді коротких електричних імпульсів, так званих потенціалів дії, амплітуда яких становить приблизно 100 мВ, а тривалість – 1 мс.

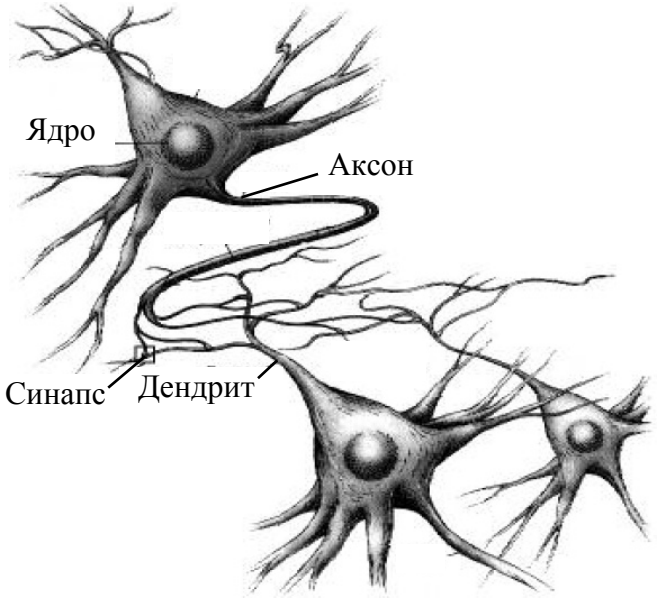


Рис. 3.1. Нервові клітини головного мозку людини

Але не будь-який імпульс, що надійшов у нейрон по дендритах, передається на аксон. Лише за досягнення певного критичного (порогового) значення потенціалу, клітина генерує сигнал, що проходить по аксону до інших нейронів. Швидкість поширення нервового імпульсу по аксону досягає 100 м/с.

3.2. Історія штучних нейронних мереж

Здатність нейроподібних структур до навчання вперше дослідили Дж. Маккалок та У. Пітт. У 1943 році вийшла їх робота «Логічне чис-

лення ідей, що стосуються нервової діяльності», в якій було описано модель нейрона і сформульовано принципи побудови штучних нейронних мереж (ШНМ).

Великий поштовх розвиткові нейрокібернетики дав американський нейрофізіолог Ф. Розенблат, який запропонував 1962 року свою модель нейронної мережі – персептрон. Сприйнятий спочатку з великим ентузіазмом, він невздовзі зазнав нищівної критики з боку великих наукових авторитетів. І хоча детальний аналіз їх аргументів показує, що вони заперечували не зовсім той персептрон, який пропонував Ф. Розенблат, дослідження ШНМ було згорнуто майже на 10 років. Попри це в 70-ті роки було запропоновано багато цікавих розробок, таких, наприклад, як *когнітрон*, здатний добре розпізнавати досить складні образи незалежно від повороту і зміни масштабу зображення.

У 1982 році американський біофізик Дж. Хопфілд запропонував оригінальну модель ШНМ, названу його іменем. За наступні декілька років було винайдено мережу зустрічного потоку, двонапрямлену асоціативну пам’ять та багато інших алгоритмів.

В інституті кібернетики ім. В. М. Глушкова НАН України з 70-х років ведуть роботу над стохастичними нейронними мережами.

3.3. Штучна нейронна мережа

Найбільш містке визначення ШНМ таке:

Штучна нейронна мережа – це суттєво паралельно розподілений процесор, здатний зберігати і відтворювати дослідне знання.

Вона схожа з мозком людини у двох аспектах:

– мережа набуває знань у процесі навчання;

– для збереження знань мережа використовує сили міжнейронних з’єднань (синаптичних ваг).

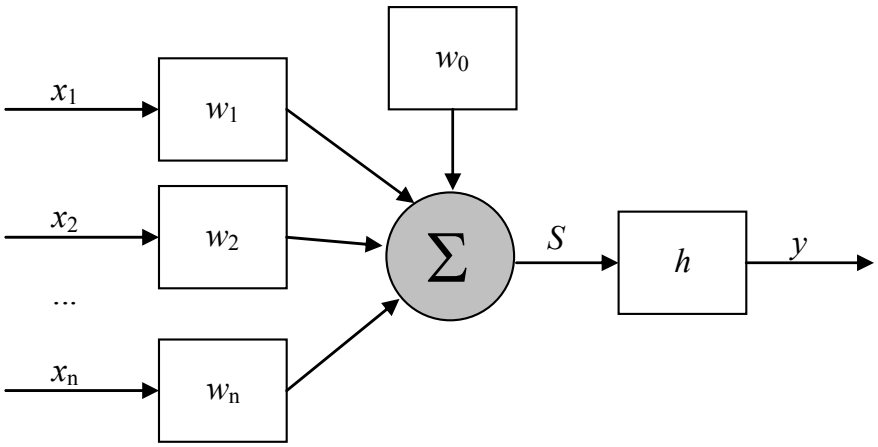


Рис. 3.2. Схема штучного нейрона

Штучний нейрон, схему якого показано на рис. 3.2, працює так. Сигнали x_i $(i = 1,...,n)$, що надходять на вхід нейрона, множаться на вагові коефіцієнти w_i (синаптичні ваги). Далі вони додаються і вислідний сигнал зсувається на величину w_0 порогового зсуву:

$$S = \sum_{i=1}^{n} w_i x_i + w_0 \,. \tag{3.1}$$

Потім вихідний сигнал S подається на вхід блоку, що реалізує нелінійну активаційну функцію нейрона $h(S)$.

3.4. Типи активаційних функцій

Традиційно активаційні функції східчасті, тобто сигнал на виході нейрона з'являється тільки тоді, коли сумарний вхідний вплив перевищує певне порогове значення.

Одна з умов реалізації якогось перетворення за допомогою ШНМ – це *нелінійність* її активаційної функції. Для перетворень найчастіше використовують такі функції: лінійні, лінійні з насиченням (рис. 3.3, *а, б*); порогові функції (рис. 3.3, *в, г*); сигмоподібні тангенціальні функції (рис. 3.3, *д*); сигмоподібні логарифмічні (рис. 3.3, *е*); гармонічні функції та радіально-базисні.

Хоча лінійна функція у чистому вигляді й не дозволяє реалізувати будь-якого перетворення, проте вона часто може виявитись корисною для простого масштабування вихідного сигналу нейрона.

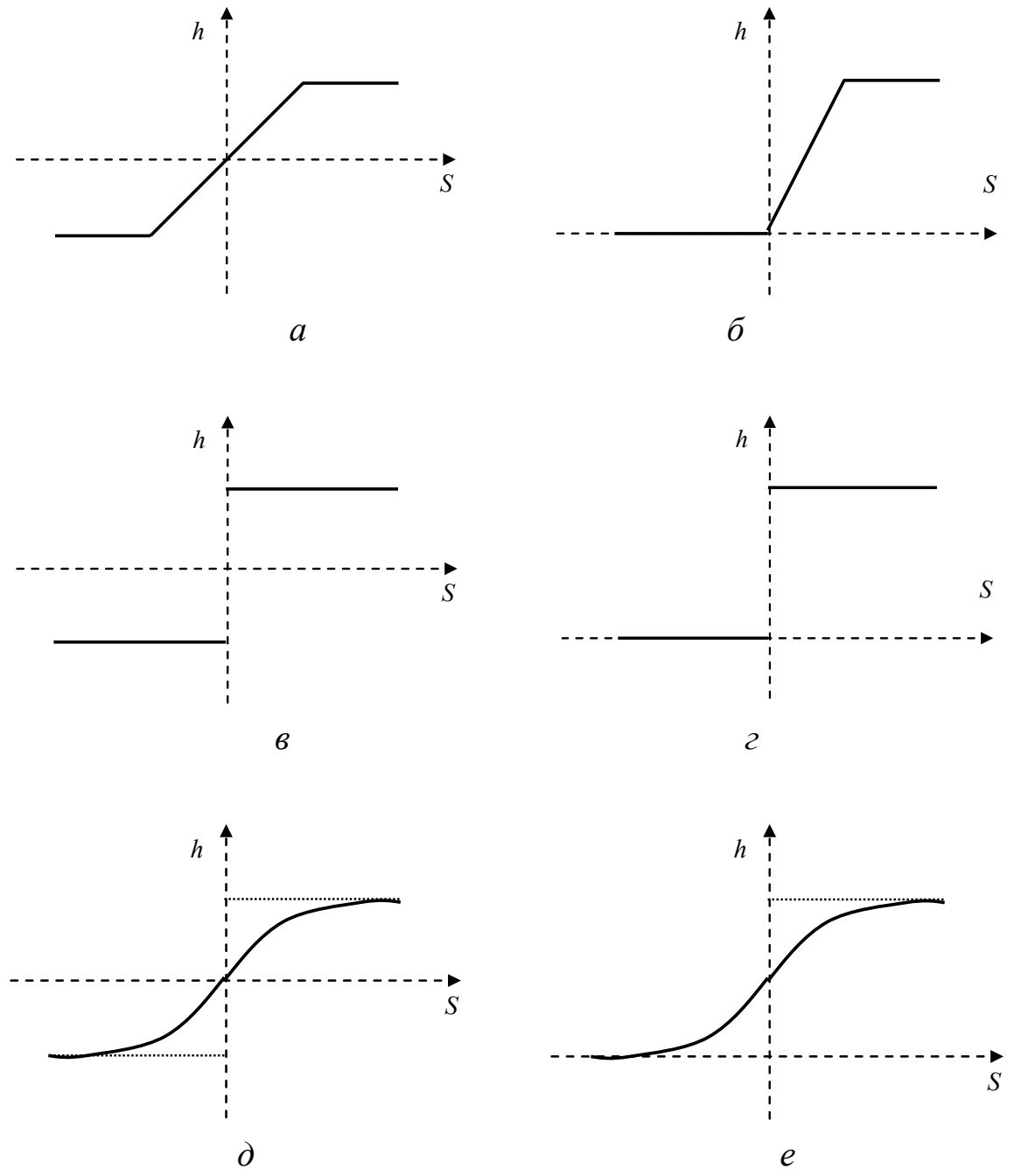


Рис. 3.3. Графіки активаційних функцій: *а* – лінійної з насиченням; *б* – лінійної з насиченням (зміщеної); *в* – порогової; *г* – порогової (зміщеної); *д* – сигмоподібної тангенціальної; *е* – сигмоподібної логарифмічної

Сигмоподібні функції найчастіше застосовують у нейронних мережах. Їх описують такими функціональними залежностями:

$$h(S) = \frac{2}{1+e^{-x}} - 1$$ – тангенціальною;

$$h(S) = \frac{1}{1+e^{-x}}$$ – логарифмічною.

Функції синуса або косинуса (гармонічні функції) досить часто використовують у нейронних мережах для подання періодичних перетворень.

Радіально-базисна функція має такий вигляд: $h(S) = e^{-x^2}$.

Наведені активаційні функції найбільш вживані, але в жодному разі не перекривають усіх можливих.

3.5. Багатошаровий персептрон

У своїй найпростішій версії багатошаровий персептрон являє собою мережу нейронів з одним вхідним, одним вихідним і одним прихованим шарами (див. рис. 3.4).

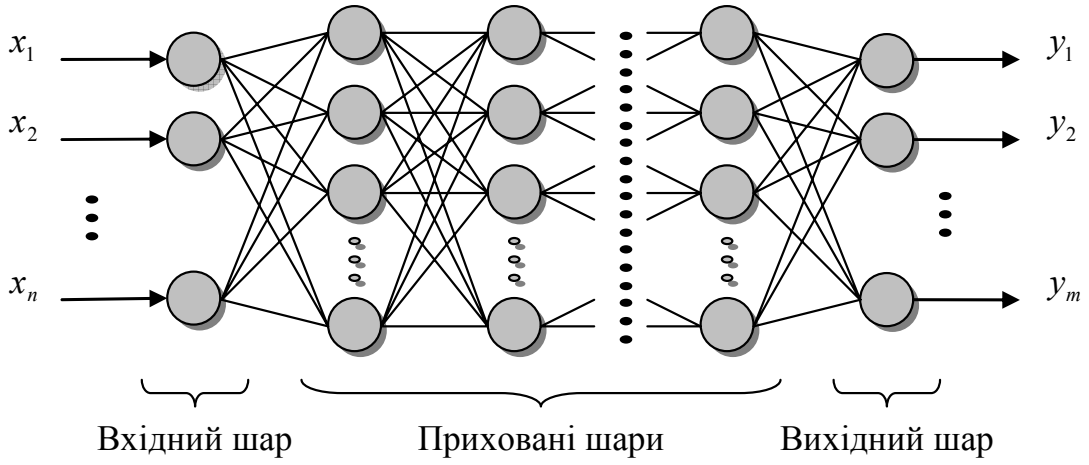


Рис. 3.4. Багатошаровий персептрон

На рисунку кружками позначено нейрони, які мають входи і виходи. В загальному випадку вихід кожного нейрона шару (i–1) надходить на вхід кожного нейрона шару (i). Спільною ознакою для всіх багатошарових персептронів є пряма напрямленість передавання інформації від вхідного шару до вихідного через певну кількість прихованих шарів.

Модифіковані версії можуть мати прямі зв’язки між несуміжними шарами, зв’язки в межах одного шару, хаотичні зв’язки між шарами замість регулярних. Слід зазначити, що в інтелектуальних системах керування найбільшого поширення набули традиційні багатошарові персептрони.

4. МЕТОДИ НАВЧАННЯ НЕЙРОННИХ МЕРЕЖ

4.1. Навчання нейронних мереж і еталонні дані

Для того, щоб ШНМ могла успішно представляти дані або функціональну залежність, її потрібно спершу навчити.

Навчання – це процес адаптації мережі до пред'явлених еталонних зразків модифікацією (відповідно до того чи іншого алгоритму) вагових коефіцієнтів зв'язків між нейронами.

Зауважимо, що цей процесс – результат алгоритму функціонування мережі, а не попередньо закладених у неї знань людини, як це часто буває в системах штучного інтелекту. Процес навчання повторюється ітеративно доти, доки мережа не набуде потрібних властивостей.

Якщо навчання ШНМ відбувається із використанням еталонних даних, то такий підхід називають «навчанням з учителем» (*supervised training*). Еталонні дані складаються з шаблонів. Кожен шаблон, в свою чергу, складається із вектора відомих входів мережі $\bar{X}^* = \{x_1^*, x_2^*, \ldots, x_n^*\}$ і вектора відповідних їм бажаних виходів $\bar{D} = \{d_1, d_2, \ldots, d_m\}$. Коли на вхід ненавченої мережі подається еталонний вектор $\bar{X}^*$, вихідний вектор $\bar{Y} = \{y_1, y_2, \ldots, y_m\}$ буде відрізнятися від вектора бажаних вихідних значень $\bar{D}$. У цьому разі функцію похибки роботи ШНМ можна задати у вигляді, що відповідає методу найменших квадратів:

$$E = \frac{1}{2}\sum_p (\bar{Y}_p - \bar{D}_p)^2 , \qquad (4.1)$$

де індекс *p* означає номер шаблона із еталонних даних. Таким чином похибка обчислюється як сума похибок за всіма шаблонами еталонних даних.

Слід зазначити, що розроблення адекватних еталонних даних для навчання – завдання трудомістке і не завжди здійсненне.

4.2. Метод випадкового пошуку (*random search*)

Під час навчання ШНМ методом випадкового пошуку відбувається динамічне підстроювання вагових коефіцієнтів синапсів, під час якого вибираються здебільшого найслабкіші зв'язки і змінюються на малу величину в той чи інший бік. Після цього обчислюють функцію похибки (4.1) і зберігають тільки ті зміни, які зумовили зменшення похибки на виході мережі порівняно з попереднім станом.

Очевидно, що цей підхід, попри свою уявну простоту, потребує значної кількості ітерацій доки буде досягнуто прийнятних похибок на виході мережі.

4.3. Метод зворотного поширення похибки (*Error Back Propagation*)

Метод зворотного поширення похибки вперше був сформульований Вербосом 1974 року. Але визнаний він був значно пізніше, 1986 року, коли його «перевідкрив» Румельхарт, Хінтон і Вільямс. За своєю суттю – це метод градієнтного спуску, котрий застосовують для настроювання вагових коефіцієнтів так, щоб мінімізувати функцію похибки (4.1).

Мінімізація за методом градієнтного спуску означає підстроювання вагових коефіцієнтів

$$\Delta w_{ij}^{(n)} = -\text{з}\,\frac{\partial E}{\partial w_{ij}}, \qquad (4.2)$$

де w_{ij} – ваговий коефіцієнт синаптичного зв'язку, що з'єднує i-й нейрон шару n–1 з j-м нейроном шару n, η – коефіцієнт швидкості навчання, $0<\eta<1$. Користуючись правилами диференціального числення, можна записати:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial y_j}\cdot\frac{dy_j}{ds_j}\cdot\frac{\partial s_j}{\partial w_{ij}}, \qquad (4.3)$$

де y_j – вихід j-го нейрона, а s_j – зважена сума його вхідних сигналів, тобто аргумент його нелінійної активаційної функції:

$$s_j^{(n)} = \sum_{i=0}^{M} y_i^{(n-1)}\, w_{ij}^{(n)} + b_j^{(n)} .$$

Оскільки множник dy_j / ds_j є похідною цієї функції за її аргументом, то з цього випливає, що похідна активаційної функції повинна бути визначеною на всій осі абсцис. У зв'язку з цим активаційні функції з розривами першого і другого родів не підходять для нейронних мереж, що навчаються методом зворотного поширення похибки.

Наприклад, у випадку гіперболічного тангенса маємо:

$$\frac{dy}{ds} = 1 - s^2 .$$

Третій множник $\partial s_j / \partial w_{ij}$ дорівнює виходу i-го нейрона з попереднього шару, $y_i^{(n-1)}$. Щодо першого множника в (4.3), то його легко розкласти так:

$$\frac{\partial E}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} \cdot \frac{\partial s_k}{\partial y_j} = \sum_k \frac{\partial E}{\partial y_k} \cdot \frac{dy_k}{ds_k} w_{jk}^{(n+1)} .$$

Тут додавання за k виконують серед нейронів шару n+1. Вводячи нову змінну

$$\delta_j^{(n)} = \frac{\partial E}{\partial y_j} \cdot \frac{dy_j}{ds_j},$$

отримуємо рекурсивну формулу для розрахунків величин $\delta_j^{(n)}$ шару n за величинами $\delta_k^{(n+1)}$ наступного шару n+1:

$$\delta_j^{(n)} = \left[\sum_k \delta_k^{(n+1)}\, w_{jk}^{(n+1)} \right] \frac{dy_j}{ds_j} . \qquad (4.4)$$

Для останнього ж, вихідного шару маємо:

$$\delta_l^{(N)} = (y_l^{(N)} - d_l)\frac{dy_l}{ds_l}. \qquad (4.5)$$

Тепер ми можемо записати (4.2) в розгорнутому вигляді:

$$\Delta w_{ij}^{(n)} = -\eta\delta_j^{(n)} y_i^{(n-1)}. \qquad (4.6)$$

Інколи для надання процесу корекції ваг деякої інерційності, що згладжує різкі стрибки під час переміщення поверхнею цільової функції, вираз (4.6) доповнюється значенням зміни ваги на попередній ітерації:

$$\Delta w_{ij}^{(n)}(t) = -\eta(\mu\Delta w_{ij}^{(n)}(t-1) + (1-\mu)\delta_j^{(n)} y_i^{(n-1)}), \qquad (4.7)$$

де μ – коефіцієнт інерційності, t – номер поточної ітерації.

Отже, повний алгоритм навчання ШНМ за допомогою процедури зворотного поширення похибки має такий вигляд:

Крок 1. Ініціалізація ваги і зміщення, які задають випадково. Наприклад, в діапазоні від –1 до 1.

Крок 2. Розрахування виходу мережі і похибки за вектором входу і відповідним йому вектором бажаного виходу із еталонних даних.

Крок 3. Розрахування $\delta^{(N)}$ для вихідного шару за формулою (4.5).

Крок 4. Розрахування зміни ваг $\Delta w^{(N)}$ для вихідного шару за формулою (4.6) або (4.7).

Крок 5. Розрахування за формулами (4.4) і (4.6), або (4.4) і (4.7) відповідно $\delta^{(n)}$ і $\Delta w^{(n)}$ для всіх інших шарів.

Крок 6. Коригування всіх ваг у ШНМ:

$$w_{ij}^{(n)}(t) = w_{ij}^{(n)}(t-1) + \Delta w_{ij}^{(n)}(t). \qquad (4.8)$$

Крок 7. Перехід на крок 2, якщо похибка мережі суттєва, а якщо ні – то закінчення навчання.

4.4. Навчання нейронних мереж за допомогою генетичних алгоритмів

Оскільки навчання ШНМ – це пошук такого стану параметрів мережі (синаптичних ваг і порогових зміщень), який мінімізує функцію похибки, котру можна задати у вигляді (4.1). Одним із найбільш ефективних методів глобальної оптимізації є генетичні алгоритми (ГА). Під час використання ГА для навчання ШНМ параметрами оптимізації є синаптичні ваги і порогові зміщення. Функція пристосованості буде обернено пропорційною до функції похибки (4.1).

4.5. Навчання без вчителя (*unsupervised training*)

Розглянуті вище алгоритми навчання ШНМ за допомогою процедур зворотного поширення похибки, випадкового пошуку або генетичних алгоритмів припускають наявність деякої зовнішньої ланки, що надає мережі крім вхідних образів також і цільові вихідні образи. Алгоритми, що використовують подібну концепцію, називають алгоритмами навчання з вчителем. Для їх успішного функціонування потрібні експерти, які створюють на попередньому етапі для кожного вхідного образу еталонний вихідний. Оскільки створення штучного інтелекту розвивається шляхом копіювання природних прообразів, вчені не припиняють сперечатись щодо того, чи можна вважати алгоритми навчання з вчителем натуральними, чи вони повністю штучні. Наприклад, навчання людського мозку, на перший погляд, відбувається без вчителя: на зорові, слухові, тактильні та інші рецептори надходить інформація ззовні, і всередині нервової системи відбувається певна самоорганізація. Однак не можна заперечувати й того, що в житті людини немало вчителів – і в буквальному, і в переносному сенсі, – тих, що координують зовнішні впливи. Проте, чим би не закінчилася су-

перечка прихильників цих двох концепцій навчання, вони обидві мають право на існування.

Головна перевага навчання без вчителя – це його «самостійність». Процес навчання без вчителя, як і навчання з вчителем, полягає у підстроюванні ваг синапсів. Деякі алгоритми, однак, змінюють і структуру мережі, тобто кількість нейронів і їх взаємозв'язки, але такі перетворення правильніше назвати більш широким терміном «самоорганізація», та в межах цього матеріалу ми їх не розглядатимемо. Очевидно, що підстроювати синапси можна тільки на основі інформації, доступної в нейроні, тобто його стану і вже наявних вагових коефіцієнтів. Виходячи із цієї думки і, що більш важливо, за аналогією з відомими принципами самоорганізації нервових клітин, побудовані алгоритми навчання Хебба.

Сигнальний метод навчання Хебба полягає в зміні ваг за таким правилом обчислень:

$$w_{ij}(t) = w_{ij}(t-1) + \alpha y_i^{(n-1)} y_j^{(n)}, \qquad (4.9)$$

де $y_i^{(n-1)}$ – вихідне значення i-го нейрона (n–1) шару; $y_j^{(n)}$ – вихідне значення j-го нейрона шару n; $w_{ij}(t)$ і $w_{ij}(t-1)$ – ваговий коефіцієнт синапса, що з'єднує ці нейрони на ітераціях t і $t-1$ відповідно; α – коефіцієнт швидкості навчання. Тут і надалі, для загальності, літерою n позначають довільний шар мережі. Під час навчання за цим методом підсилюються зв'язки між збудженими нейронами.

Є також і диференціальний метод навчання Хебба:

$$w_{ij}(t) = w_{ij}(t-1) + \text{б}\left[y_i^{(n-1)}(t) - y_i^{(n-1)}(t-1) \right]\left[y_j^{(n)}(t) - y_j^{(n)}(t-1) \right], \qquad (4.10)$$

де $y_i^{(n-1)}(t)$ і $y_i^{(n-1)}(t-1)$ – вихідне значення нейрона і шару $n-1$ відповідно на ітераціях t і $t-1$; $y_j^{(n)}(t)$ і $y_j^{(n)}(t-1)$ – те ж саме для ней-

рона *j* шару *n*. Як видно з формули (4.10), найбільше навчаються синапси, що з'єднують ті нейрони, виходи яких динамічно збільшаться.

Повний алгоритм навчання у цьому випадку буде виглядати так:

Крок 1. Привласнення малих випадкових значень на стадії ініціалізації всім ваговим коефіцієнтам.

Крок 2. Подача вхідних образів на входи мережі і поширення сигналів збудження по всіх шарах відповідно до принципів класичних прямопоточних (*feedforward*) мереж, тобто для кожного нейрона розраховується зважена сума його входів, до якої потім застосовується активаційна (передатна) функція нейрона, в результаті чого отримується його вихідне значення.

Крок 3. Зміна вагових коефіцієнтів на основі отриманих вихідних значень нейронів за формулою (4.9) або (4.10).

Крок 4. Повторення, починаючи з кроку 2, доки вихідні значення мережі не установляться із заданою точністю.

Завершення навчання за новим способом, відмінним від використовуваного для мережі зворотного поширення, зумовлене тим, що підстроювані значення синапсів фактично не обмежені. На другому кроці циклу поперемінно подаються всі образи із вхідного набору.

Слід зазначити, що відгук на кожен з класів вхідних образів не відомий заздалегідь і являтиме собою довільне сполучення станів нейронів вихідного шару, зумовлене випадковим розподілом ваг на стадії ініціалізації. Разом з тим, мережа здатна узагальнювати схожі образи, визначаючи їх приналежність до одного класу.

5. НЕЙРОННІ МЕРЕЖІ В СИСТЕМАХ КЕРУВАННЯ

5.1. Основи нейрокерування

Ідеї застосування ШНМ в системах керування вперше з'явилися в середині 70-х років XX століття. Зокрема, сам термін «*нейрокерування*» вперше з'явився в роботах Вербоса вже 1976 року. Однак вирішальну роль у втіленні ШНМ у сферу розв'язання задач керування зіграли роботи Нарендри та його співавторів 1989 року.

Так ШНМ в системах керування застосовують у вигляді нейроконтролерів (нейрорегуляторів) та нейроемуляторів, що імітують динамічну поведінку ОК в цілому, або описуючих його окремі характеристики, які не піддаються математичному моделюванню.

Нейронні мережі почали застосовуватись у СК завдяки таким властивостям:

– нейронні мережі можуть реалізовувати довільні гладкі функції будь-якої складності;

– для реалізації нейромережних СК потрібна мінімальна інформація про ОК;

– під час реалізації ШНМ у вигляді спеціалізованих інтегральних мікросхем можливе паралельне оброблення інформації, що значно збільшує швидкість роботи і підвищує надійність системи.

5.2. Послідовна схема нейромережевого керування

Найпростіша схема послідовного керування за допомогою ШНМ зображена на рис. 5.1.

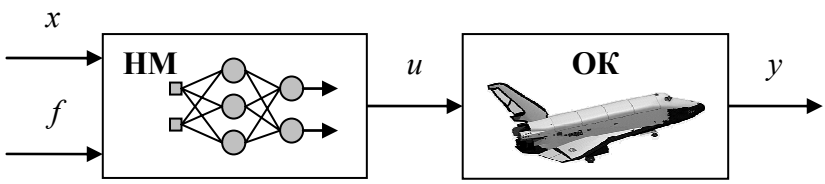


Рис. 5.1. Послідовна схема нейромережевого керування: *x* – вхідний задавальний сигнал системи; *f* – сигнали, що несуть інформацію про контрольовані збурення; *u* – керування, що діє на ОК; *y* – вихідний сигнал системи

Розрізняють два варіанти навчання ШНМ для схеми на рис. 5.1. А саме, *інверсне* і *предикатне*.

Інверсне керування

Очевидно, що в ідеальному випадку контролер у системі реалізує *зворотну* (інверсну) динаміку ОК. На рис. 5.2 зображено схему попереднього узагальненого інверсного навчання ШНМ. За такого підходу на вхід ШНМ подають реальний вихід системи, а мережа навчається за похибкою розузгодження її виходу з реальним входом. Після завершення процесу навчання ШНМ реалізує зворотну динаміку ОК. Потім вона використовується у схемі, показаній на рис. 5.1.

Однак, за нестаціонарності ОК застосування попереднього навчання ШНМ не дозволяє отримати хороших показників керування. Тому розробили схеми, які уможливлюють оперативне (в реальному часі) навчання.

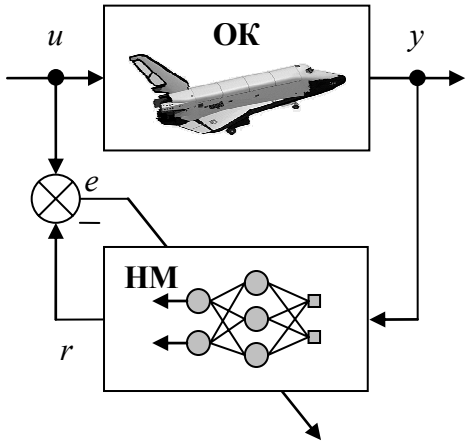


Рис. 5.2. Схема попереднього інверсного навчання

Розглянемо схему на рис. 5.3. Нейроемулятор НМ2 навчається зворотній динаміці ОК, а нейроконтролер НМ1 просто копіює свої параметри з НМ2.

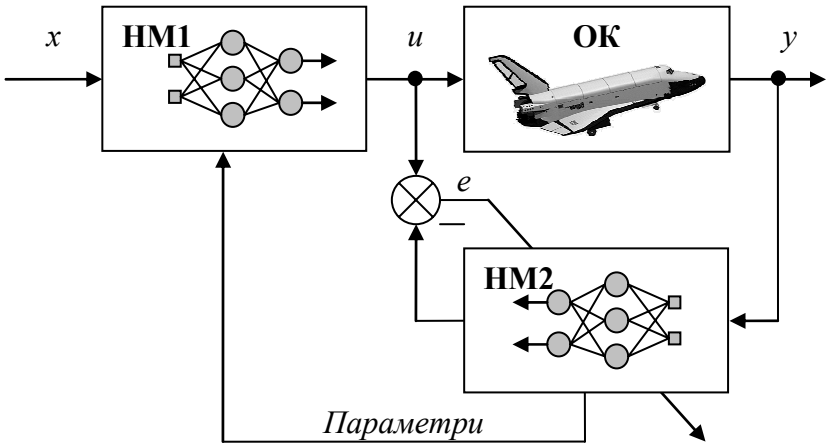


Рис. 5.3. Схема оперативного навчання ШНМ зворотній динаміці

Тут використовується розімкнена схема керування без від'ємного зворотного зв'язку. Перевагами такої схеми є, зазвичай, простота і стійкість. До недоліків можна віднести таке:

– за виконання умови квазістаціонарності ОК дана схема не гарантує, що вихідний сигнал ОК буде відповідати опорному сигналові;

– схема не здатна керувати нестійким об'єктом;

– труднощі виникають, якщо динаміка ОК не має зворотної.

Предикатне керування

Концепція предикатного керування побудована на керуванні з передбаченням. Рис. 5.4 ілюструє процедуру спеціалізованого навчання ШНМ для послідовної схеми керування.

У цьому разі ШНМ навчається так, щоб отримати найкраще виконання рівності $y = x$.

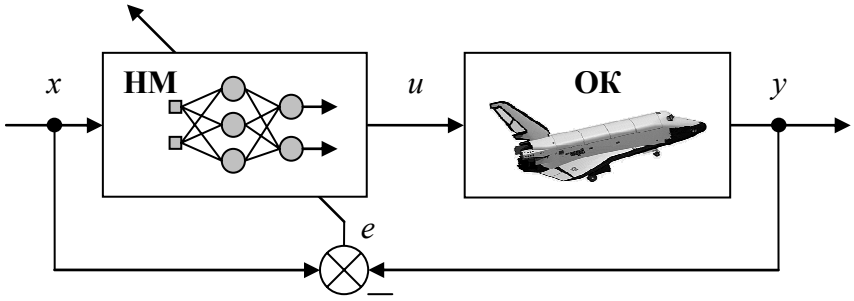


Рис. 5.4. Схема спеціалізованого навчання

Слід зазначити, що у цьому разі для навчання ШНМ не можна застосовувати класичного методу зворотного поширення похибки, оскільки між виходом системи і ШНМ стоїть ОК, якобіан якого в загальному випадку невідомий, і доводиться застосовувати або числову апроксимацію якобіана системи, або інші модернізації методу зворотного поширення, що не потребують інформації про якобіан системи.

Також для цього випадку може застосовуватися навчання ШНМ за допомогою генетичних алгоритмів. Крім цього, система, зображена на рис. 5.4, не може навчатися в оперативному режимі.

Інший підхід полягає у тому, щоб у систему (рис. 5.4) додати ще одну ШНМ (емулятор), який виконує імітаційне моделювання ОК (рис. 5.5).

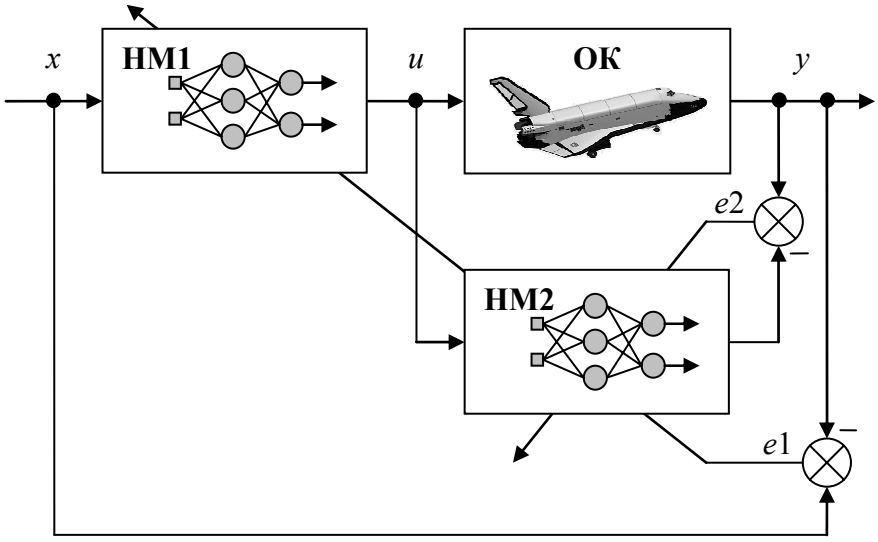


Рис. 5.5. Схема з нейроемулятором і нейроконтролером

На рис. 5.5 НМ1 виконує функції контролера, а НМ2 – емулятора. При цьому нейроемулятор НМ2 може використовуватись як для визначення якобіана ОК, так і для навчання нейроконтролера НМ1. У цьому випадку ідентифікатор НМ2 настроюється на пряму динаміку об’єкта, а НМ1 настроюється через ідентифікатор НМ2 так, щоб оптимізувати критерій якості керування на визначеному інтервалі часу. Після реалізації керування на даному інтервалі часу процес повторюється. Цей метод також називають «*зворотне поширення у часі*» або «*принцип горизонту, що віддаляється*».

Керування з передбаченням, порівняно з інверсійним керуванням, дає кращі результати. Особливо це проявляється у випадку нереалізовності точної зворотної динаміки об’єкта. Разом з тим і обчислювальні витрати для цього методу значно вищі. Розглядувана схема керування, як і попередня, належить до розімкнутих, і за невиконання умови квазістаціонарності об’єкта вона не гарантує, що вихідний сигнал ОК буде відповідати опорному сигналові.

5.3. Паралельна схема контролера нейромережевого керування

На рис. 5.6 наведена схема паралельного вмикання ШНМ як контролера для послідовної схеми керування.

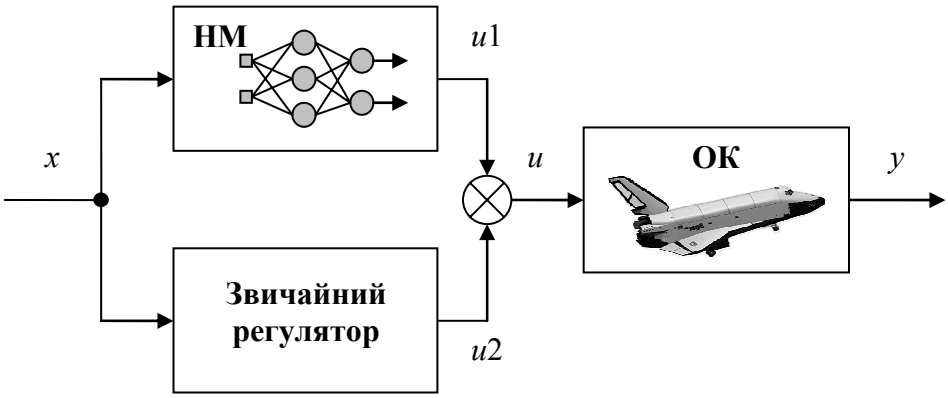


Рис. 5.6. Паралельна схема вмикання нейроконтролера

Навчання ШНМ полягає в тому, щоб скоригувати сигнал звичайного контролера *и*2, якщо він не забезпечує належної якості керування. Один із способів реалізації такого підходу зображено на рис. 5.7. Такий підхід називають «*навчанням із похибкою зворотного зв'язку*».

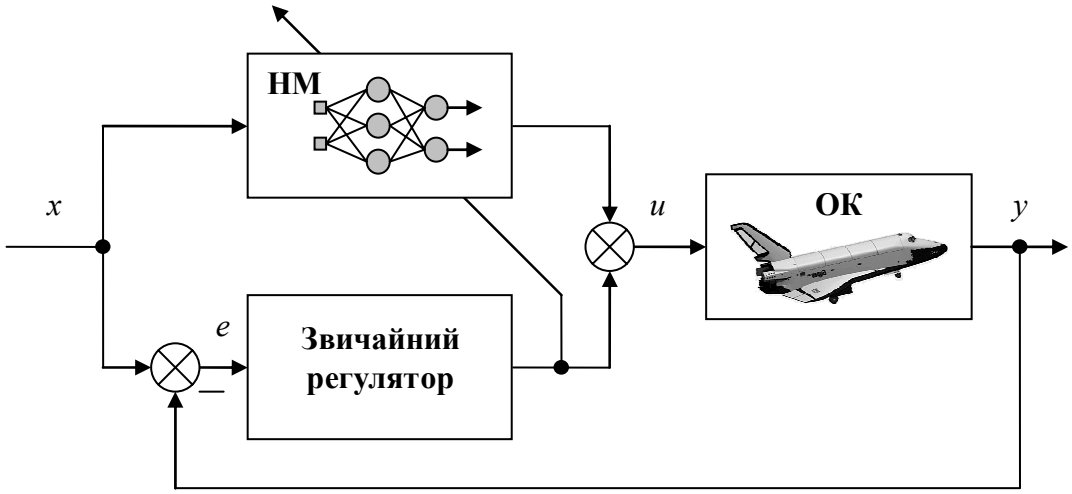


Рис. 5.7. Схема навчання із похибкою зворотного зв'язку

У цій схемі ШНМ після завершення навчання приймає на себе керування об'єктом, усуваючи дію контролера зворотного зв'язку. Зазначимо, що зразковим регулятором у цьому випадку може виступати і людина-оператор.

5.4. Нейромережеве керування із зворотним зв'язком

Найпростіша схема нейромережевого керування із зворотним зв'язком подана на рис. 5.8.

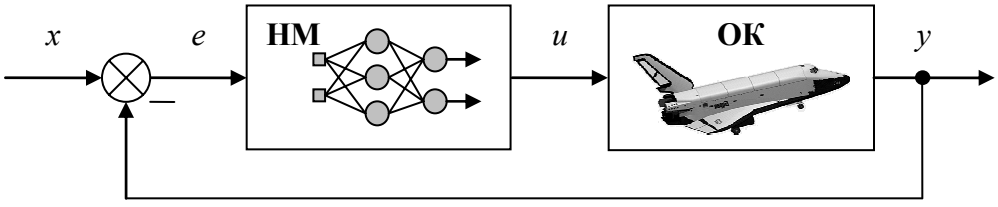


Рис. 5.8. Нейромережеве керування із зворотним зв'язком

Нейронна мережа виконує функції регулятора замкненої системи. Перевагою такої схеми є здатність забезпечувати високу якість керу-

вання за наявності неконтрольованих збурень, а також нестаціонарності і нестійкості ОК.

За такого підходу задача навчання ШНМ стає складнішою. Якщо для послідовної схеми керування відомо, що ШНМ має реалізовувати зворотну динаміку об'єкта, то в схемі зі зворотним зв'язком перетворення, яке має реалізовувати ШНМ, невідоме. Крім цього, ні зв'язок, ні якобіан зв'язку між показником якості регулювання і параметрами ШНМ невідомі. Система на рис. 5.8 не може навчатися в оперативному режимі.

Розглянемо більш складну схему керування. У схемі на рис. 5.9 використано контролер зворотного зв'язку, побудований з використанням мережі НМ1, що навчається через ідентифікатор НМ2.

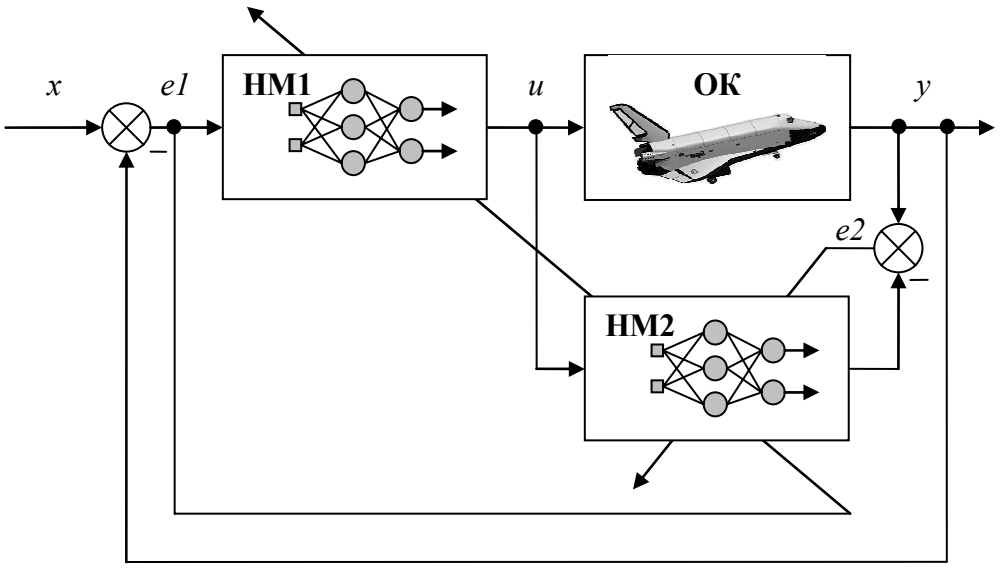


Рис. 5.9. Керування із зворотним зв'язком

Навчання через ідентифікатор, а не безпосередньо на об'єкті, потрібне для того, щоб не заважати нормальному функціонуванню об'єкта пробними впливами, які використовують у процесі навчання.

До недоліків схеми можна віднести високі вимоги до обчислювальних ресурсів і можливі проблеми із стійкістю системи.

5.5. Схема із звичайним контролером, що керується нейронною мережею

У деяких системах використовують ідею настроювання параметрів звичайних регуляторів (наприклад, широковідомого ПІД-регулятора) для підвищення якості керування.

У системі, наведеній на рис. 5.10, ШНМ застосовують для настроювання параметрів ПІД-регулятора.

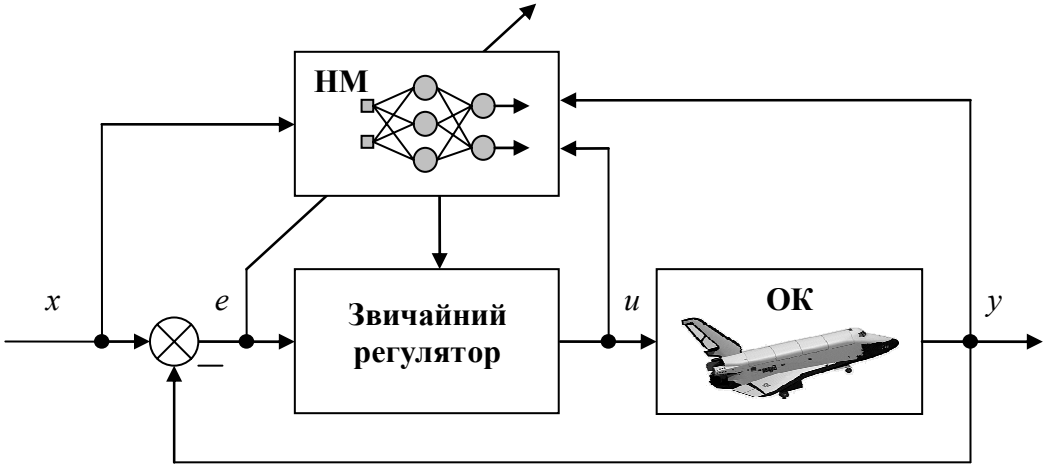


Рис. 5.10. Схема нейромережевого керування із самонастроюванням

5.6. Недоліки систем керування з нейромережами

Попри численні достоїнства, СК на основі ШНМ мають багато недоліків:

– під час оптимізації ваг ШНМ алгоритм навчання може зупинитись у локальному мінімумі, що зумовлює застосування алгоритмів глобальної оптимізації, які працюють досить повільно;

– немає строгої теорії щодо вибору типу і архітектури ШНМ, що спонукає до застосовування алгоритмів самоорганізації, які також працюють повільно;

– всю інформацію ШНМ отримує в процесі навчання і ніяку апріорну інформацію ввести в ШНМ неможливо.

6. НЕЧІТКІ МНОЖИНИ

6.1. Нечітка логіка. Історія виникнення

Теорія нечіткої логіки (*fuzzy logic*) була розроблена порівняно недавно. Лотфі Заде (Lotfi A. Zadeh), професор Університету Каліфорнії в Берклі і відомий тепер як засновник нечіткої логіки, зауважив, що традиційна двійкова комп'ютерна логіка не була здатна маніпулювати даними, котрі являють собою суб'єктивні або такі суто людські поняття, як «приваблива особистість» або «досить гаряче». І тому нечітка логіка була розроблена так, щоб дати змогу обчислювальним системам визначати розбіжності між даними «з відтінками сірого», аналогічно процесам людського мислення. У 1965 році Л. Заде опублікував свою фундаментальну роботу «Нечіткі множини», де описав математичну теорію нечітких множин і, як наслідок, нечітку логіку. Згідно з цією теорією, функції належності (або значення *істина* і *неправда*) відтепер визначались у діапазоні дійсних чисел від 0 до 1.

Хоча сама технологія і з'явилась у Сполучених Штатах, тамтешні вчені й дослідники проігнорували її здебільшого через «незручну» назву. Вони відмовлялися прийняти те, що звучало настільки «подитячому», несерйозно. Деякі математики доводили, що нечітка логіка була ні чим іншим, як замаскованим поняттям імовірності.

І тільки останнім часом нечітка логіка стала однією з найбільш успішних сучасних технологій розроблення досконалих систем керування. Нечітка логіка ідеально підходить для розв'язання таких задач з тієї простої причини, що об'єднує досвід людини у вирішенні проблем із здатністю обчислюваних засобів подавати їх у вигляді точних розв'язків на основі повної або часткової інформації. Вона займає важливе місце серед методів проектування технічних систем, які не обме-

жуються суто математичним підходом чи підходами, заснованими на чистій логіці (експертними системами).

Що ж являє собою «нечітка логіка»? Нечітка логіка – це надмножина традиційної (булевої) логіки, яка була розширена для того, щоб оперувати над *частково* істинними значеннями, розміщеними між «*повна істина*» і «*повна неправда*». Висновки в нечікій логіці швидше наближені, ніж точні. Важливість нечіткої логіки випливає з того, що більшість висновків, які робить людина, особливо ґрунтуючись на здоровому глузді, – неточні за своєю природою.

Найбільш суттєві такі характеристики нечіткої логіки:

– точний висновок є граничним випадком наближеного;

– все визначено лише в тій чи іншій мірі;

– будь-яку логічну систему можна розглядати у «нечіткому» сенсі, іншими словами, нечітка логіка є надмножиною традиційної логіки;

– знання інтерпретують як множину *не жорстких* або, що те ж саме, *нечітких* обмежень набору змінних;

– висновок розглядають як процес поширення нежорстких обмежень.

6.2. Класична теорія множин

Перед тим, як зайнятися детальним вивченням нечіткої логіки і нечітких множин, згадаємо, що являє собою класична «двозначна» теорія множин.

Множина – це набір об'єктів із спільними властивостями у певному контексті.

Наведемо декілька прикладів класичних множин:

– множина натуральних чисел, менших за 5: $A = \{1, 2, 3, 4\}$;

– коло одиничного радіуса у комплексній площині:

$A = \{z \mid z \in C, \mid z \mid \leq 1\}$;

– пряма на площині: $A = \{(x, y) \mid ax + by + c = 0, (x, y, a, b, c) \in \Re\}$.

Множина може бути задана (визначена) такими способами:

– переліком її елементів: $A = \{x_1, x_2, \ldots, x_n\}$;

– заданням її властивостей: $A = \{x \in \Re \mid x \leftarrow P\}$;

– функцією належності $\mu_A(x)$, яка за умови $X \to [0, 1]$ має такий вигляд:

$$\mu_A(x) = \begin{cases} 1, & x \in A \\ 0, & x \notin A \end{cases}.$$

Множина дійсних чисел в діапазоні від 5 до 10: $A = \{x \in \Re \mid 5 \le x \le 10\}$.

Графік функції належності такої множини подано на рис. 6.1.

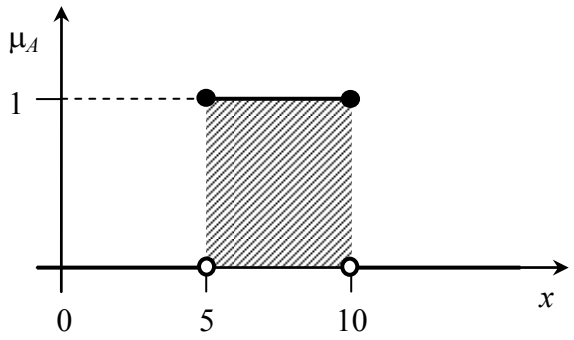


Рис. 6.1. Множина дійсних чисел в діапазоні [5, 10]

У останньому прикладі числа, для яких функція належності набуває значення 1, ми інтерпретуємо як елементи, що *належать* множині, а числа, для яких значення функції дорівнює 0, як такі, що *не належать* множині. Такої концепції задання та інтерпретації множин зазвичай достатньо для більшості застосувань. Проте, не важко знайти ситуації, в яких такому підходові вже не вистачає «гнучкості». Наприклад, розглянемо множину «молодих людей». Задавши верхню вікову межу поняття «молодий» у 20 років (нижня, природно, 0), маємо таку функцію належності (рис. 6.2).

У цьому випадку, додержуючись традиційної логіки, якщо вік індивідуума без одного дня 20 років, то він молодий, а наступного дня – вже ні. Бачимо, що в наведеному прикладі традиційна логіка не здатна адекватно відобразити реальної ситуації.

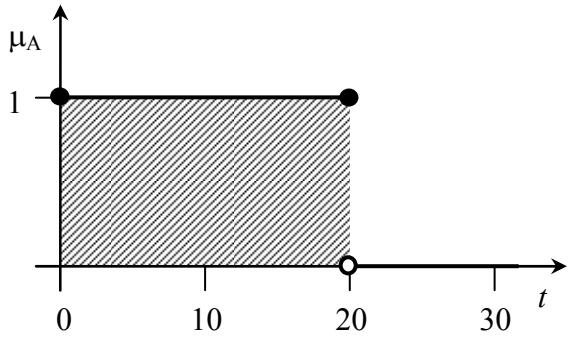


Рис. 6.2. Функція належності множини «молоді люди»

6.3. Елементи нечітких множин

Природним вирішенням подібної проблеми може бути послаблення *жорсткості* верхнього обмеження на належність множині. Задамо функції належності таким чином:

$$\mu_A(x) = \begin{cases} \text{не молодий}, x > 25 \\ \text{частково молодий}, x \in [15, 25] \\ \text{молодий}, 15 > x > 0 \end{cases}. \qquad (6.1)$$

Графік функції належності (6.1) наведено на рис. 6.3.

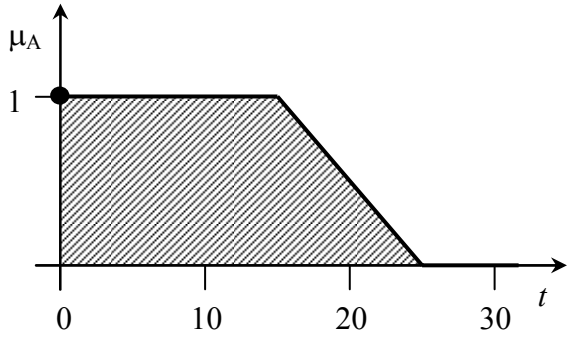


Рис. 6.3. «Нечітка» функція належності

У цьому випадку можна сказати, що коли вік людини 15 років, то ступінь належності до множини «молодий» дорівнює 1, якщо 20 років, то ступінь належності 0,5, а якщо вік більший за 25, то 0.

Таким чином, нечіткій множині можна дати таке визначення:

Нечіткою множиною A в деякому (непорожньому) просторі X, що позначається як $A \subseteq X$, називається множина пар $A = \{(x, \mu_A(x)); x \in X\}$, де $\mu_A : X \to [0, 1]$ – функція належності нечіткої множини A.

Ця функція приписує кожному елементу $x \in X$ міру його належності до нечіткої множини A. При цьому можна виділити три випадки:

1) $\mu_A(x) = 1$ означає повну належність елемента x до нечіткої множини A, тобто $x \in A$;

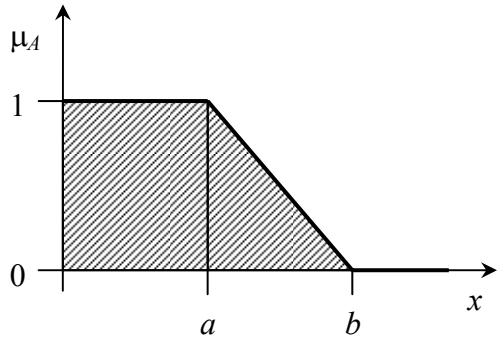
2) $\mu_A(x) = 0$ означає, що елемент x не належить до нечіткої множини A, тобто $x \notin A$;

3) $0 < \mu_A(x) < 1$ означає часткову належність елемента x до нечіткої множини A.

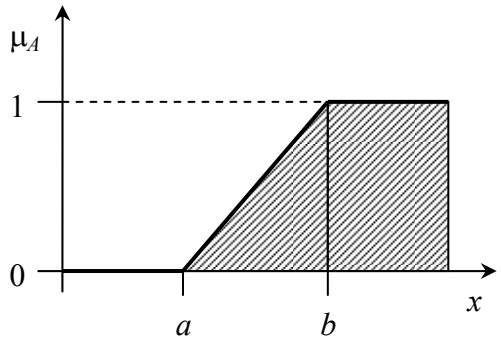
6.4. Види функцій належності та їх побудова

Для визначення вигляду функції належності розроблено різні експертні методи. У деяких випадках використовують типові форми функцій належності. Тоді методом експертних оцінок визначають тип функцій належності і їх параметри. Наведемо деякі типові функції належності.

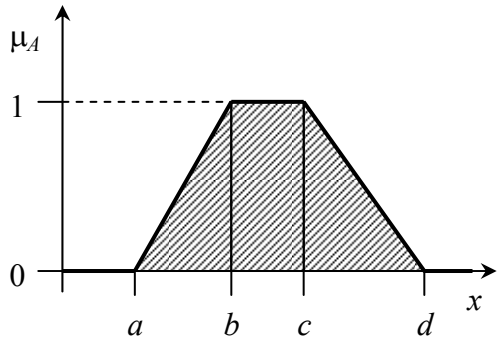
Кусково-лінійні (трапецієподібні і трикутні)



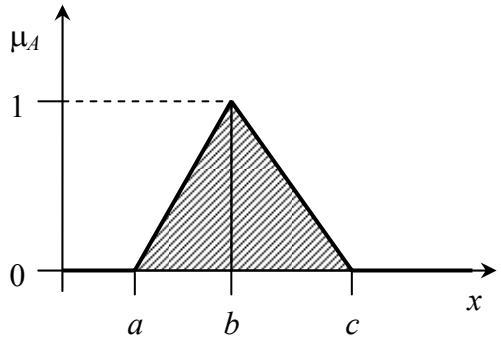
$$\mu_A(x) = \begin{cases} 1, & x < a \\ \dfrac{b-x}{b-a}, & a \le x \le b \\ 0, & x > b \end{cases}.$$



$$\mu_A(x) = \begin{cases} 0, & x < a \\ \dfrac{x-a}{b-a}, & a \le x \le b \\ 1, & x > b \end{cases}.$$



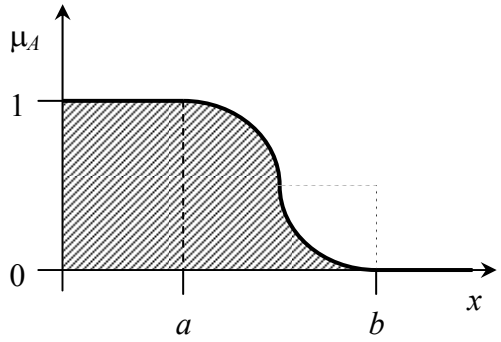
$$\mu_A(x)=\begin{cases}0,\ x<a\\ \dfrac{x-a}{b-a},\ a\le x\le b\\ 1,\ b<x<c\\ \dfrac{d-x}{d-c},\ c\le x\le d\\ 0,\ x>d\end{cases}. \qquad (6.2)$$



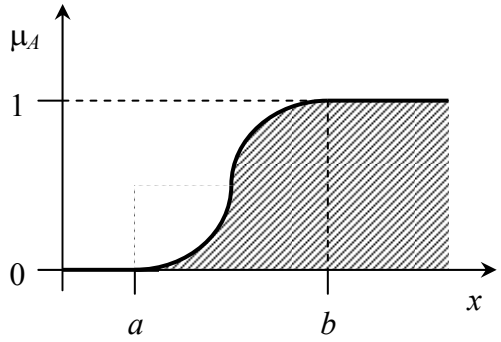
$$\mu_A(x)=\begin{cases}0,\ x<a\\ \dfrac{x-a}{b-a},\ a\le x\le b\\ \dfrac{c-x}{c-b},\ b<x\le c\\ 0,\ x>c\end{cases}. \qquad (6.3)$$

Бачимо, що трикутна функція належності (6.3) є частинним випадком узагальненої трапецієподібної (6.2).

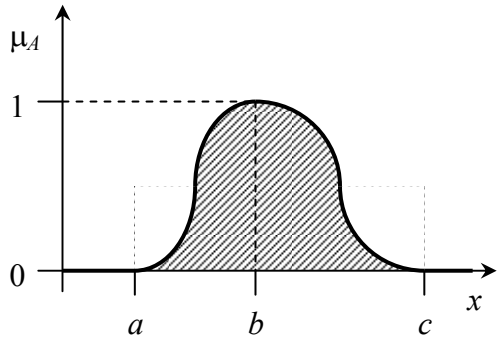
S-подібні та дзвоноподібні



$$\mu_A(x)=\begin{cases}1,\ x<a\\ 1-2\left(\dfrac{a-x}{a-b}\right)^2,\ a\le x\le \dfrac{a+b}{2}\\ 2\left(\dfrac{b-x}{a-b}\right)^2,\ \dfrac{a+b}{2}<x\le b\\ 0,\ x>b\end{cases}.$$



$$\mu_A(x)=\begin{cases}0,\ x<a\\ 2\left(\dfrac{x-a}{b-a}\right)^2,\ a\le x\le \dfrac{a+b}{2}\\ 1-2\left(\dfrac{x-b}{b-a}\right)^2,\ \dfrac{a+b}{2}<x\le b\\ 1,\ x>b\end{cases}.$$



$$\mu_A(x)=\begin{cases}0, & x<a\\ 2\left(\dfrac{x-a}{b-a}\right)^2, & a\le x\le\dfrac{a+b}{2}\\ 1-2\left(\dfrac{x-b}{b-a}\right)^2, & \dfrac{a+b}{2}<x\le b\\ 1-2\left(\dfrac{b-x}{b-c}\right)^2, & b<x\le\dfrac{b+c}{2}\\ 2\left(\dfrac{c-x}{b-c}\right)^2, & \dfrac{b+c}{2}<x\le c\\ 0, & x>c\end{cases}. \quad (6.4)$$

Слід зазначити, що дзвоноподібна функція належності (6.4) не обов'язково повинна бути симетричною.

Розглянемо приклад побудови нечітких функцій належності, виходячи з інформації, поданої у лінгвістичній формі.

Візьмемо, наприклад, таке словосполучення, як *температура повітря*. З погляду суб'єктивного сприйняття температури людиною, вона може відповідати трьом рівням (множинам): *холодно*, *тепло* і *гаряче*. Процес переходу до нечітких множин схематично подано на рис. 6.4.

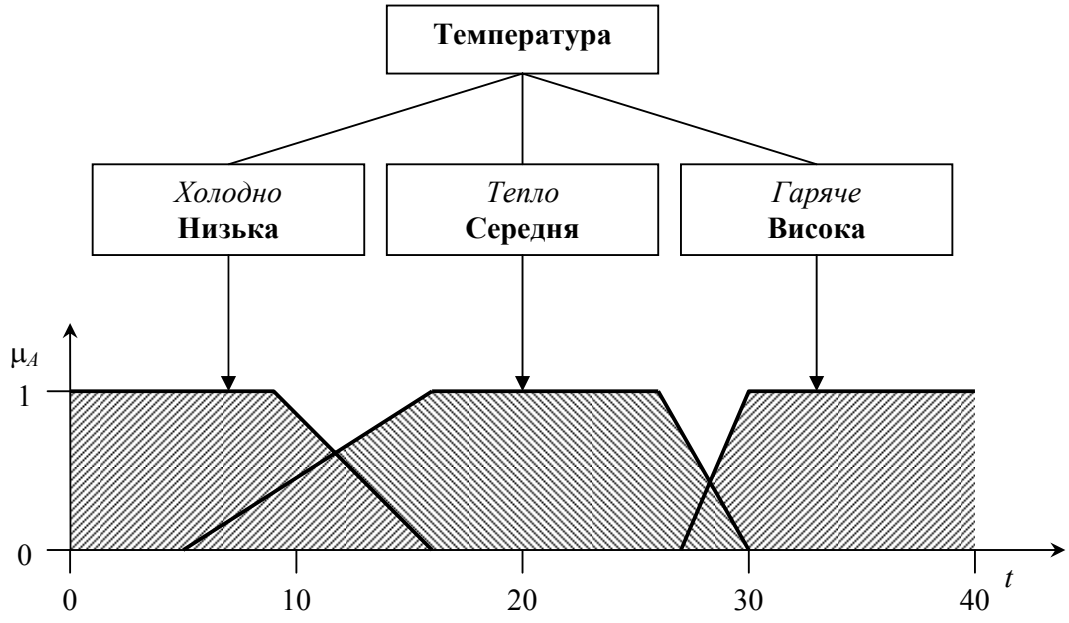


Рис. 6.4. Перехід до нечітких функцій належності

Слід зазначити, що слова «холодно», «тепло» і «гаряче», які відповідають нечітким функціям належності, повинні вибиратися так, щоб мати очевидний сенс і покривати весь діапазон можливих значень для базової змінної (в нашому випадку температури).

6.5. Операції над нечіткими множинами

Аналогічно із звичайною теорією множин, в теорії нечітких множин вводяться логічні операції над множинами: *перетин*, *об'єднання* та *доповнення* (заперечення).

Нехай A і B – нечіткі множини на універсальній множині E (рис. 6.5).

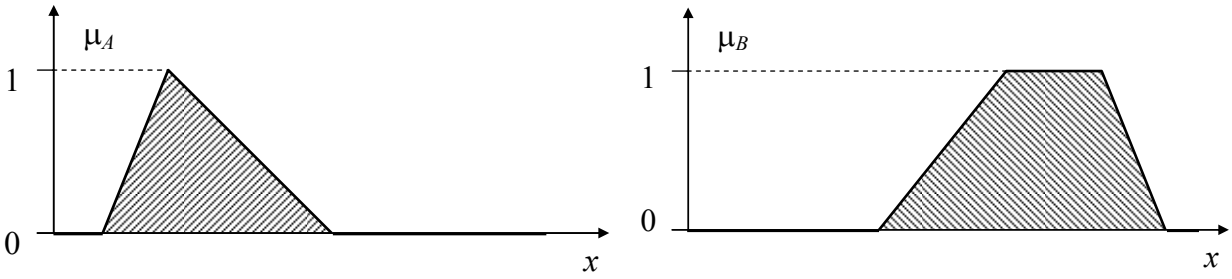


Рис. 6.5. Функції належності початкових множин A і B

Тоді можемо визначити такі операції над нечіткими множинами A і B.

Рівність

A і B рівнім, якщо $\forall x \in E$, $\mu_A(x) = \mu_B(x)$.

Перетин

Перетином нечітких множин A і B називають нечітку множину $A \cap B$, яка має для $\forall x \in E$ таку функцію належності:

$$\mu_{A \cap B}(x) = \mu_A(x) \wedge \mu_B(x) = \min(\mu_A(x), \mu_B(x)). \qquad (6.5)$$

Графічна інтерпретація цієї операції подана на рис. 6.6. (жирна суцільна лінія відповідає функції належності вислідної множини).

Об'єднання

Об'єднанням нечітких множин A і B називають нечітку множину $A \cup B$, яка має для $\forall x \in E$ таку функцію належності:

$$\mu_{A \cup B}(x) = \mu_A(x) \vee \mu_B(x) = \max(\mu_A(x), \mu_B(x)) . \qquad (6.6)$$

Графічна інтерпретація операції об'єднання подана на рис. 6.7.

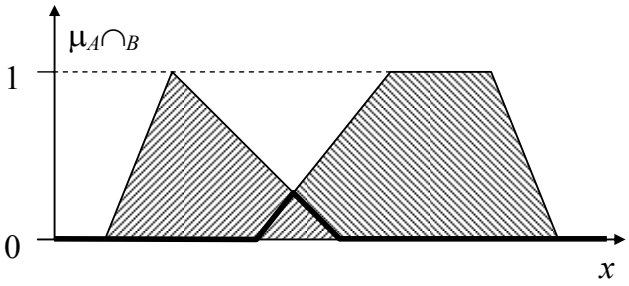


Рис. 6.6. Перетин двох множин

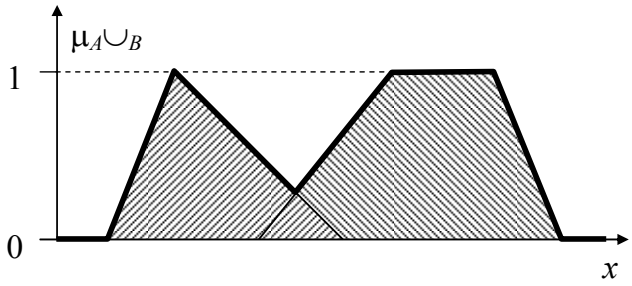


Рис. 6.7. Об'єднання двох множин

Доповнення

Доповненням нечіткої множини A називають нечітку множину $\overline{A}$, яка має для $\forall x \in E$ таку функцію належності: $\mu_{\overline{A}}(x) = 1 - \mu_A(x)$.

Графічну інтерпретацію операції доповнення подано на рис. 6.8.

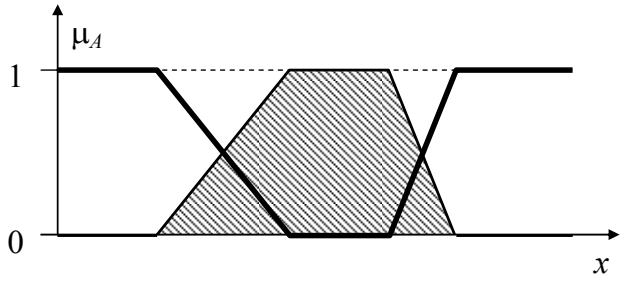


Рис. 6.8. Доповнення множини

Застосування операцій доповнення, об'єднання і перетину дає змогу взяти до уваги різноманітні змістові відтінки відповідних їм зв'язків «НІ», «АБО», «І».

Зазначимо також, що застосування операцій min і max у виразах (6.5) і (6.6) дозволяє для нечітких множин зберегти більшість властивостей традиційних множин, зокрема комутативність, асоціативність, ідемпотентність, дистрибутивність.

7. МЕТОДИ НЕЧІТКОГО ВИСНОВКУ

7.1. Правила висновку в традиційній логіці

У традиційній двійковій логіці висновки щодо істинності одних суджень виводять ґрунтуючись на істинності інших суджень. Розглянемо два правила висновку, які застосовують у двійковій логіці.

Modus Ponens

Умова	A
Імплікація	$A \to B$
Висновок	B

Нехай судження A – «це літак», а судження B – «він літає». Відповідно до правила *modus ponens*, якщо A правильне, то і B правильне. Іншими словами, з істинності передумови та імплікації випливає істинність висновку.

Modus Tollens

Умова	$\overline{B}$
Імплікація	$A \to B$
Висновок	$\overline{A}$

У цьому випадку з істинності передумови та імплікації випливає істинність висновку. Наприклад, якщо «він не літає», то «це не літак».

Наведені вище дві (із багатьох існуючих) схеми висновку в двійковій логіці можна узагальнити на випадок нечіткості.

7.2. Правила висновку в нечіткій логіці

Припустимо, що наявні у правилах *modus ponens* і *modus tollens* судження характеризуються деякими нечіткими множинами. Далі буде записувати залежності типу «якщо А, то В» використовуючи службові слова мов програмування: **if** А **then** В.

Modus Ponens

Умова	x **is** A'
Імплікація	**if** x **is** A **then** y **is** B
Висновок	y **is** B'

Modus Tollens

Умова	y **is not** B'
Імплікація	**if** x **is** A **then** y **is** B
Висновок	x **is not** A'

,

де A, A', B, B' – нечіткі множини, а x і y – нечіткі лінгвістичні змінні.

7.3. Нечітка імплікація

Функції належності в логічних висновках залежать від функції належності $\mu_{A\to B}(x,y)$ нечіткої імплікації $A\to B$, рівнозначної деякому нечіткому відношенню $R\subseteq X\times Y$. Подамо різні способи задання функції $\mu_{A\to B}(x,y)=\mu_R$ на основі відомих функцій належності $\mu_A(x)$ і $\mu_B(y)$.

Нехай A і B – це нечіткі множини, $A\subseteq X$, $B\subseteq Y$. Нечіткою імплікацією $A\to B$ називають відношення R, визначене на $X\times Y$, що відповідає таким правилам:

1. Правило типу «мінімум» (правило Мамдані)

$$\mu_{A\to B}(x,y)=\mu_A(x)\wedge\mu_B(y)=\min\left[\mu_A(x),\mu_B(y)\right].$$

2. Правило типу «добуток» (правило Ларсена)

$$\mu_{A\to B}(x,y)=\mu_A(x)\,\mu_B(y).$$

3. Правило Лукашевича

$$\mu_{A\to B}(x,y)=1\wedge\left[1-\mu_A(x)+\mu_B(y)\right]=\min\left[1,1-\mu_A(x)+\mu_B(y)\right].$$

4. Правило типу «максимум-мінімум» (правило Заде)

$$\mu_{A\to B}(x,y)=\left[\mu_A(x)\wedge\mu_B(y)\right]\vee\left[1-\mu_A(x)\right]=\max\left\{\min\left[\mu_A(x),\mu_B(y)\right],1-\mu_A(x)\right\}.$$

5. Бінарне правило (правило Кліна–Дейна)

$$\mu_{A\to B}(x,y) = [1-\mu_A(x)] \vee \mu_B(y) = \max[1-\mu_A(x), \mu_B(y)].$$

Окрім наведених є й інші означення нечіткої імплікації.

7.4. Нечіткий логічний висновок за методом Мамдані

Механізм нечіткого логічного висновку (*inference*) ґрунтується на знаннях, сформованих спеціалістами цієї предметною галузі у вигляді сукупності нечітких породжувальних правил (правил логічного висновку):

if x_1 **is** A_1 **and** x_2 **is** A_2 **and** … x_n **is** A_n **then** y **is** B

Частину правила перед ключовим словом **then** («то») називають умовою або передумовою (*antecendent*), а завершальну частину «y є B» – наслідком або висновком (*consequent*).

Проілюструємо механізм нечіткого логічного висновку на прикладі обчислень значень функції $y = f(x_1, x_2)$. Припустимо, що маємо базу знань, яка складається з двох правил:

R1: **if** x_1 **is** A_{11} **and** x_2 **is** A_{22} **then** y **is** B_1,

R2: **if** x_1 **is** A_{12} **or** x_2 **is** A_{22} **then** y **is** B_2,

де A_{ij} і B_i – це нечіткі множини, визначені для відповідних нечітких змінних, котрі мають функції належностей $\mu_{Aij}(x)$ і $\mu_{Bj}(y)$.

Тепер за наданими значеннями $x_1 = x_{10}$ і $x_2 = x_{20}$ знайдемо конкретне y_0. Слід зазначити, що цей приклад легко узагальнити для довільної кількості вхідних (x) і вихідних (y) змінних.

Для логічного висновку приходимо за чотири кроки:

Крок 1. **Введення нечіткості** (*fuzzification*). Для чітко заданих вихідних значень розраховують ступені належності до окремих множин. Для розглядуваного прикладу визначають числові значення $\mu_{A1j}(x_{10})$ і $\mu_{A2j}(x_{20})$.

Крок 2. **Нечітка імплікація**. Знаходять функції належності передумов кожного окремого правила за конкретних вхідних сигналів

$\alpha_j = \mu_{A1j}(x_{10}) \cap \mu_{A2j}(x_{20})$ – для оператора **and**.

$\alpha_j = \mu_{A1j}(x_{10}) \cup \mu_{A2j}(x_{20})$ – для оператора **or**.

Потім знаходять вислідні функції належності кожного правила

$$\mu_j(y) = \alpha_j \cap \mu_{Bj}(y)$$

Крок 3. **Нечітка композиція** (*aggregation*). Знаходять вислідну функцію належності всієї сукупності правил при вхідних сигналах x_{10} і x_{20}: $\mu_\Sigma(y) = \mu_1(y) \cup \mu_2(y)$.

Крок 4. **Зведення до чіткості** (*defuzzification*). Використовують, коли потрібно перетворити вихідну функцію належності у конкретне значення y_0. Крім розглянутих є багато різних методів зведення до чіткості, але найбільш поширений *центроїдний* метод.

Якщо в наведеному алгоритмі логічна операція перетину реалізується як функція «мінімум», а об'єднання – як «максимум» (рис. 7.1), то це *алгоритм Мамдані* (або Мамдані–Заде).

7.5. Методи зведення до чіткості

Одним з найбільш поширених методів зведення до чіткості (*defuzzification*) є *центроїдний* метод:

$$y_0 = y_c = \frac{\int_{-\infty}^{\infty} y\,\mu_\Sigma(y)dy}{\int_{-\infty}^{\infty} \mu_\Sigma(y)dy}. \qquad (7.1)$$

Як бачимо формула (7.1) досить важка для обчислень, тож часто в практичних розрахунках виконують наближені обчислення, замінюючи інтеграли відповідними сумами.

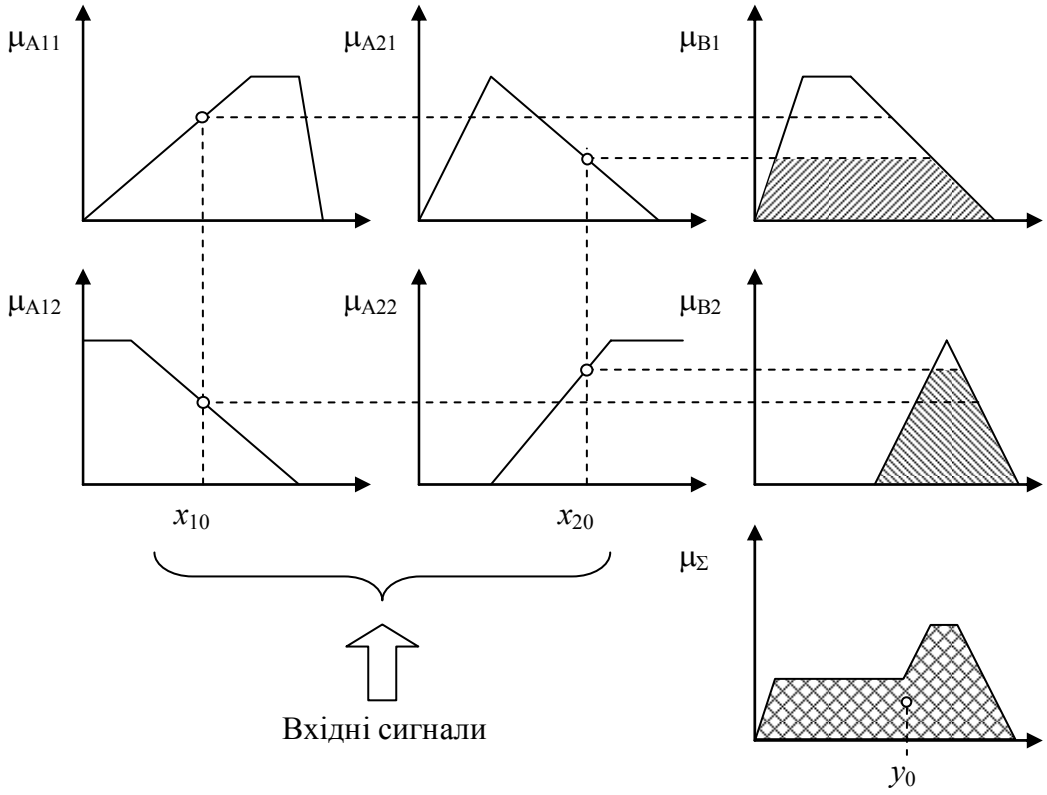


Рис. 7.1. Ілюстрація роботи алгоритму Мамдані

На практиці також часто використовуються такі методи:

– мінімальний максимум: результат y_{cl} – найменша точка, в якій $\mu_\Sigma(y)$ досягає максимуму;

– максимальний максимум: y_{cr} – найбільша точка, в якій $\mu_\Sigma(y)$ досягає максимуму;

– середній максимум: $y_0 = y_{cm} = \frac{1}{m}\sum_{i=1}^{m} y_{i\max}$, де $y_{i\max}$ – точки, в яких $\mu_\Sigma(y)$ досягає локальних максимумів, m – кількість максимумів;

– зведення до чіткості по висоті: елементи області визначення R, для яких значення функції належності, менші ніж певний рівень α, до

уваги не беруть; чітке значення знаходять за такою формулою: $y_0 = y_{ch} = \frac{\int_{C_\alpha} y\, C(y)dy}{\int_{C_\alpha} C(y)dy}$, де C_α – нечітка множина α-рівня.

7.6. Нечіткий логічний висновок за методом Сугено

На практиці широко застосовують алгоритм нечіткого логічного висновку *Сугено* (*Sugeno*), відомий також як алгоритм *Такагі–Сугено–Канга* (TSK). Відмінною рисою цього алгоритму є простота обчислень.

Проджувальні правила в алгоритмі Сугено мають такий вигляд:

if x_1 **is** A_1 **and** x_2 **is** A_2 **and** … x_n **is** A_n **then** $y = f_r(x_1,...,x_n)$,

де f_r – звичайна чітка функція; r – номер правила.

Принципова відмінність від алгоритму Мамдані в цьому разі – висновок, який подають у формі функціональної залежності.

Реалізація алгоритму Сугено складається із трьох кроків:

Крок 1. **Введення нечіткості**. Цілком аналогічне алгоритмові Мамдані.

Крок 2. **Нечітка імплікація**. Знаходяться функції належності передумов кожного окремого правила за конкретних вхідних сигналів x_{i0}:

$$\alpha_r,\ r = 1, 2, ..., m,$$

де m – кількість породжувальних правил. У класичному алгоритмі Сугено логічна операція перетину реалізується як min.

Крок 3. **Зведення до чіткості**. Визначається чітке значення вихідної змінної:

$$y_0 = \frac{\sum_{r=1}^{m} \alpha_r\ f_r(\overline{x}_0)}{\sum_{r=1}^{m} \alpha_r}.$$

Як функцію f_r часто використовують поліноми нульового порядку:

$$f_r(\bar{x}) = w_r ,$$

або першого порядку: $f_r(\bar{x}) = w_r + \sum_{j=1}^{n} p_{rj} x_j$, де w_r і p_{rj} – деякі сталі. Їх називають *алгоритмами Сугено нульового* або *першого порядку* відповідно.

Зазначимо, що відомий алгоритм Ванга–Менделя відрізняється від алгоритму Сугено нульового порядку тільки тим, що ступінь належності передумов правил у ньому знаходять за допомогою операції множення.

Існує безліч алгоритмів нечіткого висновку, які відрізняються набором вихідних правил, видом функцій належності, способами нечіткої імплікації та композиції, а також методом зведення до чіткості.

8. СИНТЕЗ СИСТЕМ З НЕЧІТКОЮ ЛОГІКОЮ

8.1. Вступ

Під час розв'язування більшості прикладних задач регулювання, інформацію, потрібну для побудови і реалізації системи керування, можна поділити на дві частини:

– *числову* (кількісну), що отримується з вимірювальних датчиків;

– *лінгвістичну* (якісну), що надходить від експерта.

Значна частина нечітких систем регулювання використовує другий вид знань, що частіше за все подають у формі бази нечітких правил.

У випадку, коли виникає потреба спроектувати нечітку систему, але наявні лише числові дані, ми стикаємося з серйозними проблемами. Один із способів їх розв'язання – це застосування так званих *нейро-нечітких* (*neuro-fuzzy*) систем. Вони мають багато переваг, але їх база правил повільно наповнюється знаннями в процесі ітеративного навчання.

8.2. Нечіткі нейронні мережі

Основна перевага систем з нечіткою логікою – це здатність використовувати умови і методи розв'язування задач, описані мовою, близькою до природної. Однак класичні системи з нечіткою логікою, нездатні навчатися автоматично, мають недоліки. Набір нечітких правил, вид і параметри функцій належності, що описують вхідні і вихідні змінні системи, а також вид алгоритму нечіткого висновку, вибираються суб'єктивно експертом-людиною, і вони не завжди відповідають дійсності.

Для усунення вказаного недоліку був запропонований апарат нечітких нейронних мереж (*fuzzy neural networks*). Ці системи відомі та-

кож як *адаптивні нейро-нечіткі системи висновку* (*Adaptive Neuro-Fuzzy Inference System, ANFIS*).

Нечітка нейронна мережа – це багатошарова нейронна мережа, в якій шари виконують функції елементів системи нечіткого висновку. Нейрони такої мережі характеризуються набором параметрів, настроювання яких виконують під час навчання, як у звичайних ШНМ. На рис. 8.1 зображена нечітка ШНМ на базі алгоритму Сугено нульового порядку, за якого значення вихідної змінної обчислюють за формулою

$$y = \frac{\sum_{r=1}^{m} \alpha_r \, w_r}{\sum_{r=1}^{m} \alpha_r},$$

де α_r – значення функцій належності передумов кожного окремого правила *r* за конкретних значень вхідних сигналів.

Шар 1 здійснює зведення до нечіткості. Нелінійні функції $\mu_{ri}(x_i)$, де *r* – номер продукційного правила, а *i* – номер компоненти вхідного вектора, відповідають функціям належності передумов правил. Настроювані параметри цього шару – параметри використовуваних функцій належності.

Шар 2 обчислює вислідні функції належності передумов нечітких правил. У цьому випадку шар не має настроюваних параметрів.

Шар 3 (складається з двох нейронів) і виконує додавання і зважене додавання вихідних сигналів шару 2. Параметрами цього шару є вагові коефіцієнти w_r.

Шар 4 реалізує операція ділення $y = f_1 / f_2$. Він не містить настроюваних параметрів.

Якщо в такій мережі використовувати функції належності гауссового типу, а для обчислення вислідних функцій належності переду-

мов правил (шар 2) – операцію множення замість операції «мінімум», то отримаємо досить поширену нечітку мережу Ванга–Менделя.

Для мережі Ванга–Менделя можна аналітично подати градієнт функції похибки від параметрів мережі, що дає змогу використовувати для її навчання метод зворотного поширення похибки, котрий застосовують у багатошарових персептронах.

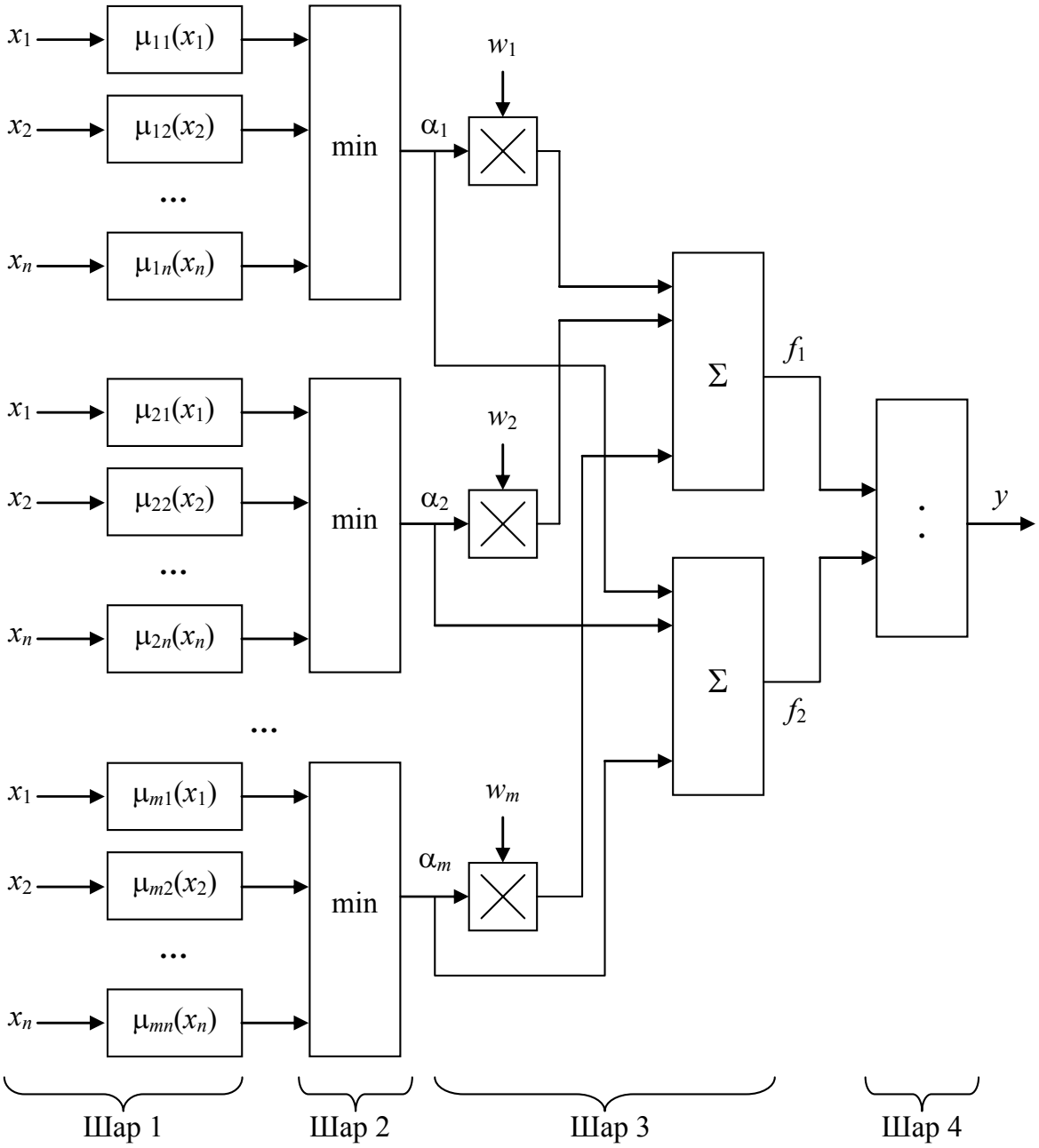


Рис. 8.1. Нечітка нейронна мережа на основі алгоритму Сугено

8.3. Синтез нечітких правил на основі числових даних

Розглянемо один з найпростіших, але в той же час досить універсальний метод побудови бази нечітких правил на основі числових даних. Переваги цього методу полягають в його простоті і високій ефективності. Крім того, він дає змогу об'єднувати числову інформацію, подану у формі даних для навчання, з лінгвістичною інформацією у вигляді бази правил доповненням наявної бази правилами, створеними на основі числових даних.

Припустимо, що ми створюємо базу правил для нечіткої системи з двома входами і одним виходом. Очевидно, для цього потрібні дані для навчання у вигляді множини пар $(x_1(i), x_2(i), d(i))$, де $x_1(i), x_2(i)$ – сигнали, які подаються на вхід модуля нечіткого керування, а $d(i)$ – очікуване (еталонне) значення вихідного сигналу.

Задача полягає у формуванні нечітких правил, щоб побудований на їх основі модуль керування під час отримання вхідних сигналів генерував коректні (що мають мінімальну похибку) вихідні сигнали.

Крок 1. Поділ простору вхідних і вихідних сигналів на інтервали

Припустимо, що нам відоме мінімальне і максимальне значення кожного сигналу. За ними можна визначити інтервали, в яких містяться припустимі значення. Наприклад, для вхідного сигналу x_1 такий інтервал позначимо $[x_1^-, x_1^+]$. Якщо значення x_1^- і x_1^+ невідомі, то можна скористатися навчаючими даними і вибрати з них відповідно мінімальне і максимальне значення:

$$x_1^- = \min(x_1), \quad x_1^+ = \max(x_1).$$

Аналогічно для сигналу x_2 визначимо інтервал $[x_2^-, x_2^+]$, а для бажаного вихідного сигналу d – інтервал $[d^-, d^+]$.

Кожен визначений у такий спосіб інтервал поділимо на $2n + 1$ (непарне число) інтервалів (відрізків), причому значення n для кожно-

го сигналу підбирається індивідуально, а відрізки можуть мати однакову або різну довжину. Окремі інтервали позначимо так: S_n (малий n), ... , S_n (малий 1), M (середній), B_1 (великий 1), ... , B_n (великий n).

На рис. 8.2 подано приклад такого поділу, де область визначення сигналу x_1 розбита на п’ять інтервалів ($n = 2$), сигналу x_2 – на сім інтервалів ($n = 3$), а область визначення вихідного сигналу y – на п’ять інтервалів ($n = 2$).

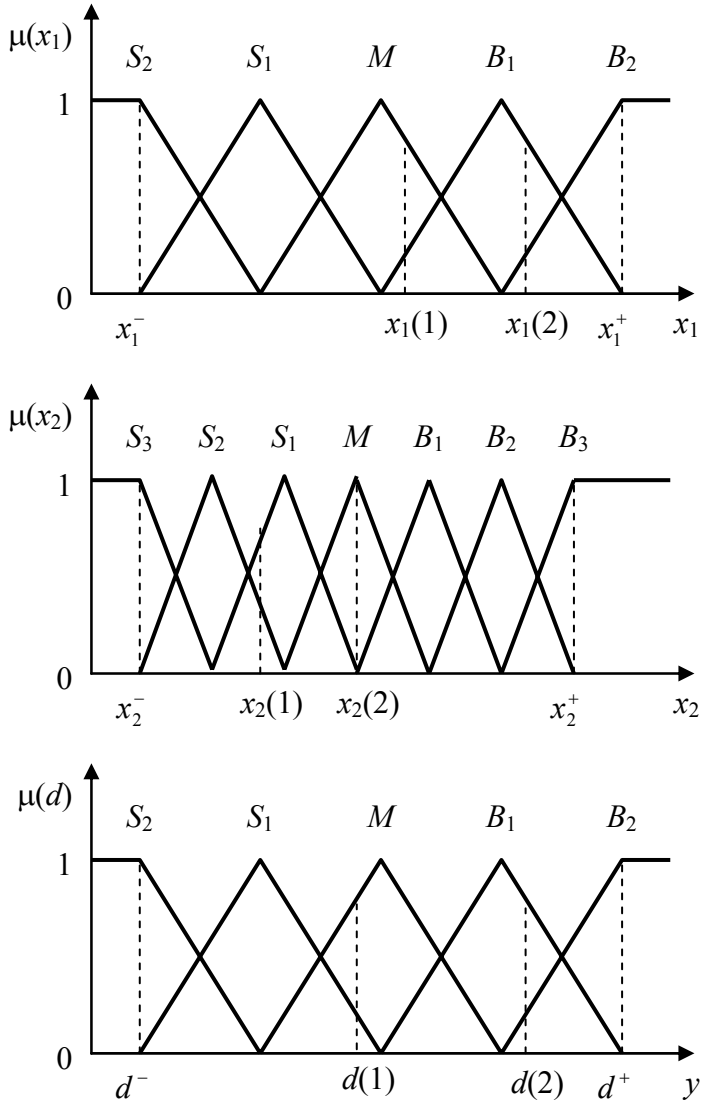


Рис. 8.2. Поділ простору сигналів на інтервали

Кожна функція належності в даному прикладі має трикутну форму. Одна з вершин розміщена у центрі відрізка і їй відповідає значення функції, що дорівнює 1. Дві інших вершини лежать в центрах сусідніх відрізків. Їм відповідають значення функції належності, що дорівнюють 0. Зазначимо, що можна запропонувати багато інших способів поділу вхідного і вихідного простору на окремі відрізки і використовувати інші форми функцій належності.

Крок 2. Побудова нечітких правил на основі даних для навчання

Визначимо ступені належності даних для навчання кожного відрізка, виділеного на кроці 1.

Ці ступені будуть виражені значеннями функцій належності відповідних нечітких множин для кожної групи даних.

Тепер співставимо дані для навчання $(x_1(i), x_2(i), d(i))$ з областями значень, в яких вони мають максимальні ступені належності. Наприклад, для випадку, показаного на рис. 8.2, $x_1(1)$ має найбільший ступінь належності до інтервалу M, а $x_2(1)$ – до інтервалу S_1. Остаточно для кожної пари даних для навчання можна записати правило:

$$\begin{aligned}
&\{x_1(1), x_2(1); d(1)\} \Rightarrow \\
&\{x_1(1)[\max = 0.8 \text{ in M}], x_2(1)[\max = 0.6 \text{ in } S_1]; d(1)[\max = 0.8 \text{ in M}]\} \Rightarrow \\
&\text{R}(1): \textbf{ if } x_1 \textbf{ is M and } x_2 \textbf{ is } S_1 \textbf{ then } y \textbf{ is M}. \\
&\{x_1(2), x_2(2); d(2)\} \Rightarrow \\
&\{x_1(2)[\max = 0.8 \text{ in } B_1], x_2(2)[\max = 1 \text{ in M}]; d(2)[\max = 0.8 \text{ in } B_1]\} \Rightarrow \\
&\text{R}(2): \textbf{ if } x_1 \textbf{ is } B_1 \textbf{ and } x_2 \textbf{ is M then } y \textbf{ is } B_1.
\end{aligned} \tag{8.1}$$

Крок 3. Приписування кожному правилу ступеня істинності

Зазвичай є багато пар даних для навчання за кожною з яких можна сформулювати одне правило. Тому є висока ймовірність того, що

деякі з цих правил виявляться взаємовиключальними. Це стосується правил з одним і тим же посиланням (умовою), але з різними наслідками (висновками).

Один з методів вирішення цієї проблеми полягає у приписуванні кожному правилу так званого *ступеня істинності* з подальшим вибором серед правил, що суперечать одне одному, того, у якого цей ступінь виявиться найбільшим. Так не тільки вирішують проблему взаємовиключальних правил, але й значно зменшується їх кількість. Для правила вигляду

R: **if** x_1 **is** A_1 **and** x_2 **is** A_2 **then** y **is** B

ступінь істинності, що позначається як SP(R), визначається як

$$\mathrm{SP(R)} = \mu_{A1}(x_1)\ \ \mu_{A2}(x_2)\ \ \mu_B(y).$$

Наприклад, правило R(1) з (8.3) матиме ступінь істинності

$$\mathrm{SP(R_1)} = \mu_M(x_1)\,\mu_{S1}(x_2)\,\mu_M(y) = 0{,}8 \cdot 0{,}6 \cdot 0{,}8 = 0{,}384.$$

Крок 4. Створення бази даних нечітких правил

Спосіб побудови бази нечітких правил подано на рис. 8.3.

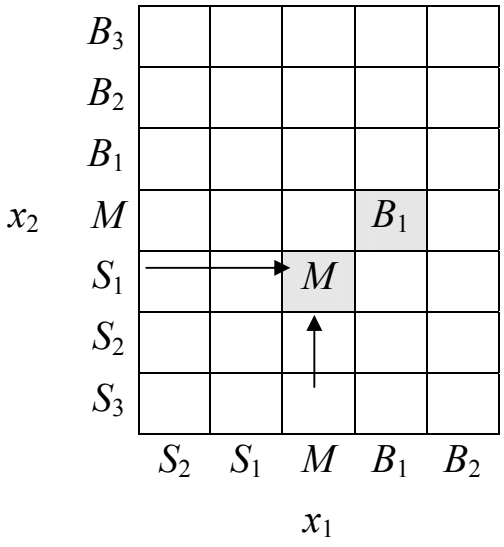


Рис. 8.3. Форма бази нечітких правил

Ця база зображена у вигляді таблиці, яку заповнюють нечіткими правилами таким чином: якщо правило має вигляд як, наприклад, правило R(1) у (8.1), то на перетині рядка S_1 і стовпця M вписують назву нечіткої множини, отриманої в результаті, тобто M (відповідної вихідному сигналові y). Якщо є декілька нечітких правил з одним і тим же посиланням, то з них вибирають те, яке має найвищий ступінь істинності.

Крок 5. Зведення до чіткості

Наша задача зазвичай полягає у визначенні за допомогою бази правил відображення $f:(x_1,x_2)\to y$, де y – вихідна величина нечіткої системи. Під час визначення кількісного значення керуючого впливу y для даних вхідних сигналів потрібно виконувати операцію зведення до чіткості.

Спочатку для вхідних сигналів (x_1, x_2) з використанням операції добутку об'єднаємо посилання (умови) k-го нечіткого правила. Так визначають ступінь активності k-го правила. Його значення знаходять за формулою

$$\tau^{(k)} = \mu_{A1}^{(k)}(x_1)\,\mu_{A2}^{(k)}(x_2)\,.$$

Для розрахунку вихідного значення y скористаємось способом зведення до чіткості за середнім центром:

$$y = \frac{\sum_{k=1}^{N} \tau^{(k)}\ y^{(k)}}{\sum_{k=1}^{N} \tau^{(k)}}\,.$$

Розглянутий метод можна легко узагальнити для нечіткої системи з будь-якою кількістю входів і виходів.

8.4. Нечітка одноелементна модель

Ефективність застосування апарату нечіткої логіки ґрунтується на теоремах, аналогічних теоремам про повноту для нейронних мереж, суть яких полягає в тому, що система на основі алгоритму нечіткого виводу під час виконання визначених, але не дуже жорстких умов – це універсальний апроксиматор.

Найбільш простою системою апроксимації функцій є нечітка одноелементна модель (*singleton fuzzy model*), що ґрунтується на системі Сугено нульового порядку:

$$\textbf{if } x \textbf{ is } \mathrm{A}_i \textbf{ then } y = w_i. \qquad (8.2)$$

До довільної функціональної залежності $y = f(x)$ може бути застосована процедура кусково-лінійної апроксимації, як показано на рис. 8.4.

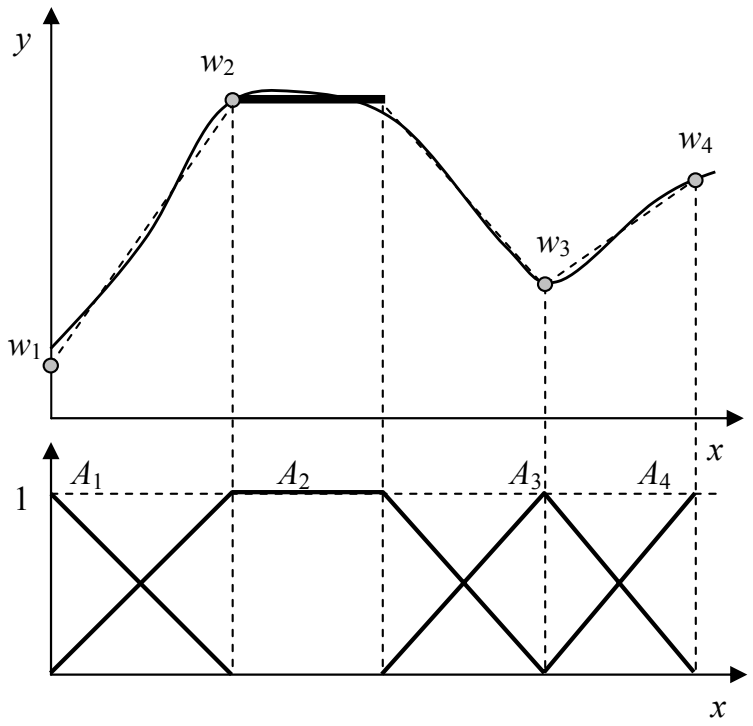


Рис. 8.4. Кусково-лінійна апроксимація і синтез множин

Вузли кусково-лінійної апроксимуючої функції відповідатимуть значенням коефіцієнта функції наслідку w_i в моделі (8.2). Висновок і зведення до чіткості здійснюються за формулою

$$y = \frac{\sum_{i=1}^{m} \mu_{Ai}(x)\, w_i}{\sum_{i=1}^{m} \mu_{Ai}(x)}.$$

Наведений підхід має такі переваги:

– добре прогнозовані апроксимаційні властивості;

– просте отримання параметрів.

Це викладення методів синтезу нечітких систем не можна вважати повним. Слід пам’ятати, що є багато алгоритмів і методів, які різняться набором вихідних правил, виглядом функцій належності, способами нечіткої імплікації і композиції, а також методом зведення до чіткості.

9. АЛГОРИТМИ ЗНАХОЖДЕННЯ ОПТИМАЛЬНИХ ТРАЄКТОРІЙ

9.1. Загальні положення

Серед всіх алгоритмів, що належать до стратегічного і тактичного рівнів структури інтелектуальної системи керування рухомим об'єктом, алгоритми пошуку оптимальної траєкторії, враховуючи обмеження, накладені перешкодами, є найбільш важливими для успішного розв'язання задач керування. В загальному вигляді такі задачі формулюються так: *знайти шлях для руху об'єкта із початкового положення в кінцеве*. Немає потреби нагадувати, що розв'язання такої задачі нетривіальне, це зумовлено різними перешкодами на шляху об'єкта, котрі блокують просування до цілі. Проте розрізняють безліч алгоритмічних підходів до розв'язання цієї задачі. Хоча не всі вони однаково ефективні, але їх послідовне розглядання дає змогу розібратися у проблемах, що виникають, і способах їх вирішення. У загальному вигляді алгоритм обходу перешкод такий:

поки ціль не досягнута
  вибрати напрям руху до цілі
  якщо шлях вільний,
    то рухатись
      інакше вибрати інший напрям відповідно
        до вибраного *алгоритму обходу перешкоди*

Розглядувані алгоритми дають змогу обходити перешкоди під час руху за картою, заданою у вигляді прямокутного масиву значень, що характеризують ступінь «прохідності» цієї ділянки. Попри уявну обмеженість такого підходу, більшість алгоритмів можна узагальнити для інших способів задання простору руху.

9.2. Найпростіші алгоритми обходу перешкод

Найбільш прості алгоритми обходу перешкод – це *метод руху у випадковому напрямку* і *метод відслідковування меж перешкод*.

Рух у випадковому напрямку

Якщо перешкода невелика і опукла, то об’єкт швидше за все зможе її обійти, рухаючись від неї, і намагаючись рухатись навмання в іншому напрямку доти, доки досягне не цілі. На рис. 9.1 показано роботу цього алгоритму.

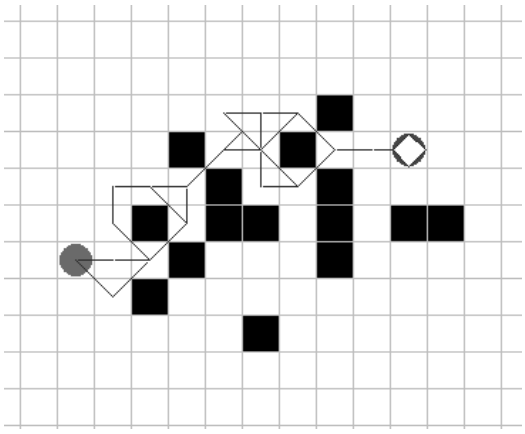


Рис. 9.1. Алгоритм руху у випадковому напрямку

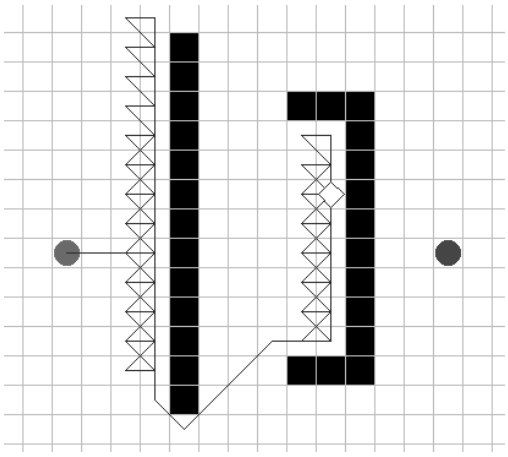


Рис. 9.2. Збій алгоритму на увігнутих перешкодах

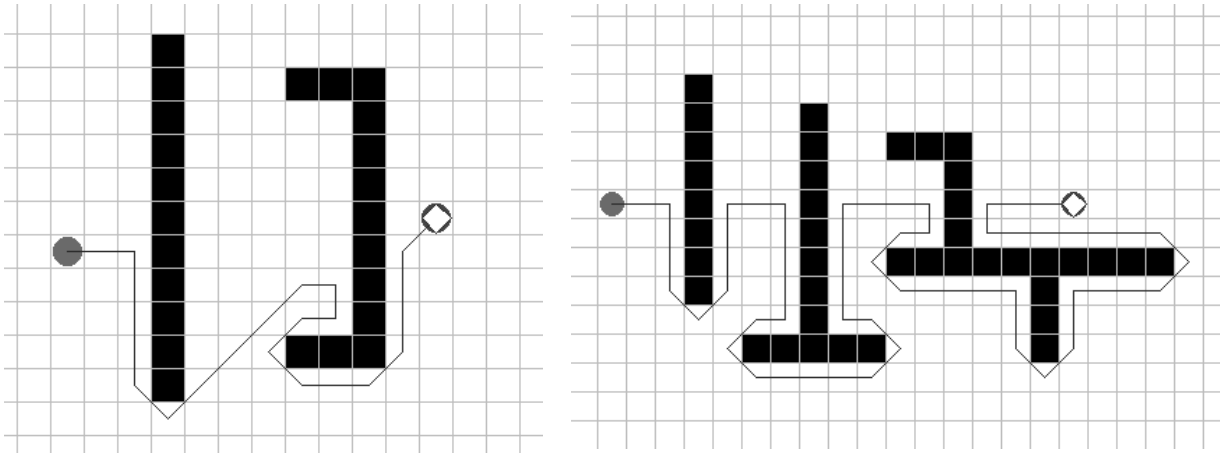
Тут і далі сірим кружечком позначено рухомий об’єкт, білим – ціль руху, а чорними квадратами – перешкоди.

Проблеми виникають тоді, коли перешкоди досить великі або коли вони увігнуті, як показано на рис. 9.2. Об’єкт може здійснити чимало спроб, перш ніж правильний шлях буде знайдено, або правильного шляху не буде знайдено взагалі. На щастя, за допомогою наступного методу об’єкт успішно долає подібні перешкоди.

Метод відслідковування меж перешкод

Якщо перешкода досить велика і не має опуклої форми, то після зіткнення із нею можна спробувати її обійти, прямуючи за видимими

обрисами доти, доки напрямок на ціль не буде блокуватися цією перешкодою. На рис. 9.3, *а* і 9.3, *б* показано, як цей алгоритм вирішує проблеми, з котрими не може впоратись алгоритм руху у випадковому напрямку.



а *б*

Рис. 9.3. Метод відслідковування обрисів перешкод

Основною перевагою наведених вище алгоритмів є те, що вони не потребують апріорної інформації про всі перешкоди, розміщені на шляху до цілі. Обхід перешкод відбувається по мірі їх виникнення, що знижує вимоги до пам'яті для таких алгоритмів. З другого боку, бачимо, що знайдені траєкторії не оптимальні за довжиною. У такому разі єдиний розумний підхід – це планування всього шляху перед початком переміщень. Такі алгоритми відомі під загальною назвою *методи обходу зв'язних графів*. Крім того, ці методи дозволяють вирішувати проблему пошуку оптимальної траєкторії з урахуванням різної «вартості» руху за різними елементами карти.

9.3. Метод «пошуку в ширину»

Найбільш простим алгоритмом серед методів обходу зв'язних графів – це метод «пошуку в ширину» (*breadth-first search*). Принцип його роботи такий: починаючи з першого вузла (положення), цей ал-

горитм перевіряє спочатку всі вузли, що межують з ним, потім всі вузли в двох кроках від нього, потім у трьох кроках, і так далі доти, доки цільового вузла (положення) не буде знайдено. Зазвичай всі необстежені сусідні вузли вміщуються у список «відкритих» вузлів, який є стеком типу «першим зайшов, першим вийшов» (FIFO, *first-in-first-out*). Алгоритм цього методу такий:

```
function BreadthFirstSearch : Boolean;
var
   n, n', s : node;
   Open     : queue;
begin
   s.parent := nil;  // s is a node for the start
   push s on Open
   while Open is not empty do begin
      pop node n from Open
      if n is a goal node then begin                    (9.1)
        construct path
        result := true; // success
        exit;
      end;
      for each successor n' of n do begin
        if n' has been visited already,
          then continue;
        n'.parent := n;
        push n' on Open
      end;
   end;
   result := false;  // no path found
end
```

На рис. 9.4 показано результат роботи алгоритму (9.1).

Бачимо, що алгоритм дійсно знаходить шлях обходу перешкод і цей шлях гарантовано оптимальний (найкоротший) за умови, що рух усіма дозволеними елементами карти має однакову «вартість». Проте є декілька проблем, пов’язаних з алгоритмом «пошуку в ширину». По-перше, пошук здійснюється в усіх напрямках, замість того, щоб проводити його за напрямком до цілі. По-друге, не всі кроки мають однакову довжину. Перехід на сусідній елемент по діагоналі довший, ніж перехід під прямим кутом.

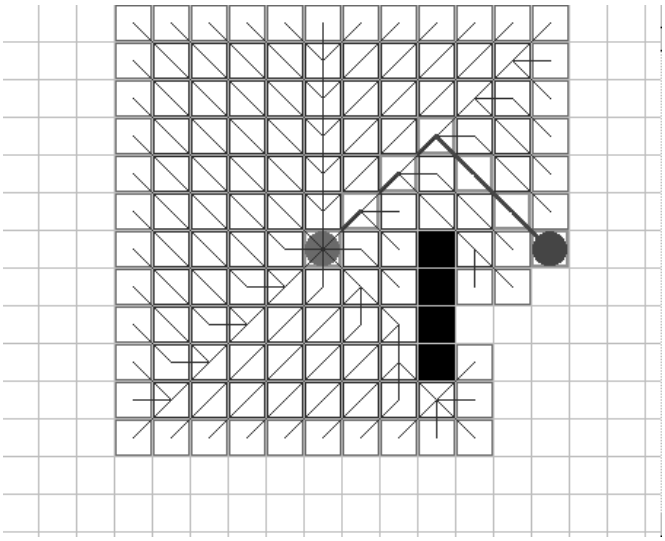


Рис. 9.4. Робота алгоритму «пошуку в ширину»

Одне з можливих поліпшень такого алгоритму полягає в тому, що пошук ведеться одночасно від початкового положення до цілі та від цілі до початкового положення. Результат роботи такої модифікації подано на рис. 9.5.

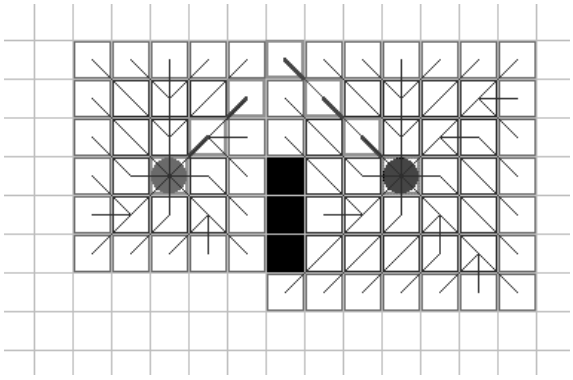


Рис. 9.5. Двонапрямлений пошук у ширину

Подібні методи поліпшення можуть використовуватись і для інших алгоритмів пошуку.

9.4. Метод «пошуку в глибину»

Логічною альтернативою методу «пошуку в ширину» став метод «пошуку в глубину» (*depth-first search*), за якого пошук ведеться переважно в бік віддалення від розглядуваного вузла, а не в бік пошуку в сусідніх до нього вузлах. Очевидно, що в цьому випадку потрібні додаткові умови припинення такого процесу, наприклад, за досягнення заданої глибини пошуку.

З погляду реалізації цей алгоритм відрізняється від методу «пошуку в ширину» (9.1) лише тим, що замість черги FIFO використовують чергу типу «останній зайшов, перший вийшов» (LIFO, *last-in-first-out*).

9.5. Алгоритм Дейкстри

Дейкстра (*Edsger Wybe Dijkstra, 1930–2002*) розробив класичний алгоритм обходу графів із зв’язками, що мають різні вагові коефіцієнти. На кожному кроці алгоритм шукає не оброблений раніше вузол, найближчий до початкового, обробляє його «сусудів» і встановлює їх відстані від початкового. Такий підхід має деякі переваги перед методом «пошуку в ширину». По-перше, враховується відстань від одного вузла до іншого, а по-друге, поновлюється інформація про відстань до розглядуваного вузла, якщо кращу траєкторію було знайдено. Реалізація алгоритму Дейкстри відрізняється від методу «пошуку в ширину» тільки тим, що замість черги FIFO використовують пріоритетну чергу, в якій спершу обробляється вузол, який має найкращу «вартість» у розумінні відстані від початкового вузла. Результат роботи алгоритму подано на рис. 9.6.

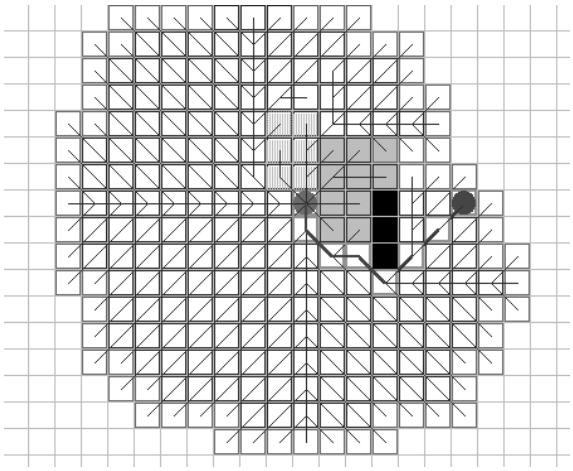


Рис. 9.6. Алгоритм Дейкстри

Не зважаючи на те, що алгоритм Дейкстри успішно вирішує проблему оцінювання оптимальності вибраного шляху, розрахувати напрямок на ціль все ще неможливо.

9.6. «Зірка пошукових алгоритмів» – A*

Найбільш успішним із усіх алгоритмів пошуку оптимальних шляхів обходу перешкод є алгоритм A* (вимовляється як «*ей-стар*»). Евристичний пошук оцінює кожен вузол пропорційно до ефективності траєкторії, що проходить через нього. Типова формула для оцінювання рейтингу вузла має такий вигляд:

$$f(n) = g(n) + h(n),$$

де $f(n)$ – рейтинг ефективності, що присвоюється вузлу n; $g(n)$ – найбільш ефективна «вартість» шляху з початкового вузла в даний; $h(n)$ – евристична оцінка «вартості» прямування з даного вузла в цільовий.

Наведемо реалізацію алгоритму A*.

```
function AStarSearch : Boolean;
var
   n, n', s : node;
   Open     : priorityqueue;
   Closed   : list;
```

```
begin
    s.parent := nil;     // s is a node for the start
    s.g := 0;            // s is the start node
    s.h := GoalDistEstimate( s );
    s.f = s.g + s.h;

    push s on Open
    while Open is not empty do begin
        pop node n from Open
        if n is a goal node then begin
          construct path
          result := true; // success
          exit;
        end;
        for each successor n' of n do begin
           newg := n.g + cost(n,n');
           if (n' is in Open or Closed) and n'.g <= newg
             then continue;
           n'.parent := n;
           n'.g := newg;
           n'.h := GoadDistEstimate( n' );
           n'.f := n'.g + n'.h;
           if n' is in Closed then remove n' from Closed;
           if n' is not in Open then push n' on Open;
        end;
        push n on Closed
    end;
    result := false;  // no path found
end
```

Алгоритм A* має декілька цікавих властивостей:

– він гарантовано знаходить найкоротший (оптимальний) шлях, якщо евристична оцінка $h(n)$ прийнятна у тому розумінні, що вона ніколи не більша від істинної відстані до цілі;

– кількість обчислень евристичної функції мінімальна. Це означає, що ніякий інший алгоритм, що використовує ту ж саму функцію $h(n)$, не обробить менше вузлів, ніж алгоритм A*.

Приклад роботи алгоритму A* для розглянутої задачі подано на рис. 9.7.

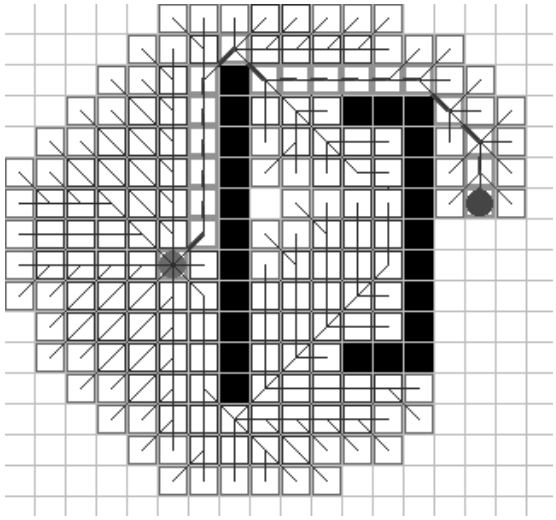


Рис. 9.7. Результат роботи алгоритму A*

9.7. Різні форми задання простору пошуку

Було показано роботу алгоритмів з дискретними картами, зображуваних у вигляді двовимірного прямокутного масиву даних. Але у реальних застосуваннях таке подання не завжди відповідає дійсності. Однак є декілька способів перетворення простору пошуку до вигляду, придатного для пошукових алгоритмів.

Дуже часто блокуючі рух об’єкти у просторі пошуку можуть бути задані у векторному вигляді, як показано на рис. 9.8, *а*. Розглянемо способи подання інформації про об’єкти у вигляді, придатному для пошукових алгоритмів.

Зведення до дискретного масиву даних

Найбільш очевидним підходом буде відобразити інформацію про об'єкти на двовимірний масив даних. У цьому разі комірки карти, де об'єктів зовсім немає, можуть розглядатися як вільні, а ті комірки, які зайняті об'єктами повністю або частково, як непрохідні (рис. 9.8, *б*). Єдиним недоліком такого підходу є велика кількість елементів у просторі пошуку.

Задання об'єктів вершинами у зоні прямої видимості

Найбільш критичні елементи об'єктів з погляду алгоритмів пошуку оптимального шляху – це їх вершини. Таким чином, задавши для кожної з вершин вузол зв'язного графу, що їй відповідає, і задавши зв'язки між вузлами, які знаходяться в зоні прямої видимості один від одного, отримуємо прийнятне подання для простору пошуку (рис. 9.8, *в*). Недоліком такого підходу є те, що вислідна траєкторія буде прокладена близько до вершин об'єктів; це може бути неприпустимим з погляду гарантування безпеки руху.

Метод опуклих багатокутників

Альтернативою попередньому може бути використання методу опуклих багатокутників (рис. 9.8, *г*). У цьому випадку вільний від об'єктів простір розбивають на опуклі багатокутники. Вузли простору пошуку розміщують у серединах сторін, вільних від об'єктів, і з'єднуються з іншими вузлами за прямої видимості. Таке розбиття хоча й гарантує безпечну відстань до блокуючих об'єктів, але не забезпечує найкоротшої траєкторії.

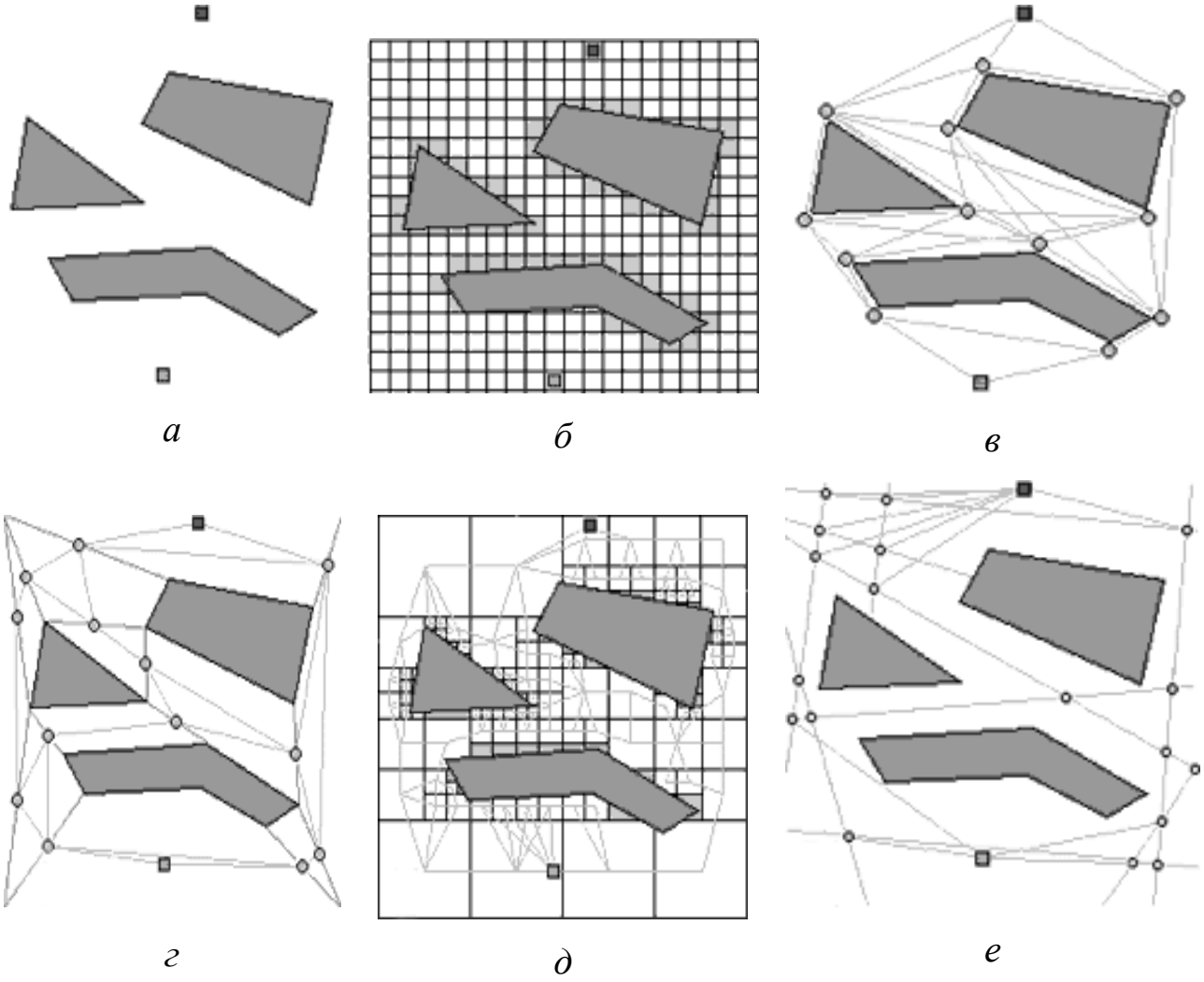


Рис. 9.8. Перешкоди в суцільному просторі та методи перетворення простору пошуку: *а* – перешкоди в суцільному просторі; *б* – зведення до дискретного масиву даних; *в* – задання об'єктів вершинами у зоні прямої видимості; *г* – метод опуклих багатокутників; *д* – метод послідовного розбиття простору; *е* – метод узагальнених циліндрів

Метод послідовного розбиття простору

Зменшуючи крок дискретизації під час перетворення простору пошуку до дискретного вигляду можна отримати будь-яку наперед задану точність подання перешкод, але необмежено зростаючу кількість елементів у просторі пошуку. Збільшуючи ж крок, точність подання перешкод може виявитись незадовільною. Компромісним розв'язанням може стати метод адаптивного розбиття простору. Під час

першого проходу крок дискретизації вибирають досить великим. Надалі «подрібнюються» тільки ті елементи, які частково містять перешкоди. Результат такого адаптивного розбиття показано на рис. 9.8, *д*.

Метод узагальнених циліндрів

Вільні простори між сусідніми перешкодами розглядають як циліндри, форма яких змінюється вздовж їх осей. Такі осі будують для кожної пари сусідніх перешкод і меж простору пошуку й розглядають як можливі елементи траєкторій руху.

Всі описані алгоритми дискретизації суцільного простору мають певні переваги і недоліки, що визначають сфери їх застосування. У деяких випадках конкретний вид алгоритму побудови зв’язаних графів повністю визначається поставленим завданням. Наприклад, для прокладення оптимальної траєкторії руху вулицями міста, кожне перехрестя можна представити як вузол відповідного графу, а зв’язуючим елементом для такого графу будуть вулиці.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. *Вороновский Г. К.* Генетические алгоритмы, искусственные нейронные сети и проблемы виртуальной реальности [Текст] / Г. К. Вороновский, К. В. Махотило, С. Н. Петрашев, С. А. Сергеев. – Харьков: Основа, 1997. – 112 с. – Библиогр.: с. 78–81. – 800 экз. – ISBN 5–7768–0293–8.

2. *Уськов А. А*. Интеллектуальные технологии управления: искусственные нейронные сети и нечеткая логика [Текст] / А. А. Уськов, А. В. Кузьмин. – М.: Горячая линия – Телеком, 2004. – 143 с. – Библиогр.: с. 124–141. – 1000 экз. – ISBN 5–93517–181–3.

3. *Рутковская Д.* Нейронные сети, генетические алгоритмы и нечеткие системы [Текст] / Д. Рутковская, М. Пилиньский, Л. Рутковский. – М.: Горячая линия – Телеком, 2004. – 452 с. – Библиогр.: с. 379–380. – 1500 экз. – ISBN 5–93517–103–1.

4. *P. Wasserman.* Neural computing: theory and practice / Van Nostrand Reinhold, New-York, 1989.

5. *J. S. Albus and A.M. Meystel.* Engineering of Mind: An Introduction to the Science of Intelligent Systems / Wiley, New York, 2001.

6. *J. S. Albus and A. M. Meystel.* Intelligent Systems: Architecture, Design, and Control / Wiley, New York, 2002.

7. *A. P. Engelbrecht.* Computational Intelligence: An Introduction / Wiley, Chichester, U.K., 2002.

8. *R. Babuska.* Fuzzy Modeling for Control / Kluwer Academic Publishers, Boston, 1998.

9. *L. Fausett.* Fundamentals of Neural Networks: Architectures, Algorithms and Applications / Prentice Hall, Englewood Cliffs, NJ, 1997.

10. *S. Haykin.* Neural Networks: A Comprehensive Foundation / Macmillan, New York, 1994.

11. *C. R. Reeves and J.W. Rowe.* Genetic Algorithms Principles and Perspectives: A Guide to GA Theory / Kluwer, Boston, 1997.

12. *R. A. Aleev and R. R. Aleev.* Soft Computing and its Applications / World Scientific, Singapore, 2001.

13. *L. A. Zadeh.* Fuzzy sets / Information and Control, 8:338-353, 1965.

14. *H. Bandemer and S. Gottwald.* Fuzzy Sets, Fuzzy Logic Fuzzy Methods with Applications / John Wiley & Sons, Chichester, 1995.

15. *R. J. Schalkoff.* Artificial Neural Networks / McGraw-Hill, New York, 2002.

16. *A. B. Badiru and J. Y. Cheung.* Fuzzy Engineering Expert Systems with Neural Network Applications / John Wiley, New York, NY, 2002.

ДЛЯ НОТАТОК

ДЛЯ НОТАТОК

Навчальне видання

Апостолюк Владислав Олександрович
Апостолюк Олександр Семенович

Інтелектуальні системи керування

Конспект лекцій

Редактор *Т. М. Дудар*
Комп'ютерна верстка *Т. Є. Клименко*

Темплан 2007 р., поз. 2-011

Підп. до друку 12.08.2008. Формат 60×84$^{1}/_{16}$. Папір офс. Гарнітура Times.
Спосіб друку – ризографія. Ум. друк. арк. 5,11. Обл.-вид. арк. 8,50. Зам. № . Наклад 50 пр.

НТУУ «КПІ» ВПІ ВПК «Політехніка»
Свідоцтво ДК № 1665 від 28.01.2004 р.
03056, Київ, вул. Політехнічна, 14, корп. 15
тел./факс (044) 241-68-78