МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

ТЕХНОЛОГІЇ ШТУЧНОГО ІНТЕЛЕКТУ

ДОСЛІДЖЕННЯ МЕТОДІВ ПРИЙНЯТТЯ РІШЕНЬ В СИСТЕМАХ УПРАВЛІННЯ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання лабораторних робіт для студентів спеціальності „Автоматизоване управління технологічними процесами”

Рекомендовано Методичною радою НТУУ „КПІ”

Київ

НТУУ “КПІ”

2011

Технології штучного інтелекту: Дослідження методів прийняття рішень в системах управління: Методичні вказівки до викон. лабор. робіт для студ. спец. „Автоматизоване управління технологічними процесами” / Уклад.: Д.О. Ковалюк, Л.Д. Ярощук. – К.: НТУУ ”КПІ“, 2011. – 47 с.

Гриф надано Методичною радою НТУУ „КПІ”

(Протокол №10 від 16.06.2011 р.)

Навчальне видання

ДОСЛІДЖЕННЯ МЕТОДІВ ПРИЙНЯТТЯ РІШЕНЬ В СИСТЕМАХ УПРАВЛІННЯ

Методичні вказівки до виконання лабораторних робіт для студентів спеціальності „Автоматизоване управління технологічними процесами”

Укладачі: Ковалюк Дмитро Олександрович, канд. техн. наук, ст. викладач
Ярощук Людмила Дем’янівна, канд. техн. наук, доц.

Відповідальний редактор А.І. Жученко, докт. техн.наук, проф.

Рецензент Д.Е. Сідоров, канд. техн.наук, доц.

Авторська редакція

ВСТУП

Дисципліна “Технології штучного інтелекту” призначена для отримання студентами знань у галузі автоматизації технологічних процесів. Задачі прийняття рішень виникають тут як при проектуванні систем керування та розрахунку їх складових, так і при оптимізації технологічних режимів протікання процесів.

Прийняття рішень розглядається в двох напрямках – це задачі з формалізованими та суб’єктивними моделями. До першої групи включені графові моделі та алгоритми на мережах. Це пояснюється тим, що в багатьох задачах часто виникає необхідність наглядного представлення відношень між об’єктами. Орієнтовані та неорієнтовані графи – природна модель для таких відношень.

Друга група задач охоплює прийняття рішення в умовах невизначеності та багатокритеріальні задачі прийняття рішень. Дана частина дозволить майбутнім фахівцям приймати рішення у випадках відсутності всієї інформації про параметри технологічного процесу.

Метою проведення лабораторних робіт є отримання знань про алгоритми прийняття рішень на графових моделях, за умови відсутності всієї інформації та при багатьох критеріях. Дані ситуації є поширеними в системах керування технологічними процесами.

Тематика лабораторних робіт дозволяє для їх виконання застосовувати різні інструментарії, починаючи від математичних пакетів *MathCAD* та *MATLAB* до мов програмування високого рівня в поєднанні з інтегрованими середовищами для розробки візуальних інтерфейсів.

Проведення лабораторних робіт передбачає виконання досліджень методів прийняття рішень в залежності від типу задачі, структури моделі та її складності.

ЛАБОРАТОРНА РОБОТА №1

ДОСЛІДЖЕННЯ ТЕРМІНУ ЕКСПЛУАТАЦІЇ ЕЛЕМЕНТІВ СИСТЕМИ КЕРУВАННЯ З ВИКОРИСТАННЯМ АЛГОРИТМУ ПОШУКУ НАЙКОРОТШОГО ШЛЯХУ МІЖ ВЕРШИНАМИ ГРАФУ

Мета роботи – ознайомитися з типовою задачею пошуку найкоротшого шляху між вершинами графу. Дослідити алгоритм Дейкстри на прикладі задачі визначення терміну експлуатації елементів системи керування.

Постановка задачі

Комп'ютерна система керування буде модернізована на протязі наступних 6 років (2011–2016). Кожний мікроконтроллер цієї системи повинен відпрацювати не менше 2 і не більше 4 років. В таблиці 1.1. наведена вартість заміни мікроконтроллерів в залежності від року купівлі та терміну експлуатації.

Таблиця 1.1. Вартість заміни залежно від терміну експлуатації

	Вартість заміни обладнання залежно від терміну експлуатації		
Рік	2	3	4
2011	2000	3400	7800
2012	2200	3600	7000
2013	2400	3800	–
2014	2600	–	–

Необхідно знайти такий термін експлуатації елементів системи керування (мікроконтроллерів), щоб сума витрачених коштів була мінімальною.

Основні теоретичні відомості

Поставлену задачу можна сформулювати як мережеву (рис. 1.1) з шістьма вузлами 1–6, що відповідають 2011–2016 рокам.

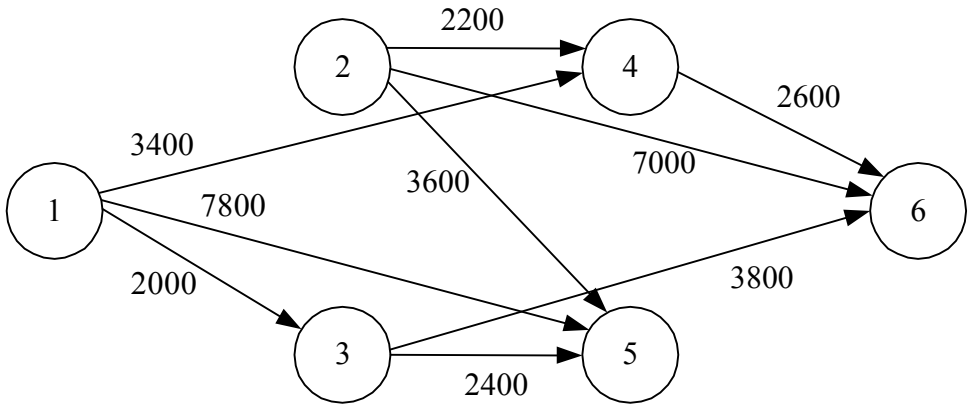

Рис. 1.1. Графова модель задачі розрахунку терміну експлуатації

З вузла 1 (2011 рік) дуги йдуть до вузлів 3,4 і 5, оскільки мікроконтроллери можуть експлуатуватися не менше двох і не більше ніж 4 роки (2011+2, 2011+3, 2011+4). Дуги з інших вузлів інтерпретуються аналогічно. Значення дуг дорівнюють вартостям заміни. Розв’язок задачі зводиться до пошуку найкоротшого шляху між вершинами графу 1 (джерело) та 6 (кінцева вершина).

Алгоритм розв’язання задачі пошуку найкоротшого шляху

На сьогодні найбільш ефективним алгоритмом для розв’язання задачі пошуку найкоротшого шляху вважається алгоритм Дейкстри та його модифікації. Основним припущенням для застосування цього алгоритму є відсутність дуг від’ємної довжини у вихідному графі.

Ідея алгоритму Дейкстри

Алгоритм будує множину S вершин, для яких найкоротші шляхи від джерела вже відомі. На кожному кроці до множини S вершин додається та із залишених вершин, відстань до якої від джерела менша, ніж до решти вершин, що залишилися.

Формалізація алгоритму Дейкстри

Крок 1. Ініціалізація. Перед початком виконання алгоритму всі вершини і дуги не розфарбовані. Кожній вершині в ході виконання алгоритму присвоюється число (мітка) $d(x)$, що дорівнює довжині найкоротшого шляху з початкової вершини s в x, тобто включає лише зафарбовані вершини.

Позначити $d(s)=0$ та $d(x)=\infty$, для всіх x відмінних від s. Зафарбувати вершину s, і присвоїти $y=s$ (y – остання із зафарбованих вершин).

Крок 2. Включення в шлях нової вершини. Для кожної із не зафарбованих вершин x наступним чином перерахувати величину $d(x)$:

$$d(x)=\min\{d(x),\, d(y)+a(y,x)\} \qquad (1.1)$$

Якщо $d(x)=\infty$ для всіх незафарбованих вершин x, закінчити процедуру алгоритму – в вихідному графі відсутні шляхи з вершини s в незафарбовані вершини. В протилежному випадку зафарбувати вершину x, для якої величина $d(x)$ є найменшою. Крім того, зафарбувати дугу, що веде до вибраної на даному кроці вершини x. Позначити $y=x$.

Крок 3. Перевірка умови закінчення роботи алгоритму. Якщо $y=t$, (де t – кінцева вершина) закінчити процедуру – найкоротший шлях з вершини s в t знайдено. В протилежному випадку виконується крок 2.

Порядок виконання роботи

1. Задати вхідні дані задачі розрахунку термінів експлуатації елементів системи керування згідно варіанту.

2. Подати модель заміни елементів системи керування у вигляді орієнтованого графу.

3. Представити графову модель структурою даних, придатною для застосування на ЕОМ та обґрунтувати спосіб представлення.

4. Скласти блок-схему алгоритму пошуку найкоротшого шляху.

5. Виконати програмну реалізації алгоритму.

6. Знайти розв’язок вихідної задачі.

7. Дослідити роботу алгоритму у випадку неорієнтованого графа. Для цього розрахувати оптимальний маршрут між заданими вершинами в прямому та зворотному напрямку.

8. Змінити структуру вихідного графу з таким чином, щоб розірвати отриманий найкоротший маршрут. Промоделювати ситуацію відсутності шляху між двома вершинами.

Додаткове завдання.

1. Розв’язати вихідну задачу, використовуючи симплекс метод. Порівняти ефективність розв’язання.

Зміст звіту

1. Назва, мета, порядок виконання роботи.

2. Математична модель задачі у вигляді графа та структури даних для використання на ЕОМ.

3. Блок-схема алгоритму розв’язання задачі.

4. Лістинг програми з результатами виконання.

5. Контрольний тест, що містить набір вхідних даних і отримані в ході виконання програми результати.

6. Висновки за результатами досліджень.

Контрольні запитання

1. Визначення орієнтованого та неорієнтованого графу.
2. Приклади задачі пошуку найкоротшого шляху в галузі автоматизації технологічних процесів.
3. Основні припущення використання алгоритму Дейкстри.
4. Способи представлення графу, їх переваги та недоліки.
5. Робочий крок алгоритму пошуку найкоротшого шляху. Формула визначення наступної вершини.
6. Способи виведення оптимального маршруту на екран.

Приклад виконання роботи.

```
% Dijkstra's algorithm
n_w = 10^10; %no way between vertex

          %1   2   3   4   5   6   7
Graph = [0.0 0.8 0.3 0.6 n_w n_w n_w  %1
         n_w 0.0 n_w 0.9 0.5 n_w n_w  %2
         n_w n_w 0.0 n_w n_w 0.9 n_w  %3
         n_w n_w 0.8 0.0 0.7 0.6 n_w  %4
         n_w n_w n_w n_w 0.0 0.5 0.8  %5
         n_w n_w n_w n_w 0.0 0.0 0.9  %6
         n_w n_w n_w n_w n_w n_w 0.0  %7
        ];

source = 1;
target = 7;

infinity = 10^10;
undefined = 0;

for vertex = 1 : length(Graph)           % Initializations
    dist(vertex) = infinity;             % Unknown distance function from
source to v
    previous(vertex) = 0;                % Previous node in optimal path
from source
    not_marked_vertex(vertex) = vertex;
end;
```

```
dist(source) = 0;                          % Distance from source to source
last_marked_vertex = source;
not_marked_vertex(source) = [];
way_not_found = false;


while (last_marked_vertex ~= target)
    %get neibers
    min = 10^10;

    for i = 1 : length(not_marked_vertex)
        vertex = not_marked_vertex(i);
        next_marked_vertex = -1;
        distance_from_sorce = dist(vertex);
        distance_from_last_marked = Graph(last_marked_vertex, vertex);
        distance_to_last_marked = dist(last_marked_vertex);
        distance_alt = distance_to_last_marked + distance_from_last_marked;

        % one of the way is not infinity
        if(distance_alt < n_w) || (distance_from_sorce < n_w)
            if distance_from_sorce > distance_alt
                dist(vertex) = distance_alt;
            end;

            if dist(vertex) < min
                min = dist(vertex);
                next_marked_vertex = vertex;
            end;
        end;
    end;
    if next_marked_vertex == -1
        way_not_found = true;
        break;
    else
       % marked next the nearest vertex
       last_marked_vertex = new_marked_vertex;
       index = find(not_marked_vertex == last_marked_vertex);
       not_marked_vertex(index) = [];
    end;
end;

if(way_not_found)
%   'way not found'
else
%   print the way from source to target and its distance
end
```

ЛАБОРАТОРНА РОБОТА №2

ПЕРЕДАВАННЯ ІНФОРМАЦІЇ МІЖ ЕЛЕМЕНТАМИ СИСТЕМИ КЕРУВАННЯ З ВИКОРИСТАННЯМ АЛГОРИТМУ ПОШУКУ ВСІХ НАЙКОРОТШИХ ШЛЯХІВ

Мета роботи – ознайомитися з типовою задачею пошуку всіх найкоротших шляхів між вершинами графу. Дослідити алгоритм Флойда-Уоршелла на прикладі задачі передавання інформації між елементами системи керування.

Постановка задачі

Існує розподілена система керування, що містить шість мікроконролерів, пов’язаних так, як показано на рисунку 2.1.

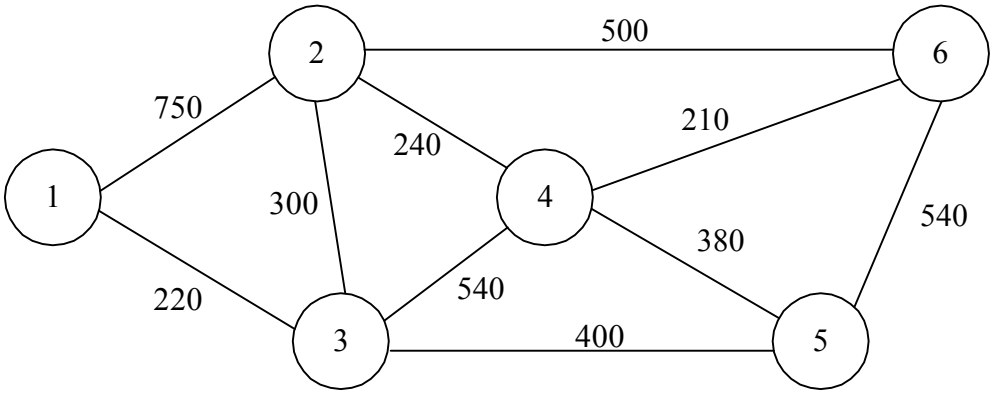

Рис. 2.1. Розподілена система керування

Задані також відстані між контролерами. Визначити найбільш ефективні маршрути пересилання повідомлень між довільними двома контролерами.

Основні теоретичні відомості

В попередній лабораторній роботі була розглянута задача знаходження на графі найкоротшого шляху з деякої виділеної вершини до іншої вершини. Задача пошуку всіх найкоротших шляхів передбачає знаходження найкоротшого шляху між кожною парою вершин.

Звичайно ця більш загальна задача може бути розв'язана шляхом багатократного застосування алгоритму Дейкстри з послідовним вибором кожної вершини графу в якості початкової вершини s. Проте на сьогодні розроблено більш ефективні процедури ніж багатократне повторення алгоритму Дейкстри. Одним з таких алгоритмів є алгоритм Флойда-Уоршелла.

Даний алгоритм допускає наявність дуг від'ємної довжини, але не допускає наявність контурів від'ємної довжини.

Алгоритм Флойда-Уоршелла

Дано орієнтований граф $G=(V,E)$, кожній дузі (x,y) якого поставлена у відповідність невід'ємна вартість $c(x,y)$. Задача полягає у знаходженні для кожної впорядкованої пари вершин $\{x,y\}$ довільного шляху від вершини x до вершини y, довжина якого мінімальна серед всіх можливих шляхів від x до y.

Ідея алгоритму

Припустимо, що вершини графа послідовно пронумеровані від 1 до n. Алгоритм використовує матрицю A розміру $n\times n$, в якій обчислюються довжини найкоротших шляхів. На початку елемент (i,j) дорівнює відстані

d_{ij} від вузла i до вузла j, якщо існує дуга (i, j) і дорівнює нескінченності в іншому випадку. Кожен діагональний елемент матриці A дорівняю нулю.

Над матрицею A виконується n ітерацій. Після k-ї ітерації елемент $A[i, j]$ містить значення найкоротшого шляху з вершини i в вершину j, що не проходить через вершини з номером більшим k. Іншими словами, між кінцевими вершинами i та j можуть знаходитися лише вершини, номера яких менші або дорівнюють k.

На k-й ітерації для обчислення матриці A використовується формула:

$$A_k[i, j] = \min\left\{A_{k-1}[i, j], A_{k-1}[i, k] + A_{k-1}[k, j]\right\}, \qquad (2.1)$$

де індекс k - означає значення матриці після k-ї ітерації.

Формула (2.1) інтерпретується наступним чином. Нехай є три вузли i, j і k к та задані відстані між ними (рис. 2.2).

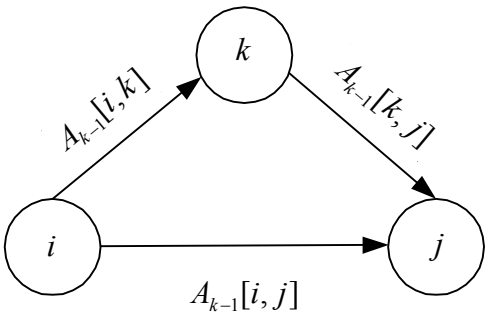

Рис.2.2. Трикутний оператор Флойда

Для обчислення $A_k[i, j]$ виконується порівняння величини $A_{k-1}[i, j]$ (тобто вартість шляху від вершини i до вершини j без участі вершини k) з величиною $A_{k-1}[i, k] + A_{k-1}[k, j]$ (тобто вартість шляху від вершини i до вершини k плюс вартість шляху від вершини k до вершини j). Якщо шлях через проміжну вершину k дешевше, ніж $A_{k-1}[i, j]$, то величина $A_k[i, j]$ змінюється.

Така заміна (далі її умовно називатимемо трикутним оператором) виконується систематично в процесі виконання алгоритму Флойда-Уоршелла.

Формалізація алгоритму Флойда-Уоршелла

Етап 0. Визначаємо початкову матрицю відстаней D_0 і матрицю послідовності вузлів S_0. Діагональні елементи обох матриць позначаються знаком "–", що показує, що ці елементи в обчисленнях участь не беруть. Вважаємо $k = 1$.

D_0		1	2	…	j	…	n
	1	—	d_{12}	…	d_{1j}	…	d_{1n}
	2	d_{21}	—	…	d_{2j}	…	d_{2n}
	…	…	…	…	…	…	…
	i	d_{i1}	d_{i2}	…	d_{ij}	…	d_{in}
	…	...	…	…	...	…	...
	N	d_{n1}	d_{n2}	…	d_{nj}	…	—

		1	2	…	j	…	n
S_0	1		2	...	j	...	п
	2	1	—	...	j	...	п
	…	...	...	...	...	...	...
	i	1	2	...	j	...	п
	…	...	...	...	...	...	...
	N	1	2		j	...	—

Основний етап к. Задаємо рядок k і стовпець k як провідний рядок і провідний стовпець. Розглядаємо можливість використання трикутного оператора до всіх елементів d_{ij} матриці. Якщо виконується нерівність

$$d_{ik} + d_{kj} < d_{ij}$$

то робимо наступне:

- створюємо матрицю D_k шляхом заміни в матриці D_{k-1} елемента d_{ij} сумою $d_{ik} + d_{kj}$;

- створюємо матрицю S_k, змінюючи в матриці S_{k-1} елемент s_{ij} на k. Вважаємо $k = k+1$ і повторюємо етап k.

Після реалізації n етапів алгоритму визначення по матрицях D_n і S_n найкоротшої відстані між вузлами i та j виконується по наступних правилах.

1. Відстань між вузлами *i* та j дорівнює елементу d_{ij} в матриці D_n.
2. Проміжні вузли шляху від вузла i до вузла j визначаються по матриці S_n. Хай $s_{ij} = k$, тоді маємо шлях $i - k - j$. Якщо далі $s_{ik} = k$ і $s_{kj} = j$, тоді вважаємо, що весь шлях визначений, оскільки знайдені всі проміжні вузли. Інакше повторюємо описану процедуру для шляхів від вузла i до вузла k і від вузла k до вузла j.

Порядок виконання роботи

1. Задати вхідні дані задачі передавання інформації між елементами системи керування згідно варіанту.
2. Подати модель передавання інформації в системі керування у вигляді орієнтованого графу.
3. Представити графову модель за допомогою відповідної структури даних та обґрунтувати спосіб представлення.
4. Скласти блок-схему алгоритму пошуку найкоротших шляхів між усіма вершинами графу.
5. Виконати програмну реалізації алгоритму.
6. Знайти розв’язок вихідної задачі.
7. Дослідити залежність між кількістю вершин графу та кількістю ітерацій для побудови всіх найкоротших маршрутів – збільшувати кількість верши до 30. Представити дану залежність графічно.

8. Дослідити роботу алгоритму за наявності у графі дуг від'ємної довжини.

Додаткове завдання.

1. Розв'язати вихідну задачу, використовуючи алгоритм Дейкстри для всіх пар вершин графа. Порівняти результати з алгоритмом Флойда.

Зміст звіту

1. Назва, мета, порядок виконання роботи.
2. Математична модель задачі у вигляді графа та структури даних для використання на ЕОМ.
3. Блок-схема алгоритму розв'язання задачі.
4. Лістинг програми з результатами виконання.
5. Контрольний тест, що містить набір вхідних даних і отримані в ході виконання програми результати.
6. Висновки за результатами досліджень.

Контрольні запитання

1. Приклади задачі пошуку найкоротшого шляху в галузі автоматизації технологічних процесів.
2. Який спосіб представлення графа є найбільш ефективним.
3. Що характеризують значення дуг в графі.
4. В чому полягає відмінність у розв'язку задачі на орієнтованому та неорієнтованому графі.
5. Основні припущення використання алгоритму Флойда-Уоршелла.
6. Структури збереження даних в алгоритмі Флойда-Уоршелла.
7. Структурна схема трикутного оператора.
8. Підпрограма виведення найкоротшого маршруту між вершинами.

ЛАБОРАТОРНА РОБОТА №3

ДОСЛІДЖЕННЯ ОПТИМАЛЬНОГО МАРШРУТУ ТРАНСПОРТУВАННЯ НАФТИ З ВИКОРИСТАННЯМ АЛГОРИТМУ ПОШУКУ МАКСИМАЛЬНОГО ПОТОКУ

Мета роботи – ознайомитися з типовою задачею пошуку максимального потоку між вершинами графу. Дослідити алгоритмом Форда-Фалкерсона для задачі знаходження оптимального маршруту транспортування нафти.

Постановка задачі

Задана мережа трубопроводів для транспортування сирої нафти від бурових свердловин до нафтопереробних заводів (рис. 3.1).

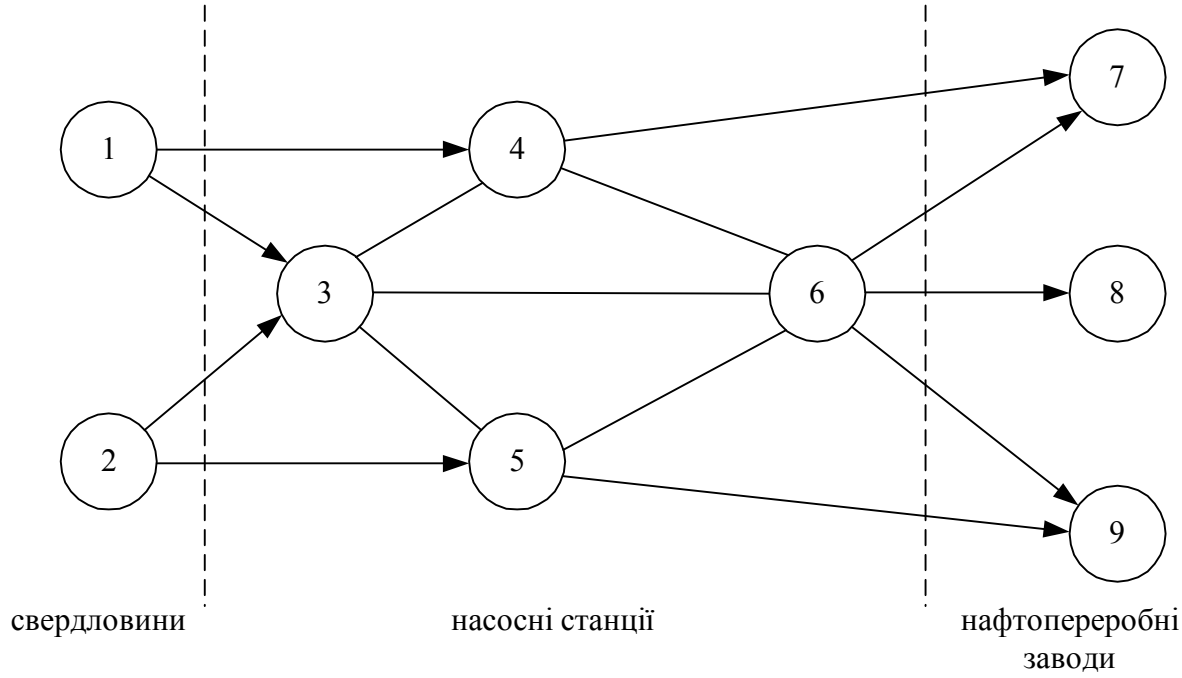

Рис. 3.1. Мережа трубопроводів транспортування нафти

Кожен сегмент трубопроводу має свою пропускну спроможність. Сегменти можуть бути як однонапрвлені так і двухнапрвленні. Необхідно визначити максимальну пропускну спроможність (тобто максимальний потік) між свердловинами та нафтопереробними заводами.

Основні теоретичні відомості

Потік – визначає спосіб пересилання деяких об'єктів з одного пункту в інший. Відносно графів, потік задає спосіб пересилки деяких об'єктів з однієї вершини графа в іншу по його дугам (переміщення по дузі здійснюється в заданому на ній напрямку).

Вершина з якої починається переміщення називається джерелом і зазвичай позначається s. Вершина, в якій закінчується переміщення об'єктів, називається стоком і зазвичай позначається t. Об'єкти, що переміщуються або «протікають» з джерела в сток називаються одиницями потоку або просто одиницями.

Якщо кількість одиниць потоку, що може проходити по дузі обмежена, то це означає, що дуга має обмежену пропускну спроможність.

Поставлена задача транспортування нафти є типовою задачею про максимальний потік і полягає у пошуку способу пересилання максимальної кількості одиниць потоку з джерела в сток за умови відсутності перевищення пропускних спроможностей вихідного графу.

Будемо через $f(x, y)$ позначати кількість одиниць потоку, що проходять по дузі (x, y). Для довільного потоку з s в t кількість одиниць, що виходять з довільної вершини x (при цьому тільки $x \neq s$, $x \neq t$) повинно дорівнювати кількості одиниць, що входять в дану вершину. Тобто:

$$\sum_{y \in X} f(x, y) - \sum_{y \in X} f(y, x) = 0 \ \ (x \neq s, \ x \neq t), \qquad (3.1)$$

де X – множина всіх вершин графу

Крім того, кількість одиниць потоку, що проходять по кожній дузі (x, y) не повинно перевищувати пропускної спроможності цієї дуги.

$$0 \le f(x, y) \le c(x, y) \;\; (x, y) \in A, \tag{3.2}$$

де А – множина всіх дуг графу.

Нарешті, сумарне число одиниць потоку, що виходить з джерела, повинно дорівнювати сумарній кількості одиниць потоку, що входять в сток. Якщо вказане число позначити через v, то ці умови можна записати наступним чином:

$$\sum_{y \in X} f(x, y) - \sum_{y \in X} f(y, s) = v \tag{3.3}$$

$$\sum_{y \in X} f(y, t) - \sum_{y \in X} f(t, y) = v \tag{3.4}$$

Будь-який потік з s в t повинен задовольняти всім чотирьом умовам (3.1)-(3.4). І навпаки, якщо може бути знайдений набір величин $f(x, y)$ при $(x, y) \in A$, для якого виконуються ці чотири умови, то такий набір представляє собою потік з джерела s в сток t.

Тепер можна сказати, що задача про максимальний потік полягає в пошуку потоку, що задовольняє умовам (3.1-3.4), для якого величина v максимальна.

Ідея алгоритму Форда-Фалкерсона

Очевидно, що задача максимізації v при обмеженнях (3.1-3.4) є задачею лінійного програмування, і відповідно, може бути розв’язана з використанням симплекс-методу, спеціально призначеного для розв’язання задач лінійного програмування. Однак існує більш елегантний і природній підхід до розв’язання даної задачі. Це алгоритм пошуку

максимального потоку, розроблений Фордом і Фалкерсоном. Розглянемо основну ідею даного алгоритму.

Припустимо, що є граф, в якому певна кількість одиниць потоку проходить від джерела до стоку і для кожної одиниці потоку відомий маршрут руху.

Позначимо потік в дузі (x, y) через $f(x, y)$. Максимальну пропускну спроможність позначимо $c(x, y)$ Очевидно, що $0 \le f(x, y) \le c(x, y)$.

Дуги графа можна віднести до трьох різних категорій:

1) дуги, в яких потік не може ні збільшуватися ні зменшуватися (множина таких дуг позначається через N)
2) дуги, в яких потік може збільшуватися (множина таких дуг позначається через I)
3) дуги, в яких потік може зменшуватися (множина таких дуг позначається через R)

Наприклад, дуги, що мають нульову пропускну спроможність, повинні належати множині N. Дуги, в яких потік менше пропускної спроможності повинні належати множині I. Дуги по яким проходить деякий потік, повинні належати множині R. Цілком можливо, що деякі дуги належать як множині I так і множині R.

Позначимо через $i(x, y)$ максимальну величину, на яку може бути збільшений потік в дузі (x, y). Відповідно позначимо через $r(x, y)$ максимальну величину, на яку може бути зменшений потік в дузі (x, y). Очевидно, що $i(x, y) = c(x, y) - f(x, y)$ і $r(x, y) = f(x, y)$.

Припустимо, що ми хочемо надіслати додаткову кількість одиниць потоку з джерела в сток. Існує декілька способів виконання даної задачі.

Перший спосіб може бути реалізований, якби ми знайшли шлях P з s в t, який би складався лише із дуг множини I.

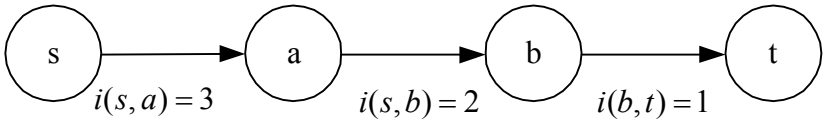

Рис. 3.2. Ланцюг, що збільшує потік і містить лише дуги множини I

Оскільки $i(x,y)$ представляє собою максимально можливе збільшення потоку в дузі (x,y), то величина додаткового потоку з s в t по шляху P складе:

$$\min_{(x,y)\in P}\{i(x,y)\}$$

Для рис 3.2. по шляху P можна переслати з s в t максимум одну одиницю потоку

$$\min\{i(s,a),i(a,b),i(d,t)\}=\min\{3,2,1\}=1$$

Другий спосіб збільшення потоку може бути реалізований, якби ми знайшли шлях P з t в s, який цілком складається із дуг множини R, які зменшують потік у дузі. При цьому можна було б зменшити потік в кожній дузі (x,y), що призвело б до зменшення потоку з вершини t в s, і відповідно до збільшення чистого потоку з s в t.

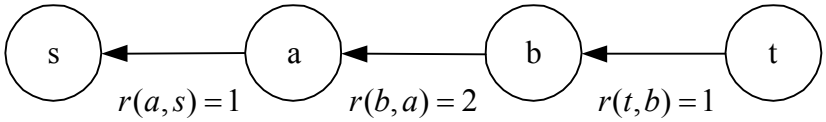

Рис. 3.3. Ланцюг, що збільшує потік і містить лише дуги множини R

Очевидно, що максимальна величина, на яку можна зменшити потік з t в s визначається як

$$\min_{(x,y)\in P}\{r(x,y)\}$$

Для прикладу, показаного на рис 3.3., по шляху P можна переслати назад з

з t в s максимум одну одиницю потоку

$$\min\{r(t,b), r(b,a), r(a,s)\} = \min\{1,2,1\} = 1$$

Третій спосіб збільшення потоку з s в t – це комбінація перших двох способів.

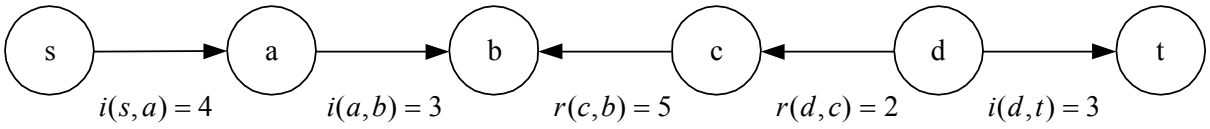

Рис. 3.4. Ланцюг, що збільшує потік і включає прямі та зворотні дуги

Кожен із трьох розглянутих вище ланцюгів з s в t, називається збільшуючим ланцюгом. Таким чином, для пошуку максимального потоку треба визначити, чи є у вихідній мережі збільшууючий ланцюг, що веде з s в t.

Алгоритм пошуку збільшуючого ланцюга.

Крок 1. Визначити склад множин N, I та R. Дуги множини N з подальшого розгляду виключити. Зафарбувати вершину s.

Крок 2. Зафарбовувати дуги та вершини у відповідності з наступними правилами, поки не буде зафарбована вершина t, або зафарбовування нових вершин буде неможливим. Правила зафарбовування вершини y та дуги (x, y) при вже зафарбованій вершині полягають в наступному:

а) якщо $(x, y) \in I$, то зафарбовується вершина y та дуга (x, y).

б) якщо $(y, x) \in R$, то зафарбовується вершина y та дуга (y, x).

в) в протилежному випадку зафарбовування вершини y та дуги (x, y) не виконується.

Формалізація алгоритму Форда-Фалкерсона

Знаючи як знайти ланцюг, що збільшує потік, розглянемо тепер алгоритм Форда-Фалкерсона. Формується початковий потік з з джерела s в сток t і далі виконується пошук збільшуючого ланцюга. Якщо він виявляється успішним, то потік вздовж знайденого ланцюга збільшується до максимального значення. Потім виконується пошук нового збільшеного ланцюга і т.д. Якщо на деякому етапі цієї процедури не вдається знайти збільшуючий ланцюг, виконання алгоритму завершується – поточний потік з s в сток t є максимальним.

Крок 1. Вибрати довільний початковий потік з вершини s (джерело) в вершину t (сток), тобто довільний набір величин $f(x, y)$, що задовольняють умовам (3.1)-(3.4).

Крок 2. Для всіх дуг (x, y) виконати наступне.

Якщо $f(x, y) < c(x, y)$, то визначити $i(x, y) = c(x, y) - f(x, y)$ і вважати дугу (x, y) належною до множини *I.*

Якщо $f(x, y) > 0$, то позначити $r(x, y) = f(x, y)$ і вважати дугу (x, y), такою, що належить множині R.

Крок 3. На множинах I та R, сформованих на попередньому кроці, застосувати до вихідного графа алгоритм пошуку збільшуючого ланцюга. Якщо при цьому збільшуючий потік ланцюг знайти не вдалося, то закінчити процедуру алгоритму: поточний потік є максимальним. В протилежному випадку виконати максимально можливе збільшення потоку вздовж знайденого ланцюга. Потім повернутися до кроку 2.

Порядок виконання роботи

1. Задати вхідні дані задачі транспортування нафти згідно варіанту.
2. Подати модель трубопроводів перекачування нафти у вигляді орієнтованого графу.
3. Представити графову модель за допомогою відповідної структури даних та обґрунтувати спосіб представлення.
4. Скласти блок-схему алгоритму пошуку максимального потоку.
5. Виконати програмну реалізації алгоритму.
6. Знайти розв’язок вихідної задачі.
7. Дослідити процедуру побудови та відобразити шлях входження вершин до маршруту оптимального потоку.

Додаткове завдання.

1. Розв’язати вихідну задачу, з використанням методів лінійного програмування.

Зміст звіту

1. Назва, мета, порядок виконання роботи.
2. Математична модель задачі у вигляді графа та структури даних для використання на ЕОМ.
3. Блок-схема алгоритму розв’язання задачі.
4. Лістинг програми з результатами виконання.
5. Контрольний тест, що містить набір вхідних даних і отримані в ході виконання програми результати.
6. Висновки за результатами досліджень.

Контрольні запитання

1. Дайте визначення потоку, його фізичний зміст в реальних задачах.
2. Приклади задачі пошуку максимального потоку в галузі автоматизації технологічних процесів хімічної промисловості.
3. Ідея алгоритму пошуку максимального потоку.
4. Алгоритм пошуку збільшуючого ланцюга.
5. Які множини дуг виділяють в задачі пошуку збільшуючого ланцюга. Умови переходу дуги з однієї множини до іншої.
6. Математична постановка задачі пошуку максимального потоку.
7. Обґрунтування алгоритму пошуку максимального потоку.

Лабораторна робота №4

ПРОЕКТУВАННЯ АРХІТЕКТУРИ КОМП'ЮТЕРНОЇ МЕРЕЖІ В УМОВАХ НЕВИЗНАЧЕНОСТІ

Мета роботи – ознайомитися з прийняттям рішень в умовах невизначеності. Дослідити критерії прийняття рішень в умовах невизначеності на прикладі задачі вибору архітектури комп'ютерної мережі.

Постановка задачі

Необхідно побудувати локальну комп'ютерну мережу підприємства. Задача полягає у виборі відповідної кількості серверів, що можуть бути включені до складу мережі, за умови відсутності точної інформації про кількість робочих станцій, які будуть обслуговуватись в майбутньому.

Основні теоретичні відомості

Відомо, що постановка задач прийняття рішень та методи їх розв'язання істотно залежать від ступеня невизначеності параметрів технологічного процесу та стану зовнішнього середовища. В зв'язку з цим загальноприйнятою є класифікація задач прийняття рішень, наведена на рис. 4.1.

Задачі першої групи характеризуються наявністю чітко визначеної та формалізованої моделі процесу, що дозволяє для їх розв'язання застосовувати відомі методи оптимізації та математичного програмування.

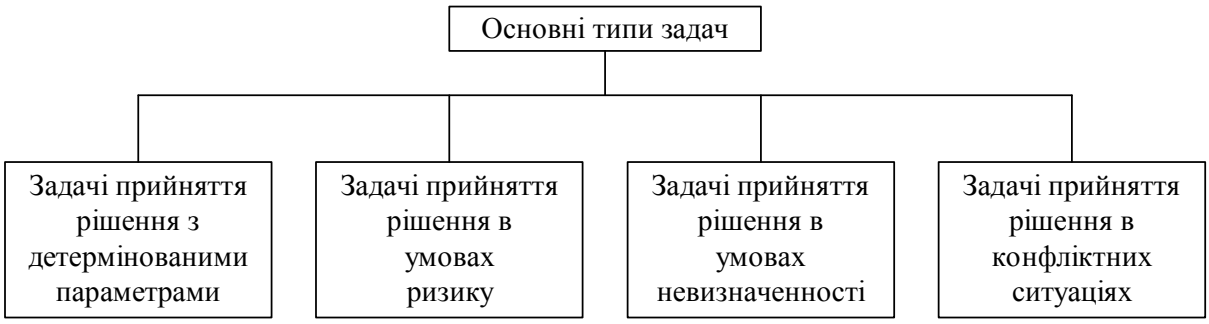

Рис. 4.1. Основні класи задач прийняття рішень залежно від наявної інформації про параметри системи

Задачі другого типу характеризуються тим, що для ряду параметрів не відомі точні значення, а визначені лише діапазони їх зміни і на кожному діапазоні задані щільності розподілу випадкових величин. Необхідно вибрати таке рішення, яке для заданих розподілів ймовірностей забезпечує екстремум показника ефективності. Найбільш відомими з цієї групи є задачі управління запасами, управління марковськими процесами, аналізу і синтезу систем масового обслуговування.

Задачі третьої групи характеризується тим, що для кожного з параметрів системи задані можливі дискретні значення і для них визначено значення показників ефективності, що відповідають кожному з варіантів альтернативних рішень.

До цієї групи задач відноситься задача проектування архітектури комп'ютерної мережі. В даному випадку альтернативні варіанти рішень – це різна кількість серверів, що можуть бути включені до складу мережі $E = \{E_1, E_i, ..., E_n\}$, а в якості значень параметра використовується кількість робочих станції $a = \{a_1, a_2, a_j, ..., a_m\}$. В якості показника ефективності – можуть виступати грошові витрати, швидкодія мережі, тощо.

Таблиця 4.1. Матриця прийняття рішень для задачі проектування комп'ютерної мережі

Альтернативні варіанти рішень	Дискретні значення параметра a					
	a_1	a_2	...	a_j	...	a_m
E_1	α_{11}	α_{12}	...	α_{1j}	...	α_{1m}
E_i	α_{i1}	α_{i2}	...	α_{ij}	...	α_{im}
E_n	α_{n1}	α_{n2}	...	α_{nj}	...	α_{nm}

В більш загальному випадку в задачі прийняття рішень в умовах невизначеності співвідношення між альтернативами та дискретними значеннями параметра системи (наприклад, таке як в табл. 4.1.) називається *платіжною* матрицею. А множина значень параметра, що впливає на ефективність кожної з альтернатив, називається *станами природи*. Таке визначення підкреслює, що точне значення параметра, яке буде на момент прийняття рішення, нам не відоме, і воно встановиться випадковим чином.

Платіжну матрицю в задачі прийняття рішень з m можливими діями та n станах природи можна представити наступним чином.

Таблиця 4.2. Платіжна матриця

	S_1	S_2	…	S_n
a_1	$v(a_1, s_1)$	$v(a_1, s_2)$	…	$v(a_1, s_n)$
a_2	$v(a_2, s_1)$	$v(a_2, s_2)$	…	$v(a_2, s_n)$
…	…	…	…	…
a_m	$v(a_m, s_1)$	$v(a_m, s_2)$	…	$v(a_m, s_n)$

Елемент a_i являє i-те можливе рішення, а елемент s_j – j-й стан природи. Плата (дохід або втрати), пов'язана з рішенням a_i та станом s_j, дорівнює $v(a_i, s_j)$.

В деяких випадках зручно представляти співвідношення між альтернативами і станами природи не у вигляді платіжної матриці, а у вигляді матриці ризиків.

Ризиком $r_{i,j}$ при реалізації альтернативи a_i за умови знаходження природи в стані s_j, називається різниця між виграшем, який би отримав гравець якби знав стан природи s_j, і виграшем який він отримає, застосовуючи альтернативу a_i в умовах відсутності інформації про стан природи.

$$r(a_i, s_j) = \begin{cases} \max\{v(a_k, s_j)\} - v(a_i, s_j), & \textit{якщо } v - \textit{дохід} \\ v(a_i, s_j) - \min\{v(a_k, s_j)\}, & \textit{якщо } v - \textit{втрати} \end{cases}$$

Матриця ризиків дає найбільш наглядну картину невизначеності ситуації, так як з матриці ризиків $R = \|r_{ij}\|$ видно чим ризикує гравець, використовуючи ту чи іншу альтернативу. Інакше, величина ризику – це розмір плати за відсутність інформації про стан природи.

При виборі оптимальної стратегії виходять з міркувань отримання деякого гарантованого виграшу і використовують наступні критеріїв для аналізу ситуації, пов'язаної з прийняттям рішень.

1. Критерій Лапласа.
2. Максимінний (мінімаксний) – критерій Вальда.
3. Критерій Севіджа.
4. Критерій Гурвиця.

Критерії прийняття рішень в умовах невизначеності

Критерій Лапласа опирається на принцип недостатнього обґрунтування, який говорить про те, що оскільки розподіл ймовірностей станів природи $P(s_j)$ невідомий, немає причин вважати їх відмінними. Отже, використовуючи припущення, що імовірність всіх станів природи рівні між собою, тобто:

$$P(s_1) = P(s_1) = ... = P(s_n) = 1/n$$

Якщо при цьому $v(a_i, s_j)$ виражає можливий прибуток, то найкращим рішенням є те, яке забезпечує:

$$\max_{a_i} \left\{ \frac{1}{n} \sum_{j=1}^{n} v\left(a_i, s_j\right) \right\}$$

Якщо величина $v(a_i, s_j)$ являє собою втрати, то оператор «max» заміняється на «min».

Максимінний (мінімаксний) критерій ґрунтується на обережній поведінці особи, яка приймає рішення, і зводиться до вибору найкращої альтернативи із найгірших. Якщо величина $v(a_i, s_j)$ виражає можливий прибуток, то у відповідності з максимінним критерієм в якості оптимального обирається рішення, що забезпечує:

$$\max_{\mathrm{a_i}} \left\{ \min_{s_j} v\left(a_i, s_j\right) \right\}$$

Якщо величина $v(a_i, s_j)$ являє собою втрати, використовується мінімаксний критерій, що визначається наступним відношенням:

$$\min_{\mathrm{a_i}} \left\{ \max_{s_j} v\left(a_i, s_j\right) \right\}$$

Критерій Севіджа послаблює обережність мінімаксного (максимінного) критерію шляхом заміни платіжної матриці (виграшів чи програшів) $v(a_i, s_j)$ матрицею втрат $r(a_i, s_j)$, і визначається наступним чином:

$$\min_{A_j} \max_{S_j} \{r_{ij}\}$$

Критерій Гурвиця. Цей критерій охоплює ряд різних підходів до прийняття рішень – від найбільш оптимістичного до найбільш песимістичного. Нехай $0 \le \alpha \le 1$ та величини $v(a_i, s_j)$ представляють прибуток. Тоді рішенню, обраному за критерієм Гурвиця, відповідає:

$$\max_{a_i} \left\{ \alpha \max_{s_j} v\left(a_i, s_j\right) + \left(1-\alpha\right) \min_{s_j} v\left(a_i, s_j\right) \right\}$$

Параметр α - показник оптимізму. Якщо $\alpha = 0$, критерій Гурвиця стає критерієм Вальда. Якщо $\alpha = 1$, критерій Гурвиця стає занадто оптимістичним, бо розраховує на кращі з найкращих умов. Можна конкретизувати степінь оптимізму (або песимізму) належним чином величини α з інтервалу [0,1]. При відсутності яскраво вираженої схильності до оптимізму чи песимізму вибір $\alpha = 0.5$ є найбільш виправданим. Якщо величини $v(a_i, s_j)$ виражають втрати, то критерій приймає наступний вигляд:

$$\min_{a_i} \left\{ \alpha \min_{s_j} v\left(a_i, s_j\right) + \left(1-\alpha\right) \max_{s_j} v\left(a_i, s_j\right) \right\}$$

Розв'язання задачі проектування комп'ютерної мережі

Нехай для задачі, що розглядається, задана платіжна матриця, яка містить: альтернативи – кількість серверів, стани природи – кількість робочих станцій, показник ефективності – витрати (тис. грн.), пов'язані з вартістю оптимальної кількості серверів, купівлею зайвих серверів,

втратою прибутку у випадку недостатніх потужностей для підключення нових станцій.

Табл. 4.3. Платіжна матриця для задачі вибору архітектури мережі

Кількість серверів	Кількість робочих станцій			
	$s_1 = 15$	$s_2 = 25$	$s_3 = 35$	$s_4 = 50$
$a_1 = 2$	5	11	19	26
$a_2 = 3$	9	8	14	23
$a_3 = 4$	20	17	12	22
$a_4 = 6$	30	21	18	17

Опис ситуації аналізується з точки зору чотирьох розглянутих вище критеріїв.

Критерій Лапласа. При заданих імовірностях $P(s_j) = 1/4$, $j = 1, 2, 3, 4$, очікувані значення витрат для різних можливих рішень розраховуються наступним чином:

$M\{a_1\} = 1/4\ (5 + 11 + 19 + 26) = 15{,}25$

$M\{a_2\} = 1/4\ (9 + 8 + 14 + 23) = 13{,}5 \leftarrow$ Оптимум

$M\{a_3\} = 1/4(20 + 17 + 12 + 22) = 17{,}75$

$M\{a_4\} = 1/4(30 + 21 + 18 + 17) = 21{,}5.$

Мінімаксний критерій: $\min\{26,23,22,30\} = 22$

Критерій Севіджа. Матриця ризиків визначається шляхом віднімання чисел 5, 8, 12 та 17 від елементів стовбців з першого до четвертого відповідно.

Табл. 4.4. Матриця ризиків для задачі вибору архітектури мережі

	s_1	s_2	s_3	s_4	максимум рядків
a_1	0	3	7	9	9
a_2	4	0	2	6	6 мінімакс
a_3	15	9	0	5	16
a_4	25	13	6	0	25

Критерій Гурвиця. Результати розрахунків знаходяться в наступній таблиці.

Табл. 4.5. Критерій Гурвиця для задачі вибору архітектури мережі

Альтернатива	Мінімум рядків	Максимум рядків	α(максимум рядка) + (1-α)(мінімум рядка)	α = 0,5
a_1	5	26	$26-21\alpha$	15,5- опт
a_2	8	23	$23-15\alpha$	15,5- опт
a_3	12	22	$22-10\alpha$	17
a_4	17	30	$30-13\alpha$	23,5

Порядок виконання роботи

1. Сформулювати задачу прийняття рішення в умовах невизначеності станів природи для галузі автоматизованих систем управління.
2. Представити співвідношення між альтернативами і станами системи у вигляді платіжної матриці.
3. Визначити найкращі альтернативи за кожним з критеріїв.
4. Дослідити процес визначення найкращої альтернативи за різних значень α -параметру критерію Гурвиця.

5. Провести дослідження даних критеріїв у випадку, якщо вихідна задача описується матрицею втрат.

6. Ввести нову альтернативу. Дослідити впорядкованість альтернатив за кожним з критеріїв за наявної додаткової інформації.

Зміст звіту

1. Назва, мета, порядок виконання роботи.
2. Словесна постановка задачі та її математичний опис у вигляді платіжної матриці.
3. Розраховані критерії прийняття рішень в умовах невизначеності.
4. Лістинг програми з результатами виконання.
5. Висновки за результатами досліджень.

Контрольні запитання

1. Класи задач прийняття рішень залежно від способу задання наявної інформації про параметри системи.
2. Постановка задачі прийняття рішення в умовах невизначеності.
3. Види невизначеності та способи її опису.
4. Критерії прийняття рішень в умовах невизначеності станів природи.

Приклад виконання роботи

ORIGIN:= 1

платіжна матриця

$$A := \begin{pmatrix} 5 & 11 & 19 & 26 \\ 9 & 8 & 14 & 23 \\ 20 & 17 & 12 & 22 \\ 30 & 21 & 18 & 17 \end{pmatrix}$$

count_states := cols(A) count_states = 4

count_alternatives := rows(A) count_alternatives = 4

Критерій Лапласа

$$M := \text{for } i \in 1..\text{count\_alternatives}$$

$$M_i \leftarrow \frac{1}{\text{count\_states}} \left(\sum_{j=1}^{\text{count\_states}} A_{i,j} \right)$$

$$M = \begin{pmatrix} 15.25 \\ 13.5 \\ 17.75 \\ 21.5 \end{pmatrix}$$

min(M) = 13.5 згідно даного критерію оптимальною є 2-га альтернатива

Мінімаксний критерій

$$M := \text{for } i \in 1..\text{count\_alternatives}$$

$$M_i \leftarrow \max((\text{submatrix}(A, i, i, 1, \text{count\_states})))$$

$$M = \begin{pmatrix} 26 \\ 23 \\ 22 \\ 30 \end{pmatrix}$$

min(M) = 22 згідно даного критерію оптимальною є 3-тя альтернатива

Критерій Гурвиця

$\alpha := 0.5$

показник оптимізму

$$M := \text{for } i \in 1..4$$

$$M_i \leftarrow \alpha \cdot \min((\text{submatrix}(A, i, i, 1, 4))) + (1 - \alpha) \cdot \max((\text{submatrix}(A, i, i, 1, 4)))$$

$$M = \begin{pmatrix} 15.5 \\ 15.5 \\ 17 \\ 23.5 \end{pmatrix}$$

min(M) = 15.5 згідно критерію оптимальною є 1 або 2-га альтернатива

Критерій Севіджа

$A1 := \text{for } i \in 1..4$

$\quad \text{for } j \in 1..4$

$\quad\quad A1_{i,j} \leftarrow A_{i,j} - \min((\text{submatrix}(A, 1, 4, j, j)))$

$$A1 = \begin{pmatrix} 0 & 3 & 7 & 9 \\ 4 & 0 & 2 & 6 \\ 15 & 9 & 0 & 5 \\ 25 & 13 & 6 & 0 \end{pmatrix}$$

$M := \text{for } i \in 1..4$

$\quad M_i \leftarrow \max((\text{submatrix}(A1, i, i, 1, 4)))$

$$M = \begin{pmatrix} 9 \\ 6 \\ 15 \\ 25 \end{pmatrix}$$

$\min(M) = 6$ згідно даного критерію оптимальною є 2-га альтернатива

Лабораторна робота №5

БАГАТОКРИТЕРІАЛЬНЕ ПОРІВНЯННЯ КОНФІГУРАЦІЙ СИСТЕМ КЕРУВАННЯ З ВИКОРИСТАННЯМ МЕТОДУ АНАЛІЗУ ІЄРАРХІЙ

Мета роботи – ознайомитися з особливостями багатокритеріальних задач прийняття рішень, дослідити метод аналізу ієрархій на прикладі задачі порівняння конфігурацій систем керування.

Постановка задачі

Припустимо, що розглядається чотири (A, B, C, D) конкуруючих варіанти архітектури АСУТП, кожен з яких характеризується кількісними показниками вартості, надійності, терміном експлуатації (табл. 5.1.).

Табл. 5.1. Альтернативи вибору АСУТП

Архітектура АСУТП	Вартість (тис. грн.)	Надійність (ймовірність безвідмовної роботи)	Термін експлуатації (роки)
A – фірма Мікрол	300	0.9930	3
B – фірма ОВЕН	400	0.9950	4
C – фірма Siemens	800	0.9990	6
D – фірма Mitsubishi	900	0.9998	6

На основі наявної інформації необхідно вибрати найкращу конфігурацію системи керування.

Основні теоретичні відомості

Аналіз багатьох практичних проблем зумовив виникнення нового класу – багатокритеріальних задач прийняття рішень. За наявності декількох критеріїв задачі вибору найкращого рішення набувають наступних особливостей:

- Задача має новий унікальний характер – немає статистичних даних, які б дозволили обґрунтувати співвідношення між різними критеріями.
- На момент прийняття рішення принципово відсутня інформація, яка б дозволила об’єктивно оцінити наслідки прийнятого рішення

Виділяють дві групи багатокритеріальних задач прийняття рішення.

Задачі *першої групи* можна сформулювати наступним чином.

Дано: група з n альтернатив і N критеріїв, призначених для оцінки альтернатив. Кожна із альтернатив має оцінку по кожному з критеріїв, отриману від експертів або на основі об’єктивних розрахунків.

Необхідно: побудувати розв’язуючі правила на основі переваг ОПР, що дозволяють:

- виділити найкращу альтернативу;
- впорядкувати альтернативи по якості;
- віднести альтернативи до впорядкованих по якості класам рішень.

Задачі *другої групи* можна сформулювати наступним чином.

Дано: група з N критеріїв, призначених для оцінки будь-яких можливих альтернатив; альтернативи задані частково або з’являються після побудови розв’язуючого правила.

Необхідно: побудувати розв’язуючі правила на основі переваг ОПР, що дозволяють:

- впорядкувати по якості всі можливі альтернативи;
- віднести всі можливі альтернативи до одного з декількох (вказаних ОПР) класів рішень.

Очевидно, що поставлена задача вибору системи керування відноситься до першої групи, коли наперед задані всі альтернативи та критерії.

Для розв’язання даного класу задач у випадку невеликої кількості альтернатив використовується *метод аналізу ієрархій* (Analitic Hierarhy Process - AHP)

Алгоритм методу аналізу ієрархій

Метод аналізу ієрархій включає наступні етапи:

1. Структуризація задачі у вигляді ієрархічної структури з декількома рівнями: цілі–критерії–альтернативи.
2. ОПР виконує парні порівняння елементів кожного рівня. Результати порівнянь переводяться в кількісні оцінки з використанням відповідної шкали.
3. Розраховуються коефіцієнти важливості для елементів кожного рівня. При цьому перевіряється узгодженість міркувань ОПР.
4. Розраховується кількісний індикатор якості кожної з альтернатив і визначається найкраща альтернатива.

Застосування методу аналізу ієрархій до поставленої задачі

1. Структуризація задачі.

Розв'язувана задача може бути представлена у вигляді ієрархічної

структури, представленої на рис. 5.1.

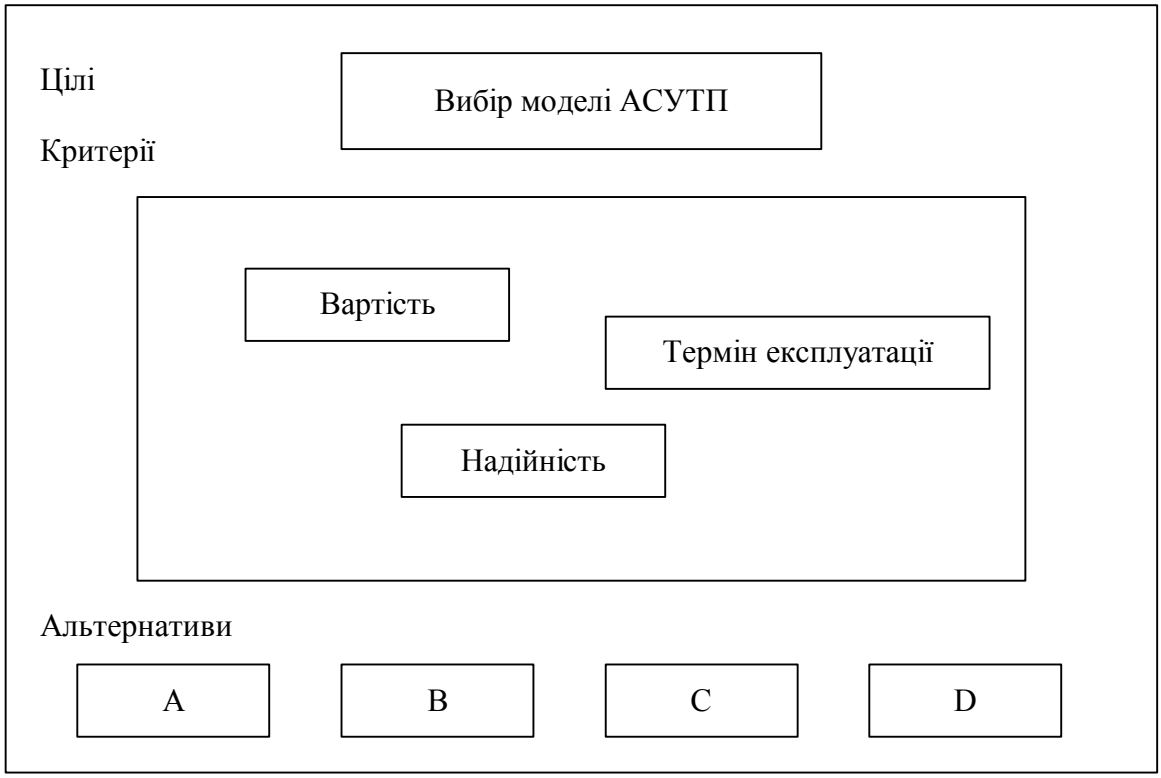

Рис. 5.1. Ієрархічна схема проблеми вибору АСУТП

2. Парні порівняння

При парних порівняннях в розпорядження ОПР надається шкала словесних визначень рівня важливості, причому кожному визначенню ставиться у відповідність число (табл. 5.2).

Табл. 5.2. Шкала порівнянь альтернатив і критеріїв

Рівень важливості	Кількісне значення
Рівна важливість	1
Поміркована перевага	3
Істотна або сильна перевага	5
Значна (велика) перевага	7
Дуже велика перевага	9

При порівнянні елементів, що належать одному рівню ієрархії, ОПР

висловлює свою думку, використовуючи одне з наведених у таблиці 5.2 визначень. У матрицю порівняння заноситься відповідне число. Матриця порівнянь критеріїв вибору АСУТП наведена в табл. 5.3.

Табл. 5.3. Матриця порівнянь для критеріїв

Критерій	C1 вартість	C2 надійність	C3 термін експлуатації	Власний вектор	Вага критерію
C1 вартість	1	5	3	2.466	0.637
C2 надійність	1/5	1	1/3	0.405	0.105
C3 термін експлуатації	1/3	3	1	1	0.258

Матриця відповідає наступним перевагам ОПР: критерій «Вартість» істотно перевершує критерій «Надійність» і помірковано перевершує критерій «Термін експлуатації»; критерій «Термін експлуатації» помірковано перевершує критерій «Надійність».

На нижньому рівні ієрархічної схеми порівнюються задані альтернативи (конкретні АСУТП) за кожним критерієм окремо.

Табл. 5.4. Відносна важливість альтернатив за окремими критеріями

По критерію C1 (Вартість)						
Альтернатива	A	B	C	D	Власний вектор	Вага
A	1	3	7	9	3.708	0.576
B	1/3	1	5	7	1.848	0.287
C	1/7	1/3	1	3	0.615	0.095
D	1/9	1/7	1/3	1	0.27	0.042
По критерію C2 (надійність)						
A	1	1/3	1/5	1/7	0.312	0.054
B	3	1	1/5	1/5	0.589	0.101
C	5	5	1	1/3	1.699	0.293
D	7	5	3	1	3.201	0.552

По критерію С3 (термін експлуатації)						
A	1	1/3	1/3	1/3	0.439	0.088
B	3	1	1/5	1/5	0.589	0.119
C	3	5	1	1	1.968	0.397
D	3	5	1	1	1.968	0.397

3. Розрахунок коефіцієнтів важливості

Таблиці 5.3 і 5.4 дозволяють розрахувати коефіцієнти важливості відповідних елементів ієрархічного рівня. Для цього потрібно обчислити власні вектори матриці, а потім нормувати їх.

Формула для цих обчислень: добувається корінь n-го степеня (n - розмірність матриці порівнянь) з добутку елементів кожного рядка.

4. Визначення найкращої альтернативи

Синтез отриманих коефіцієнтів важливості здійснюється за формулою

$$S_j = \sum_{i=1}^{N} w_i \cdot V_{ij} \qquad (5.1)$$

де S_j – показник якості j–ї альтернативи; w_i – вага i-го критерію, V_{ji} – важливість j-ї альтернативи по i-му критерію.

Для чотирьох АСУТП проведені наступні обчислення:

$S_A = 0.637 \cdot 0.576 + 0.105 \cdot 0.054 + 0.258 \cdot 0.088 = 0.395$

$S_B = 0.637 \cdot 0.287 + 0.105 \cdot 0.101 + 0.258 \cdot 0.119 = 0.224$

$S_A = 0.637 \cdot 0.095 + 0.105 \cdot 0.293 + 0.258 \cdot 0.397 = 0.194$

$S_A = 0.637 \cdot 0.042 + 0.105 \cdot 0.552 + 0.258 \cdot 0.397 = 0.187$

Отже, альтернатива А згідно уподобань ОПР є найкращою.

Порядок виконання роботи

1. Задати значення альтернатив і критеріїв задачі вибору конфігурації системи керування згідно варіанту.
2. Утворити із заданих альтернатив множину Еджвода-Парето.
3. Виконати програмну реалізацію методу аналізу ієрархій.
4. Розв’язати поставлену задачу.
5. Дослідити метод аналізу ієрархій за рахунок зміни ОПР і відповідно матриць парних порівнянь.
6. Виявити недоліки методу шляхом введення ще одного критерію.

Зміст звіту

1. Назва, мета, порядок виконання роботи.
2. Постановка задачі у вигляді таблиці значень альтернатив і критеріїв.
3. Матриці парних порівнянь для альтернатив і критеріїв.
4. Лістинг програмної реалізації методу аналізу ієрархій.
5. Результати розв’язання задачі.
6. Результати розв’язання задачі при зміні ОПР та введені додаткового критерію.

Контрольні запитання

1. Особливості багатокритеріальних задач прийняття рішень.
2. Область застосування методу аналізу ієрархій.
3. Принцип створення матриці парних порівнянь.
4. Переваги та недоліки методу.

Приклад виконання роботи

ORIGIN:= 1

функції для розрахунку власних векторів та їх ваг

$$\text{calculate\_eigen\_vectors}(\text{matrix}) := \left| \begin{array}{l} \text{matrix\_size} \leftarrow \text{rows}(\text{matrix}) \\ \text{for } i \in 1..\text{matrix\_size} \\ \quad \lambda_i \leftarrow \sqrt[\text{matrix\_size}]{\prod_{j=1}^{\text{matrix\_size}} \text{matrix}_{i,j}} \\ \text{return } \lambda \end{array} \right.$$

$$\text{calculate\_weight}(\text{eigen\_vector}) := \left| \begin{array}{l} \text{vector\_size} \leftarrow \text{rows}(\text{eigen\_vector}) \\ \text{for } i \in 1..\text{vector\_size} \\ \quad w_i \leftarrow \dfrac{\text{eigen\_vector}_i}{\sum_{n=1}^{\text{vector\_size}} \text{eigen\_vector}_n} \\ \text{return } w \end{array} \right.$$

матриця парних порівнянь для критеріїв

$$C := \begin{pmatrix} 1 & 5 & 3 \\ \frac{1}{5} & 1 & \frac{1}{3} \\ \frac{1}{3} & 3 & 1 \end{pmatrix}$$

власні вектори порівняння критеріїв та їх ваги

$$\lambda := \text{calculate\_eigen\_vectors}(C) \qquad \lambda = \begin{pmatrix} 2.466 \\ 0.405 \\ 1 \end{pmatrix}$$

$$c\_w := \text{calculate\_weight}(\lambda) \qquad c\_w = \begin{pmatrix} 0.637 \\ 0.105 \\ 0.258 \end{pmatrix}$$

матриці парних порівнянь альтернатив за кожним з критеріїв

$$A1 := \begin{pmatrix} 1 & 3 & 7 & 9 \\ \frac{1}{3} & 1 & 5 & 7 \\ \frac{1}{7} & \frac{1}{3} & 1 & 3 \\ \frac{1}{9} & \frac{1}{7} & \frac{1}{3} & 1 \end{pmatrix} \qquad A2 := \begin{pmatrix} 1 & \frac{1}{3} & \frac{1}{5} & \frac{1}{7} \\ 3 & 1 & \frac{1}{5} & \frac{1}{5} \\ 5 & 5 & 1 & \frac{1}{3} \\ 7 & 5 & 3 & 1 \end{pmatrix} \qquad A3 := \begin{pmatrix} 1 & \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ 3 & 1 & \frac{1}{5} & \frac{1}{5} \\ 3 & 5 & 1 & 1 \\ 3 & 5 & 1 & 1 \end{pmatrix}$$

власні вектори порівняння альтернатив та їх ваги

$\lambda := \text{calculate\_eigen\_vectors}\ (A1)$ $\qquad \lambda = \begin{pmatrix} 3.708 \\ 1.848 \\ 0.615 \\ 0.27 \end{pmatrix}$

$A1\_w := \text{calculate\_weight}\ (\lambda)$ $\qquad A1\_w = \begin{pmatrix} 0.576 \\ 0.287 \\ 0.095 \\ 0.042 \end{pmatrix}$

$\lambda := \text{calculate\_eigen\_vectors}\ (A2)$ $\qquad \lambda = \begin{pmatrix} 0.312 \\ 0.589 \\ 1.699 \\ 3.201 \end{pmatrix}$

$A2\_w := \text{calculate\_weight}\ (\lambda)$ $\qquad A2\_w = \begin{pmatrix} 0.054 \\ 0.101 \\ 0.293 \\ 0.552 \end{pmatrix}$

$\lambda := \text{calculate\_eigen\_vectors}\ (A3)$ $\qquad \lambda = \begin{pmatrix} 0.439 \\ 0.589 \\ 1.968 \\ 1.968 \end{pmatrix}$

$A3\_w := \text{calculate\_weight}\ (\lambda)$ $\qquad A3\_w = \begin{pmatrix} 0.088 \\ 0.119 \\ 0.397 \\ 0.397 \end{pmatrix}$

$E_1 := c\_w_1 \cdot A1\_w_1 + c\_w_2 \cdot A2\_w_1 + c\_w_3 \cdot A3\_w_1 = 0.395$

$E_2 := c\_w_1 \cdot A1\_w_2 + c\_w_2 \cdot A2\_w_2 + c\_w_3 \cdot A3\_w_2 = 0.224$

$E_3 := c\_w_1 \cdot A1\_w_3 + c\_w_2 \cdot A2\_w_3 + c\_w_3 \cdot A3\_w_3 = 0.194$

$E_4 := c\_w_1 \cdot A1\_w_4 + c\_w_2 \cdot A2\_w_4 + c\_w_3 \cdot A3\_w_4 = 0.187$

СПИСОК РЕКОМЕНДОВАНОЇ ЛІТЕРАТУРИ

1. **Кини Р. Л.** Принятие решений при многих критериях: предпочтения и замещения / Р.Л. Кини, Х. Райфа; пер. с англ. И.Ф. Шахнова – М.: Радио и связь, 1981. – 560 с. – Библиогр.: с.547–555, – 10000 экз.

2. **Ларичев О. И.** Теория и методы принятия решений, а также Хроника событий в Волшебных странах: Учебник. Изд. второе, перераб. и доп. / О.И. Ларичев. – М. : Логос, 2003. – 392 с. – 3000 экз.– ISBN 5-94010-180-1.

3. **Таха Хемди А.** Введение в исследование операций, 7-е издание.: Пер. с англ. / Хемди А. Таха – М.: Издательский дом «Вильямс», 2005.– 912с.: ил.– Парал. тит. англ. – 3000 экз.–ISBN 5-84590740-3.

4. **Ахо Альфред В.** Структуры данных и алгоритмы.: Пер. с англ.: Уч. пос. / Альфред В. Ахо, Джон Э. Хопкрофт, Джефри Д. Ульман – М.: Издательский дом «Вильямс», 2000.– 384с. : ил.– Парал. тит. англ. – 5000 экз. – ISBN 5-8459-0122-7.

5. **Гнатієнко Г.М.** Експертні технології прийняття рішень: Монографія. / Г.М. Гнатієнко, В.Є. Снитюк – К.: ТОВ «Маклаут», – 2008.– 444с.– 1000прим. – ISBN 978-966-96939-4-8.

6. **Черноморов Г.А.** Теория принятия решений: Учебное пособие/ Г.А. Черноморов – Юж.-Рос. гос. техн. ун-т. Новочеркасск: Ред. журн. «Изв. вузов. Электромеханика», 2002, 276с. – Библиогр.: с.270–272,– 250 экз. – ISBN 588998-288-5.

7. **Черноруцкий И. Г.** Методы принятия решений / И. Г. Черноруцкий – СПб.: БХВ-Петербург, 2005. – 416 с : ил. – Библиогр.: с.395–398, – 3000 экз. ISBN 5-94157-481-9.

8. **Майника Э.** Алгоритмы оптимизации на сетях и графах: Пер. с англ. / Э. Майника – М.: Мир, 1981.–323с. – 10000 экз.

9. **Зайченко Ю.П.** Дослідження операцій. Підручник.сьоме видання, перероблене і доповнен / Ю.П. Зайченко – К.: Видавничий Дім «Слово», 2006. – 816с. – ISBN 966-8407-64-4.

10. **Кирьянов, Д. В.** Самоучитель по MathCad11 [Текст] / Д. В. Кирьянов. – СПб.: БХВ, 2003. – 560с. – 5000 экз. – ISBN 5-94157-348.0

11. **Дьяконов, В.** MATLAB 6/6.1/6.5 + Simulink 4/5 в математике и моделировании. Полное руководство пользователя [Текст] / В. Дьяконов. – М.: СОЛОН-Пресс, 2003. – 576с. – Библиогр.: с. 547.– 2000 экз. – ISBN 5-93455-177-9.

Зміст