МІНІСТЕРСТВО ОСВІТИ І НАУКИ, МОЛОДІ ТА СПОРТУ УКРАЇНИ

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ

„КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ”

ТЕХНОЛОГІЇ ШТУЧНОГО ІНТЕЛЕКТУ

ТЕОРІЯ ПРИЙНЯТТЯ РІШЕНЬ В КОМП'ЮТЕРНИХ СИСТЕМАХ УПРАВЛІННЯ

МЕТОДИЧНІ ВКАЗІВКИ

до виконання практикуму для студентів спеціальності
„Автоматизоване управління технологічними процесами”

Київ

НТУУ “КПІ”

2012

Технології штучного інтелекту. Теорія прийняття рішень в комп'ютерних системах управління: Метод. вказівки до викон. практикуму для студ. спец. „Автоматизоване управління технологічними процесами” / Уклад.: Д.О. Ковалюк. – К. : НТУУ ”КПІ“, 2012. – 26с.

Гриф надано Вченою радою ІХФ
(Протокол № 6 від 31 травня 2012р.)

Навчальне видання

ТЕХНОЛОГІЇ ШТУЧНОГО ІНТЕЛЕКТУ

ТЕОРІЯ ПРИЙНЯТТЯ РІШЕНЬ В КОМП'ЮТЕРНИХ СИСТЕМАХ УПРАВЛІННЯ

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

Методичні вказівки призначено для виконання практичних занять з дисципліни «Технології штучного інтелекту».

Укладач: Ковалюк Дмитро Олександрович, канд. техн. наук,

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

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

Авторська редакція

ЗМІСТ

ВСТУП

Задачі прийняття рішень відіграють важливу роль в сучасному процесі керування. Вони зустрічаються як при проектуванні систем так і в процесі їх експлуатації. Одним із найбільш поширених розділів прийняття рішень на сьогодні є графові моделі та алгоритми на графах.

Задачі даної групи мають багато практичних застосувань, оскільки за допомогою графів описуються елементи систем керування, стадії технологічного процесу та зв’язки між ними. Розуміння алгоритмів на графах дозволить застосувати їх в будь якій предметній області. Крім того, в роботі розглянуто прийняття рішень при багатьох критеріях.

Структура видання передбачена таким чином, що в кожній роботі описується постановка задачі, ідея та алгоритм методу, а також застосування даного алгоритму до конкретної задачі.

Робота на практичних заняттях дозволяє студентам засвоїти всі етапи прийняття рішень: постановку задачі та її кількісний (або якісний) опис, обґрунтування методу розв’язання, його реалізацію та дослідження.

МЕТА І ЗАВДАННЯ ПРАКТИЧНИХ ЗАНЯТЬ

Метою практичних занять є одержання необхідних знань та вмінь для розв'язання задач прийняття рішень при проектуванні та керуванні складними системами.

В результаті студенти закріплюють наступні знання:

- типи та класи задач прийняття рішень, їх місце в інтелектуальних системах керування;
- графові моделі систем та алгоритми прийняття рішень на графах;
- методи прийняття рішень в багатокритеріальних задачах.

Після практичних занять студенти повинні вміти:

- створювати графові моделі систем керування;
- здійснювати вибір оптимального алгоритму для розв'язання поставленої задачі;
- виконати математичний розв'язок задачі за обраним алгоритмом;
- розв'язувати багатокритеріальні задачі прийняття рішень:
- створювати блок-схеми алгоритмів або програмну реалізацію алгоритму у псевдокодах.

ПРАКТИЧНЕ ЗАНЯТТЯ №1

РОЗВ'ЯЗАННЯ ЗАДАЧІ ПОШУКУ ВСІХ НАЙКОРОТШИХ ШЛЯХІВ ЗА АЛГОРИТМОМ ФЛОЙДА

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

Постановка задачі

Задана розподілена система керування, що містить шість мікроконролерів, пов'язаних так, як показано на рисунку 1.1.

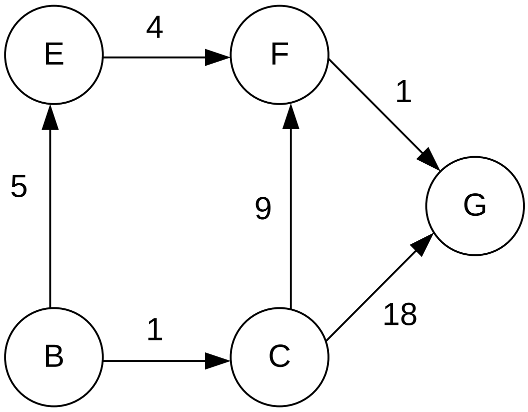

Рис. 1.1. Структура розподіленої системи керування

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

Алгоритм Флойда

Алгоритм Флойда (в літературі також зустрічається його повна назва Флойда-Уоршелла) є більш загальним в порівнянні з алгоритмом Дейкстри, оскільки знаходить найкоротші відстані між усіма вузлами мережі.

У цьому алгоритмі мережа представляється у вигляді квадратної матриці A з n-строками і n-стовпцями. Елемент A_{ij} дорівнює відстані d_{ij} від вузла i до вузла j, якщо існує дуга (i, j) і дорівнює нескінченності в протилежному випадку.

Основна ідея методу Флойда полягає в наступному. Нехай є три вузли i, j та k та задані відстані між ними (рис. 1.2.). Якщо виконується нерівність $d_{ij} < d_{ik} + d_{kj}$, то доцільно замінити відстань $i - j$ відстанню $i - k - j$. Така заміна, що має назву трикутного оператора виконується систематично в процесі виконання алгоритму [1].

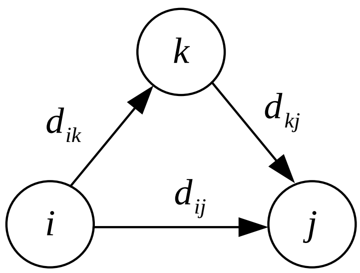

Рис. 1.2. Трикутний оператор Флойда

Алгоритм включає кількість ітерацій, що дорівнює розмірності матриці. На кожній ітерації вибирається ведучий стовпець і ведучий рядок і до всіх елементів застосовується поданий вище трикутний оператор. Схематично це можна показати наступним чином [2].

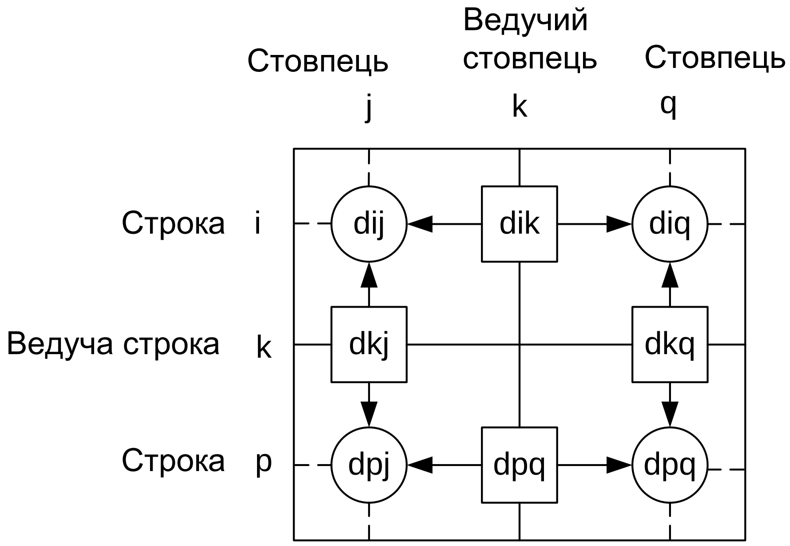

Рис. 1.3. Застосування трикутного оператора до матриці відстаней

Якщо сума елементів ведучих рядка і стовпця (показаних в квадратиках) менше елементів, що знаходяться на перетині стовпця і рядка (показані в колах), відповідних даним провідним елементам, то відстань (елемент в колі) замінюється сумою відстаней, представлених провідними елементами.

Формалізація алгоритму Флойда

При реалізації даного алгоритму [2] необхідні дві матриці для збереження відстаней D та шляху S.

Ініціалізація. Визначаємо початкову матрицю відстаней D і матрицю послідовності вузлів S. Діагональні елементи обох матриць позначаються знаком "–", оскільки ці елементи не приймають участі в обчисленнях

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

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

Робочий крок. Задаємо рядок k і стовпець k як ведучий рядок і ведучий стовпець. Розглядаємо можливість використання трикутного оператора до всіх елементів d_{ij} матриці. Якщо виконується нерівність:

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

тоді:

- замінюємо в матриці D елемент d_{ij} сумою $d_{ik} + d_{kj}$;
- змінюємо в матриці S елемент s_{ij} на k.

Після виконання n етапів алгоритму в матриці D містяться найкоротші відстані між вершинами, а в матриці S шляхи між вузлами. Для того щоб знайти відстань і шлях між вузлами i та j виконуються наступні дії.

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.

Приклад

Знайдемо всі найкоротші шляхи між вершинами графу, представленого на рис. 1.1. В лівій колонці містяться матриці відстаней в правій – матриця шляхів. Кожен рядок – одна ітерація алгоритму Флойда. Для наочності жовтим кольором виділено зміни (покращення) на кожному кроці.

3.	B	C	E	F	G
B	-	1	5	12	∞
C	∞	-	∞	9	18
E	∞	∞	-	4	∞
F	∞	∞	∞	-	1
	∞	∞	∞	∞	-

	B	C	E	F	G
B	-	C	E	F	G
C	B	-	E	F	G
E	B	C	-	F	G
F	B	C	E	-	G
G	B	C	E	F	-

	B	C	E	F	G
B	-	1	5	10	19
C	∞	-	∞	9	18
E	∞	∞	-	4	∞
F	∞	∞	∞	-	1
G	∞	∞	∞	∞	-

	B	C	E	F	G
B	-	C	E	C	C
C	B	-	E	F	G
E	B	C	-	F	G
F	B	C	E	-	G
G	B	C	E	F	-

	B	C	E	F	G
B	-	1	5	9	19
C	∞	-	∞	9	18
E	∞	∞	-	4	∞
F	∞	∞	∞	-	1
G	∞	∞	∞	∞	-

	B	C	E	F	G
B	-	C	E	E	C
C	B	-	E	F	G
E	B	C	-	F	G
F	B	C	E	-	G
G	B	C	E	F	-

	B	C	E	F	G
B	-	1	5	9	10
C	∞	-	∞	9	10
E	∞	∞	-	4	5
F	∞	∞	∞	-	1
G	∞	∞	∞	∞	-

	B	C	E	F	G
B	-	C	E	E	F
C	B	-	E	F	F
E	B	C	-	F	F
F	B	C	E	-	G
G	B	C	E	F	-

Література

1. 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.

2. 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.

Контрольні запитання

1. Результат роботи алгоритму Флойда.
2. Трикутний оператор Флойда.
3. Структури збереження даних при реалізації алгоритму.
4. Як знайти шлях між вершинами використовуючи матрицю шляхів S.

ПРАКТИЧНЕ ЗАНЯТТЯ №2

РОЗВ'ЯЗАННЯ ЗАДАЧІ ПОШУКУ НАЙКОРОТШОГО ШЛЯХУ МІЖ ВЕРШИНАМИ ГРАФУ МЕТОДОМ ГІЛОК ТА МЕЖ

Мета роботи – ознайомитися із задачею пошуку найкоротшого шляху між вершинами графу. Дослідити метод «гілок і меж» на прикладі задачі визначення найкоротшого шляху між двома вершинами графу.

Постановка задачі

Задана розподілена система керування, що містить сім мікроконролерів, пов'язаних так, як показано на рисунку 2.1.

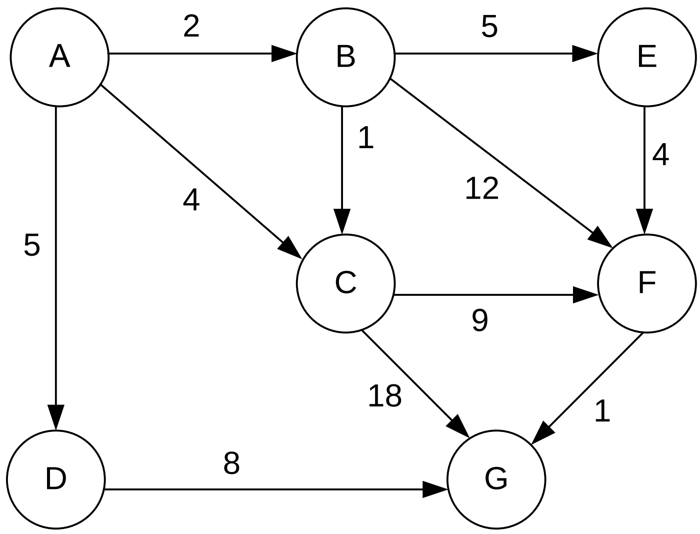

Рис. 2.1. Структура розподіленої системи керування

Необхідно знайти відстань між двома заданими вершинами методом гілок та меж.

Алгоритм методу гілок та меж

В основі методу гілок та меж лежить ідея послідовного розбиття множини допустимих рішень на підмножини (стратегія - "розділяй і володарюй"). На кожному кроці методу елементи розбиття піддаються перевірці для з'ясування, містить дана підмножина оптимальне рішення чи ні. Така перевірка здійснюється за допомогою обчислення оцінки знизу для цільової функції на даній підмножині. Після аналізу всіх підмножин – отримується найкоротша відстань між вершинами.

Для даної задачі введемо наступні позначення:

Цільова функція - довжина шляху між вершинами (контролерами)

Нижня оцінка підмножини розв'язків, що розглядається на поточному кроці, - довжина шляху від початкової вершини до поточної вершини множини.

Рекорд – довжина найкоротшого шляху від початкової до заданої вершини.

Галуження – процедура розбиття множини на підмножини.

Приклад

Для заданого на рис. 2.1. графу знайти найкоротшу відстань між вершинами (мікроконтролдерами) A та G.

Крок 1. Розбиття початкової множини на 3 підмножини, кожна з яких містить по одній вершині (рис. 2.2).

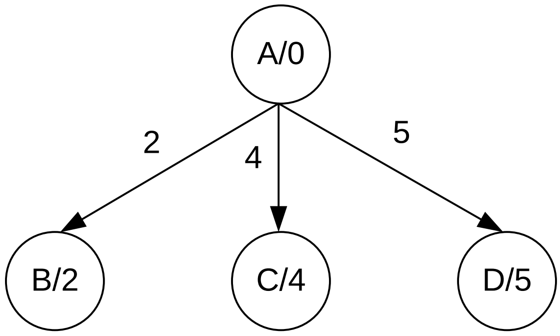

Рис. 2.2. Дерево пошуку на ітерації №1

Нижні оцінки даних множин складають відповідно 2, 4, та 5.

Крок.2. Необхідно виконати галуження. Галуження виконується з тієї вершини, яка має найменшу нижню оцінку. Таким чином з вершини B виконується галуження, в результаті якого утворюються 3 підмножини, що містять вершини C, E, F.

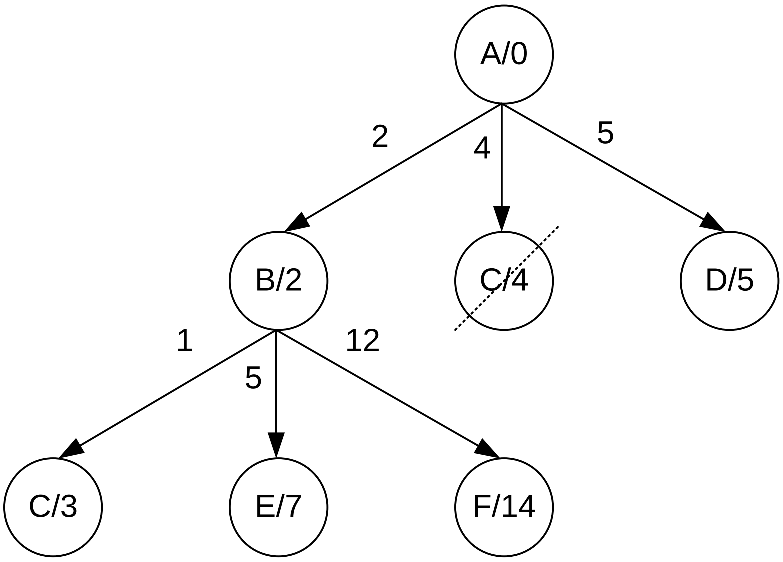

Рис. 2.3. Дерево пошуку на ітерації №2

Тоді нижню оцінку для кожної з цих підмножин можна обчислити наступним чином: *для* C*: 2+1=3, для* $E: 2+5=7$*, для* F*: 2+12=14.* Наступне галуження виконується з вершини C. Зауважимо, що із подальшого розв'язку може бути вилучений шлях $A \to C$, оскільки до вершини C є коротший шлях $A \to B \to C$.

Застосовуючи алгоритм методу отримаємо наступні дерева шляхів:

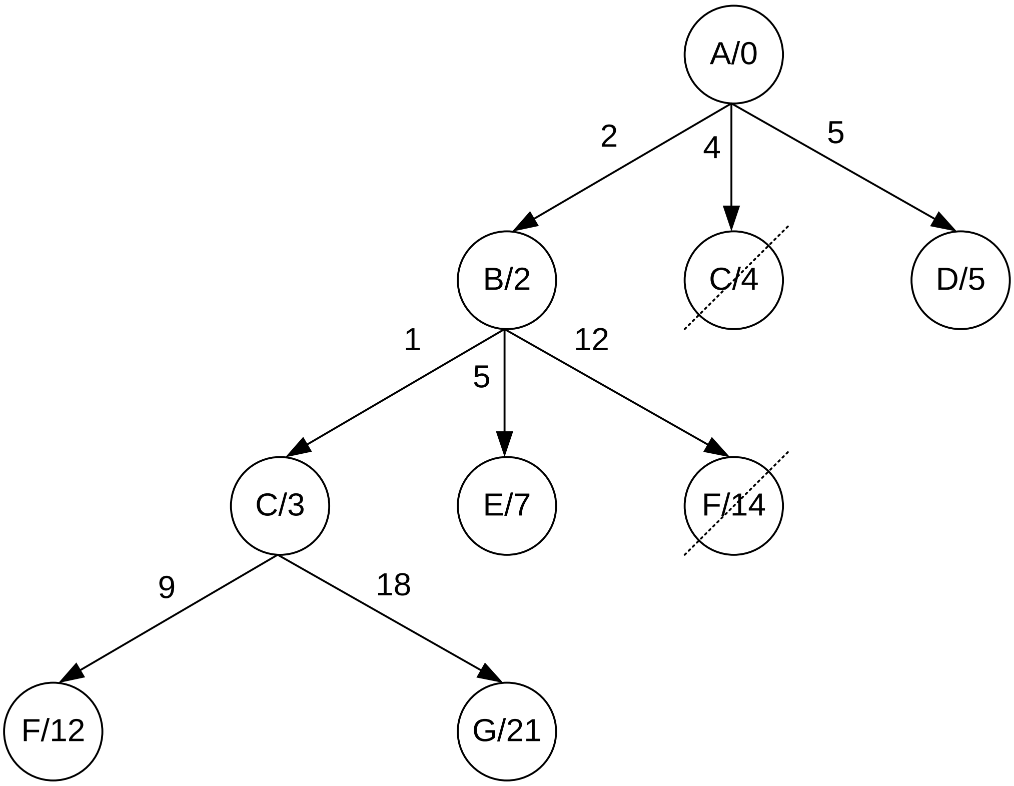

Рис.2.4. Дерево пошуку на ітерації №3

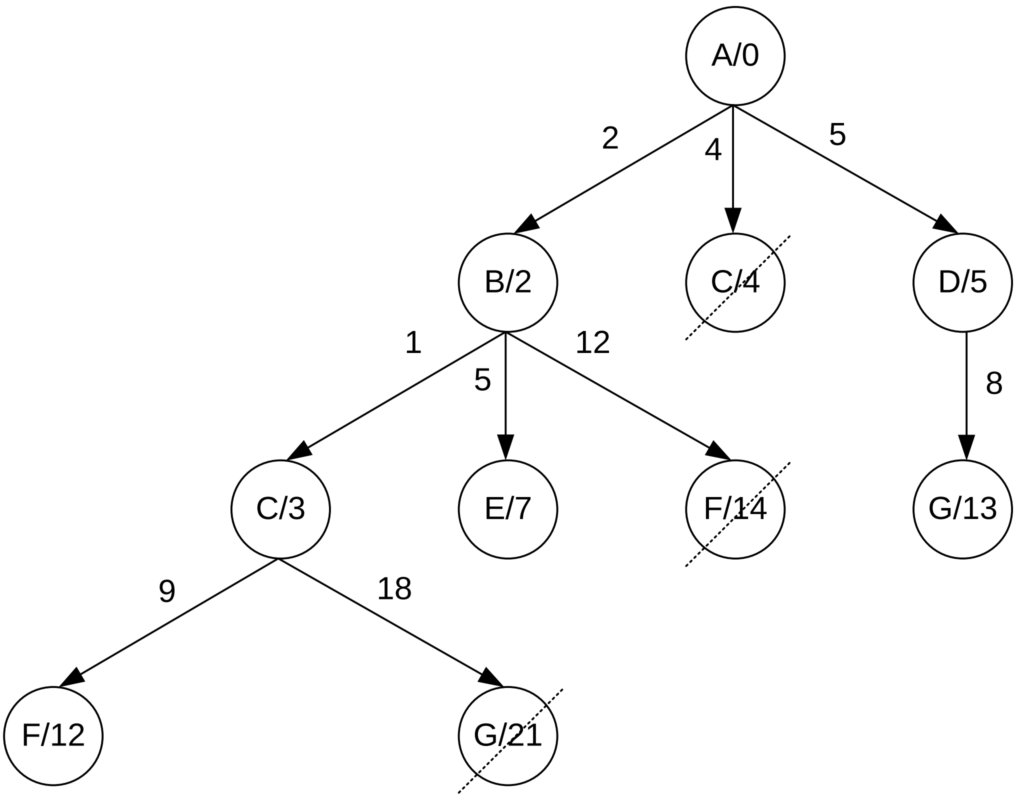

Рис. 2.5. Дерево пошуку на ітерації №4

Зауважимо, що третій ітерації отримуємо перше значення рекорду – шляху між заданими вершинами, яке дорівнює 21. Але уже на наступному кроці при галуженні з вершини D отримуємо нове значення рекорду – 13.

Крок 5. Виконуємо галуження з вершини з мінімальною оцінкою знизу з вершини E. Отримуємо наступне дерево шляхів.

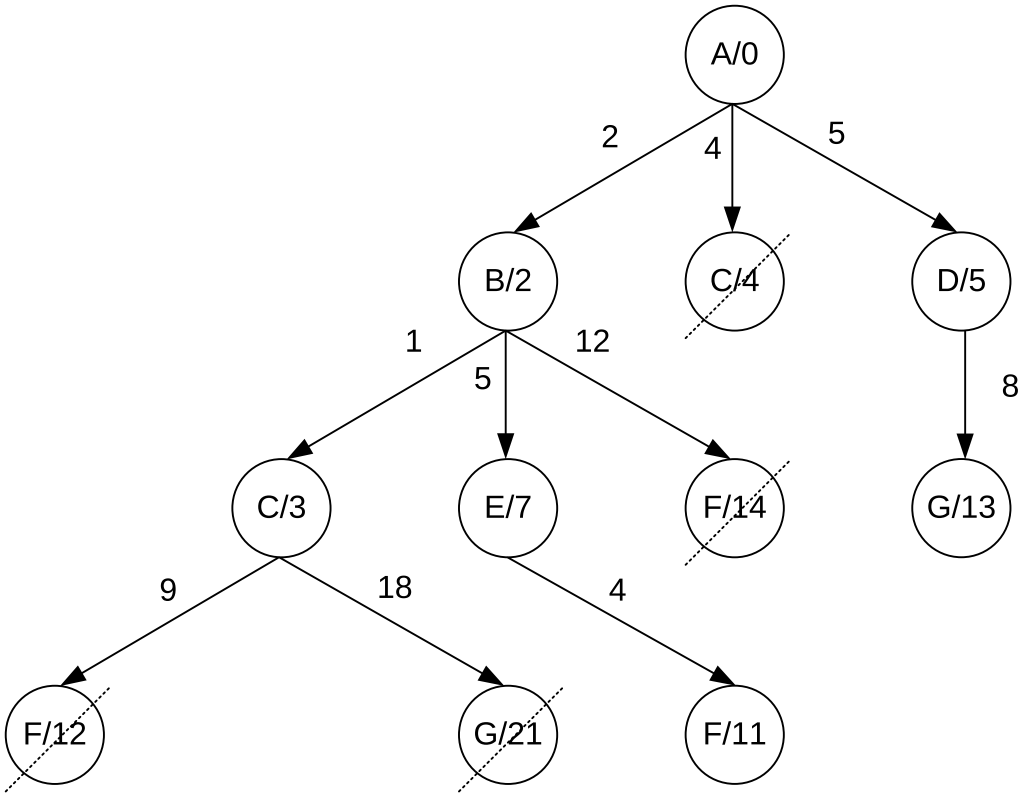

Рис. 2.6. Дерево пошуку на ітерації №5

Крок 6. Це останній крок, на якому виконується з вершини F в вершину G. Знову отримуємо рекорд, причому із значенням меншим за мінімальне. Оскільки галужень в графі вже немає, то найкоротший шлях буде містити наступні вершини. $A \rightarrow B \rightarrow E \rightarrow F \rightarrow G$.

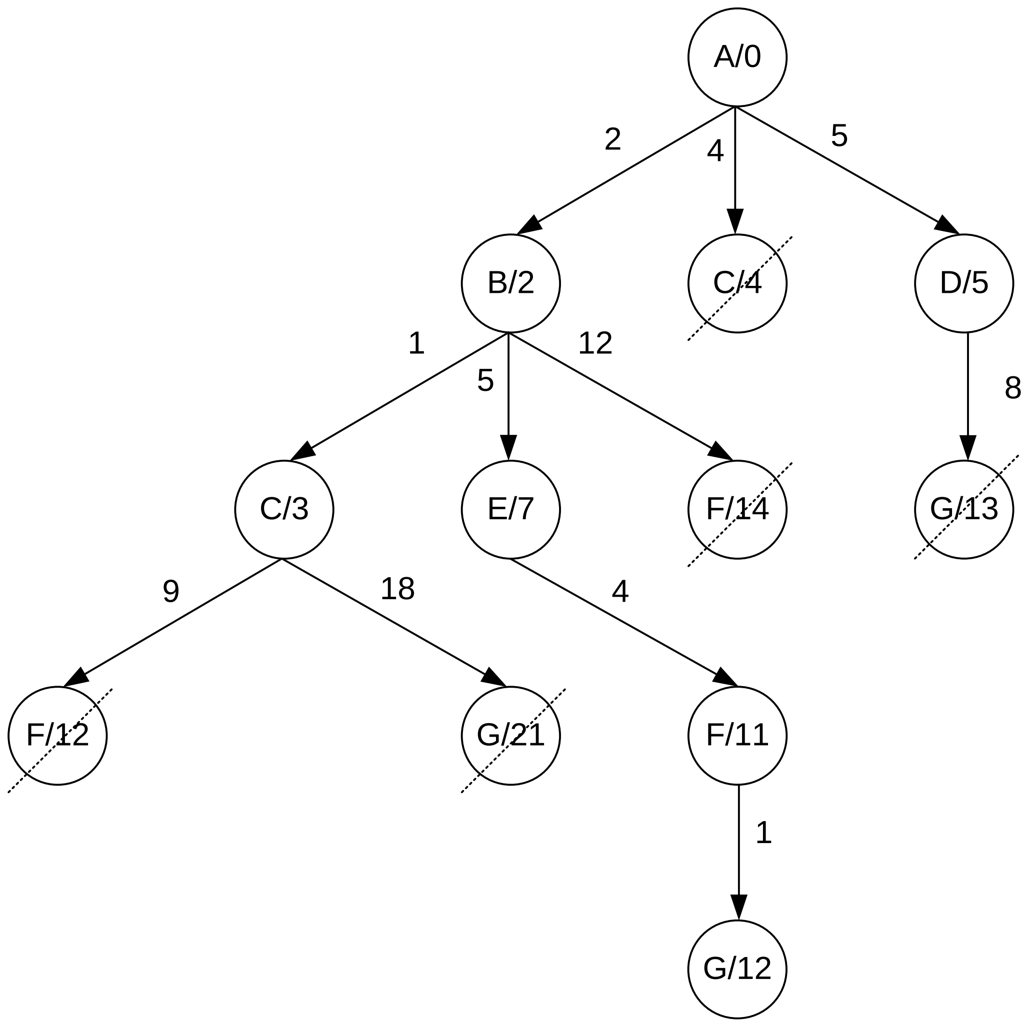

Рис. 2.7. Дерево пошуку на останній ітерації №6

Література

1. 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.

Контрольні запитання

1. Основна ідея методу гілок та меж.
2. Визначення рекорду, оцінки знизу, галуження.
3. Правило виконання галуження та обчислення оцінки знизу.

ПРАКТИЧНЕ ЗАНЯТТЯ №3

РОЗВ'ЯЗАННЯ БАГАТОКРОКОВИХ ЗАДАЧ ПРИЙНЯТТЯ РІШЕНЬ МЕТОДОМ ДИНАМІЧНОГО ПРОГРАМУВАННЯ

Мета роботи – ознайомитися з багатокроковими задачами прийняття рішень. Дослідити метод динамічного програмування на прикладі задачі комівояжера.

Постановка задачі

Класична постановка задачі комівояжера полягає в тому, що є сукупність міст, які необхідно відвідати. Потрібно знайти оптимальний (найкоротший) маршрут, який поєднує усі міста.

В технічних системах аналогом цієї задачі виступають задачі прокладання лінії електропередач, об'єднання персональних ЕОМ в комп'ютерну мережу, тощо.

Формалізуємо постановку задачі.

Задано n – об'єктів, які необхідно з'єднати між собою:

$$M = \{M_1, M_2, M_3, \dots, M_n\}$$

Задано матрицю відстаней:

$$D = |d_{i,j}|$$

Треба знайти перестановку об'єктів:

$$P_n = < i_1, i_2, i_3, \dots, i_n >,$$

для якої довжина маршруту

$$d_{i1\,i2} \to d_{i2\,i3} \to d_{i3i4} \to \dots \to d_{in\,i1} = min$$

буде оптимальною.

Алгоритм методу динамічного програмування

Основна ідея методу динамічного програмування [1] полягає в тому, що для розв'язання поставленої задачі, необхідно вирішити окремі підзадачі, після чого об'єднати їх розв'язки в один загальний розв'язок. Знаходження оптимального розв'язку частинних задач (етапів основної задачі) базується на показнику локального доходу – критерію оптимальності.

Застосування динамічного програмування для задачі комівояжера

Згідно методу динамічного програмування розв'язання починається з останнього етапу. Зафіксуємо M_n - кінцеве місто, куди має потрапити комівояжер. При цьому комівояжер може знаходитись в M_1, M_2,…, M_{n-1} місті. Стан системи будемо представляти у формі $\mathrm{M}_i\{0\}$, де A_i – місто, в якому знаходиться комівояжер перед прийняттям рішення; $\{0\}$ – означає, що між кінцевим і даним містом відсутні проміжні міста. Така форма запису, запропонована в [2], є зручною та наочною.

Кожному стану системи (місцю перебування комівояжера) відповідає певний локальний дохід, який обчислюється як відстань $D_{M_i M_n}$ від M_i до M_n міста.

Передостанній етап - комівояжеру треба потрапити в кінцеве місто, за умови, коли є одне проміжне місто. Тобто комівояжеру з міста M_i треба заїхати в проміжне місто, а звідти в M_n.

І так далі для всіх етапів, поки кількість проміжних міст не буде дорівнювати загальній кількості міст.

Приклад

Розглянемо приклад, наведений в роботі [2]. Нехай маємо 4 міста, які описуються наступною матрицю відстаней між ними:

$$D = \begin{pmatrix} \infty & 8 & 8 & 5 \\ 9 & \infty & 1 & 6 \\ 11 & 4 & \infty & 4 \\ 6 & 3 & 2 & \infty \end{pmatrix}$$

Етап 1. Комівояжер має потрапити в останнє місто. При цьому він може знаходитися в містах 1, 2, 3. Локальний дохід розраховуємо за формулою:

$$W_4(M_i\{0\}) = D_{M_i M_4}$$

Для кожного з міст маємо:

$W_4(M_1\{0\}) = 5$

$W_4(M_2\{0\}) = 6$

$W_4(M_3\{0\}) = 4$

Етап 2. Комівояжер має потрапити в останнє місто, відвідавши перед цим ще одне місто. Локальний дохід розраховуємо за формулою:

$$W_4(M_i\{0\}) = D_{M_i M_4}$$

Для кожного з міст;

$W_3(M_1\{M_2\}) = D_{12} + W_4(M_2\{0\})$ = 8 + 6 = 14

$W_3(M_1\{M_3\}) = D_{13} + W_4(M_3\{0\})$ = 8 + 4 = 12

$W_3(M_2\{A_1\}) = D_{21} + W_4(M_1\{0\})$ = 9 + 5 = 14

$W_3(M_2\{A_3\}) = D_{23} + W_4(M_3\{0\})$ = 1 + 4 = 5

$W_3(M_3\{A_1\}) = D_{31} + W_4(M_1\{0\})$ = 11 + 5 = 16

$W_3(M_1\{A_2\}) = D_{32} + W_4(M_2\{0\})$ = 4 + 6 = 10

Етап 3. Комівояжер має потрапити в останнє місто, відвідавши при цьому два проміжних міста. Локальний дохід на даному етапі визначається:

$W_2(M_1\{M_2M_3\}) = \min\big(D_{12} + W_3(M_2\{M_3\}), D_{13} + W_3(M_3\{M_2\})\big) = \min(8 + 16, 8 + 10)$
$= 18$
$W_2(M_2\{M_1M_3\}) = \min\big(D_{21} + W_3(M_1\{M_3\}), D_{23} + W_3(M_3\{M_1\})\big) = \min(9 + 12, 1 + 16)$
$= 17$
$W_2(M_3\{M_1M_2\}) = \min\big(D_{31} + W_3(M_1\{M_2\}), D_{32} + W_3(M_2\{M_1\})\big) = \min(11 + 14, 4 + 14)$
$= 18$

Останній 4 етап. Комівояжер знаходиться в останньому місті, повинен об'їхати всі міста і повернутися в те саме місто. Локальний дохід визначається наступним чином

$$W_1(M_4\{M_1M_2M_3\}) = \min(D_{41} + W_2(M_1\{M_2M_3\}),\ D_{42} + W_2(M_2\{M_1M_3\}),\ D_{43} + W_2(M_3\{M_1M_2\})) = \min(6 + 13, 3 + 17, 2 + 18) = 19$$

Таким чином оптимальний маршрут, який пов'язує всі міста буде W=4→1→3→2→4, з довжиною шляху 19.

Література

1. Акулич И.Л. Математическое программирование в примерах и задачах: Учеб. пособие. / И.Л. Акулич - М.: Высш шк., 1986.-316с.

2. 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.

Контрольні запитання

1. Постановка задачі комівояжера.
2. Визначення та приклад багатокрокового процесу прийняття рішень.
3. Основна ідея методу динамічного програмування.
4. Взаємозв'язок стану системи, керування та локального доходу.

ПРАКТИЧНЕ ЗАНЯТТЯ №4

РОЗВ’ЯЗАННЯ БАГАТОКРИТЕРІАЛЬНИХ ЗАДАЧ ПРИЙНЯТТЯ РІШЕНЬ МЕТОДОМ АНАЛІЗУ ІЄРАРХІЙ

Мета роботи – ознайомитися з багатокритеріальними задачами прийняття рішень, дослідити метод аналізу ієрархій.

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. 4.1.).

Табл. 4.1. 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.

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 (рис. 4.1.)

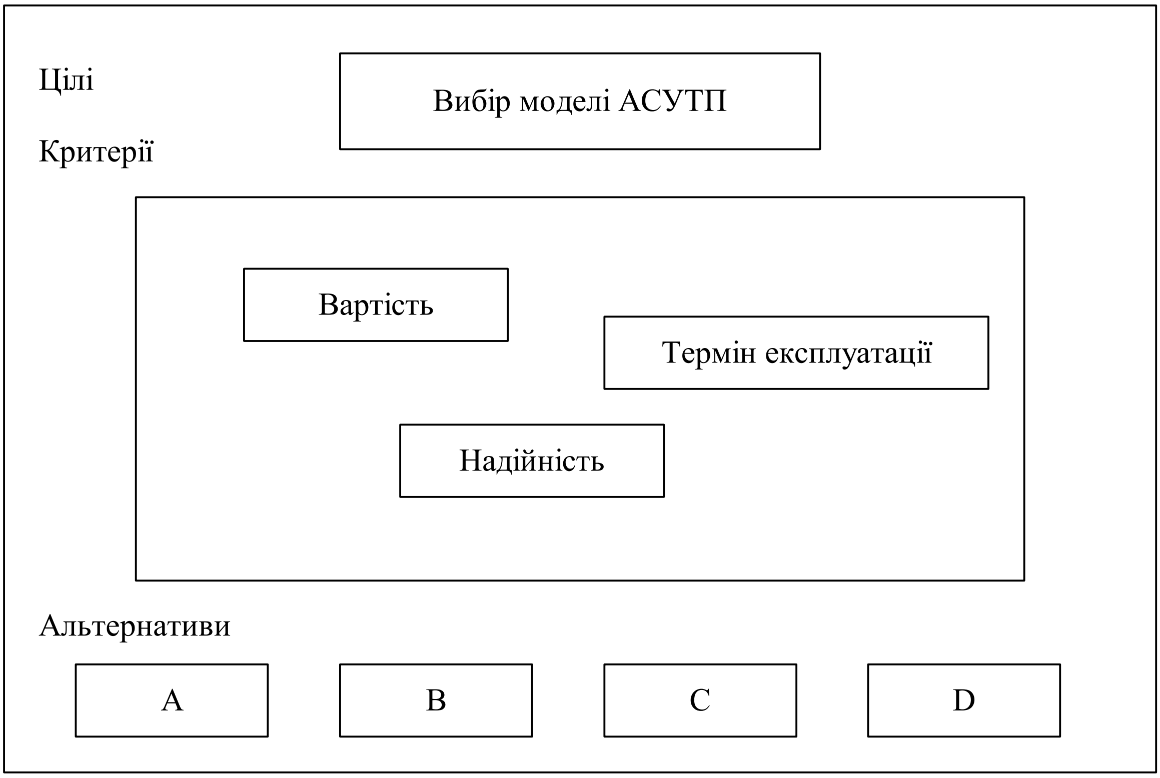

Рис. 4.1. Ієрархічна схема проблеми вибору АСУТП

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. 4.2. 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. 4.3. 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. 4.4. Відносна важливість альтернатив за окремими критеріями

По критерію С1 (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 С2 (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 4.3 і 4.4 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

$$S_j = \sum_{i=1}^{N} w_i \cdot V_{ij} \qquad (4.1)$$

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.

Література

1. 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

2. 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.

Контрольні запитання

1. Визначення багатокритеріальних задач прийняття рішень.
2. Шкала порівняння критеріїв та альтернатив.
3. Алгоритм методу аналізу ієрархій.
4. Формула розрахунку оцінок альтернатив.