Львівський державний університет безпеки життєдіяльності

О.В. Меньшикова, О.Ю. Чмир, О.О. Карабин

Дослідження операцій

НАВЧАЛЬНИЙ ПОСІБНИК

Львів -2019

УДК 519.8
ББК 22.1

Меньшикова, Ольга Володимирівна.
Дослідження операцій [Текст] : [навчальний посібник] / Меньшикова О.В., Чмир О.Ю., Карабин О.О. – Львів : ЛДУ БЖД, 2019. – 196 с.

У навчальний посібник увійшли основні теоретичні відомості дослідження операцій. Теоретичний матеріал ілюструється прикладами.

Рецензенти: ***Н.П. Процах*** – завідувач кафедри математики і фізики, доктор фізико-математичних наук, доцент Національного лісотехнічного університету України;

А.Д. Кузик – доктор сільськогосподарських наук, професор Львівського державного університету безпеки життєдіяльності.

Рекомендовано до друку вченою радою
Львівського державного університету безпеки життєдіяльності
(протокол № 4 від "26" листопада 2019 р.)

ЗМІСТ

ВСТУП

Навчальні дисципліни “Дослідження операцій в транспортних системах” та “Математичні методи дослідження операцій” є важливими складовими частинами у системі підготовки фахівців технічних і природничих напрямів.

Навчальний посібник “Дослідження операцій” призначений ознайомити читача із задачами лінійного, нелінійного та динамічного програмування, транспортними задачами, елементами теорії графів, мережевими моделями та моделями теорії масового обслуговування.

В результаті вивчення дисциплін читач знатиме основні етапи розв’язання задач дослідження операцій, типові моделі, що використовуються для розв’язання прикладних задач, блоки, що використовуються для розв’язання задач оптимізації у середовищі Maple; вмітиме формулювати задачі як задачі дослідження операцій, підбирати до них математичні моделі та розробляти програмні реалізації типових задач дослідження операцій.

Теоретичний матеріал проілюстровано прикладами, більшість яких розв’язано у пакеті Maple. Кожна тема завершується контрольними запитаннями, на які рекомендовано дати відповіді для самоперевірки засвоєного матеріалу.

Рекомендована література містить додаткову інформацію з дослідження операцій, яка не увійшла до навчального посібника.

Бажаємо вам успіху у вивченні дослідженні операцій.

ТЕМА 1. ПРЕДМЕТ ТА ЗАДАЧІ ДОСЛІДЖЕННЯ ОПЕРАЦІЙ

1.1. Основні поняття та визначення

Дослідження операцій – комплексна наукова дисципліна, яка передбачає застосування математичних методів для обґрунтування рішень в усіх галузях людської діяльності.

На обкладинці британського журналу з дослідження операцій («Operational Research Quarterly», що видається з 1950 року) наводиться таке визначення дисципліни: **дослідження операцій** (operational research) – застосування наукових методів до складних проблем, що виникають при керуванні великими системами людей, машин, матеріалів і грошей в промисловості, діловій сфері, уряді та обороні.

Характерною особливістю підходу є побудова для системи наукової моделі, що містить фактори імовірності та ризику, за допомогою якої можна розрахувати і порівняти результати різних рішень, стратегій та управлінських дій.

Фахівці з дослідження операцій Е. С. Вентцель та Т. Л. Сааті дають такі визначення: **дослідження операцій** – застосування математичних, кількісних методів для обґрунтування рішень у всіх галузях цілеспрямованої діяльності людини.

Якщо трішки пожартувати, то можна стверджувати, що дослідження операцій – мистецтво давати погані відповіді на практичні питання, на які інші методи дають ще гірші відповіді.

Вважається, що термін «дослідження операцій» вперше з'явився в роботі Чарлза Беббіджа (Charles Babbage, 1791-1871), який застосовував науковий підхід до різноманітних проблем (на пошті, при виробництві шпильок, в друкарській справі). Формування дослідження операцій як науки відносять до років Другої світової війни, коли у збройних силах США та Англії було сформовано спеціальні наукові групи з підготовки рішень

військових задач. В 1939 - 1940 роках з'явилися перші публікації, в яких методи дослідження операцій було застосовано до аналізу і дослідження бойових операцій. Звідси і походить назва науки.

З часом наука вийшла з вузької сфери військових задач. На сьогодні дослідження операцій має широке коло застосування – економіка різних галузей промисловості, торгівля, охорона здоров'я, транспорт. Сфера застосування дослідження операцій постійно розширюється.

Чим складніший впроваджуваний захід, чим більше вкладають у нього коштів, тим більшого значення набуває сукупність наукових методів, що дозволяють:

заздалегідь оцінити наслідки кожного рішення;

заздалегідь відкинути неприпустимі варіанти;

рекомендувати варіанти, що представляються найбільш вдалими.

Мета дослідження операцій – кількісне обґрунтування рішень щодо управління організаціями.

Отже, дослідження операцій має яскраво виражену практичну спрямованість. Для вирішення практичних задач дослідження операцій має значний арсенал математичних засобів. Це теорія ймовірності та її новітні розділи – теорія випадкових процесів і теорія масового обслуговування, математичні методи оптимізації, теорія ігор і статистичних рішень, методи моделювання і системного аналізу, моделі мережевого планування тощо.

Загалом моделі дослідження операцій – це оптимізаційні моделі *максимізації (мінімізації) функції мети за умови виконання певних обмежень. А саме:*

знайти змінні x_1, x_2, $\dots x_m$, $m \in N$ (керовані величини, залежні від нас фактори, які ми можемо вибирати на свій розсуд), що задовольняють систему обмежень

$$g_i(x_1, x_2, ..., x_m, a_1, a_2, ..., a_k) \le b_i,\ i = \overline{1, n},\ m, k \in N$$

та максимізують (мінімізують) функцію мети

$$W = f(x_1, x_2, ..., x_m, c_1, c_2, ..., c_k),$$

де c_1, c_2,…, c_k – некеровані величин (умови проведення операції, на які ми впливати не можемо).

Функція мети (цільова функція) – це певний показник ефективності системи, який залежить як від керованих, так і від некерованих величин. Показником ефективності може бути прибуток, витрати, сумарна відстань, час виконання певного комплексу робіт тощо.

Розглянемо основні поняття дослідження операцій.

Операція – це будь-який керований захід, спрямований на досягнення певної мети.

Результат операції залежить від організації її проведення, тобто від вибору певних параметрів.

Наведемо приклади операцій.

Приклад 1.1. Підприємство випускає кілька видів виробів, в процесі виготовлення яких використовуються обмежені ресурси різного типу. Потрібно скласти планування випуску кількості виробів кожного виду так, щоб максимізувати прибуток, виконавши обмеження на споживані ресурси.

Приклад 1.2. Потрібно створити мережу тимчасових торгівельних закладів так, щоб забезпечити максимальну ефективність продажів. Для цього потрібно визначити:
кількість торгівельних закладів, їхнє розміщення, кількість персоналу і їхню зарплату, ціни на товари.

Приклад 1.3. Потрібно організувати будівництво деякого об’єкта. При цьому необхідно вказати порядок виконання робіт у часі й розподілити необхідні ресурси між роботами так, щоб завершити будівництво вчасно і мінімізувати його вартість.

Всякий визначений вибір керованих величин, що характеризують спосіб організації операції, називають **рішенням.**

Рішення називають **допустимим**, якщо воно задовольняє систему обмежень.

Рішення називають **оптимальним**, якщо воно допустиме і за такого рішення функція мети набуває свого оптимального значення (максимуму або мінімуму).

Характерною рисою операційних досліджень є те, що їх проводять комплексно, за багатьма напрямами. Для цього створюють операційну групу, до складу якої входять фахівці з різних галузей знань: математики, інженери, економісти, соціологи.

Саме прийняття рішення входить до компетенції відповідальної особи (або групи осіб) – людини, якій надано право остаточного вибору (командира, особи, яка приймає рішення). При виборі рішення відповідальна особа враховує поряд з рекомендаціями, що випливають з математичних розрахунків, ще й ряд міркувань кількісного та якісного характеру, які не можуть бути враховані розрахунком, так званий людський фактор, наприклад, при виборі ділового партнера.

1.2. Типові класи задач дослідження операцій

Практичні задачі дослідження операцій за змістовною постановкою можна поділити на класи задач, що об'єднані своєю цільовою спрямованістю і різняться підходами та методами рішення. Стисло викладемо їхні особливості.

1. Задачі управління запасами

Умови: зі збільшенням запасів зростають витрати на їхнє зберігання, але зменшуються втрати через їхню нестачу.

Задача: визначити такий рівень запасів, який мінімізує суму витрат на зберігання і втрат через дефіцит:

$$S_{збереження} + S_{втрат} \to \min .$$

Залежно від умов, задачі управління запасами поділяють на три групи:

- моменти поставок або замовлень на запаси фіксовані. Визначити обсяги запасів, що виробляються чи закуповуються;
- обсяги запасів, що виробляються чи закуповуються, фіксовані. Визначити моменти оформлення замовлень на поставки;
- моменти оформлення замовлень і обсяги запасів, що виробляються чи закуповуються, не фіксовані. Визначити ці величини, виходячи з критерію мінімізації суми витрат і втрат.

2. Задачі розподілу ресурсів

Умови: є набір робіт, які необхідно виконати, а ресурсів для виконання робіт найкращим чином не вистачає.

Задачі розподілу ресурсів залежно від умов поділяють на три групи:

- задано і роботи, і ресурси. Розподілити ресурси між роботами так, щоб максимізувати прибуток (або мінімізувати затрати);
- задано тільки ресурси. Визначити перелік робіт, які можна виконати і забезпечити при цьому максимум певного показника ефективності;
- задано тільки роботи. Визначити, які потрібно ресурси, щоб мінімізувати затрати на виробництво.

3. Задачі ремонту і заміни обладнання

Умови: зношене обладнання підлягає або ремонту або заміні. Визначити такий строк ремонту або момент заміни, щоб мінімізувати суму затрат на ремонт і заміну за весь термін експлуатації, тобто:

$$S_{ремонт} + S_{заміна} \to \min .$$

4. Задачі масового обслуговування

Умови: потік заявок (вимог) випадковий і некерований, внаслідок чого виникають черги у системах обслуговування.

Задача: визначити кількість каналів обслуговування так, щоб мінімізувати втрати від несвоєчасного обслуговування та простоювання обладнання.

5. Задачі впорядкування (інші назви – складання розкладів, календарного планування)

Умови: є різні деталі з певними технологічними маршрутами і кілька різних обробних станків; одночасно обробляти більше однієї деталі на станку неможливо (тобто можливі черги).

Задача: визначити черговість обробки деталей на кожному станку, за якої мінімізується:

- загальна тривалість робіт;
- загальне запізнення по всіх деталях відносно встановленого терміну виконання;
- максимальне запізнення (відносно деталі, для якої запізнення є найбільшим).

6. Задачі мережевого планування і управління (СПУ)

Умови: комплексна робота складається з ряду окремих робіт (операцій). Для кожної операції відомо, які операції їй передують, а які – йдуть слідом за нею (тобто вони впорядковані і взаємообумовлені). Відомі тривалість та вартість виконання комплексу робіт і окремих робіт, сировинні, енергетичні та людські ресурси.

Задача 1: визначити моменти початку кожної операції так, щоб укластися в термін виконання комплексу робіт.

Задача 2: розподілити матеріальні та трудові ресурси між окремими роботами.

7. Задачі вибору маршруту (мережеві задачі на транспорті)

Умови: проїзд з міста *А* до міста *Б* можливий декількома маршрутами з різними проміжними пунктами. Вартість проїзду і витрати часу залежать від вибраного маршруту.

Задача: визначити найбільш економічний маршрут за обраним критерієм оптимальності (найбільш швидкий, найбільш короткий, найбільш дешевий).

8. Комбіновані задачі

Комбіновані задачі включають кілька типових задач одночасно.

Приклад 1.4. При плануванні та управлінні виробництвом потрібно:

визначити обсяги випуску виробів кожного виду (задача планування);

розподілити замовлення за видами обладнання (задача розподілу);

визначити послідовність виконання замовлень (задача впорядкування).

1.3. Основні етапи операційного дослідження

Кожне операційне дослідження проходить послідовно такі основні етапи.

1. Постановка задачі. А. Ейнштейн якось сказав, що правильна постановка задачі навіть більш важлива, ніж саме рішення. Дослідження задачі звичайно починають з вивчення системи під керівництвом відповідальної особи.

Насамперед задачу формулюють з точки зору замовника. Така постановка звичайно не буває остаточною. Аналізуючи систему – об'єкт дослідження, операційна група вивчає множину факторів, що впливають на результати досліджуваного процесу, визначає сукупність істотних факторів і формулює остаточно змістовну (словесну) постановку задачі.

2. Побудова математичної моделі. Маючи строгу несуперечливу змістовну постановку задачі, потрібно побудувати її математичну модель. Цей процес називають формалізацією задачі.

3. Знаходження рішення. Залежно від виду і структури цільової функції та обмежень для знаходження оптимального рішення задачі застосовують ті чи інші методи оптимізації.

У дослідженні операцій немає єдиного загального методу рішення всіх математичних моделей, які трапляються на практиці. Замість цього вибір методу рішення визначають тип і складність досліджуваної математичної моделі. Найбільш відомими та ефективними методами дослідження операцій є методи лінійного програмування, коли цільова функція і всі обмеження є лінійними функціями. Для рішення математичних моделей інших типів призначені методи цілочисельного програмування (всі змінні повинні набувати тільки цілих значень), динамічного програмування (вихідне завдання можна розбити на менші підзадачі) і нелінійного програмування (цільова функція і (або) обмеження є нелінійними функціями). Згадані методи становлять тільки частину великої кількості різноманітних доступних методів дослідження операцій.

4. Перевірка і коригування моделі. У складних системах модель є абстракцією реального процесу. Тому необхідною є перевірка наскільки близькі між собою модель і реальний процес (перевірка адекватності моделі). Формальним загальноприйнятим методом перевірки адекватності моделі є порівняння отриманого розв'язку (поведінки моделі) з відомими розв'язками або поведінкою реальної системи. Модель вважається адекватною, якщо за певних початкових умов її поведінка співпадає з поведінкою вихідної системи з тими самими початковими даними. Звичайно, це не гарантує того, що за інших початкових умов поведінка моделі буде збігатися з поведінкою реальної системи.

5. Реалізація знайденого рішення на практиці. Впровадження можна розглядати як самостійну задачу, яка завершує операційне дослідження і до якої також слід застосовувати системний підхід. Одержаному математичному розв'язку надають відповідної змістовної форми і передають замовнику у вигляді інструкцій та рекомендацій.

Контрольні запитання

1. Дайте визначення поняття «дослідження операцій».
2. Яка мета дослідження операцій?
3. Сформулюйте математичну модель дослідження операцій в загальному виді.
4. Що таке функція мети, операція, рішення?
5. Дайте визначення понять допустимий та оптимальний розв'язок.
6. Наведіть приклади типових задач дослідження операцій.
7. З яких етапів складається операційне дослідження?

ТЕМА 2. ЗАДАЧІ ЛІНІЙНОГО ПРОГРАМУВАННЯ

2.1. Постановка задачі лінійного програмування

Лінійне програмування – це метод оптимізації моделей, у яких функція мети й обмеження лінійні.

Лінійне програмування успішно застосовують у військовій справі, індустрії, сільському господарстві, транспортній галузі, економіці й навіть у соціальних науках. Широке використання цього методу також підкріплюється високоефективними комп'ютерними алгоритмами, що реалізують цей метод. На алгоритмах лінійного програмування (з огляду на їхню комп'ютерну ефективність) базуються алгоритми для інших, більш складних типів моделей і завдань дослідження операцій, включаючи цілочисельне, нелінійне й стохастичне програмування.

Прикладами реальних задач, математичні моделі яких є задачами лінійного програмування, є транспортна задача, задача планування виробництва, задача складання раціону, макроекономічна лінійна модель як модель міжгалузевого балансу, розроблена й вивчена американським ученим Василем Леонтьевим в 1920-х роках. Модель одержала назву автора, а сам автор одержав за її розробку Нобелівську премію в галузі економіки.

В стандартній формі задачу лінійного програмування можна записати так:

Задача на максимум

$$f(x_1, x_2, ..., x_n) = c_1x_1 + c_2x_2 + ... + c_nx_n \to \max,$$

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n \le b_1, \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n \le b_2, \\ ..., \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n \le b_m, \end{cases}$$

$$x_1 \ge 0,\ x_2 \ge 0,\ \ldots,\ x_n \ge 0,$$

де a_{ij}, b_i, c_j, $i=\overline{1,m}$, $j=\overline{1,n}$ – задані сталі величини, x_i – змінні.

Задача на мінімум

$$f(x_1, x_2, ..., x_n) = c_1x_1 + c_2x_2 + ... + c_nx_n \to \min,$$

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n \geq b_1, \\ a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n \geq b_2, \\ \qquad\qquad ... \\ a_{m1}x_1 + a_{m2}x_2 + ... + a_{mn}x_n \geq b_m, \end{cases}$$

$$x_1 \geq 0,\ x_2 \geq 0,\ \ldots,\ x_n \geq 0,$$

де a_{ij}, b_i, c_j, $i=\overline{1,m}$, $j=\overline{1,n}$ – задані сталі величини, x_i – змінні.

В обмеженнях задачі можуть входити як нерівності, так і рівняння.

Зауваження. Якщо в задачі лінійного програмування необхідно знайти максимум функції мети, то врахувавши, що $\max f(x_1, x_2, ..., x_n) = -\min(-f(x_1, x_2, ..., x_n))$, задачу зводимо до пошуку мінімуму лінійної функції $-f(x_1, x_2, ..., x_n)$.

2.2. Приклади задач, що зводяться до моделей лінійного програмування

Задача про розподіл ресурсів (задача планування виробництва). Для виробництва певних товарів використовуються деякі ресурси (сировина, механізми, кошти, робоча сила тощо). Відомо: скільки одиниць кожного ресурсу необхідно для виробництва одиниці кожного товару, запас кожного ресурсу, а також прибуток від реалізації одиниці кожного товару. Потрібно спланувати виробництво товарів так, щоб за наявних ресурсів загальний прибуток від виробництва був найбільшим.

Задача складання раціону (задача про суміші). Для годування тварин використовуються деякі корми, що містять в певній кількості поживні речовини. Відомо: скільки одиниць кожної поживної речовини є в одиниці кожного корму, мінімальна добова потреба в кожній поживній речовині та вартість одиниці кожного корму. Потрібно скласти добовий раціон так, щоб задовольнялась мінімальна добова потреба в поживних речовинах, а вартість раціону була найменшою.

Задача про завантаження обладнання. Підприємству потрібно за час t випустити n одиниць продукції i-го типу. Відомі: продуктивність верстатів, а також витрати на виготовлення певної продукції на кожному верстаті за одиницю часу. Потрібно скласти план роботи верстатів так, щоб загальні витрати на виробництво продукції були б мінімальними.

Транспортна задача. У пунктах призначення міститься товар, який потрібно перевезти в пункти споживання. Відомо: скільки одиниць товару потребує кожен пункт споживання, а також вартість перевезення одиниці товару. Запланувати так перевезення товару, щоб весь товар з пунктів постачання був вивезений, потреби всіх пунктів споживання були задоволені і водночас загальна вартість усіх перевезень була мінімальною.

Приклад 2.1. З пункту A в пункт B щоденно відправляються швидкі та пасажирські потяги. Дані про склад потягів, кількість та пасажиромісткість вагонів наведено в таблиці 2.1.

Сформулювати задачу, як задачу лінійного програмування для визначення кількості швидких та пасажирських потягів, які необхідні для перевезення максимальної кількості пасажирів.

Таблиця 2.1

Тип вагона	плацкарт	купейний	м'який	багажний	поштовий
К-ть вагонів у пасажирському потягу, шт.	5	7	3	–	1
К-ть вагонів у швидкому потягу, шт.	10	5	1	1	1
Кількість вагонів у депо, шт.	70	32	15	8	11
Пасажиромісткість, пас.	54	36	18	–	–

Розв'язання.

Позначимо x_1 – кількість пасажирських потягів, x_2 – кількість швидких потягів. Пасажиромісткість одного пасажирського потяга 54·5+36·7+18·3=576 пас., одного швидкого потяга 54·10+36·5+18·1=738 пас. Таким чином, загальна кількість перевезених пасажирів

$$f(x_1, x_2) = 576x_1 + 738x_2 \to \max .$$

При цьому використовується

плацкартних вагонів $5x_1 + 10x_2$;

купейних вагонів $7x_1 + 5x_2$;

м'яких вагонів $3x_1 + x_2$;

багажних вагонів x_2;

поштових вагонів $x_1 + x_2$.

Враховуючи обмеження на кількість вагонів у депо, математична модель задачі запишеться так

$$f(x_1, x_2) = 576x_1 + 738x_2 \to \max,$$

$$\begin{cases} 5x_1 + 10x_2 \le 70, \\ 7x_1 + 5x_2 \le 32, \\ 3x_1 + 1x_2 \le 15, \\ x_2 \le 8, \\ x_1 + x_2 \le 11, \end{cases}$$

$$x_1 \ge 0,\ x_2 \ge 0.$$

2.3. Графічний метод розв'язання задачі лінійного програмування

Геометрична інтерпретація задачі лінійного програмування особливо наочна у випадку двох змінних, оскільки лінії, які задають область допустимих розв'язків – прямі, а графіком функції мети теж є пряма. Розглянемо це на прикладі.

Приклад 2.2. Розв'язати графічно задачу лінійного програмування:

$$F = x_1 + 3x_2 \to \min\ (\max),$$

$$\begin{cases} x_1 + 2x_2 \le 10, \\ x_1 \le 6, \\ x_1 + x_2 \ge 2, \end{cases}$$

$$x_1 \ge 0,\ x_2 \ge 0.$$

Розв'язання.

1. Згідно з обмеженнями математичної моделі задачі будуємо допустиму область. Для цього всі нерівності обмежень подамо рівняннями та побудуємо прямі

$$x_1 + 2x_2 = 10,\ x_1 = 6,\ x_1 + x_2 = 2,\ x_1 = 0,\ x_2 = 0.$$

Кожна пряма поділяє площину на дві півплощини – допустиму та недопустиму. Щоб знайти допустиму півплощину потрібно взяти довільну точку в одній з півплощин (наприклад, початок координат) та підставити її координати у відповідне цій прямій обмеження. Якщо при цьому обмеження порушується, то півплощина, якій належить вибрана точка, є недопустимою, інакше – допустимою (допустиму область доцільно відмічати на малюнку, наприклад, штриховкою). Після побудови всіх допустимих півплощин знаходимо допустиму область, як перетин цих півплощин (рис. 2.1).

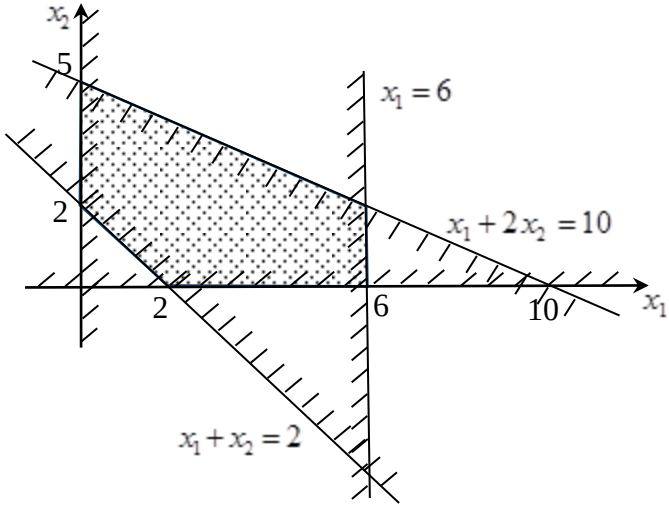


Рис 2.1. Побудова допустимої області

2. Будуємо пряму функції мети. Для цього знаходимо градієнт функції мети – вектор, координати якого дорівнюють коефіцієнтам, що стоять біля відповідних змінних функції мети

$$grad\ F = (1;3).$$

Перпендикулярно до градієнта будуємо пряму функції мети, наприклад, через початок координат (рис. 2.2).

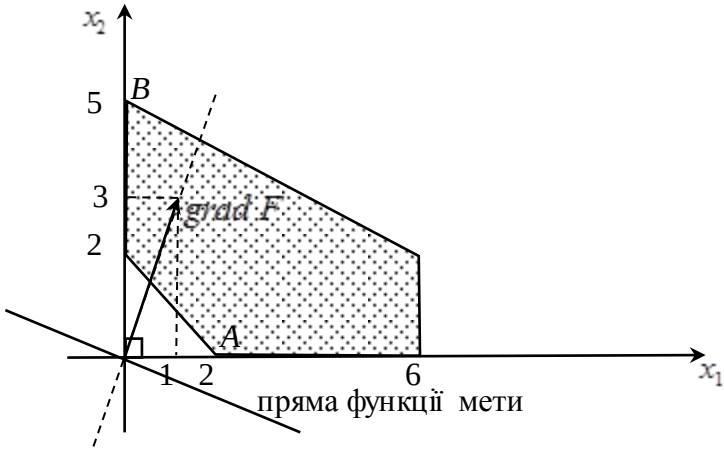


Рис 2.2. Побудова функції мети

3. Знаходимо оптимальні точки. Для цього пряму функції мети переміщуємо перпендикулярно до напрямку градієнта. Найближча точка дотику області допустимих розв'язків відповідає точці мінімуму – точка *A*, а найвіддаленіша відповідає точці максимуму – точка *B*.

4. Знаходимо координати оптимальних точок як точок перетину прямих: *A*(2;0), *B*(0;5).

5. Обчислюємо значення функції мети в оптимальних точках $F_{\min} = 2$, $F_{\max} = 15$.

Описаний метод називають **графічним методом розв'язання задач лінійного програмування**.

Зауваження. Якщо пряма функції мети паралельна одній з границь допустимої області, задача має множину оптимальних розв'язків.

Зауваження. Якщо допустима область – опуклий многокутник, то оптимальна точка задачі лінійного програмування є або кутовою, як у наведеному вище прикладі, або, якщо є декілька кутових оптимальних точок, то розв'язком є будь-яка точка відрізка, що сполучає оптимальні вершини.

Зауваження. Якщо допустима область порожня, то допустимого розв'язку немає. Якщо допустима область не обмежена, то оптимальний розв'язок може не існувати.

Зауваження. Графічний метод можна також застосовувати у випадку, коли різниця між кількістю змінних n та кількістю обмежень m задовольняє умову $n-m=2$.

Для розв'язування задачі лінійного програмування довільної розмірності частіше використовують **симплекс-метод** або метод послідовного поліпшення плану. Метод був розроблений американським математиком Джорджем Данцігом у 1947 році. Симплекс-метод – це алгоритм рішення задачі лінійного програмування шляхом перебору вершин опуклого багатогранника в багатовимірному просторі. Основна ідея симплекс-методу полягає в тому, що екстремум функції мети завжди досягається в кутових точках області допустимих рішень. Симплекс-метод реалізує перебір кутових точок області допустимих рішень у напрямку поліпшення значення функції мети.

2.4. Розв'язання задачі лінійного програмування в пакеті Maple

В системі ***Maple*** вбудовано пакет для розв'язання задач лінійного програмування ***simplex***, який базується на симплекс-методі.

Пакет ***simplex*** підключається командою

>with(simplex):

Для отримання довідки для цього пакета можна виконати команду

>help(simplex);

Основними функціями цього пакета є

***maximize*(*f, C, vartype*)**

та

***minimize*(*f, C, vartype*)**

з такими параметрами:

f – алгебраїчний вираз, лінійна функція мети;

C – система лінійних обмежень;

vartype – спеціальні опції команди (див. довідку ***Maple***).

Приклад 2.3. Розв'язати задачу лінійного програмування

$$x_1+3x_2\to\max,$$
$$\begin{cases} x_1+2x_2\le 10, \\ x_1\le 6, \\ x_1+x_2\ge 2, \end{cases}$$
$$x_1\ge 0,\ x_2\ge 0.$$

Розв'язання.

Розв'язання задачі в пакеті ***Maple*** наведено на рис. 2.3.

```
> restart;
```
Підключаємо пакет ***simplex***
```
> with(simplex) :
```
Задаємо функцію мети ***f*** та систему обмежень -- нерівностей ***ineq***
```
> f := x1 + 3·x2; ineq := {x1 + 2·x2 ≤ 10, x1 ≤ 6, x1 + x2 ≥ 2, 0 ≤ x1, 0 ≤ x2};
```
$$f := x_1 + 3\,x_2$$
$$ineq := \{0 \le x_1,\ 0 \le x_2,\ 2 \le x_1 + x_2,\ x_1 \le 6,\ x_1 + 2\,x_2 \le 10\}$$
Знаходимо максимум функції ***f*** при заданій системі обмежень -- нерівностей ***ineq***
```
> maximize(f, ineq);
```
$$\{x_1 = 0,\ x_2 = 5\}$$
Знаходимо максимальне значення функції ***f*** в оптимальній точці (0;5)
```
> assign(maximize(f, ineq)); f;
```
$$15$$

Рис. 2.3. Розв'язання задачі симплекс-методом в пакеті ***Maple***

Зауваження. За допомогою команди

***assign*(*t*)**

невідомим присвоюються знайдені значення t (t – послідовність, список чи множина рівностей вигляду *невідома = значення*).

Таким чином, функція набуває свого максимуму $f_{\max} = 15$ в точці (0;5).

Засобами ***Maple*** можна візуалізувати область допустимих розв'язків (рис. 2.4).

Для цього підключаємо пакет

>with(plots):

Для побудови області, яку обмежує система нерівностей, призначена функція

***inequal*(*ineqs, xspec, yspec, options*)**

з такими параметрами:

ineqs – система нерівностей;

xspec, ***yspec*** – діапазон змінних, для яких буде побудовано графік;

options – інші опції функції.

```
> restart;
> with(plots) :
> ineq := {x1 + 2·x2 ≤ 10, x1 ≤ 6, x1 + x2 ≥ 2, 0 ≤ x1, 0 ≤ x2};
                    ineq := {0 ≤ x1, 0 ≤ x2, 2 ≤ x1 + x2, x1 ≤ 6, x1 + 2 x2 ≤ 10}
> inequal(ineq, x1 = -1 ..8, x2 = -1 ..8);
```

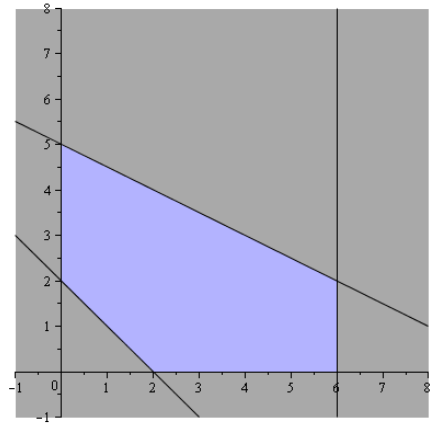


Рис. 2.4. Область допустимих розв’язків

Контрольні запитання

1. Які моделі розв’язуються методами лінійного програмування?
2. Сформулюйте задачу лінійного програмування у стандартній формі.
3. Наведіть приклади задач, моделі яких зводяться до моделей лінійного програмування.
4. Опишіть алгоритм графічного методу розв’язання задач лінійного програмування.
5. Сформулюйте основні положення симплекс-методу.

ТЕМА 3. ДВОЇСТА ЗАДАЧА ЛІНІЙНОГО ПРОГРАМУВАННЯ

3.1. Взаємно двоїсті задачі

Теорія двоїстості в лінійному програмуванні вивчає загальні властивості пари тісно пов'язаних між собою так званих двоїстих задач. Кожна задача лінійного програмування пов'язана з деякою іншою, також цілком визначеною задачею лінійного програмування. Їх зв'язок взаємний і настільки тісний, що розв'язування однієї фактично дає розв'язок і іншої. Таку пару задач називають *парою взаємно двоїстих задач лінійного програмування*. Наскільки важлива двоїстість свідчить той факт, що за її відкриття та застосування в оптимальному плануванні в 1975р. Л. В. Канторович отримав Нобелівську премію.

Для складання математичної моделі двоїстої задачі потрібно, щоб пряма задача мала таку структуру:

- знаки обмежень повинні мати одну спрямованість;
- якщо функція мети прямує до мінімуму, то нерівності повинні бути типу "$\geq$", а якщо до максимуму, то "$\leq$".

Таким чином, коли складаємо моделі двоїстої задачі, пряма задача повинна мати модель так званої *стандартної або канонічної форми*.

Зауваження. Якщо система обмежень прямої задачі має мішані знаки нерівностей, то треба помножити на (-1) деякі нерівності, щоб одержати одну спрямованість знаків нерівностей, причому вони повинні бути узгоджені з напрямком функції мети згідно із структурою стандартної форми.

Двоїсті задачі бувають двох типів:

- обмеження математичної моделі як прямої, так і двоїстої задачі записуються у вигляді нерівностей. Значення змінних таких задач не повинні бути від'ємними. Таку двоїсту пару задач називають *симетричною*;
- математична модель прямої задачі має обмеження у вигляді рівнянь та нерівностей, а двоїста задача має всі обмеження у

вигляді нерівностей. Значення змінних двоїстої задачі можуть бути від'ємними. Таку двоїсту задачу називають *несиметричною*.

3.2. Алгоритм перетворення

Побудову математичної моделі двоїстої задачі здійснюють за таким алгоритмом:

- замість змінних x_j, $j=\overline{1,n}$ прямої задачі вводять змінні y_i, $i=\overline{1,m}$ двоїстої задачі. Кількість змінних y_i дорівнює кількості обмежень прямої задачі;
- вільні члени b_i, $i=\overline{1,m}$ обмежень прямої задачі стають коефіцієнтами відповідних змінних функції мети двоїстої задачі;
- максимум (мінімум) функції мети прямої задачі змінюють на мінімум (максимум) функції мети двоїстої задачі;
- коефіцієнти a_{ij} з обмежень прямої задачі записують у матрицю $\| a_{ij} \|$, потім цю матрицю транспонують. Елементи матриці $\| a_{ij} \|^T$ є коефіцієнтами відповідних змінних y_i обмеження двоїстої задачі;
- коефіцієнти функції мети прямої задачі є вільними членами відповідних обмежень двоїстої задачі;
- знаки обмежень змінюються на протилежні.

Наведемо симетричну двоїсту пару:

Пряма задача	**Двоїста задача**
$L=\sum_{j=1}^{n} c_j x_j \to \max$	$W=\sum_{i=1}^{m} b_i y_i \to \min$
$\sum_{j=1}^{n} a_{ij} x_j \le b_i \quad (i=\overline{1,m}),$	$\sum_{i=1}^{m} a_{ij} y_i \ge c_j \quad (j=\overline{1,n}),$
$x_j \ge 0 \quad (j=\overline{1,n}).$	$y_i \ge 0 \quad (i=\overline{1,m}).$

Приклад 3.1.

Пряма задача:

$$L = 7x_1 + 5x_2 \to \max,$$
$$2x_1 + 3x_2 \le 18,$$
$$3x_1 + x_2 \le 12,$$
$$x_2 \le 5,$$
$$x_1 \le 3,$$
$$x_1 \ge 0, \quad x_2 \ge 0.$$

Двоїста задача:

$$W = 18y_1 + 12y_2 + 5y_3 + 3y_4 \to \min,$$
$$2y_1 + 3y_2 + y_4 \ge 7,$$
$$3y_1 + y_2 + y_3 \ge 5,$$
$$y_1 \ge 0, \quad y_2 \ge 0,$$
$$y_3 \ge 0, \quad y_4 \ge 0.$$

За пряму або двоїсту задачу можна вважати будь-яку з пари взаємно двоїстих задач.

Зауваження. Кількість обмежень прямої задачі дорівнює кількості змінних двоїстої задачі. І навпаки, кількість обмежень двоїстої задачі дорівнює кількості змінних прямої задачі, тобто кожній змінній однієї задачі відповідає одне обмеження другої задачі двоїстої пари.

3.3. Основні теореми двоїстості

Перша теорема двоїстості. *Якщо одна з пари двоїстих задач має розв'язок, то й інша задача має розв'язок. При цьому для довільних допустимих планів* $X = (x_1, x_2, ..., x_n)$ *та* $Y = (y_1, y_2, ..., y_m)$ *правильна нерівність:* $\sum_{j=1}^{n} c_j x_j \le \sum_{i=1}^{m} b_i y_i$, *яка переходить у рівність, коли X та Y – оптимальні плани відповідних задач, причому*

$$\max L = \min W.$$

Друга теорема двоїстості (для симетричної пари взаємно двоїстих задач). *Допустимий розв'язок двоїстої задачі є оптимальним, тоді і тільки тоді, коли виконуються умови:*

$$x_j(\sum_{i=1}^{m} a_{ij}y_i - c_j) = 0, \quad (j = \overline{1,n});$$

$$y_i(\sum_{j=1}^{n} a_{ij}x_j - b_i) = 0, \quad (i = \overline{1,m}).$$

Третя теорема двоїстості. *Значення змінних* y_i, $i = \overline{1,m}$ *оптимального розв'язку двоїстої задачі оцінюють вплив зміни вільних членів* b_i, $i = \overline{1,m}$ *на значення функції мети, тобто*

$$\Delta L = y_i \cdot \Delta b_i .$$

Слід відзначити, що для знаходження оптимального розв'язку доцільно вибрати ту із взаємно двоїстих задач, розв'язування якої менш трудомістке, а розв'язок другої задачі цієї пари знайти за допомогою теорем двоїстості.

3.4. Економічна інтерпретація двоїстої задачі

Поняття двоїстості розглянемо на прикладі задачі розподілу ресурсів.

Приклад 3.2. Нехай на підприємстві вирішили раціонально використовувати відходи основного виробництва. У плановому періоді з'явилися відходи сировини. З цих відходів, враховуючи спеціалізацію підприємства, можна налагодити випуск n видів неосновної продукції. Позначимо через a_{ij} норму витрати одиниць i-го ресурсу, що йде на виготовлення j-го товару, c_j – ціна реалізації одиниці j-го товару (реалізація забезпечена), b_i – величина запасу i-го ресурсу. Невідомі величини x_j – кількість одиниць j-го товару, що планується виготовити.

Потрібно скласти такий план виробництва товарів, щоб загальний прибуток від виробництва був найбільшим. Отримуємо таку математичну модель:

$$L = \sum_{j=1}^{n} c_j x_j \to \max,$$

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \quad (i = \overline{1,m}),$$

$$x_j \ge 0, \quad (j = \overline{1,n}).$$

Припустимо, що з самого початку вивчення питання про використання відходів основного виробництва на підприємстві показало можливість реалізації їх деякій організації. Необхідно встановити приблизні оцінки (ціни) на ці відходи. Позначимо їх y_i. Оцінки мають бути встановлені враховуючи вимоги, які відображають інтереси підприємства та організації, що не збігаються:

- загальну вартість відходів сировини, що купує організація, в її інтересах мінімізувати;
- підприємство погоджується продати відходи тільки за такими цінами, за яких воно отримає прибуток, не менший ніж воно могло б отримати, організувавши власне виробництво.

Ці вимоги формалізуються у вигляді такої задачі лінійного програмування

$$W = \sum_{i=1}^{m} b_i y_i \to \min,$$

$$\sum_{i=1}^{m} a_{ij} y_i \ge c_j, \quad (j = \overline{1,n}),$$

$$y_i \ge 0, \quad (i = \overline{1,m}).$$

Вимога 1. $W = \sum_{i=1}^{m} b_i y_i \to \min$ – мінімізація витрат на купівлю сировини організацією.

Вимога 2. Підприємство відмовиться від випуску кожної одиниці продукції першого виду, якщо $a_{11}y_1 + a_{21}y_2 + ... + a_{m1}y_m \geq c_j$, $j = \overline{1,n}$, де ліва частина означає виручку за сировину, яка витрачається на одиницю продукції першого виду, права – її ціну.

За змістом задачі оцінки y_i, $i = \overline{1,m}$ не повинні бути від'ємними. Змінні y_i, $i = \overline{1,m}$ називають **двоїстими оцінками** або **об'єктивно зумовленими оцінками**.

Таким чином, двоїста задача формулюється так: яку ціну призначити кожному ресурсу, щоб прибуток від продажу ресурсів, що витрачаються на виготовлення одиниці кожного товару, був не меншим ніж прибуток від реалізації одиниці товару і водночас загальна вартість усіх ресурсів була б мінімальною. Іншими словами, двоїсті оцінки y_i, $i = \overline{1,m}$, гарантують рентабельність оптимального плану, дозволяють збалансувати затрати і результати рішення.

Двоїсті оцінки є інструментом аналізу й прийняття рішень у випадках, коли постійно змінюються умови функціонування об'єкта моделювання. За допомогою двоїстих оцінок та теорем двоїстості можливо оцінити вплив зміни ресурсів на функцію мети, провести аналіз дефіцитності ресурсів, рентабельності виробництва продукції та доцільності розширення асортименту продукції.

Приклад 3.3. Нехай для випуску двох видів продукції на підприємстві використовують три види ресурсів: праця, сировина, обладнання. Обсяги виділених ресурсів, норми їх витрат і прибуток на одиницю продукції для виготовлення кожного виду продукції наведені в таблиці 3.1.

Таблиця 3.1

Ресурси	Затрати ресурсів на одиницю продукції		Наявність ресурсів
	А	Б	
Праця	2	4	2000
Сировина	4	1	1400
Обладнання	2	1	800
Прибуток на одиницю продукції	40	60	

Необхідно:

- сформулювати та розв'язати пряму оптимізаційну задачу, вказати оптимальну виробничу програму;
- сформулювати двоїсту задачу і знайти її оптимальний план;
- дати оцінку міри дефіцитності ресурсів і рентабельності продукції, що випускається.

Розв'язання.

Нехай x_1 та x_2 – кількості першого та другого видів продукції відповідно. Тоді математична модель задачі така:

$$L(x_1, x_2) = 40x_1 + 60x_2 \to \max,$$

$$2x_1 + 4x_2 \le 2000,$$

$$4x_1 + x_2 \le 1400,$$

$$2x_1 + x_2 \le 800,$$

$$x_1 \ge 0, \quad x_2 \ge 0.$$

Розв'яжемо задачу засобами пакета ***Maple***: оптимальний план виробництва $\overline{X} = (200, 400)$. В результаті отримуємо оптимальний прибуток $L_{\max} = 32000$ (рис. 3.1).

```
> restart;
```
Підключаємо пакет ***simplex***
```
> with(simplex) :
```
Задаємо функцію мети L та систему обмежень -- нерівностей ***ineq***
```
> L := 40·x1 + 60·x2; ineq := {2·x1 + 4·x2 ≤ 2000, 4·x1 + x2 ≤ 1400, 2·x1 + x2 ≤ 800, 0 ≤ x1, 0 ≤ x2};
```
$$L := 40\,x_1 + 60\,x_2$$

$$ineq := \{0 \le x_1,\ 0 \le x_2,\ 2\,x_1 + x_2 \le 800,\ 2\,x_1 + 4\,x_2 \le 2000,\ 4\,x_1 + x_2 \le 1400\}$$

Знаходимо максимум функції L при заданій системі обмежень -- нерівностей ***ineq***
```
> maximize(L, ineq);
```
$$\{x_1 = 200,\ x_2 = 400\}$$

Знаходимо максимальне значення функції L в оптимальній точці (200;400)
```
> assign(maximize(L, ineq)); L;
```
$$32000$$

Рис. 3.1. Розв'язання оптимізаційної задачі в середовищі ***Maple***

Модель двоїстої задачі має вигляд:

$$W(y_1, y_2, y_3) = 2000y_1 + 1400y_2 + 800y_3 \to \min,$$

$$2y_1 + 4y_2 + 2y_3 \ge 40,$$

$$4y_1 + y_2 + y_3 \ge 60,$$

$$y_1 \ge 0, \quad y_2 \ge 0, \quad y_3 \ge 0.$$

Оптимальний розв'язок двоїстої задачі $\overline{Y} = (\frac{40}{3}; 0; \frac{20}{3})$, $L(\overline{X}) = W(\overline{Y}) = 32000$ (рис. 3.2).

```
> restart;
```
Підключаємо пакет ***simplex***
```
> with(simplex) :
```
Задаємо функцію мети W та систему обмежень -- нерівностей ***ineq***
```
> W := 2000·y1 + 1400·y2 + 800·y3; ineq := {2·y1 + 4·y2 + 2·y3 ≥ 40, 4·y1 + y2 + y3
    ≥ 60, 0 ≤ y1, 0 ≤ y2, 0 ≤ y3};
```
$$W := 2000\,y_1 + 1400\,y_2 + 800\,y_3$$

$$ineq := \{0 \le y_1,\ 0 \le y_2,\ 0 \le y_3,\ 40 \le 2\,y_1 + 4\,y_2 + 2\,y_3,\ 60 \le 4\,y_1 + y_2 + y_3\}$$

Знаходимо мінімум функції W при заданій системі обмежень -- нерівностей ***ineq***
```
> minimize(W, ineq);
```
$$\left\{y_1 = \frac{40}{3},\ y_2 = 0,\ y_3 = \frac{20}{3}\right\}$$

Знаходимо мінімальне значення функції W в оптимальній точці (40/3;0;20/3)
```
> assign(minimize(W, ineq)); W;
```
$$32000$$

Рис. 3.2. Оптимальний розв'язок двоїстої задачі в середовищі ***Maple***

Зауваження. За допомогою команди

dual(F, ineq, y),

де ***F*** – цільова функція, для якої знаходимо **максимум**,
ineq – система обмежень-нерівностей,
y – змінні двоїстої задачі
програма ***Maple*** складе двоїсту задачу до цієї задачі (рис. 3.3).

```
> restart;
```
Підключаємо пакет ***simplex***
```
> with(simplex) :
```
Задаємо функцію мети ***L*** та систему обмежень -- нерівностей ***ineq***
```
> L := 40·x1 + 60·x2; ineq := {2·x1 + 4·x2 ≤ 2000, 4·x1 + x2 ≤ 1400, 2·x1 + x2 ≤ 800, 0 ≤ x1, 0 ≤ x2};
```
$$L := 40\,x_1 + 60\,x_2$$

$$ineq := \{0 \le x_1,\ 0 \le x_2,\ 2\,x_1 + x_2 \le 800,\ 2\,x_1 + 4\,x_2 \le 2000,\ 4\,x_1 + x_2 \le 1400\}$$

```
> dual(L, ineq, y);
```
$$800\,y3 + 2000\,y4 + 1400\,y5,\ \{40 \le -y1 + 2\,y3 + 2\,y4 + 4\,y5,\ 60 \le -y2 + y3 + 4\,y4 + y5\}$$

```
> minimize(dual(L, ineq, y), NONNEGATIVE)
```
$$\left\{y1 = 0,\ y2 = 0,\ y3 = \frac{20}{3},\ y4 = \frac{40}{3},\ y5 = 0\right\}$$

```
> assign(minimize(dual(L, ineq, y), NONNEGATIVE)); dual(L, ineq, y);
```
$$32000,\ \{40 \le 40,\ 60 \le 60\}$$

Знаходимо максимум функції ***L*** при заданій системі обмежень -- нерівностей ***ineq***
```
> maximize(L, ineq);
```
$$\{x_1 = 200,\ x_2 = 400\}$$

Знаходимо максимальне значення функції ***L*** в оптимальній точці (200;400)
```
> assign(maximize(L, ineq)); L;
```
$$32000$$

Рис. 3.3. Побудова двоїстої задачі в середовищі ***Maple***

Міра дефіциту ресурсів. В прикладі об'єктивно обумовлені оцінки ресурсів дорівнюють: «праця» $\frac{40}{3}$ ($y_1 = \frac{40}{3}$); «сировина» – 0 ($y_2 = 0$); «обладнання» – $\frac{20}{3}$ ($y_3 = \frac{20}{3}$).

Дефіцитний ресурс, повністю використовується в оптимальному плані $\left(\sum_{j=1}^{2} a_{ij}x_j = b_i\right)$, має додатну оцінку $(y_i > 0)$; недефіцитний ресурс (для якого $\sum_{j=1}^{2} a_{ij}x_j < b_i$), має нульову

оцінку $(y_i = 0)$. В нашому прикладі «сировина» не є дефіцитним ресурсом, оскільки

$4x_1 + x_2 \le 1400$,

$4 \cdot 200 + 400 = 1200 < 1400 = b_2$;

а «праця» і «обладнання» – дефіцитні ресурси:

$2x_1 + 4x_2 \le 2000$,

$2 \cdot 200 + 4 \cdot 400 = 2000 = b_1,\ y_1 = \frac{40}{3}$;

$2x_1 + x_2 \le 800$,

$2 \cdot 200 + 400 = 800 = b_3,\ y_3 = \frac{20}{3}$.

Чим вища величина оцінки y_i, тим гостріша дефіцитність *i*-го ресурсу.

В прикладі «праця» більш дефіцитна, ніж «обладнання»: $\frac{40}{3} > \frac{20}{3}$. Таким чином, найбільш вигідним є збільшення обсягів ресурсу праці.

Чутливість розв'язку до зміни запасів ресурсів. За допомогою двоїстих оцінок y_i можна проаналізувати зміну функції мети залежно від зміни обсягів початкових ресурсів. Для цього використовують третю теорему двоїстості: $\Delta L = y_i \cdot \Delta b_i$.

Припустимо, що запас ресурсу «праця» змінився на одиницю і становить 2001 одиниць. Ця зміна запасу викликає приріст прибутку на $\Delta L = y_1 \cdot \Delta b_1 = \frac{40}{3} \cdot 1 = \frac{40}{3}$.

Збільшення кількості обладнання на одиницю дає приріст прибутку на $\Delta L = y_3 \cdot \Delta b_3 = \frac{20}{3} \cdot 1 = \frac{20}{3}$.

Збільшення запасів ресурсу «сировина» не впливає на збільшення прибутку, оскільки $y_2 = 0$.

Зауваження. За допомогою теорем теорії двоїстості можна:

- визначати інтервали зміни запасів ресурсів, в яких план двоїстої задачі не змінюється;
- проаналізувати чутливість рішення до зміни коефіцієнтів функції мети;
- проаналізувати рентабельність виготовлення продукції та доцільність включення до плану нових виробів.

Контрольні запитання

1. Назвіть пункти алгоритму перетворення прямої задачі у двоїсту.
2. Сформулюйте основні теореми двоїстості.
3. Сформулюйте економічний зміст пари двоїстих задач.
4. Як визначити міру дефіциту ресурсів та вплив зміни запасів ресурсів на значення функції мети?

ТЕМА 4. ТРАНСПОРТНА ЗАДАЧА

4.1. Транспортна задача як задача лінійного програмування

Одним із типів задач лінійного програмування є лінійна транспортна задача. Наведемо класичну постановку транспортної задачі: потрібно перевезти товар з m пунктів відправлення до n пунктів призначення. Кожен пункт відправлення містить товар в кількості b_i, $i=\overline{1,m}$. Кожен пункт призначення може прийняти товар у кількості a_j, $j=\overline{1,n}$. Вартість перевезення одиниці продукції з i-го пункту відправлення до j-го пункту призначення становить p_{ij}. Вважаємо, що транспортні витрати лінійно залежать від кількості перевезеного товару. Розв'язком транспортної задачі є план перевезень за якого:

- сумарна вартість перевезень є мінімальною;
- потреби всіх пунктів призначення забезпечені;
- з кожного пункту відправлення вивезено весь товар.

Кількість товару, перевезена з i-го пункту відправлення до j-го пункту призначення, є невідомою і її позначають змінною x_{ij}.

Лінійну транспортну задачу можна записати у вигляді

$$F=\sum_{i=1}^{m}\sum_{j=1}^{n}p_{ij}x_{ij}\to\min, \tag{4.1}$$

$$\begin{aligned}&\sum_{i=1}^{m}x_{ij}=a_j,\\&\sum_{j=1}^{n}x_{ij}=b_i,\end{aligned} \tag{4.2}$$

$$x_{ij}\geq 0,\ i=\overline{1,m},\ j=\overline{1,n}.$$

Зауваження. Для розв'язку транспортної задачі необхідне виконання умови рівності кількості товару у пунктах відправлення та кількості необхідного товару в пунктах призначення, тобто $\sum_{i=1}^{m} b_i = \sum_{j=1}^{n} a_j$. Якщо кількість товару в пунктах відправлення менша за необхідну кількість, тобто $\sum_{i=1}^{m} b_i < \sum_{j=1}^{n} a_j$, то задача не має розв'язку. Якщо ж $\sum_{i=1}^{m} b_i \geq \sum_{j=1}^{n} a_j$, то це означає, що товару в пунктах відправлення більше, ніж необхідно. Задача буде розв'язана, якщо ввести "фіктивний" пункт призначення, який може прийняти надлишки. В цьому випадку вартість перевезення з будь-якого пункту відправлення до фіктивного пункту дорівнює 0.

Транспортну задачу найзручніше задавати у вигляді транспортної таблиці (рис. 4.1). Початковими даними є числа a_j, b_i, p_{ij}. Під час розв'язування у транспортну таблицю потрібно вписати значення x_{ij}.

		Пункти призначення (споживачі)			
		a_1	a_2	…	a_n
Пункти відправлення (склади)	b_1	p_{11} x_{11}	p_{12} x_{12}	… …	p_{1n} x_{1n}
	b_2	p_{21} x_{21}	p_{22} x_{22}	… …	p_{2n} x_{2n}
	…	… …	… …	… …	… …
	b_m	p_{m1} x_{m1}	p_{m2} x_{m2}	… …	p_{mn} x_{mn}

Рис. 4.1. Транспортна таблиця

4.2. Розв'язання транспортної задачі. Метод потенціалів

Транспортну задачу можна розв'язувати симплекс-методом. Однак, велика кількість змінних ($m \times n$) робить застосування симплекс-методу громіздким та нераціональним. Для розв'язання транспортної задачі існують більш ефективні методи пошуку оптимального рішення, зокрема *метод потенціалів*.

Для застосування методу потенціалів необхідно мати деякий початковий допустимий розв'язок. Його можна знайти за допомогою *правила північно-західного кута*. Розглянемо це правило на такому прикладі.

Приклад 4.1. Для заданої транспортної задачі (рис. 4.2) за правилом північно-західного кута знайти допустимий розв'язок.

		Пункти призначення (споживачі)			
		15	**5**	**10**	**15**
Пункти відправлення (склади)	**10**	**4**	**3**	**6**	**1**
	15	**9**	**7**	**4**	**5**
	20	**2**	**8**	**3**	**2**

Рис. 4.2. Транспортна таблиця прикладу 4.1

Розв'язання.

Спочатку вибираємо верхню ліву клітинку (рис. 4.3). У ній має бути записана кількість товару, перевезеного зі складу № 1 споживачу № 1. В цьому складі знаходиться 10 одиниць товару,

а перший споживач може прийняти 15 одиниць. Тому весь товар з першого складу передаємо першому споживачеві. В цьому складі залишається 0 одиниць (записуємо ліворуч в першому рядку), в клітинку першого рядка та першого стовпчика записуємо 10, решта клітинок першого рядка викреслюємо, а першому споживачу бракує ще 5 одиниць (записуємо вище в першому стовпчику)

				0		0	
				5	0	5	0
				15	**5**	**10**	**15**
		0	**10**	**4** **10**	**3** –	**6** –	**1** –
0	5	10	**15**	**9** **5**	**7** **5**	**4** **5**	**5** –
	0	15	**20**	**2** –	**8** –	**3** **5**	**2** **15**

Рис. 4.3. Розрахункова таблиця методу північно-західного кута

Цих 5 одиниць для першого споживача можемо передати зі складу № 2, на що вкажемо, заповнивши клітинку числом 5 у другому рядку та першому стовпчику (у складі № 2 тоді залишиться 10 одиниць, що відзначаємо в другому рядку ліворуч). Перший споживач вже забезпечений товаром, це відзначаємо вище в першому стовпчику 0, а решту незаповнених клітинок першого стовпчика викреслюємо. У другому складі залишається ще 10 одиниць товару, з яких максимально можливу кількість – 5 передаємо споживачу № 2 (записуємо 5 у другому рядку та другому стовпчику). Таким чином споживач № 2 вже забезпечений товаром (відзначаємо вище в другому

стовпчику 0 та викреслюємо решту клітинок). Залишок, а це 5 одиниць з другого складу передаємо споживачу № 3, про що робимо відмітки у відповідних рядках та стовпчиках. Споживач № 3 внаслідок цього ще не повністю забезпечений. Тому йому передають товар у кількості 5 одиниць зі складу № 3. Після цього залишок на складі № 3, а саме 15 одиниць, повністю забезпечує потреби споживача № 4. Таким чином, ми побудували допустимий розв'язок, знайшовши значення допустимих змінних $x_{11} = 10$, $x_{21} = 5$, $x_{22} = 5$, $x_{23} = 5$, $x_{33} = 5$, $x_{34} = 15$, решта змінних мають нульові значення.

Зауважимо, що застосовуючи правило північно-західного кута, ми не враховуємо вартості перевезень (числа у правих верхніх кутах клітинок транспортної таблиці). Але допустимий розв'язок транспортної задачі доцільніше шукати з урахуванням цих вартостей. Це дасть змогу прискорити процес розв'язку транспортної задачі, зменшивши кількість його кроків.

Знайдемо допустимий розв'язок *методом мінімальних вартостей* (рис. 4.4). Для цього виберемо клітинку з найменшою вартістю перевезень $\min\{p_{ij}\} = p_{14} = 1$. Здійснимо перевезення зі складу № 1 до споживача № 4. Ми можемо перевезти 10 одиниць продукції, після чого склад № 1 стане порожнім. Викреслимо перший рядок. Після цього другу клітинку з мінімальною вартістю перевезень вибираємо з частини таблиці, яка залишилася. Такою вартістю будуть $p_{34} = p_{31} = 2$. Виберемо, наприклад, третій склад і четвертого споживача. Останній вже буде забезпечений товаром, тому викреслюємо четвертий стовпчик. У таблиці, що залишилася, найменшу вартість буде мати перевезення зі складу № 3 споживачу № 1 ($p_{31} = 2$). Тому перевозимо 15 одиниць і викреслюємо стовпчик № 1 та рядок № 3. Наступним перевезенням з найменшою вартістю є перевезення зі складу № 2 до споживача № 3 ($p_{23} = 4$). Тут перевозимо 10 одиниць. І, нарешті, перевозимо товар зі складу № 2 до споживача № 2. Це

останнє перевезення в кількості 5 одиниць. В результаті ми заповнили таблицю значеннями, які вказують скільки потрібно перевезти одиниць, звідки і куди.

											0
							0		0		5
				15		**5**		**10**		**15**	
		0	**10**	–	**4**	–	**3**	–	**6**	**10**	**1**
0	5	10	**15**	–	**9**	**5**	**7**	**10**	**4**	–	**5**
	0	15	**20**	**15**	**2**	–	**8**	–	**3**	**5**	**2**

Рис. 4.4. Розрахункова таблиця методу мінімальних вартостей

Обчислимо вартість планів перевезення, знайдених за допомогою правила північно-західного кута та методу мінімальних вартостей:

$$F_{\text{півн.-зах.}} = 10\cdot4+5\cdot9+5\cdot7+5\cdot4+5\cdot3+15\cdot2 = 185,$$

$$F_{\text{мін. варт.}} = 10\cdot1+5\cdot7+10\cdot4+15\cdot2+5\cdot2 = 125.$$

В другому випадку сумарна вартість транспортних перевезень є меншою. Але чи є знайдене рішення оптимальним? Для знаходження оптимального плану перевезень використовують *метод потенціалів*.

Нехай маємо деяку транспортну таблицю (рис. 4.5) з допустимим розв'язком. Клітинки зі заповненою кількістю перевезень називаємо *базисними клітинками*.

	a_1	a_2	…	a_n	u_i
b_1	p_{11} x_{11}	p_{12} x_{12}		p_{1n} x_{1n}	u_1
…					…
b_m	p_{m1} x_{m1}	p_{m2} x_{m2}		p_{mn} x_{mn}	u_n
v_j	v_1	v_2	…	v_m	

Рис. 4.5. Транспортна таблиця з допустимим розв'язком

Доповнимо транспортну таблицю рядком та стовпчиком, де запишемо значення *потенціалів* – чисел u_i та v_j таких, щоб для базисних клітинок виконувалася рівність

$$p_{ij} = u_i + v_j .$$

Отримаємо систему лінійних алгебраїчних рівнянь, яка має $m+n$ невідомих. Матриця системи може бути зведена до трикутного вигляду з рангом $m+n-1$. Для її розв'язку одну із змінних вибирають рівною 0, а решту – послідовно знаходять.

Після знаходження потенціалів для небазисних клітинок обчислюємо значення

$$\Delta_{ij} = p_{ij} - u_i - v_j .$$

Якщо всі $\Delta_{ij} \geq 0$, тоді розв'язок оптимальний.

Якщо існують $\Delta_{ij} < 0$, тоді вибираємо клітинку з найменшим від'ємним значенням Δ_{ij} та позначаємо її знаком «+». Змінна, що відповідає цій клітинці, повинна ввійти в базис. Крім неї позначимо знаками «+» та «–» базисні клітинки, так,

щоб утворився цикл, а знаки «+» та «–» чергувались під час обходу цього циклу.

Знаходимо значення M – мінімум чисел x_{ij}, позначених знаком «–». До значень x_{ij}, клітинок, позначених знаком «+» додаємо число M, від значень x_{ij}, клітинок, позначених знаком «–» віднімаємо число M. Клітинку, що відповідає мінімуму чисел x_{ij}, позначених знаком «–» (тобто $x_{ij} = M$) залишаємо порожньою. Отримаємо нову таблицю.

Після цього повторюємо все спочатку, доки не отримаємо оптимальний розв'язок.

Приклад 4.2. Знайти оптимальний розв'язок транспортної задачі, записаної у вигляді таблиці (рис. 4.6).

	8	**6**	**20**	**8**
12	**2**	**8**	**1**	**6**
14	**4**	**4**	**2**	**3**
16	**1**	**5**	**7**	**9**

Рис. 4.6. Транспортна таблиця прикладу 4.2

Розв'язання.

Знайдемо допустимий розв'язок методом мінімальних вартостей (рис. 4.7).

	8	**6**	**20**	**8**
12	**2**	**8**	**1** **12**	**6**
14	**4**	**4**	**2** **8**	**3** **6**
16	**1** **8**	**5** **6**	**7**	**9** **2**

Рис. 4.7. Розрахункова таблиця методу мінімальних вартостей

Запишемо значення *потенціалів* (рис. 4.8). Для цього один із потенціалів (наприклад, u_1) покладаємо рівним 0, а решта послідовно знаходимо.

	8	**6**	**20**	**8**	u_i
12	**2**	**8**	**1** **12**	**6**	0
14	**4**	**4**	**2** **8** –	**3** **6** +	1
16	**1** **8**	**5** **6**	**7** +	**9** **2** –	7
v_j	–6	–2	1	2	

Рис. 4.8. Таблиця значень потенціалів

Для незаповнених клітинок обчислимо значення Δ_{ij} .

Оскільки є від'ємні Δ_{ij} ($\Delta_{33} = 7 - 7 - 1 = -1$), то розв'язок неоптимальний. Позначимо клітинку, що стоїть на перетині третього рядка та третього стовпця, знаком "+". Змінна, що відповідає цій клітинці, повинна ввійти в базис. Крім неї позначимо знаками "+" та "–" базисні клітинки так, щоб утворився цикл, а знаки чергувались в процесі обходу цього циклу (рис. 4.8). Знаходимо мінімум чисел x_{ij}, позначених знаком "–". Це число $M = 2$. Модифікуємо транспортну таблицю (рис. 4.9). Клітинку з числом M залишаємо незаповненою. Для інших клітинок, в яких стоїть знак "–", число M віднімаємо від записаного там числа. В клітинках, позначених знаком "+", число M додаємо до значення клітинки. Решту значень переписуємо без зміни.

Для нової таблиці шукаємо потенціали та обчислюємо значення Δ_{ij}. Оскільки всі значення Δ_{ij} додатні, то знайдений розв'язок є оптимальним.

	8	**6**	**20**	**8**	u_i
12	**2**	**8**	**1** **12**	**6**	0
14	**4**	**4**	**2** **6**	**3** **8**	1
16	**1** **8**	**5** **6**	**7** **2**	**9**	6
v_j	–5	–1	1	2	

Рис. 4.9. Модифікована транспортна таблиця

Таким чином, оптимальним буде перевезення, за якого кількості товару з i-го складу до j-го споживача будуть визначатися кількостями $x_{13} = 12$, $x_{23} = 6$, $x_{24} = 8$, $x_{31} = 8$, $x_{32} = 6$,

$x_{33} = 2$. В такому випадку вартість перевезення становить $F = 12 \cdot 1 + 6 \cdot 2 + 8 \cdot 3 + 8 \cdot 1 + 6 \cdot 5 + 2 \cdot 7 = 100$.

4.3. Розв'язання транспортної задачі засобами Maple

Транспортна задача в ***Maple*** розв'язується як задача лінійного програмування (рис. 4.10). Розв'яжемо засобами ***Maple*** приклад 4.2.

```
> restart:
```

Підключаємо пакет ***simplex***

```
> with(simplex):
```

Задаємо кількість товару в пунктах відправлення ***a***, в пунктах призначення ***b***, вартості перевезень ***p*** та описуємо матрицю невідомих ***x***

```
> a := [12, 14, 16]; b := [8, 6, 20, 8]; p := [2 8 1 6, 4 4 2 3, 1 5 7 9];
  x := [m1,1 m1,2 m1,3 m1,4, m2,1 m2,2 m2,3 m2,4, m3,1 m3,2 m3,3 m3,4];
```

$$a := \begin{bmatrix} 12 \\ 14 \\ 16 \end{bmatrix}$$

$$b := \begin{bmatrix} 8 \\ 6 \\ 20 \\ 8 \end{bmatrix}$$

$$p := \begin{bmatrix} 2 & 8 & 1 & 6 \\ 4 & 4 & 2 & 3 \\ 1 & 5 & 7 & 9 \end{bmatrix}$$

$$x := \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} \end{bmatrix}$$

Перевіряємо рівність обсягу попиту та пропозиції

```
> sum('a_i', i=1..3) = sum('b_j', j=1..4);
```

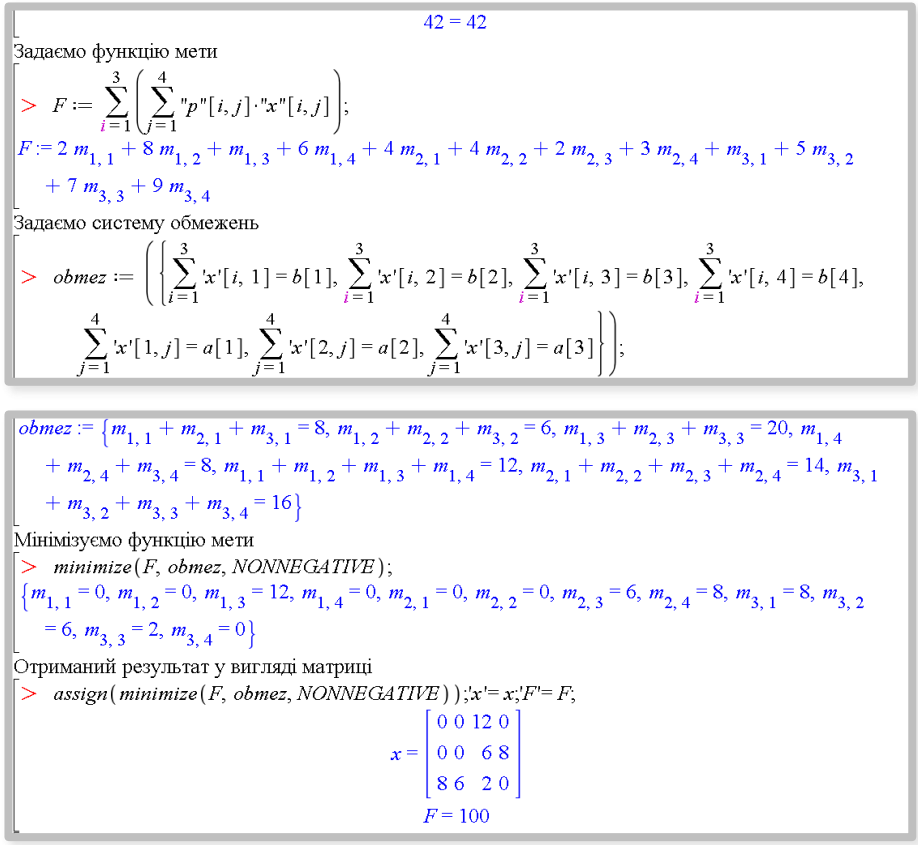


Рис. 4.10. Розв’язування транспортної задачі в пакеті Maple

Контрольні запитання

1. Постановка транспортної задачі. Яким умовам задовольняє оптимальний план перевезень транспортної задачі?
2. Сформулюйте транспортну задачу як задачу лінійного програмування.

3. Опишіть метод північно-західного кута та метод мінімальних вартостей. Який розв'язок транспортної задачі можна знайти цими методами?
4. Опишіть метод потенціалів для знаходження розв'язку транспортної задачі.

ТЕМА 5. МОДИФІКАЦІЯ ТРАНСПОРТНОЇ ЗАДАЧІ

5.1. Транспортна задача з додатковими обмеженнями на вивезення

Нехай в транспортній задачі пунктами відправлення є склади, які розбиті на групи так, що кожна група обслуговується певним заводом. Тому загальна кількість продукції, що може зберігатись на цих складах, обмежена продуктивністю заводу. Тоді обмеження транспортної задачі записуються так:

$$K = \sum_{i=1}^{m}\sum_{j=1}^{n} p_{ij}x_{ij} \to \min,$$

$$\sum_{i=1}^{m} x_{ij} = a_j, \qquad \sum_{j=1}^{n} x_{ij} = b_i,$$

$$\sum_{i\in I_k}\sum_{j=1}^{n} x_{ij} \le S_k,$$

$$x_{ij} \ge 0,\ i=\overline{1,m},\ j=\overline{1,n},\ k=\overline{1,r},$$

де k – номер заводу, r – кількість заводів, I_k – множина номерів складів, які обслуговуються k-тим заводом, S_k – продуктивність k-го заводу.

Задачу мінімізації загальних витрат на перевезення за наведених обмежень називають **транспортною задачею з додатковими обмеженнями на вивезення продукції**.

Зрозуміло, що продуктивність заводу необхідно враховувати тоді, коли вона менша, ніж місткість складів, що обслуговуються цим заводом. Це відповідає нерівності $\sum_{i\in I_k} b_i \ge S_k$. Якщо для деяких заводів ця нерівність не виконується, то умову $\sum_{i\in I_k}\sum_{j=1}^{n} x_{ij} \le S_k$ записувати не треба,

оскільки вона буде автоматично виконана, якщо виконується умова $\sum_{j=1}^{n} x_{ij} = b_i$.

5.2. Транспортна задача з обмеженою пропускною здатністю

На практиці фізичні умови накладають додаткові вимоги на обсяг перевезень. Такою умовою може бути обмежена пропускна здатність маршруту: з пункту відправлення i в пункт призначення j можна ввезти таку кількість продукції, що не перевищує заданого значення d_{ij}. Тоді обмеження транспортної задачі записуються так

$$\sum_{j=1}^{n} x_{ij} = b_i, \quad \sum_{i=1}^{m} x_{ij} = a_j,$$

$$d_{ij} \geq x_{ij} \geq 0,\ i = \overline{1,m},\ j = \overline{1,n},$$

де d_{ij} – задані невід'ємні числа.

Задачу мінімізації загальних витрат на перевезення за наведених обмежень називають **транспортною задачею з обмеженою пропускною здатністю**.

Для того, щоб транспортна задача з верхнім обмеженням пропускної здатності мала розв'язок, необхідно, щоб числа d_{ij} задовольняли такі умови:

$$\sum_{j=1}^{n} d_{ij} \geq b_i,\ i = \overline{1,m},$$

$$\sum_{i=1}^{m} d_{ij} \geq a_j,\ j = \overline{1,n}.$$

Звичайна транспортна задача може розглядатись як частковий випадок транспортної задачі з обмеженою пропускною здатністю за умови $d_{ij} = \infty$.

Крім верхнього обмеження, пропускна здатність може бути обмеженою знизу, тобто обмеження транспортної задачі записуються так:

$$\sum_{j=1}^{n} x_{ij} = b_i, \quad \sum_{i=1}^{m} x_{ij} = a_j,$$

$$x_{ij} \geq d_{ij} \geq 0, \; i = \overline{1,m}, \; j = \overline{1,n},$$

де d_{ij} – задані невід'ємні числа.

Для того, щоб транспортна задача з нижнім обмеженням пропускної здатності мала розв'язок, необхідно, щоб числа задовольняли таким умовам:

$$\sum_{j=1}^{n} d_{ij} \leq b_i, \; i = \overline{1,m},$$

$$\sum_{i=1}^{m} d_{ij} \leq a_j, \; j = \overline{1,n}.$$

Інколи фізичні умови не дають змоги організувати перевезення за певним маршрутом (відсутність дороги, або небажання *j*-го споживача приймати продукцію *i*-го відправника). Таку задачу можна розглядати як частковий випадок транспортної задачі з обмеженою пропускною здатністю за умови $d_{ij} = 0$. Тоді в системі обмежень частину нерівностей $x_{ij} \geq 0$ потрібно замінити на рівності $x_{ij} = 0$.

Якщо умовами перевезень передбачається перевезення фіксованого обсягу продукції d_{ij} від *i*-го відправника до *j*-го споживача, то в систему обмежень вводиться нове обмеження: $x_{ij} = d_{ij}$.

5.3. Транспортна задача з проміжними пунктами

Транспортна модель з проміжними пунктами відповідає реальній ситуації, коли між початковими та кінцевими пунктами перевезень є проміжні пункти для зберігання вантажів (транзитні пункти). Ця модель є більш загальною, ніж звичайна транспортна, в якій перевезення здійснюються безпосередньо між пунктами відправлення та призначення.

Розглянемо транспортну модель з двома пунктами відправлення ($P1$ та $P2$), двома транзитними пунктами ($T1$ та $T2$) і трьома пунктами призначення ($D1$, $D2$ та $D3$). Можливі напрямки транспортування продукції наведено на рис. 5.1.

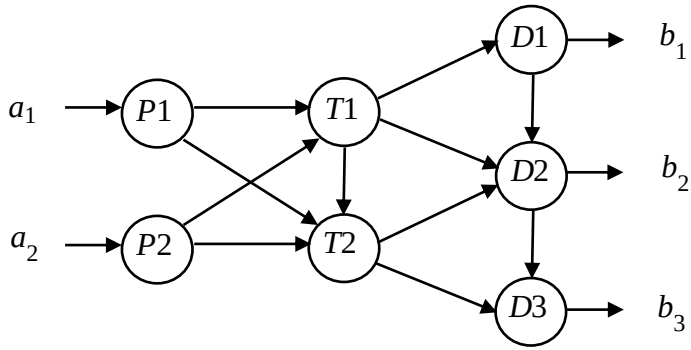


Рис. 5.1. Можливі напрями транспортування

В транспортній моделі з проміжними пунктами транзитні перевезення можуть здійснюватись через будь-які пункти мережі (відповідно до напрямків дуг на схемі), в тому числі через деякі пункти відправлення і призначення. Пункти, яким відповідають як вхідні, так і вихідні дуги називають **транзитними** (на рис. 5.1 – пункти $T1$, $T2$, $D1$ та $D2$). Решта пунктів є або **істинними пунктами відправлення** (на рис. 5.1 – пункти $P1$ та $P2$), або **істинними пунктами призначення** (на рис. 5.1 – пункт $D3$).

Таку модель можна звести до звичайної транспортної задачі, пунктами відправлення в якій є істинні пункти відправлення та транзитні пункти ($P1$, $P2$, $T1$, $T2$, $D1$, $D2$), а пунктами призначення – транзитні пункти та істинні пункти призначення ($T1$, $T2$, $D1$, $D2$, $D3$).

Обсяги попиту та пропозиції, що відповідають цим пунктам, обчислюються за такими правилами:

Обсяг пропозиції істинного пункту відправлення = обсягу вихідної пропозиції;

Обсяг пропозиції транзитного пункту = обсягу вихідної пропозиції + обсяг буфера;

Обсяг попиту істинного пункту призначення = обсягу вихідного попиту.

Обсяг буфера повинен бути таким, щоб вмістити в себе обсяг всієї пропозиції (або всього попиту), тобто

$$B = \sum_{i=1}^{m} b_i = \sum_{j=1}^{n} a_j \ .$$

Вартість перевезення з пункту відправлення i до пункту призначення j (якщо між пунктами існує зв'язок) становить p_{ij}. Вартість перевезення в межах одного пункту приймається нульовою.

Приклад 5.1. На рис. 5.2. зображено транспортну мережу перевезення автомобілів між трьома автомобільними заводами (пункти 1, 2 та 3) та трьома дилерами (пункти 6, 7 та 8) через два розподільчі центри (пункти 4 та 5). Вартість перевезення (в сотнях у. о.) наведено на схемі біля відповідних дуг.

Знайти найдешевший план перевезення автомобілів.

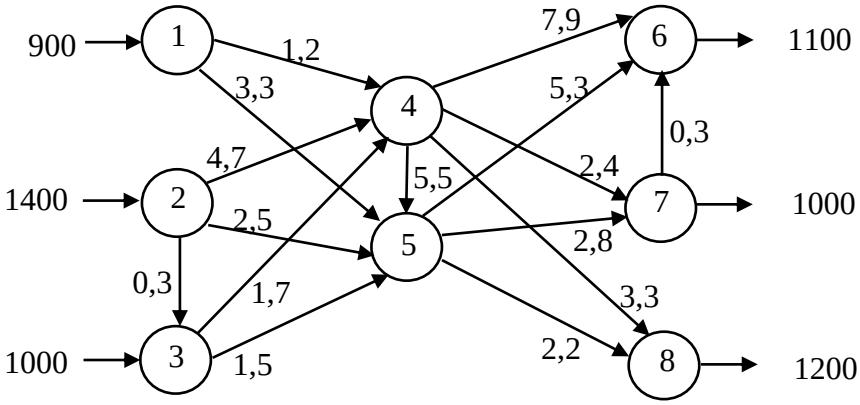


Рис. 5.2. Транспортна мережа між трьома заводами

Розв'язання.

Транспортну задачу, еквівалентну вихідній задачі, наведено в таблиці 5.1. Пунктами відправлення є пункти 1, 2, 3, 4, 5 та 7; пунктами призначення – пункти 3, 4, 5, 6, 7 та 8. Істинними пунктами відправлення є пункти 1 та 2, істинними пунктами призначення – пункти 6 та 8. Всі решта пункти є транзитними.

Обсяг буфера становить: $B = 900 + 1400 + 1000 = 3300$.

Таблиця 5.1

			Пункт призначення					
		Обсяг пропозиції	3	4	5	6	7	8
Обсяг попиту		**16500**	**3300**	**3300**	**3300**	**1100**	**4300**	**1200**
Пункт відправлення	1	**900**	∞	1,2	3,3	∞	∞	∞
	2	**1400**	0,3	4,7	2,5	∞	∞	∞
	3	**4300**	0	1,7	1,5	∞	∞	∞
	4	**3300**	∞	0	5,5	7,9	2,4	3,3
	5	**3300**	∞	∞	0	5,3	2,8	2,2
	7	**3300**	∞	∞	∞	0,3	0	∞

Розв'язок цієї задачі, отриманий з допомогою ***Maple***, наведено на рис. 5.3. Загальна вартість перевезень становить 13350 у. о.

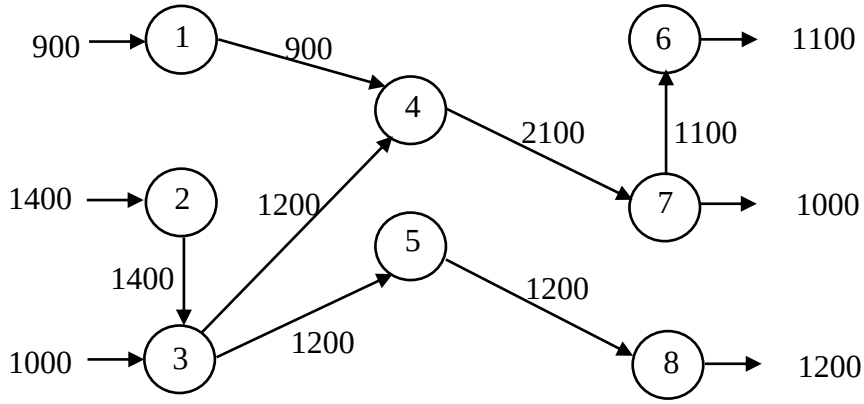


Рис. 5.3.Розв'язок транспортної задачі

З рис. 5.3 видно «транзитний» ефект розв'язку: з пункту 3 відправляється 2400 автомобілів (1000 власних і 1400,

отриманих з пункту 2). Аналогічно пункт 7 отримує 2100 автомобілів, з яких собі залишає 1000, а 1100 відправляє в пункт 6.

5.4. Відкрита модель транспортної задачі

Нехай у транспортній задачі відсутні обмеження на попит $\sum_{j=1}^{n} a_j$ або на пропозицію $\sum_{i=1}^{m} b_i$. Це означає, що будь-який пункт призначення може взяти всю суму наявних у пунктах відправлення товарів, або навпаки, будь-який з пунктів відправлення може задовольнити попит всіх пунктів призначення. Для розв'язання задачі у випадку, коли відсутні обмеження на пропозицію, обмеження попиту a_j в кожному стовпчику таблиці переносяться в клітинки з мінімальною вартістю перевезення p_{ij}. За відсутності обмежень щодо попиту в клітинки з мінімальною вартістю перевезення в кожному рядку таблиці переносяться обмеження пропозиції b_i. Отримане рішення завжди буде оптимальним.

Приклад 5.2. Щоб задовольнити потреби в будівельному піску п'яти споживачів, його можна добувати в необмеженій кількості в будь-якій з трьох точок: b_1, b_2, b_3. Попит становить: a_1 – 300, a_2 – 200, a_3 – 400, a_4 – 250 і a_5 – 400 т на добу. Відстані між споживачами та можливими місцями видобутку піску вказані в правих верхніх кутах транспортної таблиці 5.2.

Скільки в кожній точці треба добувати піску і як спланувати вантажопотоки, щоб сумарна відстань була мінімальною?

Таблиця 5.2

	300	**200**	**400**	**250**	**400**
b_1	**2**	**4**	**6**	**3**	**4**
b_2	**5**	**3**	**8**	**5**	**4**
b_3	**3**	**1**	**4**	**5**	**3**

Розв'язання.

Переносимо обмеження попиту за стовпцями в клітки з мінімальними відстанями x_{11}, x_{32}, x_{33}, x_{14}, x_{35}. Таким чином, ці клітки отримають завантаження в 300, 200, 400, 250 і 400 т відповідно.

Таблиця 5.3

	300	**200**	**400**	**250**	**400**
b_1	**2** 300	**4**	**6**	**3** 250	**4**
b_2	**5**	**3**	**8**	**5**	**4**
b_3	**3**	**1** 200	**4** 400	**5**	**3** 400

Отриманий в таблиці 5.3 розв'язок є оптимальним. При цьому рішенні видобуток піску в точках b_1 і b_3 повинен дорівнювати відповідно $300 + 250 = 550$ та $200 + 400 + 400 = 1000$ т. У точці b_2 видобуток вести нераціонально.

Відкрита модель транспортної задачі використовується для вирішення цілого ряду завдань: визначення оптимального розміщення та потужності автотранспортних підприємств, закріплення маршрутів за АТП, розміщення та місткість складів тощо.

Приклад 5.3. Для перевезення вантажу клієнтам a_1, a_2, a_3, a_4 потрібно відповідно 100, 70, 120 та 50 одиниць рухомого складу. Парк рухомого складу можна розмістити в трьох пунктах b_1, b_2, b_3. Крім цього в кожному з пунктів можна розмістити весь рухомий склад, необхідний для обслуговування клієнтів. Відстані між клієнтами та можливими пунктами розміщення рухомого складу наведені в таблиці 5.4. Скласти план оптимального перевезення.

Таблиця 5.4

	100	**70**	**120**	**50**
b_1	**3**	**2**	**4**	**1**
b_2	**2**	**3**	**5**	**4**
b_3	**3**	**4**	**5**	**2**

Розв'язання.

Маємо відкриту модель транспортної задачі. По стовпцях вибираємо клітинки з мінімальною відстанню і записуємо в них необхідну кількість рухомого складу (значення з першого рядка). Сума по рядках вказує скільки одиниць рухомого складу потрібно розмістити в кожному пункті (таблиця 5.5).

Таблиця 5.5

	100	**70**	**120**	**50**
240	**3**	**2** 70	**4** 120	**1** 50
100	**2** 100	**3**	**5**	**4**
0	**3**	**4**	**5**	**2**

Оптимальність рішення можна перевірити методом потенціалів. Отже, в пункті b_1 потрібно розмістити 240 одиниць рухомого складу, в пункті b_2 – 100 одиниць рухомого складу, від пункту b_3 можна відмовитись.

Контрольні запитання

1. Запишіть математичну модель транспортної задачі з додатковими обмеженнями на вивезення.
2. Запишіть математичну модель транспортної задачі з обмеженою пропускною здатністю.
3. Яка умова існування розв’язку транспортної задачі з верхнім (нижнім) обмеженням пропускної здатності?
4. Опишіть алгоритм запису транспортної задачі з проміжними пунктами як звичайної транспортної задачі.
5. Як обчислюється обсяг буфера транспортної задачі з проміжними пунктами?

ТЕМА 6. ЦІЛОЧИСЕЛЬНЕ ЛІНІЙНЕ ПРОГРАМУВАННЯ

6.1. Постановка задачі цілочисельного лінійного програмування та методи її розв'язання

Цілочисельне лінійне програмування орієнтоване на розв’язання задач лінійного програмування, в яких всі або деякі змінні набувають цілих або дискретних значень. Задачі цілочисельного програмування виникають там, де є неподільні елементи, наприклад, людські ресурси, машини, транспорт. До задач цілочисельного програмування належать задачі вибору оптимального маршруту, складання календарних планів, задачі про призначення.

Якщо розв’язувати лінійну цілочисельну задачу методами лінійного програмування без врахування цілочисельності змінних, а потім усі дробові змінні округлити, то в багатьох випадках такий розв’язок не буде оптимальним або навіть допустимим.

Приклад 6.1. Авіакомпанія на замовлення армії повинна перевезти не менше 210 військовослужбовців. Авіакомпанія має у своєму розпорядженні два типи літаків. Літак першого типу перевозить 20 пасажирів і має екіпаж 2 особи. Літак другого типу перевозить 60 пасажирів і має екіпаж 3 особи. Експлуатація одного літака першого типу коштує 10 тис. грошових одиниць, а літака другого типу – 80 тис. грошових одиниць. Скільки потрібно використати літаків кожного типу, щоб затрати були мінімальними? Для формування екіпажів літаків можна використати не більше 16 осіб.

Розв’язання.

Складемо математичну модель

$$10x_1 + 80x_2 \to \min,$$
$$\begin{cases} 20x_1 + 60x_2 \geq 210, \\ 2x_1 + 3x_2 \leq 16, \end{cases}$$
$$x_1 \geq 0, \quad x_2 \geq 0,$$
$$x_1, x_2 \in Z.$$

Якщо не враховувати умову цілочисельності розв'язку, то сформульована задача є задачею лінійного програмування. Розв'язуючи цю задачу наприклад, графічним методом отримаємо $x_1 = \frac{11}{2}$, $x_2 = \frac{5}{3}$ (рис. 6.1).

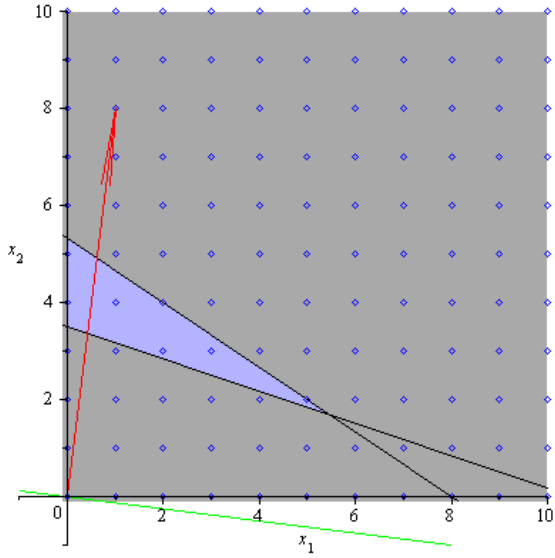


Рис. 6.1. Область допустимих розв'язків

Такий розв'язок з точки зору постановки задачі є безглуздим. Заокруглення отриманого результату (наприклад, $x_1 = 6$, $x_2 = 2$) дає недопустимий розв'язок, оскільки порушуються обмеження задачі.

Одним із методів розв'язання задач цілочисельного лінійного програмування є метод перебору всіх допустимих точок задачі з відповідним порівнюванням між собою значень функції мети (для прикладу 6.1 це точки області допустимих розв'язків з цілочисельними координатами: (0;4), (0;5), (1;4), (2;3), (2;4), (3;3), (5;2)). Метод перебору дає непомірно велику кількість варіантів розв'язків. Наприклад, якщо задача містить 100 змінних, кожна з яких може набувати лише одного з двох можливих значень, кількість варіантів розв'язку є 2^{100}. За досить сильного припущення, що за 1 секунду можна виявити найкращий з трильйона варіантів, для повного перебору всіх варіантів знадобиться більше 30 мільярдів років.

Ще однією неприємною особливістю задач цілочисельного лінійного програмування є те, що для них немає простого способу перевірки знайденого допустимого розв'язку на оптимальність.

Таким чином, вимога цілочисельності розв'язку потребує інших методів розв'язку.

Для розв'язання задач цілочисельного лінійного програмування використовують такі методи:

- метод відтинання (метод Гоморі);
- метод гілок та меж;
- евристичні методи.

Найпоширенішим методом розв'язування задач цілочисельного лінійного програмування є підхід, який використовує ***метод послідовного відтинання*** частини області допустимих розв'язків, яка не має цілочисельних точок. Типовим методом, який використовує таку ідею є алгоритм Гоморі.

Алгоритм Гоморі має таку послідовність:

- розв'язують задачу лінійного програмування без врахування цілочисельності (для цього використовують симплекс-метод);

- якщо розв'язок цілочисельний, то знайдено оптимальний розв'язок задачі цілочисельного лінійного програмування;
- якщо хоча б одна змінна не є цілочисельною, то складають додаткове обмеження;
- модель з додатковим обмеженням знову розв'язують симплекс-методом.

Такий процес пошуку продовжують доти, поки не знаходять оптимального цілочисельного розв'язку або не доведуть відсутність такого розв'язку.

Метод гілок і меж (англ. branch and bound) – є варіацією повного перебору з відсіюванням підмножин допустимих розв'язків, які не містять оптимальних рішень.

Для знаходження розв'язку задачі цілочисельного лінійного програмування методом гілок і меж необхідні дві процедури: розгалуження і знаходження оцінок (меж). Процедура розгалуження полягає в розбитті множини допустимих значень змінної x на підмножини менших розмірів. Процедура знаходження оцінок полягає в пошуку верхніх і нижніх меж для функції мети на підмножинах допустимих значень змінної x. Якщо нижня межа значень функції мети (у випадку мінімізації функції мети) на підмножині пошуку A більша, ніж верхня межа на будь-якій раніше розглянутій підмножині, то підмножина A може бути виключена з подальшого розгляду (правило відсіву).

Евристичні методи використовують тоді, коли застосування двох попередніх є неможливим або не дає розв'язку. Ці методи не забезпечують знаходження точного розв'язку, але можуть знайти розв'язок близький до оптимального.

Розглянемо деякі класичні задачі цілочисельного лінійного програмування.

6.2. Задача про призначення

Нехай є n видів робіт, які потрібно розподілити між n виконавцями по одній роботі на кожного виконавця. Вартість призначення i-го виконавця на j-ту роботу становить c_{ij}. Матрицю змінних моделі визначимо за таким правилом: $x_{ij} = 1$, якщо i-ий виконавець виконує j-ту роботу та $x_{ij} = 0$ в протилежному випадку. Розв'язком задачі є квадратна матриця, в якій в кожному рядку та кожному стовпці є одиниця. Потрібно знайти розподіл виконавців, який є найдешевшим. Таку задачу називають **задачею про призначення**.

Обмеження задачі можна сформулювати так:

- кожна робота повинна бути виконана одним виконавцем;
- кожен виконавець може виконати лише одну роботу.

Математична модель задачі про призначення наступна

$$K = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \to \min,$$

$$\sum_{i=1}^{n} x_{ij} = 1,$$

$$\sum_{j=1}^{n} x_{ij} = 1,$$

$$x_{ij} \in \{0,1\},\ i = \overline{1,n},\ j = \overline{1,n}.$$

Замість матриці вартості призначення i-го виконавця на j-ту роботу може бути задана матриця ефективності використання i-го виконавця на j-ій роботі (c_{ij}). За наведених обмежень функція мети записують так: $M = \sum_{i=1}^{n}\sum_{j=1}^{n} c_{ij}x_{ij} \to \max$.

Якщо розглянути протилежну функцію $F = -M = \sum_{i=1}^{n}\sum_{j=1}^{n}(-c_{ij}x_{ij})$, то задача про призначення перетворюється в транспортну задачу лінійного програмування, в якій необхідно мінімізувати функцію F.

Задачу про призначення можна розглядати, коли кількість виконавців не дорівнює кількості робіт. В цьому випадку вводяться фіктивні роботи або виконавці, для яких коефіцієнти матриці (c_{ij}) рівні нулю. Якщо в оптимальному плані виконавець призначається на виконання фіктивної роботи, то це означає, що він роботи не виконує.

Таким чином, задача про призначення є частковим випадком транспортної задачі, в якій працівники відповідають пунктам відправлення, а роботи – пунктам призначення. В такому випадку всі величини попиту і пропозиції дорівнюють одиниці. Ця особливість призвела до створення спрощеного алгоритму розв'язування транспортної задачі, що називають ***угорським методом***[1]. Як і метод потенціалів, угорський метод базується на симплекс-методі. Цей метод розроблявся для «швидких» ручних обчислень. Проте такі задачі розв'язуються на комп'ютерах, як звичайні задачі лінійного програмування.

6.3. Задача про кільцевий маршрут (комівояжера)

Припустимо, що автомобілі фірми повинні розвезти вантаж по всіх районних містах області й повернутись назад. Відома вартість проїзду між двома будь-якими містами. Фірма бажає знизити свої дорожні витрати. Математичну модель такої

[1]Метод розроблений на основі праць угорських математиків Денеша Кьоніга та Ейгена Егерварі, що і дало назву методу.

ситуації називають задачею комівояжера[2] або задачею про кільцевий маршрут. Сформуємо цю задачу у загальному виді.

Маємо n міст, відстані між якими c_{ij} відомі. Якщо між містами i та j немає дороги, то $c_{ij} = \infty$. Потрібно відвідати всі міста по одному разу й повернутися в місто, з якого почали. Крім цього, загальна довжина маршруту має бути мінімальною.

Матрицю змінних визначимо за правилом: $x_{ij} = 1$, якщо комівояжер переїжджає з міста i до міста j безпосередньо та $x_{ij} = 0$ в протилежному випадку.

Математична модель задачі комівояжера така:

$$K = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} \to \min,$$

$$\sum_{i=1}^{m} x_{ij} = 1,$$

$$\sum_{j=1}^{n} x_{ij} = 1,$$

$$x_{ij} \in \{0,1\},\ i = \overline{1,m},\ j = \overline{1,n},$$

$x_{ij} = 0$, при $i = j$, маршрут у вигляді циклу.

Можна спробувати перебрати всі маршрути, порахувати їх довжини та вибрати найменший. Але такий метод повного перебору майже неможливий навіть за достатньо невеликої кількості міст, оскільки кількість всеможливих маршрутів для k міст становить $(k - 1)!$

Основним методом розв'язування задачі комівояжера є метод гілок та меж.

---

[2] **Комівояже́р** (фр. *commis voyageor*) – посередник із збуту, агент торгової організації, що пропонує покупцям товари за зразками та каталогами.

Зауважимо, що до задачі комівояжера можна звести задачі оптимального використання транспортних засобів (наприклад, перевезення зі складу товарів до магазинів, збір тари та замовлення, складання оптимальних маршрутів експедиторів), знаходження послідовності обробки деталей верстатами з точки зору мінімізації часу на їх перелагодження та інші.

6.4. Розв'язання задач цілочисельного програмування засобами пакета Maple

В систему ***Maple*** починаючи з версії 9.5 добавлено пакет ***Optimization***, який використовує новітні методи оптимізації. Пакет використовує вбудовану бібліотеку процедур, які розроблені ***NumericalAlgorithmsGroup (NAG)***. За допомогою цього пакета можна розв'язувати не лише задачі лінійної оптимізації.

Підключаємо пакет командою

>with(Optimization):

Довідку для цього пакета можна отримати за допомогою команди

>help(Optimization)

Для розв'язання задач лінійного програмування, зокрема з умовою цілочисельності розв'язку, призначена функція

***LPSolve*(*obj, constr, bd, opts*),**

яка має такі параметри:

obj – функція мети;

constr – множина лінійних обмежень;

bd – послідовність у вигляді **name = range,** яка задає межі змінних;

opts – додаткові опції, може приймати значення **assume**, **binaryvariables**, **depthlimit**, **feasibilitytolerance**, **infinitebound**, **initialpoint**, **integertolerance**, **integervariables**, **iterationlimit**, **maximize**, **nodelimit** або **output.**

За замовчуванням, команда ***LPSolve*** знаходить мінімум функції мети. Для знаходження максимуму функції потрібно задати опцію **maximize**. Для розв'язання задач цілочисельного лінійного програмування описана функція застосовується з опціями **assume = binary** (змінні набувають значеннь 0 або 1), **integer** (змінні набувають цілих значень) або **nonnegint (**змінні набувають невід'ємних цілих значень).

Розв'яжемо приклад 6.1 в пакеті ***Maple*** (рис. 6.2).

```
> restart :
Задаємо функцію мети та систему обмежень
> f := 10·x1 + 80·x2; ineq := {x1 ≥ 0, x2 ≥ 0, 20·x1 + 60·x2 ≥ 210, 2·x1 + 3·x2 ≤ 16}
                              f := 10 x1 + 80 x2
                  ineq := {0 ≤ x1, 0 ≤ x2, 210 ≤ 20 x1 + 60 x2, 2 x1 + 3 x2 ≤ 16}
Підключаємо пакет Optimization
> with(Optimization)
    [ImportMPS, Interactive, LPSolve, LSSolve, Maximize, Minimize, NLPSolve, QPSolve]
Знаходимо мінімальне значення функції мети при умові виконання заданих обмеженнях та
цілочисельності розв'язку
> LPSolve(f, ineq, assume = integer)
                                  [210, [x1 = 5, x2 = 2]]
```

Рис. 6.2. Розв'язання задачі цілочисельного лінійного програмування прикладу 6.1 в пакеті ***Maple***

Приклад 6.2. Автотранспортне підприємство для обслуговування п'яти маршрутів використовує п'ять автомобілів. Різні техніко-економічні характеристики маршруту та різні технічні параметри автомобілів зумовлюють різний прибуток від виконання перевезень (див. таб. 6.1). Знайти такий розподіл автомобілів на маршрути, який забезпечить найбільший прибуток підприємству.

Таблиця 6.1

Номер автомобіля	***Номер маршруту***				
	1	**2**	**3**	**4**	**5**
1	4	8	6	8	9
2	7	5	4	4	11
3	2	2	1	2	6
4	19	9	3	6	9
5	6	7	2	5	8

Розв'язання.

Розв'яжемо задачу в ***Maple***, як задачу цілочисельного лінійного програмування (рис. 6.3).

```
> restart;
> with(Optimization);
    [ImportMPS, Interactive, LPSolve, LSSolve, Maximize, Minimize, NLPSolve, QPSolve]
```

Задаємо матрицю вартості призначень та матрицю змінних

$$> c := \begin{bmatrix} 4 & 8 & 6 & 8 & 9 \\ 7 & 5 & 4 & 4 & 11 \\ 2 & 2 & 1 & 2 & 6 \\ 19 & 9 & 3 & 6 & 9 \\ 6 & 7 & 2 & 5 & 8 \end{bmatrix}; x := \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} & m_{1,5} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} & m_{2,5} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} & m_{3,5} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} & m_{4,5} \\ m_{5,1} & m_{5,2} & m_{5,3} & m_{5,4} & m_{5,5} \end{bmatrix};$$

$$c := \begin{bmatrix} 4 & 8 & 6 & 8 & 9 \\ 7 & 5 & 4 & 4 & 11 \\ 2 & 2 & 1 & 2 & 6 \\ 19 & 9 & 3 & 6 & 9 \\ 6 & 7 & 2 & 5 & 8 \end{bmatrix}$$

$$x := \begin{bmatrix} m_{1,1} & m_{1,2} & m_{1,3} & m_{1,4} & m_{1,5} \\ m_{2,1} & m_{2,2} & m_{2,3} & m_{2,4} & m_{2,5} \\ m_{3,1} & m_{3,2} & m_{3,3} & m_{3,4} & m_{3,5} \\ m_{4,1} & m_{4,2} & m_{4,3} & m_{4,4} & m_{4,5} \\ m_{5,1} & m_{5,2} & m_{5,3} & m_{5,4} & m_{5,5} \end{bmatrix}$$

Задаємо функцію мети

$$> \quad F := \sum_{i=1}^{5} \left(\sum_{j=1}^{5} "c"[i,j] \cdot "x"[i,j] \right);$$

$$F := 4\,m_{1,1} + 8\,m_{1,2} + 6\,m_{1,3} + 8\,m_{1,4} + 9\,m_{1,5} + 7\,m_{2,1} + 5\,m_{2,2} + 4\,m_{2,3} + 4\,m_{2,4} + 11\,m_{2,5} + 2\,m_{3,1} + 2\,m_{3,2} + m_{3,3} + 2\,m_{3,4} + 6\,m_{3,5} + 19\,m_{4,1} + 9\,m_{4,2} + 3\,m_{4,3} + 6\,m_{4,4} + 9\,m_{4,5} + 6\,m_{5,1} + 7\,m_{5,2} + 2\,m_{5,3} + 5\,m_{5,4} + 8\,m_{5,5}$$

Задаємо систему обмежень

$$> \quad obmez := \left(\left\{ \sum_{i=1}^{5} 'x'[i,1] = 1, \sum_{i=1}^{5} 'x'[i,2] = 1, \sum_{i=1}^{5} 'x'[i,3] = 1, \sum_{i=1}^{5} 'x'[i,4] = 1, \sum_{i=1}^{5} 'x'[i,5] = 1, \sum_{j=1}^{5} 'x'[1,j] = 1, \sum_{j=1}^{5} 'x'[2,j] = 1, \sum_{j=1}^{5} 'x'[3,j] = 1, \sum_{j=1}^{5} 'x'[4,j] = 1, \sum_{j=1}^{5} 'x'[5,j] = 1 \right\} \right);$$

$$obmez := \{ m_{1,1} + m_{1,2} + m_{1,3} + m_{1,4} + m_{1,5} = 1,\ m_{1,1} + m_{2,1} + m_{3,1} + m_{4,1} + m_{5,1} = 1,\ m_{1,2} + m_{2,2} + m_{3,2} + m_{4,2} + m_{5,2} = 1,\ m_{1,3} + m_{2,3} + m_{3,3} + m_{4,3} + m_{5,3} = 1,\ m_{1,4} + m_{2,4} + m_{3,4} + m_{4,4} + m_{5,4} = 1,\ m_{1,5} + m_{2,5} + m_{3,5} + m_{4,5} + m_{5,5} = 1,\ m_{2,1} + m_{2,2} + m_{2,3} + m_{2,4} + m_{2,5} = 1,\ m_{3,1} + m_{3,2} + m_{3,3} + m_{3,4} + m_{3,5} = 1,\ m_{4,1} + m_{4,2} + m_{4,3} + m_{4,4} + m_{4,5} = 1,\ m_{5,1} + m_{5,2} + m_{5,3} + m_{5,4} + m_{5,5} = 1 \}$$

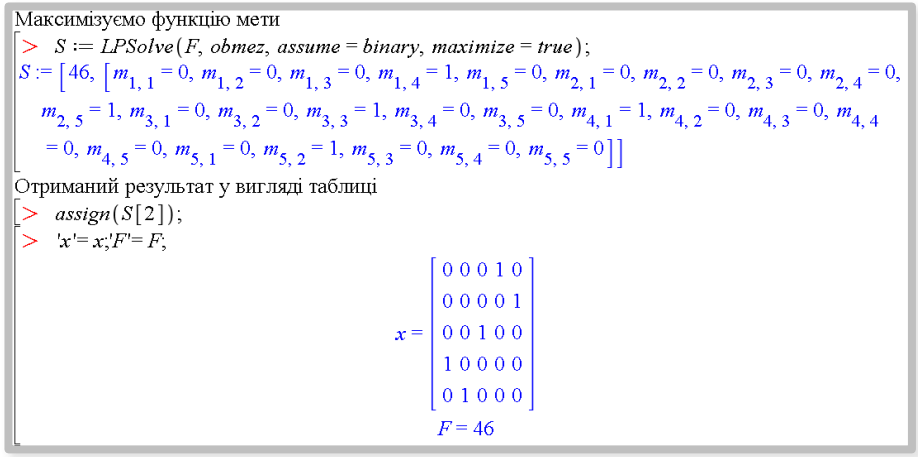


Рис. 6.3. Метод оптимізації в пакеті ***Maple***

Приклад 6.3. Згідно з матрицею відстаней (див. таб. 6.2), знайти кільцевий маршрут постачання готової продукції від заводу до чотирьох баз призначення з мінімізацією загальної відстані пробігу автотранспорту.

Таблиця 6.2

C	**1**	**2**	**3**	**4**	**5**
1	∞	7	16	21	2
2	13	∞	21	15	43
3	25	3	∞	31	17
4	13	10	27	∞	33
5	9	2	19	14	∞

Розв'язання.

Для розв'язання задачі комівояжера, як задачі теорії графів, в ***Maple*** існує функція

***TravelingSalesman*(*G, M*)**

пакета ***GraphTheory***. Основні поняття теорії графів будемо розглядати в наступних розділах. Розв'яжемо задачу комівояжера в пакеті ***Maple*** (рис. 6.4).

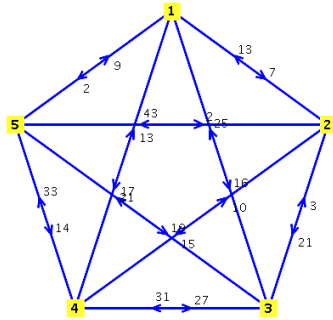
```
> restart;
> with(GraphTheory) :
```

Задаємо матрицю відстаней

$$> A := \begin{bmatrix} 0 & 7 & 16 & 21 & 2 \\ 13 & 0 & 21 & 15 & 43 \\ 25 & 3 & 0 & 31 & 17 \\ 13 & 10 & 27 & 0 & 33 \\ 9 & 2 & 19 & 14 & 0 \end{bmatrix};$$

$$A := \begin{bmatrix} 0 & 7 & 16 & 21 & 2 \\ 13 & 0 & 21 & 15 & 43 \\ 25 & 3 & 0 & 31 & 17 \\ 13 & 10 & 27 & 0 & 33 \\ 9 & 2 & 19 & 14 & 0 \end{bmatrix}$$

```
> G := Graph(A) : DrawGraph(G, style = circle);
```




```
> w, tour := TravelingSalesman(G); tour;
```

$$w, tour := 52, [1, 5, 3, 2, 4, 1]$$

$$[1, 5, 3, 2, 4, 1]$$

Рис. 6.4. Задача комівояжера в пакеті ***Maple***

Таким чином, отримано кільцевий маршрут 1→5→3→2→4→1 мінімальної довжини $w = 52$, що проходить через усі

вершини графа. Вимога відвідування кожного пункту тільки один раз виконана.

Контрольні запитання

1. Які задачі лінійного програмування називають цілочисельними?
2. Які методи розв'язання задач цілочисельного лінійного програмування ви знаєте? Чи можна розв'язати задачі цілочисельного лінійного програмування симплекс-методом?
3. Сформулюйте умову задачі комівояжера.
4. Сформулюйте умову задачі про призначення.
5. Сформулюйте математичну модель задачі комівояжера та задачі про призначення.
6. Сформулюйте задачу про призначення, як транспортну задачу.
7. Які існують методи розв'язання задачі про призначення?

ТЕМА 7. ЕЛЕМЕНТИ ТЕОРІЇ ГРАФІВ

7.1. Основні поняття теорії графів

Теорія графів, як теоретична дисципліна, є розділом дискретної математики. Як прикладна дисципліна, теорія графів допомагає описувати й досліджувати багато технічних, економічних, біологічних і соціальних систем. Спочатку теорія графів здавалась досить незначним розділом математики оскільки застосовувалась в основному до математичних розваг і головоломок.

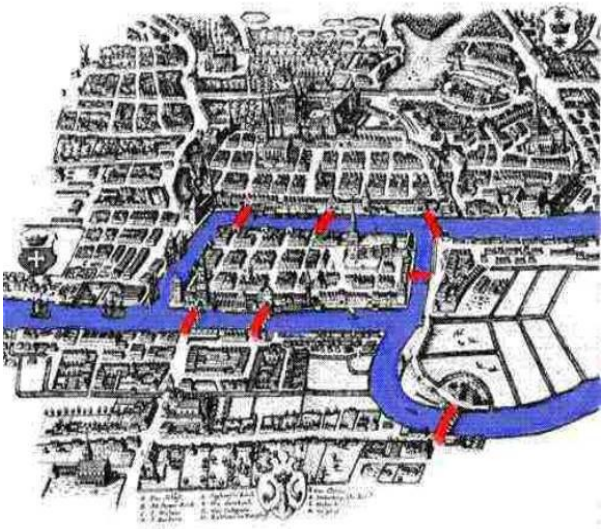


Рис. 7.1. Задача про кенігсберзькі мости

Початок теорії графів датують 1736 роком, коли Л. Ейлер розв'язав популярну в той час "задачу про кенігсберзькі мости" (рис. 7.1), а саме: до XVIII століття через ріку, на якій стояло місто Кенігсберг (нині Калінінград), було побудовано 7 мостів, які сполучали з берегами й один з одним два острови, розташовані в межах міста. Потрібно пройти (якщо це можливо) по всіх мостах так, щоб на кожному з них побувати лише по одному разу й повернутися до того місця, звідки почав маршрут. Ейлер довів, що це неможливо. Відповідну теорему, яка є наслідком цієї головоломки, ми розглянемо пізніше.

Термін "граф" уперше був уведений через 200 років (в 1936 р.) Д. Кенигом. Подальший розвиток математики дав сильний поштовх до розвитку теорії графів. Вже в XIX столітті графи використовувалися для побудови схем електричних ланцюгів і молекулярних схем, для встановлення різного виду відповідностей, для вирішення транспортних задач, задач про потоки в мережі нафтопроводів тощо. Теорія графів тепер застосовується і в таких областях, як економіка, психологія і біологія. Розглянемо основні поняття теорії графів.

Граф – це сукупність об'єктів та зв'язків між ними. Геометрично граф – це фігура, що складається з точок (вершин) і ліній (ребер), що сполучають деякі з цих вершин. Ребра можуть бути неорієнтованими та орієнтованими (дуги).

Якщо дві вершини з'єднує кілька ребер (дуг), то такі ребра (дуги) називають **кратними**. Ребро (дугу), що з'єднує вершину саму зі собою називають **петлею.** Це геометричний спосіб задання графа (рис.7.2).

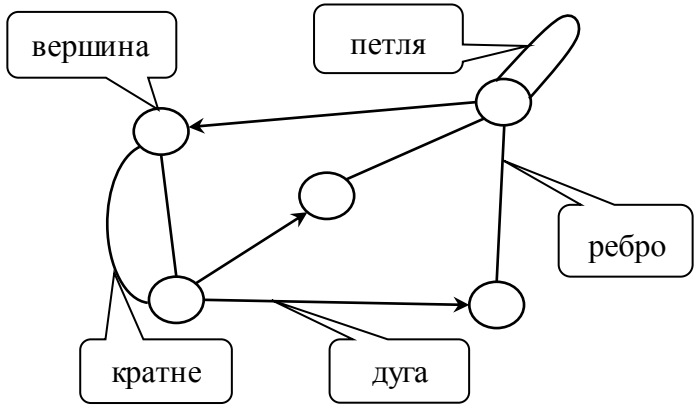


Рис. 7.2. Геометричний спосіб задання графа

Граф, у якому заданий напрямок ребер (всі ребра є дугами), називають **орієнтованим**, інакше – граф **неорієнтований.**

Надалі будемо вважати, що граф не має петель та кратних ребер.

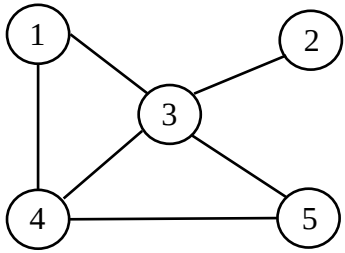
Дві вершини називають **суміжними**, якщо вони з'єднані ребром (дугою). Суміжні вершини називають **граничними вершинами** відповідного ребра (дуги), а це ребро (дуга) – **інцидентним до** відповідних вершин.

У неорієнтованому графі **степенем вершини** називають кількість d_i інцидентних до неї ребер (i – номер вершини). Граф, степені всіх вершин якого дорівнюють $n-1$, де n – кількість вершин графа, називають **повним.**

Вершину, для якої не існує інцидентних до неї ребер $(d_i = 0)$, називають **ізольованою**. Вершину, для якої існує тільки одне інцидентне до неї ребро $(d_i = 1)$ називають **висячою.** Наприклад на рисунку рис. 7.3 *а*) вершина 2 є висячою.

Графи також можна задавати ***матрицею суміжності,*** елементи якої a_{ij} дорівнюють одиниці, якщо вершина i є суміжною з вершиною j, та дорівнюють нулю, якщо вершина i є несуміжною з вершиною j.

Наприклад, графу на рис.7.3 *а*) відповідає матриця суміжності, зображена на рис. 7.3 б).



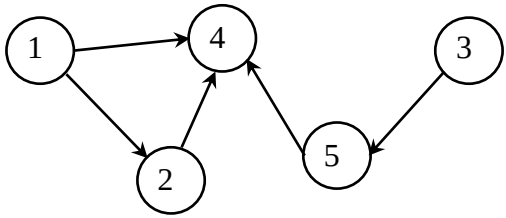
$$\begin{pmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 \end{pmatrix}$$

а) *б*)

Рис. 7.3. Неорієнтований граф та його матриця суміжності

Зауважимо, що для неорієнтовного графа матриця суміжності завжди є симетричною. Якщо в графі немає петель, то діагональні елементи матриці суміжності дорівнюють нулю, тобто $a_{ii} = 0$.

У випадку орієнтованого графа, матриця суміжності є несиметричною (рис.7.4).



$$\begin{pmatrix} 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Рис. 7.4. Орієнтований граф та його матриця суміжності

Твердження. *Сума степенів вершин графа є число парне, яке дорівнює подвоєній кількості m ребер графа*

$$\sum_{i=1}^{n} d_i = 2m.$$

Наслідок. *Кількість вершин непарного степеня у графі є парною.*

Приклад 7.1. Чи існує граф з вершинами таких степенів? Якщо так, то скільки ребер він має?
а) 3 3 3 3 2; б) 3 4 3 4 3.

Розв'язання.

а) Існує, оскільки кількість вершин непарного степеня є парним числом. Сума степенів вершин дорівнює 14. Отже, кількість ребер $m = 7$.

б) Не існує, оскільки кількість вершин непарного степеня є непарним числом.

Шляхом називають послідовність ребер (дуг) графа, у якій кінцева вершина одного ребра (дуги) є початком іншого ребра (дуги).

Якщо будь-які дві вершини графа можна з'єднати шляхом, то такий граф називають **зв'язним.**

Замкнутий шлях називають **циклом.**

Простим шляхом (циклом) називають шлях (цикл), в якому ребра (дуги) не повторюються.

Наведені означення не є загальноприйнятими.

Шляхом (циклом) Ейлера називають простий шлях (цикл) графа, що містить всі його ребра. Граф, що має цикл Ейлера називають **графом Ейлера.**

Іншими словами, якщо граф володіє циклом Ейлера, то його можна накреслити не відриваючи олівець від паперу, проводячи по кожному ребру лише один раз. Рух можна почати з будь-якої вершини й закінчити в тій самій вершині.

Теорема Ейлера. *Зв'язний граф є графом Ейлера тоді й тільки тоді, коли степені всіх його вершин парні.*

Теорема. *Неорієнтований граф має шлях Ейлера тоді й тільки тоді, коли він зв'язний і число вершин непарного степеня дорівнює* 0 *або* 2.

Якщо граф має дві вершини непарного степеня, то його можна накреслити не відриваючи олівець від паперу, при цьому рух треба починати з однієї з цих вершин непарного степеня і закінчити в другій з них.

Граф, який відповідає задачі про кенігсберзькі мости наведено на рис. 7.5. Оскільки він має чотири вершини непарного степеня, то він не є графом Ейлера, тобто неможливо пройти по всіх мостах (рис.7.1) так, щоб на кожному з них

побувати лише по одному разу й повернутися до того місця, звідки почато маршрут.

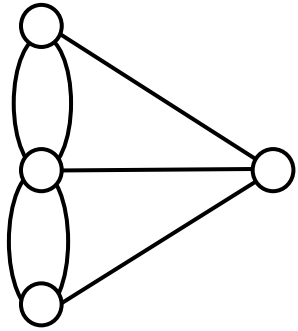


Рис.7.5. Граф до задачі про кенігсберзькі мости

Одним із алгоритмів знаходження Ейлеревого циклу у графі є алгоритм Фльорі, який ґрунтується на таких двох правилах:

1. Починають цикл з довільної вершини, проходять вздовж ребер графа довільним чином. Після проходження ребра, йому присвоюють черговий номер в циклі і це ребро викреслюють, тобто вилучають з графа (вершини при цьому залишають).

2. Якщо є декілька можливостей продовжити рух вздовж ребер, то вибираємо рух вздовж того ребра, вилучення якого не призведе до втрати зв'язності графа.

Мова графів є зручною для опису багатьох фізичних, технічних, економічних, біологічних, соціальних та інших систем. Розглянемо деякі приклади застосування теорії графів.

Транспортні задачі, у яких вершинами графа є пункти, а ребрами – дороги (автомобільні, залізниці й ін.) і/або інші транспортні (наприклад, авіаційні) маршрути. Інший приклад – мережі постачання (енергопостачання, газопостачання, постачання товарами й т. ін.), у яких вершинами є пункти

виробництва й споживання, а ребрами – можливі маршрути переміщення (лінії електропередач, газопроводи, дороги й т. ін.). Підкласом є задачі про вантажоперевезення.

Технологічні задачі, у яких вершини відображають виробничі елементи (заводи, цехи, верстати й т.д.), а дуги – потоки сировини, матеріалів і продукції, які між ними пролягають. Такі задачі полягають у визначенні оптимального завантаження виробничих елементів та визначенні потоків, що забезпечують оптимальне завантаження.

Обмінні схеми, які є моделями таких явищ, як бартер, взаємозаліки й т. ін. Вершини графа при цьому описують учасників обмінної схеми (ланцюга), а дуги – потоки матеріальних і фінансових ресурсів між ними. Задача полягає у визначенні ланцюга обмінів, оптимального з погляду, наприклад, організатора обміну й погодженого з інтересами учасників ланцюга й існуючих обмежень.

Управління проектами. З погляду теорії графів проект – сукупність операцій і залежностей між ними (сітьовий графік). Хрестоматійним прикладом є проект будівництва деякого об'єкта. Сукупність моделей і методів, що використовують мову й результати теорії графів й орієнтованих на рішення завдань управління проектами, одержала назву "календарно-мережевого планування й управління". Такі задачі визначають послідовності виконання операцій і розподілу ресурсів між ними, оптимальних з погляду тих або інших критеріїв (часу виконання проекту, витрат, ризику й ін.).

7.2. Екстремальні шляхи на графах

Задача пошуку найкоротших і найдовших шляхів на графах виникає у різних областях керування. Розглянемо задачу про найкоротший шлях.

7.2.1. Задача про найкоротший шлях між двома парами вершин. Алгоритм Дейкстри.

Нехай задано орієнтований граф з $n+1$ вершинами, у якому виділено дві вершини: вхід (нульова вершина) і вихід (вершина з номером n). Такі графи називають **мережевими графіками або мережами.** Нехай для кожної дуги задано число l_{ij}, яке будемо називати **довжиною або вагою дуги**. Граф, для якого задано довжини дуг, називають **зваженим.**

Довжиною шляху називають суму довжин вхідних у нього дуг. Якщо довжини дуг не задано, то довжину шляху визначають як число вхідних у нього дуг. Завдання полягає в пошуку найкоротшого шляху (шляху мінімальної довжини) від початкової до кінцевої вершини.

Припустимо, що в сітьовому графіку можна пронумерувати вершини таким чином, що для будь-якої дуги (i, j) $i > j$. Така нумерація називається **правильною.**

Найкоротший шлях у сітьовому графіку, що має правильну нумерацію, визначають **алгоритмом Дейкстри.**

Алгоритм Дейкстри:

- позначаємо нульову вершину індексом $\lambda_0 = 0$;
- позначаємо вершину k індексом $\lambda_k = \min\limits_{i<k}(\lambda_i + l_{ik})$;
- індекс виходу λ_n дорівнює довжині найкоротшого шляху;
- коли індекси встановлені, найкоротший шлях визначаємо методом зворотного ходу від виходу до входу.

Описаний алгоритм дає змогу знайти найкоротший шлях від нульової вершини до всіх інших. Цей алгоритм можна застосовувати лише для дуг невід'ємної довжини.

На рис. 7.6 наведено приклад застосування алгоритму Дейкстри для визначення найкоротшого шляху (числа біля дуг дорівнюють довжинам дуг, індекси вершин λ_k поміщені у

квадратні дужки, найкоротший шлях виділено жирними лініями).

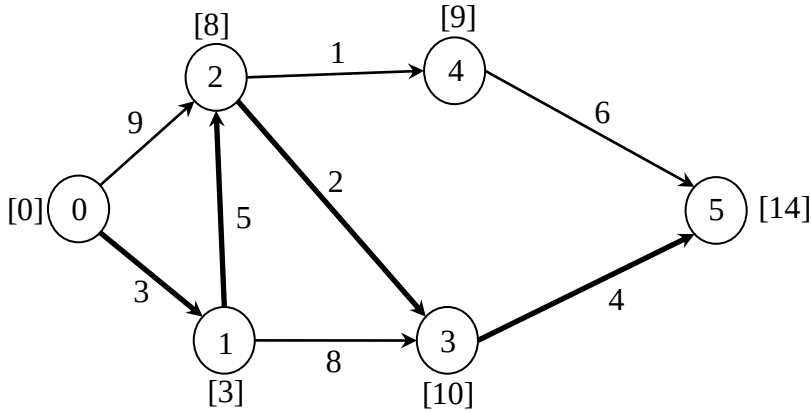


Рис. 7.6.Застосування алгоритму Дейкстри

7.2.2. Знаходження найкоротших шляхів між всіма парами вершин. Алгоритм Флойда

Для знаходження найкоротших шляхів між всіма парами вершин використовують алгоритм Флойда. Умовою застосування є відсутність в графі циклів з сумарною від'ємною вагою, але алгоритм дає змогу виявити такі контури.

Розглянемо орієнтований граф $G = (V, A)$, де V – вершини графа, A – ребра графа. Нехай $W^0 = \| w_{ij} \|$ – матриця довжин дуг цього графа. Покладемо $w_{ij} = \infty$, якщо дуга (i, j) відсутня і $w_{ii} = 0$ для всіх $i \in V$. Вважаємо, що петель немає. Алгоритм Флойда будує послідовність матриць W^0, W^1, …, W^n таку, що елемент w_{ij}^n матриці W^n дорівнює довжині найкоротшого шляху від

вершини i до вершини j в графі G. Матриця W^k визначається за матрицею W^{k-1} такою формулою

$$w_{ij}^k = \min\{w_{ij}^{k-1}, w_{ik}^{k-1} + w_{kj}^{k-1}\}.$$

В алгоритмі Флойда одночасно з довжинами найкоротших шляхів шукаємо самі шляхи за допомогою матриць Z^0, Z^1, …, Z^n, де елемент z_{ij}^k матриці Z^k вказує на вершину, яка передує вершині j на шляху від вершини i до j. Спочатку для всіх i та j покладаємо $z_{ij}^0 = i$, якщо існує дуга (i, j) (тобто відповідний елемент матриці W^0 не дорівнює ∞). В іншому випадку $z_{ij}^0 = 0$. Далі на кожній ітерації матриця Z^k перераховується разом з матрицею W^k за таким правилом:

$$z_{ij}^k = \begin{cases} z_{kj}^{k-1}, & \textit{якщо } w_{ij}^{k-1} > w_{ik}^{k-1} + w_{kj}^{k-1}; \\ z_{ij}^{k-1}, & \textit{якщо } w_{ij}^{k-1} \le w_{ik}^{k-1} + w_{kj}^{k-1}, \end{cases}$$

тобто, якщо на k-ій ітерації елемент w_{ij}^k міняється, то елемент z_{ij}^k отримує значення, що дорівнює елементу, який розташований в тому ж стовпці у k-му рядку. Розглянемо алгоритм Флойда на такому прикладі.

Приклад 7.2. Задано орієнтований граф (рис. 7.7).

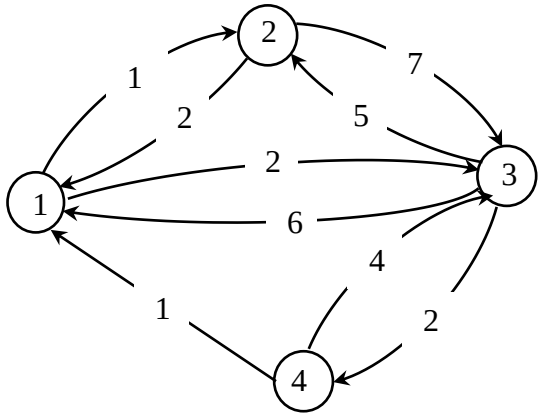


Рис. 7.7. Застосування алгоритму Флойда

Знайти найкоротший шлях між всіма парами вершин та їх довжини.

Розв'язання.

Побудуємо матрицю довжин дуг

$$W^0 = \begin{array}{|c|cccc|} \hline & 0 & 1 & 2 & \infty \\ \hline 2 & 2 & 0 & \underline{7} & \infty \\ 6 & 6 & \underline{5} & 0 & 2 \\ 1 & 1 & \infty & \underline{4} & 0 \\ \hline \end{array}.$$

Крок 1. Виділяємо перший стовпець та перший рядок матриці W^0. Виділені елементи перераховувати не треба, оскільки 4-ий елемент виділеного рядка дорівнює ∞, то відповідний стовпчик (4-ий) ми не перераховуємо. Аналогічне твердження правильне і для виділеного стовпця, але на цій ітерації виділений стовпець не має елементів, що дорівнюють ∞.

Оскільки довжини всіх дуг невід'ємні, то діагональні елементи матриці сталі й дорівнюють нулю. Таким чином, потрібно перерахувати лише елементи w_{23}^0, w_{32}^0, w_{42}^0, w_{43}^0 (підкреслені в таблиці). Отримаємо

$w_{23}^1 = \min(7; 2+2) = 4;$

$w_{32}^1 = \min(5; 6+1) = 5;$

$w_{42}^1 = \min(\infty; 1+1) = 2;$

$w_{43}^1 = \min(4; 1+2) = 3.$

Отримаємо таку матрицю довжин

$$W^1 = \begin{vmatrix} 0 & 1 & 2 & \infty \\ 2 & 0 & 4 & \infty \\ 6 & 5 & 0 & 2 \\ 1 & 2 & 3 & 0 \end{vmatrix}.$$

Матриця W^0 відрізняється від матриці W^1 елементами w_{23}^0, w_{42}^0, w_{43}^0.

Крок 2. В матриці W^1 виділяємо другий рядок та другий стовпчик. Аналогічно до попереднього кроку отримаємо таку послідовність матриць

$$W^2 = \begin{vmatrix} 0 & 1 & 2 & \infty \\ 2 & 0 & 4 & \infty \\ 6 & 5 & 0 & 2 \\ 1 & 2 & 3 & 0 \end{vmatrix}; \quad W^3 = \begin{vmatrix} 0 & 1 & 2 & 4 \\ 2 & 0 & 4 & 6 \\ 6 & 5 & 0 & 2 \\ 1 & 2 & 3 & 0 \end{vmatrix}; \quad W^4 = \begin{vmatrix} 0 & 1 & 2 & 4 \\ 2 & 0 & 4 & 6 \\ 3 & 4 & 0 & 2 \\ 1 & 2 & 3 & 0 \end{vmatrix}.$$

Таким чином, отримали найкоротші відстані між будь-якою парою вершин. Наприклад, найменша відстань між вершинами 3 та 1 дорівнює $w_{31}^4 = 3$.

Знайдемо відповідні шляхи. Матриця Z^0 має вигляд

$$Z^0 = \begin{vmatrix} 0 & 1 & 1 & 0 \\ 2 & 0 & 2 & 0 \\ 3 & 3 & 0 & 3 \\ 4 & 0 & 4 & 0 \end{vmatrix}.$$

На першій ітерації (крок 1) міняються елементи w_{23}^0, w_{42}^0, w_{43}^0. Відповідні елементи матриці Z^1 будуть дорівнювати елементам першого рядка матриці Z^0, які розташовані в тому ж стовпці, тобто

$$z_{23}^1 = z_{13}^0 = 1;$$
$$z_{42}^1 = z_{12}^0 = 1;$$
$$z_{43}^1 = z_{13}^0 = 1,$$

$$Z^1 = \begin{vmatrix} 0 & 1 & 1 & 0 \\ 2 & 0 & 1 & 0 \\ 3 & 3 & 0 & 3 \\ 4 & 1 & 1 & 0 \end{vmatrix}.$$

На другому кроці матриця W^2 не змінилась ($W^2 = W^1$), а отже, не змінюється й матриця Z^2 ($Z^2 = Z^1$).

Далі

$$Z^3 = \begin{vmatrix} 0 & 1 & 1 & 3 \\ 2 & 0 & 1 & 3 \\ 3 & 3 & 0 & 3 \\ 4 & 1 & 1 & 0 \end{vmatrix}, \qquad Z^4 = \begin{vmatrix} 0 & 1 & 1 & 3 \\ 2 & 0 & 1 & 3 \\ 4 & 1 & 0 & 3 \\ 4 & 1 & 1 & 0 \end{vmatrix}.$$

Остання матриця Z^4 дає змогу знайти найкоротший шлях для будь-якої пари вершин, наприклад, з 3 в 2. Елемент $z_{32}^4 = 1$, тобто вершині 2 передує вершина 1. В свою чергу їй передує вершина 4, бо $z_{31}^4 = 4$, вершині 4 передує 3, бо $z_{34}^4 = 3$, тобто

початкова вершина. Таким чином, маємо найкоротший шлях між вершинами 3 та 2: 3→4→1→2. Довжина найкоротшого шляху $w_{32}^{4} = 4$.

7.3. Дерева

Зв'язний граф, який не має циклів, називають **деревом**. Зв'язний граф *T*, що містить всі вершини графа *G* і не містить циклів, називають **кістяковим деревом графа *G***.

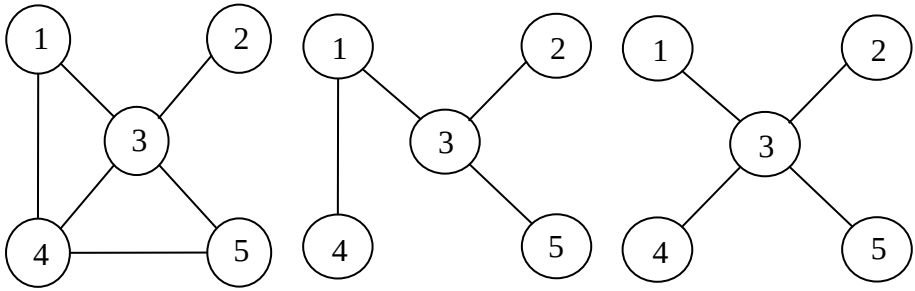


Рис. 7.8. Приклад графа та його кістякових дерев

На рисунку 7.8 зображено граф та два його кістякових дерева.

Однією з важливих задач теорії графів є пошук кістякового дерева мінімальної довжини для зваженого графа. Така задача має широке практичне застосування в проектуванні доріг, електричних мереж, трубопроводів, тобто в ситуаціях, де необхідно з'єднати задану множину об'єктів комунікаційними лініями так, щоб сумарна їх вартість (або довжина) були мінімальними. Одним з алгоритмів знаходження мінімального кістякового дерева є **алгоритм Краскала.**

Алгоритм Краскала. Нехай задано зважений неорієнтований граф *G* з *n* вершинами. Позначимо: *V* – множина всіх вершин графа *G*, E_T – множина ребер мінімального

кістякового дерева T графа G. За означенням, множина вершин дерева T дорівнює множині вершин графа G. Множина ребер E_T дерева T формується поступово за таким алгоритмом.

Крок 0. Покладаємо $E_{T,0} = \varnothing$.

Крок i ($i = \overline{1, n-2}$). Покладаємо $E_{T,i+1} = E_{T,i} \cup e$, де e – ребро графа G, яке має мінімальну вагу, не належить $E_{T,i}$ та не утворює циклу з ребрами множини $E_{T,i}$.

Приклад 7.3. Телевізійна компанія планує підключення до своєї кабельної мережі п'ять нових районів (рис. 7.9). На графі показана структура мережі, що планується та відстані (в км) між районами (вершини 2,3,4,5,6) й телецентром (вершина 1). Необхідно спланувати найбільш економну кабельну мережу.

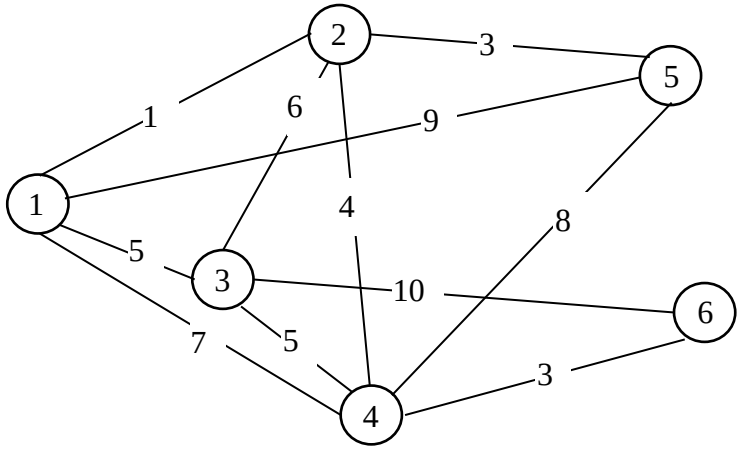


Рис. 7.9. Граф структури кабельної мережі

Розв'язання.

Побудову мінімального кістякового дерева починаємо з відбору всіх вершин графа, заданого на рис. 7.9 та вибору ребра з найменшою довжиною. Таким є ребро (1, 2). Наступні ребра,

які увійдуть до шуканого дерева (2,5) та (4,6), бо жодне з них не утворює циклу з побудованими ребрами кістякового дерева. Далі (2,4), (1,3) або (3,4). Мінімальне кістякове дерево зображено на рис.7.10 жирними лініями.

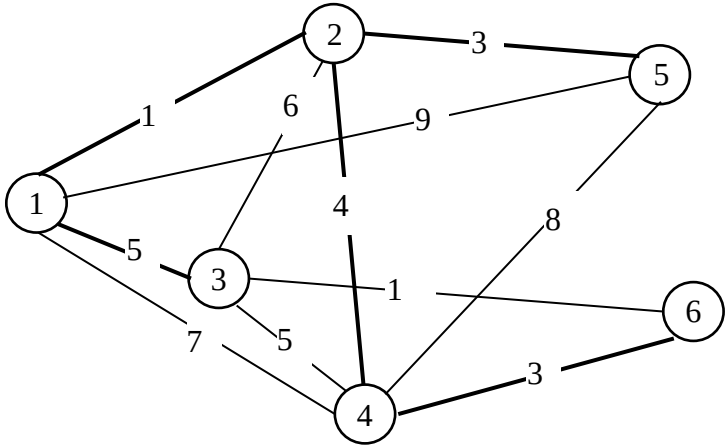


Рис. 7.10. Мінімальне кістякове дерево

7.4. Приклади задач

Задача про заміну обладнання. Компанія з прокату автомобілів розробляє план заміни парку транспортних засобів на наступні 5 років. Кожен автомобіль має відпрацювати не менше одного і не більше трьох років. В таблиці 7.1 наведено вартість заміни автомобіля залежно від року купівлі та тривалості експлуатації (включає вартість нового автомобіля, експлуатаційні витрати та доходи від продажу старого автомобіля).

Таблиця 7.1

Рік експлуатації	Вартість заміни залежно від тривалості експлуатації, тис. у.о.		
	1	2	3
Перший	40	54	98
Другий	43	62	87
Третій	48	71	–
Четвертий	49	–	–

Розв'язання.

Задачу можна зобразити у вигляді графа (мережі) з п'ятьма вершинами, що відповідають п'ятьом рокам (рис. 7.11). З вершини 1 (перший рік експлуатації) дуги йдуть лише до вершин 2, 3 та 4, оскільки автомобіль може експлуатуватись не менше одного і не більше трьох років. Дуги з інших вершин інтерпретуються аналогічно. Довжини дуг є вартістю заміни автомобілів. Розв'язування задачі є еквівалентним пошуку найкоротшого шляху між вершинами 1 та 5.

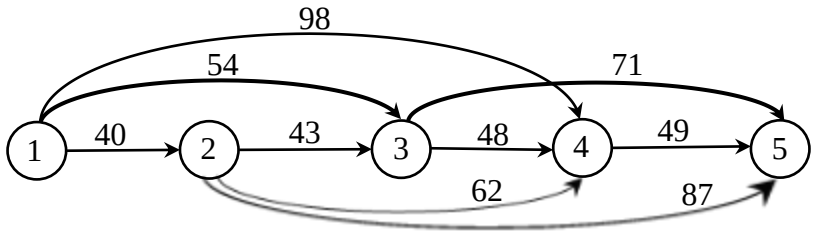


Рис. 7.11. Граф до задачі про заміну обладнання

Найкоротшим шляхом є шлях 1→3→5. Цей розв'язок означає, що придбані автомобілі, які експлуатуються один рік (вершина 1), будуть експлуатуватись 2 роки (вершина 3), потім вони замінюються новими, які будуть експлуатуватись до кінця

запланованого періоду. Загальна вартість заміни одного автомобіля становить: 54+71=125 тис. у.о.

Найбільш надійний маршрут. На рис. 7.12 наведено схему мережі вулиць, якими певний водій щоденно їздить автомобілем з дому на роботу. Нехай ця ділянка вуличко-дорожньої мережі посилено контролюється нарядами поліції, і автомобіль цього водія часто зупиняють за перевищення швидкості. Тому цей водій вирішує розробити маршрут, на якому він би мав найбільшу ймовірність не бути зупиненим інспекторами поліції. Імовірність проїзду без зупинки для кожної вулиці наведено на схемі (рис. 7.12).

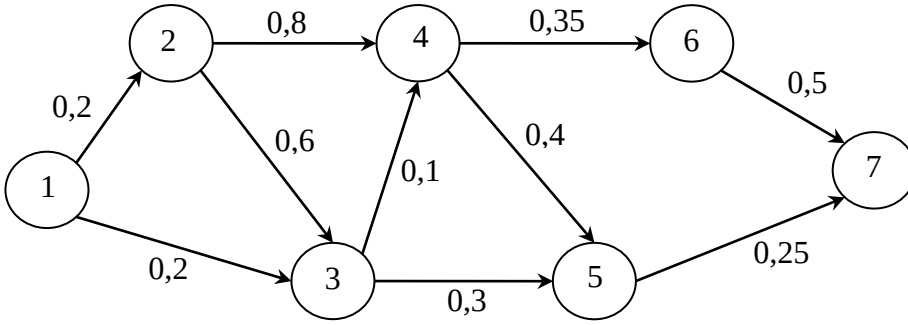


Рис. 7.12. Граф мережі вулиць з імовірностями

Імовірність не бути зупиненим на всій ділянці маршруту P дорівнює добутку ймовірностей не бути зупиненим на кожній ділянці шляху p_i, тобто $P = p_1 \cdot p_2 \cdot ... \cdot p_k$. Цю задачу можна сформулювати як задачу пошуку найкоротшого шляху, якщо замість ймовірностей використовувати логарифми ймовірностей. Тоді добуток ймовірностей перетвориться в суму логарифмів ймовірностей

$$\lg P = \lg p_1 + \lg p_2 + ... + \lg p_k .$$

Максимізація ймовірностей p_i еквівалентна максимізації величини $\lg p_i$. Оскільки $p_i \le 1$, то $\lg p_i \le 0$. Замінивши ймовірності p_i на величини $(-\lg p_i)$, отримуємо мережу, для якої потрібно знайти найкоротший шлях (рис. 7.13).

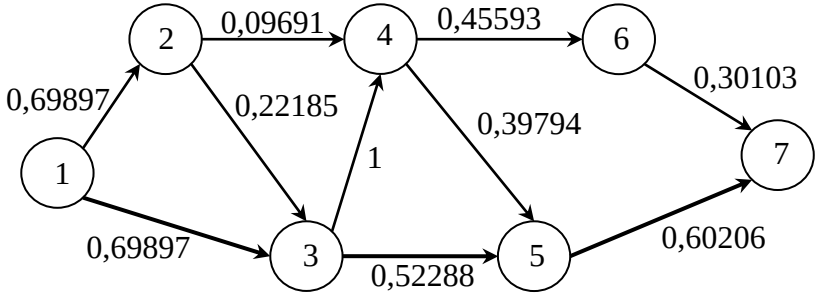


Рис. 7.13. Мережевий граф

Найкоротшим шляхом є шлях 1→3→5→7 з «довжиною» $-\lg P = 1,82391$. Тоді максимальна ймовірність не бути зупиненим становить $P = 0,0149$.

7.5. Розв'язання задач теорії графів в Maple

Для розв'язання задач теорії графів в ***Maple*** потрібно підключити пакет ***GraphTheory***:

>with(*GraphTheory*)*:*

Пакет містить понад 150 функцій, детальніше з якими можна ознайомитись в довідковій системі ***Maple***. Зокрема, зважений граф (орієнтований та неорієнтований) можна задати функцією

***Graph*(*A*)**

з параметром ***A*** – матриця довжин дуг (ребер) графа. Якщо дуга (ребро) (*i, j*) відсутня у графі, то елемент матриці $a_{ij} = 0$.

Зобразити граф можна за допомогою функції

***DrawGraph*(*G*),**

де ***G*** – заданий граф (рис. 7.14).

```
> restart:
> with(GraphTheory):
> A := Matrix([[0, 1, 2, 0], [2, 0, 7, 0], [6, 5, 0, 2], [1, 0, 4, 0]]);
```

$$A := \begin{bmatrix} 0 & 1 & 2 & 0 \\ 2 & 0 & 7 & 0 \\ 6 & 5 & 0 & 2 \\ 1 & 0 & 4 & 0 \end{bmatrix}$$

```
> G := Graph(A);
```

G := Graph 1: a directed weighted graph with 4 vertices and 9 arc(s)

```
> DrawGraph(G);
```

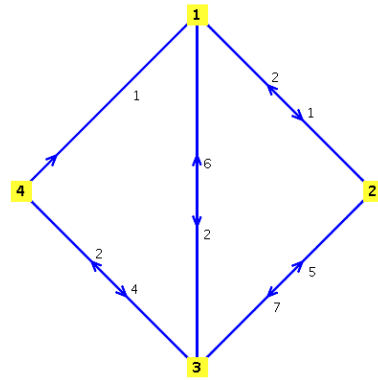


Рис. 7.14. Зображення графа в пакеті ***Maple***

Для знаходження відстаней між будь-якою парою вершин (алгоритм Флойда) використовується функція

***AllPairsDistance*(*G*)**

Відстань та шлях між парою вершин ***s*** та ***t*** вершин знаходимо за допомогою функції

***DijkstrasAlgorithm*(*G*, *s*, *t*).**

Приклад 7.4. Для графа *G*, заданого на рис. 7.14, знайти найменші відстані між будь-якою парою вершин. Знайти найкоротший шлях між вершинами 3 та 2.

Розв'язання.

Розв'язання задачі в пакеті ***Maple*** наведено на рис. 7.15.

```
> Матриця найменших відстаней між будь-якою парою вершин графа G
> AllPairsDistance(G);
```

$$\begin{bmatrix} 0 & 1 & 2 & 4 \\ 2 & 0 & 4 & 6 \\ 3 & 4 & 0 & 2 \\ 1 & 2 & 3 & 0 \end{bmatrix}$$

```
> Шлях та відстань між вершинами 3 та 2 графа G
> DijkstrasAlgorithm(G, 3, 2);
                          [[3, 4, 1, 2], 4]
```

Рис. 7.15. Знаходження найкоротшої відстані в пакеті ***Maple***

Тобто найкоротший шлях між вершинами 3 та 2 є: 3→4→1→2. Його довжина становить 4.

Задачу комівояжера, як задачу пошуку у неорієнтованому зваженому графі циклу, що проходить через всі вершини графа, крім першої, рівно по одному разу (такий цикл називають Гамільтоновим) розв'язуємо за допомогою функції

***TravelingSalesman*(*G*).**

Зразок вирішення задачі комівояжера в пакеті ***Maple*** показано на рис. 7.16.

```
> restart;
> with(GraphTheory) :
```

Задаємо матрицю відстаней

$$> A := \begin{bmatrix} 0 & 7 & 16 & 21 & 2 \\ 13 & 0 & 21 & 15 & 43 \\ 25 & 3 & 0 & 31 & 17 \\ 13 & 10 & 27 & 0 & 33 \\ 9 & 2 & 19 & 14 & 0 \end{bmatrix};$$

$$A := \begin{bmatrix} 0 & 7 & 16 & 21 & 2 \\ 13 & 0 & 21 & 15 & 43 \\ 25 & 3 & 0 & 31 & 17 \\ 13 & 10 & 27 & 0 & 33 \\ 9 & 2 & 19 & 14 & 0 \end{bmatrix}$$

```
> G := Graph(A) : TravelingSalesman(G);
```

$$52, [1, 5, 3, 2, 4, 1]$$

Рис. 7.16. Задача комівояжера в пакеті ***Maple***

Контрольні запитання

1. Дайте визначення основних понять теорії графів.
2. Який граф називають графом Ейлера?
3. Сформулюйте умову існування циклу (шляху) Ейлера в графі.
4. Як обчислити довжину шляху у незваженому графі?
5. За яким алгоритмом можна знайти найкоротший шлях між парою вершин у графі з правильною нумерацією? Опишіть кроки цього алгоритму.
6. За яким алгоритмом можна знайти найкоротший шлях між будь-якою парою вершин у графі?

ТЕМА 8. МЕРЕЖІ ТА ПОТОКИ

8.1. Задача про максимальний потік

Мережа (сітьовий графік) може зображати систему, яка транспортує деякий продукт з одного пункту в інший. Цим продуктом можуть бути люди, електроенергія, природний газ тощо. Прикладом може бути мережа трубопроводів для транспортування нафти від свердловин через насосні станції до нафтопереробних заводів.

Використовуючи таке представлення, розглянемо мережу як орієнтований граф, де кожній дузі (i, j) відповідає додатне дійсне число c_{ij}, яке називають **пропускною здатністю** у напрямку $i \to j$. Такий граф не може мати петель, оскільки розглядаються задачі для транспортування продукту тільки між різними вершинами. Орієнтований граф повинен бути зв'язним. Вхід можна ще називати **джерелом**, а вихід – **стоком**. Джерело не має вхідних дуг, а сток – вихідних. Таку мережу аналітично можна задавати за допомогою **матриці пропускних здатностей.** На схемі мережі пропускну здатність дуги, яка розташована ближче до вершини i будемо позначати c_{ij}, а пропускну здатність дуги, яка розташована ближче до вершини j будемо позначати c_{ij} (рис. 8.1).

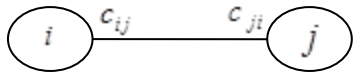


Рис. 8.1. Пропускні здатності дуг

Потрібно знайти максимальну величину потоку (кількості машин, рідини, сировини тощо), який може увійти до мережі та вийти з неї за заданий період часу.

Для розв'язання задачі про максимальний потік використовують метод Форда - Фалкерсона. В ***Maple*** задача про максимальний потік розв'язується за допомогою функції

***MaxFlow*(*G, s, t*),**

де ***G*** – зважений граф;
s – вершина, що є джерелом графа;
t – вершина, що є стоком графа.

Приклад 8.1. Мережа автодоріг Львівської області може забезпечити пропускні здатності (тис. автомашин за годину), які вказані на рис. 8.2. Потрібно визначити максимальний потік у заданій мережі.

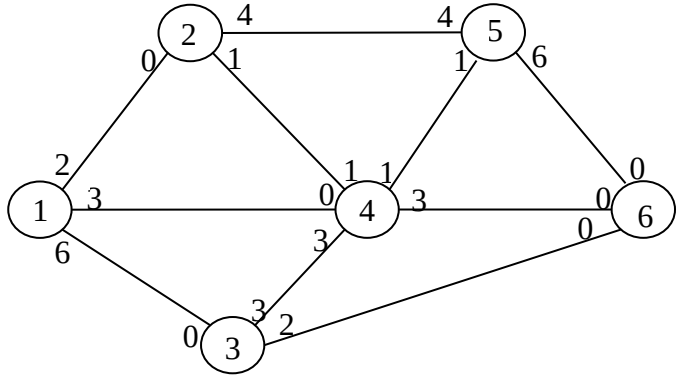


Рис. 8.2. Граф мережі доріг

Розв'язання.

Розв'яжемо задачу в пакеті ***Maple*** (рис. 8.3).

```
> restart:
> with(GraphTheory):
```

Задаємо матрицю пропускних здатностей

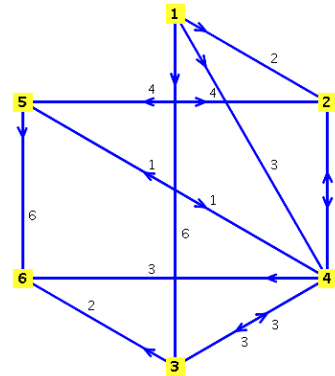
$$> A := \begin{bmatrix} 0&2&6&3&0&0 \\ 0&0&0&1&4&0 \\ 0&0&0&3&0&2 \\ 0&1&3&0&1&3 \\ 0&4&0&1&0&6 \\ 0&0&0&0&0&0 \end{bmatrix};$$

$$A := \begin{bmatrix} 0&2&6&3&0&0 \\ 0&0&0&1&4&0 \\ 0&0&0&3&0&2 \\ 0&1&3&0&1&3 \\ 0&4&0&1&0&6 \\ 0&0&0&0&0&0 \end{bmatrix}$$

Задаємо орієнтований зважений граф, що відповідає матриці A та будуємо його

```
> N := Digraph(A, weighted); DrawGraph(N);
```

N := Graph 1: a directed weighted graph with 6 vertices and 14 arc(s)



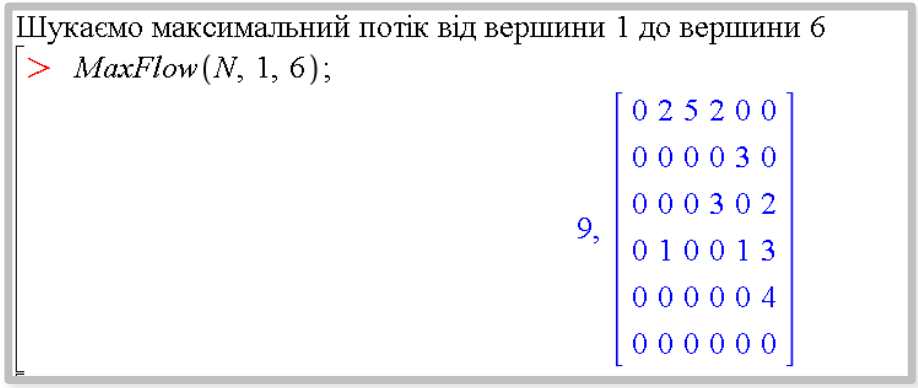


Рис. 8.3. Визначення максимального потоку в пакеті ***Maple***

Отримали величину максимального потоку $F_{\max} = 9$ та матрицю величин потоків вздовж кожної дуги.

8.2. Формалізація задачі про максимальний потік як задачі лінійного програмування

Припустимо, що потрібно знайти максимальний потік між джерелом *s* та стоком *t*. Позначимо:

x_{ij} – величина потоку, що проходить вздовж дуги (i, j),

c_{ij} – пропускна здатність цієї дуги.

Для кожної проміжної вершини записуємо обмеження, що задає баланс потоку, який проходить через цю вершину у вигляді:

загальний вхідний потік = загальний вихідний потік.

Для кожної дуги записується обмеження, що потік не перевищує пропускної здатності дуги та є невід'ємним:

0 ≤ *потік по дузі* ≤ *пропускна здатність дуги*.

Функцією мети, яку потрібно максимізувати, є величина потоку, що виходить з джерела *s* або входить у сток *t*.

Приклад 8.2. Запишемо задачу з попереднього прикладу, як задачу лінійного програмування. Обмеження для кожної проміжної вершини та функцію мети наведено у табл. 8.1. Структура коефіцієнтів, що формують обмеження, має таку особливість: в стовпці, що відповідає змінній x_{ij}, завжди в рядку, що відповідає вершині i стоїть –1, а в рядку, що відповідає вершині j стоїть +1. Решта коефіцієнтів дорівнюють нулю. Така структура коефіцієнтів типова для мережевих моделей.

Таблиця 8.1

Дуги / *Вершини*	x_{12}	x_{13}	x_{14}	x_{24}	x_{25}	x_{34}	x_{36}	x_{42}	x_{43}	x_{45}	x_{46}	x_{52}	x_{54}	x_{56}	*Вільні члени*
	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	y_{10}	y_{11}	y_{12}	y_{13}	y_{14}	
2	**1**	**0**	**0**	**-1**	**-1**	**0**	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	= 0
3	**0**	**1**	**0**	**0**	**0**	**-1**	**-1**	**0**	**1**	**0**	**0**	**0**	**0**	**0**	= 0
4	**0**	**0**	**1**	**1**	**0**	**1**	**0**	**-1**	**-1**	**-1**	**-1**	**0**	**1**	**0**	= 0
5	**0**	**0**	**0**	**0**	**1**	**0**	**0**	**0**	**0**	**1**	**0**	**-1**	**-1**	**-1**	= 0
Пропускна здатність d_k	2	6	3	1	4	3	2	1	3	1	3	4	1	6	
Коефіцієнти функції мети F_1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	*Max*
Коефіцієнти функції мети F_2	0	0	0	0	0	0	1	0	0	0	1	0	0	1	*Max*
Оптимальний план	2	4	3	3	0	2	2	1	0	1	3	0	0	4	

Перепозначимо невідомі величини потоків y_k, пропускні здатності дуг d_k, $k=\overline{1,14}$. Отримаємо таку задачу лінійного програмування:

$$F_1 = y_1 + y_2 + y_3 \to \max \qquad F_2 = y_7 + y_{11} + y_{14} \to \max$$
$$A \cdot \bar{y} = \bar{0}, \quad \text{або} \quad A \cdot \bar{y} = \bar{0},$$
$$0 \le y_k \le d_k, \qquad 0 \le y_k \le d_k,$$

де A – матриця коефіцієнтів рівнянь-обмежень (2-5-ий рядок, 2-15-ий стовпець табл. 8.1), $\bar{y} = (y_1, y_2, \ldots, y_{14})$, $\bar{0} = (0,0,0,0)$, $k=\overline{1,14}$. Розв'язавши задачу засобами ***Maple*** (рис. 8.4) отримуємо оптимальні значення величин потоків $\bar{y} = (2,4,3,3,0,2,2,1,0,1,3,0,0,4)$.

```
> restart;
```
Підключаємо пакет ***simplex***
```
> with(simplex) :
```
Задаємо функцію мети L та систему обмежень -- нерівностей ***ineq***
```
> L := y1 + y2 + y3; ineq := {y1 - y4 - y5 + y8 + y12 = 0, y2 - y6 - y7 + y9 = 0, y3 + y5 + y6
      - y8 - y9 - y10 - y11 + y13 = 0, y4 + y10 - y12 - y13 - y14 = 0, y1 ≥ 0, y1 ≤ 2, y2 ≥ 0, y2
      ≤ 6, y3 ≥ 0, y3 ≤ 3, y4 ≥ 0, y4 ≤ 4, y5 ≥ 0, y5 ≤ 1, y6 ≥ 0, y6 ≤ 3, y7 ≥ 0, y7 ≤ 2, y8
      ≥ 0, y8 ≤ 1, y9 ≥ 0, y9 ≤ 3, y10 ≥ 0, y10 ≤ 1, y11 ≥ 0, y11 ≤ 3, y12 ≥ 0, y12 ≤ 4, y13
      ≥ 0, y13 ≤ 1, y14 ≥ 0, y14 ≤ 6};
                                  L := y1 + y2 + y3
ineq := {y2 - y6 - y7 + y9 = 0, y1 - y4 - y5 + y8 + y12 = 0, y4 + y10 - y12 - y13 - y14 = 0, y3
   + y5 + y6 - y8 - y9 - y10 - y11 + y13 = 0, 0 ≤ y1, 0 ≤ y2, 0 ≤ y3, 0 ≤ y4, 0 ≤ y5, 0 ≤ y6, 0
   ≤ y7, 0 ≤ y8, 0 ≤ y9, 0 ≤ y10, 0 ≤ y11, 0 ≤ y12, 0 ≤ y13, 0 ≤ y14, y1 ≤ 2, y2 ≤ 6, y3 ≤ 3, y4
   ≤ 4, y5 ≤ 1, y6 ≤ 3, y7 ≤ 2, y8 ≤ 1, y9 ≤ 3, y10 ≤ 1, y11 ≤ 3, y12 ≤ 4, y13 ≤ 1, y14 ≤ 6}
```

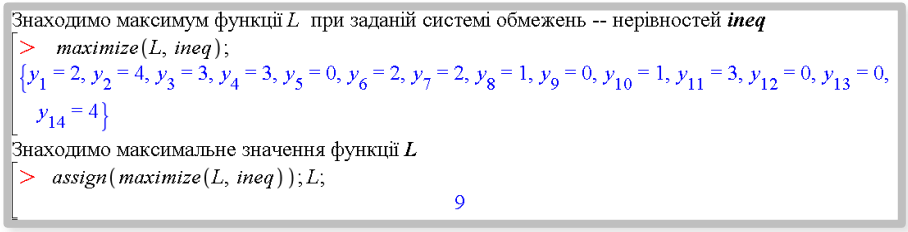


Рис. 8.4. Оптимальні значення величин потоків в пакеті ***Maple***

8.3. Задача про потік найменшої вартості

Задача **пошуку потоку найменшої вартості** в мережі з обмеженою пропускною здатністю узагальнює задачу визначення максимального потоку за такими параметрами:

1) кожній дузі поставлена у відповідність невід'ємна вартість проходження одиниці потоку по цій дузі;

2) дуги можуть мати нижнє додатне обмеження пропускної здатності;

3) будь-яка вершина мережі може бути як джерелом, так і стоком.

В **задачі про потік найменшої вартості** потрібно знайти такий розподіл потоків вздовж дуг, вартість якого є мінімальною, враховуючи обмеження на пропускні здатності дуг і на величини попиту і пропозиції окремих (чи всіх) вершин.

Розглянемо мережу G з обмеженою пропускною здатністю. Позначимо:

x_{ij} – величина потоку, що проходить по дузі (i, j);

l_{ij} – нижня межа пропускної здатності дуги (i, j);

u_{ij} – верхня межа пропускної здатності дуги (i, j);

c_{ij} – вартість проходження одиниці потоку по дузі (i, j);

f_j – величина потоку, що проходить через вершину j.

На рис. 8.5. наведено приклад позначення параметрів вершин та дуг на схемі мережі. Додатне значення $[f_j]$ вказує значення попиту, від'ємне – значення пропозиції.

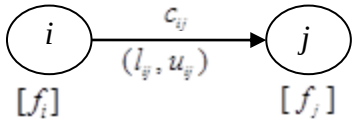


Рис. 8.5. Схема мережі з позначенням параметрів

8.4. Формалізація задачі про потік найменшої вартості як задачі лінійного програмування

Задачу про потік найменшої вартості можна подати у вигляді задачі лінійного програмування. Для кожної вершини записується обмеження, що задає баланс потоку, який проходить через цю вершину:

загальний вхідний потік − загальний вихідний потік = = потік, що проходить через вершину.

Для кожної дуги записується обмеження, що потік є не меншим, ніж нижня межа пропускної здатності та не перевищує верхньої межі пропускної здатності:

нижня межа ≤ потік по дузі ≤ верхня межа.

Використовуючи прийняті позначення, задачу лінійного програмування для мережі з обмеженою пропускною здатністю можна записати так:

$$F = \sum_{(i,j)\in E} c_{ij} x_{ij} \to \min,$$

$$\sum_{\substack{k \\ (j,k)\in E}} x_{jk} - \sum_{\substack{i \\ (i,j)\in E}} x_{ij} = f_j,$$

$$l_{ij} \le x_{ij} \le u_{ij},$$

де E – множина дуг.

Приклад 8.3. Для перевезення зерна з трьох зерносховищ до трьох агропромислових підприємств використовується залізничний та автомобільний транспорт. Можливі маршрути перевезень зображено на рис. 8.6. Пропозиція зерносховищ (пункти 1, 2, 3) становить відповідно 100, 200 та 50 тон, а попит агропромислових підприємств (пункти 4, 5, 6) – 150, 80 та 120 тонн. На маршрутах, де використовується автомобільний транспорт, є нижнє та верхнє обмеження пропускної здатності. Пропускна здатність залізничного транспорту практично необмежена. Вартість транспортування однієї тонни зерна на кожному маршруті (в сотнях у.о.) наведено біля відповідної дуги (рис. 8.6). Потрібно визначити план перевезення найменшої вартості.

Розв'язання.

Нехай x_{ij} – величина потоку, що проходить по дузі (i, j). Обмеження для кожної вершини та функцію мети наведено в таблиці 8.2.

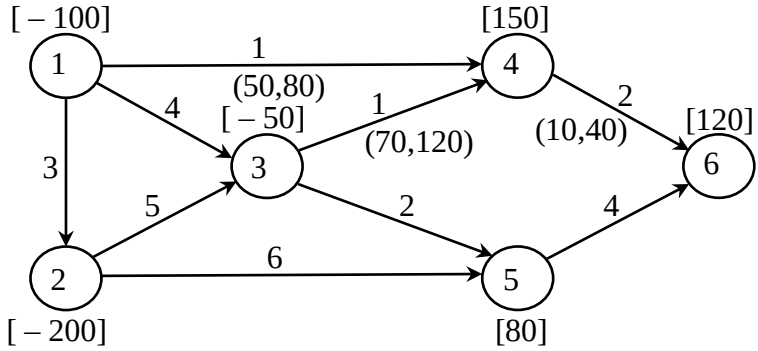


Рис. 8.6. Мережеві маршрути

Таблиця 8.2

Дуги / *Вершини*	x_{12}	x_{13}	x_{14}	x_{23}	x_{25}	x_{34}	x_{35}	x_{46}	x_{56}	*Вільні члени*
	y_1	y_2	y_3	y_4	y_5	y_6	y_7	y_8	y_9	
1	–1	–1	–1	0	0	0	0	0	0	–100
2	1	0	0	–1	–1	0	0	0	0	–200
3	0	1	0	1	0	–1	–1	0	0	–50
4	0	0	1	0	0	1	0	–1	0	150
5	0	0	0	0	1	0	1	0	–1	80
6	0	0	0	0	0	0	0	1	1	120
Нижня межа, l_k	0	0	50	0	0	70	0	10	0	
Верхня межа, u_k	∞	∞	80	∞	∞	120	∞	40	∞	
Коефіцієнти c_k *функції мети F*	3	4	1	5	6	1	2	2	4	min

Перенумеруємо невідомі величини потоків – y_k, коефіцієнти функції мети – c_k, нижні межі пропускної здатності дуг – l_k, верхні межі пропускної здатності дуг – u_k, $k=\overline{1,9}$. Отримаємо таку задачу лінійного програмування:

$$F=\sum_{k=1}^{9} c_k y_k \to \min,$$

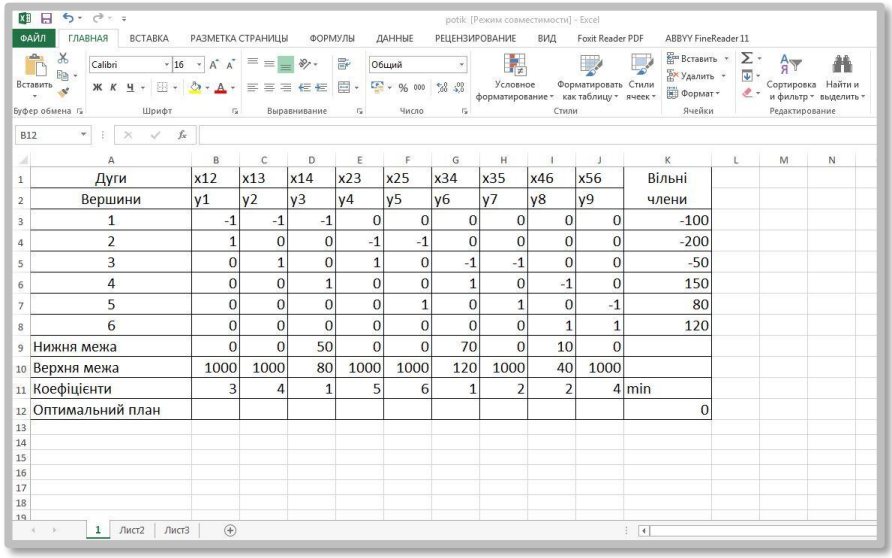
$$A\cdot \overline{y}=\overline{f},$$

$$l_k \le y_k \le u_k,\ k=\overline{1,9},$$

де A – матриця коефіцієнтів рівнянь-обмежень (2-7-ий рядок, 2-10-ий стовпець таблиці 8.2), $\overline{y}=(y_1,y_2,\ldots,y_9)$, $\overline{f}=(-100,-200,-50,150,80,120)$ – вектор вільних членів.

Розв'язавши задачу засобами ***Maple*** отримуємо оптимальні значення величин потоків $\bar{y} = (0, 20, 80, 40, 160, 110, 0, 40, 80)$. Загальна вартість перевезень становить 1830 у.о. (рис. 8.8).

Розв'яжемо задачу в ***Maple***. Вхідні дані таблиці 8.2 задамо у файлі ***Excel***, називаємо його potik.xls (рис. 8.7), а лист перейменовуємо на 1. У комірку K12 вводимо формулу =СУММПРОИЗВ(B11:J11;B12:J12), в якій буде обчислено найменшу вартість перевезень (F). Файл має бути розміщений у тій самій папці, що й файл програми ***Maple***. Значення $u_k = \infty$ замінено на достатньо велике скінчене значення, в нашому випадку 1000.



	A	B	C	D	E	F	G	H	I	J	K
1	Дуги	x12	x13	x14	x23	x25	x34	x35	x46	x56	Вільні
2	Вершини	y1	y2	y3	y4	y5	y6	y7	y8	y9	члени
3	1	-1	-1	-1	0	0	0	0	0	0	-100
4	2	1	0	0	-1	-1	0	0	0	0	-200
5	3	0	1	0	1	0	-1	-1	0	0	-50
6	4	0	0	1	0	0	1	0	-1	0	150
7	5	0	0	0	0	1	0	1	0	-1	80
8	6	0	0	0	0	0	0	0	1	1	120
9	Нижня межа	0	0	50	0	0	70	0	10	0	
10	Верхня межа	1000	1000	80	1000	1000	120	1000	40	1000	
11	Коефіцієнти	3	4	1	5	6	1	2	2	4	min
12	Оптимальний план										0

Рис. 8.7. Вхідні дані задачі

Після цього файл potik.xls закриваємо.

В ***Maple*** підключаємо необхідні пакети. Пакет ***ExcelTools*** допоможе нам отримати вхідні дані з таблиці ***Excel*** (рис. 8.7).

```
> restart :
> with(Optimization) : with(ExcelTools) :
```

Імпортуємо дані (табл. 8.2), що задані у файлі potik.xls (рис. 8.7).

Коефіцієнти рівнянь-обмежень:

```
> A := Import("potik.xls", "1", "B3:J8") :A := convert(A, Matrix);
```

$$A := \begin{bmatrix} -1.0 & -1.0 & -1.0 & 0. & 0. & 0. & 0. & 0. & 0. \\ 1.0 & 0. & 0. & -1.0 & -1.0 & 0. & 0. & 0. & 0. \\ 0. & 1.0 & 0. & 1.0 & 0. & -1.0 & -1.0 & 0. & 0. \\ 0. & 0. & 1.0 & 0. & 0. & 1.0 & 0. & -1.0 & 0. \\ 0. & 0. & 0. & 0. & 1.0 & 0. & 1.0 & 0. & -1.0 \\ 0. & 0. & 0. & 0. & 0. & 0. & 0. & 1.0 & 1.0 \end{bmatrix}$$

Коефіцієнти функції мети:

```
> c := Import("potik.xls", "1", "B11:J11") : c := convert(c, Vector);
```

$$c := \begin{bmatrix} 3.0 \\ 4.0 \\ 1.0 \\ 5.0 \\ 6.0 \\ 1.0 \\ 2.0 \\ 2.0 \\ 4.0 \end{bmatrix}$$

Вектор вільних членів:

```
> f := Import("potik.xls", "1", "K3:K8") : f := convert(f, Vector);
```

$$f := \begin{bmatrix} -100.0 \\ -200.0 \\ -50.0 \\ 150.0 \\ 80.0 \\ 120.0 \end{bmatrix}$$

Нижні межі обмежень:

```
> l := Import("potik.xls", "1", "B9:J9") : l := convert(l, Vector);
```

$$l := \begin{bmatrix} 0. \\ 0. \\ 50.0 \\ 0. \\ 0. \\ 70.0 \\ 0. \\ 10.0 \\ 0. \end{bmatrix}$$

Верхні межі обмежень:

```
> u := Import("potik.xls", "1", "B10:J10") : u := convert(u, Vector);
```

$$u := \begin{bmatrix} 1000.0 \\ 1000.0 \\ 80.0 \\ 1000.0 \\ 1000.0 \\ 120.0 \\ 1000.0 \\ 40.0 \\ 1000.0 \end{bmatrix}$$

Розв'язуємо задачу лінійного програмування у матричній формі. Отримуємо оптимальне значення функції мети та оптимальний розв'язок:

```
> Rozv := LPSolve(c, [NoUserValue, NoUserValue, A, f], [l, u]);
```

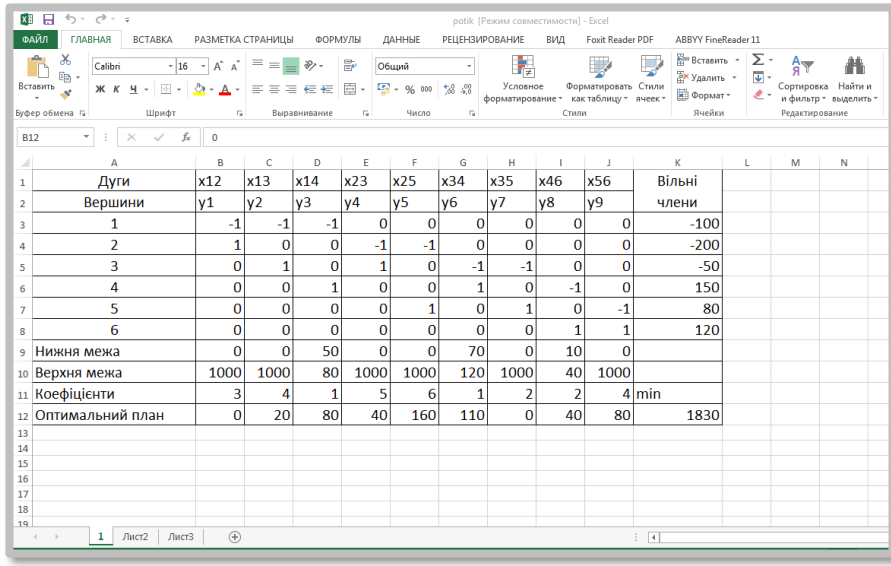
$$Rozv := \begin{bmatrix} 1830.00000000000, & \begin{bmatrix} 0. \\ 19.9999999999997016 \\ 80. \\ 40.0000000000002842 \\ 159.999999999999970 \\ 110.000000000000014 \\ 0. \\ 40. \\ 79.9999999999999574 \end{bmatrix} \end{bmatrix}$$

Експортуємо отриманий розв'язок у файл potik.xls, попередньо транспонуємо його (рис. 8.8).

```
> x := op(2, Rozv);
```

$$x := \begin{bmatrix} 0. \\ 19.9999999999997016 \\ 80. \\ 40.0000000000002842 \\ 159.999999999999970 \\ 110.000000000000014 \\ 0. \\ 40. \\ 79.9999999999999574 \end{bmatrix}$$

```
> x := convert(x, Vector[row]) :
> Export(x, "potik.xls", "1", "B12:J12") :
```



	A	B	C	D	E	F	G	H	I	J	K
1	Дуги	x12	x13	x14	x23	x25	x34	x35	x46	x56	Вільні
2	Вершини	y1	y2	y3	y4	y5	y6	y7	y8	y9	члени
3	1	-1	-1	-1	0	0	0	0	0	0	-100
4	2	1	0	0	-1	-1	0	0	0	0	-200
5	3	0	1	0	1	0	-1	-1	0	0	-50
6	4	0	0	1	0	0	1	0	-1	0	150
7	5	0	0	0	0	1	0	1	0	-1	80
8	6	0	0	0	0	0	0	0	1	1	120
9	Нижня межа	0	0	50	0	0	70	0	10	0	
10	Верхня межа	1000	1000	80	1000	1000	120	1000	40	1000	
11	Коефіцієнти	3	4	1	5	6	1	2	2	4	min
12	Оптимальний план	0	20	80	40	160	110	0	40	80	1830

Рис. 8.8. Таблиця розв’язку задачі про потік найменшої вартості

Контрольні запитання

1. Сформулюйте постановку задачі про максимальний потік.
2. Сформулюйте задачу про максимальний потік як задачу лінійного програмування.
3. Опишіть метод розв’язання задачі про максимальний потік.
4. Сформулюйте постановку узагальненої задачі пошуку потоку найменшої вартості.

ТЕМА 9. МЕРЕЖЕВЕ ПЛАНУВАННЯ ТА УПРАВЛІННЯ

9.1. Поняття мережевого планування та управління

Мережеве планування й управління – це сукупність розрахункових методів, організаційних і контрольних заходів щодо планування й управління комплексом робіт за допомогою мережевого графіка (мережевої моделі).

Методики мережевого планування були розроблені наприкінці 50-х років у США. У 1956 р. М. Уолкер та Д. Келлі спробували використати ЕОМ для складання планів-графіків великих комплексів робіт з модернізації заводів фірми "Дюпон". У результаті був створений раціональний і простий метод опису проекту з використанням ЕОМ. Пізніше метод отримав назву методу критичного шляху (CPM - Critical Path Method).

Паралельно і незалежно у військово-морських силах США був створений метод аналізу й оцінки програм PERT (Program Evaluation and Review Technique). Цей метод був створений для реалізації проекту розробки ракетної системи "Поларис", що поєднував близько 3800 основних підрядників і включав 60 тис. операцій, роботи велися на території 48 штатів. Використання методу PERT давало змогу керівництву програми точно знати, що потрібно робити в кожен момент часу і хто саме повинен це робити, а також імовірність своєчасного завершення окремих операцій. Завдяки розробленому методу проект вдалося завершити на два роки раніше запланованого терміну. Незабаром цей метод управління почали застосовувати для планування інших проектів Збройних сил США.

Великі промислові корпорації почали застосування подібної методики управління практично одночасно з військовими для розробки нових видів продукції і модернізації виробництва. Широкого застосування методики мережевого планування та управління набули в будівництві, наприклад, для керування проектом спорудження гідроелектростанції на ріці

Черчілль у Ньюфаундленді (півострів Лабрадор). Вартість проекту становила 950 млн доларів.

У теперішній час склалися глибокі традиції використання систем управління проектами в багатьох галузях життєдіяльності. Причому, основну частку серед планованих проектів становлять невеликі за розмірами проекти, що складаються з 500-1000 робіт.

9.2. Основні елементи мережевого планування та управління

Чим складніший запланований комплекс робіт (проект), тим складніші задачі оперативного планування, контролю й управління. Застосування мережевих моделей забезпечує продуману детальну організацію робіт, створює умови для ефективного керівництва, дає змогу формувати календарний план комплексу робіт, виявляти резерви: часові, трудові, матеріальні, грошові, прогнозувати і запобігати можливим зривам у ході робіт.

Щоб скласти план виконання проекту потрібно описати його за допомогою мережевої моделі (мережевого графіка). Мережева модель дає можливість формувати календарний план комплексу робіт, виявляти резерви часові, трудові, матеріальні, грошові, прогнозувати і уникати можливих зривів у ході робіт.

У мережевій моделі враховуються всі роботи від проектування до реалізації, визначаються найбільш важливі, критичні роботи, від виконання яких залежить термін закінчення проекту. У процесі виконання проекту є можливість коригувати план, забезпечувати безперервність в оперативному плануванні. Методи аналізу мережевого графіка дають змогу оцінити ступінь впливу внесених змін на хід здійснення програми, прогнозувати стан робіт на майбутнє. Мережевий графік точно вказує на роботи, від яких залежить термін виконання програми.

Під **комплексом робіт** або **проектом** ми будемо розуміти будь-яку задачу, для виконання якої необхідно здійснити досить велику кількість різноманітних робіт.

Головними елементами мережевої моделі є **роботи** й **події.**

Роботи поділяються на:

- реальні роботи – роботи, які потребують затрат часу, праці, ресурсів;
- очікувані роботи – роботи, які потребують лише затрат часу;
- фіктивні роботи – умовна залежність між іншими роботами.

Подія – це момент часу, коли закінчуються одні роботи та починаються інші.

Події поділяються на:

- вихідні – не мають попередніх робіт;
- проміжні – виконання цих робіт дає можливість перейти до виконання інших робіт;
- завершальні – не мають наступних робіт.

В мережевому графіку дуги – це роботи, що з'єднують певні події – вершини графа. Початок та закінчення будь-якої роботи описується парою подій, які називають початковими та кінцевими подіями для цієї роботи.

9.3. Порядок і правила побудови мережевих графіків

Мережеві графіки складають на початковому етапі планування. Спочатку запланований процес розбивають на окремі роботи, складають перелік робіт і подій, визначають їхні логічні зв'язки і послідовність виконання, роботи закріплюють за відповідальними виконавцями. З їхньою допомогою і за допомогою нормативів, якщо такі існують, оцінюється тривалість кожної роботи.

З метою впорядкування робіт будують **структурну таблицю.** У структурній таблиці для кожної роботи вказують, які роботи їй безпосередньо передують. Прочерк означає, що ця

робота може бути розпочата безпосередньо. Для упорядкування всі роботи поділяють на ранги:

- робота 1-го рангу – для її початку не потрібне виконання інших робіт;
- робота 2-го рангу – їй передує одна або декілька робіт 1-го рангу;
- робота k-го рангу – їй передує одна або декілька робіт не вище $(k-1)$-го рангу, серед яких є хоча б одна робота $(k-1)$-го рангу.

Після розподілу робіт за рангами роботам присвоюють нові номери, починаючи з робіт 1-го рангу і складають нову впорядковану структурну таблицю, де кожній роботі передують лише роботи з меншим порядковим номером.

На основі впорядкованої структурної таблиці складають мережевий графік. Після його упорядкування розраховують параметри подій і робіт, визначають резерви часу і критичний шлях. На основі побудованого мережевого графіка складають часовий графік реалізації проекту. Основні етапи мережевого планування зображено на рис. 9.1.

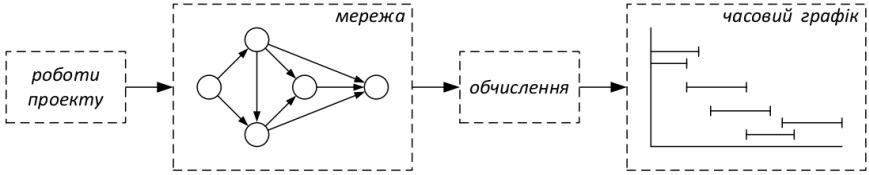


Рис. 9.1. Основні етапи мережевого планування

Для побудови мережевого графіка (рис. 9.2) необхідно дотримуватись таких правил:

- довжина дуги не залежить від часу виконання роботи;
- для дійсних робіт використовують суцільні дуги, а для фіктивних – пунктирні;
- варто уникати перетинання дуг;

- номер початкової події повинен бути меншим від номера кінцевої події;
- не використовувати:
  - кратних дуг;
  - дуг, спрямованих справа наліво;
  - подій, які не мають попередніх подій, крім вихідної;
  - подій, які не мають наступних подій, крім завершальної;
  - циклів.

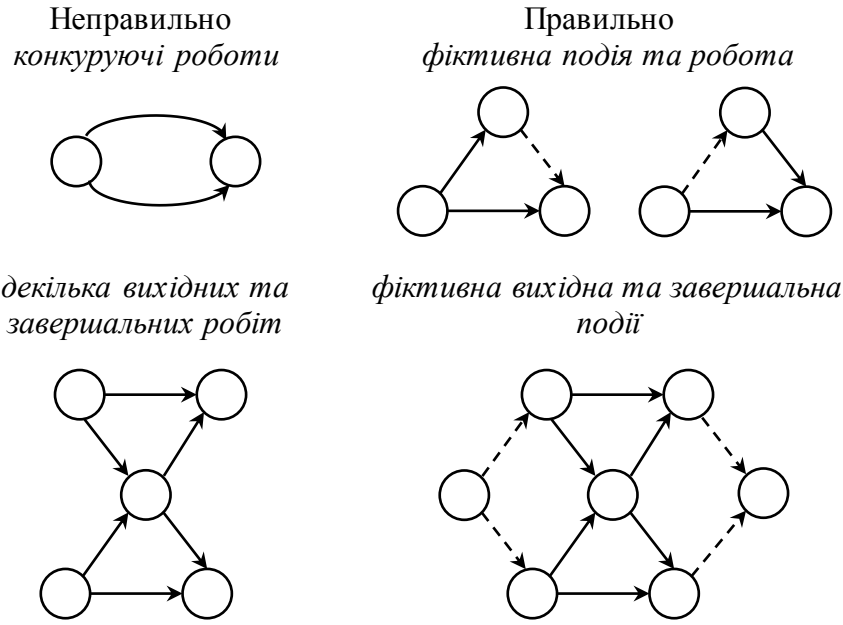


Рис. 9.2. Правильно та неправильно побудовані мережеві моделі

Приклад 9.1. Автотранспортне підприємство складає бюджет для перевезення додаткового виду товару. В таблиці 9.1

наведено етапи підготовки бюджету та їхню тривалість. Потрібно побудувати мережеву модель.

Таблиця 9.1

	Робота	**Спирається на роботи**	**Тривалість роботи**
A:	Прогнозування обсягу перевезень	_	10
B:	Вивчення ринку конкуруючих фірм	–	7
C:	Вибір маршруту	A	5
D:	Підготовка плану закупівель необхідної техніки	C	3
E:	Оцінка витрат	D	2
F:	Визначення цінової політики	B, E	1
G:	Підготовка бюджету	E, F	14

Розв'язання.

Використовуючи дані структурної таблиці отримаємо мережеву модель, зображену на рис. 9.3.

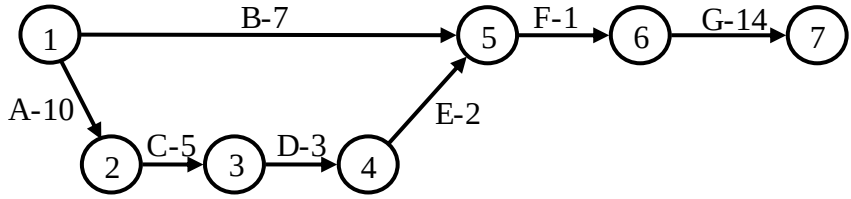


Рис. 9.3. Мережева модель

9.4. Критичний шлях

Нехай побудовано мережевий графік деякого комплексу робіт. Задано час виконання окремих робіт. За який час можна виконати всі роботи?

Для визначення мінімального часу, необхідного для виконання комплексу робіт, потрібно знайти шлях від вихідної події до завершальної із найбільшою тривалістю $L_{кр}$. Такий шлях називають **критичним шляхом**. Роботи та події, розташовані на цьому шляху, називають **критичними роботами та подіями**. Від тривалості критичних робіт залежить загальний термін закінчення всіх робіт. Скорочення або збільшення термінів виконання критичних робіт відповідно скорочує або збільшує загальний термін виконання всіх робіт.

Роботи, що не лежать на критичному шляху, називають **некритичними**. Некритичні роботи дозволяють деяке запізнення у їх виконанні, яке не призведе до зміни термінів виконання всього проекту.

Для знаходження критичного шляху будемо використовувати алгоритм, аналогічний алгоритму Дейкстри.

Крок 0. Позначаємо початкову вершину індексом $\lambda_0 = 0$.

Крок *k*. Позначаємо вершину k індексом $\lambda_k = \max\limits_{i<k}(\lambda_i + l_{i,k})$, де $l_{i,k}$ – час виконання роботи (i,k).

Круг, що зображає події, будемо розділяти на три сектори (рис. 9.4). В нижньому секторі будемо проставляти номер події, в лівому – індекс λ_k.

Приклад 9.2. В результаті аналізу комплексу робіт складено мережевий графік, який відображає порядок виконання робіт (рис. 9.4). Потрібно знайти критичний шлях і розрахувати його протяжність за часом.

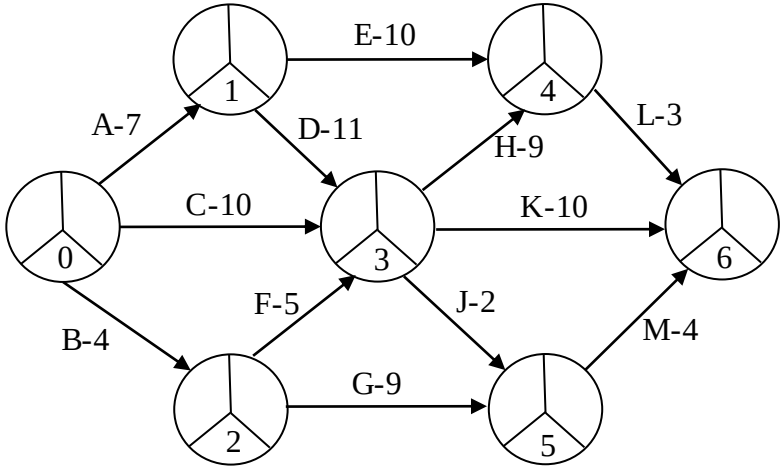


Рис. 9.4. Мережевий графік порядку виконання робіт

Розв'язання.

Позначимо нульову вершину індексом $\lambda_0 = 0$. Перша та друга вершини мають по одній вхідній дузі, тому $\lambda_1 = l_{0,1} = 7$, $\lambda_2 = l_{0,2} = 4$. Для решти подій скористаємось формулою

$$\lambda_k = \max_{i<k}(\lambda_i + l_{i,k}).$$

Отримані значення $\lambda_3 = 18$, $\lambda_4 = 27$, $\lambda_5 = 20$, $\lambda_6 = 30$ проставимо в лівому секторі вершин графа (рис. 9.5). Критичний шлях визначимо методом зворотнього ходу. Таким чином, критичним є шлях 0→1→3→4→6, його тривалість $L_{кр} = 30$, критичними є роботи A, D, H, L.

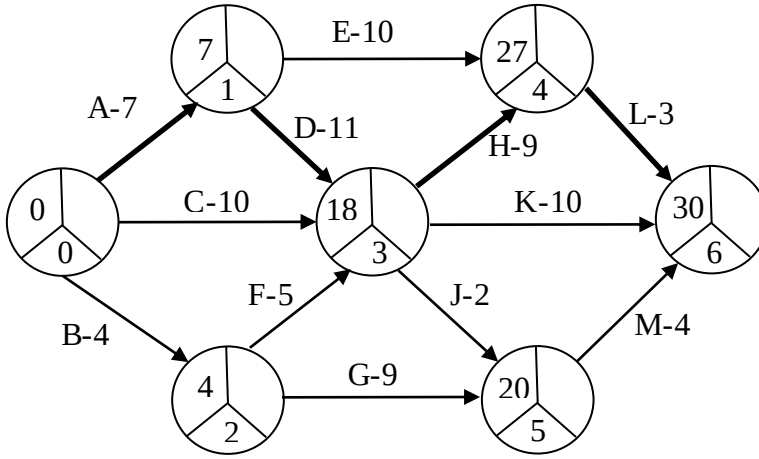


Рис. 9.5. Мережевий графік з індексами вершин

9.5. Параметри подій

Для характеристики подій використовують такі параметри, як **ранній термін настання події** $t_p(i)$, **пізній термін настання події** $t_n(i)$ та **резерв часу** $R(i)$.

Ранній термін настання події визначають величиною максимального шляху, що передує цій події $t_p(i) = \lambda_i$. На мережевому графіку його проставляють у лівому секторі круга події.

Пізній термін настання події, за якого запланований термін виконання проекту не зміниться, обчислюємо за формулою

$$t_n(i) = \min_{j>i}(t_n(j) - l_{i,j}),$$

де j – події, наступні за подією i. Цей параметр проставляють у правому секторі круга події.

Для подій, що лежать на критичному шляху ранні та пізні терміни настання події збігаються.

Резерв часу i-ої події $R(i)$ визначають за формулою

$$R(i) = t_n(i) - t_p(i).$$

Резерв часу вказує, на який допустимий період часу можна затримати настання цієї події, не збільшуючи терміну виконання комплексу робіт.

Поняття ранніх та пізніх термінів настання події відіграють важливу роль у процесі виконання проекту. Якщо всі події настають не пізніше за терміни $t_n(i)$, то проект виконується нормально. Якщо деяка подія настає пізніше ніж $t_n(i)$, то потрібно застосовувати додаткові заходи для прискорення виконання робіт в цій частині проекту.

Приклад 9.3. Для мережевого графіка (рис.9.5.) знайти пізні терміни настання подій та резерви часу для кожної події.

Розв'язання.

Для обчислення пізніх термінів настання подій $t_n(k)$ користуємось формулою

$$t_n(k) = \min_{j>k}(t_n(j) - l_{k,j}),$$

де j – це події, наступні за подією k. Отримані значення $t_n(6) = 30$, $t_n(5) = 26$, $t_n(4) = 27$, $t_n(3) = 18$, $t_n(2) = 13$, $t_n(1) = 7$, $t_n(0) = 0$ проставимо в правому секторі вершин графа (рис. 9.6).

Визначимо резерви подій. Критичні події не мають запасу часу, для решти робіт маємо $R(2) = 13 - 4 = 9$, $R(5) = 26 - 20 = 9$.

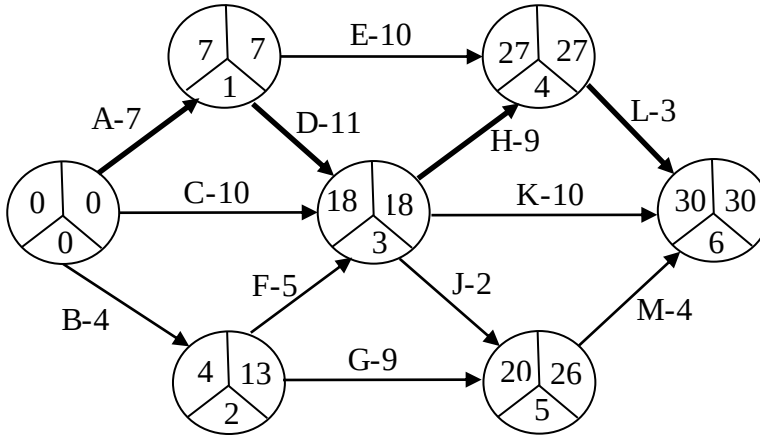


Рис. 9.6. Граф з позначеними параметрами вершин

9.6. Побудова часового графіка

Мережевий графік проекту будують не вказуючи масштаб часу. У зв'язку з цим мережевий графік хоч і дає уявлення про порядок виконання робіт, але є недостатньо наочним для визначення робіт, які виконуються в кожен момент часу. Тому, крім мережевого графіка проекту будують також **часовий графік** (діаграма Ганта). На осі абсцис часового графіка відкладають час, а на осі ординат – роботи. Критичні роботи будують суцільними лініями послідовно без часових щілин та перекривань. Таким чином, сумарна тривалість критичних робіт дорівнює тривалості виконання всього процесу.

Некритичні роботи зображають максимальними інтервалами виконання, які перевищують реальну тривалість виконання цих робіт. Некритичні роботи зображають пунктирними лініями.

Виникає питання: як вибрати час початку виконання некритичних робіт? Зазвичай некритичні роботи починають

якомога раніше. В такому випадку залишається запас часу, який можна використати для вирішення проблем, що неочікувано виникли під час виконання роботи. Разом з тим, за необхідності можна перенести початок виконання деякої роботи. Припустимо, що для двох некритичних робіт використовується одне і те ж обладнання. Тоді можна виключити часове накладання цих робіт, почавши одну з робіт (якщо це можливо) після завершення іншої. Також, якщо ми починаємо всі роботи якомога раніше, то в такому випадку може порушуватись черговість виконання робіт.

Побудуємо часовий графік для сітьового графіка, заданого на рис. 9.4. Критичні роботи будуємо послідовно одну за одною без часових щілин та перекривань суцільними лініями (рис. 9.7). Некритичні роботи зображаємо максимальними інтервалами виконання, які перевищують реальну тривалість виконання цих процесів. Для кожної некритичної роботи вибираємо початок виконання так, щоб або всі роботи починались якомога раніше, або щоб кількість робіт, що виконуються одночасно, була мінімальною (жирні лінії).

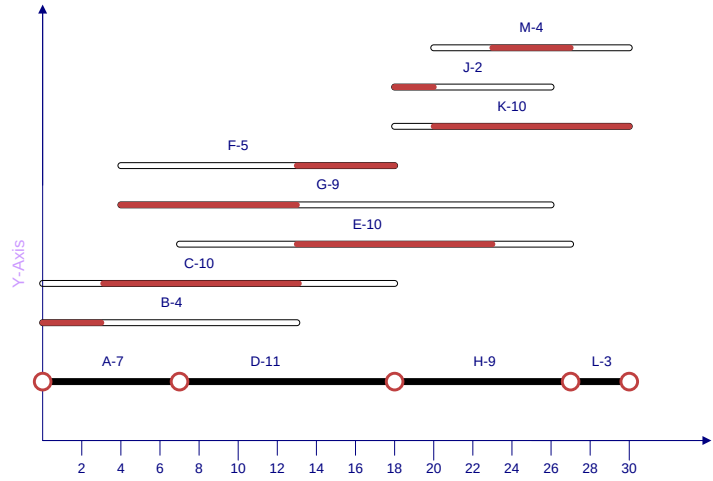


Рис. 9.7. Часовий графік для сітьового графа

Контрольні запитання

1. Що називають мережевим плануванням управління?
2. Які є основні методики мережевого планування управління?
3. Дайте означення основних елементів мережевого планування управління: подія, робота.
4. Опишіть особливості побудови мережевого графіка.
5. Опишіть правила складання структурної таблиці.
6. Що називають критичним шляхом?
7. Опишіть алгоритм знаходження критичного шляху у мережевому графіку.
8. Опишіть основні параметри подій.
9. Як вибрати час початку некритичної роботи?

ТЕМА 10. НЕЛІНІЙНЕ ПРОГРАМУВАННЯ

10.1. Основні поняття нелінійного програмування

Задачею нелінійного програмування називають таку задачу математичного програмування, у якій функція мети та/або система обмежень є нелінійними.

Виникають задачі нелінійного програмування в практичних завданнях досить часто. Наприклад, нехай критерієм оптимальності є собівартість одиниці виробленої продукції. Очевидно, що вона залежить від розміру підприємства. Так, із збільшенням обсягу продукції собівартість її зменшується. Проте таке зменшення не є безмежним. Настає такий момент, коли внутрішні витрати підприємства починають зростати (збільшуються витрати на перевезення, збереження продукції тощо), що у свою чергу призводить до збільшення собівартості. Функція, яка і спадає, і зростає, вже не може бути лінійною. Крім того, якщо врахувати в моделях лінійного програмування додаткові фактори, то ці моделі трансформуються в нелінійні. Наприклад, припустивши, що в задачі про використання ресурсів обсяг реалізації впливає на прибуток, маємо функцію мети з нелінійністю.

Класифікують задачі нелінійного програмування за наявністю обмежень (**задачі умовної та безумовної оптимізації**) та за кількістю змінних (**однопараметричні та багатопараметричні задачі оптимізації**).

На відміну від лінійного програмування, не існує загальних методів розв’язання задач нелінійного програмування. В кожному конкретному випадку метод вибирається з урахуванням виду функції мети (дробово-лінійна, квадратична та ін.). У випадку 2-х змінних для розв’язання задач нелінійного програмування можна використовувати графічний метод, аналогічно як у задачах лінійного програмування. Деякі задачі нелінійного програмування можна наблизити до задач лінійного програмування та знайти розв’язок, близький до оптимального.

Методи розв'язання задач нелінійного програмування перебувають у процесі розвитку, тому їх використання є обмеженим. Найчіткіше розроблено методи розв'язування нелінійних задач з опуклою областю допустимих розв'язків та опуклою чи увігнутою функцією мети. Такі задачі складають клас задач опуклого програмування і є окремим випадком задач нелінійного програмування. Методи розв'язання задач опуклого програмування можна умовно поділити на дві групи. До першої групи належать методи, які зводять задачу до задачі безумовної оптимізації. У цьому випадку розв'язок задачі знаходять як границю послідовності безумовних екстремумів за допомогою спеціально складених допоміжних функцій. До них включають **метод множників Лагранжа**, **метод штрафних функцій**, **метод центрів**.

Для знаходження стаціонарної точки цими методами треба розв'язати систему n нелінійних рівнянь з n змінними. Проте, розв'язування такої системи є складною задачею. Крім того, задачі, що трапляються на практиці, мають функцію мети не завжди диференційовну, що також утруднює процес розв'язування задачі.

До другої групи входять методи, що базуються на ідеї поступового наближення до точки екстремуму. При цьому має виконуватися умова монотонної зміни функції мети. Такі методи розв'язування нелінійних задач називають *ітеративними*. Основним класом цієї групи є градієнтні методи: метод покоординатного спуску (метод Гаусса-Зейделя), метод крутого спуску (метод Бокса-Уілсона) або метод найшвидшого спуску та інші.

У випадку, коли для задач нелінійного програмування не вдається побудувати ефективний алгоритм розв'язування, то використовують деякі штучні підходи:

- зведення нелінійної задачі до задачі лінійного програмування, якщо це можливо, не впливаючи на точність

і зміст розв'язку; це доречно у випадках слабкої нелінійної залежності;

- зведення нелінійної задачі до структури багатокрокових задач, що дає можливість використовувати метод динамічного програмування.

Розглянемо **метод множників Лагранжа** та **метод поділу відрізка навпіл**, як приклади методів першої та другої групи відповідно.

10.2. Метод множників Лагранжа

Метод множників Лагранжа є класичним методом знаходження екстремуму функції багатьох змінних у задачах нелінійного програмування з обмеженнями та без них. Основними вимогами до задачі нелінійного програмування, яку можна розв'язувати методом множників Лагранжа, є такі:

- функція мети та обмеження описуються неперервно-диференційовними функціями;
- обмеження є рівняннями;
- відсутні вимоги невід'ємності та цілочисельності розв'язку.

Нехай задано задачу нелінійного програмування

$$F(x_1, x_2, ..., x_n) \to \max(\min), \qquad (10.1)$$

$$g_i(x_1, x_2, .., x_n) = 0,\ i = \overline{1, m}, \qquad (10.2)$$

де $F(x_1, x_2, ..., x_n)$, $g_i(x_1, x_2, .., x_n)$, $i = \overline{1, m}$, – неперервно-диференційовні функції в деякій області $D \subset R^n$.

Нехай $X = (x_1, x_2, ..., x_n)$, $\lambda = (\lambda_1, \lambda_2, ..., \lambda_m)$. Для задачі (10.1)-(10.2) запишемо функцію

$$L(X, \lambda) = L(x_1, x_2, ..., x_n, \lambda_1, \lambda_2, ..., \lambda_m) =$$

$$= F(x_1, x_2, ..., x_n) + \sum_{i=1}^{m} \lambda_i\ g_i(x_1, x_2, ..., x_n),$$

яку називають **функцією Лагранжа**, величини λ_1, λ_2, …, λ_m – називають **множниками Лагранжа**.

Розв'язок шукають серед стаціонарних точок функції Лагранжа, тобто серед таких точок, в яких усі частинні похідні першого порядку функції $L(X,\lambda)$ дорівнюють нулю.

Знайдемо та прирівняємо до нуля частинні похідні $\frac{\partial L(X,\lambda)}{\partial x_j}$, $\frac{\partial L(X,\lambda)}{\partial \lambda_i}$, $j=\overline{1,n}$, $i=\overline{1,m}$. Одержимо систему рівнянь

$$\begin{cases} \dfrac{\partial F(x_1,x_2,...,x_n)}{\partial x_j}+\sum\limits_{i=1}^{m}\lambda_i\dfrac{\partial g_i(x_1,x_2,...,x_n)}{\partial x_j}=0, \quad j=\overline{1,n}, \\ g_i(x_1,x_2,...,x_n)=0, \quad i=\overline{1,m}. \end{cases} \tag{10.3}$$

Розв'язавши систему (10.3), отримаємо стаціонарні точки $X^0=(x_1^0,x_2^0,...,x_n^0)$, $\lambda^0=(\lambda_1^0,\lambda_2^0,...,\lambda_m^0)$ функції $L(X,\lambda)$, тобто точки, в яких можливий екстремум. Достатні умови екстремуму (дослідження знака другого диференціала $d^2L(X,\lambda)=\sum\limits_{i=1}^{n}\sum\limits_{j=1}^{m}\dfrac{\partial^2 L}{\partial x_i\partial x_j}dx_i dx_j$) дають відповідь на питання про існування екстремуму в знайдених стаціонарних точках.

Достатні умови умовного екстремуму. Нехай $X^0=(x_1^0,x_2^0,...,x_n^0)$, $\lambda^0=(\lambda_1^0,\lambda_2^0,...,\lambda_m^0)$ є стаціонарною точкою функції Лагранжа $L(X,\lambda)$. Тоді

1. *X^0 є точкою умовного максимуму функції* $F(x_1,x_2,...,x_n)$ за умов (10.2), якщо для довільних значень dx_1, …, dx_n, які всі одночасно не дорівнюють нулю і пов'язані між собою умовами

$$dg_i=\sum_{j=1}^{n}\frac{\partial g_i(X^0)}{\partial x_j}dx_j=0,\ i=\overline{1,m}\ , \tag{10.4}$$

другий диференціал функції Лагранжа є від'ємним: $d^2L(X^0, \lambda^0) < 0$.

2. X^0 *є точкою умовного мінімуму функції* $F(x_1, x_2, ..., x_n)$ за умов (10.2), якщо для довільних значень dx_1, …, dx_n, які всі одночасно не дорівнюють нулю і пов'язані між собою умовами (10.4), другий диференціал функції Лагранжа є додатнім: $d^2L(X^0, \lambda^0) > 0$.

3. X^0 *не є точкою умовного екстремуму функції* $F(x_1, x_2, ..., x_n)$ за умов (10.2), якщо другий диференціал функції Лагранжа може набувати як додатних, так і від'ємних значень, залежно від значень dx_1, …, dx_n пов'язаних між собою умовами (10.4).

4. Питання про наявність умовного екстремуму в точці X^0 залишається відкритим, якщо не справджується жодна з умов 1-3.

Зауваження. Система (10.3) в загальному випадку є системою з $n+m$ рівнянь та невідомих. Знаходження розв'язків такої системи є складною з математичної точки зору задачею.

Приклад 10.1. Дослідити функцію

$$F = F(x_1, x_2, x_3) = x_1x_2 + x_2x_3$$

на умовний екстремум методом множників Лагранжа, якщо обмеження мають вигляд: $x_1 + x_2 = 2$, $x_2 + x_3 = 2$.

Розв'язання.

Складемо функцію Лагранжа

$$L(X, \lambda) = L(x_1, x_2, x_3, \lambda_1, \lambda_2) =$$

$$= x_1x_2 + x_2x_3 + \lambda_1(x_1 + x_2 - 2) + \lambda_2(x_2 + x_3 - 2).$$

Знайдемо та прирівняємо до нуля частинні похідні $\frac{\partial L(X,\lambda)}{\partial x_j}$, $\frac{\partial L(X,\lambda)}{\partial \lambda_j}$. Одержимо систему рівнянь:

$$\begin{cases} \dfrac{\partial L}{\partial x_1} = x_2 + \lambda_1 = 0, \\ \dfrac{\partial L}{\partial x_2} = x_1 + x_3 + \lambda_1 + \lambda_2 = 0, \\ \dfrac{\partial L}{\partial x_3} = x_2 + \lambda_2 = 0, \\ \dfrac{\partial L}{\partial \lambda_1} = x_1 + x_2 - 2 = 0, \\ \dfrac{\partial L}{\partial \lambda_2} = x_2 + x_3 - 2 = 0. \end{cases}$$

Розв'язуємо систему методом виключення невідомих. Отримуємо стаціонарну точку $x_1 = x_2 = x_3 = 1$, при $\lambda_1 = \lambda_2 = -1$. Для того, щоб дослідити стаціонарну точку на екстремум, знайдемо другий диференціал функції Лагранжа:

$$\frac{\partial^2 L}{\partial x_1^2} = \frac{\partial^2 L}{\partial x_2^2} = \frac{\partial^2 L}{\partial x_3^2} = 0;\ \frac{\partial^2 L}{\partial x_1 \partial x_2} = 1;\ \frac{\partial^2 L}{\partial x_1 \partial x_3} = 0;\ \frac{\partial^2 L}{\partial x_2 \partial x_3} = 1;$$

$$d^2 L = 2dx_1 dx_2 + 2dx_2 dx_3.$$

Умови (10.4) мають вигляд:

$$dx_1 + dx_2 = 0;$$

$$dx_2 + dx_3 = 0;$$

звідки

$$dx_1 = -dx_2;$$

$$dx_3 = -dx_2.$$

Тоді

$$d^2L = -2dx_2dx_2 - 2dx_2dx_2 = -4dx_2^2 < 0.$$

Отже функція $F = F(x_1, x_2, x_3)$ досягає в точці $x_1 = x_2 = x_3 = 1$ свого умовного максимуму, $F_{\max} = 2$.

Зауваження. Якщо деякі змінні можна явно виразити через решту змінних з рівнянь (10.2), то **методом виключення** задачу можна спростити: звести до задачі безумовної оптимізації, зменшити кількість невідомих.

Наприклад, для попередньої задачі з рівнянь-обмежень маємо:

$$x_1 = 2 - x_2,$$

$$x_3 = 2 - x_2,$$

Тому $F = (2 - x_2) \cdot x_2 + (2 - x_2) \cdot x_2 = 4x_2 - 2x_2^2$.

Отримали функцію однієї змінної $F = F(x_2)$, для знаходження екстремуму якої скористаємось методами диференціального числення. Знайдемо стаціонарні точки цієї функції:

$$F' = 4 - 4x_2 = 0 \Rightarrow x_2 = 1.$$

Дослідимо знак похідної ліворуч і праворуч від стаціонарної точки (рис. 10.1).

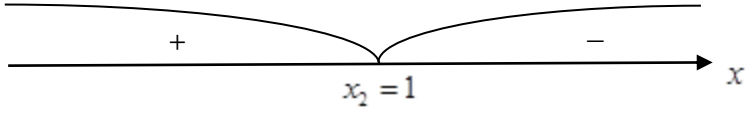


Рис. 10.1. Дослідження функції на екстремум

Отже, точка $x_2 = 1$ є точкою локального максимуму функції $F = F(x_2)$. Якщо $x_2 = 1$, то $x_1 = 1$, $x_3 = 1$. Таким чином, функція $F = F(x_1, x_2, x_3)$ досягає в точці $x_1 = x_2 = x_3 = 1$ свого умовного максимуму, $F_{\max} = 2$.

10.3. Метод поділу відрізка навпіл

Методом поділу відрізка навпіл можна знайти наближений розв'язок задачі однопараметричної безумовної оптимізації

$$F(x) \to \max(\min),$$

якщо відомий відрізок невизначеності $[a;b]$, на якому функція однієї змінної $y = F(x)$ має тільки один максимум (мінімум).

Ідея методу поділу відрізка навпіл полягає у поступовому звуженні відрізка невизначеності доти, поки із заданою точністю Δ не буде знайдена точка екстремуму (див. [15]).

Алгоритм методу поділу відрізка навпіл.

Нехай задано інтервал невизначеності $I_0 = [a;b]$ та необхідна точність Δ пошуку максимуму. Позначимо $I_{i-1} = [x_L; x_R]$ інтервал невизначеності на *i*-му кроці (на нульовому кроці $x_L = a$ і $x_R = b$).

Крок i. Обчислюємо

$$x_1 = \frac{x_R + x_L - \Delta}{2}; \qquad x_2 = \frac{x_R + x_L + \Delta}{2}.$$

1. Якщо $F(x_1) > F(x_2)$, то точка максимуму x^* лежить між x_L та x_2. Тоді покладаємо $x_R = x_2$ і одержимо новий інтервал невизначеності $I_i = [x_L; x_2]$ (рис. 10.2).

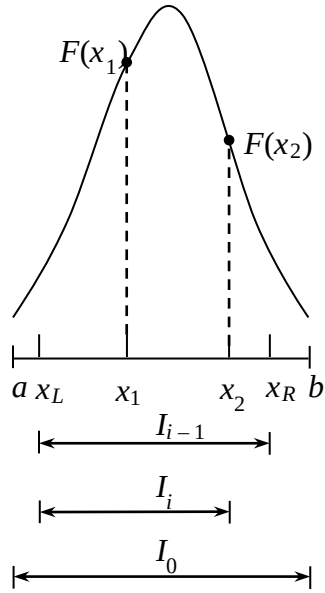


Рис. 10.2. Пошук інтервалу невизначеності

2. Якщо $F(x_1) < F(x_2)$, то точка максимуму x^* лежить між x_1 та x_R. Тоді покладаємо $x_L = x_1$ і $I_i = [x_1; x_R]$ (рис. 10.3).

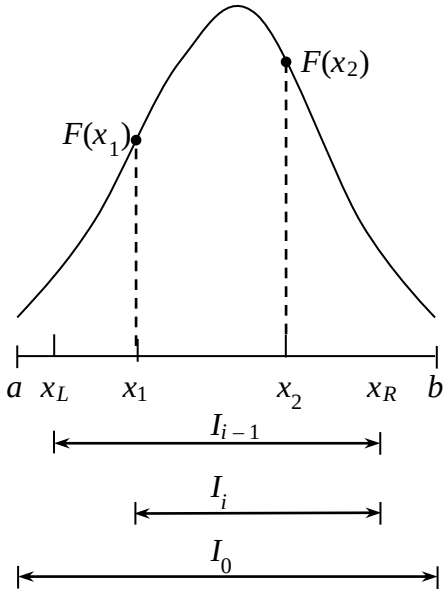


Рис. 10.3. Пошук інтервалу невизначеності

3. Якщо $F(x_1) = F(x_2)$, то точка максимуму x^* лежить між x_1 та x_2. Тоді покладаємо $x_L = x_1$, $x_R = x_2$ і $I_i = [x_1; x_2]$.

Оскільки на кожному кроці гарантовано $I_i \subset I_{i-1}$, в результаті звужується інтервал, в якому розташована точка максимуму функції $F(x)$.

Алгоритм закінчуємо на k-му кроці, коли довжина відрізка $I_k < \Delta$.

Зауваження. Алгоритм легко реалізувати програмними засобами, проте цей метод потребує досить великої кількості обчислень. Значно швидше знаходити розв'язок **методом золотого перерізу**, який відрізняється лише формулою для обчислення $x_1 = x_R - \frac{\sqrt{5}-1}{2}(x_R - x_L)$ та $x_2 = x_L + \frac{\sqrt{5}-1}{2}(x_R - x_L)$.

10.4. Розв'язання задач нелінійного програмування в пакеті Maple

Для розв'язання задач нелінійного програмування в середовищі ***Maple*** у пакеті **Optimization** призначено функції

***NLPSolve* (*obj, constr, bd, opts*)**

та

***QPSolve* (*obj, constr, bd, opts*).**

Параметрами функції ***NLPSolve* (*obj, constr, bd, opts*)** є:

obj – алгебраїчна цільова функція,

constr – множина обмежень,

bd – послідовність у вигляді **name = range,** яка задає межі змінних,

opts – рівняння у вигляді **option = value,** де **option** може набувати значень **assume**, **feasibilitytolerance**, **iterationlimit**, **maximize**,**method**, **optimalitytolerance** або **output**.

Функція ***QPSolve* (*obj, constr, bd, opts*)** має такі параметри:

obj – алгебраїчна, квадратична цільова функція,

constr – множина обмежень,

bd – послідовність у вигляді **name = range,** яка задає межі змінних,

opts – рівняння у вигляді **option = value,** де **option** може набувати значень **assume**, **feasibilitytolerance**, **iterationlimit**, **maximize** або **output.**

Задаючи параметри для функції ***NLPSolve*** можна вибрати метод пошуку розв'язку задачі нелінійного програмування. Більш детально з можливостями та параметрами функції пакета **Optimization** можна ознайомитись у довіднику системи ***Maple***.

За допомогою функції ***LagrangeMultipliers*** пакета **Student[MultivariateCalculus]** можна знайти стаціонарні точки функції Лагранжа.

Приклад 10.2. Методом множників Лагранжа знайти екстремум функції $F = 3x_1x_2 + 4$, при умові $\frac{x_1^2}{8} + \frac{x_2^2}{2} = 100$.

Розв'язання.

Розв'яжемо задачу в ***Maple*** (рис. 10.4).

```
> restart:
Підключаємо пакет
> with(Student[MultivariateCalculus]):
Задаємо функцію мети F та обмеження g
> F := 3·x1·x2 + 4; g := x1^2/8 + x2^2/2 - 100; X := [x1, x2];
```

$$F := 3\,x_1\,x_2 + 4$$

$$g := \frac{1}{8}x_1^2 + \frac{1}{2}x_2^2 - 100$$

$$X := [x_1, x_2]$$

```
Знаходимо стаціонарні точки функції Лагранжа
> LagrangeMultipliers(F, [g], X);
```

$$[20, 10], [-20, -10], [-20, 10], [20, -10]$$

```
> LagrangeMultipliers(F, [g], X, output = detailed);
```

$$[x_1 = 20, x_2 = 10, \lambda_1 = 6, 3\,x_1\,x_2 + 4 = 604], [x_1 = -20, x_2 = -10, \lambda_1 = 6, 3\,x_1\,x_2 + 4 = 604], [x_1 = -20, x_2 = 10, \lambda_1 = -6, 3\,x_1\,x_2 + 4 = -596], [x_1 = 20, x_2 = -10, \lambda_1 = -6, 3\,x_1\,x_2 + 4 = -596]$$

Рис. 10.4. Метод множників Лагранжа в пакеті ***Maple***

Отже, маємо чотири стаціонарні точки. Зауважимо, що детальний розв'язок отримано за допомогою опції **output = detailed.**

Для функції двох змінних розв'язок можна відобразити графічно (рис. 10.5).

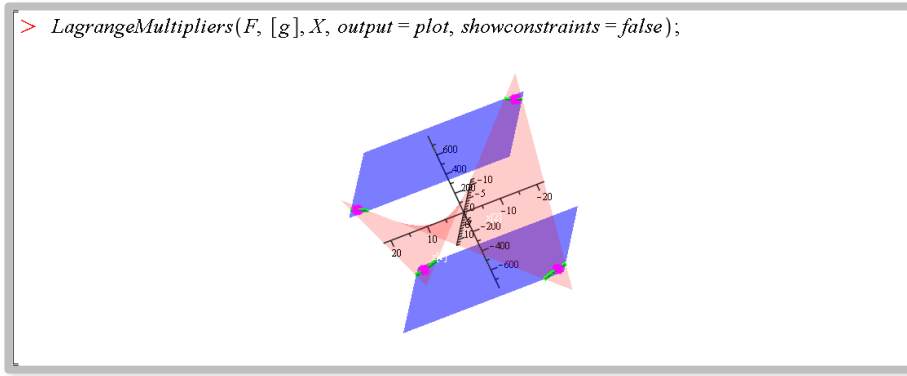


Рис. 10.5. Графічне зображення розв'язку

Отже, $F_{\max} = F(-20,-10) = F(20,10) = 604$,
$F_{\min} = F(-20,10) = F(20,-10) = -596$.

Приклад 10.3. Розвя'зати задачу нелінійного програмування

$$F = -x_1^2 - x_2^2 + 8x_1 + 10x_2 \to \max,$$
$$6 - 3x_1 - 2x_2 \geq 0,$$
$$x_1 \geq 0, \quad x_2 \geq 0.$$

Розв'язання.
Розв'яжемо задачу в ***Maple***. (рис. 10.6).

```
> restart :
Підключаємо пакет
> with(Optimization) :
Задаємо функцію мети F та обмеження g
> F := -x1^2 - x2^2 + 8·x1 + 10·x2; g := 6 - 3· x1 - 2·x2;
                    F := -x1^2 - x2^2 + 8 x1 + 10 x2
                    g := 6 - 3 x1 - 2 x2
```

Рис. 10.6. Розв'язок задачі нелінійного програмування в пакеті ***Maple***

Знаходимо максимум функції мети (опція **maximize**) для невід'ємних значень змінних (опція **assume = nonnegative**) (рис. 10.7).

```
Знаходимо максимум функції мети для невід'ємних значень змінних
> NLPSolve(F, {g ≥ 0}, assume = nonnegative, maximize);
    [21.3076923076923066, [x1 = 0.307692307692307986, x2 = 2.53846153846153788]]
```

Рис. 10.7. Максимум функції мети в пакеті ***Maple***

Отже, $F_{\max} = F(0,31; 2,54) \approx 21,31$.

Контрольні запитання

1. Сформулюйте задачу нелінійного програмування.
2. Як класифікують методи нелінійного програмування?
3. Сформулюйте ідею методу поділу відрізка навпіл.
4. Які задачі можна розв'язувати методом поділу відрізка навпіл?
5. Які недоліки має метод поділу відрізка навпіл?
6. Сформулюйте основні вимоги до задачі нелінійного програмування, яку можна розв'язувати методом множників Лагранжа.
7. Опишіть метод множників Лагранжа.
8. Які недоліки має метод множників Лагранжа?

ТЕМА 11. ДИНАМІЧНЕ ПРОГРАМУВАННЯ

11.1. Загальні поняття про задачі динамічного програмування

Динамічне програмування – це метод розв'язання задач математичного програмування, який ґрунтується на принципі поетапної, тобто послідовної оптимізації. Динамічне програмування визначає оптимальний розв'язок *n*-вимірної задачі шляхом її декомпозиції на *n* етапів, кожен з яких є підзадачею відносно однієї змінної. В такому випадку потрібно розв'язувати одновимірні оптимізаційні задачі замість великої *n*-вимірної.

Поділ на етапи здійснюють залежно від змісту задачі, розв'язок якої шукаємо. Наприклад, в задачах календарного планування етапи – це дискретні відрізки часу (доба, тиждень, місяць). В інших задачах розподіл процесу на кроки є умовним. Наприклад, у випадку розподілу ресурсів між роботами кількість етапів дорівнює кількості робіт.

Типовими задачами, що розв'язують методами динамічного програмування є:

- задача розподілу ресурсів (людських, енергетичних, сировини);
- задача про заміну обладнання;
- задачі про раціональне завантаження літака, вантажного автомобіля.

Задачі, які можна розв'язувати методами динамічного програмування, повинні мати такі властивості:

- задача повинна передбачати багатокрокову структуру;
- загальна ефективність розв'язування дорівнює сумі окремих ефективностей розв'язування на кожному *j*-му кроці оптимізації, тобто функція мети повинна бути адитивною

$$F = \sum_{j=1}^{n} f_j(x_j) \to extr;$$

- подальші пошуки оптимального розв'язку ведуться відносно стану об'єкта, якого він досяг на початку цього кроку оптимізації, і не залежать від того, яким чином цей об'єкт потрапив у цей самий стан. Таку умову називають **принципом оптимальності Беллмана.**

Переваги методу динамічного програмування:

- цей метод поки що єдиний для ефективного розв'язування задач багатокрокової структури, для яких немає інших практично-припустимих методів розв'язування;
- визначення глобального екстремуму не залежить від кількості локальних екстремумів;
- метод дає змогу вводити до моделі будь-які види обмежень, причому чим більше їх буде, тим швидше можна знайти оптимальний варіант, якщо існує область допустимих розв'язків;
- обчислювальна процедура методу дає змогу аналізувати знайдений розв'язок по етапах, переглядаючи динаміку поведінки об'єкта;
- використання методу не залежить від характеру функції мети, вона може бути недиференційовна, таблична, у вигляді графіка чи діаграми.

Недоліки методу динамічного програмування:

- метод не має універсального алгоритму, який, наприклад, є у лінійному програмуванні;
- формулювання задачі у термінах динамічного програмування досить трудомісткий етап, і від того, як вибрані кроки оптимізації, залежить якість розв'язування задачі, причому найбільш складний момент у процесі розв'язування задачі – складання функціонального рівняння, що передбачає принцип оптимальності.

Обчислення в динамічному програмуванні виконують рекурентно, тобто оптимальний розв'язок одного етапу використовується в якості вихідних даних наступного етапу.

Обчислення можуть проводитись від першого етапу до останнього – **метод прямої прогонки**, а можуть від останнього до першого – **метод зворотної прогонки**.

Основними елементами моделей динамічного програмування є:

- визначення етапів,
- знаходження альтернативних розв'язків на кожному етапі,
- визначення стану на кожному етапі.

Поняття «стану» змінюється залежно від ситуації, що моделюється. Продемонструємо методи динамічного програмування на прикладі задачі про завантаження та задачі про заміну обладнання.

11.2. Задача про завантаження

Припустимо, літак вантажопідйомністю W завантажують вантажами n видів. Максимізувати сумарний прибуток, який приносить перевезення всього вантажу.

Позначимо:

w_i – вага одного предмета вантажу i-го виду,

r_i – прибуток, який приносить перевезення одного предмету вантажу i-го виду,

m_i – невідома кількість предметів вантажу i-го виду, яку потрібно завантажити.

Математична модель цієї задачі має вигляд

$$F = \sum_{i=1}^{n} r_i m_i \to \max,$$

$$\sum_{i=1}^{n} w_i m_i \le W,$$

$$m_i \geq 0,\ m_i \in Z,\ i = \overline{1,n}.$$

Це задача цілочисельного лінійного програмування.

Три елементи моделі динамічного програмування визначаються таким чином:

1. Етап *i* відповідає предмету *i*-го типу ($i = \overline{1,n}$).
2. Альтернативи розв'язку на *i*-му етапі описуються кількістю m_i предметів *i*-го типу, які потрібно завантажити. Кількість предметів m_i належить проміжку $0 \leq m_i \leq \left[\frac{W}{w_i}\right]$, де $\left[\frac{W}{w_i}\right]$ – ціла частина числа $\frac{W}{w_i}$. Відповідний прибуток дорівнює добутку $r_i m_i$.
3. Стан $0 \leq x_i \leq W$ на *i*-му етапі виражає сумарну вагу предметів, рішення про завантаження яких прийнято на етапах *i*, *i*+1, … *n*.

Нехай $f_i(x_i)$ – максимальний сумарний прибуток від етапів *i*, *i*+1, … *n* на заданому стані x_i. Рекурентне рівняння методу зворотної прогонки визначають за допомогою такої двокрокової процедури:

1. $f_i(x_i) = \max\limits_{m_i,\, x_i}\{r_i m_i + f_{i+1}(x_{i+1})\}$, $i = \overline{1,n}$, де $f_{n+1}(x_{n+1}) = 0$.
2. Виразимо x_{i+1} через x_i. За означенням $x_i - x_{i+1}$ – вага вантажу, який завантажено на *i*-му етапі, тобто $x_i - x_{i+1} = w_i m_i$. Тому, $x_{i+1} = x_i - w_i m_i$,
   $f_i(x_i) = \max\limits_{m_i, x_i}\{r_i m_i + f_{i+1}(x_i - w_i m_i)\}$, $i = \overline{1,n}$.

Приклад 11.1. У літак з вантажопідйомністю чотири тонни завантажують предмети 3-ох типів. В таблиці 11.1 наведено дані про вагу одного предмета w_i (в тоннах) і прибуток r_i (в тис. ум.

од.), які одержують від одного завантаженого предмета. Як необхідно завантажити літак, щоб отримати максимальний прибуток?

Таблиця 11.1

Тип вантажу	Вага, w_i	Прибуток, r_i
1	2	31
2	3	47
3	1	14

Розв'язання.

Оскільки вага одного предмета w_i та максимальна вага $W = 4$ набувають цілих значень, то стан x_i – сумарна вага предметів, рішення про завантаження яких прийнято на етапах *i*, *i*+1, … *n*, може набувати лише цілочисельних значень.

Етап 3. Точна вага x_3, яка може бути завантажена на етапі № 3 (завантажуємо вантаж 3-го типу) невідома, але вона може набувати одного із значень 0,1,2,3,4 (оскільки $W = 4$ т.). Стани $x_3 = 0$ та $x_3 = 4$ виникають тоді, коли предмет 3-го типу зовсім не завантажується або завантажуються лише предмети 3-го типу. Інші значення x_3 (1, 2 та 3) означають часткове завантаження літака предметами 3-го типу.

Оскільки $w_3 = 1$, то $0 \le m_3 \le \left[\frac{4}{1}\right] = 4$. Це означає, що максимальна кількість одиниць цього типу, яка може бути завантажена, дорівнює 4. Тобто можливими значеннями m_3 будуть 0, 1, 2, 3, 4. Розв'язок є допустимим за умови: $w_3 m_3 \le x_3$. Отже, всі неприпустимі альтернативи потрібно виключити. Наступне рівняння є основою для порівняння альтернатив на 3-му етапі:

$$f_3(x_3) = \max_{m_3}\{r_3 m_3\} = \max_{m_3}\{14 m_3\}.$$

Запишемо таблицю допустимих розв'язків для кожного значення x_3.

x_3 \ m_3	$14m_3$					$f_3(x_3)$	m_3^*
	$m_3 = 0$	$m_3 = 1$	$m_3 = 2$	$m_3 = 3$	$m_3 = 4$		
0	0	–	–	–	–	0	0
1	0	14	–	–	–	14	1
2	0	14	28	–	–	28	2
3	0	14	28	42	–	42	3
4	0	14	28	42	56	56	4

Комірки зі знаком "–" не задовольняють умову $m_3 \le \dfrac{x_3}{w_3} = x_3$.

Етап 2. Стан x_2 на етапі № 2 виражає сумарну вагу предметів 2-го та 3-го типів, рішення про завантаження яких прийнято на етапах 2 та 3 відповідно. Оскільки $w_2 = 3$ та $0 \le m_2 \le \left[\dfrac{4}{3}\right] = 1$, то максимальна кількість одиниць вантажу 2-го типу, яка може бути завантажена, дорівнює 1.

Можливими значеннями m_2 будуть 0 або 1. Допустимі альтернативи для кожного стану x_2 визначаються нерівністю $m_2 \le \dfrac{x_2}{w_2} = \dfrac{x_2}{3}$ (комірки зі знаком "–" не задовольняють цієї умови). Тоді

$$f_2(x_2) = \max_{m_2,\, x_2}\{r_2 m_2 + f_3(x_3)\} = \max_{m_2,\, x_2}\{r_2 m_2 + f_3(x_2 - w_2 m_2)\} =$$

$$= \max_{m_2,\, x_2}\{47 m_2 + f_3(x_2 - 3m_2)\}.$$

x_2 \ m_2	$47m_2 + f_3(x_2 - 3m_2)$		$f_2(x_2)$	m_2^*
	$m_2 = 0$	$m_2 = 1$		
0	0+0	–	0	0
1	0+14	–	14	0
2	0+28	–	28	0
3	0+42	47+0	47	1
4	0+56	47+14	61	1

Етап 1. Стан x_1 на етапі № 1 виражає сумарну вагу предметів 1, 2-го та 3-го типів, рішення про завантаження яких прийнято на етапах 1, 2 та 3 відповідно. Оскільки $w_1 = 2$ та $0 \le m_1 \le \left[\frac{4}{2}\right] = 2$, то максимальна кількість одиниць вантажу 1-го типу, яка може бути завантажена, дорівнює 2.

Можливими значеннями m_1 будуть 0, 1, 2. Допустимі альтернативи для кожного стану x_1 визначаються нерівністю $m_1 \le \frac{x_1}{w_1} = \frac{x_1}{2}$. Тоді

$$f_1(x_1) = \max_{m_1,\, x_1} \{r_1 m_1 + f_2(x_2)\} = \max_{m_1,\, x_1} \{r_1 m_1 + f_2(x_1 - w_1 m_1)\} =$$

$$= \max_{m_1,\, x_1} \{31m_1 + f_2(x_1 - 2m_1)\}.$$

x_1 \ m_1	$31m_1 + f_2(x_1 - 2m_1)$			$f_1(x_1)$	m_1^*
	$m_1 = 0$	$m_1 = 1$	$m_1 = 2$		
0	0+0	–	–	0	0
1	0+14	–	–	14	0
2	0+28	31	–	31	1
3	0+47	31+14	–	47	0
4	0+61	31+28	62+0	62	2

Оптимальний розв'язок будується таким чином. З умови $W = 4$ випливає, що перший етап задачі при $x_1 = 4$ дає оптимальний розв'язок $m_1^* = 2$, який означає, що два предмети першого типу будуть завантажені у літак. Це завантаження залишає $x_2 = x_1 - w_1 m_1^* = x_1 - 2m_1^* = 4 - 2 \cdot 2 = 0$. Отже, на другому етапі оптимальний розв'язок при $x_2 = 0$, тобто $m_2^* = 0$. Аналогічно, $x_3 = x_2 - w_2 m_2^* = x_2 - 3m_2^* = 0 - 3 \cdot 0 = 0$, тобто на третьому етапі оптимальний розв'язок отримуємо при $x_3 = 0$, тобто $m_3^* = 0$.

Отже, оптимальний розв'язок $m_1^* = 2$, $m_2^* = 0$, $m_3^* = 0$, прибуток $F = 2 \cdot 31 = 62$.

Отримана таблиця на етапі 1 дає можливість визначити оптимальне завантаження у випадку, коли $W = 3$. Новий розв'язок тоді будується починаючи з $x_1 = 3$ на першому етапі.

11.3. Задача про заміну обладнання

Нехай потрібно визначити оптимальну політику заміни обладнання на наступні *n* років. На початку кожного року приймається рішення про заміну обладнання або експлуатацію його протягом наступного року.

Позначимо:

$r(t)$ – прибуток від експлуатації обладнання, що має *t* років,

$c(t)$ – витрати протягом року на обслуговування обладнання, що має вік *t* років,

$s(t)$ – залишкова вартість обладнання (вартість від продажу обладнання, яке експлуатувалося *t* років),

I – вартість нового обладнання, яка залишається незмінною.

Елементи моделі динамічного програмування визначаються таким чином:

1. Етап *i* відповідає порядковому номеру року ($i = \overline{1,n}$).
2. Альтернативами на *i*-му етапі є рішення про продовження експлуатації або заміну обладнання.
3. Стан на *i*-му етапі – це вік обладнання на початку *i*-го року.

Нехай $f_i(t)$ – максимальний прибуток, отриманий за роки від *i* до *n* за умови, що на початку *i*-го року маємо *t*-літнє обладнання. Рекурентне співвідношення матиме вигляд:

$$f_i(t) = \max \begin{Bmatrix} r(t) + f_{i+1}(t+1) - c(t), \text{ якщо експлуатуємо} \\ r(0) + s(t) + f_{i+1}(1) - c(0) - I, \text{ якщо заміняємо} \end{Bmatrix},$$

де $f_{n+1}(\bullet) \equiv 0.$

Приклад 11.2. Мікроавтобус експлуатується протягом 1 року. Компанія планує визначити оптимальну політику заміни мікроавтобуса протягом наступних 4 років. На початку кожного року може бути прийняте рішення про заміну мікроавтобуса новим. Вартість нової машини вважається незмінною і становить 9500 у. о. Компанія вимагає обов'язкової заміни мікроавтобуса, який експлуатується протягом 3-х років. Визначити оптимальний план заміни мікроавтобуса.

Термін експлуатації t	Прибуток $r(t)$, у.о.	Витрати на обслуговування $c(t)$, у.о.	Залишкова вартість $s(t)$, у.о.
0	2500	750	9500
1	2200	825	6800
2	1960	910	5840
3	1770	990	5070

Розв'язання.

Для визначення допустимих значень віку мікроавтобуса побудуємо граф станів (рис. 11.1).

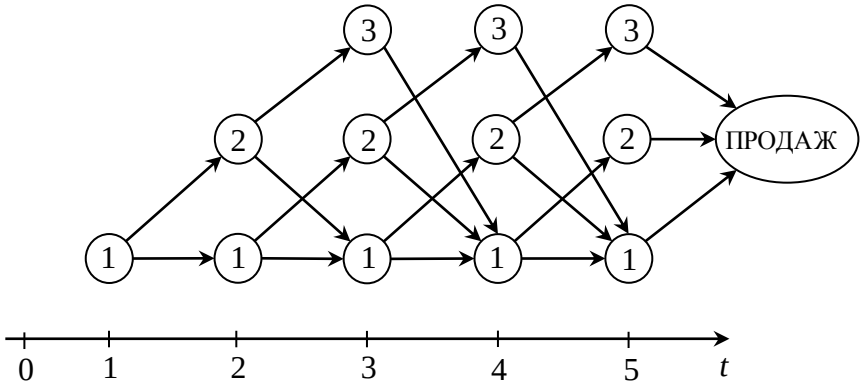


Рис. 11.1. Граф станів

На початку першого року маємо мікроавтобус, який уже експлуатується один рік. Ми можемо замінити (З) його або продовжити експлуатацію (Е) на наступний рік. Якщо мікроавтобус замінили, то на початку другого року йому буде один рік, інакше його вік буде 2 роки. Такий підхід використовуємо на початку кожного року. Розв'язання задачі еквівалентне знаходженню маршруту максимальної довжини (тобто такого, який приносить максимальний прибуток) від початку першого до кінця четвертого в мережі. Для розв'язання цієї задачі використовуємо табличну форму запису.

Етап 4.

t	**Експлуатація (Е)**	**Заміна (З)**	**Оптимум**	
	$r(t) + s(t+1) - c(t)$	$r(0) + s(t) + s(1) - c(0) - I$	$f_4(t)$	**Рішення**
1	2200+5840– –825=7215	2500+6800+6800– –750–9500=5850	7215	Е
2	1960+5070– –910=6120	2500+5840+6800– –750–9500=4890	6120	Е
3	–	2500+5070+6800– –750–9500=4120	4120	З

Етап 3.

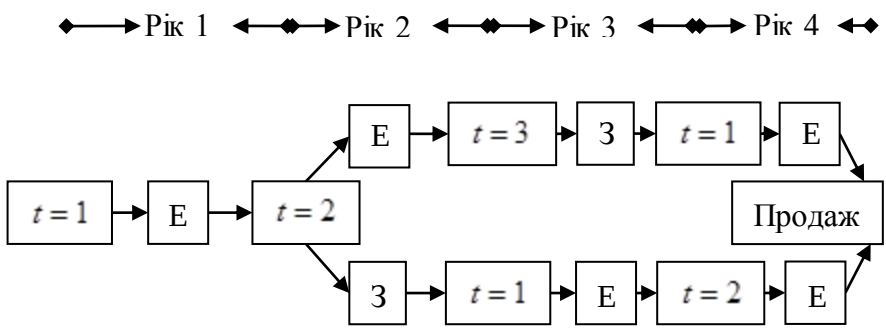
t	**Експлуатація (Е)**	**Заміна (З)**	**Оптимум**	
	$r(t) + f_4(t+1) - c(t)$	$r(0) + s(t) + f_4(1) - c(0) - I$	$f_3(t)$	**Рішення**
1	2200+6120– –825=7495	2500+6800+7215– –750–9500=6265	7495	Е
2	1960+4120– –910=5170	2500+5840+7215– –750–9500=5305	5305	З
3	–	2500+5070+7215– –750–9500=4535	4535	З

Етап 2.

t	**Експлуатація (Е)**	**Заміна (З)**	**Оптимум**	
	$r(t) + f_3(t+1) - c(t)$	$r(0) + s(t) + f_3(1) - c(0) - I$	$f_2(t)$	**Рішення**
1	2200+5305– –825=6680	2500+6800+7495– –750–9500=6545	6680	Е
2	1960+4535– –910=5585	2500+5840+7495– –750–9500=5585	5585	Е або З

Етап 1.

t	**Експлуатація (Е)**	**Заміна (З)**	**Оптимум**	
	$r(t) + f_2(t+1) - c(t)$	$r(0) + s(t) + f_2(1) - c(0) - I$	$f_1(t)$	**Рішення**
1	2200+5585– –825=6960	2500+6800+6680– –750–9500=5730	6960	Е



Отже, починаючи з першого року експлуатації мікроавтобуса, альтернативними оптимальними стратегіями

відносно заміни мікроавтобуса будуть (Е, Е, З, Е) і (Е, З, Е, Е). Загальний прибуток становитиме 6960 у.о.

Контрольні запитання

1. Що таке динамічне програмування?
2. Яким вимогам має задовольняти задача динамічного програмування?
3. Сформулюйте переваги та недоліки методу динамічного програмування.
4. Опишіть основні елементи моделі динамічного програмування.
5. Які типові задачі дослідження операцій розв'язуються методом динамічного програмування?
6. Як визначаються етапи в задачі про завантаження? Про заміну обладнання?
7. Сформулюйте та знайдіть розв'язок прикладу 11.2 як задачі про пошук найкоротшого маршруту.

ТЕМА 12. БАГАТОКРИТЕРІАЛЬНА ОПТИМІЗАЦІЯ

12.1. Суть задачі багатокритеріальної оптимізації

Моделі математичного програмування, розглянуті в попередніх темах, передбачали оптимізацію лише однієї функції мети (один критерій). На практиці реальна задача характеризується кількома показниками ефективності. Так, часто зустрічається вираз «досягти максимального ефекту при найменших витратах», який вже означає прийняття рішення при двох критеріях. Оцінка діяльності підприємств та планування як системи прийняття рішень проводиться на основі більше десятка критеріїв: виконання плану виробництва за обсягом, за номенклатурою, плану реалізації, прибутку за показниками рентабельності, продуктивності праці тощо. Отже, в реальній постановці, задачі є багатокритеріальними.

У теорії багатокритеріальної оптимізації шукають оптимальні рішення за кількома критеріями. Задачу багатокритеріальної оптимізації формулюють таким чином: потрібно знайти числа x_1, $x_2, \ldots,$ x_n, які максимізують (мінімізують) функції мети

$$\begin{aligned} & f_1(x_1, x_2, ..., x_n) \to \max(\min); \\ & f_2(x_1, x_2, ..., x_n) \to \max(\min); \\ & \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots; \\ & f_m(x_1, x_2, ..., x_n) \to \max(\min); \end{aligned} \qquad (12.1)$$

за заданих обмежень

$$\begin{cases} g_1(x_1, x_2, ..., x_n) \le b_1; \\ g_2(x_1, x_2, ..., x_n) \le b_2; \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots; \\ g_k(x_1, x_2, ..., x_n) \le b_k. \end{cases} \qquad (12.2)$$

Оскільки задачу пошуку мінімуму функції завжди можна

звести до задачі пошуку максимуму функції, надалі будемо розглядати лише задачі максимізації функцій мети.

Множина точок $X = (x_1, x_2, ..., x_n)$, що задовольняють систему обмежень (12.2), утворюють допустиму область $D \subset R^n$.

У векторній формі математичну модель багатокритеріальної оптимізації (12.1) - (12.2) можна записати такою формулою:

$$\vec{f}(X) = (f_1(X), ..., f_m(X)) \to \max \text{ при } X \in D, \qquad (12.3)$$

де $\vec{f}(X)$ – вектор-функція аргументу $X \in D$.

Вперше проблема багатокритеріальної оптимізації виникла у італійського економіста В. Парето в 1904 р. під час математичного дослідження товарного обміну. Надалі інтерес до проблеми багатокритеріальної оптимізації посилився у зв'язку з розробкою і використанням обчислювальної техніки, і вже пізніше стало зрозуміло, що багатокритеріальні задачі виникають також і в техніці, наприклад, під час проектування складних технічних систем.

В теорії багатокритеріальної оптимізації можна виділити такі основні напрямки розвитку:

1. розробка концепції оптимальності;
2. доведення існування розв'язку, оптимального у відповідному сенсі;
3. розробка методів знаходження оптимального розв'язку.

Якщо функції f_1, f_2,…, f_m досягають максимумів в одній і тій самій точці $X^* \in D$, то кажуть, що задача (12.1) - (12.2) має **ідеальне рішення**.

Випадки існування ідеального рішення в багатокритеріальної задачі дуже рідкісні. У разі відсутності «ідеального рішення» в задачі (12.1) - (12.2) шукають **компромісне рішення**. Однією з проблем в задачах

багатокритеріальної оптимізації є формалізація принципу оптимальності, тобто визначення того, в якому сенсі одне рішення є кращим за інше.

12.2. Оптимальність за Парето

Нехай X_1, $X_2 \in D$. Якщо для всіх критеріїв f_1, f_2,…, f_m мають місце нерівності $f_j(X_2) \geq f_j(X_1)$, $j = \overline{1,m}$, причому хоча б одна нерівність строга, то говорять, що рішення X_2 **домінує над рішенням** X_1. Відношення переваги прийнято позначати у вигляді $X_2 \succ X_1$. Рішення X_1 можна відкинути, як неконкурентоспроможне. Внаслідок такого відкидання "гірших" розв'язків множина так званих ефективних допустимих рішень значно звужується.

У задачі багатокритеріальної оптимізації точку $X_0 \in D$ називають **оптимальною за Парето**, якщо не існує іншої точки $X \in D$, яка б домінувала над X_0. Точки, оптимальні за Парето, утворюють множину точок, оптимальних за Парето (множину ефективних точок) $D_p \subset D$.

Оптимальні рішення багатокритеріальної задачі слід шукати лише серед елементів множини D_p. У цій області жоден критерій не може бути поліпшений без погіршення хоча б одного з інших. Важливою властивістю множини Парето є можливість позбутися заздалегідь невдалих рішень, які поступаються іншим за всіма критеріями.

Приклад 12.1. Проілюструємо вибір рішень, оптимальних за Парето, на прикладі задачі з двома критеріями f_1 та f_2 (обидва максимізуються).

Розв'язання.

Нехай множина D складається зі скінченої кількості рішень X_1, X_2,…, X_{10}. Кожному рішенню відповідає пара показників f_1 та f_2. Будемо зображати рішення точкою на площині з координатами f_1, f_2 і пронумеруємо точки відповідно до номера рішення (рис. 12.1).

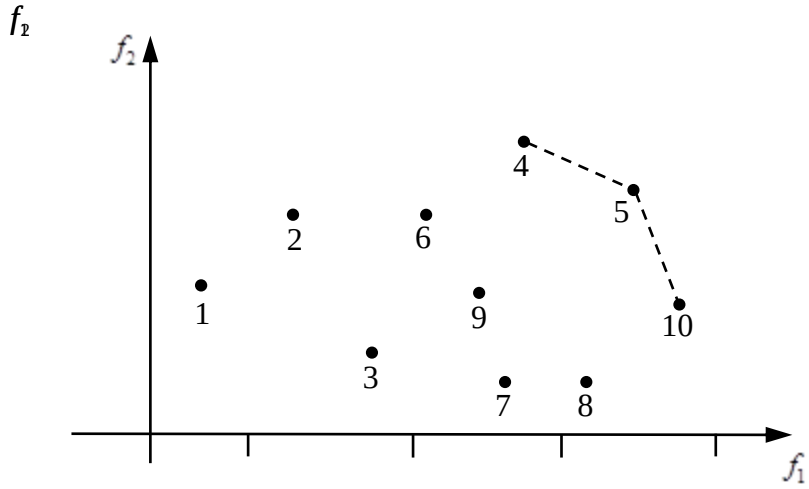


Рис. 12.1. Зображення рішень точками на площині

Оптимальними за Парето будуть точки X_4, X_5, X_{10}, які лежать на правій верхній межі області: X_4 – найкращий розв'язок за критерієм f_2, X_{10} – найкращий розв'язок за критерієм f_1. Для X_4, X_5, X_{10} не існує домінуючих рішень. Для решти точок існує принаймні одне рішення, для якого показник ефективності або f_1, або f_2, або обидва більше ніж для даної точки. Саме серед рішень, що відповідають оптимальним за Парето точкам, потрібно вибрати оптимальне, за певними міркуваннями, рішення. Цей вибір має робити особа, яка приймає рішення та є відповідальною за цей вибір.

12.3. Методи розв'язання задач багатокритеріальної оптимізації

Для розв'язування задач багатокритеріальної оптимізації існує кілька десятків методів та їх модифікацій. Розглянемо детальніше окремі методи, а саме:

- метод вагових коефіцієнтів;
- метод послідовних поступок;
- метод мінімізації загальної поступки.

Ці методи є різними за своєю природою і загалом дають розв'язки, що не збігаються між собою. Разом з тим не можна стверджувати, що один з методів є кращим за інші; по суті наведені методи призначені для розв'язування задач з різними перевагами в процесі прийняття рішень.

В **методі вагових коефіцієнтів** формується єдина функція мети у вигляді зваженої суми вихідних функцій мети. Нехай w_i, $i=\overline{1,m}$ – додатний ваговий коефіцієнт, який відображає важливість кожної функції мети, причому $\sum_{i=1}^{m} w_i = 1$.

Узагальнена функція мети запишеться так:

$$F = w_1 f_1 + w_2 f_2 + ... + w_m f_m \to \max .$$

Встановлення важливості кожної функції мети є дуже суб'єктивним. До тепер розроблено методи, які зменшують суб'єктивний фактор під час визначення вагових коефіцієнтів. Одним з недоліків методу вагових коефіцієнтів є те, що недостатня ефективність одного критерію компенсується іншим критерієм.

В **методі послідовних поступок** критеріям вихідної задачі присвоюють ранг, який вказує на важливість критерію. Максимізують перший за рейтингом критерій. Потім призначають величину припустимого зниження значення цього критерію δ і максимізують другий за важливістю критерій за

умови, що значення першого критерію не повинне відрізнятися від максимального більше, ніж на величину встановленого зменшення (поступки). Знову призначають величину поступки, але вже за другим критерієм і знаходять максимум третього за важливістю критерію за умови, що значення перших двох критеріїв не відрізнялися від раніше знайдених максимальних значень більше ніж на величини відповідних поступок. Аналогічно максимізують інші критерії. Тобто, на будь-якому етапі i при розв'язку задачі лінійного програмування з'являється $(i-1)$ додаткових обмежень:

$$\begin{cases} f_1(x_1, x_2, ..., x_n) \geq M_1 - \delta_1; \\ f_2(x_1, x_2, ..., x_n) \geq M_2 - \delta_2; \\ .....; \\ f_{i-1}(x_1, x_2, ..., x_n) \geq M_{i-1} - \delta_{i-1}, \end{cases}$$

де M_j – максимальне значення функції мети F_j, знайдене на етапі j, $j = \overline{1, i-1}$, δ_j – величини припустимого зниження значення j-го критерію.

Для використання методу послідовних поступок потрібно попередньо нормалізувати критерії. Основним недоліком методу послідовних поступок є суб'єктивність призначення і узгодження величин поступок, а також необхідність формування незмінного для всієї задачі апріорного ранжування критеріїв.

В методі **мінімізації загальної поступки** при заданих обмеженнях визначаються оптимальні значення кожної функції мети окремо (незалежна оптимізація). Потім шукають мінімальне значення загальної поступки z, при якому кожна функція мети $f_j(x_1, x_2, ..., x_n)$ на множині допустимих рішень відхиляється від свого оптимального значення $f_{j,\max}$ не більше, ніж на величину загальної поступки. Тобто, при заданих обмеженнях (12.2) розв'язується задача оптимізації $z \to \min$.

$$\begin{cases} f_1(x_1, x_2, ..., x_n) \geq f_{1,\max} - f_{1,\max} \cdot z; \\ f_2(x_1, x_2, ..., x_n) \geq f_{2,\max} - f_{2,\max} \cdot z; \\ .....; \\ f_n(x_1, x_2, ..., x_n) \geq f_{n,\max} - f_{n,\max} \cdot z; \\ z \geq 0. \end{cases}$$

Приклад 12.2. Автотранспортне підприємство розробляє план доставки вугілля та руди до металургійного комбінату. Для перевезення 100 т вугілля потрібно 12 автомобілів, а для 100 т руди потрібно 7 автомобілів. Максимальна кількість автомобілів, яку можна залучити до перевезення, становить 127. Розвантажують вугілля і руду в приймальному пункті; тривалість цієї роботи становить 16 год. 100 т вугілля розвантажують за 0,5 год., 100 т руди – 1,4 год. За будь-яких умов потрібно перевезти 200 т вугілля та 400 т руди. За перевезення 100 т вугілля автотранспортне підприємство отримує дохід 12 тис. у.о., за 100 т руди – 7 тис. у.о. Експлуатаційні витрати на перевезення 100 т вугілля становлять 7 тис. у.о., на 100 т руди – 5 тис. у.о. Знайти такий план перевезень, при якому загальний обсяг перевезень та дохід від перевезень будуть максимальними, а експлуатаційні витрати – мінімальними.

Встановлено, що найбільш важливою є кількість перевезеного вантажу ($w_1 = 0,5$), а найменш важливою – собівартість перевезень ($w_3 = 0,2$). Максимальне зниження кількості перевезеного вантажу не має перевищувати 10%, максимальне зниження доходу – 15% за умови, що оптимізується лише одна відповідна цільова функція.

Розв'язання.

Позначимо x_1 – кількість перевезеного вугілля (сотень т.), а x_2 – кількість перевезеної руди (сотень т.). Тоді задачу можна записати так:

$$f_1 = x_1 + x_2 \to \max,$$
$$f_2 = 12x_1 + 7x_2 \to \max,$$
$$f_3 = 7x_1 + 5x_2 \to \min,$$
$$\begin{cases} 12x_1 + 7x_2 \le 127; \\ 0,5x_1 + 1,4x_2 \le 16; \\ x_1 \ge 2; \\ x_2 \ge 4. \end{cases}$$

Розв'яжемо задачу методом вагових коефіцієнтів (рис. 12.2), методом послідовної поступки (рис. 12.3), методом мінімізації загальної поступки (рис. 12.4) та порівняємо отримані результати. Обчислення зробимо в пакеті ***Maple***. Функцію f_3 помножимо на (-1) і зведемо задачу лише до максимізації заданих функцій мети.

МЕТОД ВАГОВИХ КОЕФІЦІЄНТІВ

```
> restart:
```

Задаємо функції мети, обмеження та вагові коефіцієнти

```
> f1 := x1 + x2; f2 := 12·x1 + 7·x2; f3 := -1·(7·x1 + 5·x2);
```

$$f_1 := x_1 + x_2$$
$$f_2 := 12\,x_1 + 7\,x_2$$
$$f_3 := -7\,x_1 - 5\,x_2$$

```
> obmez := {12·x1 + 7·x2 ≤ 127, 0.5·x1 + 1.4·x2 ≤ 16, x1 ≥ 2, x2 ≥ 4};
```

$$obmez := \{2 \le x_1,\ 4 \le x_2,\ 12\,x_1 + 7\,x_2 \le 127,\ 0.5\,x_1 + 1.4\,x_2 \le 16\}$$

```
> w1 := 0.5; w3 := 0.2; w2 := 1 - w1 - w3; # вагові коефіцієнти
```

$$w_1 := 0.5$$
$$w_3 := 0.2$$
$$w_2 := 0.3$$

```
> with(Optimization) :
> F := w1·f1 + w2·f2 + w3·f3; # узагальнена функція мети.
                        F := 2.7 x1 + 1.6 x2
> Fmax := Maximize(F, obmez); # максимізуємо узагальнену функцію мети
   Fmax := [28.8165413533835, [x1 = 4.94736842105263186, x2 = 9.66165413533834538]]
> assign(Fmax[2]); # надамо змінним x1 та x2  отримані значення
```

```
> В результаті оптимізації методом вагових коефіцієнтів отримано;
   Максимальний обсяг перевезення = evalf4(f1); Максимальний дохід = evalf4(f2);
   Мінімальна собівартість = evalf4(-1·f3); План перевезень 'x1'= evalf4(x1); 'x2'
   = evalf4(x2);
        В результаті оптимізації методом вагових коефіцієнтів отримано
                Максимальний обсяг перевезення = 14.61
                Максимальний дохід = 127.0
                Мінімальна собівартість = 82.94
                План перевезень x1 = 4.947, x2 = 9.662
```

Рис. 12.2. Метод вагових коефіцієнтів в пакеті ***Maple***

МЕТОД ПОСЛІДОВНОЇ ПОСТУПКИ

```
> restart :
```

Задаємо функції мети, обмеження

```
> f1 := x1 + x2; f2 := 12·x1 + 7·x2; f3 := -1·(7·x1 + 5·x2);
                        f1 := x1 + x2
                        f2 := 12 x1 + 7 x2
                        f3 := -7 x1 - 5 x2
> obmez := {12·x1 + 7·x2 ≤ 127, 0.5·x1 + 1.4 ·x2 ≤ 16, x1 ≥ 2, x2 ≥ 4};
        obmez := {2 ≤ x1, 4 ≤ x2, 12 x1 + 7 x2 ≤ 127, 0.5 x1 + 1.4 x2 ≤ 16}
```

Оптимізацію починаємо з функції мети з найвищим пріоритетом (з найбільшим ваговим коефіцієнтом). Це функція *f1*

```
> with(Optimization):
> f1max := Maximize(f1, obmez);
    f1max := [14.6090225563910, [x1 = 4.94736842105263098, x2 = 9.66165413533834716]]
```

Задаємо значення поступки для функції мети *f1*

```
> δ1 := 0.1·f1max[1];
                        δ1 := 1.460902256
```

Оптимізуємо наступну за пріоритетом функцію мети *f2*, використовуючи додаткове обмеження на величину максимальної поступки для функції *f1*

```
> obmez := obmez union {f1 ≥ f1max[1] − δ1};
  obmez := {2 ≤ x1, 4 ≤ x2, 13.14812030 ≤ x1 + x2, 12 x1 + 7 x2 ≤ 127, 0.5 x1 + 1.4 x2 ≤ 16}
> f2max := Maximize(f2, obmez);
    f2max := [127.000000000000, [x1 = 6.99263158000000118, x2 = 6.15548872000000014]]
```

Задаємо значення поступки для функції мети *f2*

```
> δ2 := 0.15·f2max[1];
                        δ2 := 19.05000000
```

Оптимізуємо наступну за пріоритетом функцію мети *f3*, використовуючи додаткове обмеження на величину максимальної поступки для функцій *f1* та *f2*

```
> obmez := obmez union {f2 ≥ f2max[1] − δ2};
obmez := {2 ≤ x1, 4 ≤ x2, 13.14812030 ≤ x1 + x2, 107.9500000 ≤ 12 x1 + 7 x2, 12 x1 + 7 x2
   ≤ 127, 0.5 x1 + 1.4 x2 ≤ 16}

> f3min := Maximize(f3, obmez);
    f3min := [−72.1058646600000, [x1 = 3.18263157999999756, x2 = 9.96548872000000330]]
> assign(f3min[2]); # надамо змінним x1 та x2 отримані значення
```

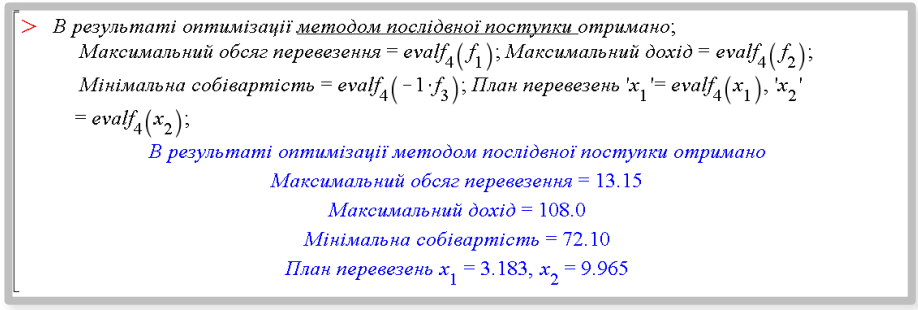


Рис. 12.3. Метод послідовної поступки в пакеті ***Maple***

Рис. 12.4. Метод мінімізації загальної поступки в пакеті ***Maple***

Позначимо величину загальної поступки z – значення (десятковий процент), на яке ми можемо "погіршити" максимальні значення кожної функції мети, знайдені незалежно. Доповнимо початкові обмеження обмеженнями, які задають величину максимальної поступки для кожної функції (рис. 12.5).

```
> obmez1 := obmez union {f1_1 - fl1_1 + z·fl1_1 ≥ 0, f2_1 - f21_1 + z·f21_1 ≥ 0, z ≥ 0, f3_1 - f31_1 + z·f31_1·(
      -1) ≥ 0};
obmez1 := {0 ≤ z, 0 ≤ -7 x1 - 5 x2 + 34. + 34. z, 0 ≤ x1 + x2 - 14.60902256
   + 14.6090225563910 z, 0 ≤ 12 x1 + 7 x2 - 127. + 127. z, 2 ≤ x1, 4 ≤ x2, 12 x1 + 7 x2 ≤ 127,
   0.5 x1 + 1.4 x2 ≤ 16}
```

Знайдемо мінімальне значення загальної поступки

```
> zmin := Minimize(z, obmez1);
zmin := [0.418504468659165, [z = 0.418504468659165130, x1 = 2.87687262368625074, x2
   = 5.61820871372157349]]
```

```
> assign(zmin[2]); # надамо змінним x1 та x2 отримані значення
> В результаті оптимізації методом мінімізації загальної поступки отримано;
    Максимальний обсяг перевезення = evalf4(f1); Максимальний дохід = evalf4(f2);
    Мінімальна собівартість = evalf4(-1·f3); План перевезень 'x1'= evalf4(x1), 'x2'
    = evalf4(x2);
      В результаті оптимізації методом мінімізації загальної поступки отримано
                Максимальний обсяг перевезення = 8.495
                Максимальний дохід = 73.85
                Мінімальна собівартість = 48.23
                План перевезень x1 = 2.877, x2 = 5.618
```

Рис. 12.5. Додаткові обмеження

Для аналізу отриманих результатів знайдемо при заданих обмеженнях максимум кожної функції мети, незалежно від значень інших функцій мети (незалежна оптимізація (рис. 12.6)).

НЕЗАЛЕЖНА ОПТИМІЗАЦІЯ

```
> restart :
```

Задаємо функції мети, обмеження

```
> f1 := x1 + x2;f2 := 12·x1 + 7·x2;f3 := -1·(7·x1 + 5·x2);
```

$$f_1 := x_1 + x_2$$

$$f_2 := 12\,x_1 + 7\,x_2$$

$$f_3 := -7\,x_1 - 5\,x_2$$

```
> obmez := {12·x1 + 7·x2 ≤ 127, 0.5·x1 + 1.4 ·x2 ≤ 16, x1 ≥ 2, x2 ≥ 4};
```

$$obmez := \{2 \le x_1,\ 4 \le x_2,\ 12\,x_1 + 7\,x_2 \le 127,\ 0.5\,x_1 + 1.4\,x_2 \le 16\}$$

```
> with(Optimization) :
> f1 := Maximize(f1, obmez);f2 := Maximize(f2, obmez);f3 := Maximize(f3, obmez);
```

$$f1 := [14.6090225563910,\ [x_1 = 4.94736842105263098,\ x_2 = 9.66165413533834716]]$$

$$f2 := [127.,\ [x_1 = 8.25000000000000000,\ x_2 = 4.]]$$

$$f3 := [-34.,\ [x_1 = 2.,\ x_2 = 4.]]$$

```
> В результаті незалежної оптимізації отримано; Максимальний обсяг перевезення
  = evalf4(f11); Максимальний дохід = evalf4(f21); Мінімальна собівартість = evalf4(-1
  · f31);
```

результаті незалежної оптимізації отримано

Максимальний обсяг перевезення = 14.61

Максимальний дохід = 127.

Мінімальна собівартість = 34.

Рис. 12.6. Незалежна оптимізація

Отже, отримані такі допустимі, оптимальні в певному сенсі, рішення (таблиця 12.1).

Таблиця 12.1

	Кількість вугілля, $x_1, 10^2$ т	Кількість руди, x_2, 10^2 т	Обсяг перевезення, f_1, 10^2 т	Дохід, f_2, тис. у.о.	Собівартість, f_3, тис. у.о.
Незалежна оптимізація функції f_1	4,95	9,66	**14,61**	127	82,94
Незалежна оптимізація функції f_2	8,25	4	12,25	**127**	77,75
Незалежна оптимізація функції f_3	2	4	6	52	**34**
Метод вагових коефіцієнтів	4,95	9,67	14,61	127	82,94
Метод послідовної поступки	3,18	9,97	13,15	108	72,1
Метод мінімізації загальної поступки	2,88	5,62	8,49	73,85	48,23

Оскільки ідеального рішення задача не має, то остаточний вибір серед запропонованих варіантів покладається на відповідальну особу.

Контрольні запитання

1. Сформулюйте в загальному виді задачу багатокритеріальної оптимізації.
2. Яке рішення задачі багатокритеріальної оптимізації називають ідеальним?
3. Дайте визначення точки, оптимальної за Парето.
4. Які методи багатокритеріальної оптимізації ви знаєте?
5. Назвіть недоліки методу вагових коефіцієнтів.

ТЕМА 13. СИСТЕМИ МАСОВОГО ОБСЛУГОВУВАННЯ

13.1. Основні поняття та функціонування систем масового обслуговування

Теорія масового обслуговування або **теорія черг** – це один з розділів теорії ймовірностей. Основи теорії системи масового обслуговування закладені у працях датського інженера і математика А. К. Ерланга та становлять важливу частину теорії керування та планування виробництва. Черги – це явище, яке часто виникає у повсякденному житті. Вони можуть набувати різних форм. Наведемо у таблиці 13.1 деякі приклади виникнення черг у системах масового обслуговування.

Таблиця 13.1

Ситуація	**Хто/що чекає**	**Процес обслуговування (на що чекає)**
Супермаркет	Покупець	Оплата покупок
Приймальня лікаря	Пацієнт	Консультація
Комп'ютер	Програма	Виконання процесором
Телефонний номер	Абонент	З'єднання з процесором

Предметом дослідження теорій масового обслуговування стали такі галузі як транспорт, торгівля, медицина, комп'ютерна техніка та інші, де маємо справу з обслуговуванням. Зрозуміло, що діяльність оперативних служб, таких як, пожежна охорона, служба цивільного захисту, поліція, швидка допомога, теж може розглядатись як система масового обслуговування.

Вивчення черг в системах масового обслуговування дає змогу визначити критерії функціонування систем масового обслуговування, серед яких найбільш важливими є середній час очікування в черзі, довжина черги та інші. Ця інформація

використовується потім для розрахунку витрат на обслуговування та витрат системи в результаті очікування клієнта. Звичайно, феномен очікування не можна виключити без додаткових видатків, але в деяких системах масового обслуговування, наприклад у швидкій допомозі, ціна втрат, що пов'язані з довгим очікуванням, може бути дуже високою.

Основними елементами систем масового обслуговування є **клієнт** (вимога, заявка на обслуговування, або просто об'єкт обслуговування) і **сервіс** (канал, засіб обслуговування).

Вимоги поступають в системи масового обслуговування з джерела. Потрапивши до системи, вони можуть потрапити одразу на обслуговування або очікувати у черзі, якщо канал обслуговування зайнятий.

Основними етапами, які проходить вимога у системі масового обслуговування є:

- поява у системі з джерела (вхід);
- проходження черги (очікування);
- процес обслуговування, після якого вимога залишає систему.

Кожен етап має свої характеристики, які потрібно врахувати при побудові математичної моделі.

Основними ***характеристиками входу (джерела)*** є:

- число вимог на вході;
- режим надходження вимог в систему масового обслуговування;
- поведінка вимог (клієнтів).

Розглянемо детальніше кожну характеристику.

Число вимог на вході. Число потенційно можливих вимог може вважатися скінченним або нескінченним. Якщо число вимог, що надійшли на вхід системи з моменту початку процесу обслуговування до будь-якого заданого моменту часу, є лише малою частиною потенційно можливої кількості вимог, то число на вході вважається нескінченним. Прикладом можуть бути автомобілі, що проїжджають через пропускні пункти на

швидкісних дорогах, покупці в супермаркеті. В більшості моделей черг на вході розглядають саме необмежений випадок.

Прикладом системи масового обслуговування зі скінченним числом можуть бути комп'ютери конкретних організацій, що надходять на обслуговування в ремонтну майстерню.

Режим надходження вимог в систему масового обслуговування. Вимоги можуть надходити в систему обслуговування згідно з певним графіком (один пацієнт на прийом до стоматолога кожні півгодини, одна деталь на конвеєрі кожні 20 хв) або випадковим чином.

Появи вимог вважаються випадковими, якщо вони не залежні одна від одної й точно передбачити їх неможливо. Часто в задачах масового обслуговування число появ вимог за одиницю часу можна оцінити за допомогою Пуассонівського закону розподілу, при заданій інтенсивності вимог (наприклад, дві вимоги за годину, десять викликів за добу). Дискретний розподіл Пуассона описується формулою:

$$P_n = \frac{e^{-a} \cdot a^n}{n!},\ n = 0, 1, \ldots, \qquad (13.1)$$

де λ – інтенсивність потоку за одиницю часу, $a = \lambda\tau$ – середнє число подій, що відбуваються протягом часу τ, P_n – ймовірність надходження n вимог за час τ.

Приклад 13.1. Нехай інтенсивність потоку вимог, що описується законом Пуассона, – дві вимоги за годину. Знайти ймовірність того, що протягом години до системи не надійде жодної вимоги, дві вимоги, дев'ять?

Розв'язання.

Інтенсивність вимог $\lambda = 2$ (год$^{-1}$), $a = \lambda\tau = \lambda$. Ймовірність того, що протягом години до системи не надійде жодної вимоги за формулою (13.1)

$$P_0 = \frac{e^{-\lambda} \cdot \lambda^0}{0!} = e^{-2} \approx 0{,}135 .$$

Ймовірність того, що протягом години до системи надійдуть дві вимоги

$$P_2 = \frac{e^{-\lambda} \cdot \lambda^2}{2!} = \frac{e^{-2} \cdot 2^2}{2} \approx 0{,}135 \cdot 2 = 0{,}27 .$$

Ймовірність того, що протягом години до системи надійде дев'ять вимог

$$P_9 = \frac{e^{-\lambda} \cdot \lambda^9}{9!} = \frac{e^{-2} \cdot 2^9}{9!} \approx 0{,}00019 .$$

На практиці ймовірність появи вимог не завжди підлягає закону Пуассона, вони можуть мати якийсь інший розподіл, тому потрібно проводити додаткові дослідження для того, щоб переконатись, що Пуассонівський розподіл може бути хорошою апроксимацією.

Поведінка вимог (клієнтів). Більшість моделей черг базується на припущенні, що кожна вимога, що надходить до системи, стає в чергу, очікує обслуговування й не залишає систему доти, поки не буде обслужена. Але на практиці може виникнути ситуація, коли вимоги (клієнти) йдуть з системи, не дочекавшись обслуговування.

Основними ***характеристиками черги*** є:

- довжина черги;
- правила обслуговування.

Довжина черги може бути обмежена або необмежена. Довжина черги *обмежена*, якщо у зв'язку з деякими причинами вона не може збільшуватися до безмежності. Якщо черга досягає свого максимального значення, то наступна вимога в систему не допускається. Довжина черги *необмежена*, якщо в черзі може перебувати значна кількість вимог. Наприклад, черга авто на митниці.

Правила обслуговування. Більшість реальних систем використовує правило «перший прийшов – перший пішов», тобто обслуговування у порядку надходжень вимог у систему. В деяких випадках до цього правила можуть встановлюватися різні пріоритети. Наприклад, пацієнт з інфарктом у критичному стані буде мати пріоритет у обслуговуванні порівняно з пацієнтом, який зламав собі палець; спец. автомобілі (пожежні, медичні, автомобілі поліції) мають пріоритет при регулюванні дорожнього руху.

У деяких системах діють інші правила обслуговування, наприклад, «останній обслуговується першим».

Основними ***характеристиками процесу обслуговування*** є:

- конфігурація систем масового обслуговування;
- режим обслуговування.

Конфігурація систем масового обслуговування – це кількість каналів обслуговування. Зазвичай кількість каналів обслуговування можна визначити як кількість вимог (клієнтів), обслуговування яких може проводитись одночасно. Наприклад, кількість касирів, кількість колонок на заправці, кількість процесорів. Інша характеристика конфігурації – кількість фаз обслуговування та послідовність етапів обслуговування однієї вимоги.

Прикладом *одноканальної* системи масового обслуговування може бути банк, де відкрите тільки одне віконце для обслуговування клієнтів. *Однофазовими* є системи, в яких клієнт обслуговується в одному пункті, а потім залишає систему. Ресторан, в якому офіціант отримує замовлення, приносить його та отримує гроші – це приклад однофазової системи. Якщо замовлення потрібно зробити в одному місці, оплатити його в іншому, а отримати його в третьому, то ми маємо справу з *багатофазовою* системою масового обслуговування.

Режим обслуговування, так само як режим надходження вимог, може характеризуватися сталим або випадковим часом

обслуговування. При сталому часі обслуговування на одного клієнта затрачається один і той самий час. Така ситуація може спостерігатися на автоматичній мийці авто. Але частіше трапляються ситуації, коли час обслуговування – це випадкова величина. Причому часто можна припустити, що час обслуговування має експоненціальний розподіл з функцією розподілу.

$$F(\tau) = P(t < \tau) = 1 - e^{-\tau\mu}, \qquad (13.2)$$

де $P(t < \tau)$ – ймовірність того, що фактичний час t обслуговування вимоги не буде більшим за τ; μ – середня кількість вимог, що обслуговуються за одиницю часу.

13.2. Випадковий характер надходження вимог і обслуговування

Потік вимог у процесах масового обслуговування найчастіше характеризується умовами, притаманними потокам, що описуються законом Пуассона. Вони мають особливо прості властивості. Розглянемо деякі поняття, які описують потоки подій.

Потік подій називають **стаціонарним**, якщо ймовірність попадання певної кількості подій у проміжок часу довжиною τ залежить тільки від довжини проміжку і не залежить від того, де саме на осі часу розташовано цей проміжок.

Для стаціонарного потоку характерна стала інтенсивність (щільність) вимог. Потік може мати місцеві згущення та розрідження, але вони не мають закономірного характеру.

На практиці часто трапляються потоки вимог, які можна вважати стаціонарними. Наприклад, потік автомобілів на автомагістралі в інтервалі часу з 10-ої до 12-ої години ранку можна розглядати як стаціонарний, хоча цей же потік протягом доби вже не буде стаціонарним, основна інтенсивність руху вдень значно більша, ніж вночі.

Така неоднозначність щодо стаціонарності властива більшості фізичних процесів, які називають стаціонарними, а в дійсності вони стаціонарні лише на обмеженому проміжку часу.

Потік подій називають **потоком без післядії**, якщо для будь-яких проміжків часу, що не перекриваються, кількість подій, які потрапили в один з них, не залежить від кількості подій, які потрапили в інші.

Відсутність післядії в потоці означає, що події, які утворюють потік, з'являються в послідовні моменти часу не залежно одна від одної. Наприклад, транспортний потік, що рухається автомобільною дорогою без світлофорного регулювання, можна вважати потоком без післядії, тому що він утворений з авто, що потрапили на цю дорогу з причин, не пов'язаних між собою. Проте умову відсутності післядії буде порушено при світлофорному регулюванні, основні моменти звільнення перехрестя від авто залежні між собою (аналогічно потік пасажирів у метро).

Доцільно відзначити, що вихідний потік, тобто потік вимог, які після обслуговування залишають систему, зазвичай мають післядію, навіть якщо вхідний потік її не має. Післядію вихідного потоку потрібно враховувати, якщо він є вхідним для іншої системи масового обслуговування або при багаторазовому обслуговуванні.

Потік подій називають **ординарним**, якщо ймовірністю потрапляння на елементарну ділянку Δt двох або більше подій можна знехтувати порівняно з імовірністю потрапляння однієї події.

Якщо потік має всі три властивості, тобто він стаціонарний, без післядії і ординарний, то його називають **найпростішим** або **стаціонарним Пуассонівським потоком.** Далі будемо розглядати стаціонарні Пуассонівські потоки.

Для стаціонарного Пуассонівського потоку ймовірність появи на фіксованому проміжку часу τ точно n подій виражається формулою (13.1).

Важлива характеристика потоку – розподіл довжини проміжку між сусідніми подіями. Якщо проміжок часу між двома довільними подіями в стаціонарному Пуассонівському потоці випадкова величина t, то щільність розподілу випадкової величини t визначається формулою

$$f(\tau) = F'(\tau) = \lambda e^{-\lambda\tau},$$

тобто випадкова величина t має експоненціальний розподіл.

Таким чином, якщо випадковий процес описується законом Пуассона з параметром λ, то інтервали між подіями розподілені за експоненціальним законом з тими же параметром λ. Відповідності між експоненціальним розподілом з інтенсивністю λ і розподілом Пуассона наведено у таблиці 13.2.

Таблиця 13.2

	Експоненціальний розподіл	**Розподіл Пуассона**
Випадкова змінна	Час t між надходженням вимог	Кількість вимог, що надходять до системи протягом часу t
Значення випадкової величини	$t \geq 0$	$n = 0, 1, 2, \ldots$
Щільність розподілу ймовірностей (ймовірність)	$f(t) = \lambda e^{-\lambda t}$	$P_n = \dfrac{e^{-a} \cdot a^n}{n!}$
Середнє значення	$\dfrac{1}{\lambda}$ часових одиниць	$\lambda \cdot t$ протягом часу t

Функція розподілу ймовірностей	$P(t \le A) = 1 - e^{-\lambda A}$	$P_{n \le N}(t) = \sum_{i=0}^{N} P_i(t)$
Ймовірність того, що не надійде жодної вимоги протягом часу A	$P(t > A) = e^{-\lambda A}$	$P_0(A) = e^{-\lambda A}$

13.3. Загальна модель системи масового обслуговування

Нехай маємо систему масового обслуговування з стаціонарним Пуасонівським потоком вимог. Позначимо:

n – кількість вимог в системі масового обслуговування (в черзі та на обслуговуванні);

λ_n – інтенсивність надходження вимог, за умови, що в системі вже є n вимог;

μ_n – інтенсивність вихідного потоку за умови, що в системі є n вимог;

P_n – ймовірність того, що в системі є n вимог.

В загальній моделі системи масового обслуговування встановлюється залежність імовірностей P_n від параметрів λ_n та μ_n. Ці ймовірності використовуються для визначення функціональних характеристик таких систем масового обслуговування, як довжина черги, середній час очікування та середній коефіцієнт використання сервісів (каналів).

Стани системи масового обслуговування зручно зображати у вигляді графа, вершини якого – стани, а дуги – переходи від одного стану до іншого. Дугам ставиться у відповідність інтенсивність переходу від одного стану до другого.

Оскільки ймовірність появи в системі більше однієї вимоги протягом малого проміжку часу прямує до нуля, то граф станів

системи масового обслуговування для загальної моделі можна зобразити рисунком (рис. 13.1), де S_n – стан, при якому в системі знаходиться n вимог.

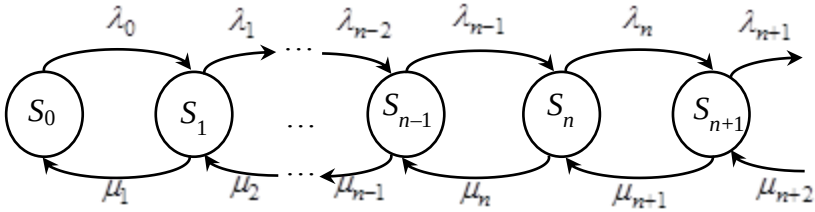


Рис. 13.1. Граф станів системи масового обслуговування

З рисунка видно, що при $n > 0$ стан S_n може змінитися у двох можливих напрямках: до S_{n-1} з інтенсивністю μ_n, коли вимога, яку обслужили, залишає систему, або до стану S_{n+1}, з інтенсивністю λ_n, коли вимога надійшла до системи. Стан S_0 може перейти лише до стану S_1, оскільки залишити вже порожню систему вимоги не можуть.

При вивчені загальних моделей системи припускається, що інтенсивності вихідного потоку та вхідного у стані S_n однакові, тобто

$$\lambda_{n-1} \cdot P_{n-1} + \mu_{n+1} \cdot P_{n+1} = \lambda_n \cdot P_n + \mu_n \cdot P_n, \; n = 1, 2, \ldots . \quad (13.3)$$

Рівняння (13.3) називають **рівняннями балансу**. Рівняння балансу розв'язується рекурентно.

При $n = 0$ з рис. 13.1 рівняння балансу матиме вигляд $\lambda_0 P_0 = \mu_1 P_1$, звідки $P_1 = \left(\frac{\lambda_0}{\mu_1} \right) P_0$.

При $n = 1$ маємо $\lambda_0 P_0 + \mu_2 P_2 = \lambda_1 P_1 + \mu_1 P_1$, звідки $P_2 = \left(\frac{\lambda_1 \lambda_0}{\mu_2 \mu_1} \right) P_0$ і т.д.

В результаті отримаємо

$$P_n = \left(\frac{\lambda_{n-1} \lambda_{n-2} \dots \lambda_0}{\mu_n \mu_{n-1} \dots \mu_1} \right) P_0, \; n = 1, 2, \dots . \qquad (13.4)$$

Приклад 13.2. Система масового обслуговування задана графом станів. Знайти ймовірності p_n.

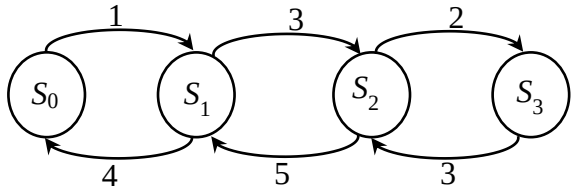


Рис. 13.2. Граф станів системи масового обслуговування

Розв'язання.

За формулою (13.4) та з рис. 13.2 маємо:

$$p_1 = \left(\frac{\lambda_0}{\mu_1} \right) p_0 = \frac{1}{4} p_0;$$

$$p_2 = \left(\frac{\lambda_0 \lambda_1}{\mu_1 \mu_2} \right) p_0 = \frac{1 \cdot 3}{4 \cdot 5} p_0 = \frac{3}{20} p_0;$$

$$p_3 = \left(\frac{\lambda_0 \lambda_1 \lambda_2}{\mu_1 \mu_2 \mu_3} \right) p_0 = \frac{1 \cdot 3 \cdot 2}{4 \cdot 5 \cdot 3} p_0 = \frac{1}{10} p_0.$$

Оскільки $p_0 + p_1 + p_2 + p_3 = 1$, то

$$p_0 + \frac{1}{4} p_0 + \frac{3}{20} p_0 + \frac{1}{10} p_0 = 1 \Rightarrow \frac{30}{20} p_0 = 1 \Rightarrow p_0 = \frac{2}{3} \approx 0,667.$$

Тоді $p_1 = \frac{1}{4} p_0 \approx 0{,}167$; $p_2 = \frac{3}{20} p_0 = 0{,}1$; $p_3 = \frac{1}{10} p_0 \approx 0{,}067$.

Зауважимо, що більшу частину часу (66,7%) система перебуває в стані S_0.

Системи масового обслуговування класифікують за різними ознаками: за кількістю каналів обслуговування (одноканальні та багатоканальні), за довжиною черги (з обмеженням за довжиною черги та без обмеження за довжиною черги), за наявністю черги (з відмовами та з чергою) тощо.

Залежно від типу системи масового обслуговування виведені формули для визначення функціональних характеристик систем масового обслуговування, основними з яких є:

- ймовірність відмови у обслуговуванні, $P_{відм}$;
- пропускна здатність системи;
- середня кількість вимог у черзі ($\overline{n}_{черг}$);
- середня кількість вимог у системі ($\overline{n}_c$);
- середній час очікування в черзі ($\overline{t}_{черг}$);
- середній час перебування у системі($\overline{t}_c$);
- середня кількість зайнятих каналів обслуговування $\overline{c}$.

Розглянемо деякі типи систем масового обслуговування. Будемо вважати, що всі потоки є стаціонарними Пуассонівськими потоками.

13.4. Одноканальна система масового обслуговування з обмеженням за довжиною черги

Розглянемо роботу одноканальної СМО з обмеженням за довжиною черги на прикладі.

Приклад 13.3. АЗС має одну колонку зі стоянкою на 3 автомобілі. Якщо стоянка зайнята, то черговий автомобіль, що надходить до станції, проїжджає повз. Потік автомобілів, що під'їжджають до АЗС, має інтенсивність 4 автомобілі за годину, а процес заправки триває в середньому 12 хв.

Потрібно визначити

- ймовірність відмови;
- середнє число автомобілів, що очікують заправки;
- середнє число автомобілів, що перебувають на АЗС;
- середню тривалість очікування автомобілів в черзі;
- середню тривалість перебування автомобіля на АЗС.

Розв'язання.

Інтенсивність вхідного потоку не залежить від кількості машин, що перебувають на заправці, й дорівнює $\lambda = 4$ год$^{-1}$. Інтенсивність вихідного потоку $\mu = \frac{1}{12}$ хв$^{-1}$= 5 год$^{-1}$.

Побудуємо граф станів (рис. 13.3).

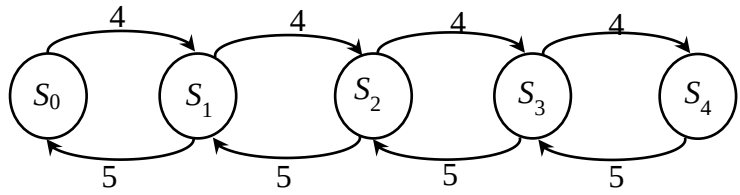


Рис. 13.3. Граф станів

Обчислимо ймовірності P_i , $i = \overline{0,4}$:

$$P_1 = \frac{\lambda}{\mu} P_0,\ P_2 = \left(\frac{\lambda}{\mu}\right)^2 P_0,\ P_3 = \left(\frac{\lambda}{\mu}\right)^3 P_0,\ P_4 = \left(\frac{\lambda}{\mu}\right)^4 P_0.$$

Позначимо $z = \frac{\lambda}{\mu} = 0,8$. Оскільки $P_0 + P_1 + P_2 + P_3 + P_4 = 1$, то

$P_0 + z \cdot P_0 + z^2 \cdot P_0 + z^3 \cdot P_0 + z^2 \cdot P_0 = 1$, тобто

$P_0(1 + 0,8 + 0,8^2 + 0,8^3 + 0,8^4) = 1.$

Для обчислення суми скористаємось формулою суми геометричної прогресії $S_n = \frac{b_1(q^n - 1)}{q - 1}$, де q – знаменник геометричної прогресії, b_1 – перший член геометричної прогресії. Отримаємо

$1 + 0,8 + 0,8^2 + 0,8^3 + 0,8^4 = \frac{1 \cdot (0,8^5 - 1)}{0,8 - 1} \approx 3,36$. Звідки $P_0 \cdot 3,36 = 1$,

$P_0 \approx 0,3$.

Отже,

$P_1 = z \cdot P_0 = 0,8 \cdot 0,3 = 0,24$;

$P_2 = z^2 \cdot P_0 = 0,8^2 \cdot 0,3 = 0,19$;

$P_3 = z^3 \cdot P_0 = 0,8^3 \cdot 0,3 = 0,15$;

$P_4 = z^4 \cdot P_0 = 0,8^4 \cdot 0,3 = 0,12$.

З ймовірністю P_4 на АЗС перебуває 4 автомобілі й вимога (автомобіль) залишає АЗС без заправки, тобто $P_4 = 0,12$ – це ймовірність відмови.

Середня довжина черги (середня кількість вимог в черзі $\overline{n}_{черг}$) визначається як математичне сподівання кількості вимог, що перебуває в черзі

Кількість вимог у черзі	0	1	2	3
Ймовірність	$P_0 + P_1$	P_2	P_3	P_4

$\overline{n}_{черг} = 0 \cdot (P_0 + P_1) + 1 \cdot P_2 + 2 \cdot P_3 + 3 \cdot P_4 = 0{,}19 + 0{,}3 + 0{,}36 = 0{,}85$,
тобто в черзі перебуває в середньому 1 автомобіль.

Середнє число автомобілів, що перебувають на АЗС (середня кількість вимог у системі $\overline{n}_c$) дорівнює сумі кількості вимог у черзі ($\overline{n}_{черг}$) та кількості вимог, що обслуговуються ($\overline{n}_{обс}$), тобто

$$\overline{n}_c = \overline{n}_{черг} + \overline{n}_{обс}.$$

Знайдемо $\overline{n}_{обс}$ як математичне сподівання випадкової величини "кількість вимог, що обслуговуються"

Кількість вимог, що обслуговується	0	1
Ймовірность	P_0	$1 - P_0$

$\overline{n}_{обс} = 0 \cdot P_0 + 1 \cdot (1 - P_1) = 0{,}7$. Тоді $\overline{n}_c = 0{,}85 + 0{,}7 = 1{,}55$.

Отже, на АЗС перебуває в середньому 1,55 авто.

Середню тривалість очікування автомобілів в черзі ($\overline{t}_{черг}$) визначимо, аналізуючи можливі стани системи з таких міркувань.

З імовірністю P_0 канал (АЗС) вільний, і автомобіль не очікуватиме, тобто $t_{черг} = 0$; з імовірністю P_1 авто потрапить на АЗС під час обслуговування іншого авто, але перед іншим не буде черги і він чекатиме протягом часу $\frac{1}{\mu}$ – середня тривалість обслуговування; з ймовірністю P_2 в черзі перед авто стоятиме ще один автомобіль, тому він буде чекати протягом часу $\frac{2}{\mu}$ і, нарешті з ймовірністю P_3 в черзі буде перед авто два інших авто

плюс машина на обслуговувані, тому авто буде чекати $\frac{3}{\mu}$.

Обчислимо

$$\overline{t}_{черг} = P_1 \cdot \frac{1}{\mu} + P_2 \cdot \frac{2}{\mu} + P_3 \cdot \frac{3}{\mu} = 0,24 \cdot \frac{1}{5} + 0,19 \cdot \frac{2}{5} + 0,15 \cdot \frac{3}{5} =$$
$$= 0,214\,(год) = 13\,(хв).$$

Зауваження. Середня тривалість очікування дорівнює середньому числу вимог у черзі, поділеному на інтенсивність потоку вимог $\overline{t}_{черг} = \frac{\overline{n}_{черг}}{\lambda}$.

Середня тривалість перебування вимоги в системі $\overline{t}_c$ дорівнює сумі тривалості очікування в черзі й тривалості обслуговування, тобто

$$\overline{t}_c = \overline{t}_{черг} + \overline{t}_{обс}.$$

Ймовірність того, що вимогу буде прийнято на обслуговування дорівнює $1 - P_{відм}$, час обслуговування дорівнює $\frac{1}{\mu}$, тому

$$\overline{t}_c = \overline{t}_{черг} + (1 - P_{відм}) \cdot \frac{1}{\mu} = \overline{t}_{черг} + (1 - P_4) \cdot \frac{1}{\mu} = 0,21 + 0,88 \cdot \frac{1}{5} =$$
$$= 0,386\,(год) = 23,16\,(хв).$$

Контрольні запитання

1. Назвіть основні елементи системи масового обслуговування.
2. Сформулюйте основні етапи, які проходить вимога у системі масового обслуговування.
3. Опишіть основні характеристики джерела у системі.
4. Опишіть основні характеристики черги у системі.

5. Опишіть основні характеристики процесу обслуговування у системі.
6. Який потік подій називають стаціонарним Пуассонівським потоком?

ДОДАТОК. Побудова математичних виразів та елементарні обчислення в пакеті Maple

Програмний пакет аналітичних обчислень ***Maple*** 13 є потужним інструментом вирішення математичних завдань. Більше двох тисяч ефективно реалізованих функцій дають змогу вирішувати задачі алгебри, математичного аналізу, диференціального й інтегрального числень, статистики, теорії графів і багато інших. Пакет включає розвинену графічну бібліотеку і мову програмування.

Maple – типове інтегроване програмне середовище, яке об'єднує в собі:

- потужну мову програмування;
- редактор для створення та редагування документів та програм;
- сучасний інтерфейс;
- потужну довідникову систему з тисячами прикладів;
- словник математичних понять та термінів;
- ядро алгоритмів та правил перетворення математичних виразів;
- чисельний та символьний процесор;
- систему діагностики;
- бібліотеки вбудованих та додаткових функцій;
- нові математичні можливості (нові функції, нові розв'язки, нові символьні перетворення тощо).

Зовнішній вигляд вікна ***Maple*** 13 представлений на рис.Д 1.

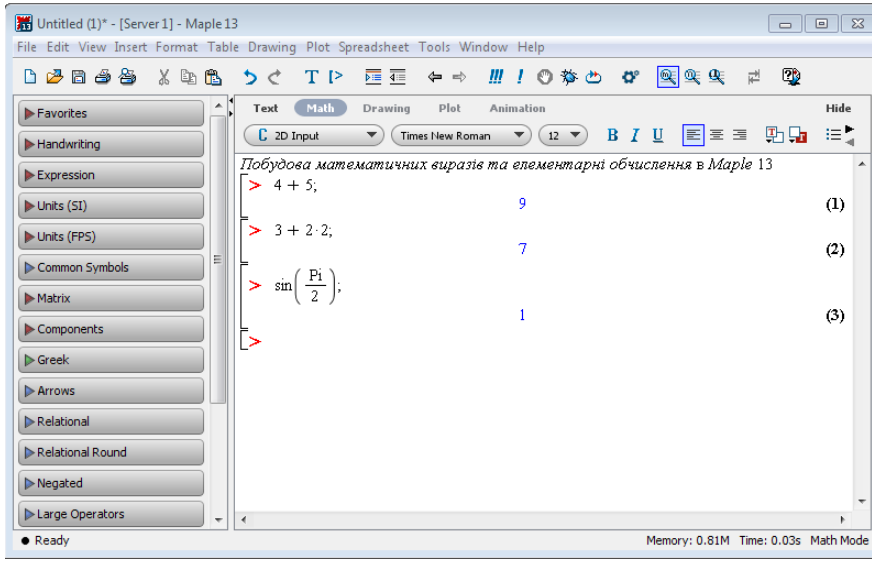


Рис.Д 1. Вікно програми Maple 13

Робота з формулами здійснюється в секціях «вхід-вихід». Кожна секція автоматично позначається лівою квадратною дужкою, яка об'єднує рядок введення (командний рядок) і отриманий результат.

Командні рядки починаються з оператора [> і мають червоний колір, а результати, автоматично вирівнюються по центру, мають синій колір. Оператор початку введення [> з'являється на робочому полі після натискання кнопки ([>) на панелі інструментів ***Maple*** 13, причому одночасно з'являється ліва квадратна дужка, яка змінює, за необхідності, свою довжину.

Для запису коментарів на панелі інструментів потрібно натиснути кнопку (T). В результаті з'явиться порожній рядок, а текст, за замовчуванням, буде відображатись курсивом. За потреби шрифт тексту можна змінити.

Для коментарів у ***Maple*** 13 передбачений символ # (решітка). Всі команди в рядку, що стоять після цього символа, не виконуються.

Оператор **";"** (крапка з комою) виводить результати обчислень на робочий лист. Якщо командний рядок закінчується цим символом, то в якому б місці рядка не був курсор, після натискання кнопки "*!!!*" з панелі інструментів або натискання на клавіатурі кнопки <Enter> виконуються обчислення, результати яких виводяться на робочий лист (рис. Д 1).

При застосуванні символа **":"** (двокрапка) команда буде виконана, але результат не буде зображений на екрані.

У ***Maple*** 13 використовують круглі, квадратні та фігурні дужки. Призначення *круглих дужок* – задавати порядок дій при побудові математичних виразів, виділяти аргументи функцій і параметри в записі команд. *Квадратні дужки* потрібні для роботи з індексними величинами. *Фігурні дужки* використовуються для формування множин.

Для очищення внутрішньої пам'яті виконуємо команду

[>*restart*;

У програмі ***Maple*** 13 багато вбудованих функцій. У таблиці Д1 наведено короткий перелік математичних функцій. З повним переліком функцій можна ознайомитися, звернувшись до довідкової системи програми.

Таблиця Д 1

Назва	**Опис**
x^y	піднесення до степеня x^y
x/y	ділення $\frac{x}{y}$
sqrt(x)	корінь квадратний $\sqrt{x}$
exp(x)	експонента e^x

Назва	Опис
ln(x)	натуральний логарифм $\ln x$
log[a](x)	логарифм $\log_a x$
n!	факторіал $n!$
abs(x)	модуль числа $\lvert x \rvert$
round(x)	заокруглення до найближчого цілого числа
trunc(x)	відкидання дробової частини числа
floor(x)	заокруглення до меншого цілого числа
ceil(x)	заокруглення до більшого цілого числа
sin(x),cos(x)	тригонометричні функції $\sin x$, $\cos x$
tan(x), cot(x)	тригонометричні функції $tg\ x$, $ctg\ x$

Останній і передостанній результати ***Maple*** 13 зберігає під іменами % та %% відповідно (рис .Д 2).

Наприклад,

Рис. Д 2. Приклад обчислення у пакеті ***Maple***

Для запису десяткових дробів використовується крапка, наприклад 2.0.

Математичні вирази записуються з використанням констант, змінних, знаків арифметичних та інших операцій.

Порядок виконання арифметичних операцій відповідає математичним правилам:

- спочатку проводиться піднесення до степеня ^,
- потім множення * і ділення /,
- потім додавання + і віднімання –.

Операції виконуються зліва направо.

Для нерівностей існують знаки >, < ,> =, < =, <>, =.

Для конструювання логічних виразів використовуються операції not (“немає”), or (“або”), and (“і”), xor (“або… або”), конструкції з яких набувають логічних значень true або false.

Зворотний слеш \ використовується для переносів.

У ***Maple*** 13 представлені всі математичні константи (таблиця Д 2). Імена констант є зарезервованими, їх значення не можуть бути перевизначені на відміну від змінних.

Таблиця Д 2

Назва	Опис
Pi	Число π
I	Уявна одиниця
Infinity	Нескінченність
true, false	Константи логіки
gamma	Константа Ейлера

Ім’я змінної в ***Maple*** 13 – це набір символів, що починаються з літери, причому великі і малі літери розрізняються. Крім букв можуть вживатися цифри і знак підкреслення (наприклад, Eq1, eq_1).

В якості назв змінних заборонено використовувати ключові слова ***Maple*** 13 і назви вбудованих функцій. Якщо ж ви випадково вибрали в якості назви змінної зарезервоване слово, система видасть повідомлення про помилку

```
Error, illegaluseofanobjectas a name
```

Для позначення службових констант використовуються імена, що починаються зі знака підкреслення. Невизначені константи, що виникають при розв'язанні диференціальних рівнянь, позначаються _C1, _C2 і т.д. Довільне ціле число позначається як _N1, _N2, і т.д., а комплексна величина відповідно як _Z1, _Z2 і т.д.

Для присвоєння значень змінної використовується **:=** (двокрапка і дорівнює).

Для перегляду значення змінної простого типу потрібно лише ввести ім'я змінної (рис. Д 3). Наприклад,

Рис. Д 3. Приклад перегляду значення змінної у пакеті ***Maple***

На рис. Д 4 проілюстровано, що потрібно зробити, для того, щоб змінна знову була невизначеною.

Рис. Д 4. Перевизначення змінної у пакеті ***Maple***

Тригонометричні функції працюють в радіанній мірі. Для перетворення градусної міри в радіани використовується функція

convert(*expr*, 'units', *form*),

де ***expr*** – вираз, який перетворюємо,

form – набуває значення ***'radians', 'degrees'*** (якщо треба вираз ***expr*** перетворити у градуси) або ***'degrees', 'radians',*** (якщо треба вираз ***expr*** перетворити у радіани).

Наприклад,

Рис. Д 5. Перетворення радіанної міри у градусну у пакеті ***Maple***

або зворотне перетворення

Рис. Д 6. Перетворення градусної міри у радіанну у пакеті ***Maple***

Для відображення результатів обчислення у вигляді десяткового дробу можна використати функцію

evalf(*f*),

де ***f*** – деякий вираз.

Наприклад,

Рис. Д 7. Виведення результатів обчислень у вигляді десяткового дробу у пакеті ***Maple***

Щоб знайти мінімум чи максимум функції використовуються відповідно команди

minimize (*expr, vars* = *ranges*)

maximize (*expr, vars* = *ranges*)

де ***expr*** – вираз, мінімум чи максимум якого потрібно знайти,

vars – змінні, за якими шукається мінімум чи максимум,
ranges – інтервал зміни змінних, якщо не вказати інтервал зміни змінних, то мінімум чи максимум буде шукатися на всій числовій осі (для попередніх версій програми ***Maple*** мала б бути присутня стрічка **'infinite'**).

Наприклад,

```
> minimize(4 + x^2)
                                  4
> maximize(4 + x^2)
                                  ∞
> maximize(4 + x^2, x = -3 ..3)
                                 13
```

Рис. Д 8. Знаходження максимуму та мінімуму функції у пакеті ***Maple***

Для дослідження на екстремум функції як однієї, так і багатьох змінних використовується команда

extrema(*expr, constr, vars, nv*),

де ***expr*** – вираз, екстремум якого потрібно знайти,
constr – обмеження,
vars – змінні, за якими шукається екстремум,
nv – ім'я змінної, якій будуть присвоєні координати точок екстремумів.

Наприклад,

```
> extrema(4 + x^2, { }, x,'s'); s;
                                {4}
                            {{x = 0}}
```

Рис. Д 9. Знаходження екстремума функції у пакеті ***Maple***

За допомогою команди **diff(*expr, var*)** можна знайти похідну виразу ***expr*** по змінній ***var***.

Для знаходження похідних вищих порядків достатньо скористатися командою **diff(*expr, var*$*n*)**, де ***n*** – число, яке визначає порядок похідної.

Наприклад,

```
> diff(x^3, x)
                                  3 x^2

> diff(x^3, x$3)
                                    6
```

Рис. Д 10. Знаходження похідної функції у пакеті ***Maple***

ЛІТЕРАТУРА

1. *Кунда Н.Т.* Дослідження операцій у транспортних системах. – К.: Видавничий Дім "Слово", 2008. – 400 с.
2. *Карагодова О.О., Кігель В.Р., Рожок В.Д.* Дослідження операцій. – К.: Центр учбової літератури, 2007. – 256 с.
3. *Вагнер Г.* Основы исследования операций: В 3 т. – М.: Мир, 1973. – 246 с.
4. *Гнеденко Б. В., Коваленко И. Н.* Введение в теорию массового обслуживания. – М.: Наука, 1966. – 524 с.
5. *Давыдов З. Г.* Исследование операций: учеб. пособ. для студ. вузов. – М., 1990. – 383 с.
6. *Домнин Л. Н.* Элементы теории графов: учеб. пособие – Пенза: Изд-во Пенз. гос. ун-та, 2007. – 144 с.
7. *Ермольев Ю. М., Ляшко И. И., Михалевич В. С, Тюптя В. И.* Математические методы исследования операций: учеб. пособие для вузов. – К, 1979. – 312 с.
8. *Зайченко С. В.* Дослідження операцій. – К.: Вища шк., 1989. – 320 с.
9. *Исследование* операций / под ред. Дж. Моудера, С. Эмалграби. – М.: Мир, 1981. – Т. 1 - 2.
10. *Кудрявцев Е. М.* Исследование операций в задачах, алгоритмах и программах. – М.: Радио и связь, 1984. – 184 с.
11. *Кулян В. Р., Юнькова Е. А., Жильцов А. Б.* Математическое программирование с злементами информационных технологий. – К: МАУП, 2000. – 124 с.
12. *Ляшенко И. Н., Карагодова Е. А., Черншова Н. В., Шор Н. З.* Линейное и нелинейное программирование / под ред. И. Н. Ляшенко. – К.: Вища шк., 1975. – 372 с.
13. *МуленЗ.* Теория игр с примерами из математической экономики: Пер. с фр. – М.: Мир, 1985. – 200 с.
14. *Попов Ю. Д.* Линейное и нелинейное программирование: учеб. пособие. – К.: Изд-во КГУ, 1988. – 180 с.
15. *Таха Х.* Введение в исследование операций. – 6-е изд.: пер. с

англ. – М.: Изд. дом "Вильямс", 2001. – 912 с.
16. *Білоус А.Б., Могила І.А.* Прикладні задачі дослідження операцій на транспорті: методичні вказівки до виконання курсової роботи з курсу «Дослідження операцій у транспортних системах». – Л.: В-во Львівської політехніки, 2012. – 36 с.