1.1. Загальна задача математичного програмування та методи її вирішення. Випадки обмежень

Математичне програмування – це область математики, яка розробляє теорію та чисельні методи розв'язку багатомірних екстремальних задач з обмеженнями, тобто задач на екстремум функції багатьох змінних з обмеженнями на область зміни цих змінних [9]. На відміну від класичної теорії екстремальних задач, основна увага у математичному програмуванні приділяється тим задачам, у яких активно беруть участь обмеження на область зміни змінних.

Функцію, екстремальне значення якої необхідно знайти, називають *цільовою, показником ефективності* чи *критерієм оптимальності.* Економічні можливості формалізуються у вигляді *системи обмежень.* Усе це складає математичну модель.

Математична модель задачі – це відображення оригінала у вигляді функцій, рівнянь, нерівностей, цифр і т.д. Модель задачі математичного програмування включає:

1) сукупність невідомих величин $\mathbf{x} = (x_1; \ldots, x_l, \ldots; x_n)$, діючі на які, систему можна удосконалювати. Їх називають планом задачі (вектором управління рішення, управлінням, стратегією, поведінкою та ін.);

2) цільову функцію (функцію цілі, показник ефективності, критерій оптимальності, функціонал задачі та ін.). Цільова функція дозволяє вибирати найкращий варіант із множини можливих. Найкращий варіант надає цільовій функції екстремальне значення. Цільову функцію позначимо буквою Z $(Z = z(\mathbf{x}))$. Цільовою функцією може бути прибуток , об'єм випуску чи реалізації, рівень обслуговування чи дефіцитності, число комплектів, відходи і т.д. ;

3) умови (чи систему умов), які накладаються на невідомі величини. Ці умови випливають із обмеженості ресурсів, з умов виробничих та технологічних процесів. Математичні обмеження виражаються у вигляді рівнянь та нерівностей. Їх сукупність утворює область допустимих рішень. Об'єднання усіх умов

(обмежень), які накладаються на невідомі (шукані) величини x_j задачі, позначимо буквою Ω ($\mathbf{x} \in \Omega$). При таких позначеннях модель математичного програмування буде виглядати $\max(\min) Z = z(\mathbf{x})$, $\mathbf{x} \in \Omega$, чи: знайти extremum $Z = z(\mathbf{x})$, $\mathbf{x} \in \Omega$.

У розгорнутому вигляді задача математичного програмування формулюється так: знайти план $\mathbf{x} = (x_1; \ldots; x_j; \ldots; x_n)$, який дає екстремальне значення цільової функції Z, тобто

$$\max(\min) Z = z(x_1, \ldots, x_j, \ldots, x_n)$$

при обмеженнях

$$\varphi_i(x_1, \ldots, x_j, \ldots, x_n) \left\{ \begin{matrix} \leq \\ = \\ \geq \end{matrix} \right\} b_i \quad (i = \overline{1, m}).$$

З економічних чи фізичних мислень на план задачі чи деякі його компоненти (координати), як правило, накладаються умови невід'ємності

$$x_j \geq 0, \quad j \in \Omega_1 \subset \Omega,$$

іноді – цілочисельності.

План $\mathbf{x}$, який задовольняє системі обмежень задачі, називається *допустимим* ($\mathbf{x} \in \Omega$). Допустимий план, який надає функції цілі екстремальне значення, називається *оптимальним.* Оптимальний план позначимо $\mathbf{x}^*$, екстремальне значення функції цілі – $\mathbf{z}(\mathbf{x}^*) = \mathbf{Z}^*$. Оптимальне рішення не обов'язково єдине, можливі випадки, коли воно не існує, є кінцева чи незчисленна множина оптимальних рішень.

Методи вирішення задачі математичного програмування. В залежності від особливостей цільової функції $z(\mathbf{x})$ та функцій, які задають обмеження $\varphi_i(\mathbf{x})$, задачі математичного програмування діляться на ряд типів.

Якщо цільова функція $Z = z(\mathbf{x})$ та функції $\varphi_i(\mathbf{x})$ $(i = \overline{1, m})$, які входять у систему обмежень, лінійні (першого степеня) відносно невідомих x_j, які входять у задачу, то такий розділ математичного програмування називається *лінійним програмуванням* (ЛП). Методи та моделі лінійного програмування широко застосовуються при оптимізації процесів у всіх галузях народного господарства.

Особливо широке застосування методи та моделі лінійного програмування отримали при рішенні задач економії ресурсів (вибір ресурсозберігаючих технологій, утворення сумішей, розкрій матеріалів), виробничо-транспортних та інших задач.

Якщо в задачі математичного програмування цільова функція $\mathbf{z(x)}$ та (чи) хоча б одна з функцій системи обмежень $\varphi_i(\mathbf{x})$ нелінійна, то такий розділ називається *нелінійним програмуванням* (НЛП). Методи та моделі нелінійного програмування можуть застосовуватися при вирішенні перерахованих вище задач, коли хоча б одна з функцій $\mathbf{z(x)}$, $\varphi_i(\mathbf{x})$ нелінійна. Крім того, методи НЛП отримали широке застосування при розрахунку економічно вигідних партій запуску деталей у виробництво, при визначенні економічно вигідної партії постачання, постачального комплекту, розмірів запасів, розподіленні обмежених ресурсів, розміщенні виробничих сил, при вирішенні багатьох виробничо-економічних задач і т.д.

Якщо на всі чи деякі змінні накладена умова дискретності, наприклад цілочисельності ($x_j = 0,1,2\ldots$), то такі задачі розглядаються у розділі математичного програмування, який називається дискретним, зокрема *цілочисельним* (ЦП), програмуванням. Методами ЦП вирішується широке коло задач оптимізації з неділимостями, комбінованого типу, з логічними умовами, з розривною цільовою функцією і т.д. До таких задач, зокрема, відносяться задачі вибору (про призначення, про комівояжера), теорії розкладів, комплектних поставок та комплектування, розміщення виробничо-складської структури і т.д.

Якщо параметри цільової функції та (чи) системи обмежень змінюються у часі чи цільова функція має адитивний вид

$$\mathbf{z(x)} = \sum_{j=1}^{\mathrm{n}} \mathbf{z}_j(\mathbf{x}_j),$$

чи мультиплікативний вид

$$\mathbf{z(x)} = \prod_{j=1}^{\mathrm{T}} \mathbf{z}_j(\mathbf{x}_j),$$

чи сам процес вироблення рішень має багатокроковий характер, то такі задачі вирішуються методами *динамічного програмування*

(ДП). Методами ДП можуть вирішуватися задачі перспективного та поточного планування, управління виробництвом, розподілу обмежених ресурсів, оновлення та відновлення складних людино-машинних організаційних систем і т.д.

У перерахованих вище розділах математичного програмування передбачається, що уся інформація про протікання процесів завчасно відома та достовірна. Такі методи оптимізації називаються *детермінованими* чи методами обґрунтування рішень в умовах визначеності.

Якщо параметри, які входять у функцію цілі, чи обмеження задачі є випадковими, недостовірними величинами чи якщо доводиться приймати рішення в умовах ризику, неповної чи недостовірної інформації, то говорять про проблему стохастичної оптимізації, а відповідний розділ називається *стохастичним програмуванням* (СП). До нього в першу чергу необхідно віднести методи та моделі вироблення рішень в умовах конфліктних ситуацій (математична теорія ігор), в умовах неповної інформації (експертні оцінки), в умовах ризику (статистичні рішення) та ін. Потім з'явилися інші типи задач, які враховують специфіку цільової функції та системи обмежень, у зв'язку з чим виникли параметричне, дробово-лінійне, блочне, сітьове (потокове), багато індексне, булеве, комбінаторне та інші типи програмування. З'явилися чисельні методи відшукування оптимальних рішень: градієнтні, штрафних та бар'єрних функцій, можливих напрямків, лінійної апроксимації, випадкового пошуку та ін.

До математичного програмування відносяться також методи рішення екстремальних задач з нескінченним числом змінних – нескінченномірне програмування.

Задачі математичного програмування з однією цільовою функцією вирішуються методами скалярної оптимізації. Однак реальні ситуації настільки складні, що нерідко доводиться одночасно враховувати декілька цільових функцій, які повинні приймати екстремальні значення. Задачі, де знаходять рішення за декількома цільовими функціями, відносяться до векторної оптимізації – це так звані *задачі багатокритеріального підходу.*

1.1.1. Основні методи лінійного програмування

Загальна задача лінійного програмування формулюється наступним чином [3]: знайти максимум (мінімум) лінійної форми

$$\mu = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n \qquad (1.1)$$

при наступних обмеженнях:

$$\begin{aligned} a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n &\le b_1, \\ a_{21}x_1 + a_{22}x_2 + \ldots + a_{2n}x_n &\le b_2, \\ \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ a_{m1}x_1 + a_{m2}x_2 + \ldots + a_{mn}x_n &\le b_m, \end{aligned} \qquad (1.2)$$

$$x_j \ge 0\,, \quad (j = 1, 2, \ldots, n)\,. \qquad (1.3)$$

Коефіцієнти c_i та a_{ij} можуть бути додатними, від'ємними чи рівними нулю.

Обмеження у вигляді нерівностей дуже просто звести до рівностей введенням додаткових змінних. У цьому випадку коефіцієнти при додаткових змінних у лінійній формі рівні нулю.

У залежності від вигляду обмежень та лінійної форми можливі три випадки:

1) умови (1.2) та (1.3) суперечливі, тобто система (1.2) не має невід'ємних розв'язків та задачу неможна вирішити.
2) умови (1.2) та (1.3) несуперечливі, але максимум (мінімум) лінійної форми дорівнює $+\infty(-\infty)$, область визначення μ не обмежена;
3) умови (1.2) та (1.3) несуперечливі, і область визначення μ обмежена.

Частіше за все зустрічається третій випадок, і саме він представляє практичний інтерес. Область, яка містить множину допустимих розв'язків задачі лінійного програмування, представляє собою опуклу багатогранну множину, а лінійна форма досягає свого максимуму (мінімуму) в одній з вершин багатогранника, причому число вершин кінцеве. Координати оптимальної вершини відповідають значенням змінних оптимального розв'язку.

У випадку, коли гіперплощина лінійної форми паралельна одній із гіперплощин обмежень чи ребру, які містять оптимальну

вершину, задача має нескінченну множину розв'язків, які належать цій гіперплощині чи ребру. До задач лінійного програмування неможна застосувати звичайні методи знаходження умовного екстремуму. Точка оптимального розв'язку завжди знаходиться на границі допустимої області, де частинні похідні взагалі не обертаються у нуль.

Навіть при порівняно невеликих значеннях m та n кількість вершин багатогранника обмежень величезна, а у більш складних задачах досягає астрономічних величин. Тому розв'язок задачі простим перебором вершин та порівнянням значень лінійної форми у них призвело б до дуже великого об'єму обчислень.

Кінцеві методи лінійного програмування дозволяють організувати той чи інший процес упорядкованого перебору вершин прямої чи двоїстої задачі, тобто звести до мінімуму кількість обчислень. Після перевірки, чи задовольняє дана вершина певному критерію оптимальності, методи лінійного програмування дозволяють визначити наступну вершину, яка лежить ближче до оптимуму, ніж попередня. Усі ці операції виконуються без обчислень лінійної форми та порівняння її значень у сусідніх вершинах. Так як кількість вершин кінцева, процес розв'язку сходиться за кінцеве число кроків.

Розглянемо коротко три основні методи лінійного програмування – симплексний, двоїстий симплексний (метод послідовного уточнення оцінок) та метод одночасного розв'язку прямої та двоїстої задачі (метод послідовного скорочення помилок). Інші кінцеві методи лінійного програмування є більш або менш вдалими модифікаціями цих трьох основних.

Найбільш широко відомий та частіше всього застосовується так званий симплексний метод, розроблений Данцигом.

Нехай є вектор $\mathbf{X} = (x_1, x_2, \ldots, x_n)$, який задовольняє обмеженням (1.2) та (1.3), він називається планом задачі. Базисом називається набір з m змінних, таких, що матриця, складена з коефіцієнтів при них, неособлива. Ці змінні називаються базисними. Інші $n - m$ змінних називаються вільними. Якщо покласти вільні змінні рівними нулю та розв'язати обмеження відносно базисних, отримаємо базисний розв'язок. Якщо цей розв'язок допустимий, то він називається опорним планом. З

визначення ясно, що опорний план представляє собою координати вершини багатогранника. Опорний план, який обертає лінійну форму μ у максимум (мінімум), називається оптимальним планом.

Припустимо, що тим або іншим способом отримано опорний план задачі. Отже, є m базисних змінних та $n-m$ вільних змінних, рівних нулю. Виразимо базисні змінні та лінійну форму через вільні змінні. Можна прослідкувати, як їх зміна впливає на лінійну форму. Збільшення вільних змінних з від'ємними коефіцієнтами призводить до зменшення лінійної форми. Залишимо їх рівними нулю і будемо збільшувати одну з вільних змінних з додатними коефіцієнтами до тих пір, поки яка-небудь базисна змінна не перетвориться у нуль. Отримаємо новий опорний план, який характеризується більшим значенням лінійної форми. Тепер необхідно виразити нові базисні змінні через вільні і т.д. до тих пір, поки у виразі лінійної форми коефіцієнти при усіх вільних змінних не будуть від'ємними. Подальше збільшення лінійної форми неможливе, даний опорний план є оптимальним. Оскільки число вершин кінцеве, а значення лінійної форми від вершини до вершини збільшується, оптимальний розв'язок буде досягнутий за кінцеве число ітерацій. За кінцеве число кроків визначається також несумісність системи обмежень або необмеженість лінійної форми. У цьому полягає ідея симплексного методу.

Два інші метода основані на теорії двоїстості, сутність якої полягає у наступному. Кожній задачі лінійного програмування

$$\mathbf{AX} \le \mathbf{B},$$
$$\mathbf{X} \ge 0,$$
$$\mathbf{CX} \le \mu,$$

де $\mathbf{A}$ – матриця коефіцієнтів a_{ij};

$\mathbf{X}$ – вектор-стовпчик змінних;

$\mathbf{B}$ – вектор-стовпчик вільних членів;

$\mathbf{C}$ – вектор-рядочок коефіцієнтів лінійної форми;

μ – лінійна форма, яка підлягає максимізації;

відповідає двоїста задача

$$\mathbf{A}^*\mathbf{Y} \ge \mathbf{C}^*,$$
$$\mathbf{Y} \ge 0,$$
$$\mathbf{B}^*\mathbf{Y} = \overline{\mu},$$

де $\mathbf{Y}$ – вектор-стовпчик змінних двоїстої задачі;

$\bar{\mu}$ – лінійна форма двоїстої задачі, яка підлягає мінімізації;

* – знак транспонування.

Доведено, що якщо одна із задач має розв'язок, то і двоїста їй також має розв'язок, а максимум лінійної форми μ дорівнює мінімуму форми $\bar{\mu}$. Крім того, обидві задачі пов'язані так званим правилом доповнюючої нежорсткості. В оптимальному розв'язку кожній рівності у прямій задачі відповідає строга нерівність у двоїстій, а кожній строгій нерівності прямої задачі – рівність у двоїстій. Наприклад, якщо одне з обмежень прямої задачі виконується як рівність $-\sum_{j=1}^{n} a_{ij}x_j = b_i$, то відповідна йому змінна двоїстої задачі y_i строго більше нуля. Маючи оптимальний план двоїстої задачі, легко побудувати відповідний план прямої і навпаки.

У методі послідовного уточнення оцінок розв'язку підлягає двоїста задача. Відправляючись від опорного плану $\mathbf{Y}$, визначаємо відповідний йому план початкової задачі $\mathbf{X}$. Якщо усі базисні змінні додатні, то знайдений оптимальний розв'язок. Якщо серед базисних є декілька від'ємних змінних, то найменша з них вказує, яку рівність необхідно виключити з опорного плану двоїстої задачі. Подальше покращення плану виконується за допомогою симплексного методу. За об'ємом обчислень обидва метода приблизно однакові, але другому необхідно віддати перевагу у випадку труднощів при відшукуванні вихідного опорного плану.

У методі послідовного скорочення помилок рух до точки оптимуму може також відбуватися ззовні, але тут компоненти вектора $\mathbf{X}$ повинні бути невід'ємними. Задоволення обмежень (1.2) необов'язкове. За рахунок цього вводиться у розгляд розширена задача (вихідна задача формулюється у вигляді рівностей) про мінімізацію форми

$$\mu' = \sum_{i=1}^{m} \varepsilon_i ,$$

при обмеженнях

$$\sum_{i=1}^{n} a_{ij} x_j + \varepsilon_i = b_i \quad (i = 1,2,\ldots,m),$$

$$\varepsilon_i \geq 0,\ x_j \geq 0,\ (j = 1,2,\ldots,n).$$

Нехай **Y** деякий план задачі, двоїстої вихідної. Виділимо множину E_Y індексів тих співвідношень двоїстої задачі, які виконуються як рівності. З планом **Y** може бути пов'язана деяка допоміжна задача, яка утворюється із розширеної введенням додаткових обмежень

$$x_j = 0 \text{ при } j \in E_Y.$$

Отримана задача розв'язується порівняно просто симплексним методом. Оптимальний розв'язок задачі, двоїстої до допоміжної, дає можливість побудувати новий план. На кожній ітерації значення мінімуму лінійної форми μ' зменшується. Коли сума помилок дорівнює нулю, отримуємо оптимальний розв'язок, адже за означенням усі компоненти вектора **X** невід'ємні і для кожного $x_j > 0$ дотримується рівність

$$\sum_{i=1}^{m} a_{ij} y_i = c_j .$$

Є ще градієнтні методи, які отримали основне розповсюдження для розв'язання задач нелінійного планування. Сутність цих методів полягає у наступному. Вибирається яка-небудь точка, яка належить багатогранній множині. Потім визначається напрямок градієнта спеціально організованої функції, обчислюється довжина кроку та робиться крок у цьому напрямку. Описані операції повторюються до тих пір, поки точка, яка зображує розв'язок, не досягне оптимальної вершини.

З точки зору математичного моделювання задач симплексний метод та метод уточнення оцінок (двоїстий симплексний метод) дають прості та чіткі алгоритми для організації ітераційного процесу. Питання про те, якому методу необхідно віддати перевагу, треба вирішувати у кожному конкретному випадку. Зводиться він до того, яку із задач – пряму чи двоїсту – простіше реалізувати на моделі.

Метод послідовного скорочення помилок володіє більш складною логікою, і навряд чи доцільно використовувати його для організації ітераційного процесу розв’язку. Однак він надає у розпорядження математичного моделювання дуже цінну ідею – ідею одночасного вирішення прямої та двоїстої задач. Метод градієнта не використовується для моделювання.

1.1.2. Електронні моделі задачі лінійного програмування

Розглянемо коротко існуючі методи моделювання систем лінійних алгебраїчних рівнянь та тісно пов’язані з ними методи моделювання задач лінійного програмування. Об’єктом моделювання є недовизначена система лінійних алгебраїчних рівнянь з додатковими обмеженнями у вигляді невід’ємності змінних. Крім того, на моделі необхідно тим або іншим чином реалізовувати умову максимуму (мінімуму) лінійної форми.

Один із розповсюджених методів моделювання систем алгебраїчних рівнянь – використання інтеграторів [3]. Система рівнянь, яка підлягає моделюванню, записується у вигляді

$$\begin{aligned} &a_{11}x_1+a_{12}x_2+\ldots+a_{1n}x_n+b_1=-\frac{dx_1}{dt}, \\ &\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \\ &a_{n1}x_1+a_{n2}x_2+\ldots+a_{nn}x_n+b_n=-\frac{dx_n}{dt}. \end{aligned} \qquad (1.4)$$

Розв’язок системи отримується при рівності нулю похідних $\frac{dx_i}{dt}$. Схема набору для розв’язку одного з рівнянь системи (1.4) наведена на рис. 1.1

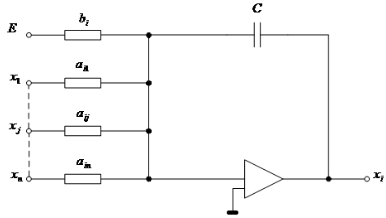


Рис. 1.1. Схема для розв’язку одного рівняння

Стійкий розв'язок отримується тільки при матриці $\mathbf{A}$, яка позитивно визначена. У випадку довільної неособливої матриці коефіцієнтів треба застосовувати різні штучні прийоми.

У схемі Маллока використовуються трансформатори для моделювання системи рівнянь з несиметричною матрицею A.

Багато з дослідників використовували властивості інтегруючого підсилювача. Так, Ферс запропонував схему для розв'язку задачі програмування з лінійними обмеженнями у вигляді нерівностей та цільової функції

$$F(x_1, \ldots, x_n).$$

Обмеження $f_i = a_{i1}x_1 + \ldots + a_{in}x_n - b_i \geq 0$ моделюються за допомогою схеми, представленої на рис. 1.2.

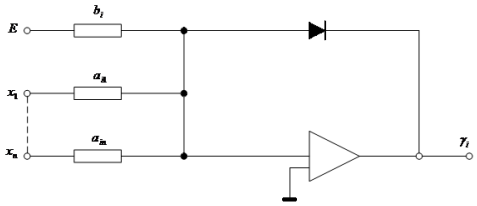


Рис. 1.2. Схема моделювання обмеження

Напруга на виході підсилювача γ_i дорівнює або нулю (при дотриманні нерівності), або більшій величині, коли $f_i < 0$. Ці напруги визначають процес мінімізації, так як вони разом з напругою $\dfrac{dF}{dx_j}$ подається на інтегруючий підсилювач (рис. 1.3).

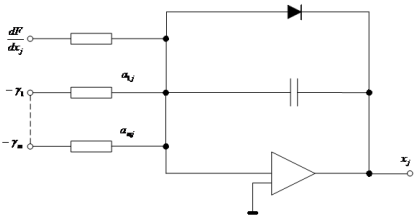


Рис. 1.3.Схема моделювання процесу мінімізації

Діод у зворотному зв'язку моделює обмеження $x_j \geq 0$. Коли мінімум цільової функції досягається на границі області, що,

зокрема, завжди виходить у задачах лінійного планування, розв'язок представляє собою коливальний режим в області точки мінімуму. Для реалізації метода необхідно
$(m+n)\div 2(m+n)$ підсилювачів.

Також є схеми для розв'язку алгебраїчних рівнянь та задач лінійного планування за допомогою підсилювачів з ємністю у зворотному зв'язку.

Розглянемо методи, які використовують властивості прямої та двоїстої їй задач лінійного програмування. Відомо, що система рівнянь, яка складається з обмежень прямої та двоїстої задач з додаванням умов доповнюючої нежорсткості або умови рівності лінійних форм, має єдиний розв'язок. Він є оптимальним для прямої та двоїстої їй задач. Перша обставина використовується у методі, запропонованому І.М. Тетельбаумом та Р. Калманом. Моделюються наступні рівняння:

$$\mathbf{E}\xi = \mathbf{AX} - \mathbf{B}, \tag{1.5}$$

$$\mathbf{E}\eta = \mathbf{A}^*\mathbf{Y} - \mathbf{C}^*, \tag{1.6}$$

$$\mathbf{X} \geq 0, \eta \geq 0, \mathbf{X}^*\eta = 0, \tag{1.7}$$

$$\mathbf{Y} \geq 0, \xi \leq 0, \mathbf{Y}^*\xi = 0. \tag{1.8}$$

Тут $\mathbf{E}$ – одинична матриця; рівняння (1.5) та (1.6) отримані із вихідної системи введенням вектора-стовпчика додаткових змінних ξ та η відповідно; (1.7) та (1.8) – умови доповнюючої не жорсткості.

Схема моделі наведена на рис. 1.4. Вона описується наступними рівняннями:

першим законом Кірхгофа

$$\begin{aligned} \mathbf{EI}_\xi &= \mathbf{Y}_A\mathbf{U}_X - \mathbf{Y}_B\mathbf{U}_B - \mathbf{D}_1\mathbf{U}_Y + \mathbf{Y}_A\varepsilon, \\ \mathbf{EI}_\eta &= \mathbf{Y}_A^*\mathbf{U}_Y - \mathbf{Y}_C\mathbf{U}_C - \mathbf{D}_2\varepsilon; \end{aligned} \tag{1.9}$$

рівняннями діодів

$$\begin{aligned} \mathbf{I}_\xi^*\mathbf{U}_Y = 0, \mathbf{I}_\xi \leq 0, \mathbf{U}_Y \geq 0, \\ \mathbf{I}_\eta^*\mathbf{U}_X = 0, \mathbf{I}_\eta \geq 0, \mathbf{U}_X \geq 0, \end{aligned} \tag{1.10}$$

де $\mathbf{D}_1$, $\mathbf{D}_2$ – діагональні матриці, $\mathbf{Y}_A$ – матриця провідностей.

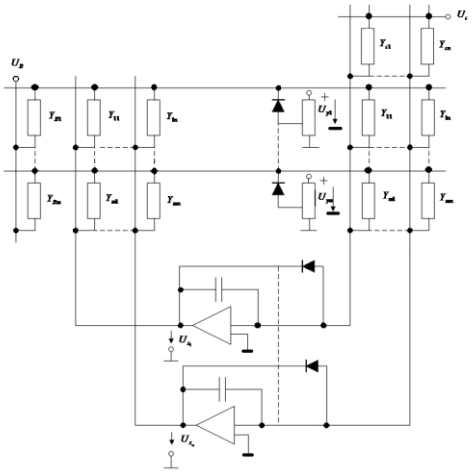


Рис. 1.4. Схема для моделювання прямої та двоїстої задачі

Якщо вибирати рівні напруг $\mathbf{U}_X$ та $\mathbf{U}_Y$ так, щоб

$$\mathbf{U}_X >> \mathbf{U}_Y >> \varepsilon, \tag{1.11}$$

то членами $\mathbf{D}_1\mathbf{U}_Y$, $\mathbf{Y}_A\varepsilon$, $\mathbf{D}_2\varepsilon$ у рівняннях (1.11) можна знехтувати. Тоді система (1.9) та (1.10) буде аналогічна системі (1.5) – (1.8). Основними недоліками схеми є труднощі реалізації нерівності (1.11), а також вплив не ідеальності діодів на точність розв'язку задачі. Дану схему можна отримати із схеми квазіоберненого перетворювача шляхом включення діодів у вузлах $\mathbf{Y}$ та введенням вектора струмів $\mathbf{I}_C = \mathbf{Y}_C\mathbf{U}_C$ у вузлі ε.

Схема, запропонована Рибашовим та Дудніковим, розв'язує систему диференціальних рівнянь, які відповідають прямій та двоїстій задачам. Система рівнянь має єдиний стійкий розв'язок – у точці оптимуму. Але для реалізації метода необхідно десь вдвоє більше вирішуючи підсилювачів, ніж у схемі Пайна.

У роботі Денніса використана аналогія між задачами теорії електричних кіл та задачами математичного програмування. Рівняння деяких кіл, які містять джерела напруги та струму, діоди, опори та трансформатори постійного струму, подібні обмеженням задач математичного програмування. Потужність у колі подібна цільовій функції, а згідно законам електричних кіл будь-який розподіл струмів та напруг у колі задовольняє мінімуму

потужності, яка розсіюється. Таким чином, вектори струмів та напруг у колі подібні оптимальним векторам відповідної задачі математичного програмування.

1.1.3. Ітераційні методи моделювання загальної задачі за допомогою зворотних та квазіобернених лінійних перетворень

Електронні та сіткові моделі, які застосовуються для дослідження тих або інших об'єктів, є не оберненими [3]. Це означає, що їх затиски чітко діляться на вхідні та вихідні. Наприклад, у моделі рівняння

$$a_1x_1 + a_2x_2 + a_3x_3 + b = 0\,, \qquad (1.12)$$

схема якої зображена на рис. 1.5, вхідними є полюси *1, 2, 3*, а вихідним – полюс 0. Тут можна задавати тільки величини x_1, x_2, x_3. Задати на вихідному полюсі функцію b з тим, щоб отримати на вихідних x_1, x_2, x_3, у даній моделі неможливо. Якщо необхідно розв'язати систему рівнянь відносно змінних, які спочатку були заданими, необхідно знов підготовити систему до набору.

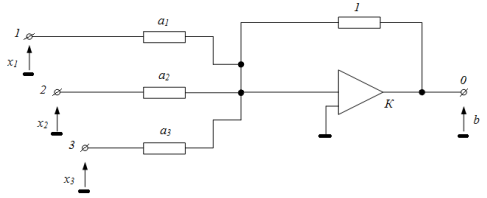


Рис. 1.5. Схема моделі рівняння

Дуже часто ця операція викликає великі труднощі. В зворотних та квазіобернених моделях усі зовнішні полюси рівноправні. При моделюванні несуперечливих математичних залежностей їх можна ділити на вхідні та вихідні довільним чином. У зворотних моделях це можна робити без будь-якої комутації, а у квазіобернених – за допомогою простих ключових схем. Для наочності покажемо, як розв'язується розглянуте вище рівняння на зворотній моделі (рис. 1.6).

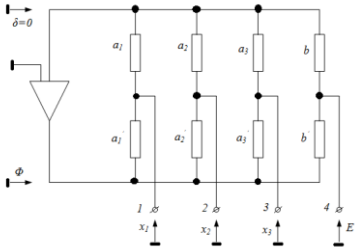


Рис. 1.6. Схема зворотної моделі

Схема зворотна відносно полюсів *1, 2, 3, 4*. Якщо задати на полюсах *1, 2, 3* функції x_1, x_2, x_3 у вигляді напруг, то на полюсі *4* отримаємо напругу, яка моделює b. При завданні функції b на полюсах *1, 2, 3* отримаємо напруги x_1, x_2, x_3, які представляють собою допустимий розв'язок вихідного рівняння. Воно визначається системою рівнянь, які описують дану схему:

$$a_1x_1 + a_2x_2 + a_3x_3 + b = (a_1 + a_2 + a_3 + 1)\varepsilon, \tag{1.13}$$

$$\begin{aligned} &(a_1 + a_1')\tilde{x}_1 - a_1\varepsilon - a_1'\Phi = 0,\\ &(a_2 + a_2')\tilde{x}_2 - a_2\varepsilon - a_2'\Phi = 0, \Phi = -k\varepsilon,\\ &(a_3 + a_3')\tilde{x}_3 - a_{31}\varepsilon - a_3'\Phi = 0, \end{aligned} \tag{1.14}$$

де a_1, a_2, a_3 – провідності, які пропорційні коефіцієнтам рівняння;

k – коефіцієнт підсилення підсилювача.

Величина коефіцієнту підсилення реальних підсилювачів досить достатня, щоб напруга ε на вході представляла собою машинний нуль. При $\varepsilon = 0$ рівняння (1.13) приймає вигляд (1.12).

Схема рис. 1.6 має один суттєвий недолік – низькі рівні вихідних напруг, так як допоміжні провідності споживають великі струми. Можна підвищити значення напруг, застосувавши квазіобернену модель. Тут напруга, яка моделює змінну, виходить на виході підсилювача. Однак змінювати полюси так, як це робиться у зворотному підсилювачі, неможна. Для того аби поміняти місцями задану та шукану величини, необхідно виконати переключення. На рис. 1.7 показана схема квазіоберненого перетворювача, який моделює рівняння (1.12) при заданих b_1, x_2 та x_3.

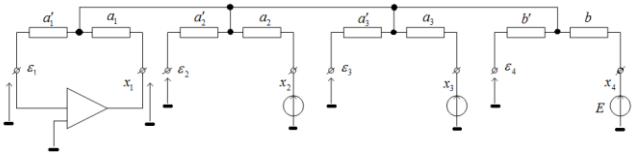


Рис. 1.7. Схема квазіоберненого перетворювача

Напруга на виході оброблюючого підсилювача буде пропорційна x_1. Зворотні та квазіобернені підсилювачі застосовуються також для розв'язку систем лінійних алгебраїчних рівнянь. На рис. 1.8 наведена схема зворотного лінійного перетворювача для розв'язку невизначеної системи рівнянь

$$\mathbf{AX} = \mathbf{B}, \qquad (1.15)$$

де $\mathbf{A} = \{a_{ji}\}$ – матриця коефіцієнтів;

$\mathbf{X}$ – вектор-стовпчик невідомих;

$\mathbf{B}$ – вектор-стовпчик правих частин.

Припускаємо, що усі коефіцієнти $a_{ji} \geq 0$. Якщо у вихідній матриці є від'ємні коефіцієнти, то її легко перетворити у матрицю з додатними коефіцієнтами. Кожний рядок матриці моделюється одним підсилювачем та двополюсниками провідностей g_{ji}, права частина – джерело струму $\mathbf{I}_i$.

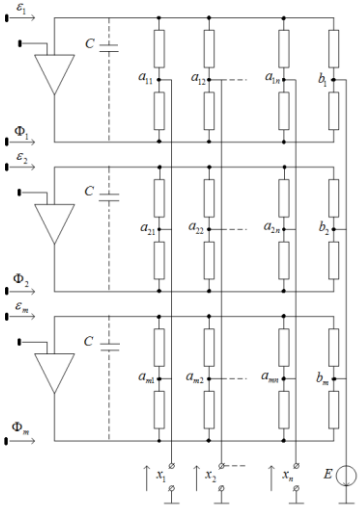


Рис. 1.8. Схема зворотного лінійного перетворювача

Запишемо рівняння схеми 1.8 для вузлів ε та X в усталеному режимі:

$$-\mathbf{m}\varepsilon + \mathbf{A}\mathbf{X} = \mathbf{B},$$
$$\mathbf{A} * \varepsilon - \mathbf{n}\mathbf{X} = -\mathbf{A} * \mathbf{\Phi}, \qquad (1.16)$$
$$-k\varepsilon = \mathbf{\Phi},$$

де ε – вектор напруг на входах підсилювачів;

$\mathbf{B}$ – вектор струмів, який моделює праву частину;

$\mathbf{\Phi}$ – вектор напруг на виходах підсилювачів;

k – коефіцієнт підсилення підсилювача;

$\mathbf{m}$ та $\mathbf{n}$ – діагональні матриці власних провідностей у вузлах ε та X відповідно.

Система (1.16) еквівалентна системі (1.15) за умовою рівності нулю вектора ε. При заданих будь-яких $n-m$ напруг на полюсах X на m полюсах, які залишилися, ми отримаємо значення шуканих невідомих – модель перетворює змінні $X_{(n-m)}$ у змінні $X_{(n)}$ згідно матриці $\mathbf{A}$. Якщо змінні $X_{(n-m)}$ не задавати, отримаємо допустиме рішення рівняння (1.31), яке визначається системою (1.16). Дана схема зручна у випадку, коли кількість невідомих значно перевищує кількість рівнянь.

Однак рівні напруг, які моделюють компоненти вектора, нижчі (у деяких випадках у десятки раз) напруг на входах підсилювачів. Щоб уникнути цього, при моделюванні певної системи рівнянь можна включити від'ємні провідності у вузли x. Величина кожної з них повинна дорівнювати сумі власних провідностей вузла. Для цієї цілі можна використати квазівід'ємні опори. Одна з можливих схем наведена на рис. 1.9.

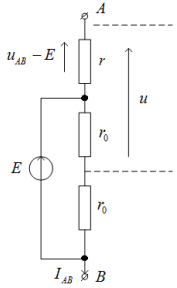


Рис. 1.9. Схеми із використанням квазівід'ємних опорів

Цю схему описують наступні рівняння:

$$U_{AB} = \frac{1}{2}E + U = -rI_{AB} + 2u,$$

$$\frac{U_{AB}}{r} - \frac{E}{r} = U_{AB}g - Eg.$$

Якщо підібрати величину ЕРС E такою, щоб напруга $U = 0$, то між затисками A та B отримаємо від'ємний опір (провідність). Припустимо тепер, що такі двополюсники включені у вузли X схеми та $g_i = \sum_{j=1}^{m} 2g_{ij}$. Тоді замість (1.16) виконуються наступні залежності:

$$\begin{aligned} &-\mathbf{m}\varepsilon + \mathbf{AX} = \mathbf{B}, \\ &\mathbf{A} * \varepsilon - \mathbf{2nX} = -\mathbf{A} * \mathbf{\Phi} - \tilde{\mathbf{n}}\mathbf{E}, \\ &\mathbf{E} = \mathbf{2X} - \mathbf{U}, \mathbf{\Phi} = -k\varepsilon, \end{aligned} \qquad (1.17)$$

де $\mathbf{E}$ – матриця-стовпчик ЕРС квазівід'ємних опорів;

$\mathbf{U}$ – матриця-стовпчик їх помилок.

Напруга ε_i – машинні нулі, напруги U_j зводиться до нуля регулюванням ЕРС E_j. Доведено, що процес урівноваження квазівід'ємних опорів сходиться теоретично за один крок, після якого (1.17) буде мати вигляд

$$\begin{aligned} &\mathbf{AX} = \mathbf{B}, \\ &\mathbf{A} * \mathbf{\Phi} = \mathbf{0}, \\ &\mathbf{E} = \mathbf{2X}, \mathbf{\Phi} = -k\varepsilon. \end{aligned}$$

Практично величини $\mathbf{\Phi}_i$ дуже мало відрізняються від нуля. Збільшення масштабу величин, які задаються, призведе до підвищення та, відповідно, до зменшення похибок. Однак з технічної точки зору у схемі є одна незручність – напруга U квазівід'ємних опорів не має загальної точки.

Схема квазіоберненого лінійного перетворювача також дозволяє моделювати системи алгебраїчних рівнянь. Напруги на виходах підсилювачів моделюють невідомі (рис. 1.10).

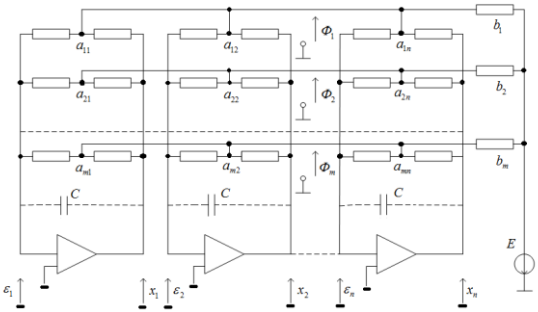


Рис. 1.10. Схема квазіоберненого лінійного перетворювача

Схема описується рівняннями

$$-\mathbf{m}\varepsilon + \mathbf{A} * \mathbf{Y} = \mathbf{0},$$
$$\mathbf{A}\varepsilon - \mathbf{n}\mathbf{Y} + \mathbf{A}\mathbf{X} = \mathbf{B}, \qquad (1.18)$$
$$\mathbf{X} = -k\varepsilon,$$

де $\mathbf{Y}$ – вектор напруг у середніх точках двополюсників;

$\mathbf{X}$ – вектор напруг на виходах підсилювачів;

$\mathbf{m}$ та $\mathbf{n}$ – діагональні матриці власних провідностей у вузлах ε та X відповідно.

При достатньо великому значенні k вектор ε прямує до нуля. Як наслідок, прямує до нуля і вектор $\mathbf{Y}$, оскільки

$$\mathbf{A} * \mathbf{Y} = \mathbf{0}.$$

Тоді друге рівняння системи (1.18) перетворюється у рівняння (1.15). У випадку, коли задані $n-m$ змінних, відповідні підсилювачі треба відключити і вузли X, які звільнилися, з'єднати з джерелами ЕРС, які моделюють вільні змінні. Схема перетворює вектор $\mathbf{X}_{(n-m)}$ у вектор $\mathbf{X}_{(n)}$ у відповідності з матрицею $\mathbf{A}$.

Доведено, що обидві описані моделі систем алгебраїчних рівнянь абсолютно стійкі.

Оскільки обидві схеми дозволяють визначити m невідомих при заданих $n-m$ будь-яких інших змінних, ними можна скористатися для реалізації ітераційного процесу розв'язання загальної задачі лінійного планування. Обмеження типу $x_i \geq 0$, $x_i \leq 0$ моделюються за допомогою діодів.

Розглянемо реалізацію метода, близького до симплексного, на зворотному лінійному перетворювачі (рис. 1.11).

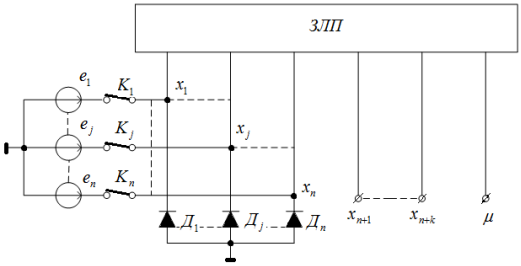


Рис. 1.11. Схема зворотного лінійного перетворювача

Припускається, що система обмежень сумісна і величина мінімуму (максимуму) на множині допустимих рішень системи кінцева. Метод забезпечує неминучу збіжність процесу розв'язку за кінцеве число кроків. Умова невід'ємності виконується за допомогою діодів $D_1,\ldots,D_j,\ldots,D_n$. Напруга джерел ЕРС, які регулюються $e_1,\ldots,e_j,\ldots,e_n$ моделюють вільні змінні, ключі $K_1,\ldots,K_j,\ldots,K_n$ з'єднують полюси зворотного лінійного перетворювача (ЗЛП) з джерелами e_j або з діодами D_j.

За початкове приймається розв'язок, отриманий при відключених джерелах e_j. Потім встановлюються $n-m$ ЕРС, рівні напругам на будь-яких $n-m$ полюсах ЗЛП, і замикаються відповідні ключі. Одне з підключених джерел ЕДС регулюється так, щоб величина μ змінювалася в сторону досягнення оптимуму. Якщо це не вдається, вибирається інше джерело ЕРС e_s, а перше з ЕРС $e_j=0$ залишається підключеним до схеми ЗЛП. У процесі регулювання e_s може або досягнути нуля при додатних величинах напруг на вільних полюсах, або при додатній e_s обертається у нуль одна або декілька вільних змінних. У першому випадку необхідно перейти до регулювання іншої ЕРС e_{s+1}. ЕРС $e_s=0$ залишається підключеною до полюсу ЗЛП. У другому випадку у момент переходу через нуль однієї (або кількох) вільної ЕРС закінчуються регулювання. Її відключають від полюса ЗЛП (розмикають ключ K_s), а полюс з $x_j=0$ з'єднують з ЕРС $e_j=0$. Описане переключення не впливає на розподіл напруг у схемі. Ці операції

повторюються до тих пір, поки ніяке регулювання будь-якої ЕРС не призводить до зміни μ у сторону оптимуму. Оскільки кожний крок призводить до збільшення (зменшення) лінійної форми, розв'язку буде досягнуто за кінцеве число кроків.

Цей же процес можна реалізувати і на квазіоберненому лінійному перетворювачі (КОЛП). На рис. 1.12 ε_j, ε_μ – напруги на входах підсилювачів, $\Phi_j = x_j$ та $\Phi_\mu = \mu$ – напруги на їх виходах. Інші позначення ті ж, що на рис. 1.11.

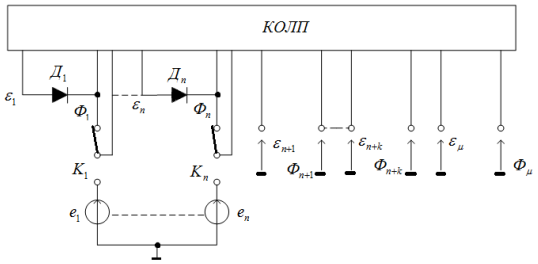


Рис. 1.12. Квазіобернений лінійний перетворювач

Вибрані $n-m$ підсилювачів відключають від схеми, а замість них включають $n-m$ джерел ЕРС, величини напруг яких пропорційні змінним допустимого розв'язку. Подальший процес розв'язку такий же, як і на ЗЛП.

Описаний метод моделювання дозволяє уникнути похибок із-за не ідеальності діодів. Після того, як отриманий початковий розв'язок, вони не приймають участь у роботі схеми. Це – достоїнство метода. До недолікам необхідно віднести те, що процес розв'язку багатокроковий і, отже, потребує відносно великого часу.

Також описаний двох кроковий метод. Оснований на примусовій зміні напруги, яка моделює μ. При цьому рівняння

$$c_1 x_1 + \ldots + c_j x_j + \ldots + c_n x_n = \mu$$

моделюється так само, як рівняння системи $\mathbf{AX} = \mathbf{B}$. Величина μ представляє собою праву частину. Ясно, що μ може приймати різні значення у межах допустимих розв'язків системи обмежень. Найбільше (найменше) значення μ знаходиться у вершині

багатокутника обмежень. Подальше збільшення (зменшення) μ призводить до несумісності системи рівнянь.

На рис. 1.13 приведена схема, яка реалізує цей метод на ЗЛП.

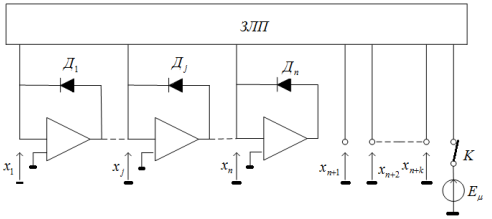


Рис. 1.13. Схема, яка реалізує метод ЗЛП

Підсилювачі включені для покращення властивостей діодів. На першому кроці вимірюють величину μ, яка відповідає допустимому рішенню системи. Ключ K розімкнутий. Встановивши величину E_μ рівній виміряній напрузі, замикають ключ. Другий крок полягає у регулюванні ЕРС у необхідному напрямку до досягнення оптимуму. Сигналом про це може слугувати скачко подібний перехід напруг на виходах підсилювачів в область нелінійної амплітудної характеристики. Як тільки $E_\mu > E_{\mu\text{макс.}}$ ($E_\mu < E_{\mu\text{мін.}}$) система рівнянь стає несумісною та виникають помилки ε_i. Якщо тепер величину E_μ зменшити до $E_{\mu\text{макс.}}$ (збільшити до $E_{\mu\text{мін.}}$), $\varepsilon_i = 0$ та напруги на виходах підсилювачів буде знаходитися на лінійній ділянці амплітудної характеристики.

Доволі часто умови задачі задаються у вигляді двосторонніх обмежень:

$$\mathbf{B_1} \le A\mathbf{X} \le \mathbf{B_2}.$$

Зазвичай для моделювання на зворотному лінійному перетворювачі їх приводять спочатку до односторонніх нерівностей, а потім – до рівнянь. Кількість рівнянь при цьому збільшується вдвічі, а кількість невідомих – на $2m$. Очевидно, збільшується об’єм апаратури, яка використовується при моделюванні.

Можна уникнути цього, застосувавши обмежувачі, які дозволяють моделювати обмеження безпосередньо у вигляді нерівностей.

Напруги, які підпирають діоди, пропорційні величинам b_1 та b_2 (компонентам векторів $\mathbf{B_1}$ та $\mathbf{B_2}$). Загальна точка кожної пари діодів з'єднується з відповідною точкою b_i зворотного підсилювача, який моделює i-й рядок обмежень (див. рис. 1.8 – при відсутності джерела ЕРС E та зв'язаних з ним з'єднань). У цьому випадку на полюсі b_i оброблюється напруга, пропорційна величині

$$y_i = \frac{\sum_{j=1}^{n} a_{ij} x_j}{b_i}.$$

Тут b_i – провідність, яка відповідає вказаній на рис. 1.8. Для зручності можна взяти провідність, яка дорівнює одиниці, як це прийнято у даній схемі. Коли обмеження зверху виконується як рівність, працює один діод, коли знизу – інший.

Оптимальний розв'язок можна отримати або методом примусової зміни величини цільової функції, або методом включення джерела струму, або організувавши ітераційний процес.

1.2. Безітераційні методи моделювання

Розглянуті вище схеми дозволяють отримати розв'язок за певну кількість кроків [3]. Але за допомогою обернених та квазіобернених лінійних перетворювачів можна також будувати моделі, які дозволяють отримати розв'язок відразу після завдання вихідних даних.

Один із можливих методів полягає в одночасному розв'язку прямої та двоїстої їй задачі з додатковою умовою у вигляді рівності цільових функцій. Із теорії лінійного програмування відомо, що така розширена задача має єдиний розв'язок. Він є оптимальним для обох (прямої та двоїстої) задач.

Система, яка моделюється, має вигляд

$$\mathbf{AX} = \mathbf{B},$$

$$\mathbf{A}^*\mathbf{Y} = \mathbf{C}^*,$$

$$\mathbf{C}^*\mathbf{X} = \mathbf{B}^*\mathbf{Y},$$

$$\mathbf{X} \geq \mathbf{0}, \mathbf{Y} \geq \mathbf{0},$$

де **A** – матриця коефіцієнтів обмежень;

X – вектор-стовпчик змінних прямої задачі;

B – вектор-стовпчик вільних членів прямої задачі;

C – вектор-рядочок коефіцієнтів цільової функції прямої задачі;

Y – вектор-стовпчик змінних двоїстої задачі.

На рис. 1.15 приведена схема для розв'язку наступного прикладу.

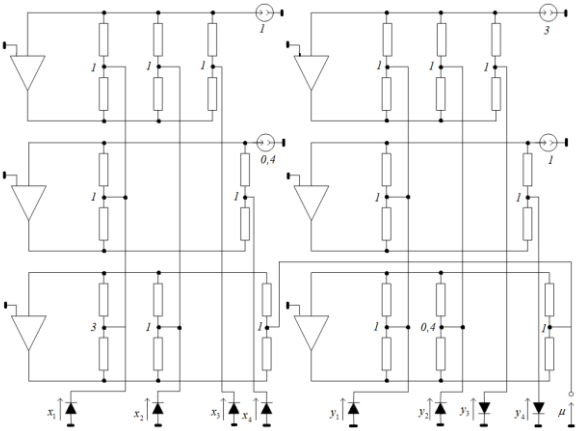


Рис. 1.15. Схема для розв'язання рівняння

Пряма задача

$$\begin{aligned} x_1 + x_2 + x_3 &= 1, \\ x_1 + x_4 &= 0{,}4, \\ 3x_1 + x_2 &= \mu \to \max. \end{aligned}$$

Двоїста задача

$$y_1 + y_2 + y_3 = 3,$$
$$y_1 + y_4 = 1,$$
$$y_1 + 0{,}4y_2 = \mu \to \min,$$
$$y_{1,2} \geq 0, y_{3,4} \leq 0.$$

Обидві задачі розв'язуються на обернених перетворювачах.

Для підвищення рівня змінних у вузли x можна включити квазівід'ємний опір. Оскільки оптимальний розв'язок є для даної схеми єдиним, процес урівноваження квазівід'ємних опорів сходиться за один цикл.

При побудові моделей для задач з обмеженнями у вигляді нерівностей діоди необхідні в обох частинах моделі, так як обмеження невід'ємності (недодатності) накладаються як на змінні прямої задачі, так і на змінні двоїстої. Коли нерівності моделюються за допомогою діодних обмежувачів, похибка із-за не ідеальності діодів залишається майже такою ж, так як кількість працюючих діодів така ж.

Якщо основні обмеження прямої задачі задані у вигляді рівностей, на змінні двоїстої задачі не накладаються ніякі обмеження. Це призводить до зменшення кількості діодів, які використовуються, і, отже, зменшенню похибки результатів розв'язку.

При виборі масштабу коефіцієнтів лінійної форми прямої задачі необхідно обов'язково враховувати, що величина U_μ для обох схем одна й та ж.

Описаний метод особливо зручний для моделювання матричних ігор. Відомо, що ці задачі теорії ігор легко звести до пари двоїстих задач лінійного програмування. Тоді змінні X та Y представляють собою частоти застосування стратегії двома гравцями, а μ – ціну гри.

Розглянемо ще одну схему, яка дозволяє отримати розв'язок безпосередньо після вводу вихідних даних (рис. 1.16). Напруга U_μ, яка моделює цільову функцію, встановлюється на затисках джерела струму I_μ.

Роботу схеми можна пояснити наступним чином.

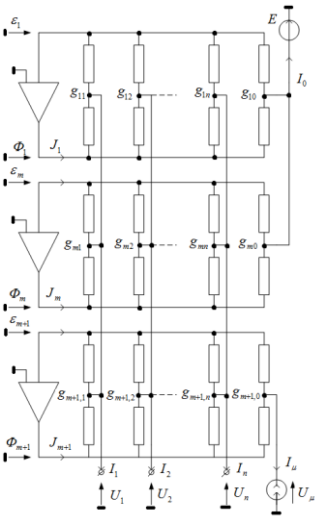


Рис. 1.16. Схема для розв'язку рівняння

При $I_\mu = 0$ вихідні напруги представляють собою допустимий розв'язок системи обмежень. Напруга на затисках джерела стуму моделює відповідне цьому розв'язку значення цільової функції. Частина діодів відкрита, частина закрита. Схему ЗЛП відносно затисків джерела струму можна замінити еквівалентними опорами $R_{\text{екв}}$ та джерелом ЕРС $\Phi_{\text{екв}}$. Якщо кількість відкритих діодів менша $n-m$, еквівалентний опір схеми додатний. Величина його визначається параметрами схеми і тим, які діоди відкриті. Якщо число відкритих діодів дорівнює $n-m$, система

$$\mathbf{AX} = \mathbf{B}$$

визначена і, отже, вихідний (еквівалентний) опір дорівнює нулю.

Дамо струму I_μ приріст $+\Delta I_\mu$. Якщо при цьому жодний з діодів не переходить у інший стан, напруга U_μ на затисках джерела струму зростає на величину $\Delta I_\mu R_{\text{екв}}$. Якщо ж один або відразу декілька діодів переходять у інший стан, схема буде характеризуватися значенням $R_{\text{екв}}$ та $\Phi_{\text{екв}}$.

Можна зняти залежність $U_\mu = f(I_\mu)$. Вона буде представляти собою ламану криву. Нахил кожної з ділянок кривої визначається опором $R_{\text{екв}}$ при відповідних комбінаціях відкритих та закритих діодів. Ламаний характер кривої обумовлений наявністю діодів у схемі.

Число комбінацій, у яких можуть знаходитися відкриті та закриті діоди, обмежене і визначається властивостями рівнянь, які описують схему. Отже, починаючи з деякого значення струму I_μ, ніяке його збільшення не викличе зміну стану діодів. Якщо б схема у цьому стані мала додатний опір $R_{\text{екв}}$, напруга U_μ збільшувалася б разом із зростанням струму I_μ до нескінченності. Однак цього не відбувається. Коли U_μ досягає значення $U_{\mu\text{макс}}$, $n-m$ змінних дорівнюють нулю. Система рівнянь визначена, і вихідний опір схеми $R_{\text{екв}} = 0$. Остання ділянка кривої паралельна осі струму.

Отже, починаючи з деякого значення $I_{\mu 0}$, напруга U_μ постійна.

Все сказане можна повторити і для нижньої гілки кривої – для від'ємних приростів струму: тільки замість $U_{\mu\text{макс}}$ там буде $U_{\mu\text{мін}}$.

Отже, задавши достатньо великий струм I_μ додатної (від'ємної) полярності, ми отримаємо розв'язок, який відповідає максимуму (мінімуму) лінійної форми.

Якщо у вузли «X» включити квазівід'ємні опори, які урівноваження їх, як і у попередній схемі, відбувається за один цикл.

1.3. Задача нелінійного програмування

У загальному вигляді задача нелінійного програмування полягає у визначенні максимального (мінімального) значення функції [9]

$$f(x_1, x_2, \ldots, x_n), \tag{1.19}$$

за умовою

$$\left.\begin{array}{l} g_i(x_1, x_2, \ldots, x_n) \le b_i, \quad i = \overline{1,k}, \\ g_i(x_1, x_2, \ldots, x_n) = b_i, \quad i = \overline{k+1,m}, \end{array}\right\}, \tag{1.20}$$

де f та g_i – деякі відомі функції n змінних, а b_i – задані числа.

Клас задач нелінійного програмування ширше класу задач лінійного програмування. Детальніше вивчення практичних задач, які домовилися вважати лінійними, показує, що вони у дійсності є нелінійними. Існуючі методи дозволяють вирішувати вузький клас задач, обмеження яких мають вигляд $g_i(x_1, x_2, \ldots, x_n) = \sum_{j=1}^{n} a_{ij} x_j$ $(i = \overline{1,m})$, а цільова функція є сепарабельною (сумою n функцій $\varphi_j(x_j)$), або квадратичною.

Навіть якщо область допустимих рішень – опукла, то у ряді задач цільова функція може мати декілька локальних екстремумів. За допомогою більшості ж обчислювальних методів можна знайти точку локального оптимуму, але неможна встановити, чи є вона точкою глобального (абсолютного) оптимуму або ні. Якщо задача містить нелінійні обмеження, то область допустимих рішень не є опуклою і крім глобального оптимуму можуть існувати точки локального оптимуму.

Розглянемо методи, які використовують у нелінійному програмуванні.

Метод множників Лагранжа. Він є класичним методом рішення задач математичного програмування (зокрема опуклого). На жаль, при практичному застосуванні метода можуть зустрітися значні обчислювальні труднощі, які звужують область його використання.

Розглянемо окремий випадок загальної задачі нелінійного програмування (1.19)–(1.20), припускаючи, що система обмежень (1.20) містить тільки рівняння, відсутні умови невід'ємності змінних, $f(x_1, x_2, \ldots, x_n)$ та $g_i(x_1, x_2, \ldots, x_n)$ – функції, неперервні разом із своїми частковими похідними. Обмеження у задачі задані рівняннями, тому для її розв'язку можна скористатися класичним методом відшукування умовного екстремуму функцій декількох

змінних. Вводять набір змінних $\lambda_1, \lambda_2, \ldots, \lambda_m$, які називаються множниками Лагранжа, і складають функцію Лагранжа

$$F(x_1, x_2, \ldots, x_n, \lambda_1, \lambda_2, \ldots, \lambda_m) = f(x_1, x_2, \ldots, x_n) + \\ + \sum_{i=1}^{m} \lambda_i (b_i - g_i(x_1, x_2, \ldots, x_n)),$$

знаходять часткові похідні

$$\frac{\partial F}{\partial x_j} (j = \overline{1, n}), \quad \frac{\partial F}{\partial \lambda_i} (i = \overline{1, m})$$

і розглядають систему $n+m$ рівнянь

$$\begin{cases} \dfrac{\partial F}{\partial x_j} = \dfrac{\partial f}{\partial x_j} - \sum\limits_{i=1}^{m} \lambda_i \dfrac{\partial g_i}{\partial x_j} = 0, & j = \overline{1, n}, \\ \dfrac{\partial F}{\partial \lambda_i} = b_i - g_i(x_1, x_2, \ldots, x_n) = 0, & i = \overline{1, m} \end{cases} \tag{1.21}$$

з $n+m$ невідомими $x_1, x_2, \ldots, x_n, \lambda_1, \lambda_2, \ldots, \lambda_m$. Розв'язавши систему рівнянь (1.21), отримують усі очки, у яких функція (1.19) може мати екстремальні значення. Подальше дослідження знайдених точок проводять так само, як і випадку безумовного екстремуму. Метод множників Лагранжа має обмежене застосування, так як система (1.21), як правило, має декілька розв'язків.

Приклад. Знайти точку умовного екстремуму функції $f(x) = x_1 x_2 + x_2 x_3$ при обмеженнях

$$\begin{cases} x_1 + x_2 = 2, \\ x_2 + x_3 = 2. \end{cases}$$

Складемо функцію Лагранжа:

$$F(x_1, x_2, x_3, \lambda_1, \lambda_2) = x_1 x_2 + x_2 x_3 + \lambda_1 (x_1 + x_2 - 2) + \\ + \lambda_2 (x_2 + x_3 - 2).$$

Продиференціюємо її по змінним $x_1, x_2, x_3, \lambda_1, \lambda_2$. Прирівнюючи отримані вирази до нуля, отримаємо наступну систему рівнянь:

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

З другого та третього рівнянь випливає, що $\lambda_1 = \lambda_2 = -x_2$; тоді

$$\begin{cases} x_1 - 2x_2 + x_3 = 0, \\ x_1 + x_2 = 2, \\ x_2 + x_3 = 2. \end{cases}$$

Розв'язавши дану систему, отримаємо:

$$x_1^* = x_2^* = x_3^* = 1 \text{ і } f(X^*) = 2.$$

Градієнтні методи. Використовуючи градієнтні методи, можна знайти розв'язок будь-якої задачі нелінійного програмування. Застосування цих методів у загальному випадку дозволяє знайти точку локального екстремуму. Тому більш доцільно використовувати їх для знаходження розв'язку задач опуклого програмування. Процес знаходження розв'язку задачі за допомогою градієнтних методів полягає у тому, що починаючи з деякої точки $\mathbf{X}^{(0)}$ здійснюється послідовний перехід до деяким іншим точкам до тих пір, поки не буде знайдено задовільний розв'язок вихідної задачі. Градієнтні методи можуть бути поділені на дві групи.

До першої групи відносяться методи, при використанні яких точки, які досліджуються, не виходять за межі області допустимих рішень задачі. У даному випадку найбільш розповсюдженими є метод Франка-Вульфа. До другої – методи, при використанні яких точки, які досліджуються, можуть як належати, так і не належати області допустимих рішень. Однак у результаті реалізації ітераційного процесу знаходиться точка області допустимих рішень, яка визначає задовільний розв'язок. Найбільш часто використовуються метод штрафних функцій та метод Ерроу-Гурвіца.

При знаходженні розв'язку задачі градієнтними методами ітераційний процес продовжується до тих пір, поки градієнт функції $f(x_1, x_2, \ldots, x_n)$ у черговій точці $\mathbf{X}^{(k+1)}$ не стане рівним нулю або ж поки не виконається нерівність $\left| f(\mathbf{X}^{(k+1)}) - f(\mathbf{X}^{(k)}) \right| < \varepsilon$, де $\varepsilon > 0$ (точність отриманого розв'язку).

Градієнтні методи, як правило, дозволяють отримати точне рішення за нескінченне число кроків і тільки у деяких – за кінцеве. У зв'язку з цим градієнтні методи відносять до наближених методів розв'язку.

Метод Франка-Вульфа. Нехай необхідно знайти максимальне значення ввігнутої функції

$$f(x_1, x_2, \ldots, x_n) \tag{1.22}$$

за умов

$$\sum_{j=1}^{n} a_{ij} x_j \le b_i, \; i = \overline{1, m}, \tag{1.23}$$

$$x_j \ge 0, \quad j = \overline{1, n}. \tag{1.24}$$

Обмеження містять тільки лінійні нерівності. Ця особливість є основною для заміни у межах точки, яка досліджується, нелінійної цільової функції лінійною, у результаті чого розв'язок вихідної задачі зводиться до послідовного розв'язку задач лінійного програмування.

Процес знаходження розв'язку починають з визначення точки, яка належить області допустимих рішень. Нехай це точка $\mathbf{X}^{(0)}$. Обчислюють у цій точці градієнт функції (1.22):

$$\nabla f(\mathbf{X}^{(0)}) = \left(\frac{\partial f(\mathbf{X}^{(0)})}{\partial x_1}, \frac{\partial f(\mathbf{X}^{(0)})}{\partial x_2}, \ldots, \frac{\partial f(\mathbf{X}^{(0)})}{\partial x_n} \right),$$

будують лінійну функцію

$$F_k(\mathbf{X}^{(0)}) = \frac{\partial f(\mathbf{X}^{(0)})}{\partial x_1} x_1 + \frac{\partial f(\mathbf{X}^{(0)})}{\partial x_2} x_2 + \ldots + \frac{\partial f(\mathbf{X}^{(0)})}{\partial x_n} x_n. \tag{1.25}$$

Знаходять максимум функції (1.25) при обмеженнях (1.23) і (1.24). Нехай розв'язок даної задачі визначається точкою $\mathbf{Z}^{(0)}$. За нове допустиме рішення вихідної задачі приймають координати точки $\mathbf{X}^{(k+1)}$, які знаходять за формулами

$$\mathbf{X}^{(k+1)} = \mathbf{X}^{(k)} + \lambda_k \left(\mathbf{Z}^{(k)} - \mathbf{X}^{(k)} \right), \tag{1.26}$$

де λ_k – деяке число, яке називається кроком обчислень ($0 < \lambda_k < 1$). За λ_k приймають найменший корінь рівняння $\frac{df\left(X^{(k+1)}\right)}{d\lambda_k} = 0$ або вибирають довільно, якщо він не належить інтервалу $(0;1)$.

Приклад. Методом Франка-Вульфа знайти максимальне значення функції

$$f(X) = 2x_1 + 4x_2 - x_1^2 - 2x_2^2 \tag{1.27}$$

за умов

$$\begin{cases} x_1 + 2x_2 \le 8, \\ 2x_1 - x_2 \le 12, \end{cases} \tag{1.28}$$

$$x_1 \ge 0, x_2 \ge 0. \tag{1.29}$$

Функція $f(X)$ є ввігнутою, так як представляє собою суму лінійної функції $f_1(X) = 2x_1 + 4x_2$ та квадратичної форми $f_2(X) = -x_1^2 - 2x_2^2$, яка є від'ємно-визначеною і, отже, також є ввігнутою. Система обмежень задачі включає тільки лінійні нерівності. Отже, ми маємо задачу квадратичного програмування.

Знайдемо градієнт функції

$$\nabla f = \left(\frac{\partial f}{\partial x_1}; \frac{\partial f}{\partial x_2} \right) = (2 - 2x_1; 4 - 4x_2),$$

у якості вихідного допустимого рішення задачі візьмемо точку $X^{(0)} = (0;0)$, а у якості критерію оцінки якості розв'язку, який отримуємо, – нерівність $\left| f\left(X^{(k+1)}\right) - f\left(X^{(k)}\right) \right| < \varepsilon$, де $\varepsilon = 0{,}01$.

І ітерація. У точці $X^{(0)}$ градієнт $\nabla f\left(X^{(0)}\right) = (2;4)$. Знаходимо максимум функції

$$F_0(X) = 2x_1 + 4x_2 \tag{1.30}$$

за умовою (1.28) та (1.29):

$$\begin{cases} x_1 + 2x_2 \le 8, \\ 2x_1 - x_2 \le 12, \end{cases} \tag{1.31}$$

$$x_1 \ge 0, x_2 \ge 0. \tag{1.32}$$

Задачу (1.30)-(1.32) розв'яжемо графічно. Оптимальний розв'язок – $Z(0;4)$. (рис. 1.17)

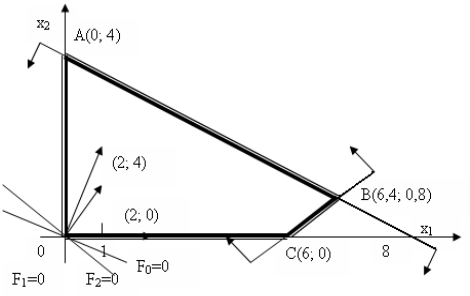


Рис. 1.17

Знайдемо нове допустиме рішення вихідної задачі за формулою (1.26):

$$X = X + \lambda_0 (Z - X), \text{ де } 0 < \lambda_0 < 1.$$

Підставляючи значення X та Z, отримаємо

$$\begin{cases} x_1 = 0 + \lambda_0 (0 - 0) = 0, \\ x_2 = 0 + \lambda_0 (4 - 0) = 4\lambda_0. \end{cases} \quad (1.33)$$

Визначимо число λ_0. Підставимо значення x_1, x_2 з (1.33) у рівність (1.27), отримаємо:

$$f(\lambda_0) = 2 \cdot 0 + 4 \cdot (4\lambda_0) - 0^2 - 2 \cdot (4\lambda_0)^2 = 16\lambda_0 - 32\lambda_0^2.$$

Знайдемо похідну цієї функції по λ_0 і прирівняємо її до нуля:

$$f'(\lambda_0) = 16 - 64\lambda_0 = 0.$$

Розв'язавши це рівняння, отримаємо $\lambda_0 = 0{,}25$. Оскільки знайдене значення λ_0 знаходиться між 0 та 1, приймаємо його за величину кроку. Таким чином,

$$X = (0;1),$$

$$f(X) = 2 \cdot 0 + 4 \cdot 1 - 0^2 - 2 \cdot 1^2 = 2;$$

$$|f(X) - f(X)| = 2 - 0 = 2 > \varepsilon = 0{,}01.$$

ІІ ітерація. Градієнт цільової функції вихідної задачі у точці X є $\nabla f(X) = (2;0)$. Знаходимо максимум функції $F_1(x) = 2x_1$ за

умов (1.31) та (1.32). Розв'яжемо цю задачу графічно. Розв'язком є $Z = (6,4;0,8)$ (рис. 1.17).

Визначимо тепер $X = X + \lambda_1 (Z - X)$. Підставляючи значення X та Z, отримаємо

$$\begin{cases} x_1 = 0 + \lambda_1 (6,4 - 0) = 6,4\lambda_1, \\ x_2 = 1 + \lambda_1 (0,8 - 1) = 1 - 0,2\lambda_1. \end{cases} \quad (1.34)$$

Для знаходження λ_1 підставимо значення x_1, x_2 з (1.34) і рівність (1.27), отримаємо:

$$f(\lambda_1) = 2 \cdot (6,4\lambda_1) + 4 \cdot (1 - 0,2\lambda_1) - (6,4\lambda_1)^2 -$$
$$- 2 \cdot (1 - 0,2\lambda_1)^2 = 2 + 12,8\lambda_1 - 41,76\lambda_1^2.$$

Звідки $f'(\lambda_1) = 12,8 - 83,52\lambda_1$. Прирівнюючи $f'(\lambda_1)$ до нуля та розв'язуючи це рівняння, отримаємо $\lambda_1 \approx 0,15$ $(0 < \lambda_1 < 1)$. Таким чином,

$$\begin{cases} x_1 = 6,4 \cdot 0,15 = 0,96, \\ x_2 = 1 + 0,2 \cdot 0,15 = 0,97, \end{cases}$$

тобто

$$X = (0,96;0,97),$$
$$f(X) = 2 \cdot 0,96 + 4 \cdot 0,97 - 0,96^2 - 2 \cdot 0,97^2 = 2,9966 ;$$
$$\left| f(X) - f(X) \right| = 2,9966 - 2,0000 = 0,9966 > \varepsilon = 0,01.$$

III ітерація. Градієнт функції f у точці X є $\nabla f(X) = (0,08;0,12)$. Знаходимо максимум функції $F_2(\lambda) = 0,08x_1 + 0,12x_2$ за умов (1.31) та (1.32). Розв'яжемо цю задачу графічно, отримаємо $Z = (6,4;0,8)$ (рис. 1.17).

Визначимо тепер $X = X + \lambda_2 (Z - X)$. Підставляючи значення X та Z, маємо

$$\begin{cases} x_1 = 0,96 + \lambda_2 (6,40 - 0,96) = 0,96 + 5,44\lambda_2, \\ x_2 = 0,97 + \lambda_2 (0,80 - 0,97) = 0,97 - 0,17\lambda_2. \end{cases} \quad (1.35)$$

Підставивши значення $x_1^{(1)}$, $x_2^{(1)}$ з (1.80) і рівність (1.27), отримаємо:

$$f(\lambda_2) = 2\cdot(0{,}96+5{,}44\lambda_2)+4\cdot(0{,}97-0{,}17\lambda_2)-(0{,}96+5{,}44\lambda_2)^2 -$$
$$-2\cdot(0{,}97-0{,}17\lambda_2)^2 = 2{,}9966+0{,}4148\lambda_2-29{,}6514\lambda_2^2;$$
$$f'(\lambda_2) = 0{,}4148-59{,}3028\lambda_2.$$

Розв'язавши рівняння $f'(\lambda_2)=0$, знайдемо $\lambda_2 \approx 0{,}007$ ($0<\lambda_2<1$). Отже,

$$\begin{cases} x_1^{(1)} = 0{,}96+5{,}44\cdot 0{,}007 = 0{,}99808, \\ x_2^{(1)} = 0{,}97-0{,}17\cdot 0{,}007 = 0{,}96881, \end{cases}$$

тобто

$$X^{(1)} = (0{,}99808;\ 0{,}96881),$$
$$f(X^{(1)}) = 2\cdot 0{,}99808 + 4\cdot 0{,}96881 - 0{,}99808^2 - 2\cdot 0{,}96881^2 = 2{,}99805$$
$$\left| f(X^{(1)}) - f(X^{(0)}) \right| = 2{,}99805 - 2{,}99660 = 0{,}00145 < \varepsilon = 0{,}01.$$

Таким чином, $Z^{(1)} = (0{,}99808;\ 0{,}96881)$ є задовільним розв'язком вихідної задачі (розв'язок задачі $X^* = (1;1)$ і $f_{\max} = 3$).

Метод штрафних функцій. Розглянемо задачу нелінійного програмування, де $g_i(x_1, x_2, \ldots, x_n)$ – опуклі функції.

Замість того, щоб розв'язувати цю задачу, знаходять максимум функції

$$F(x_1, x_2, \ldots, x_n) = f(x_1, x_2, \ldots, x_n) + H(x_1, x_2, \ldots, x_n),$$

де $H(x_1, x_2, \ldots, x_n)$ визначається системою обмежень та називається штрафною функцією. Останню можна побудувати різними способами. Найбільш часто вона має вигляд

$$H(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{m} a_i(x_1, x_2, \ldots, x_n)\, g_i(x_1, x_2, \ldots, x_n),$$

де

$$a_i(x_1, x_2, \ldots, x_n) = \begin{cases} 0, & \text{якщо } b_i - g_i(x_1, x_2, \ldots, x_n) \geq 0, \\ \alpha_i, & \text{якщо } b_i - g_i(x_1, x_2, \ldots, x_n) < 0, \end{cases}$$

а $a_i > 0$ – деякі постійні числа, які представляють собою вігові коефіцієнти.

Використовуючи штрафну функцію, послідовно переходять від однієї точки до іншої до тих пір, поки не отримають задовільний розв'язок. Координати наступної точки знаходять за формулою

$$x_j^{(k+1)} = \max\left\{0; x_j^{(k)} + \lambda\left[\frac{\partial f(X^{(k)})}{\partial x_j} + \sum_{i=1}^{m} \alpha_i \frac{\partial g_i(X^{(k)})}{\partial x_j}\right]\right\},$$

де λ – крок обчислень ($0 < \lambda < 1$).

Ітераційний процес зазвичай починають при порівняно малих значеннях a_i і, продовжуючи його, ці значення поступово збільшують.

Метод Ерроу-Гурвіца. При знаходженні розв'язку задачі нелінійного програмування методом штрафних функцій значення a_i вибирають довільно, що призводить до значних коливань віддаленості точок, які визначаються, від області допустимих рішень. Цей недолік усувається при розв'язку задачі методом Ерроу-Гурвіца, згідно якому на черговому кроці числа $a_i^{(k)}$ знаходять за формулою

$$\alpha_i^{(k)} = \max\{0; \alpha_i^{(k+1)} - \lambda g_i(X^{(k)})\}, \; i = \overline{1,m}.$$

У якості початкових значень $a_i^{(0)}$ беруть довільні невід'ємні числа.

1.3.1. Задачі опуклого програмування

Важливий клас задач математичного програмування складають задачі, моделі яких містять опуклі (ввігнуті) функції.

Опукле програмування представляє собою сукупність методів рішення нелінійних екстремальних задач з опуклими функціями – розділ нелінійного програмування [9].

В теорії опуклого програмування у якості основної зазвичай розглядається задача мінімізації опуклої функції n змінних $z = f(\mathbf{x})$ при обмеженнях $\varphi_i(\mathbf{x}) \le 0 \quad (i = \overline{1,m})$, $\mathbf{x} \ge 0$, де функції $\varphi_i(\mathbf{x})$ припускаються опуклими. Опуклість множини допустимих рішень задачі випливає з теореми.

Опуклість та ввігнутість функцій визначається тільки відносно опуклих множин

Теорема. *Якщо* $\varphi(\mathbf{x})$ *– опукла функція при усіх* $\mathbf{x} \geq 0$, *то буде опуклою і множина рішень системи* $\varphi(\mathbf{x}) \leq b$, $\mathbf{x} \geq 0$.

Якщо $f(\mathbf{x})$ та $\varphi_i(\mathbf{x})$ є ввігнутими функціями, то маємо задачу максимізації $f(\mathbf{x})$ при обмеженнях $\varphi_i(\mathbf{x}) \geq 0 \quad (i = \overline{1,m})$, $\mathbf{x} \geq 0$.

Основними властивостями опуклих та ввігнутих функцій є:

1. Множина точок мінімуму опуклої функції, заданої на опуклій множині, – опукла.

2. Нехай $f(x)$ – опукла функція, задана на замкненій опуклій множині $X \subset E^n$. Тоді локальний мінімум $f(x)$ на X є і глобальним.

3. Якщо глобальний мінімум досягається у двох різних точках, то він досягається і у будь-якій точці відрізку, який з'єднує дані точки.

4. Якщо $f(x)$ – строго опукла функція, то її глобальний мінімум на опуклій множині X досягається у єдиній точці.

5. Нехай функція $f(x)$ – опукла функція, задана на опуклій множині X, і, крім того, вона неперервна разом із своїми частковими похідними першого порядку у всіх внутрішніх точках X. Нехай $\mathbf{X}^0$ – точка, у якій $\nabla f(\mathbf{X}^0) = 0$. Тоді у точці $\mathbf{X}^0$ досягається локальний мінімум, який співпадає з глобальним мінімумом.

6. Множина точок глобальних (отже, і локальних) мінімумів опуклої функції $f(x)$, яка задана на обмеженій замкненій опуклій множині X, включає хоча б одну крайню точку; якщо множина локальних мінімумів включає у себе хоча б одну внутрішню точку множини X, то $f(x)$ є функцією-константою.

Для вирішення задачі опуклого програмування використовуються метод множників Лагранжа, градієнтні методи.

1.3.2. Задача квадратичного програмування

Квадратичне програмування. Одним з окремих видів задачі опуклого програмування є задача, у якій цільова функція містить квадратичний доданок, а обмеження носять лінійний характер. У цьому випадку говорять, що задача відноситься до *квадратичного програмування* [9]. У задачі квадратичного програмування обмеження

$$g_i(x_1, x_2, \ldots, x_n) = b_i - \sum_{j=1}^{n} a_{ij} x_j \ , \ i = \overline{1,m} \ ,$$

є лінійними, а функція $f(x_1, x_2, \ldots, x_n)$ представляє собою суму лінійної та квадратичної функції (квадратичної форми)

$$f(x_1, x_2, \ldots, x_n) = c_1 x_1 + c_2 x_2 + \ldots + c_n x_n + d_{11} x_1^2 + d_{22} x_2^2 + \ldots +$$
$$+ d_{nn} x_n^2 + 2d_{12} x_1 x_2 + 2d_{13} x_1 x_3 + \ldots + 2d_{n-1,n} x_{n-1} x_n .$$

Як у загальному випадку вирішення задач нелінійного програмування, для визначення глобального екстремуму задачі квадратичного програмування не існує ефективного обчислювального методу, якщо не відомо, що будь-який екстремум є одночасно і глобальним. Так як у задачі квадратичного програмування множина допустимих рішень опукла, то, якщо цільова функція ввігнута, будь-який локальний максимум є глобальним; якщо ж цільова функція – опукла, то будь-який локальний мінімум також і глобальний.

Функція f представляє собою суму лінійної функції (яка є і опуклою, і ввігнутою) і квадратичної форми. Якщо остання є ввігнутою (опуклою), то задачі відшукування максимуму (мінімуму) цільової функції можуть бути успішно вирішені. Питання у тому, чи буде квадратична форма ввігнутою або опуклою, залежить від того, чи є вона від'ємною-визначеною, від'ємною-напіввизначеною, додатною-визначеною, додатною-напіввизначеною або взагалі невизначеною.

1.4. Цілочисельні задачі лінійного програмування

Цілочисельне програмування – різновид лінійного програмування, в якому отримані значення повинні бути цілими числами.

При розгляді цілого ряду економічних задач необхідно враховувати вимогу цілочисельності використаних змінних. Такі задачі називаються задачами цілочисельного програмування [6]. Під задачею цілочисельного програмування розуміється задача, в якій всі або деякі змінні повинні приймати цілі значення. У тому випадку, коли обмеження і цільова функція задачі є лінійною залежністю, задачу називають цілочисельною задачею лінійного програмування. В іншому випадку, коли хоча б одна залежність буде нелінійною, це буде цілочисельною задачею нелінійного програмування.

Особливий інтерес до задач цілочисельного програмування викликаний тим, що в багатьох практичних задачах необхідно знаходити цілочисельне рішення, зважаючи на дискретність ряду значень шуканих змінних.

Цілочисельне програмування виникло в 50-60-ті роки минулого століття з потреб практики – головним чином в роботах американських математиків Дж.Данцига і Р.Гомори. Спочатку цілочисельне програмування розвивалося незалежно від геометрії чисел на основі теорії і методів математичної оптимізації, насамперед лінійного програмування. Проте, в останні час дослідження в цьому напрямі все частіше проводяться засобами математики цілих чисел. Задачі такого типу дуже актуальні, оскільки до їх вирішення зводиться аналіз різноманітних ситуацій, що виникають в економіці, техніці, військовій справі та інших областях. З появою ЕОМ, зростанням їх продуктивності підвищився інтерес до задач такого типу і до математики в цілому.

Задачу цілочисельного програмування записують так:

$$Z = \sum_{j=1}^{n} c_j x_j \to \max(\min),$$

за умов

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

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

$$x_j - \text{цілі}, \quad (j = \overline{1,n}).$$

Для знаходження оптимального розв'язку цілочисельних задач застосовують спеціальні методи. Найпростішим методом розв'язання цілочисельної задачі є знаходження її оптимального розв'язку як задачі, що має тільки неперервні змінні, з подальшим округленням останніх.

Для знаходження оптимальних планів задач цілочисельного програмування застосовують дві групи методів:

— методи відтинання;

— комбінаторні методи.

Основою методів відтинання є ідея поступового «звуження» області допустимих розв'язків розглядуваної задачі. Пошук цілочисельного оптимуму починається з розв'язування задачі з так званими послабленими обмеженнями, тобто без урахування вимог цілочисельності змінних. Далі введенням у модель спеціальних додаткових обмежень, що враховують цілочисельність змінних, многокутник допустимих розв'язків послабленої задачі поступово зменшуємо доти, доки змінні оптимального розв'язку не набудуть цілочисельних значень. До цієї групи належать:

- методи розв'язання повністю цілочисельних задач (дробовий алгоритм Гомори);
- методи розв'язання частково цілочисельних задач (другий алгоритм Гомори, або змішаний алгоритм цілочисельного програмування).

Комбінаторні методи цілочисельної оптимізації базуються на повному переборі всіх допустимих цілочисельних розв'язків, тобто вони реалізують процедуру цілеспрямованого перебору, під час якої розглядається лише частина розв'язків (досить невелика), а решта враховують одним зі спеціальних методів. Найпоширенішим у цій групі методів є метод гілок і меж.

Алгоритм розв’язування задач цілочисельного програмування наступний:

1. Розв'язують задачу лінійного програмування без обмежень на цілочисельність, наприклад, симплекс-методом.

2. Якщо оптимальне рішення задачи лінійного програмування нецілочисельне, то проводять «велику ітерацію». Будують лінійне обмеження, якому задовольняє будь-яке цілочисельне вирішення задачі і не задовольняє отримане оптимальне нецілочисельне значення. Геометрично це означає –

провести перетин (гіперплощина), який відсікав би нецілочисельну вершину, не торкаючись решти цілочисельних точок. Такий перетин називають правильним. Правильний перетин повинен задовольняти наступним умовам:

1) умова відсікання: оптимальне вирішення задачі лінійного програмування не задовольняє умові відсікання;

2) умова правильності: всі цілочисельні вирішення задачі задовольняють умові відсікання.

Оскільки для початкової задачі додаткове обмеження (відсікання) даватиме неприпустиме базисне рішення, необхідно провести «малі ітерації» для отримання оптимального рішення. Якщо отримане оптимальне рішення нецілочисельне, то проводять наступне відсікання. В іншому випадку пошук завершений.

Р. Гоморі запропонував ідею формування додаткових обмежень, яка приводить до вирішення задач цілочисельного програмування за кінцеве число кроків.

Розв'язки задач цілочисельного лінійного програмування можуть бути відсутні, якщо:

1) цільова функція є необмеженою знизу, тобто як початкова нецілочисельна, так і цілочисельна задачі лінійного програмування не мають рішення;

2) початкова задача лінійного програмування має розв'язок, а цілочисельна не має.

Метод гілок і меж використовують як до повністю цілочисельних задач, так і до частково цілочисельних задач. Алгоритм методу гілок і меж є ітеративним.

1.5. Цільова функція

Цільова функція (функція цілі, показник ефективності, критерій оптимальності, функціонал задачі ті ін.) дозволяє вибрати найкращий варіант із множини можливих [9]. Найкращий варіант доставляє цільовий функції екстремальне значення. Цільову функцію позначають буквою Z ($Z = z(\mathbf{x})$). Це може бути прибуток, об'єм випуску або реалізації, витрати виробництва, рівень обслуговування або дефіцитності, число комплектів, відходи і т.д.

В залежності від особливостей цільової функції $z(\mathbf{x})$ та функцій, які задають обмеження, задачі математичного програмування діляться на ряд типів.

Якщо цільова функція $Z = z(\mathbf{x})$ та функції, які входять у систему обмежень, лінійні (першої степені) відносно невідомих, які входять у задачу, то такий розділ математичного програмування називається лінійним програмуванням.

Якщо в задачі математичного програмування цільова функція $z(\mathbf{x})$ та (або) хоча б одна з функцій системи обмежень нелінійна, то такий розділ називається нелінійним програмуванням.

Якщо параметри цільової функції та (або) системи обмежень змінюються у часі або цільова функція має адитивний чи мультиплікативний вид, або сам процес виробки розв'язку має багатокроковий характер, то такі задачі розв'язуються методами динамічного програмування.

Якщо параметри, які входять у функції цілі, або обмеження задачі є випадковими, недостовірними величинами або якщо доводиться приймати рішення в умовах ризику, неповної або недостовірної інформації, то говорять про проблему стохастичної оптимізації, а відповідний розділ називається стохастичним програмуванням.

Задачі математичного програмування з однією цільовою функцією розв'язуються методами скалярної оптимізації. Однак реальні ситуації настільки складні, що нерідко доводиться одночасно враховувати декілька цільових функцій, які повинні приймати екстремальні значення. Задачі. де знаходять розв'язок за декількома цільовими функціями, відносяться до векторної оптимізації – це так звані задачі багатокритеріального підходу.

1.6. Задача комівояжера

Задача комівояжера – одна із найвідоміших задач комбінаторної оптимізації, яка полягає у відшукуванні найбільш вигідного маршруту, який проходить через вказані міста хоча б по одному разу з наступним повертанням у вихідне місто [7, 10, 12]. В умовах задачі вказуються критерій вигідності маршруту (найкоротший, найдешевший, сукупний критерій і т.п.) і відповідні матриці відстаней, вартості і т.п. Як правило, вказується, що

маршрут повинен проходити через кожне місто тільки один раз – к такому випадку вибір здійснюється серед гамільтонових циклів.

Існує декілька окремих випадків загальної постановки задачі, зокрема геометрична задача комівояжера (також називається планарною або евклідовою, коли матриця відстаней відображає відстані між точками на площині), трикутна задача комівояжера (коли на матриці вартостей виконується нерівність трикутника), симетрична та асиметрична задачі комівояжера. Також існує узагальнення задачі, так звана узагальнене задача комівояжера.

Методи розв'язування:

- повний перебір;
- випадковий перебір;
- жадібні алгоритми (метод найближчого сусіда, метод включення найближчого міста, метод найдешевшого включення);
- метод мінімального остівного дерева;
- метод імітації віджигу.

Усі ефективні (які скорочують повний перебір) методи розв'язання задачі комівояжеру – методи евристичні. У більшості евристичних методів знаходиться не самий ефективний маршрут, а наближений розв'язок.

На практиці застосовуються різні модифікації більш ефективних методів: метод гілок та границь та метод генетичних алгоритмів, а також алгоритм мурашкової колонії. Розглянемо декілька методів розв'язку задачі комівояжера.

Метод еластичної сітки. У 1987 році його використали Дурбін та Уіллшоу, які вказали аналогію з механізмами встановлення упорядкованих нейронних зв'язків.

Кожний з маршрутів комівояжера розглядався як відображення кола на площині (у кожне місто на площині відображається деяка точка цього кола). Сусідні точки на колі повинні відображатися у точки, по можливості найближчі і на площині.

Алгоритм. Починається із встановлення на площину невеликого кола. Воно нерівномірно розширюється, стає кільцем, яке проходить практично близько усіх міст та встановлює таким чином шуканий маршрут. На кожну точку кільця, яка рухається, діє дві складові: переміщення точки у сторону найближчого міста та

Задача Гамільтона. Видатний англійський математик та механік Гамільтон винайшов гру, яка отримала у свій час широке розповсюдження. Гра представляє собою додекаедр (правильний багатокутник з 12 п'ятикутними гранями та 20 вершинами). Вважається, що цей додекаедр приблизно зображає земну кулю. Нехай кожна з вершин зображає «столицю» однієї з держав. Необхідно, рухаючись по ребрам додекаедра, обійти усі «столиці», заходячи у кожну з них один і тільки один раз, і повернутися у вихідний пункт (рис. 1.18). Кожний такий шлях, якщо від існує, називається гамільтоновим контуром, або гамільтоновим циклом.

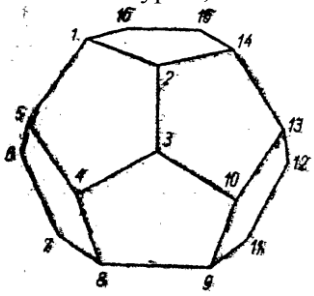


Рис. 1.18. Гамільтоновий контур

Задача про обхід шахової дошки. Майже аналогічно виглядає задача про обхід конем шахової дошки, розглянута ще Ейлером.

Перенумеруємо усі клітинки шахової дошки і будемо вважати, що «відстань» між двома клітинками дорівнює одиниці, кінь може за один хід перескочити з однієї з них на іншу, і що ця «відстань» дорівнює нескінченності у протилежному випадку.

Можна вказати велику множину задач, пов'язаних з об'їздом ряду пунктів з якої-небудь метою та поверненням у вихідну точку, наприклад, розвезення пошти, продуктів споживання і т.п. Аналогічний характер носять задачі з'єднання цих пунктів кільцевими лініями електропередач, газопостачання і т.п.

Евристичні методи та методи Монте-Карло. Евристичними називаються алгоритми, побудовані на догадці і які не претендують на знаходження точного розв'язку. Застосування евристичних алгоритмів у якій-небудь практичній задачі зазвичай приводить до рекомендаціям, які набагато кращі довільного, взятого навгад розв'язку, і які зазвичай близькі до найкращого варіанту, але вони

ніколи не дають повної впевненості у тому, що найкращого розв'язку досягнуто.

Найпростіший з евристичних алгоритмів може виглядати, наприклад, так: вибираємо довільний гамільтонів контур та міняємо у ньому місцями сусідні дві вершини. Якщо зменшення шляху не відбулося, то утримують у пам'яті попередній шлях та переходять до наступної пари точок. Якщо ж відбулося покращення маршруту, то запам'ятовують новий контур та поступають з ним аналогічним чином.

Методи Монте-Карло стають особливо ефективними, якщо їх доповнити евристичними методами покращення маршруту. У цьому випадку час на проведення одного статистичного експерименту декілька збільшується, але проте збільшується і ефективність експерименту.

Метод Монте-Карло у поєднанні з евристичними методами є одним з самих потужних засобів розв'язку задачі великого розміру; при цьому не пред'являються скільки-небудь жорсткі вимого до пам'яті та швидкодії машини. Необхідно врахувати, що проблема запам'ятовування великих масивів проміжної інформації при розв'язку задачі про комівояжера іншими методами є однією з самих гострих проблем.

Недоліком цього методу є те, що при обмеженому числі експериментів (а воно завжди є обмеженим) розв'язок носить наближений, а не точний характер, причому не завжди вдається оцінити характер похибки. Хоча з практичної точки зору це несуттєвий недолік, він все ж змушує шукати інші, детерміновані алгоритми пошуку оптимального маршруту, які необхідні хоча б для того, щоб оцінити якість розв'язків задачі, отриманих методами Монте-Карло.

Розв'язок задачі про комівояжера методами динамічного програмування. Теорія динамічного програмування – це теорія прийняття рішення у багатошарових процесах. Для того щоб застосувати методику динамічного програмування, необхідно перш за все побачити у задачі, яка розв'язується, кроки або етапи, на які можна розбити процес прийняття рішення. Потім перераховують можливі розв'язки на кожному етапі та вказують спосіб вибору найкращого розв'язку. У більшості випадків це потребує добре розвинутої інтуїції та винахідництва. Справа у тому, що ці етапи

повинні бути намічені так, щоб функцію виграшу за останні k етапів можна було б представити у вигляді суми двох величин, з яких одна виражає безпосередній виграш на поточному етапі прийняття рішення, а інша представляє виграш за $(k-1)$ етапів, які залишилися.

У випадку задачі про комівояжера ця ідея представлення задачі у вигляді багатокрокового процесу реалізується наступним чином. Відмітимо серед n точок, які необхідно обійти, довільну точку, яку будемо вважати точкою старту та фінішу. Першим етапом прийняття рішення будемо вважати вибір першої точки, в яку треба йти з місця старту, другим етапом – вибір наступної точки і т.д.

Основні труднощі при розв'язку задачі про комівояжера методами динамічного програмування – це проблема запам'ятовування проміжних результатів. Не дивлячись на ці труднощі, динамічне програмування є одним із самих перспективних методів розв'язку задачі про комівояжера та побідних з нею складних комбінаторних задач.

Метод гілок та границь. Метод гілок та границь у теперішній час є, мабуть, найбільш ефективним способом отримання точного розв'язку задачі про комівояжера, який добре пристосований як для ручної, так і для машинної реалізації.

Основна ідея методу доволі проста. Спочатку будується деяка оцінка знизу довжини маршруту для множини усіх гамільтонових контурів. Після цього множина усіх гамільтонових маршрутів розбивається на дві підмножини. Перша підмножина складається із гамільтонових контурів, які включають деяку дугу, а інша підмножина складається з контурів, які не включають цю дугу. Для кожної з підмножин за тим же правилом, що і для початкової множини гамільтонових маршрутів, визначається нижня границя. Кожна нова нижня границя виявляється не менше нижньої границі, яка визначена для усієї підмножини. Порівнюючи нижні границі, можна відокремити підмножину гамільтонових шляхів, всередині якої з більшою ймовірністю міститься оптимальний маршрут. Ця підмножина за аналогічним правилом розбивається ще на дві, і знову знаходяться змінені нижні границі, і так поступають до тих пір, поки не залишиться єдиний цикл.

Процес розбиття підмножин супроводжується побудовою деякого дерева.

Отримавши деякий гамільтонів цикл, проглядають обірвані гілки дерева і, якщо нижні границі підмножин, які відповідають обірваним гілкам, виявляться меншими, ніж довжина знайденого циклу, то ці гілки розвивають за тим же правилом, поки не отримають маршруту з меншою довжиною або не впевняться, що серед цих підмножин не може міститися подібного маршруту . Ті ж гілки, для яких нижня оцінка виявиться більше довжини отриманого маршруту, виключають із розгляду.

1.7. Фізико-геометричний смисл задач програмування

Геометрична інтерпретація задач програмування така. У тримірному просторі є якийсь замкнений або розімкнутий об'єм, всередині якого можуть знаходитися величини у декартовій системі координат x, y, z і необхідно знайти одну або декілька точок всередині допустимо області, яка обертає у максимум функцію цих координат.

Нехай задана задача лінійного програмування з двома змінними [14]:

$$a_{11}x_1 + a_{12}x_2 \le b_1;$$

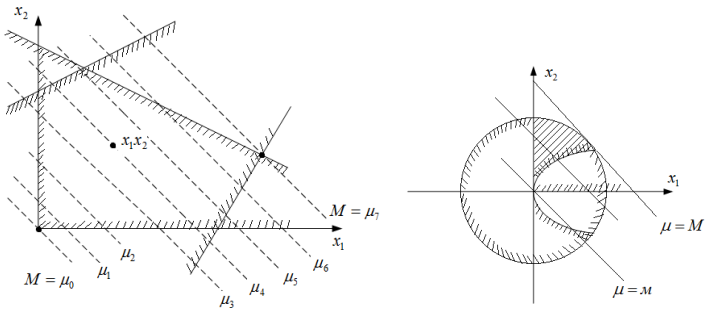
$$a_{21}x_1 + a_{22}x_2 \le b_2;$$

$$a_{31}x_1 + a_{32}x_2 \le b_3;$$

$$x_1 \ge 0,\ x_2 \ge 0;$$

$$\mu = c_1x_1 + c_2x_2 \to \text{макс(мін)}.$$

Візьмемо систему прямокутних декартових координат x_1, x_2 (рис.1.19, *а*). Кожне із лінійних нерівностей визначає півплощину, причому координати точок, розташованих по одну сторону півплощини, задовольняють відповідній нерівності. На рис. 1.19, *а* ці півплощини заштриховані. Сукупність усіх нерівностей визначить на площині багатогранну область, всередині якої усі точки мають координати, які задовольняють усім умовам задачі програмування, які обмежують, тобто які є допустимим розв'язком задачі.



а *б*

Рис. 1.19. Геометрична інтерпретація задач програмування

Якщо на площину нанести прямі лінії, які відповідають постійним значенням цільової функції, отримаємо родину паралельних прямих, які є геометричним місцем точок, для яких цільова функція приймає незмінне значення. Задача максимізації (мінімізації) цільової функції полягає. Таким чином, у знаходженні крайнього положення прямої цільової функції. Такими крайніми положеннями на рис. 1.19, *а* є M та $м$.

Геометрична інтерпретація задачі нелінійного програмування така:

$$x_1^2 + x_2^2 \le c;$$

$$x_1^2 - x_2^2 \le 0;$$

$$x_1 x_2 \ge 0;$$

$$\mu = c_1 x_1 + c_2 x_2 \to \text{макс}\left(\text{мін}\right).$$

Допустима область на площині x_1, x_2 має вигляд, показаний на рис. 1.19, *б*. Оптимальні рішення задачі відповідають положенням прямої цільової функції M та $м$.

Фізична інтерпретація: знайти, наприклад, склад продуктів, які формують добове меню якогось організму або якоїсь популяції, який би дозволяв забезпечити меню вартості при виконанні обмежень: якихось продуктів повинно бути не менше, наприклад, жирів не менше однієї кількості, вуглеводів не менше другої кількості і т.д.

Можна навести дуже багато прикладів із різних областей техніки та людського суспільства, які вкладаються у задачі

оптимального планування. Можна вказати ще такі приклади: задача знаходження оптимальної послідовності робіт по створенню нового виробу, яка мінімізує час закінчення розробки; задача визначення оптимального розподілу обмежених ресурсів військового та цивільного характеру, який мінімізує ефект їх застосування та інші.

Питання для самоконтролю

1. Класифікація задач математичного програмування.
2. Електронна схема для розв'язку алгебраїчного рівняння.
3. Схема для моделювання обмеження.
4. Електронна схема квазіоберненого перетворювача.
5. Методи вирішення задач нелінійного програмування.
6. Роль цільової функції.
7. Загальне визначення задачі комівояжера.
8. Фізичний смисл задач програмування.

2.1. Рішення задач оптимізації для без інерційних об'єктів і процесів

Задачею оптимізації у математиці називається задача про знаходження екстремуму (мінімуму або максимуму) дійсної функції у деякій області. Як правило, розглядаються області, які належать R^n і задані набором рівностей та нерівностей.

У процесі проектування ставиться зазвичай задача визначення найкращих, у деякому розумінні, структури або значення параметрів об'єктів. Така задача називається оптимізаційною [1, 4, 5, 8, 13]. Якщо оптимізація пов'язана із розрахунком оптимальних значень параметрів при заданій структурі об'єкта, то вона називається параметричною. Задача вибору оптимальної структури є структурною оптимізацією.

Розв'язати оптимізаційну задачу – означає знайти її оптимальне розв'язання або встановити, що розв'язання немає. Методи розв'язання оптимізаційних задач називаються методами математичного програмування. Оптимізаційні моделі бувають двох типів: задачі мінімізації та задачі максимізації.

Стандартна математична задача оптимізації формулюється таким чином. Серед елементів χ, які утворюють множини X, знайти такий елемент χ^*, який доставляє мінімальне значення $f(\chi^*)$ заданої функції $f(\chi)$. Для того, щоб коректно поставити задачу оптимізації необхідно задати:

1. Допустиму множину – множину
$X = \{g_i(\vec{x}) \le 0, i = 1,\ldots,m\} \subseteq R^n$;
2. Цільову функцію – відображення $f : X \to R$;
3. Критерій пошуку (max або min).

Тоді розв'язати задачу $f(\vec{x}) \to \min\limits_{\vec{x} \in X}$ означає одне із:

1. Показати, що $X = \emptyset$.
2. Показати, що цільова функція $f(\vec{x})$ не обмежена знизу.
3. Знайти $\vec{x}^* \in X : f(\vec{x}^*) = \min\limits_{\vec{x} \in X} f(\vec{x})$.

4. Якщо $\exists \vec{x}^*$, то знайти $\inf_{\bar{x}\in X} f(\bar{x})$.

Якщо функція, яка мінімізується, не є опуклою, то часто обмежуються пошуком локальних мінімумів та максимумів: точок x_0 таких, що всюди у деякому їх околі $f(\bar{x}) \geq f(\bar{x}_0)$ для мінімуму та $f(\bar{x}) \leq f(\bar{x}_0)$ для максимуму.

Якщо допустима множина $X = R^n$, то така задача називається задачею безумовної оптимізації, у протилежному випадку – задачею оптимізації з обмеженнями.

Методи оптимізації класифікуються у відповідності із задачами оптимізації:

- Локальні методи: сходяться до якого-небудь локального екстремуму цільової функції. У цьому випадку унімодальної цільової функції, цей екстремум єдиний, і буде глобальним мінімумом/максимумом.
- Глобальні метоли: мають справу з багато екстремальними цільовими функціями. При глобальному пошуку основною задачею є виявлення тенденцій глобальної поведінки цільової функції.

Існуючі у теперішній час методи пошуку можна розбити на три великі групи:

1. детерміновані;
2. випадкові (стохастичні);
3. комбіновані.

За критерієм розмірності допустимої множини, методи оптимізації ділять на методи одномірної оптимізації та методи багатомірної оптимізації.

За виглядом функції та допустимої множини, задачі оптимізації та методи їх розв'язку можна розділити на наступні класи:

- Задачі оптимізації, у яких цільова функція $f(\bar{x})$ та обмеження $g_i(\bar{x})$, $i = 1, \ldots, m$ є лінійними функціями, вирішуються так званими методами лінійного програмування.
- У протилежному випадку мають справу із задачею нелінійного програмування та застосовують відповідні методи. У свою чергу з них виділяють дві окремі задачі:

1. якщо $f(x)$ та $g_i(x)$, $i=1,\ldots,m$ – опуклі функції, то таку задачу називають задачею опуклого програмування;

2. якщо $X \subset Z$, то мають справу із задачею цілочисельного (дискретного) програмування.

За вимогами до гладкості та наявності у цільової функції часткових похідних, їх також можна розділити на:

- Прямі методи, які потребують тільки обчислень цільової функції у точках наближень;
- Методи першого порядку: потребують обчислень перших часткових похідних функцій;
- Методи другого порядку: потребують обчислень других часткових похідних, тобто гессіану цільової функції.

Окрім того, оптимізаційні методи діляться на наступні групи:

- Аналітичні методи (наприклад, метод множників Лагранжа та умови Каруша-Куна-Таккера);
- Чисельні методи;
- Графічні методи.

У залежності від природи множини X задачі математичного програмування класифікуються як:

- Задачі дискретного програмування (або комбінаторної оптимізації) – якщо X кінцева або додатна;
- Задачі цілочисельного програмування – якщо X є підмножиною множини цілих чисел;
- Задачі нелінійного програмування, якщо обмеження або цільова функція містить нелінійні функції та X є підмножиною кінцевомірного векторного простору.
- Якщо ж усі обмеження та цільова функція містять лише лінійні функції, то це – задача лінійного програмування.

Крім того, розділами математичного програмування є параметричне програмування, динамічне програмування та стохастичне програмування. Математичне програмування використовується при розв'язуванні оптимізаційних задач дослідження операцій.

Спосіб знаходження екстремуму повністю визначається класом задачі. Але перед тим, як отримати математичну модель, необхідно виконати 4 етапи моделювання:

- Визначення границі системи оптимізації;

- Вибір змінних, якими управляють;
- Визначення обмежень на змінні, якими управляють;
- Вибір числового критерію оптимізації.

2.2. Локальні і глобальні екстремуми

Терміни «максимум» та «мінімум» об'єднують одним словом «екстремум» та говорять про локальний та глобальний екстремум [8].

Екстремум – це найбільше та найменше значення функції на заданій множині.

Розрізняють:

- локальний екстремум – екстремум у деякому довільному околі даної точки;
- глобальний екстремум – екстремум у всій області значень функцій, яка розглядається.

Дійсна функція $f(x_1, x_2, \ldots, x_n)$, визначена при $x_1 = a_1$, $x_2 = a_2$, …, $x_n = a_n$, має у точці $(a_1, a_2, \ldots, a_n)$ (локальний) максимум або (локальний (мінімум) $f(a_1, a_2, \ldots, a_n)$, якщо існує таке додатне число δ, що при усіх $\Delta x_1, \Delta x_2, \ldots, \Delta x_n$, для яких виконуються нерівності $0 < \sqrt{\Delta x_1^2 + \Delta x_2^2 + \ldots + \Delta x_n^2} < \delta$ та існує значення $f(a_1 + \Delta x_1, a_2 + \Delta x_2, \ldots, a_n + \Delta x_n)$, приріст функції

$$\Delta f \equiv f(a_1 + \Delta x_1, a_2 + \Delta x_2, \ldots, a_n + \Delta x_n) - f(a_1, a_2, \ldots, a_n),$$

відповідно менший нуля або більший нуля. Локальний максимум (мінімум) називається внутрішнім максимумом (внутрішнім мінімумом) або граничним максимумом (граничним мінімумом), якщо точка $(a_1, a_2, \ldots, a_n)$ є відповідно точкою або граничною точкою області визначення функції $f(x_1, x_2, \ldots, x_n)$.

2.3. Матричні ігри як спеціалізовані задачі оптимізації

Теорія ігор займається розробкою різного роду рекомендацій по прийняттю рішень в умовах конфліктної ситуації [9]. Формалізуючи конфліктні ситуації математично, їх можна представити як гру двох, трьох та більше гравців, кожний з яких переслідує ціль максимізації свого виграшу за рахунок іншого гравця. Іноді теорію ігор визначають як розділ математики, який

займається виробленням оптимальних правил поведінки для кожної сторони, яка приймає учать у конфліктній ситуації. Сукупність правил, які однозначно визначають послідовність дій сторони у конкретній конфліктній ситуації, є *стратегія.*

Під терміном «*гра*» розуміється сукупність попередньо обговорених правил та умов, а термін «*партія*» пов'язаний з частковою можливістю реалізації цих правил. Якщо n партнерів (гравців) $P_1, P_2, \ldots, P_n$ приймають участь у даній грі, то основний зміст теорії гри полягає у вивченні наступної проблеми: як повинен вести партію j-й партнер $(j = \overline{1, n})$ для досягнення найбільш кращого для себе виходу?

У подальшому припускається, що у кінці партії кожний гравець P_j отримує суму v_j, яка називається *виграшем.* При цьому припускається, що кожний гравець керується лише метою максимізації загальної суми виграшу. Числа v_j $(j = \overline{1, n})$ можуть бути додатними, від'ємними чи рівними нулю. Якщо $v_j > 0$, то це відповідає виграшу j-го гравця, якщо $v_j < 0$, – програшу, при $v_j = 0$ – нічийний вихід.

У більшості випадків маємо ігри з нульовою сумою, тобто $v_1 + v_2 + \ldots + v_n = 0$. У цих іграх сума виграшу переходить від одного партнера до іншого, не поступаючи із зовнішніх джерел. Гра з нульовою сумою передбачає, що сума виграшем усіх гравців у кожній партії дорівнює нулю. Прикладами гри з нульовою сумою є багато економічних задач. У них загальна сума виграшу перерозподіляється між гравцями, але не змінюється. У протилежному випадку маємо гру з ненульовою сумою.

Ігри, у яких приймають участь два гравця, називаються *парними*, а ігри з більшим числом учасників – *множинними.* Прийняття гравцем того чи іншого рішення у процесі гри та його реалізація називається *ходом.* Ходи можуть бути *особисті* та *випадкові*. Якщо хід вибирається свідомо, – це особистий хід, а якщо за допомогою механізму вибору, – випадковий хід.

Шахи є грою двох партнерів з кінцевим числом особистих ходів. У подальшому ми будемо розглядати ігри двох партнерів з

нульовою сумою та кінцевим числом можливих ходів. Такі ігри математично глибоко опрацьовані та викликають найбільший інтерес, оскільки частіше використовуються у практичних додатках.

У залежності від кількості стратегій ігри діляться на *кінцеві* та *нескінченні.* Так, у кінцевій грі кожний з гравців має кінцеве число можливих стратегій. Якщо ж хоча б один з гравців має нескінченне число можливих стратегій, то гра називається нескінченною.

У залежності від взаємовідносин гравців ігри діляться на *кооперативні, коаліційні* та *безкоаліційні*. Якщо гравці не мають права вступати у згоду, то така гра відноситься до безкоаліційної, якщо ж гравці можуть вступати у згоду, створювати коаліції, – до коаліційних. Кооперативна гра – це така гра, у якій заздалегідь визначені коаліції.

У залежності від виду функції виграшів гри підрозділяють на *матричні, біматричні, неперервні, опуклі, сепарабельні* і т.д.

Розглянемо матричні ігри. В загальному випадку матрична гра задається прямокутною матрицею розмірності $m \times n$. Номер i рядочка матриці відповідає номеру стратегії A_i, яка застосовується гравцем P_1. Номер j стовпчика відповідає стратегії B_i, який застосовується гравцем P_2. Описана гра однозначно визначається матрицею

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{bmatrix} = \left\| a_{ij} \right\|_{m \times n}.$$

Кожний елемент a_{ij} матриці є дійсним числом та представляє собою суму виграшу, яка сплачується гравцем P_2 гравцю P_1, якщо P_1 обирає стратегію, яка відповідає i-му рядочку, а P_2 обирає стратегію, яка відповідає j-му стовпчику.

Матричну гру часто записують у розгорнутій формі у вигляді таблиці, яка називається *платіжною матрицею*:

	B_1	B_j	B_m
A_1	a_{11}	a_{1j}	a_{1n}
A_i	a_{i1}	a_{ij}	a_{in}
A_m	a_{m1}	a_{mj}	a_{mn}

Кожен гравець обирає для себе найбільш вигідну стратегію. При цьому перший гравець намагається обрати таку стратегію, яка надає йому максимальний виграш, тоді другий гравець обирає стратегію, яка приводить його до мінімального програшу. У зв'язку з цим вводять поняття нижньої та верхньої чистої ціни гри.

Нижньою чистою ціною гри (максиміном) називається число α, яке визначається за формулою

$$\alpha = \max_i \min_j a_{ij}. \qquad (2.1)$$

Верхньою чистою ціною гри (мінімаксом) називається число β, яке визначається за формулою

$$\beta = \min_j \max_i a_{ij}. \qquad (2.2)$$

Стратегії гравців, які відповідають максиміну (мінімаксу), називаються *максимінними (мінімаксними).*

Розрізняють стратегії *чисті* та *змішані*. Чиста стратегія A_i $(i = \overline{1,m})$ першого гравця (чиста стратегія B_j $(j = \overline{1,n})$ другого гравця) – це можливий хід першого (другого) гравця, вибраний ним з ймовірністю, яка дорівнює 1.

Якщо перший гравець має m стратегій, а другий – n стратегій, то для будь-якої пари першого та другого гравців чисті стратегії можна представити у вигляді одиничних векторів. Наприклад, для пари стратегій A_1, B_2 чисті стратегії першого та другого гравців запишуться у вигляді: $\mathbf{p}_1 = (1;0;\ldots,0)$, $\mathbf{q}_2 = (0;1,0,\ldots;0)$. Для пари стратегій A_i, B_j чисті стратегії можна записати у вигляді:

$$\mathbf{p}_i = (0;0;\ldots,\underbrace{1;}_{i\ \ е\ \ місце}0;\ldots;0),$$

$$\mathbf{q}_j = (0;0;\ldots;0;\underbrace{\quad 1;\quad}_{j\ \text{е місце}} 0;\ldots;0).$$

Теорема 1. *В матричній грі нижня чиста ціна гри не перевищує верхньої чистої ціни гри, тобто* $\alpha \le \beta$.

Якщо для чистих стратегій A_i, B_j гравців A та B відповідно має місце рівність $\alpha = \beta$, то пару чистих стратегій (A_i, B_j) називають *сідловою точкою матричної гри,* елемент a_{ij} матриці, який стоїть на перехресті i-го рядочка та j-го стовпчика – *сідловим елементом платіжної матриці,* а число $v = \alpha = \beta$ - *чистою ціною гри.*

Якщо ж матрична гра не має сідлової точки, то рішення гри утруднюється. У цих іграх $\alpha < \beta$. Застосування мінімаксних стратегій у таких іграх призводить до того, що для кожного з гравців виграш не перевищує α, а програш – не менше β. Для кожного гравця виникає питання збільшення виграшу (зменшення програшу). Рішення знаходять, застосовуючи змішані стратегії.

Змішаною стратегією першого (другого) гравців називається вектор $\mathbf{p} = (p_1;\ldots;p_m)$, де $p_i \ge 0 \quad (i = \overline{1,m})$ та $\sum_{i=1}^{m} p_i = 1$ ($\mathbf{q} = (q_1;\ldots;q_n)$, де $q_j \ge 0 \quad (j = \overline{1,n})$ та $\sum_{j=1}^{n} q_i = 1$).

Вектор $\mathbf{p}(\mathbf{q})$ означає ймовірність застосування i-ї чистої стратегії першим гравцем (j-ї чистої стратегії другим гравцем).

Оскільки гравці обирають свої чисті стратегії випадково та незалежно один від одного, гра має випадковий характер та випадковою стає величина виграшу (програшу). У такому випадку середня величина виграшу (програшу) – математичне очікування – є функцією від змішаних стратегій $\mathbf{p}$, $\mathbf{q}$:

$$f(\mathbf{p},\mathbf{q}) = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij} p_i q_j .$$

Функція $f(\mathbf{p},\mathbf{q})$ називається *платіжною функцією* гри з матрицею $\|a_{ij}\|_{m\times n}$.

Стратегії $\mathbf{p}^* = (p_1^*;\ldots;p_m^*)$, $\mathbf{q}^* = (q_1^*;\ldots;q_n^*)$ називаються *оптимальними,* якщо для довільних стратегій $\mathbf{p} = (p_1;\ldots;p_m)$, $\mathbf{q} = (q_1;\ldots;q_n)$ виконується умова

$$f(\mathbf{p},\mathbf{q}^*) \le f(\mathbf{p}^*,\mathbf{q}^*) \le f(\mathbf{p}^*,\mathbf{q}). \qquad (2.3)$$

Використання у грі оптимальних змішаних стратегій забезпечує першому гравцю виграш, не менший, ніж при використанні ним будь-якої іншої стратегії $\mathbf{p}$; другому гравцю – програш, не більший, ніж при використанні ним будь-якої іншої стратегії $\mathbf{q}$.

Сукупність оптимальних стратегій та ціни гри складає *рішення гри.*

Значення платіжної функції при оптимальних стратегіях визначає ціну гри v, тобто $f(\mathbf{p}^*,\mathbf{q}^*) = v$.

Теорема 2. *У змішаних стратегіях будь-яка кінцева матрична гра має сідлову точку.*

Теорема 3. *Для того щоб змішані стратегії* $\mathbf{p}^* = (p_1^*;\ldots;p_m^*)$ *та* $\mathbf{q}^* = (q_1^*;\ldots;q_n^*)$ *були оптимальними для гравців* A та B у грі з матрицею $\|a_{ij}\|_{m\times n}$ *та виграшем* v, *необхідно та достатньо виконання нерівностей:*

$$\sum_{i=1}^{m} a_{ij}p_i^* \ge v \qquad (j=\overline{1,n}) \qquad (2.4)$$

$$\sum_{j=1}^{n} a_{ij}q_j^* \le v \qquad (i=\overline{1,m}) \qquad (2.5)$$

Таким чином, для перевірки того, що набір $(\mathbf{p}^*,\mathbf{q}^*,v)$ є рішенням матричної гри, достатньо перевірити, чи задовольняють $\mathbf{p}^*$, $\mathbf{q}^*$ нерівностям (2.4) та (2.5) та рівнянням

$$\sum_{i=1}^{m} p_i = 1,\ \sum_{j=1}^{n} q_j = 1.$$

На основі даної теореми можна зробити висновок: *якщо гравець A застосовує оптимальну змішану стратегію $\mathbf{p}^*$, а гравець B - будь-яку чисту стратегію B_j, то виграш гравця A буде не менше ціни гри v.* Аналогічно: *якщо гравець B використовує оптимальну змішану стратегію $\mathbf{q}^*$, а гравець A - будь-яку чисту стратегію A_i, то програш гравця B не перевищить ціни гри v.*

Чисті стратегії гравця, які входять у його оптимальну змішану стратегію з ймовірностями, відмінними від нуля, називаються *активними стратегіями гравця.*

Теорема 4. *Якщо один з гравців притримується своєї оптимальної змішаної стратегії, то його виграш залишається незмінним та рівним ціні гри незалежно від того, яку стратегія застосовує другий гравець, якщо тільки той не виходить за межі своїх активних стратегій.*

На основі даної теореми рішення матричної гри можна спростити, виявивши при цьому домінування одних стратегій над іншими. Так, розглядаючи стратегії гравця A, порівнюємо елементи рядочків s та t, а саме: a_{sj} з елементами a_{tj} для $j=\overline{1,n}$. Якщо $a_{sj} \geq a_{tj}$ $(j=\overline{1,n})$, то виграш гравця A при стратегії A_s буде більше, ніж при стратегії A_t. У цьому випадку стратегія A_s домінує над стратегією A_t. Стратегію A_s називають *домінуючою,* а стратегію A_t *– такою, що домінується.*

Оскільки гравець B зацікавлений у мінімізації програшу, домінуючим буде стовпчик з найменшими елементами. Наприклад, порівнюємо елементи r-го та l-го стовпчиків. Якщо усі елементи $a_{ir} \geq a_{il}$ $(i=\overline{1,m})$, то гравцю B свій вибір вигідно зробити по l-му стовпчику. У цьому випадку стратегія B_l гравця B домінує над стратегією B_r. Стратегія B_l називається *домінуючою,* а стратегію B_r *– такою, яка домінується.*

Якщо у матричній грі маємо рядочки (стовпчики) з одними і тими ж елементами, то рядочки (стовпчики), а відповідно і стратегії гравців A та B, називається *дублюючими.*

У матричній грі домінуючі та дублюючі рядочки (стовпчики) можна опускати, що не впливає на рішення гри.

Теорема 5. *Оптимальні змішані стратегії* $\mathbf{p}^*$ *та* $\mathbf{q}^*$ *відповідно гравців* A *та* B *у матричній грі* $\|a_{ij}\|_{m\times n}$ *з ціною* v *будуть оптимальними і у матричній грі* $\|ba_{ij}+c\|_{m\times n}$ *з ціною* $v'=bv+c$, *де* $b>0$.

На основі теореми 5 платіжну матрицю, яка має від'ємні числа, можна перетворити у матрицю з додатними числами.

2.4. Моделі задач оптимізації

Математична модель об'єкта оптимізації описує об'єкт за допомогою співвідношень між величинами, які характеризують його властивості. Зазвичай хоча б частину цих величин можна змінювати у деяких межах. Величини, які змінюються при оптимізації та входять у математичну модель об'єкта оптимізації, називають параметрами оптимізації, а співвідношення, які встановлюють межі можливої зміни цих параметрів, – обмеженнями. Ці обмеження можуть бути задані у формі рівності або нерівності. Їх називають відповідно обмеженнями типу рівності або обмеженнями типу нерівності [2].

Якщо множина параметрів оптимізації є підмножиною кінцевомірного лінійного простору, то говорять про кінцевомірну задачу оптимізації. При цьому критерієм оптимальності може бути вимога досягнення найбільшого або найменшого значення одною або декількома дійсними (скалярними) функціями параметрів оптимізації, які виражають кількісно міру досягнення цілі оптимізації об'єкта, який розглядається. Кожну із таких функцій прийнято називати цільовою. Якщо цільова функція дійсна, то задачу кінцевомірної оптимізації називають задачею математичного програмування, а у протилежному випадку – задачею багатокритеріальної (векторної) оптимізації.

Якщо цільова функція та обмеження є лінійними відносно параметрів оптимізації, то говорять про задачу лінійного програмування. При нелінійній залежності цільової функції або обмежень від параметрів оптимізації говорять про задачу нелінійного програмування.

2.4.1. Методи безумовної оптимізації

Задачами безумовної оптимізації називаються такі, у яких задається лише одна цільова функція [6]. У таких задачах не існує обмежень і граничних умов. Моделі безумовної оптимізації мають теоретичний характер, оскільки на практиці граничні умови задаються завжди. У цих задачах поняття оптимуму й екстремуму збігаються, для знаходження оптимуму у них застосовуються методи знаходження екстремуму.

Цільова функція може набувати найбільше або найменше значення або мати екстремум. Екстремуми графічно подані на рис. 2.1.

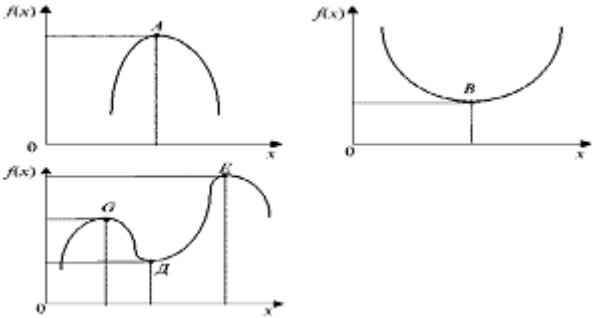


Рис. 2.1. Екстремуми функції

Оптимум – ширше поняття, ніж екстремум. Якщо екстремум є не у всіх функцій, то в практичних задачах оптимуму існує завжди. Графічно оптимуми представлені на рис. 2.2.

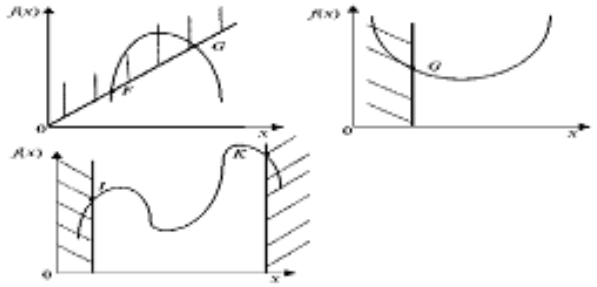


Рис. 2.2. Оптимуми функції

Аналітичний метод розв’язування задачі безумовної оптимізації.

Задана функція однієї змінної. Щоб визначити екстремум, необхідно:

1. Знайти першу похідну функції.

2. Прирівняти її до нуля.

3. Розв'язати рівняння.

4. Знайти другу похідну функції. Визначити знак цієї похідної. Якщо похідна менша за 0, то точка x – максимум функції. Якщо друга похідна більша за 0, то точка x – мінімум функції.

2.4.2. Методи оптимізації з обмеженнями

Задачі умовної оптимізації включають у себе обмеження та граничні умови, що відповідають існуючим економічним умовам.

Методи розв'язування задач умовної оптимізації [9].

1. Метод штрафних функцій. Ідея методу полягає у заміні цільової функції даної задачі деякою узагальненою функцією, значення якої співпадають із значеннями вихідної функції всередині допустимої області, але при наближенні до границі області, а тим більше при виході із неї різко зростають за рахунок другого доданку узагальненої функції – штрафної функції. Штрафні функції будуються таким чином, що забезпечують або швидке повернення у допустиму область, або неможливість виходу із неї. Методи штрафних функцій зводять задачу ну умовний екстремум до розв'язку послідовності задач на безумовний екстремум,що нерідко виявляється значно простіше. Ефективність такого підходу стає особливо відчутною, коли обмеження вихідної задачі задані нелінійними функціями. У залежності від способу формування штрафних функцій розрізняють метод штрафних та метод бар'єрних функцій.

Розглянемо метод штрафних функцій. Нехай необхідно мінімізувати функцію

$$z = f(x) \tag{2.12}$$

при обмеженнях

$$\varphi_i(x) \le 0 \qquad (i = \overline{1, m}). \tag{2.13}$$

Припускаємо, що обмеження $x \ge 0$ включені в обмеження (2.13). Функція $T(x, t)$, узагальнена для функції (2.12), має вигляд

$$T(x,t) = f(x) + t\theta(x), \qquad (2.14)$$

де t – деяке додатне число, яке називається коефіцієнтом штрафу; $\theta(x)$ – неперервна функція штрафу, яка задовольняє умовам: $\theta(x) = 0$ для усіх точок x допустимої області та $\theta(x) > 0$ для усіх інших точок.

Добре вивчені штрафні функції видів:

$$\theta_1(x) = \sum_{i=1}^{m} (\max\{0, \varphi_i(x)\})^{\alpha},$$

(де $\alpha = 1;\ \alpha = 2$) і

$$\theta_2(x) = \sum_{i=1}^{m} 0{,}25(\varphi_i(x) + |\varphi_i(x)|)^2.$$

Процедура оптимізаційного пошуку за методом штрафних функцій полягає у наступному. Розглядається деяка необмежена, монотонно зростаюча послідовність $\{t_k\}$ $(k = 1,2,\ldots)$ додатних чисел. Для першого числа t_1 цієї послідовності знаходиться точка x_1^*, яка надає мінімум функції (2.14). Знайдена точка x_1^* використовується як початкове наближення для розв'язку задачі пошуку мінімуму функції $T(x,t_2)$, де $t_2 > t_1$ і т.д. Таким чином розв'язується послідовність задач мінімізації функції $T(x,t_k)$ $(k = 1,2,\ldots)$, причому результат попередньої оптимізації x_k^* використовується у якості початкового наближення для пошуку x_{k+1}^*. Оскільки для нескінченно зростаючої послідовності $\{t_k\}$ локальні мінімуми наближуються до допустимої області, то послідовність $\{x_k^*\}$ $(k = 1,2,\ldots)$ збігається до локального оптимуму функції $f(x)$, який розташований всередині або на границі допустимої області. Точки x_k^* розташовані поза допустимої області, тому метод штрафних функцій також називають методом зовнішньої точки. У цьому методі будь-яка точка може бути вибрана у якості початкової, що значно спрощує машинне програмування алгоритму розв'язку задачі.

Метод Лагранжа для розв'язування задач оптимізації на умовний екстремум. Це метод є класичним методом розв'язання

задач математичного програмування (зокрема опуклого). Розглянемо класичну задачу оптимізації

$$\max(\min)\, z = f(x); \tag{2.6}$$

$$\varphi_i(x) = b_i \qquad (i = \overline{1,m}),\ x = (x_1, \ldots, x_n). \tag{2.7}$$

Класичний підхід до розв'язання задачі (2.6), (2.7) дає систему рівнянь (необхідні умови), яким повинна задовольняти точка x^*, яка надає функції $f(x)$ локальний екстремум на множині точок, які задовольняють обмеженням (2.7).

Припустимо, що у точці x^* функція (2.6) має локальний умовний екстремум та ранг матриці $\|\partial\varphi_i / \partial x_j\|_{m\times n}$ дорівнює m. Тоді необхідні умови запишуться у вигляді:

$$\left.\begin{aligned} \frac{\partial L}{\partial x_j} &= \frac{\partial f}{\partial x_j} - \sum_{i=1}^{m} \lambda_i \frac{\partial \varphi_i}{\partial x_j} = 0 \quad (j = \overline{1,n}), \\ \frac{\partial L}{\partial \lambda_i} &= b_i - \varphi_i = 0 \qquad\qquad (i = \overline{1,m}), \end{aligned}\right\} \tag{2.8}$$

де

$$L(x_1, \ldots, x_n, \lambda_1, \ldots, \lambda_m) = f(x_1, \ldots, x_n) + \\ + \sum_{i=1}^{m} \lambda_i (b_i - \varphi_i(x_1, \ldots, x_n)) \tag{2.9}$$

є функція Лагранжа; $\lambda_1, \ldots, \lambda_m$ – множники Лагранжа.

Існують також і достатні умови, при виконанні яких розв'язок системи рівнянь (2.8) визначає точку екстремуму функції $f(x)$. Це питання вирішується на основі дослідження знаку другого диференціала функції Лагранжа. Однак достатні умови представляють головним чином теоретичний інтерес.

Можна вказати наступний порядок розв'язання задачі (2.6), (2.7) методом множників Лагранжа:

1) скласти функцію Лагранжа (2.9);

2) знайти часткові похідні функції Лагранжа за всіма змінними $x_1, \ldots, x_n, \lambda_1, \ldots, \lambda_m$ та прирівняти їх до нуля. Тим самим буде отримана система (2.8), яка складається із $m+n$ рівнянь. Розв'язати отриману систему (якщо це виявиться можливим) і знайти таким чином усі стаціонарні точки функції Лагранжа;

3) із стаціонарних точок, взятих без координат $\lambda_1,\ldots,\lambda_m$, вибрати точки, у яких функція $f(x)$ має умовні локальні екстремуми при наявності обмежень (2.7). Цей вибір здійснюється, наприклад, із застосуванням достатніх умов локального екстремуму. Часто дослідження спрощується, якщо використовувати конкретні умови задачі.

Питання для самоконтролю

1. Теорія ігор.
2. Визначення матричної гри.
3. Безумовна оптимізація та оптимізація з обмеженнями.

3.1. Електрична модель транспортної задачі програмування

Добре відомо, що для електромагнітних систем, окремим випадком яких є електричні та електронні кола, справедливий принцип енергетичного мінімуму, який полягає у тому, що ці системи (кола) намагаються бути у найнижчому енергетичному стані, який сумісний із обмеженнями, накладеними на систему. Із цього принципу може бути виведений, зокрема, другий закон Кірхгофа [14]. Ще у минулому столітті Максвел помітив, що у колах, які містять опори та джерела струму, розподіл струмів, який здійснює «найменшу теплову дію», задовольняє законам Кірхгофа.

Принцип екстремальності для електричних кіл, які містять, крім джерел та опорів, також й ідеальні діоди, був сформульований Деннісом у книзі «Математичне програмування та електричні кола», де він детально дослідив аналогію між електричними колами та оптимальними рішеннями задач математичного програмування, основану на використанні принципу енергетичного мінімуму.

3.1.1. Діодна аналогія Денніса

Для транспортних задач є дуже проста та наочна аналогія у вигляді електричного кола, яке складається з ідеальних діодів, джерел струму та джерел ЕРС. Ця аналогія запропонована Деннісом для фізичної інтерпретації задач про розподіл потоку у колі [3].

У загальній задачі про розподіл потоку кожна гілка характеризується вартістю одиниці потоку та верхньою межею пропускної спроможності. Для моделювання даних параметрів Данніс запропонував електричну гілку, зображену на рис. 3.1.

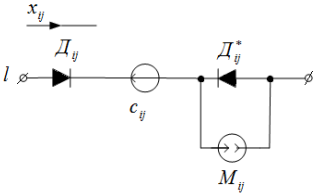


Рис. 3.1. Електрична гілка

Діод $Д_{ij}$ забезпечує однонаправленість потоку у гілці, джерело ЕРС c_{ij} моделює вартість одиниці потоку, а паралельне з'єднання джерела струму M_{ij} та діода $Д^*_{ij}$ обмежує зверху величину струму x_{ij} у гілці, який подібний шуканому потоку.

Якщо за умовами задачі потік між вузлами i та j може мати будь-який напрямок, необхідно включити паралельно дві гілки. На рис. 3.2 наведена модель загальної задачі про розподіл потоку у мережі.

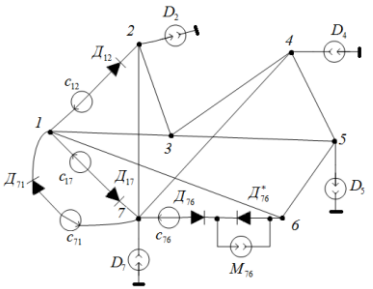


Рис. 3.2. Модель загальної задачі про розподіл потоку у мережі

На основі цієї схеми легко можуть бути побудовані моделі окремих задач про розподіл потоку.

Модель задачі про максимальний потік у мережі складається із двополюсників, зображених на рис. 3.3, в якості гілок мережі та джерела струму I із струмом нескінченної величини, який підключається між початком та кінцем сітьової моделі (рис.3.3).

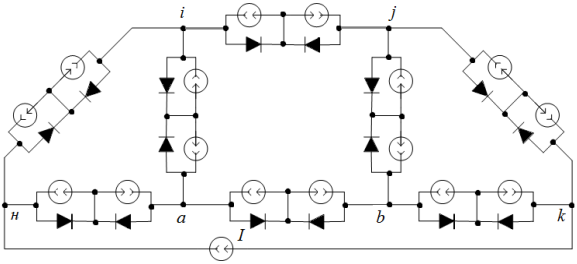


Рис. 3.3. Модель задачі про максимальний потік у мережі

Практично величина струму додаткового джерела повинна перевищувати величину максимального потоку, тобто

$$I \geq kF_{\max} .$$

Замість джерела струму I можна використати джерело одиничної ЕРС (рис. 3.4). Згідно принципу мінімуму потужності, яка споживається або виділяється елементами електричного кола, розподіл струмів у моделі відповідає розподілу потоку у колі, а струм, який протікає по зовнішньому контуру, – максимальному потоку.

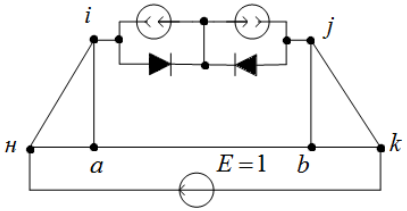


Рис. 3.4. Модель задачі про максимальний потік у мережі з джерелом ЕРС

Зупинимося більш детально на моделюванні транспортної задачі.

Розглянемо схематичне зображення транспортної мережі, яка складається з двох пунктів виробництва, трьох пунктів споживання та шести гілок, які зв'язують їх один з одним (рис. 3.5).

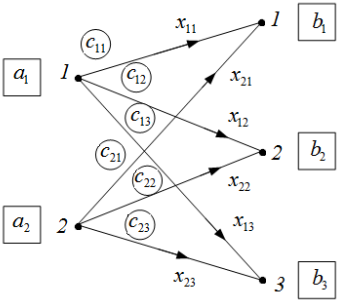


Рис. 3.5. Схематичне зображення транспортної мережі

Рівняння цієї задачі будуть мати вигляд

$$
\begin{aligned}
x_{11} + x_{12} + x_{13} &= a_1,\\
x_{21} + x_{22} + x_{23} &= a_2,\\
x_{11} + x_{21} &= b_1,\\
x_{12} + x_{22} &= b_2,\\
x_{13} + x_{23} &= b_3,
\end{aligned}
\qquad (3.1)
$$

$$C = c_{11}x_{11} + c_{12}x_{12} + c_{13}x_{13} + c_{21}x_{21} + c_{22}x_{22} + c_{23}x_{23} \to \min,\ x_{ij} \ge 0\ .$$

Електрична схема, яка моделює (3.1), наведена на рис. (3.6). Пункти виробництва та споживання моделюються джерелами струму, а транспортні гілки – електричними гілками, які складаються з діодів та джерел напруги.

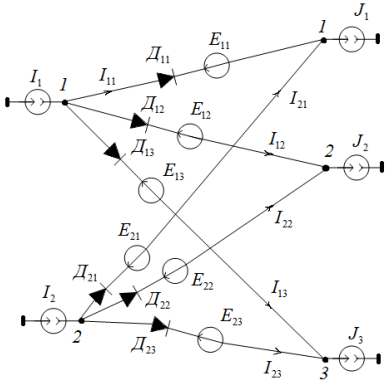


Рис. 3.6. Електрична схема транспортної задачі

Рівняння першого закону Кірхгофа, записані для вузлів електричного кола рис. 3. 6, мають вигляд

$$
\begin{aligned}
I_{11} + I_{12} + I_{13} &= I_1,\\
I_{21} + I_{22} + I_{23} &= I_2,\\
I_{11} + I_{21} &= J_1,\\
I_{12} + I_{22} &= J_2,\\
I_{13} + I_{23} &= J_3,
\end{aligned}
\qquad (3.2)
$$

Завдяки наявності діодів $Д_{ij}$ струми I_{ij} підпорядковуються співвідношенням $I_{ij} \ge 0$.

Якщо припустити, що діоди є ідеальними випрямлячами, то потужність, яка поставляється у коло джерелами струму, буде дорівнювати потужності, яка розсіюється на джерелах напруги E_{ij}:

$$P = E_{11}I_{11} + E_{12}I_{12} + E_{13}I_{13} + E_{21}I_{21} + E_{22}I_{22} + E_{23}I_{23}. \qquad (3.3)$$

На основі енергетичних законів електричних кіл розподіл струмів забезпечує мінімум потужності, які розсіюються на елементах.

Якщо прийняти, що

$$\begin{aligned} E_{ij} &= \gamma_c c_{ij}, \\ I_i &= \gamma_x a_i, \\ J_j &= \gamma_x b_j, \\ I_{ij} &= \gamma_x x_{ij}, \end{aligned} \qquad (3.4)$$

де γ_c, γ_x – масштабні коефіцієнти, то $C = \gamma_x \gamma_c P$. Значення струмів, які протікають у схемі 3.6, будуть пропорційні об'ємам перевезень оптимального плану у масштабі γ_x.

Якщо ввести в схему-аналог транспортної гілки ключ, то легко здійснювати розв'язок задач за критерієм часу і за змішаним критерієм. У першому випадку необхідно накладати заборону на гілки за одним з алгоритмів розв'язку таких задач до тих пір, поки на кроці, який йде за досягненням оптимуму, не порушується рівність $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$. У іншому випадку достатньо розімкнути гілки, на які за умовами задачі накладається заборона.

Обмеження пропускної спроможності гілки на рівні можна здійснити за допомогою паралельного з'єднання діода $Д_{ij}$ та джерела струму M_{ij}. Схема представлена на рис. 3.7.

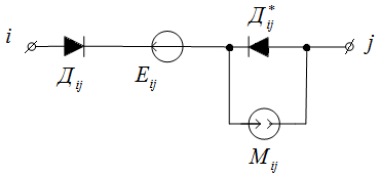


Рис. 3.7. Схема обмеження пропускної спроможності гілки

До недолікам схеми Денніса необхідно віднести необхідність застосування діодів та великого числа джерел ЕРС, які моделюють елементи матриці вартостей. Перша обставина приводить до появи похибки рішення, яка обумовлена падінням напруги на діодах, які проводять. Однак, як показав досвід, похибка не перевищую 3% для задач доволі великих розмірів – $m \times n = 600$. Кількість випрямлячів для джерел ЕРС c_{ij}, які регулюються, можна зменшити.

3.1.2. Задачі на екстремальні шляхи

Задачі на екстремальні шляхи відносяться до задач сітьового аналізу. У загальному вигляді виникає дві постановки задачі [3, 14]:

1. Визначити найкоротший шлях між заданими вузлами, тобто шлях, який має мінімальну довжину

$$\min_k l\left(\tilde{t}_{ij}^{k}\right);$$

2. Визначити максимальний (критичний) шлях, тобто шлях, який має максимальну довжину

$$\max_k l\left(\tilde{t}_{ij}^{k}\right).$$

Прикладом задачі на знаходження найкоротшого шляху є транспортна задача.

Задача на знаходження максимального (критичного) шляху є основною задачею розрахунку сітьового графіку і вона полягає у знаходженні форми та довжини максимального шляху між початковою та кінцевою подіями сітьового графіку.

Основними характеристиками сітьового графіку при знаходженні критичного шляху є такі:

$t_{\text{кр.}}$ – довжина критичного шляху (максимально можлива сума тривалості виконання робіт, які лежать на одному шляху, який поєднує початкову та кінцеву подію графіку);

$t_{\text{рп}}^{ij}$ – найбільш ранній (мінімальний) час можливого початку роботи, який поєднує події S_i та S_j, який визначається як довжина критичного шляху між початковою подією графіку та початковою подією вибраної роботи:

$$t_{\text{рп}}^{ij} = t_{\text{кр}}^{\text{пі}};$$

$t_{\text{рз}}^{ij}$ – найбільш ранній (мінімальний) час можливого закінчення роботи, який з'єднує події S_i та S_j, більший $t_{\text{рп}}^{ij}$ на величину тривалості роботи:

$$t_{\text{рз}}^{ij} = t_{\text{рп}}^{ij} + t_{ij};$$

$t_{\text{пз}}^{ij}$ – найбільш пізній (максимальний) час допустимого закінчення роботи, який поєднує події S_i та S_j:

$$t_{\text{пз}}^{ij} = t_{\text{кр}}^{\text{пк}} - t_{\text{кр}}^{j\text{к}};$$

$t_{\text{пп}}^{ij}$ – найбільш пізній (максимальний) час допустимого початку роботи, менший $t_{\text{пз}}^{ij}$ на величину тривалості виконання роботи:

$$t_{\text{пп}}^{ij} = t_{\text{пз}}^{ij} - t_{ij} = t_{\text{кр}}^{\text{пк}} - t_{\text{кр}}^{j\text{к}} - t_{ij};$$

$P_{\text{п}}^{ij}$ – повний резерв часу роботи, тобто збільшення тривалості роботи, яке не призводить до збільшення довжини критичного шляху:

$$P_{\text{п}}^{ij} = t_{\text{кр}}^{\text{пк}} - t_{\text{кр}}^{\text{пі}} - t_{\text{кр}}^{j\text{к}} - t_{ij}.$$

3.2. Механічні моделі за екстремальні шляхи на графах

Відома механічна модель задачі про найкоротший шлях використовує гнучкі нитки, які не розтягаються [3]. Кожний елемент множини Y графу $G(Y,U)$ моделюється з'єднанням ниток, а гілка – ниткою. Довжина нитки пропорційна тривалості, яка відповідає гілці графу. Найкоротший шлях визначається розтягуванням вірьовочної сітки між заданими вузлами. Натягнути нитки визначають конфігурацію найкоротшого шляху, величина якого визначається відстанню між початковим та кінцевим вузлами. Модель працює тільки для симетричних графів.

Механічна модель сітьового графіку при реалізації задачі про найдовший шлях використовує у якості моделей гілок жорсткі стержні, а у якості моделей подій – тонкі діафрагми. Стержні та діафрагми розташовують у відповідності з топологією сітьового

графіку, тобто кожний стержень – між тими діафрагмами, які відповідають вузлам даної роботи. Довжина стержня пропорційна довжині відповідної гілки. Якщо зблизити до кінця діафрагми, які зображують вихідну та завершальну події графу, то сукупність стиснених стержнів визначить конфігурацію критичного шляху, а відстань між крайніми діафрагмами – його довжину. Стержні, які зображують гілки, які не лежать на критичному шляху, будуть вільними.

3.3. Моделювання класичної транспортної задачі у матричній формі

Перейдемо до наступного окремого випадку – транспортній задачі у матричній постановці [3]. Вона отримала свою назву від особливості конфігурації кола: вузли у ній діляться на дві групи, причому кожен вузол однієї групи поєднаний гілкою з кожним вузлом другої групи. Ця властивість дозволяє зручно записувати умови у вигляді матриці.

Отже, транспортна задача формулюється таким чином. Нехай a_i $(i=1,2,\ldots,m)$ – кількість одиниць однорідного продукту, який виробляється у i-му пункті виробництва, а b_j $(j=1,2,\ldots,n)$ – кількість одиниць цього продукту, яка необхідна j-му пункту споживання.

Нехай загальна кількість продукту, який виробляється у всіх m пунктах виробництва, дорівнює сумі потреб n пунктів споживання.

Вартість перевезення з i-го пункту виробництва у j-й пункт споживання відома і дорівнює сумі c_{ij}, x_{ij} – кількість одиниць продукту, яку перевозять за маршрутом ij. Необхідно визначити такі x_{ij}, для яких сумарна вартість перевезень мінімальна. Зустрічні перевезення заборонені.

Отже, необхідно мінімізувати лінійну форму

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \qquad (3.5)$$

за умов

$$\sum_{j=1}^{n} x_{ij} = a_i \qquad (i = 1,2,\ldots,m),$$

$$\sum_{i=1}^{m} x_{ij} = b_j \qquad (j = 1,2,\ldots,n), \tag{3.6}$$

$$\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j, \qquad x_{ij} \geq 0.$$

У результаті розв'язку отримаємо план перевезень, оптимальний за критерієм вартості.

Існує два різновиди транспортної задачі за критерієм часу: про перевезення за мінімально можливий час і за час, не більший заданого (при мінімальній вартості перевезення). Такі задачі виникають при перевезенні продуктів, які швидко псуються. Вони відрізняються від звичайної транспортної задачі накладанням додаткової умови. У першому випадку мінімізується $\tau_{ij\text{макс}/x_{ij}>0}$, де τ_{ij} – елементи матриці часу. У цьому випадку вимога мінімуму $\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$ відміняється.

У другому випадку $\tau_{ij/x_{ij}>0} \leq$ [illegible]. Мається на увазі, що $\tau_{ij} = kc_{ij}$, тобто час перевезення пропорційний його вартості. Тут вирішується задача про мінімізацію вартості перевезень при накладанні заборони перевезення на ті гілки, час проїзду по яким більший заданого.

Якщо перевезенню підлягає продукт декількох видів (але не більше трьох-чотирьох) з різними коефіцієнтами взаємної замінюваності, то задачі можна вирішувати за методикою, запропонованою Бірманом. Кожний пункт розбивається на підпункти, кількість яких дорівнює числу видів продукту. Кількість приведеного продукту у кожному підпункті та відстані між ними обчислюються, виходячи із коефіцієнтів взаємної замінюваності та умов задачі.

Часто зустрічаються задачі, у яких сума запасів перевищує суму потреб або навпаки. Такі задачі називаються «відкритими». У цьому випадку необхідно організувати фіктивний пункт

споживання (виробництва), причому вартості перевезення із усіх реально існуючих пунктів виробництва (споживання) у фіктивний пункт рівні між собою. Ясно, що величина вартості перевезень у цей фіктивний пункт не має значення – мінімізується вартість перевезень в іншій частині задачі.

До «відкритої» транспортної задачі зводиться і задача розміщення підприємств, сутність якої полягає у наступному. Є декілька площадок розміщення підприємств, яке передбачається, (пунктів виробництва) та група споживачів, які передбачаються. Так як сума запасів значно перевищує суму потреб, потрібний фіктивний пункт. Якщо в оптимальному варіанті запаси деяких пунктів виробництва повністю споживаються фіктивним пунктом, то їх побудова нераціональна. Щоб визначити раціональні об'єми виробництва інших пунктів, достатньо від об'єму, який передбачається, i -го пункту відняти $x_{i\phi}$ – величину перевезення з нього у фіктивний пункт споживання.

Необхідно згадати ще однин цікавий різновид транспортної задачі – так звану неоднорідну транспортну задачу, у якій

Мінімізації підлягає функція

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} \tag{3.7}$$

де

$$c_{ij}x_{ij} = \begin{cases} 0 & \text{при} \quad x_{ij} = 0, \\ c_{ij}x_{ij} + d_{ij} & \text{при} \quad x_{ij} > 0, \end{cases}$$

при обмеженнях (3.6).

Числа d_{ij} можна інтерпретувати, як фіксовані витрати, пов'язані з організацією гілки ij і які не залежать від величини x_{ij}.

Точних аналітичних методів розв'язку цієї задачі поки що немає. Існують наближені методи, в основі яких лежить розв'язок транспортної задачі з перетвореною матрицею вартостей. Перетворення виконується за формулою

$$c_{ij}^{*} = c_{ij} + \frac{d_{ij}}{m_{ij}}, \tag{3.8}$$

де

$$m_{ij} = \min(a_i, b_j).$$

Запропоновані у літературі методи моделювання загальної задачі лінійного програмування мало придатні для розв'язку транспортної задачі із-за її специфічних особливостей. Із співвідношень

$$\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}, \tag{3.9}$$

$$\begin{aligned} &\sum_{j=1}^{n} x_{ij} = a_i \qquad (i = 1,2,\ldots,m),\\ &\sum_{i=1}^{m} x_{ij} = b_j \qquad (j = 1,2,\ldots,n),\\ &\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j, \qquad x_{ij} \ge 0, \end{aligned} \tag{3.10}$$

видно, що кількість змінних значно перевищує число рівнянь, а матриця коефіцієнтів лінійних обмежень містить тільки нулі та одиниці. Таким чином, улаштування для завдання коефіцієнтів лінійних обмежень у пристроях для розв'язку загальної задачі повністю не використовувалося б. Крім того, нераціонально використовувалася б апаратура.

Питання для самоконтролю

1. Сутність діодної аналогії Денніса.
2. Визначення задач на екстремальні шляхи.
3. Механічна модель задачі про найкоротший шлях.
4. Матрична форма транспортної задачі.

4.1. Теорія графів, що використовується в сітьовому аналізі

Багато задач аналізу та розрахунку мереж зводяться до задач лінійного програмування. При розгляді сітьових задач зручно користуватися елементами теорії графів [3]. Граф – це сукупність елементів множини Y (на рис. 4.1, *а* елементи множини Y зображуються точками a, b, i, j і т.д.) і закон перетворення елементів даної множини у цю ж множину. Наприклад, для графу рис. 4.1, а $G(Y,U)$:

$$Y = \{a, b, c, d, i, j, h, f, g\}$$

є множиною елементів, зображених вузлами мережі;

$$U = \{(a,i), (a,d), (i,c), (c,c), (c,d), (c,j), (j,h), (d,f), (d,g)\}$$

є множиною елементів, зображених направленими гілками мережі.

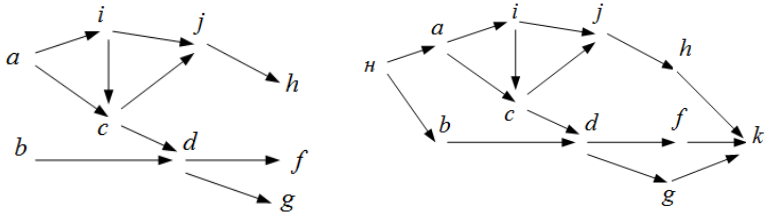


Рис. 4.1. Граф

Якщо у графі для кожної гілки (i, j) мається симетрична їй гілка (j, i), причому усі параметри цих гілок рівні, то граф називається симетричним і кожна ненаправлена гілка симетричного графа зображує дві направлені гілки.

Довжиною гілки називається числовий параметр, який характеризує дану гілку. Довжину гілки (i, j) позначимо t_{ij}. Нехай у графі від вузла i до вузла j – це послідовність гілок та вузлів у графі, яка починається від вузла i і закінчується у вузлі j. Граф рис. 4.1, *а* має три шляхи між точками a, j:

$$\mu_{aj}^1 = (a, i, j),$$
$$\mu_{aj}^2 = (a, i, c, j),$$
$$\mu_{aj}^3 = (a, c, j).$$

Довжина шляху визначається як сума довжин гілок, які розташовані на даному шляху. Довжини шляхів між вузлами a та j визначаються за наступними співвідношеннями:

$$l(\mu_{aj}^1) = \sum_{p,q \in \mu_{aj}^1} t_{pq} = t_{ai} + t_{ij},$$
$$l(\mu_{aj}^2) = \sum_{p,q \in \mu_{aj}^2} t_{pq} = t_{ai} + t_{ic} + t_{cj},$$
$$l(\mu_{aj}^3) = \sum_{p,q \in \mu_{aj}^3} t_{pq} = t_{ac} + t_{cj}.$$

Контур – це шлях, який починається та закінчується в одному й тому ж вузлі.

Дві основні проблеми сітьового аналізу полягають у визначенні найкоротшого шляху між заданими вузлами, тобто шляху, який має мінімальну довжину

$$\min_k l(\mu_{ij}^k),$$

і визначенні максимального (критичного) шляху, тобто шляху, який має максимальну довжину

$$\max_k l(\mu_{ij}^k).$$

У загальному випадку граф може містити декілька як найкоротших, так і критичних шляхів між заданими елементами.

У мережі може бути декілька початкових вузлів, тобто елементів, у які не відображається жодний елемент, і декілька кінцевих вузлів, тобто елементів, відображення який є пуста множина. Подібна мережа може бути перетворена у мережу, яка має один вихідний (початковий) та один завершальний (кінцевий) вузли. Для цього вводять додатковий початковий вузол, який з'єднують гілками нульової довжини із всіма вихідними вузлами. Аналогічні перетворення здійснюють для усіх кінцевих вузлів, вводячи додатковий кінцевий вузол. Таким чином, наприклад мережа рис. 4.1, *а* перетворюються у мережу рис. 4.1, *б* .

На рис. 4.1, *б*

$$t_{на} = t_{нb} = t_{hk} = t_{fk} = t_{qk} = 0 .$$

4.2. Задача найкоротшого шляху

У математичній формі задача про найкоротший шлях записується наступним чином [3].

Мінімізувати

$$\sum t_{ij} x_{ij}$$

за умов

$$\sum_j x_{ij} - \sum_j x_{ji} = \begin{cases} 1, & i = \text{н}, \\ -1, & i = k, \\ 0, & i \neq \text{н}, k \end{cases}$$

і

$$x_{ij} \geq 0 .$$

Нижче наведені деякі приклади задачі про найкоротший шлях.

1. Довжина гілки зображує відстань між пунктами у розгалуженій мережі доріг. У цьому випадку задача визначенні найкоротшого шляху еквівалентна визначенню найкоротшої відстані та послідовності доріг між заданими пунктами.

2. Довжина гілки визначається приведеним часом проїзду між двома пунктами. Приведений час проїзду може враховувати фактичну довжину ділянки дороги, нахил або підйом, стан дороги, завантаження її транспортом і т.д. У цьому випадку задача визначення найкоротшого шляху еквівалентна задачі визначення мінімального часу та маршруту проїзду між заданими пунктами.

3. До задачі визначення найкоротшого шляху на графі зводиться задача визначення найбільш надійного шляху у системах зв'язку.

Часто у системах зв'язку інформація може бути передана не єдиним шляхом. Якщо ж між вихідними та кінцевими вузлами передачі є декілька проміжних станцій, то уся система представляється у вигляді мережі, у якій гілки відповідають каналам зв'язку, а вузли – станціям (приймаючі-передаючі пункти). Кожній станції та каналу зв'язку відповідають належні їм характеристики, такі, як ємність, надійність, пропускна

спроможність. Необхідність швидкої та ефективної передачі потребує вибору маршруту для передачі інформації із врахуванням характеристик системи зв'язку. Однією із задач такого роду є проблема вибору найбільш надійного шляху. Сутність проблеми полягає у наступному. Відома надійність кожного каналу в окремості. Під надійністю, зокрема, розуміється ймовірність того, що даний канал діючий, тобто що він не зруйнований і не перекритий. У цьому випадку проблема визначення найбільш надійного шляху полягає у виборі з найбільшою ймовірністю діючого маршруту. Якщо через кожний канал зв'язку можуть бути передані декілька повідомлень, а пропускна спроможність каналу обмежена, то за надійність каналу можна прийняти ймовірність того, що даний канал повністю не зайнятий і може передати інформацію. Крім того, під надійністю можна розуміти ймовірність правильної передачі інформації. У всіх цих прикладах кожен канал характеризується ймовірністю ефективності його роботи. Зрозуміло, що величина надійності кожного каналу змінюється у межах

$$0 \le p_{ij} \le 1 .$$

Надійність шляху $\mu^s_{\text{н,к}}$ між вузлами Н та К визначається як добуток надійностей каналів, які складають даний шлях, тобто

$$P_s = \prod_{(i,j) \in \mu^s_{\text{н,к}}} p_{ij} .$$

При заданому початковому Н та кінцевому К вузлах задача визначення найбільш надійного шляху полягає у визначенні шляху $\mu^q_{\text{н,к}}$, який має максимальну величину надійності

$$P_q = \max_s P_s = \max_s \prod_{(i,j) \in \mu^s_{\text{н,к}}} p_{ij} .$$

Відомо, що проблема визначення найбільш надійного шляху може бути перетворена у задачу визначення найкоротшого шляху. Для цього мережа з відомими величинами надійності p_{ij} замінюють такою ж мережею, у які довжина гілки рівна абсолютній величині логарифму надійності відповідного каналу зв'язку:

$$t_{ij} = \left| \log p_{ij} \right| .$$

Задача визначення максимуму добутку надійності

$$\max \prod p_{ij}$$

зводиться до визначення мінімуму величини

$$\sum t_{ij},$$

так як шлях максимальної надійності $\mu_{\text{н,к}}^{q}$ має $\max \prod p_{ij}$, або

$$\max \log \prod p_{ij} = \max \sum \log p_{ij},$$

або мінімум

$$\sum \left(-\log p_{ij} \right) = \sum t_{ij}.$$

Остання рівність справедлива, тому що $\log p_{ij}$, так як

$$p_{ij} \leq 1.$$

Таким чином, прописавши для гілки (i, j) величину її довжини $t_{ij} = \left| \log p_{ij} \right|$, перетворимо задачу визначення найбільш надійного шляху до задачі визначення найкоротшого шляху.

Алгоритм розв'язання задачі про найкоротший шлях складається з двох простих операцій: арифметичної операції додавання та логічної операції визначення мінімальної величини серед декількох. Для кожного вузла визначається гілка, яка володіє мінімальною довжиною шляху серед гілок, які входять у даний вузол. Ця мінімальна довжина шляху приписується даному вузлу. До отриманої довжини шляху вузла додається довжина гілки, яка виходить із даного вузла. Сума є довжиною шляху для відповідної гілки. Для вихідного пункту довжина шляху дорівнює нулю. Починаючи з вихідного пункту, визначаємо довжини шляхів для гілок, які починаються у даному пункті (для гілок, які починаються у вихідному пункті, ці величини співпадають з довжинами відповідних гілок). Визначаємо довжини шляхів для тих вузлів, у яких усі гілки, які входять, мають обчислені довжини шляхів. Довжина шляху кінцевого вузла буде дорівнювати довжині найкоротшого шляху, а довжина шляху визначеного вузла дорівнює найкоротшому шляху від вихідної події до даного вузла. Процес обчислень для кожного вузла виконується один раз. Більш простий (але й більш тривалий) алгоритм відрізняється від наведеного тим, що послідовність вибору вузлів при обчисленнях

довільна. Спочатку кожному вузлу приписується величина, яка перевищує величину найкоротшого шляху. Процес обчислень триває до тих пір, поки обчислені довжини шляхів для усіх вузлів не почнуть повторюватися. Приклад такого алгоритму запишеться наступним чином.

1. Приписуємо усім елементам множини Y, які розташовані на якому-небудь шляху μ_{mn}^{s}, позначення у формі $(0, U_i)$, де $U_m = 0$ для початкового вузла та $U_i = \infty$ для усіх інших $i \in Y$.

2. Визначаємо елемент $(j) \in \mu_{mn}^{s}$, для якого

$$0 < \Delta U = U_j - U_i - U_{ij}.$$

Якщо такий елемент відсутній, переходимо до п. 4. Якщо знайдений – до п. 3.

3. Замінюємо раніше прописані позначення елемента j новим

$$(U_i + t_{ij})$$

і повертаємося до п. 2.

4. Процес закінчений, при цьому $U_n = \min \mu_{mn}^{s}$. Друга частина прописаного позначення останнього вузла дає нам величину мінімальної довжини шляху, а перша частина позначення, яке вводиться, вказує на попередній елемент, який розташований на шляху мінімальної довжини. У свою чергу, перша частина позначення цього елемента, яке приписується, вказує на попередній і т.д. Прослідкувавши почергово усі ці вузли, визначимо конфігурацію шляху мінімальної довжини.

4.2.1. Моделювання задачі найкоротшого шляху

Задачу про знаходження найкоротшого шляху у заданому графі між двома точками можна моделювати різними технічними засобами [3].

1. Відома механічна модель задачі про найкоротший шлях використовує гнучкі нитки, які не розтягаються. Кожний елемент множини Y графу $G(Y, U)$ моделюється з'єднанням ниток, а гілка – ниткою. Довжина нитки пропорційна тривалості, яка відповідає гілці графу. Найкоротший шлях визначається розтягуванням вірьовочної сітки між заданими вузлами. Натягнути нитки

визначають конфігурацію найкоротшого шляху, величина якого визначається відстанню між початковим та кінцевим вузлами. Модель працює тільки для симетричних графів.

2. Перші електричні моделі використовують газорозрядні лампи. Кожна гілка моделі містить газорозрядну лампу, напруга запалення якої пропорційна тривалості відповідної гілки. З'єднання ламп відповідають вузлам мережі. Струм протікає з мінімальним сумарним значенням напруги запалення декількох ламп, які утворюють шлях. Найкоротший шлях у графі утворюється гілками моделі, по яким тече струм. Якщо у якості газорозрядних ламп застосовувати стабілітрони. То величина напруги між певними точками пропорційна довжині найкоротшого шляху. Зміна меж запалення газорозрядних ламп може бути здійснена введенням напівпровідникових стабілітронів або додаткових джерел ЕРС. У цьому випадку можна моделювати несиметричні задачі, тобто графи, які мають гілки різної довжини у прямому та зворотному напрямках.

Однак високі напруги запалення газорозрядних ламп дозволяють використовувати подібні схеми для моделювання тільки простих графів, які складаються з невеликої кількості гілок. Ще одним недоліком цих елементів є труднощі зміни величини напруги запалення.

3. Більш ефективна електрична модель задачі про найкоротший шлях складається з джерел ЕРС, які регулюються, діодів та джерел струму. Кожна гілка в моделі представляє собою двополюсник, який складається з послідовно з'єднаних джерела ЕРС та діода. Величини напруг джерел ЕРС встановлюються пропорційними тривалостям відповідних гілок. Напрямки діода та джерела ЕРС протилежні. До точкам, між якими необхідно визначити найкоротший шлях, приєднується джерело струму. У відповідності з принципом екстремальності потужності для ланцюгів струм проходить по гілкам з мінімальним сумарним значенням ЕРС. Напруга на полюсах джерела струму пропорційна довжині найкоротшого шляху. Конфігурація найкоротшого шляху визначається гілками, по яким протікає струм.

При розгляді роботи описаної моделі припускалося, що опори джерел ЕРС рівні нулю, прямі опори діодів нескінченно малі, а зворотні – нескінченно великі, тобто елементи моделі

ідеальні. Однак прямий опір діода та внутрішній опір джерела ЕРС суттєво впливають на дозволену спроможність моделі.

4. Скориставшись гідравлічними аналогами діода та джерела ЕРС, можна побудувати гідравлічну модель задачі про найкоротший шлях. Принцип дії цієї моделі аналогічний принципу дії наведеної електричної схеми.

5. Аналогією тривалості гілок сітьового графу можуть слугувати часові співвідношення. Цифрові моделі задачі про найкоротший шлях зображують вузли графу за допомогою спеціальних логічних схем, а з'єднання елементів – за допомогою ліній затримки. У якості елементів затримки можуть бути використані. Наприклад, одновібратор, який регулюється, пасивні лінії затримки, реле часу, яке регулюється, перераховані схеми з коефіцієнтами передачі, які змінюються, та ін.

Лінії затримки, які регулюються, дозволяють встановити затримки пропорційно тривалостям відповідних гілок.

Модель призначена тільки для визначення величини найкоротшого шляху, доволі проста. Аналогічно графу збирається модель, яка складається тільки з ліній затримки. Між початковою та кінцевою точками шляху, який досліджується, підключається генератор та приймач сигналу. Час затримки сигналу між генератором та приймачем пропорційний тривалості найкоротшого шляху. Більш складною є модель, призначена для визначення як величини, так і конфігурації найкоротшого шляху. У цьому випадку елементи графу моделюються логічними схемами. Входами логічної схеми є входи ліній затримки, які приходять у даний вузол. При надходженні першого вхідного імпульсу логічна схема блокує усі інші входи. Сигнал, який прийшов до останнього вузла, фіксує час, пропорційний довжині найкоротшого шляху. Після надходження у кінцевий вузол сигнал повертається по відкритим каналам у початковий вузол. Зрозуміло, що елементи моделі повинні бути зворотними. Індикація здійснюється під час проходження сигналів у зворотному напрямку.

4.3. Задача сітьового планування та керування

Найбільш широке розповсюдження сітьовий аналіз досяг у системах сітьового планування та управління [3]. Застосування сітьового планування викликано необхідністю наукового

проектування складних взаємопов'язаних систем. Нові методи планування, контролю та управління складних систем знайшли застосування у наукових дослідженнях, проектно-конструкторських розробках, будівництві, промисловому виробництві. Метод проектування по системі сітьового планування та керування (СПК) потребує зображення проекту у вигляді сітьового графіку з направленими гілками. Графік відповідає комплексу взаємопов'язаних робіт, які необхідно виконати при реалізації проекту. Гілка сітьового графіку відповідає окремій роботі з відомою тривалістю. При цьому поняття роботи (діяльності, операції) необхідно розуміти доволі широко. Під роботою розуміється будь-який процес, який потребує витрат часу (виготовлення деталі, очікування прибуття вантажу, обговорення якого-небудь питання і т.д.). Вузол сітьового графіку – подія – фіксує здійснення усіх робіт, які входять у дану подію, і початок усіх робіт, які виходять з нього. Топологія сітьового графіку відображає взаємний зв'язок та послідовність виконання робіт. Мінімальний час, за який може біти виконаний увесь цикл робіт, визначається найбільш довшим шляхом від початкової події (п) до кінцевої події (к). Цей шлях максимальної довжини називається критичним. Сума тривалостей робіт, які утворюють критичний шлях, дорівнює довжині критичного шляху. Поняття критичного шляху може вводитися не тільки для початкової та кінцевої події, а й для будь-якої пари подій, між якими є хоча б один шлях.

Сітьовий аналіз проекту передбачає визначення наступних часових характеристик графіку.

1. Тривалість критичного шляху між двома подіями i, j. Це – максимальний шлях між цими подіями: $t_{\text{кр}}(i, j)$. Тривалість критичного шляху сітьового графіку – це тривалість максимального шляху між початком та кінцем графіку: $t_{\text{кр}}(\text{п}, \text{к})$.

2. Ранній строк здійснення події i – найбільш ранній із можливих строків здійснення події. Ранній строк здійснення події i дорівнює критичному шляху між початком графіку та подією i:

$$t_{\text{р}}(i) = t_{\text{кр}}(\text{п}, i).$$

3. Пізній строк здійснення події i – самий пізній строк здійснення події i, при якому загальний строк закінчення

комплексу операцій, який планується, не змінюється. Пізній строк здійснення події i визначається різницею між повним критичним шляхом графіку та критичним шляхом із події i у кінці графіку:

$$t_{\text{п}}(i) = t_{\text{кр}}(1, \text{к}) - t_{\text{кр}}(i, \text{к}).$$

4. Резерв часу події i – різниця між пізнім та раннім строками здійснення події i:

$$\text{Р}(i) = t_{\text{п}}(i) - t_{\text{р}}(i).$$

Для подій, які лежать на критичному шляху, резерв часу дорівнює нулю.

5. Ранній строк початку роботи (i, j) – самий ранній із можливих строків початку роботи. Ранній строк початку роботи (i, j) співпадає з раннім строком здійснення події i і дорівнює критичному шляху із початкової події у подію i:

$$t_{\text{рп}}(i, j) = t_{\text{р}}(i) = t_{\text{кр}}(1, i).$$

6. Ранній строк закінчення роботи (i, j) – самий ранній із можливих строків закінчення роботи (i, j). Ранній строк закінчення роботи відрізняється від попередньої характеристики на величину тривалості самої роботи:

$$t_{\text{рз}}(i, j) = t_{\text{рп}}(i, j) + t_{ij} = t_{\text{кр}}(1, i) + t_{ij}.$$

7. Пізній строк початку роботи (i, j) – самий пізній строк початку роботи (i, j), при якому загальний строк закінчення комплексу робіт, який планується, не змінюється. Пізній строк початку роботи (i, j) дорівнює різниці між величиною критичного шляху усього графіку та величинами тривалості роботи (i, j) та критичного шляху між подією j та кінцем графіку К, тобто

$$t_{\text{пп}}(i, j) = t_{\text{кр}}(1, \text{к}) - t_{\text{кр}}(j, \text{к}) - t_{ij}.$$

8. Пізній строк закінчення роботи (i, j) – самий пізній строк закінчення роботи (i, j), при якому загальний строк закінчення комплексу робіт не змінюється. Пізній строк закінчення роботи (i, j) дорівнює пізньому строку здійснення події j:

$$t_{\text{пз}}(i, j) = t_{\text{п}}(j) = t_{\text{кр}}(1, \text{к}) - t_{\text{кр}}(j, \text{к}).$$

9. Вільний резерв часу роботи (i, j) $P_c(i, j)$ – частина повного резерву часу, на яку може бути збільшена тривалість роботи (i, j), щоб при цьому змінилися строки здійснення події j та ранній строк здійснення події i. Вільний резерв часу роботи (i, j) дорівнює різниці між частковим критичним шляхом від події i до події j і тривалістю роботи (i, j).

10. Повний резерв часу роботи (i, j) – це максимальний час, за який можна збільшити тривалість роботи (i, j), не змінюючи при цьому величину загального критичного шляху. Повний резерв часу роботи (i, j) виходить при відніманні від величини повного критичного шляху із вихідної події у подію i, критичного шляху із події j у завершальну подію та тривалості роботи (i, j):

$$P_п(i, j) = t_{кр}(н, к) - t_{кр}(н, i) - t_{кр}(j, к) - t_{ij}.$$

Повний резерв часу робіт, які лежать на критичному шляху, дорівнює нулю.

11. Коефіцієнт напруженості роботи (i, j) – це відношення тривалості відрізків шляху максимальної тривалості, які не співпадають, який проходить через дану роботу, і повного критичного шляху:

$$K_н(i, j) = \frac{t'_{кр}(н, i) + t_{ij} + t'_{кр}(j, к)}{t'_{кр}(н, к)}.$$

Тут

$t'_{кр}(н, i)$ – частина критичного шляху із вихідної події у подію i, яка не співпадає з повним критичним шляхом;

$t'_{кр}(j, к)$ – частина критичного шляху із події j у завершальну подію, яка не співпадає з повним критичним шляхом;

$t'_{кр}(н, к)$ – частина повного критичного шляху, яка не співпадає з вказаними вище окремими критичними шляхами.

Коефіцієнт напруженості робіт, які лежать на критичному шляху, дорівнює одиниці.

4.3.1. Електронна модель сітьового графу з використанням операційних підсилювачів з діодами

Побудова електронних моделей сітьових графіків, основаних на діод ній аналогії Денніса, було пов'язане із подоланням цілого ряду технічних труднощів, пов'язаних із необхідністю побудови великого числа гальванічно розв'язаних джерел напруги [3]. Порівняно низька дозволяюча спроможність схем з реальними діодами призвела до використання у моделі роботи електронних еквівалентів від'ємних опорів.

Вказані недоліки діод них моделей сітьових графіків легко усуваються, якщо застосувати запропоновану нижче модель мережі, яка використовує операційні підсилювачі постійного струму, діоди та джерела напруги. Кількість електронних підсилювачів, необхідних для моделювання сітьових графіків, дорівнює сумі числа робіт та подій мережі, яка досліджується.

Описаний нижче спосіб моделювання сітьових графіків, не дивлячись на значне число електронних підсилювачів, представляє інтерес.

Розглянемо спосіб моделювання сітьових графіків за системами сітьового планування та управління, яка використовує операційні підсилювачі постійного струму, діоди та джерела напруги.

У якості основних машинних змінних візьмемо ранній час здійснення подій (ранній час початку роботи). Для будь-якої пари подій i та j, безпосередньо пов'язаних між собою роботою (i, j) з тривалістю t_{ij}, очевидна наступна нерівність:

$$t_{\text{р}}(j) \geq t_{\text{р}}(i) + t_{ij}. \qquad (4.1)$$

Знак рівності у формулі (4.1) буде у тому випадку, якщо робота (i, j) лежить на частковому критичному шляху до події j, тобто

$$t_{\text{р}}(j) = \max_{i \in I_0} (t_{\text{р}}(i) + t_{ij}), \qquad (4.2)$$

де I_0 – множина подій, безпосередньо передуючих j і пов'язаних з ними роботами.

Співвідношення (4.1) та (4.2) дозволяють побудувати електронну модель роботи за допомогою операційного підсилювача та діода (рис. 4.2).

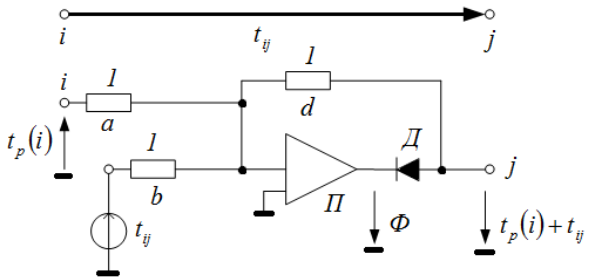


Рис.4.2. Електронна модель роботи з операційним підсилювачем та діодом

Спосіб з'єднання моделей робіт у моделі події показаний на рис. 4.3. Діоди, які включаються у коло зворотного зв'язку підсилювачів моделей робіт здійснюють вибір максимуму (4.2) з високою дозволяючою спроможністю. Висока дозволяюча спроможність схеми пояснюється тим, що падіння напруги на діоді, який проводить, включене у контур зворотного зв'язку підсилювача і завдяки високому коефіцієнту підсилення останнього не впливає на точність розв'язку.

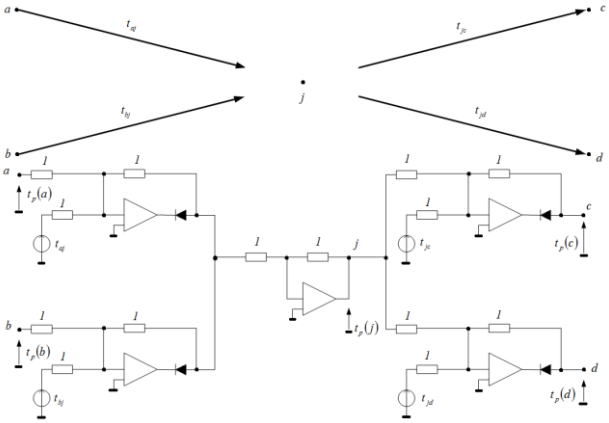


Рис. 4.3. Спосіб з'єднання моделей робіт у моделі події

Розглянемо основні рівняння схеми рис. 4.2 (усі провідності прийняті однаковими):

$$-3\varepsilon + t_{\text{р}}(\ ) + t_{ij} - t_{\text{р}}(\ ) = 0,$$

$$\Phi + U_{\text{Д}} = -t_{\text{р}}(\ ),$$

$$\Phi = -k\varepsilon.$$

Виконуючи елементарні перетворення, отримаємо

$$t_{\text{р}}(\ ) = \frac{t_{\text{р}}(\ ) + t_{ij} - 3\dfrac{U_0}{k}}{1 + \dfrac{3}{k}}.$$

Якщо робота, яка розглядається, не лежить на частковому або загальному критичному шляху, то у відповідній моделі роботи діод *Д* буде закритий, а операційний підсилювач – виведений із лінійного режиму завдяки відсутності кола зворотного зв'язку. Ця обставина дозволяє здійснити індикацію робіт, які лежать на критичному шляху.

При переході від однієї моделі роботи до другої змінюється лише величина напруги, яка моделює тривалість роботи та величини вхідної та вихідної напруги. Параметри інших елементів схеми залишаються без змін. Тому легко виявляється частина схеми, яка може бути виділена у якості групового вирішуючого елемента. Порядок комутації входу та виходу групового вирішуючого елемента задається спеціальною схемою управління.

4.3.2. Моделювання задачі мінімізації вартості розробки при сітьовому плануванні за допомогою діодних ланцюгів

Однією із основних задач, які виникають при плануванні та управлінні будівництвом складних комплексів сітьовими методами, є оптимізація сітьового графіку [3]. Ця оптимізація передбачається методом PERT-Cost, сутність якого полягає у наступному.

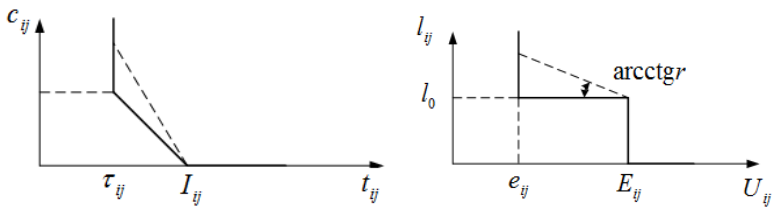
Величини тривалостей окремих робіт, на відміну від системи СПУ, можуть змінюватися у деякому діапазоні

$$0 \le \tau_{ij} \le t_{ij} \le T_{ij} < \infty, \qquad (4.3)$$

причому постійні τ_{ij} та T_{ij} задані.

Приймається, що вартість скорочення кожної роботи зростає лінійно із зменшенням тривалості у діапазоні її зміни (рис. 4.4, *а*):

$$c_{ij} = a_{ij}t_{ij} + b_{ij},\ a_{ij} \le 0\,. \qquad (4.4)$$



а *б*

Рис. 4.4. Залежність вартості роботи від її тривалості

Вартість виконання усього комплексу робіт визначається виразом

$$c = \sum_{ij} (a_{ij}t_{ij} + b_{ij}). \qquad (4.5)$$

Оскільки тривалості окремих робіт можуть змінюватися в певному діапазоні, тривалість виконання усього комплексу також буде змінюватися у деякому діапазоні

$$T_{\text{мін}} \le T \le T_{\text{макс}}, \qquad (4.6)$$

де $T_{\text{макс}}$ – довжина критичного шляху сітьового графіку за умови, що $t_{ij} = T_{ij}$; $T_{\text{мін}}$ – теж саме при $t_{ij} = \tau_{ij}$.

Задача полягає у визначенні t_{ij} при заданому T, які мінімізують співвідношення (4.5).

Необхідність застосування великого числа електронних підсилювачів є суттєвим недоліком. Достатньо сказати, що для моделювання сітьових графіків, які нараховують n робіт та m подій, необхідно або $n+3(m-2)$, або $3n+2m$ підсилювачів.

Нижче показана принципова можливість використання діодно-логічних кіл для моделювання тієї ж задачі.

Для того щоб розподіл напруг у колі відповідав оптимальному розподілу тривалості робіт сітьового графіку, необхідно, щоб вартість додаткового скорочення тривалостей робіт була аналогічна потужності, яка поглинається моделлю роботи при зменшенні напруги. Ця вимога виконується, якщо вольт-амперна характеристика моделі роботи (рис. 4.5, *а*) подібна залежності інтенсивності додаткових витрат від тривалості роботи (рис. 4.4, *б*). Побідну вольт-амперну характеристику мають еквівалентні схеми

моделей робіт, які зображені на рис. 4.5, *б, в, г*. Аналогічним способом можуть бути побудовані еквівалентні схеми робіт, які мають квадратичну залежність вартості скорочення роботи від її тривалості вигляду

$$\tilde{c}_{ij} = \frac{r_{ij}}{2} t_{ij}^2 + \tilde{b}_{ij}. \qquad (4.7)$$

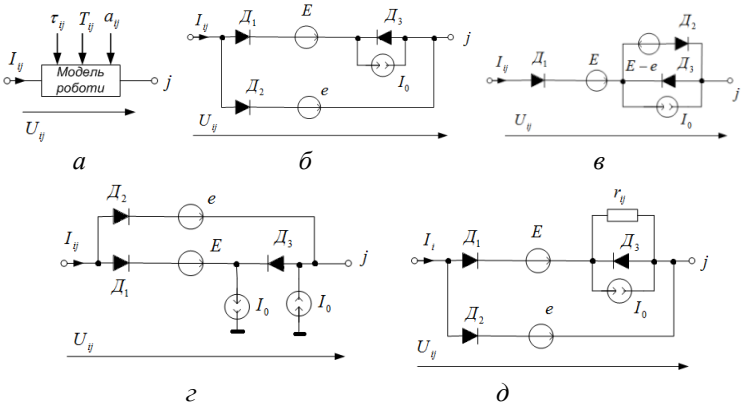


Рис. 4.5. Еквівалентні схеми моделі

Приклад такої схеми дано на рис. 4.5, *д*; відповідні ділянки характеристик показані на рис.4.4, *а, б* пунктиром.

Для отримання моделі сітьового графіку двополюсники-моделі окремих робіт з'єднуються між собою у відповідності з топологією сітьового графіку так, як це робиться і при моделюванні сітьових графіків за системою СПУ.

Оптимізацію сітьового графіку за допомогою описаної моделі можна виконувати наступним чино.

1. До полюсів моделі, які зображують початкову та кінцеву події графіку, підключається джерело струму з величиною струму, яка менша мінімального I_{0ij}. Напруга U_k^0 на полюсах цього джерела пропорційна нормальній тривалості критичного шляху сітьового графіку $T_{\text{макс}}$, яка відповідає нескороченим тривалостям окремих робіт T_{ij}. Вимір довжини критичного шляху можна виконувати також без підключення джерела струму у режимі холостого ходу моделі.

2. Замість джерела струму та моделі сітьового графіку підключається двополюсник, який складається із послідовного з'єднання джерела напруги $E_к$, ключа K та обмежувача струму на рівні I (діод з джерелом струму). Напруга джерела встановлюється рівною U_k^0 і замикається ключем K.

3. Значення $E_к$ зменшується до тих пір, поки струм I не закриє діод. При подальшому зменшенні $E_к$ напруга буде обмежуватися на рівні $\overline{U}_к \equiv T_{мін}$.

Дії, які передбачаються пунктами 1-3, дозволяють визначити інтервал зміни довжин критичного шляху $(T_{мін}, T_{макс})$ сітьового графіку.

Оптимальний розподіл тривалостей окремих робіт виходить автоматично для кожного значення $U_к$ у інтервалі $(\overline{U}_к, U_k^0)$ у силу дії принципу екстремальності потужності для електричних кіл.

Величина струму I виявляється при цьому пропорційною інтенсивності додаткових витрат, пов'язаних із скороченням критичного шляху у цілому.

Описаний спосіб моделювання PERT-Cost та еквівалентні схеми можуть бути використані при побудові відповідних моделей.

4.4. Задачі на шляхи та потоки в сітях

Теорія потоків виникла спочатку у зв'язку із розробкою методів розв'язання задач, пов'язаних із раціональним перевезенням вантажів [9]. Схема доставки вантажу представлялася у вигляді графа, по ребрам якого проходить потік цього вантажу, який необхідно максимізувати. Пізніше виявилося, що до задачі про максимальний потік зводяться і інші важливі оптимізаційні практичні задачі, такі, наприклад, як задача відшукування мінімального за вартістю виконання комплексу робіт при заданій його тривалості; задача про визначення максимальної кількості інформації, яка може бути передана по розгалуженій мережі каналів зв'язку із одного пункту у інший; задача про оптимальні призначення; різні задачі організації забезпечення; задачі автоматизації проектування вузлів обчислювальних машин; задачі. пов'язані з найбільш економічним будівництвом

енергетичних мереж нафто- та газопроводів, залізничних та шосейних доріг, іригаційних систем та багато інших прикладних задач.

Основним у теорії потоків є поняття мережі. Мережа – це зважений кінцевий граф без циклів та петель, орієнтований в одному загальному напрямку від вершини I, яка є входом (витоком) графа, до вершини S, яка є виходом (стоком) графа (рис. 4.6).

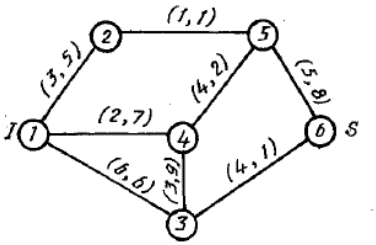


Рис. 4.6. Мережа

Для наочності будемо представляти, що по ребрам (i, j) мережі із витоку I у стік S направляється деяка речовина (вантаж, ресурс, інформація і т.п.). В теорії потоків припускається, що якщо ребро (i, j) входить у мережу, то у мережу входить і ребро (j, i). Ребрам мережі присвоюється одна або декілька числових характеристик.

Загальна кількість вершин мережі будемо позначати через n. На рис. 4.6 витоком I є вершина *1*, а стоком S – вершина *6*.

Максимальна кількість r_{ij} речовини, яку може пропустити за одиницю часу ребро (i, j), називається його пропускною спроможністю. У загальному випадку $r_{ij} \neq r_{ji}$. Якщо вершини k та l на мережі не з'єднані, то $r_{kl} = r_{lk} = 0$. На мережі (рис. 4.6) пропускні спроможності ребер вказані у дужках. При цьому перше число – це пропускна спроможність у напрямку від вершини i до вершини j, друге – у протилежному напрямку. Пропускні спроможності мережі можна задати квадратною матрицею R n-го порядку. Оскільки $r_{ii} = 0$, то на головній діагоналі цієї матриці стоять нулі.

Кількість x_{ij} речовина, яка проходить через ребро (i, j) в одиницю часу, називається потоком по ребру (i, j).

Довільно задати n^2 чисел x_{ij} $(i, j = \overline{1,n})$ неможна. Вони повинні підпорядковуватися певним обмеженням. Якщо потік із вершини i у вершину j дорівнює x_{ij}, то потік із вершини j у вершину i буде дорівнювати $-x_{ij}$, тобто

$$x_{ji} = -x_{ij}. \tag{4.8}$$

Приймається також, що $x_{ii} = 0$.

Якщо потік x_{ij} по ребру (i, j) менший його пропускної спроможності, тобто $x_{ij} < r_{ij}$, то ребро (i, j) називають ненасиченим, якщо ж $x_{ij} = r_{ij}$, – насиченим.

Сукупність $X = \{x_{ij}\}$ потоків по усім ребрам (i, j) мережі називають потоком по мережі або просто потоком.

Із фізичного смислу вантажопотоку випливає, що потік по кожному ребру (i, j) не може перевищувати його пропускну спроможність, тобто

$$x_{ij} \le r_{ij} \quad (i, j = \overline{1,n}). \tag{4.9}$$

Зрозуміло також, що для будь-якої вершини, крім витоку I та стоку S, кількість речовини, яка поступає у цю вершину, дорівнює кількості речовини, яка витікає із неї. Із врахуванням погодження (4.8) цю вимогу можна виразити записом

$$\sum_{j=1}^{n} x_{ij} = 0 \qquad (i \ne I, S). \tag{4.10}$$

Це обмеження називають умовою збереження потоку: у проміжних вершинах потоки не створюються і не зникають. Звідси випливає, що загальна кількість речовини, яка витікає із витоку I, співпадає із загальною кількістю речовини, яка поступає у стік S, тобто

$$f = \sum_j x_{Ii} = \sum_i x_{iS}, \tag{4.11}$$

де j – кінцеві вершини ребер, які виходять із I; i – початкові вершини ребер, які входять у S.

Лінійну функцію f називають потужністю потоку на мережі.

Враховуюче сказане, задачу про максимальний потік можна сформулювати наступним чином: знайти сукупність $X^* = x_{ij}^*$ потоків x_{ij}^* по усім ребрам (i, j) мережі, яка задовольняє умовам (4.8) – (4.10) та максимізує лінійну функцію (4.11). Це типова задача лінійного програмування.

Відмітимо, що числа x_{ij} утворюють квадратну матрицю n-го порядку. На головній діагоналі якої стоять нулі ($x_{ij} = 0$), а елементи, розташовані симетрично головній діагоналі, рівні за абсолютною величиною та протилежні за знаком. Звідси випливає, що задати потік $X = x_{ij}$ на мережі – це значить задати n^2 чисел x_{ij}, які задовольняють умовам (4.8) – (4.10).

4.5. Екстремальні задачі на мережах і графах

Однією з найважливіших задач лінійного програмування є так звана транспортна задача, або задача про оптимальний розподіл потоку у мережі [3]. У загальному вигляді вона формулюється наступним чином.

Є деяке коло, яке складається із вузлів та гілок, і потік деяких однорідних величин, який розподіляється у цьому колі (рис. 4.7). Коло може представляти собою мережу магістральних доріг, систему зв'язку, енергосистему і т.п. Тоді величини, які складають потік, будуть представлені відповідно у вигляді транспорту, повідомлень, електричного струму і т.п. Кожний вузол кола підлягає закону неперервності потоку. Кожна гілка характеризується вартістю одиниці потоку у гілці, направленого із вузла i у вузол j, і пропускною спроможністю у цьому напрямку. Необхідно знайти такий розподіл потоку, який забезпечує мінімум загальної вартості при виконанні накладених обмежень.

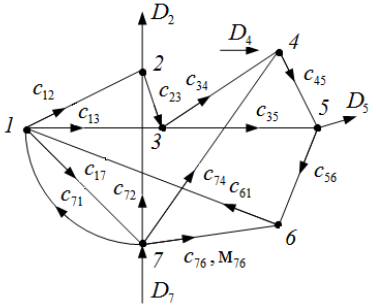


Рис. 4.7. Коло

Математично задача формулюється так. Мінімізувати

$$\mu = \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

при обмеженнях

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

$$0 \le x_{ij} \le M_{ij} \,,$$

де x_{ij} – величина потоку у гілці ij, направленого із вузла i у вузол j; D_i – величина потоку, який вводять у коло ззовні через вузол i; M_{ij} – пропускна спроможність гілки.

Як видно з постановки задачі, гілки можуть бути як направленими, так і ненаправленими (в останньому випадку вузли i та j з'єднуються двома направленими гілками – ij та ji – з відповідними пропускними спроможностями.

Окремими випадками розглянутої задачі є задачі про максимальні потоки у мережі, про найкоротший та найдовший шляхи через мережу, транспортна задача у матричній постановці та її різновиди.

Задача про максимальний потік у мережі полягає у наступному. Задана мережа, яка містить початковий та кінцевий вузли (рис. 4.8). Кожна гілка характеризується відповідною пропускною спроможністю. Гілки мережі можуть бути

направленими або ненаправленими. Система підкоряється закону про неперервність потоку у мережі.

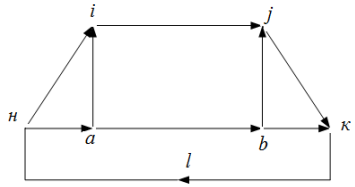


Рис. 4.8. Мережа

У математичній формі задача записується наступним чином.

Максимізувати Y за умов

$$\sum_{j=1}^{n} x_{ij} - \sum_{j=1}^{n} x_{ji} = \begin{cases} Y & \text{при} \quad i = \text{н}, \\ -Y & \text{при} \quad i = \text{к}, \\ 0 & \text{при} \quad i \neq \text{н}, \text{к}. \end{cases}$$

$$0 \le x_{ij} \le M_{ij}.$$

Тут Y – сумарний потік, який проходить через мережу.

Розв'язок задачі про максимальний потік може бути реалізований у поняттях перетинів.

Перетином називається сукупність гілок, віддалення яких веде до розділення графу на дві частини або більше, причому початковий та кінцевий вузли розташовані у різних частинах. Наприклад, сукупність перетинів $(i, j), (a, b)$ або $(\text{н}, i), (\text{н}, a), (a, i)$ і т.д.

Перетин називається простим, якщо він не володіє надлишковістю, тобто якщо із сукупності гілок, утворюючих простий перетин, видалити гілку, то дана сукупність гілок перестає бути перетином.

У наведених вище прикладах перший перетин простий, а другий надлишковий, так як сукупність гілок $(\text{н}, i), (\text{н}, a)$ також утворює перетин.

Пропускна спроможність перетину дорівнює сумі пропускних спроможностей гілок даного перетину. Пропускна спроможність гілок враховується тільки у напрямку, який відповідає напрямку потоку.

Основна теорема про максимальний потік полягає у наступному. Максимальний потік зліва направо через мережу дорівнює мінімальному значенню пропускної спроможності усіх простих перетинів. Або скорочено – максимальний потік дорівнює мінімальному перетину.

До задач про найкоротший та найдовший шляхи зводяться задачі сітьового планування.

Питання для самоконтролю

1. Визначення поняття графа.
2. Сутність задачі найкоротшого шляху.
3. Види моделей задач найкоротшого шляху.
4. Електронна модель сітьового графу.

Ілюстративні приклади до частини 2

Приклад 1. Мінімізувати функцію із врахуванням обмежень на змінні у програмному середовищі Mathematica:

$$\mu = x_1 + x_2 + x_3 \to \min,$$
$$2x_1 + 3x2 + x3 = 8,$$
$$x_1 + 2x_2 + 2x_3 = 5,$$
$$x_1, x_2, x_3 \geq 0$$

У програмі Mathematica для мінімізації функції з обмеженнями використовується вбудована функція `NMinimize[{}, {}]`, де у першій фігурній дужці вказується функція, яку необхідно мінімізувати, та обмеження, а у другій – змінні. Результати обчислень:

```
In[1]:= NMinimize[{x1 + x2 + x3, 2 x1 + 3 x2 + x3 == 8, x1 + 2 x2 + 2 x3 == 5, x1 ≥ 0,
          x2 ≥ 0, x3 ≥ 0}, {x1, x2, x3}]

Out[1]= {3., {x1 → 1., x2 → 2., x3 → 0.}}
```

Приклад 2. Знайти максимум цільової функції при заданих умовах:

$$\mu = -2x_1 - 3x_2 + 2x_4 + 3x_5 - x_1 - x_8 + 2x_9 - x_{10} - \\ - 3x_{11} + 2x_{12} - x_{13} + x_{14} + 2x_{15},$$

$$2x_1 - x_2 + x_3 + x_6 + x_7 + x_9 - x_{10} - 2x_{11} - x_{12} +$$
$$+ 2x_{13} + x_{15} + x_{16} = 12,$$
$$x_1 + x_4 - x_6 + x_7 + 2x_8 + x_{10} + x_{11} + x_{12} + x_{14} + x_{17} = 5,$$
$$2x_1 + 2x_2 + x_3 - 2x_5 + x_6 + 2x_7 - x_8 + x_9 - 2x_{10} +$$
$$+ 2x_{12} - 2x_{13} + 2x_{15} + x_{18} = 20,$$
$$x_1 - x_2 - 2x_3 + 2x_4 - 2x_5 + x_6 - x_8 + 2x_{10} +$$
$$+ 2x_{11} - x_{12} + 2x_{13} + x_{19} = 10,$$
$$-2x_1 + 2x_2 - 2x_3 - 2x_4 + x_5 + x_6 + x_7 + x_8 + x_{10} +$$
$$+ 2x_{11} + 2x_{12} + x_{14} - 2x_{15} + x_{20} = 24,$$
$$x_j \geq 0.$$

Вирішення задачі у програмному середовищі Mathematica:

```
In[1]:= NMaximize[
        {-2 x1 - 3 x2 + 2 x4 + 3 x5 - x1 - x8 + 2 x9 - x10 - 3 x11 + 2 x12 - x13 + x14 + 2 x15,
         2 x1 - x2 + x3 + x6 + x7 + x9 - x10 - 2 x11 - x12 + 2 x13 + x15 + x16 == 12,
         x1 + x4 - x6 + x7 + 2 x8 + x10 + x11 + x12 + x14 + x17 == 5,
         2 x1 + 2 x2 + x3 - 2 x5 + x6 + 2 x7 - x8 + x9 - 2 x10 + 2 x12 - 2 x13 + 2 x15 + x18 == 20,
         x1 - x2 - 2 x3 + 2 x4 - 2 x5 + x6 - x8 + 2 x10 + 2 x11 - x12 + 2 x13 + x19 == 10,
         -2 x1 + 2 x2 - 2 x3 - 2 x4 + x5 + x6 + x7 + x8 + x10 + 2 x11 + 2 x12 + x14 -
           2 x15 + x20 == 24, x1 ≥ 0, x2 ≥ 0, x3 ≥ 0, x4 ≥ 0, x5 ≥ 0, x6 ≥ 0,
         x7 ≥ 0, x8 ≥ 0, x9 ≥ 0, x10 ≥ 0, x11 ≥ 0, x12 ≥ 0, x13 ≥ 0, x14 ≥ 0,
         x15 ≥ 0, x16 ≥ 0, x17 ≥ 0, x18 ≥ 0, x19 ≥ 0, x20 ≥ 0},
        {x1, x2, x3, x4, x5, x6, x7, x8, x9, x10, x11, x12, x13, x14, x15,
         x16, x17, x18, x19, x20}]

Out[1]= {208., {x1 → 0., x10 → 0., x11 → 0., x12 → 0., x13 → 0., x14 → 0., x15 → 12.,
         x16 → 0., x17 → 0., x18 → 112., x19 → 116., x2 → 0., x20 → 0.,
         x3 → 0., x4 → 5., x5 → 58., x6 → 0., x7 → 0., x8 → 0., x9 → 0.}}
```

Приклад 3. Розв'язати транспортну задачу лінійного програмування у середовищі Multisim за допомогою діодної аналогії Денніса та порівняти результати з аналітичними результатами.

Умови задачі задані у табл. 1.

Таблиця 1

a_i	b_j						
	52	38	55	91	69	84	61
100	$c_{11}=15$	$c_{12}=25$	$c_{13}=18$	$c_{14}=35$	$c_{15}=40$	$c_{16}=23$	$c_{17}=0$
120	$c_{21}=22$	$c_{22}=36$	$c_{23}=40$	$c_{24}=60$	$c_{25}=50$	$c_{26}=38$	$c_{27}=0$
150	$c_{31}=26$	$c_{32}=38$	$c_{33}=45$	$c_{34}=52$	$c_{35}=45$	$c_{36}=48$	$c_{37}=0$
80	$c_{41}=18$	$c_{42}=24$	$c_{43}=32$	$c_{44}=40$	$c_{45}=35$	$c_{46}=26$	$c_{47}=0$

a_i – кількість одиниць однорідного продукту, виготовленому на i-му пункті виробництва;

b_j – кількість одиниць цього продукту, яка необхідна j-му пункту споживання;

c_{ij} – вартості перевезення одиниці продукту з i-го пункту виробництва у j-й пункт споживання.

На рис. 1 наведена віртуальна схема моделі транспортної задачі.

Знаходження аналітичного розв'язку задачі у програмному середовищі Mathematica:

```
In[2]:= NMinimize[
        {15 x11 + 25 x12 + 18 x13 + 35 x14 + 40 x15 + 23 x16 + 22 x21 + 36 x22 + 40 x23 + 60 x24 + 50 x25 +
          38 x26 + 26 x31 + 38 x32 + 45 x33 + 52 x34 + 45 x35 + 48 x36 + 18 x41 + 24 x42 + 32 x43 + 40 x44 +
          35 x45 + 26 x46, x11 + x12 + x13 + x14 + x15 + x16 + x17 == 100,
         x21 + x22 + x23 + x24 + x25 + x26 + x27 == 120, x31 + x32 + x33 + x34 + x35 + x36 + x37 == 150,
         x41 + x42 + x43 + x44 + x45 + x46 + x47 == 80, x11 + x21 + x31 + x41 == 52, x12 + x22 + x32 + x42 == 38,
         x13 + x23 + x33 + x43 == 55, x14 + x24 + x34 + x44 == 91, x15 + x25 + x35 + x45 == 69,
         x16 + x26 + x36 + x46 == 84, x17 + x27 + x37 + x47 == 61, x11 ≥ 0, x12 ≥ 0, x13 ≥ 0, x14 ≥ 0,
         x15 ≥ 0, x16 ≥ 0, x17 ≥ 0, x21 ≥ 0, x22 ≥ 0, x23 ≥ 0, x24 ≥ 0, x25 ≥ 0, x26 ≥ 0, x27 ≥ 0,
         x31 ≥ 0, x32 ≥ 0, x33 ≥ 0, x34 ≥ 0, x35 ≥ 0, x36 ≥ 0, x37 ≥ 0, x41 ≥ 0, x42 ≥ 0, x43 ≥ 0,
         x44 ≥ 0, x45 ≥ 0, x46 ≥ 0, x47 ≥ 0},
        {x11, x12, x13, x14, x15, x16, x17, x21, x22, x23, x24, x25, x26, x27, x31, x32, x33,
         x34, x35, x36, x37, x41, x42, x43, x44, x45, x46, x47}]

Out[2]= {12806., {x11 → 0., x12 → 0., x13 → 55., x14 → 45., x15 → 0., x16 → 0., x17 → 0., x21 → 52., x22 → 0.,
         x23 → 0., x24 → 0., x25 → 0., x26 → 68., x27 → 0., x31 → 0., x32 → 0., x33 → 0., x34 → 20., x35 → 69.,
         x36 → 0., x37 → 61., x41 → 0., x42 → 38., x43 → 0., x44 → 26., x45 → 0., x46 → 16., x47 → 0.}}
```

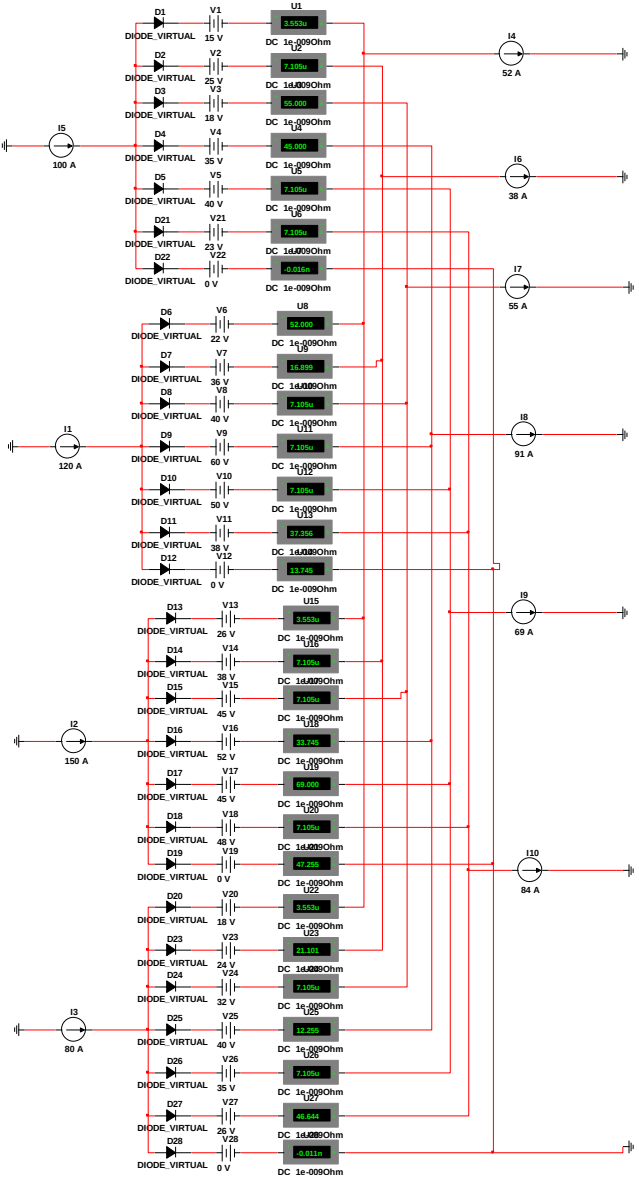


Рис. 1. Віртуальна модель задачі

У табл. 2 наведені результати розрахунків. Над рискою показані значення розрахунку у програмному середовищі Mathematica, а під рискою – значення, отримані за допомогою діодної аналогії Денніса.

Таблиця 2

a_i	b_j						
	52	38	55	91	69	84	61
100	0/0	0/0	55/55	45/45	0/0	0/0	0/0
120	52/52	0/17	0/0	0/0	0/0	68/37	0/14
150	0/0	0/0	0/0	20/33	69/69	0/0	61/47
80	0/0	38/21	0/0	26/12	0/0	16/46	0/0

ДОДАТОК 1. Основи роботи з програмним середовищем Mathematica.

Система Mathematica фірми Wolfram Research Inc. це потужна універсальна система комп'ютерної алгебри, яка орієнтована на символічні обчислення [16, 18]. При цьому вона чудово доводить до числового результату найскладніші обчислення.

Система Mathematica є однією із універсальних математичних систем, яка дає можливість вирішувати велику кількість складних задач. Сильною стороною системи є двох та трьохмірна графіка, яка застосовується для візуалізації кривих та поверхонь. Система інтерактивна, тобто працює у режимі постійного діалогу із користувачем). Вона гнучка та універсальна. Арифметичні дії у Mathematica проводяться так само, як у електронному калькуляторі. Можна проводити точні обчислення, а можна наближені з будь-якою степеню точністю.

Усі записи, які задаються у вікні системи Mathematica (побудувати графік, записати функція, розв'язати задачу), називаються виразами.

У системі Mathematica є область задачі, яка позначається на екрані In[ ]:, та область отримання розв'язку задачі, яка позначається Out[ ].

Основними символами, які використовуються при роботі з Mathematica, є такі:

+ – додавання;

− – віднімання;

* (або прогалина між співмножниками) – скалярне множення;

. – векторно-матричне множення;

/ – ділення;

^ – піднесення до степеня;

() – круглі дужки використовуються для зміни природного порядку виконання операцій;

[] – квадратні дужки використовуються тільки для введення аргументів функцій та інших виразів;

{} – фігурні дужки використовуються для задання векторів, списків, множин та інших багатоелементних об'єктів;

{{}} – подвійні фігурні дужки використовуються для задання матриць;

a = b – команда присвоєння значення b величині a;

y := f[x] – команда визначення функції або виразу;

== - задання рівняння;

$f /. x \to w$ – команда заміни у виразі f величину x на w;

Solve, Integrate і т.д. – вбудовані функції і вирази починаються з великої літери;

FindRoot, ParametricPlot і т.д. – в складних вбудованих функціях та виразах складові частини виразів починаються з великих літер, а частини вводяться без прогалин між ними;

; – крапка з комою, яка слідує за введеним визначенням функції або виразу, запобігає виведення результату, який може бути проміжним або громіздким .

Для виконання програми або її частини, використовується команда Shift+Enter.

Початкові навички роботи у системі можна отримати з електронного помічника (Help).

ДОДАТОК 2. Сучасні програмні засоби візуального (графічного) програмування MatLab/Simulink

Програма Simulink є додатком до пакету MatLab [11, 15]. У певному смислі Simulink можна розглядати як самостійний продукт фірми MathWorks, однак він працює тільки при наявності ядра MatLab та використовує багато функцій, які входять до його складу.

Необхідно відмітити, що пакет MatLab орієнтований у першу чергу на обробку масивів даних (матриць, векторів і т.п.). Це дозволяє суттєво підвищити ефективність процедур, які працюють із вказаними типами даних, у порівнянні з мовами програмування «загального призначення», та суттєво відрізняє MatLab від інших систем, таких як Maple, MathCAD, Mathematica. Векторна обробка даних забезпечує високу швидкість обчислень, у більшості випадків позбавляє користувача від написання циклів та гарантує необхідну точність.

Додаток Simulink є інструментом, за допомогою якого можна об'єднувати блоки, які відповідають окремим елементам динамічної системи у єдине ціле та вивчати їх поведінку у часі.

Розробка моделей засобами Simulink (S-моделі) основана на технології drag-and-drop («перетягнути та вставити»). Для побудови S-моделі використовуються модулі (або блоки), які зберігаються у бібліотеці Simulink.

Бібліотека Simulink добра тим. що , з однієї сторони, забезпечує користувачу доступ до усіх основних можливостей пакету MatLab, а з іншої – є достатньо самостійною його компонентою, у тому розумінні, що при роботі з нею не обов'язково мати навички у використанні інших інструментів, які входять у склад пакету.

Блоки. Які включаються у модель, яка створюється, можуть бути пов'язані один з іншим як за інформацією, так і за управлінням. Дані, якими обмінюються блоки, можуть бути скалярними величинами, векторами або матрицями довільної розмірності.

Будь-яка S-модель може мати ієрархічну структуру, тобто складатися із моделей більш низького рівня, причому число рівнів ієрархії практично не обмежене. Поряд з іншими параметрами

моделювання користувач може задавати спосіб зміни модельного часу (з постійним або змінним кроком), а також умови закінчення моделювання.

У ході моделювання є можливість спостерігати за процесами, які відбуваються у системі. Для цього використовуються спеціальні «вікна спостереження», які входять у склад бібліотеки Simulink.

Характеристики, які цікавлять користувача, можуть бути представлені як у числовій, так і у графічній формі.

У складі MatLab є багато інших додатків, основаних на методах графічного (візуального) програмування, які допускають спільну роботу з додатком Simulink.

Aerospace Blockset – містить спеціальні інструменти для моделювання авіаційних, космічних, реактивних та турбореактивних систем.

DSP Blockset – призначений для проектування систем та моделювання задач цифрової обробки сигналів.

Nonlinear Control Design Blockset – надає у розпорядження користувача графічний інтерфейс для налагодження параметрів динамічних об'єктів.

SimPowerSystems – призначений для моделювання електротехнічних та електроенергетичних пристроїв та систем.

SimMechanics – дозволяє моделювати системи управління за допомогою ненаправлених сигнальних графів, об'єднувати їх за фізичними моделями та моделями із інших бібліотек.

Пакет Simulink запускається із програмного середовища MatLab. Відповідна піктограма розташована на панелі інструментів.

Для початку роботи необхідно створити новий файл *.mdl , у який можна за допомогою миші перетаскувати блоки із бібліотеки Simulink.

У головному меню знаходиться вкладка Simulation/Configuration Parameters, де можна змінювати параметри процесу моделювання, таки як час початку та кінець моделювання, вибір розв'язувача, крок з яким відбувається розрахунок і т.д.

Розглянемо блоки Simulink, які часто зустрічаються.

Для будь-якого блоку за подвійним натисненням кнопки миші відкривається вікно параметрів Function Block Parameters, де можна змінювати установки.

Джерела сигналів (Sources). На рис. 1 представлені деякі джерела сигналів.

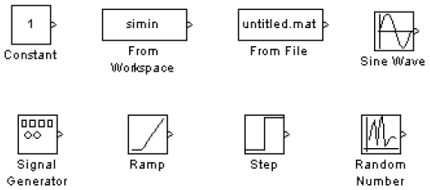


Рис. 1. Джерела сигналів

Constsnt – задає постійний за рівнем сигнал;

From Workspace – блок, який зраховує дані із робочої області;

From File – блок, який зраховує дані із файлу;

Sine Wave – формує синусоїдальний сигнал із заданою частотою, амплітудою, фазою та зміщенням;

Signal Generator – формує один із чотирьох видів періодичних сигналів (синусоїдальний, прямокутний, пилоподібний, випадковий сигнали);

Ramp – формує лінійний сигнал;

Step – формує ступінчатий сигнал;

Random Number – формує випадковий сигнал з нормальним розподілом рівня сигналу.

Приймачі сигналів (Sinks). На рис. 2 представлені блоки-приймачі сигналів.

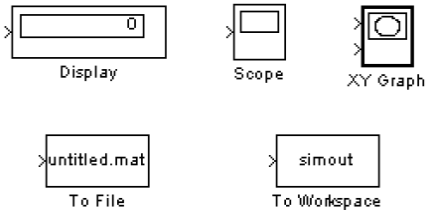


Рис. 2. Приймачі сигналів

Цифровий дисплей (Display) відображає значення сигналу у вигляді числа;

Віртуальний осцилограф (Scope) будує графік сигналу від функції часу;

Віртуальний графобудівник (XY Graph), будує графік одного сигналу у залежності від іншого (графік вигляду Y(X));

Блок «To File» записує дані, які поступають на його вхід, у файл;

Блок «To Workspace» записує дані, які поступають на його вхід, у робочу область MatLab.

Блоки математичних операцій (рис. 3).

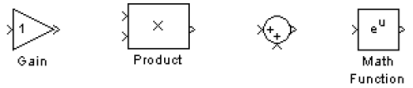


Рис. 3. Блоки математичних операцій

Блок Gain множить вхідний сигнал на заданий коефіцієнт;

Блок Product перемножує два (або більше) сигналів;

Блок Sum додає два (або більше) сигналів;

Блок Math Function перетворює вхідний сигнал за заданою із списку функцій (наприклад, підведення у квадрат, корінь квадратний, логарифм і т.д.).

Відмітимо ще деякі блоки, які часто зустрічаються (рис. 4).

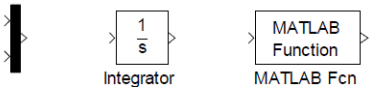


Рис. 4. Приклади блоків функцій, які часто зустрічаються

Блок Mux (мультиплексом) об’єднує сигнали;

Блок Integrator інтегрує сигнал;

Блок MATLAB Fcn формує функцію, задану користувачем на мові MatLab.

ДОДАТОК 3. Сучасні програмні засоби візуального (графічного) програмування Multisim

Програма Multisim компанії Electronics Workbench – це програма для моделювання електричних схем [17]. Multisim є інтерактивним емулятором схем

Multisim може проводити аналіз схем на постійному та змінному струмах. При аналізі на постійному струмі визначається робоча точка схеми в режимі роботи, який установився. Результати цього аналізу не відображаються на приладах, вони використовуються для подальшого аналізу схеми. Аналіз на змінному струмі використовує результати аналізу на постійному струмі для отримання лінеаризованих моделей нелінійних компонентів. Аналіз схем у режимі AC може проводитися як у часовій, так і у частотній областях.

В Multisim можна досліджувати перехідні процеси при впливі на схеми вхідних сигналів різної форми. Програма також дозволяє проводити аналіз цифро-аналогових та цифрових схем великої степені складності. Бібліотеки, які є у програмі, включають у себе великий набір широко розповсюджених електронних компонентів. Є можливість підключення та створення нових бібліотек компонентів.

Широкий набір приладів дозволяє виконувати виміри різних величин, задавати вхідні впливи, будувати графіки. Всі прилади зображуються у вигляді, максимально наближеному до реального, тому працювати з ними просто та зручно.

У бібліотеці компонентів програми входять пасивні елементи, індикатори, логічні елементи, тригерні пристрої, цифрові та аналогові елементи, спеціальні комбінаційні та послідовні схеми. Активні елементи можуть бути представлені моделями як ідеальних, так і реальних елементів. Можливе також створення своїх моделей елементів та додавання їх у бібліотеки елементів.

У програмі використовується великий набір приладів для проведення вимірів: амперметр, вольтметр, осцилограф, мультиметр, Боде-плотер (будівник графів частотних характеристик схем), функціональний генератор, генератор слів, логічний аналізатор та логічний перетворювач.

Multisim дозволяє будувати схеми різної степені складності за допомогою наступних операцій:

− вибір елементів та приладів з бібліотек;

− переміщення елементів та схем у будь-яке місце робочого поля;

− поворот елементів та груп елементів на кути, кратні 90°;

− копіювання, вставка чи видалення елементів, груп елементів, фрагментів схем та цілих схем;

− вимір кольору провідників;

− виділення кольором контурів схем для більш зручного сприйняття;

− одночасне підключення декількох вимірювальних пристроїв та спостереження їх показів на екрані монітору;

− присвоєння елементу умовного позначення;

− вимір параметрів елементів у широкому діапазоні.

Шляхом настроювання приладів можна:

− вимірювати шкали приладів у залежності від діапазону вимірювань;

− задавати режим роботи приладу;

− задавати вид вхідних впливів на схему (постійні та гармонічні струми, трикутні та прямокутні імпульси).

Графічні можливості програми дозволяють:

− одночасно спостерігати декілька кривих на графіку;

− відображати криву на графіках різними кольорами;

− вимірювати координати точок на графіку.

На рис. 1 зображене вікно програми Multisim.

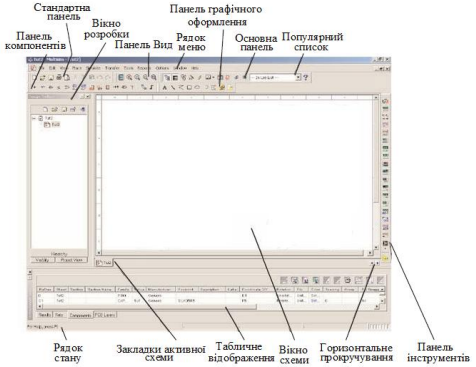


Рис. 1. Середовище програми Multisim

На панелі компонентів обираються необхідні компоненти (реальні або ідеальні) і за допомогою натиску на них відображаються у робочій області. З'єднати елементи можна, якщо виділити затиск одного елемента та не відпускаючи клавіші миші, з'єднати цей затиск із затиском необхідного елементу. На рис. 2 показаний приклад з'єднання двох елементів.

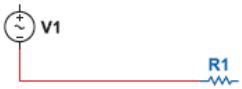


Рис. 2. Приклад з'єднання двох елементів

Для вимірювання необхідних параметрів, на панелі інструментів можна обрати певний вимірюючий прилад (осцилограф, Боде плоттер, мультиметр та ін.).

Процес моделювання починається натисненням на клавішу вимикача  .

Список літератури до частини 2

1. *Акулич И.Л.* Математическое программирование в примерах и задачах: Учеб. Пособие для студентов эконом. спец. вузов. – М.: Высшая школа, 1986. – 319 с., ил.

2. *Аттетков А.В., Галкин С.В., Зарубин В.С.* Методы оптимизации: Учеб. для вузов / Под ред. В.С. Зарубина, А.П. Крищенко. – 2-е изд., стереотип. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2003. – 440 с. ISBN 5-7038-1270-4.

3. *Васильев В.В., Клепикова А.Н., Тимошенко А.Г.* Решение задач оптимального планирования на электронных моделях. – К.: Наукова думка. 1966. – 166 с.

4. *Гилл Ф., Мюррей У., Райт М.* Практическая оптимизация. Пер. с англ. – М.: Мир, 1985. – 509 с.

5. *Карманов В.Г.* Математическое программирование: Учеб. Пособие. – 5-е изд., стереотип. – М.: ФИЗМАТЛИТ, 2004. – 264 с.

6. Конспект лекцій з дисципліни «Економіко-математичне моделювання» (для студентів 3 курсу заочної форми навчання за напрямком підготовки 0501 (6.030509) «Облік і аудит») / Авт. К.А. Мамонов.: Харк. нац. акад. міськ. госп-ва. – Х.: ХНАМГ, 2009. – 86 с.

7. *Кормен Т. Х., Лейзерсон Ч. И., Ривест Р. Л., Штайн К.* Алгоритмы: построение и анализ. – 2-е изд. – М.: «Вильямс». 2006. – 1296. ISBN 0-07-013151-1.

8. *Корн Г., Корн Т.* Справочник по математике для научных работников и инженеров. – М.: Наука, 1977. – 831 с.

9. *Кузнецов А.В. и др.* Высшая математика: Мат. программир.: Учеб. / А.В. Кузнецов. В.А. Сакович, Н.И. Холод; Под общ. ред. А.В. Кузнецова. – Мн.: Высш. шк., 1994. – 286 с.: ил.

10. *Левитин А. В.* Алгоритмы: введение в разработку и анализ. – М.: «Вильямс», 2006. – 576 с., с ил. ISBN 5-8459-0987-2, 0-201-74395-7.

11. Математическое и компьютерное моделирование прицессов и систем в среде MATLAB/SIMULINK. Учебное пособие для студентов и аспирантов / В.В. Васильев, Л.А. Симак, А.М. Рыбникова. – К.: НАН Украины, 2008. – 91 с.

12. *Мудров В.И.* Задача о коммивояжере. – М.: «Знание», 1969. – 62 с.

13. *Плотников А.Д.* Математическое программирование: экспресс-курс. – 2006. – 171 с. ISBN 985-475-186-4.

14. *Пухов Г.Е., Васильев В.В.* Теория электронного моделирования и математические машины непрерывного действия. – К.: КИИГА, 1971. – 110 с.

15. *Терёхин В.В.* Моделирование в системе MATLAB: Учебное пособие / Кемеровский государственный университет. – Новокузнецк: Кузбассвузиздат, 2004. – 376 с.

16. *Шмидский Я.К.* Mathematica 5. Самоучитель.: Издательский дом «Вильямс», 2004. – 592 с.

17. Electronics Workbench and Multisim. Введение в Multisim // ni.com./ Russia. – National Instruments, 2006. – 44 с. http://digital/ni/com/worldwide/russia.nsf/.../Multisim%20Getting%20Started.pdf.

18. *Stephen Wolfram.* The Mathematica Book. – Wolfram Media & Cambridge University Press, 1999. – 1470 p.