МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ДВНЗ «УЖГОРОДСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ»
Факультет інформаційних технологій
Кафедра інформаційних управляючих систем та технологій

МЕРЕЖЕВІ ЗАДАЧІ ОПТИМІЗАЦІЇ: МОДЕЛІ ТА ПРОГРАМИ

Методичні рекомендації

УЖГОРОД – 2024

Мережеві задачі оптимізації: моделі та програми. Методичні рекомендації для студентів і аспірантів, які навчаються за спеціальностями галузі знань 12 «Інформаційні технології».

У методичних рекомендаціях розглянуто спосіб розв'язання задач математичного програмування за допомогою мережевого NEOS-сервера та мови моделювання AMPL. Наведено приклади задач лінійного, цілочислового та нелінійного програмування. Значна увага приділена транспортній задачі. Розглянуті приклади можуть бути основою для розробки студентами власних програмних проєктів, пов'язаних з оптимізацією.

Розробники: Стецюк П.І., Міца О.В., Стовба В.О.

Рецензенти:

– Головач Й.І., д.т.н., професор, професор Закарпатського угорського інституту імені Ференца Ракоці II;

– Гече Ф.Е., д.т.н., професор, професор ДВНЗ «Ужгородський національний університет».

Рекомендовано до друку науково-методичною комісією факультету інформаційних технологій ДВНЗ «Ужгородський національний університет» (протокол № 1 від 27.08.2024 р.)

ЗМІСТ

ВСТУП

Навчальна дисципліна «Мережеві інформаційні технології» є обов'язковою компонентою освітньо-професійної програми підготовки фахівців за освітньо-кваліфікаційним рівнем «магістр» спеціальності 122 – Комп'ютерні науки в ДВНЗ «Ужгородський національний університет».

Метою дисципліни «Мережні інформаційні технології» є навчити студентів способам опису задач математичного програмування на мові моделювання AMPL та дистанційному способу розв'язання транспортних задач, задач лінійного та нелінійного програмування за допомогою програм NEOS-сервера.

Завдання навчальної дисципліни «Мережні інформаційні технології» є розвиток аналітичного мислення у студентів, вміння застосовувати одержані знання в процесі побудови та аналізу алгоритмів розв'язання екстремальних задач.

За результатами проведення лекцій з курсу «Мережні інформаційні технології» студенти повинні:

Знати – загальні формулювання задач лінійного та нелінійного програмування, способи їх опису за допомогою мови моделювання AMPL, найбільш відомі програми для розв'язання задач лінійного програмування, задач цілочислового лінійного програмування та задач нелінійного програмування.

Вміти – надсилати AMPL-коди для задач математичного програмування на відповідні програми із NEOS-солвера, слідкувати за процесом розв'язання задач, отримувати та аналізувати розв'язки, знайдені програмами Gurobi та MINOS.

ПРОГРАМА НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Тема 1. NEOS-сервер та NEOS-солвер як інтерфейс для розв'язання оптимізаційних задач. (2 год)

1. NEOS-сервер, його призначення та основні компоненти.
2. Ознайомлення з NEOS-солвером та його оптимізаційними програмами.

Література: [1], [2], [16].

Тема 2. Мова моделювання AMPL як засіб опису задач математичного програмування. (2 год)

1. Ознайомлення з мовою моделювання AMPL.
2. Основні групи операторів для опису задач математичного програмування в AMPL (param, var, maximize, minimize, subject to, data, solve, display).

Література: [3], [4].

Тема 3. Задачі лінійного програмування (ЛП) та їх розв'язання за допомогою NEOS-солвера. (2 год)

1. Задача лінійного програмування. Її формулювання. Симплекс метод та методи внутрішніх точок.
2. Програми NEOS-солвера для розв'язання ЛП-задач.

Література: [5], [6], [2].

Тема 4. Програма Gurobi як сучасний засіб розв'язання ЛП-задач великих розмірів. (2 год)

1. Програма Gurobi як сучасний засіб розв'язання ЛП-задач великих розмірів.
2. Опис роботи програми Gurobi для структурованих ЛП-задач з сотнями тисяч змінних та обмежень.

Література: [7], [2].

Тема 5. Задачі цілочислового лінійного програмування та програми NEOS-солвера. (2 год)

1. Задача цілочислового лінійного програмування. Типи змінних integer та boolean в AMPL.

2. Програми NEOS-солвера для задач цілочислового лінійного програмування.

Література: [3], [11].

Тема 6. Задачі нелінійного програмування та їх розв’язання за допомогою NEOS-солвера. (2 год)

1. Задача нелінійного програмування. Її формулювання.
2. Програми NEOS-солвера для розв'язання задач нелінійного програмування.

Література: [9], [2].

Тема 7. Програма MINOS – засіб розв’язання задач лінійного та нелінійного програмування. (2 год)

1. Програма MINOS для розв'язання задач нелінійного програмування.
2. Використання програми MINOS для задач великих розмірів.

Література: [8], [2].

Тема 8. Транспортна матрична задача та її AMPL-код. (2 год)

1. Транспортна задача. Метод потенціалів.
2. Можливості AMPL для опису транспортних задач.

Література: [3], [8], [10], [11].

Тема 9. Двоетапна транспортна задача та її AMPL-реалізація. (2 год)

1. Формулювання задачі та її властивості.
2. AMPL-реалізація задачі та тестовий приклад.
3. Квадратична двоетапна транспортна задача.

Література: [13], [14], [15].

Тема 10. Дві ЛП-задачі для відмовостійкої мережі. (2 год)

1. Дві ЛП-задачі з булевими змінними.
2. Тестові розрахунки за допомогою програми Gurobi.

Література: [11], [12].

I. NEOS-сервер та NEOS-солвер як інтерфейс для розв'язання оптимізаційних задач

NEOS (Network-Enabled Optimization System) сервер – це клієнт-серверна програма в Інтернеті, яка надає безкоштовний доступ до бібліотеки оптимізаційних солверів (програм для розв'язання задач математичного програмування онлайн). Для розв'язання задачі за допомогою цього сервера математичну модель задачі необхідно описати однією з так званих мов моделювання, найбільш популярними серед яких є GAMS (https://www.gams.com/) та AMPL (https://ampl.com/), і передати її в роботу через Web-інтерфейс одному з NEOS-солверів.

Переходимо за посиланням http://www.neos-server.org/neos/ і отримуємо стартову сторінку NEOS-сервера (рис. 1.1.). Натиснувши на посилання «Submit a job to NEOS», переходимо до переліку типів задач математичного програмування **(**рис. 1.2.), які можна розвязувати з використанням низки солверів, доступних на вкладці «Solvers».

Рис. 1.1. Стартова сторінка NEOS-сервера

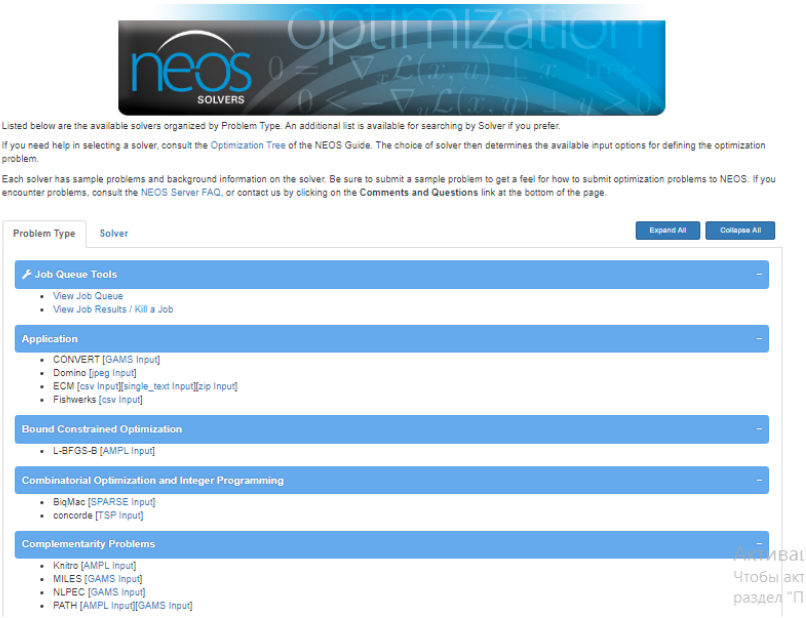

Рис. 1.2. Перелік задач математичного програмування, які можна розв'язувати на NEOS-сервері

За допомогою NEOS-солверів можна розв'язувати різноманітні оптимізаційні задачі: задачі лінійного програмування, задачі цілочислового лінійного програмування, задачі нелінійної оптимізації з обмеженнями та багато інших. Наприклад, для розв'язання задач *лінійного* програмування використовуються такі програми:

- Clp [MPS]
- COPT [AMPL] [GAMS] [LP] [MPS] [NL]
- CPLEX [AMPL] [GAMS] [LP] [MPS] [NL]
- FICO-Xpress [AMPL] [GAMS] [MOSEL] [MPS] [NL]
- Gurobi [AMPL] [GAMS] [LP] [MPS] [NL]
- HiGHS [AMPL] [GAMS] [LP] [MPS]
- LINDO [GAMS]

- MOSEK [AMPL] [GAMS] [LP] [MPS] [NL]
- OOQP [AMPL]
- SOPLEX [GAMS]

Для розв'язання задачі за допомогою солвера Gurobi, описаній мовою моделювання AMPL, вибираємо відповідне посилання Gurobi [AMPL]. Відкриється форма подання задачі, на якій необхідно завантажити файл моделі **Model File**, файл даних **Data File**, файл команд **Commands File,** і ввести адресу електронної пошти, на яку прийде результат розв'язання задачі (результат також з'явиться на даній сторінці, після закінчення обчислень).

Для задач *змішаного цілочислового лінійного* програмування використовуються такі програми:

- Cbc [AMPL] [GAMS] [MPS]
- COPT [AMPL] [GAMS] [LP] [MPS] [NL]
- CPLEX [AMPL] [GAMS] [LP] [MPS] [NL]
- FICO-Xpress [AMPL] [GAMS] [MOSEL] [MPS] [NL]
- Gurobi [AMPL] [GAMS] [LP] [MPS] [NL]
- HiGHS [AMPL] [GAMS] [LP] [MPS]
- MINTO [AMPL]
- MOSEK [AMPL] [GAMS] [LP] [MPS] [NL]
- ODHCPLEX [GAMS]
- RAPOSa [AMPL]
- scip [AMPL] [CPLEX] [GAMS] [MPS] [OSIL] [ZIMPL]
- SOPLEX [GAMS]
- SYMPHONY [MPS]

Для задач *нелінійного* програмування *з обмеженнями* використовуються такі програми:

- ANTIGONE [GAMS]
- CONOPT [AMPL] [GAMS]
- FICO-Xpress [MOSEL]

- filter [AMPL]
- Ipopt [AMPL] [GAMS] [NL]
- Knitro [AMPL] [GAMS] [NL]
- LANCELOT [AMPL]
- LINDO [GAMS]
- LOQO [AMPL]
- MINOS [AMPL] [GAMS]
- PATHNLP [GAMS]
- SNOPT [AMPL] [GAMS] [NL]

Розв’язання задач цих типів здійснюється так само, як і в попередньому випадку.

ІІ. Алгебраїчна мова моделювання AMPL

У розділі наведено короткий опис мови моделювання AMPL на основі препринта [3].

2.1. Основні особливості програмування на AMPL

AMPL (A Mathematical Programming Language) [2] – це мова високого рівня для опису задач математичного програмування, що використовує декларативно-алгебраїчний стиль представлення моделей математичного програмування, близьке до традиційної математичної нотації. Разом з тим AMPL дає можливість описати і складні моделі оптимізації з різними логічними умовами, з використанням складних систем індексації змінних і обмежень. AMPL дозволяє задати модель математичного програмування незалежно від даних, що використовуються для конкретного прикладу моделі.

У даному розділі наведені лише базові відомості про AMPL, які можуть знадобитися для опису моделей і виконання лабораторних робіт цього курсу.

AMPL вимагає також завдання вхідних даних (оснащення моделі). Модель і один (або більше) файлів даних направляються в систему AMPL. AMPL працює подібно компілятору: модель і дані з'єднуються в проміжний файл, який передається солверу. Солвер фактично знаходить оптимальне рішення задачі, використовуючи проміжний файл, побудований AMPL, і застосовуючи відповідний алгоритм. Солвер видає рішення у вигляді текстового файлу.

Робота з AMPL з командного рядка

Моделі записуються у вигляді текстового файлу <*назва файлу*> .mod. При написанні моделі мовою AMPL можна використовувати будь-який текстовий редактор.

При написанні моделей використовуються основні команди AMPL. AMPL-модель містить описання об'єктів моделі, тобто множин, змінних, параметрів, цільової функції

і обмежень. Для опису об'єктів використовуються службові слова **set, var, param, minimize/maximize, subject to** (або коротко **s. t.**). При цьому AMPL-модель містить кілька типів елементів, які докладніше описані нижче: декларації з ключовими словами: **set** (множина індексів), **param** (параметр), **var** (змінна), **arc** (дуга – для опису мережевих моделей); цільові функції виду **maximize/minimize**, обмежень, **subject to** (при обмеженнях), **node** (вершина – для опису мережевих моделей).

Багато в чому синтаксис команд AMPL дуже подібний до C. AMPL підтримує такі функції, як abs(), cos(), sin(), log(), sqrt(), exp() з використанням основних операцій +, -, *, /, ^ чи **. Всі команди закінчуються крапкою з комою «;». До команд виводу відносяться **display**, а також команди **write** і **print**. До інших корисних елементів відносяться **option** – для зміни опцій AMPL або вирішувача, **include** – читання з окремого файлу, а також **quit** – для виходу з AMPL.

Імена (ідентифікатори) складаються з латинських букв (прописних і малих), цифр і знаків підкреслення. Символ # означає початок коментаря. Все, що знаходиться за цим символом ігнорується AMPL. Коментарі можуть бути також обмежені символами /* і */, причому вони можуть бути відокремлені один від одного кількома рядками.

Команди AMPL використовують простий синтаксис:

Змінні описуються з використанням службового слова **var**.

Параметри описуються з використанням службового слова **param**.

Сума $\sum_{i=1}^{n}$ записується так: sum {i in 1..n } .

Службові слова AMPL (такі як **var, param, solve, maximize** та ін), а також імена функцій (наприклад, **sum, log, sin**) зарезервовані і не можуть використовуватися для імен об'єктів. До службових зарезервованих слів належать також **for, if, elseif, else, while, file, system.**

Індекси змінних і обмежень вставляються у квадратні дужки (наприклад, a[i]).

Числа можуть записуватися у різних форматах. Так, 0.0123, 1.23D-2, 1.23e-2, 1.23E-2 – це еквівалентні записи одного і того ж числа 0,0123.

Літерали – це рядки, взяті в лапки (одинарні або подвійні). Наприклад, 'abc', 'x', 'y', "ABC".

Команда вигляду

include <назва файлу>

вставляє вказаний файл.

Команди

model; include <назва файла>

data; include <назва файла>

можуть бути скорочені до наступних:

model <назва файла>;

data <назва файла>;

Команда **commands** аналогічна команді **include**, але це оператор і повинен закінчуватися крапкою з комою.

У AMPL модель і дані розділені. Множини, описані в моделі не мають розмірів або заданих елементів. Використання в моделі множини індексів означає лише те, що використовуються елементи цієї множини, причому неважливо скільки їх є. Задання величин здійснюється тільки у файлі даних, який має вигляд ***<назва файла>***. dat. Тут елементи множин і параметри визначаються явно, повторно записуючи службове слово, назву об'єкта і перераховуючи значення після ": =".

Функціонування моделі та даних у системі моделювання AMPL наведено на рис. 2.1.

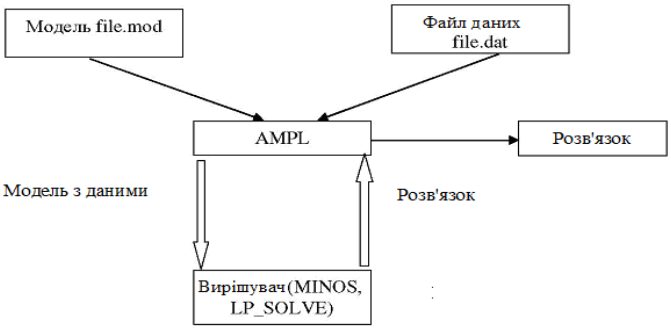

Рис. 2.1. Схема роботи системи моделювання AMPL

Приклад створення моделі AMPL

Розв’язати наступну задачу, використовуючи MINOS.

$z = x_1 + x_2 \to \min$

при обмеженнях

$x_1 \cdot x_2 \geq 4,$

$0 \leq x_1, x_2 \leq 3.$

Створимо наступний текстовий файл з назвою ex1.mod

```
    # ex1.mod – Model
    var x1 > = 0, < = 3;     # Задання
нижньої та верхньої меж змінної x1
    var x2 >= 0, <= 3;       # Задання
нижньої та верхньої меж змінної x2
    minimize z: x1 + x2;          # Цільова
функція z
    s.t. con1: x1 * x2 >= 4;     # Обмеження
1 – му присвоєно ім’я con1
    data;                                  #
Задання початкової точки для вирішувача
    var x1 := 1;
```

```
var x2 := 1;
```

Потім наберіть **>ampl** (перебуваючи в папці, де знаходиться виконуючий файл ampl). З'явиться командне вікно AMPL з підказкою **ampl:**

У вікні AMPL наберіть наступні команди для розв'язання цього завдання (не забудьте про крапку з комою ";" вкінці кожної команди) за допомогою вирішувача MINOS (за замовчуванням).

```
ampl: include ex1.mod;
ampl: solve;
ampl: display x1, x2, z;
```

AMPL видає наступну інформацію про розв'язок x1 = 2, x2 = 2 з оптимальним значенням цільової функції z = 4.

```
MINOS 5.5: optimal solution found.
15 iterations, objective 4
Nonlin evals: constrs = 38, Jac = 37.
x1 = 2
x2 = 2
z = 4
```

Для редагування текстового файлу (у зв'язку з виявленими помилками або необхідністю зміни моделі), внесіть необхідні зміни в текстовому файлі моделі і збережіть його. Потім у вікні AMPL наберіть наступні команди для введення зміненої моделі:

```
ampl: reset;
ampl: model ex1.mod;
```

Запишемо на AMPL модель цілочислового лінійного програмування (ЦЛП) вигляду:

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

при обмеженнях

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

$$x_j = 0,1, \quad j = 1,...,n.$$

Опис параметра m в AMPL виглядає так:

```
param m;
```

Аналогічно опишемо параметр n:

```
param n;
```

Опишемо множину індексів ***i = 1*** , ..., ***m*** та ***j = 1*** , ..., ***n*** :

```
set I=1..m;
set J=1..n;
```

Для описання параметрів c_j, b_i, a_{ij} використовуємо запис:

```
param c{J};
param b{I};
param a{I, J};
```

Для опису змінних використовуємо запис:

```
var  x{J} binary;
```

який означає, що x_j – бінарна змінна, що приймає тільки значення 0 або 1.

Опис цільової функції $\sum_{j=1}^{n} c_j x_j \to \max$ має наступний вигляд:

```
maximize z: sum {j in 1..n} c[j] * x[j];
```

Обмеження $\sum_{j=1}^{n} a_{ij} x_j \le b_i,\ i = 1,...,m,$ запишуться так:

```
con{i in 1..m}: sum {j in 1..n} a[i,j]
* x[j] <= b[i];
```

Таким чином, модель ЦЛП записується на AMPL у вигляді ip.mod:

```
param m;
param n;
set I:=1..m;
set J:=1..n;
param c {J};
param b {I};
param a {I, J};
var x {J} binary;
maximize z: sum{j in J}c[j]*x[j];
subject to res {i in I}:
sum{j in J}a[i,j]*x[j]<= b[i];
```

Для розв'язку конкретної задачі ЦЛП вигляду

$$z = 2x_1 + 3x_2 + x_3 \to \max$$

при обмеженнях

$$x_1 + 2x_2 + x_3 \le 2,$$
$$2x_1 + 3x_2 + 2x_3 \le 4,$$
$$x_1, x_2, x_3 \in \{0,1\},$$

можна використовувати описаний вище файл моделі ip.mod разом з файлом даних ip.dat наступного вигляду:

```
param m:=2;
param n:=3;
param c [1] 2    [2] 3   [3] 1;
```

```
param b [1] 2    [2] 4;
param a: 1    2    3:=
1             1    2    1
2             2    3    2;
```

У вікні AMPL набираємо оператори:

```
model ip.mod;
data ip.mod;
option solver lpsolve;
solve;
display x, obj;
```

В результаті отримаємо:

```
LP_SOLVE 4.0.1.0: optimal, objective 3
5 simplex iterations
3 branch & bound nodes: depth 2
x [*] :=
1 1
2 0
3 1
;

obj = 3
```

Зауваження. Запис `option solver lpsolve;` означає явне задання вирішувача завдань задач змішаного цілочислового лінійного програмування lp_solve (http://lpsolve.sourceforge.net/).

Загальні синоніми для змінних, обмежень і цільових функцій

У багатьох випадках потрібно роздрукувати або перевірити всі наявні в моделі змінні, обмеження, цільові

функції. З цією метою AMPL використовує спільні синоніми всіх компонентів:

_nvars – число змінних в розглянутій моделі;
_ncons – число обмежень у розглянутій моделі;
_nobjs – число цільових функцій в розглянутій моделі.

Відповідно параметри з індексами містять імена всіх компонент моделі AMPL:

_varname {1..nvars} – імена змінних у розглянутій моделі;
_conname {1..ncons} – імена обмежень у розглянутій моделі;
_objname {1..nobjs} – імена цільових функцій в розглянутій моделі.

І, нарешті, є наступні синоніми для компонент:
_var {1..nvars} – синоніми змінних в розглянутій моделі;
_con {1.._ncons} – синоніми обмежень у розглянутій моделі;
_obj {1.._nobjs} – синоніми цільових функцій в розглянутій моделі.

Графічний інтерфейс AMPL

Існує графічний інтерфейс AMPL, який називається AMPL Studio і доступний на Web-сторінці **http://www.optirisk-systems.com.**

2.2. Декларації компонент моделі

Надалі для формального опису операторів використовуємо **форму Бекуса-Наура** (скор. БНФ, Бекуса-Наура форма) – формальну систему опису синтаксису, в якій одні синтаксичні категорії послідовно визначаються через інші категорії. БНФ-конструкція складається з рядочків (речень) вигляду:

<визначуваний символ> :: = <посл.1> | <посл.2> |. . . | <посл.n>,

які описують правила. Таке правило означає, що символ <визначуваний символ> може замінюватися на одну з

послідовностей посл.1, посл.2, ..., посл.n. Знак визначення зазвичай виглядає як :: =. Крім того, використовуються квадратні дужки: [A] – елемент A входить або не входить.

Декларації об'єктів моделі AMPL мають наступний загальний вигляд <об'єкт> <назва об'єкта> [<аліас>] [<індексний вираз>] [<тіло>] <об'єкт> :: = set | param | var | arc | minimize | maximize | subject to | node

Декларації параметрів

Параметрами можуть бути скалярні величини, вектори, матриці й масиви відомих даних. Декларація параметра має вигляд:

param *<назва_змінної> <аліас> <індексний вираз> <атрибути>*;

<аліас> ::=<літерал>

<индексний вираз>::=<set_expression_list

*<арифметичний_оператор>::=+ | – | less | * | / | mod | div | ^| ** |*

<унарний оператор>::= + | -

<оператор редукції>::= sum | prod | max | min

Необов'язкові атрибути декларацій параметрів можуть розділятися комами, нижче наведено зазначені атрибути:

Атрибут:	Примітка
Binary	Обмежує можливі значення змінної 0 або 1.
Integer	Обмежує можливі значення змінної цілочисловими значеннями

Symbolic	Якщо задано **symbolic**, то мають також бути задані **in sexpr**, а також виключаються атрибути, що вимагають задання числових значень, такі як > = <вираз>
> = <Вираз>	> = Задає нижню межу
<= <Вираз>	<= Задає верхню межу
: = <Вираз>	: = Задає початкове значення
default <вираз>	Задає значення за замовчуванням для початкових значень, які можуть бути задані в розділі даних **data**
= <Вираз>	Задає певну змінну

AMPL дозволяє визначати параметри через інші, які вже раніше були визначені. Параметри можуть також обчислюватися і визначатися рекурсивно. Наприклад, для числа комбінацій C_n^m по ***m*** із ***n*** предметів може бути визначено, використовуючи наступну декларацію

```
    param comb {i in 0..n, k in 0..m} :=
    if k = 0 or k = i then 1 else comb[i-
1,k-1] + comb[i-1,k];
```

Декларації змінних

Декларації змінних починаються зі службового слова ***var:***

var *<назва змінної> <аліас> <індексний вираз> <атрибути>,*

де всі елементи декларації такі ж, як у декларації параметра (див. вище).

Декларації обмежень

Обмеження записуються в наступній загальній формі:

<Декларація обмеження> :: =
[**subject to**] <*назва обмеження*> [<*аліас*>] [<*індексний вираз*>] <*атрибути*>
[: <Вираз обмеження>];

< вираз обмеження > **::** = <*вираз*> <= <*вираз*> |
< *вираз* > =<*вираз*> | <*вираз*> > = <*вираз*> |

Декларація цільової функції

Декларація (опис) цільової функції має вигляд:

< декларація цільової функції > :: = **maximize** <назва> [<аліас>] [<*індексний вираз*>] [: вираз]; |
minimize < назва > [< аліас >] [<*індексний вираз* >] [: вираз];

Огляд команд AMPL

Команда	Коментар
call	Виклик імпортованої функції
Cd	Перехід в інший каталог
check	Виконує всі команди check
close	Закриває файл
commands	Читання та інтерпретація команд з файлу
data	Перехід до даних
delete	Видалення компонент моделі
display	Друк компонент моделі і виразів

drop	Виняток обмеження або цільової функції
end	Закінчення введення з поточного файлу введення
environ	Встановити середовище для моделі
exit	Вийти з AMPL зі значенням статусу
expand	Показати детально компоненти моделі
Fix	Зафіксувати змінну на її поточному значенні
include	Включити вміст файлу
Let	Змінити значення даних
load	Завантажити динамічну бібліотеку
model	Завантажує модель
objective	Вибрати цільову функцію для оптимізації
option	Встановити або видати значення опцій
print	Неформатований друк компонент моделі і виразів
printf	Форматований друк компонент моделі і виразів
problem	Визначити задачу або перейти до задачі
purge	Видалення компонент моделі
quit	Закінчити роботу AMPL
read	Читання з файлу
read table	Читання з таблиці даних
redeclare	Зміна декларації об'єкта
reload	Перезавантаження динамічної бібліотеки функцій
remove	Видалення файлу
reset	Скидання об'єктів, відновлення їх початкового стану

restore	Скасувати дію команди **drop**
shell	Тимчасовий вихід в операційну систему
show	Показати імена компонент моделі
solexpand	Показати детальне розширення, видиме вирішувачем
solution	Імпортує значення змінної з вирішувача
solve	Конкретний приклад моделі надсилається вирішувачу і повертається знайдене рішення
update	Дозволити зміну даних
unfix	Скасувати дію команди **fix**
unload	Вивантажити динамічну бібліотеку
write	Видача конкретного прикладу задачі
write table	Записати таблицю в таблицю даних
xref	Показати залежності між компонентами моделі

Файли, що згадуються в командах **include, model, data, commands**, мають прості імена (наприклад, не містять слеш /), шукаються в папках, що задаються опцією **ampl_include:** кожний непорожній рядок **$ampl_include** задає таку папку; якщо ж **$ampl_include** порожній або є чистим, файли шукаються в поточній папці.

Перенаправлення виводу

Для перенаправлення виводу у файл, досить додати > і назву файла. Для відкриття вже існуючого файлу і додавання до нього виведення потрібно використовувати >> замість >.

Наприклад:

```
ampl: model ip.mod;
ampl: data ip.dat;
ampl: solve;
```

```
LP_SOLVE 4.0.1.0: optimal, objective 3
5 simplex iterations
3 branch & bound nodes: depth 2
```

```
ampl: display x, obj > ip.out;
```

Скрипти та оператори потоку управління

В AMPL передбачені оператори, подібні операторам потоку управління у звичайних мовах програмування, які дають можливість писати програму за допомогою операторів, яка буде виконуватися автоматично.

При роботі з моделлю користувач часто застосовує послідовності одних і тих же команд, які потрібно набирати на екрані. Для прискорення процесу можна записати ці команди в спеціальний скриптовий файл <file>.run, викликаючи його потім за допомогою

include <file>. Run

Так, для виклику файлів з прикладу можна записати у файлі ip.run

```
model ip.mod;
data ip.dat;
option solver lpsolve;
solve;
display x, z;
```

Для запуску скрипта на екрані AMPL набирається `include ip. run;`

Стандартний формат даних

AMPL підтримує стандартний формат, щоб описати значення множин і параметрів, які об'єднуються з моделлю, щоб отримати конкретну задачу оптимізації.

Розділ даних складається з послідовності токенів (лексем) (літералів і рядків друкованих символів), відділених пробілами, символами табуляції і переведення рядка.

Лексеми включають ключові слова, літерали, числа, і розділювачі () []:;: = *. Затвердження (оператор) – послідовність лексем, що закінчується крапкою з комою. Коментарі можуть використовуватися так само, як в деклараціях. У всіх випадках розташування даних в чітких рядках і стовпцях – для зручності читання людиною; AMPL ігнорує форматування.

Розділ даних починається з команди `data` і закінчується кінцем-вводу або командою, яка повертає в режим моделювання.

У розділі даних, об'єктам моделі можуть бути призначені значення в будь-якому зручному порядку, незалежному від порядку їх декларування.

Дані для множин

Оператори, що визначають множину, складаються з ключового слова **set** (множина), назви множини, опціональних: =, і членів. Одновимірну множину найбільш просто визначити, задаючи список її членів, опціонально відокремлених комами. Одиничні або подвійні лапки символьного рядка можуть бути опущені, якщо рядок алфавітно-цифровий, але не визначає число.

Приклади задання даних для параметрів наведені вище в прикладі для моделі ЦЛП.

Більш докладні відомості про мову алгебраїчного моделювання AMPL можна знайти в книзі [4] і на веб-сайті https://ampl.com/.

ІІІ. Задачі математичного програмування та їх AMPL-код

У розділі наведено різноманітні формулювання задач лінійного [5, 6, 8], цілочислового та нелінійного [9, 10, 16] програмування, способи їх опису за допомогою мови моделювання AMPL, результати розв'язання програмами Gurobi [7] та MINOS [8]. Значна увага приділена розв'язанню транспортної задачі у матричній формі [10].

3.1. Задача лінійного програмування

3.1.1. Найпростіша математична модель та її AMPL-код

Розглянемо таку модель задачі лінійного програмування:

$$z = x_1 + 2x_2 - 3x_3 \to \min$$

$$\begin{cases} -2x_1 + x_2 + 3x_3 \le 4, \\ 2x_1 + 3x_2 \le 6, \\ x_1 + x_3 \le 2, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

AMPL-код для вказаної моделі має такий вигляд:

```
var x1 >=0;
var x2 >=0;
var x3 >=0;

minimize z: x1+2*x2-3*x3;
s.t. con1: -2*x1+x2+3*x3 <= 4;
s.t. con2: 2*x1+3*x2 <= 6;
s.t. con3: x1+x3 <= 2;

solve;
```

```
display z;
display x1, x2, x3;
```

Результат виконання на NEOS-сервері:

```
Solved in 2 iterations and 0.00 seconds
Optimal objective -4.400000000e+00
Gurobi 5.5.0: optimal solution; objective -
4.4
2 simplex iterations

z = -4.4
x1 = 0.4
x2 = 0
x3 = 1.6
```

Результат виконання на AMPL:

```
MINOS 5.5: optimal solution found.
2 iterations, objective -4.4

z = -4.4
x1 = 0.4
x2 = 0
x3 = 1.6
```

3.1.2. Математична модель загального виду та її AMPL-код

Вона включає 3 змінних та 3 обмеження, і має такий вигляд:

$$Z = c^T x \to \min \qquad (3.1.1)$$

при обмеженнях:

$$Ax \le b, \qquad (3.1.2)$$
$$x \ge 0, \qquad (3.1.3)$$

де вектори c, b та матриця A є такими:

$$c = (1,2,-3)^T;\ b = (4,6,2)^T;\ A = \begin{pmatrix} -2 & 1 & 3 \\ 2 & 3 & 0 \\ 1 & 0 & 1 \end{pmatrix}.$$

AMPL-код для моделі (3.1.1) – (3.1.3) має такий вигляд:

```
param m;
param n;

set I=1..m;
set J= 1..n;

param c{J};
param b{I};
param a{I,J};

var x{J} >=0;

minimize z: sum {j in J} c[j] * x[j];
subject to con{i in I}: sum {j in J} a[i,j]
* x[j] <= b[i];

data;
param m:=3;
param n:=3;
param c [1] 1 [2] 2 [3] -3;
param b [1] 4 [2] 6 [3] 2;
param a:     1  2  3:=
         1   -2  1  3
         2   2  3  0
         3   1  0  1;
```

```
solve;
display z, x;
```

Результат виконання на NEOS-сервері:

```
Solved in 2 iterations and 0.00 seconds
Optimal objective -4.400000000e+00
Gurobi 5.5.0: optimal solution; objective -
4.4
2 simplex iterations
z = -4.4
x [*] :=
1  0.4
2  0
3  1.6
```

Результат виконання на AMPL:

```
CPLEX 12.6.0.0: optimal solution; objective
-4.4
2 dual simplex iterations (2 in phase I)
z = -4.4
x [*] :=
1  0.4
2  0
3  1.6
```

3.1.3. Задачі для розв'язання

1. $z = 2x_1 - x_2 - 3x_3 \to \max$

$$\begin{cases} 2x_1 + x_3 \le 6, \\ x_1 - x_2 - x_3 = -2, \\ -x_1 + 6x_3 \ge -4, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

2. $z = 3x_1 + 2x_2 - 2x_3 + x_4 \to \min$

$$\begin{cases} -x_1 + x_2 - x_4 = -5, \\ 2x_2 + x_3 \le 4, \\ -x_1 - x_2 + 2x_3 \ge -8, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,4}.$$

3. $z = 2x_1 + x_2 - x_3 \to \max$

$$\begin{cases} x_1 + x_2 \le 1, \\ -x_2 - x_3 \ge -2, \\ x_1 + x_3 \le 2, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

4. $z = -2x_1 - 3x_2 + x_3 \to \min$

$$\begin{cases} 2x_1 + x_2 - x_3 = -2, \\ x_1 + 2x_2 \le 5, \\ 3x_1 - 2x_2 \ge -4, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

5. $z = 4x_1 + 3x_2 + 5x_3 \to \max$

$$\begin{cases} x_1 - x_3 \le 6, \\ x_1 + x_2 + 2x_3 = 5, \\ -2x_1 + 2x_3 \ge -4, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

6. $z = x_1 + 2x_2 - 3x_3 \to \min$

$$\begin{cases} -2x_1 + x_2 + 3x_3 \le 4, \\ 2x_1 + 3x_2 \le 6, \\ x_1 + x_3 \le 2, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

7. $z = x_1 - 5x_2 - x_3 \to \max$

$$\begin{cases} x_1 + 3x_2 + x_3 \ge 3, \\ 2x_1 + x_3 = 4, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

8. $z = 5x_1 + 4x_2 + 2x_3 \to \min$

$$\begin{cases} x_1 + 2x_2 + x_3 = 5, \\ 2x_1 + 3x_2 + x_3 \ge 8, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

9. $z = 5x_1 + 4x_2 + 2x_3 \to \min$

$$\begin{cases} x_1 + 2x_2 + x_3 = 5, \\ 2x_1 + 3x_2 + x_3 \geq 8, \end{cases}$$

$$x_j \geq 0, \quad j = \overline{1,3}.$$

10. $z = 2x_1 + 3x_2 - 5x_3 \to \min$

$$\begin{cases} x_1 + x_2 + x_3 = 7, \\ 2x_1 - 5x_2 + x_3 \geq 10, \end{cases}$$

$$x_j \geq 0, \quad j = \overline{1,3}.$$

11. $z = 2x_1 + 4x_2 + 4x_3 \to \max$

$$\begin{cases} x_1 + x_2 = 4, \\ x_1 + 4x_2 + x_3 < 8, \end{cases}$$

$$x_j \geq 0, \quad j = \overline{1,4}.$$

12. $z = 3x_1 + 2x_2 + 3x_3 \to \min$

$$\begin{cases} x_1 + 4x_2 + x_3 \geq 8, \\ 2x_1 + x_2 + x_3 = 10, \end{cases}$$

$$x_j \geq 0, \quad j = \overline{1,3}.$$

13. $z = 3x_1 + 2x_2 + 3x_3 \to \min$

$$\begin{cases} x_1 + 4x_2 + x_3 \geq 8, \\ 2x_1 + x_2 + x_3 = 10, \end{cases}$$

$$x_j \geq 0, \quad j = \overline{1,3}.$$

14. $z = x_1 + x_2 + x_3 + x_4 \to \min$

$$\begin{cases} 2x_1 + 2x_2 + x_3 \geq 7, \\ 4x_1 + 2x_2 - x_3 + x_4 = 8, \end{cases}$$

$$x_j \geq 0, \quad j = \overline{1,4}.$$

15. $z = x_1 + x_2 \to \max$

$$\begin{cases} 2x_1 + 3x_2 = 5, \\ 3x_1 + 2x_2 \leq 6, \end{cases}$$

$$x_j \geq 0, \quad j = \overline{1,2}.$$

16. $z = -3x_1 + 2x_2 + 5x_3 \to \min$

$$\begin{cases} x_1 - x_2 + 3x_3 \leq 10, \\ x_1 + 3x_2 + 2x_3 = 8, \end{cases}$$

$$x_j \geq 0, \quad j = \overline{1,3}.$$

17. $z = x_1 - x_2 + 3x_3 \to \max$

$$\begin{cases} x_1 + 3x_3 \leq 10, \\ x_1 - x_2 - x_3 = -3, \end{cases}$$

$$x_j \geq 0, \quad j = \overline{1,3}.$$

18. $z = 3x_1 - 2x_2 - x_3 \to \min$

$$\begin{cases} -x_1 - 3x_2 + 3x_3 \geq -6, \\ -2x_1 + x_2 + x_3 \leq 4, \end{cases}$$

$$x_j \geq 0, \quad j = \overline{1,3}.$$

19. $z = 30x_1 + 30x_2 + 45x_3 \to \min$

$$\begin{cases} 3x_1 + x_2 - x_3 \le 0, \\ -5x_1 + 3x_2 - x_3 \le 0, \end{cases}$$

$$x_j \ge 0,\ j = \overline{1,3}$$

20. $z = x_1 + x_2 + x_3 - 4x_4 \to \max$

$$\begin{cases} x_1 + 4x_2 + x_3 - x_4 < 10, \\ 2x_1 - x_2 + x_3 + x_4 < 10, \end{cases}$$

$$x_j \ge 0,\ j = \overline{1,4}$$

3.2. Задача цілочислового лінійного програмування

3.2.1. Математична модель з цілочисловими та лінійними змінними та її AMPL-код

Розглянемо задачу змішаного лінійно-цілочислового програмування у такому вигляді:

$$z = x_1 + 2x_2 - 3x_3 \to \min$$

$$\begin{cases} -2x_1 + x_2 + 3x_3 \le 4, \\ 2x_1 + 3x_2 \le 6, \\ x_1 + x_3 \le 2, \end{cases}$$

$$x_1 - \textit{ціле}, x_j \ge 0,\ \ j = \overline{1,3}.$$

Вона включає одну цілочислову та дві лінійні змінні.

AMPL-код для вказаної моделі має такий вигляд:

```
var x1 integer >=0;
var x2 >=0;
var x3 >=0;

minimize z: x1+2*x2-3*x3;
s.t. con1: -2*x1+x2+3*x3 <= 4;
s.t. con2: 2*x1+3*x2 <= 6;
s.t. con3: x1+x3 <= 2;

solve;
```

```
display z;
display x1, x2, x3;
```

Результат виконання на NEOS-сервері:

```
Solved in 2 iterations and 0.00 seconds
Optimal objective -4.000000000e+00
Gurobi 5.5.0: optimal solution; objective -4
plus 2 simplex iterations for intbasis
z = -4
x1 = 0
x2 = 0
x3 = 1.33333
```

Результат виконання на AMPL:

```
CPLEX 12.6.0.0: optimal integer solution;
objective -4
0 MIP simplex iterations
0 branch-and-bound nodes
z = -4
x1 = 0
x2 = 0
x3 = 1.33333
```

3.2.2. Математична модель з цілочисловими змінними та її AMPL-код

Розглянемо задачу цілочислового лінійного програмування у такому вигляді:

$$z = x_1 + 2x_2 - 3x_3 \to \min$$

$$\begin{cases} -2x_1 + x_2 + 3x_3 \le 4, \\ 2x_1 + 3x_2 \le 6, \\ x_1 + x_3 \le 2, \end{cases}$$

$$x_1, x_2, x_3 - цілі, x_j \ge 0, \quad j = \overline{1,3}.$$

AMPL-код для вказаної моделі має такий вигляд:

```
var x1 integer >=0;
var x2 integer >=0;
var x3 integer >=0;

minimize z: x1+2*x2-3*x3;
s.t. con1: -2*x1+x2+3*x3 <= 4;
s.t. con2: 2*x1+3*x2 <= 6;
s.t. con3: x1+x3 <= 2;

solve;

display z;
display x1, x2, x3;
```

Результат виконання на NEOS-сервері:

```
Solved in 0 iterations and 0.00 seconds
Optimal objective -3.000000000e+00
Gurobi 5.5.0: optimal solution; objective -3
z = -3
x1 = 0
x2 = 0
x3 = 1
```

Результат виконання на AMPL:

```
CPLEX 12.6.0.0: optimal integer solution;
objective -3
0 MIP simplex iterations
0 branch-and-bound nodes
z = -3
x1 = 0
x2 = 0
x3 = 1
```

3.2.3. Задачі для розв'язання

Ними є задачі підрозділу 3.1.3, в яких перша або усі змінні є цілочисловими. Для них обмеження $x_j \geq 0,\ \ j = \overline{1,3}.$ потрібно замінити на обмеження $x_1 - \textit{ціле}, x_j \geq 0,\ \ j = \overline{1,3}$, або на обмеження $x_1, x_2, x_3 - \textit{цілі}, x_j \geq 0,\ \ j = \overline{1,3}$. Якщо задача включає чотири змінних, то обмеження $x_j \geq 0,\ \ j = \overline{1,4}.$ замінюється обмеженням $x_1 - \textit{ціле}, x_j \geq 0,\ \ j = \overline{1,4}$, або обмеженням $x_1, x_2, x_3, x_4 - \textit{цілі}, x_j \geq 0,\ \ j = \overline{1,4}$.

3.3. Задача нелінійного програмування

3.3.1. Математична модель та її AMPL-код

Розглянемо задачу нелінійного програмування у такому вигляді

$$z = x_1 + 5x_2 + 3x_3 \to \min$$
$$\begin{cases} 2x_1 + x_2 = 4, \\ x_1 + 2x_2 + x_3 = 6, \\ x_1 x_2 \geq 2, \end{cases}$$
$$x_j \geq 0, \quad j = \overline{1,3}.$$

Вона включає лінійну цільову функцію, два лінійних та одне нелінійне обмеження.

AMPL-код для вказаної моделі має такий вигляд:

```
var x1 >=0;
var x2 >=0;
var x3 >=0;

maximize z: x1+5*x2+3*x3;
s.t. con1: 2*x1+x2 = 4;
s.t. con2: x1+2*x2+x3 = 6;
s.t. con3: x1*x2 >= 2;

solve;

display z;
display x1, x2, x3;
```

Результат виконання на NEOS-сервері:

```
MINOS 5.51: optimal solution found.
0 iterations, objective 14
Nonlin evals: constrs = 16, Jac = 15.
z = 14
x1 = 1.00006
x2 = 1.99988
x3 = 1.00018
```

Результат виконання на AMPL:

```
MINOS 5.5: optimal solution found.
0 iterations, objective 14
Nonlin evals: constrs = 16, Jac = 15.
z = 14

x1 = 1.00006
x2 = 1.99988
x3 = 1.00018
```

3.3.2. Задачі для розв'язання

$z = 2x_1 - 2x_2 - 3x_3 \to \max$

1. $\begin{cases} 2x_1 + x_3 \le 6, \\ x_1 - x_2 - x_3 = -2, \\ x_1 x_3 \le 4, \end{cases}$ $x_j \ge 0, \quad j = \overline{1,3}.$

$z = 3x_1 + 2x_2 - 2x_3 + 2x_4 \to \min$

2. $\begin{cases} -x_1 + x_2 - x_4 = -5, \\ x_2 x_3 \le 14, \\ -x_1 - x_2 + 2x_3 \ge -8, \end{cases}$ $x_j \ge 0, \quad j = \overline{1,4}.$

$z = 2x_1 + x_2 - 2x_3 \to \max$

3. $\begin{cases} x_1 x_2 \le 1, \\ -x_2 - x_3 \ge -2, \\ x_1 + x_3 \le 2, \end{cases}$ $x_j \ge 0, \quad j = \overline{1,3}.$

$z = -2x_1 - 3x_2 + 2x_3 \to \min$

4. $\begin{cases} 2x_1 + x_2 - x_3 = -2, \\ x_1 x_2 \le 5, \\ 3x_1 - 2x_2 \ge -4, \end{cases}$ $x_j \ge 0, \quad j = \overline{1,3}.$

$z = 4x_1 + 6x_2 + 5x_3 \to \max$

5. $\begin{cases} x_1 x_3 \le 6, \\ x_1 + x_2 + 2x_3 = 5, \\ -2x_1 + 2x_3 \ge -4, \end{cases}$ $x_j \ge 0, \quad j = \overline{1,3}.$

$z = 3x_1 + 2x_2 - 3x_3 \to \min$

6. $\begin{cases} -2x_1 + x_2 + 3x_3 \le 4, \\ 2x_1 + 3x_2 \le 6, \\ x_1 x_3 \le 2, \end{cases}$ $x_j \ge 0, \quad j = \overline{1,3}.$

7. $z = x_1 - 5x_2 - 2x_3 \to \max$

$$\begin{cases} x_1 + 3x_2 + x_3 \le 13, \\ x_1 x_3 \le 14, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

8. $z = 5x_1 + 4x_2 + 3x_3 \to \min$

$$\begin{cases} x_1 x_2 x_3 \ge 5, \\ 2x_1 + 3x_2 + x_3 \ge 8, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

9. $z = 5x_1 + 4x_2 + 2x_3 \to \min$

$$\begin{cases} x_1 x_2 \le 5, \\ 2x_1 + 3x_2 + x_3 \ge 8, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

10. $z = 2x_1 + 3x_2 - 5x_3 \to \min$

$$\begin{cases} x_1 x_2 x_3 \ge 7, \\ 2x_1 - 5x_2 + x_3 \ge 10, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

11. $z = 2x_1 + 7x_2 + 4x_3 \to \max$

$$\begin{cases} x_1 x_2 = 4, \\ x_1 + 4x_2 + x_3 < 8, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,4}.$$

12. $z = 3x_1 + 2x_2 + 7x_3 \to \min$

$$\begin{cases} x_1 x_2 \ge 8, \\ 2x_1 + x_2 + x_3 \ge 10, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

13. $z = 3x_1 + 6x_2 + 3x_3 \to \min$

$$\begin{cases} x_1 + 4x_2 + x_3 \ge 8, \\ x_1 x_3 = 10, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

14. $z = x_1 + x_2 + 4x_3 + x_4 \to \min$

$$\begin{cases} x_1 x_2 x_3 \ge 7, \\ 4x_1 + 2x_2 - x_3 + x_4 = 8, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,4}.$$

15. $z = x_1 + 2x_2 \to \max$

$$\begin{cases} x_1 x_2 \le 4, \\ 3x_1 + 2x_2 \le 6, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,2}.$$

16. $z = -3x_1 + 2x_2 + 8x_3 \to \min$

$$\begin{cases} x_1 - x_2 + 3x_3 \le 10, \\ x_1 x_2 \le 8, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

17. $z = x_1 - 2x_2 + 3x_3 \to \max$

$$\begin{cases} x_1 x_3 \le 10, \\ x_1 - x_2 - x_3 = -3, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

18. $z = 3x_1 - 3x_2 - x_3 \to \min$

$$\begin{cases} -x_1 - 3x_2 + 3x_3 \ge -6, \\ x_2 x_3 \le 4, \end{cases}$$

$$x_j \ge 0, \quad j = \overline{1,3}.$$

19. $z = 30x_1 + 10x_2 + 45x_3 \to \min$

$$\begin{cases} x_1 x_2 \le 10, \\ -5x_1 + 3x_2 - x_3 \ge 0, \end{cases}$$

$$x_j \ge 0, \; j = \overline{1,3}$$

20. $z = x_1 + 2x_2 + x_3 - 4x_4 \to \max$

$$\begin{cases} x_1 x_3 < 10, \\ 2x_1 - x_2 + x_3 + x_4 < 10, \end{cases}$$

$$x_j \ge 0, \; j = \overline{1,4}$$

3.4. Транспортна задача

3.4.1. Приклад транспортної задачі

Компанія контролює три фабрики (постачальники) A_1, A_2, A_3, здатні виготовляти 150, 60 та 80 тис. одиниць продукції щотижня. Компанія уклала договір із чотирма фірмами-замовниками (споживачі) B_1, B_2, B_3, B_4, яким потрібно щотижня відповідно 110, 40, 60 та 80 тис. одиниць продукції. Вартість транспортування 1000 одиниць продукції фірмам-замовникам з кожної фабрики наведено в таблиці.

Фабрика	Вартість транспортування 1000 од. продукції за фірмами-замовниками			
	B_1	B_2	B_3	B_4
A_1	4	4	2	5
A_2	5	3	1	2
A_3	2	1	4	2

Визначити для кожної фабрики оптимальний план перевезення продукції до замовників, що мінімізує загальну вартість транспортних послуг.

3.4.2. Математична модель та AMPL-код

Нехай x_{ij} – кількість продукції, що перевозиться від i-ї фабрики до j-го замовника $(i=\overline{1,3};\ j=\overline{1,4})$. Оскільки транспортна задача за умовою є збалансованою, закритою $\left(\sum_{i=1}^{3} a_i = \sum_{j=1}^{4} b_j = 290\right)$, то обмеження задачі можна записати за допомогою двох груп лінійних рівностей: перша група рівностей стосується фабрик, а друга – фірм-замовників.

Лінійні рівності для фабрик мають наступний вигляд:

$$\begin{cases} x_{11}+x_{12}+x_{13}+x_{14}=150, \\ x_{21}+x_{22}+x_{23}+x_{24}=60, \\ x_{31}+x_{32}+x_{33}+x_{34}=80. \end{cases}$$

Економічний зміст записаних лінійних обмежень полягає у тому, що уся вироблена на фабриках продукція має повністю вивозитися до фірм-замовників.

Аналогічні обмеження можна записати відносно фірм-замовників: продукція, що надходить до споживача, має повністю задовольняти його попит. Математично це записується так:

$$\begin{cases} x_{11}+x_{21}+x_{31}=110, \\ x_{12}+x_{22}+x_{32}=40, \\ x_{13}+x_{23}+x_{33}=60, \\ x_{14}+x_{24}+x_{34}=80. \end{cases}$$

Загальні витрати, пов’язані з транспортуванням продукції, складаються як добуток обсягу перевезеної продукції та питомої вартості перевезень за відповідним маршрутом і за

умовою задачі мають бути мінімальними. Тому математичну модель поставленої задачі можна записати так:

$$Z = 4x_{11} + 4x_{12} + 2x_{13} + 5x_{14} + 5x_{21} + 3x_{22} + x_{23} + 2x_{24} + \\ + 2x_{31} + x_{32} + 4x_{33} + 2x_{34} \to \min$$

при обмеженнях

$$\begin{cases} x_{11} + x_{12} + x_{13} + x_{14} = 150, \\ x_{21} + x_{22} + x_{23} + x_{24} = 60, \\ x_{31} + x_{32} + x_{33} + x_{34} = 80. \\ x_{11} + x_{21} + x_{31} = 110, \\ x_{12} + x_{22} + x_{32} = 40, \\ x_{13} + x_{23} + x_{33} = 60, \\ x_{14} + x_{24} + x_{34} = 80. \end{cases}$$

$$x_{ij} \geq 0,\ i = \overline{1,3},\ j = \overline{1,4}.$$

AMPL-код для вказаної моделі має такий вигляд:

```
var x11 >= 0; var x12 >= 0; var x13 >= 0;
var x14 >= 0;
var x21 >= 0; var x22 >= 0; var x23 >= 0;
var x24 >= 0;
var x31 >= 0; var x32 >= 0; var x33 >= 0;
var x34 >= 0;

minimize z:
4*x11+4*x12+2*x13+5*x14+5*x21+3*x22+x23
            +2*x24+2*x31+x32+4*x33+2*x34;
subject to A1:  x11+x12+x13+x14=150;
subject to A2:  x21+x22+x23+x24=60;
subject to A3:  x31+x32+x33+x34=80;
subject to B1:  x11+x21+x31=110;
```

```
subject to B2:  x12+x22+x32=40;
subject to B3:  x13+x23+x33=60;
subject to B4:  x14+x24+x34=80;
solve;
display z;
display x11, x12, x13, x14;
display x21, x22, x23, x24;
display x31, x32, x33, x34;
```

Результат виконання на NEOS-сервері:

```
Gurobi 5.5.0: outlev=1
threads=4
Optimize a model with 7 rows, 12 columns and
24 nonzeros
Presolve time: 0.00s
Presolved: 7 rows, 12 columns, 24 nonzeros

Iteration    Objective       Primal Inf.
Dual Inf.      Time
       0    4.8000000e+02   3.000000e+02
0.000000e+00      0s
       3    7.2000000e+02   0.000000e+00
0.000000e+00      0s

Solved in 3 iterations and 0.00 seconds
Optimal objective  7.200000000e+02
Gurobi 5.5.0: optimal solution; objective
720
3 simplex iterations
z = 720

x11 = 90
x12 = 0
x13 = 60
x14 = 0
```

```
x21 = 0
x22 = 0
x23 = 0
x24 = 60

x31 = 20
x32 = 40
x33 = 0
x34 = 20
```

Результат виконання на AMPL:

```
MINOS 5.5: optimal solution found.
5 iterations, objective 720
z = 720

x11 = 90
x12 = 0
x13 = 60
x14 = 0

x21 = 0
x22 = 0
x23 = 0
x24 = 60

x31 = 20
x32 = 40
x33 = 0
x34 = 20
```

3.4.3. Математична модель у вигляді задачі лінійного програмування загального виду та її AMPL-код

Ця задача включає $12 = 3 * 4$ змінних та $7 = 3 + 4$ обмежень. Вона має такий вигляд:

$$Z = c^T x \to \min \tag{3.4.1}$$

при обмеженнях:

$$Ax = b, \tag{3.4.2}$$
$$x \geq 0, \tag{3.4.3}$$

де вектори c, b та матриця A є такими:

$c = (4,4,2,5,5,3,1,2,2,1,4,2)^T$; $b = (150,60,80,110,40,60,80)^T$;

$$A = \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}.$$

AMPL-код для моделі (3.4.1) – (3.4.3) має такий вигляд:

```
param m;
param n;

set I=1..m;
set J= 1..n;
param c{J};
param b{I};
param a{I,J};
```

```
var x{J}>=0;
minimize z: sum {j in 1..n} c[j] * x[j];
subject to con{i in 1..m}: sum {j in 1..n}
a[i,j] * x[j] = b[i];
data;
param m:=7;
param n:=12;
param c [1] 4 [2] 4 [3] 2 [4] 5 [5] 5 [6] 3
[7] 1 [8] 2 [9] 2 [10] 1 [11] 4 [12] 2;
param b [1] 150 [2] 60 [3] 80 [4] 110 [5] 40
[6] 60 [7] 80;
param a:     1 2 3 4 5 6 7 8 9 10 11 12 :=
         1   1 1 1 1 0 0 0 0 0  0  0  0
         2   0 0 0 0 1 1 1 1 0  0  0  0
         3   0 0 0 0 0 0 0 0 1  1  1  1
         4   1 0 0 0 1 0 0 0 1  0  0  0
         5   0 1 0 0 0 1 0 0 0  1  0  0
         6   0 0 1 0 0 0 1 0 0  0  1  0
         7   0 0 0 1 0 0 0 1 0  0  0  1;

solve;
display z, x;
```

Результат виконання на NEOS-сервері:

```
Gurobi 5.5.0: outlev=1
threads=4
Optimize a model with 7 rows, 12 columns and
24 nonzeros
Presolve time: 0.00s
Presolved: 7 rows, 12 columns, 24 nonzeros
Iteration    Objective       Primal Inf.
Dual Inf.      Time
       0    4.8000000e+02   3.000000e+02
0.000000e+00      0s
```

```
       3    7.2000000e+02   0.000000e+00
0.000000e+00      0s

Solved in 3 iterations and 0.00 seconds
Optimal objective  7.200000000e+02
Gurobi 5.5.0: optimal solution; objective
720
3 simplex iterations
z = 720
x [*] :=
 1  90
 2   0
 3  60
 4   0
 5   0
 6   0
 7   0
 8  60
 9  20
10  40
11   0
12  20
;
```

Результат виконання на AMPL:

```
MINOS 5.5: optimal solution found.
5 iterations, objective 720
z = 720
x [*] :=
 1  90
 2   0
 3  60
 4   0
 5   0
 6   0
```

```
 7   0
 8  60
 9  20
10  40
11   0
12  20
;
```

3.4.4. Загальна математична модель та її AMPL-код

Для транспортної задачі з m постачальниками та n споживачами загальна математична модель має такий вигляд:

$$Z = \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij} \cdot x_{ij} \to \min \qquad (3.4.4)$$

при обмеженнях:

$$\sum_{j=1}^{n} x_{ij} = a_i \ \left(i = \overline{1,m}\right), \qquad (3.4.5)$$

$$\sum_{i=1}^{m} x_{ij} = b_j \ \left(j = \overline{1,n}\right), \qquad (3.4.6)$$

$$x_{ij} \geq 0 \ \left(i = \overline{1,m};\ j = \overline{1,n}\right), \qquad (3.4.7)$$

де x_{ij} – кількість продукції, що перевозиться від i-го постачальника до j-го споживача; c_{ij} – вартість перевезення одиниці продукції від i-го постачальника до j-го споживача; a_i – запаси продукції i-го постачальника; b_j – попит на продукцію j-го споживача.

AMPL-код для задачі (3.4.4) – (3.4.7) має такий вигляд:

```
set I;    # постачальники
set J;    # споживачі
```

```
param A {I} >= 0; # продукти постачальників
param B {J} >= 0;   # продукти споживачів

check: sum {i in I} A[i] = sum {j in J}
B[j];

param c {I,J} >= 0;   # транспортні затрати

var x {i in I, j in J} >= 0;
minimize z:  sum {i in I, j in J} c[i,j] *
x[i,j];
subject to Supply {i in I}:  sum {j in J}
x[i,j] = A[i];
subject to Demand {j in J}:  sum {i in I}
x[i,j] = B[j];
data;
param:  I:  A :=  # постачальники та їх
продукти
        A1   150
        A2    60
        A3    80;
param:  J:  B :=  # споживачі та їх
продукти
        B1    110
        B2     40
        B3     60
        B4     80;
param c:
        B1  B2  B3 B4  :=
   A1    4    4    2    5
   A2    5    3    1    2
   A3    2    1    4    2   ;
 display I; display J;
display A; display B;
display c;
```

```
solve;
display z;
display x;
```

Результат виконання на NEOS-сервері:

```
set I := A1 A2 A3;
set J := B1 B2 B3 B4;

A [*] :=
A1  150
A2   60
A3   80
;

B [*] :=
B1  110
B2   40
B3   60
B4   80
;
c :=
A1 B1   4
A1 B2   4
A1 B3   2
A1 B4   5
A2 B1   5
A2 B2   3
A2 B3   1
A2 B4   2
A3 B1   2
A3 B2   1
A3 B3   4
A3 B4   2
;
12 variables, all linear
```

```
7 constraints, all linear; 24 nonzeros
       7 equality constraints
1 linear objective; 12 nonzeros.

Gurobi 5.5.0: outlev=1
threads=4
Optimize a model with 7 rows, 12 columns and
24 nonzeros
Presolve time: 0.00s
Presolved: 7 rows, 12 columns, 24 nonzeros

Iteration    Objective       Primal Inf.
Dual Inf.      Time
       0    4.8000000e+02   3.000000e+02
0.000000e+00      0s
       3    7.2000000e+02   0.000000e+00
0.000000e+00      0s

Solved in 3 iterations and 0.00 seconds
Optimal objective  7.200000000e+02
Gurobi 5.5.0: optimal solution; objective
720
3 simplex iterations
z = 720
x :=
A1 B1   90
A1 B2    0
A1 B3   60
A1 B4    0
A2 B1    0
A2 B2    0
A2 B3    0
A2 B4   60
A3 B1   20
A3 B2   40
A3 B3    0
```

```
A3 B4   20
;
```

Результат виконання на AMPL:

```
set I := A1 A2 A3;
set J := B1 B2 B3 B4;
A [*] :=
A1  150
A2   60
A3   80
;
B [*] :=
B1  110
B2   40
B3   60
B4   80
;
c :=
A1 B1   4
A1 B2   4
A1 B3   2
A1 B4   5
A2 B1   5
A2 B2   3
A2 B3   1
A2 B4   2
A3 B1   2
A3 B2   1
A3 B3   4
A3 B4   2
;
MINOS 5.5: optimal solution found.
5 iterations, objective 720
z = 720
```

```
x :=
A1 B1    90
A1 B2     0
A1 B3    60
A1 B4     0
A2 B1     0
A2 B2     0
A2 B3     0
A2 B4    60
A3 B1    20
A3 B2    40
A3 B3     0
A3 B4    20
;
```

3.4.5 Математична модель у вигляді задачі нелінійного програмування та її AMPL-код

Для m постачальників та n споживачів математична модель має такий вигляд:

$$Z_1 = \sum_{i=1}^{m}\sum_{j=1}^{n} \varepsilon \left(x_{ij} + \frac{c_{ij}}{2\varepsilon} \right)^2 \to \min \qquad (3.4.8)$$

при обмеженнях:

$$\sum_{j=1}^{n} x_{ij} = a_i \ \left(i = \overline{1,m} \right), \qquad (3.4.9)$$

$$\sum_{i=1}^{m} x_{ij} = b_j \ \left(j = \overline{1,n} \right), \qquad (3.4.10)$$

$$x_{ij} \geq 0 \ \left(i = \overline{1,m};\ j = \overline{1,n} \right). \qquad (3.4.11)$$

При малих значеннях ε оптимальний план задачі співпадає із оптимальним планом задачі (3.4.4) – (3.4.7), а оптимальні

значення цільових функцій Z_1^* та Z^* відрізняються на величину

$$Z_1^* - Z^* = \sum_{i=1}^{m}\sum_{j=1}^{n} \varepsilon \left(x_{ij}^*\right)^2 + \frac{1}{4\varepsilon}\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}^2 .$$

AMPL-код для задачі (3.4.8) – (3.4.11) має наступний вигляд:

```
set I;
set J;
param A {I} >= 0;
param B {J} >= 0;
check: sum {i in I} A[i] = sum {j in J}
B[j];
param c {I,J} >= 0;
param eps :=  0.001;
var x {i in I, j in J} >= 0;
minimize z1: sum {i in I, j in J}
eps*(x[i,j]+c[i,j]/(2*eps))*(x[i,j]+c[i,j]/(
2*eps));
subject to Supply {i in I}: sum {j in J}
x[i,j] = A[i];
subject to Demand {j in J}: sum {i in I}
x[i,j] = B[j];
data;
param: I: A :=
       1  150
       2  60
       3  80;
param: J: B :=
       1  110
       2  40
       3  60
       4  80;
param c:
        1    2    3    4  :=
```

```
   1    4    4    2    5
   2    5    3    1    2
   3    2    1    4    2;
solve;
display z1;
display z1 - sum {i in I, j in J}
eps*x[i,j]*x[i,j]
- sum {i in I, j in J}
eps*(c[i,j]/(2*eps))*(c[i,j]/(2*eps));
display x;
```

Результат виконання на NEOS-сервері:

```
Gurobi 5.5.0: optimal solution; objective
31987.7
8 barrier iterations
No basis.
z1 = 31987.7
z1 - sum{i in I, j in J}
eps*(c[i,j]/(2*eps))*(c[i,j]/(2*eps)) -
  sum{i in I, j in J} eps*x[i,j]*x[i,j] =
720
x :=
1 1   90
1 2    2.70289e-08
1 3   60
1 4    1.4378e-08
2 1    2.75146e-08
2 2    2.0718e-08
2 3    4.85608e-08
2 4   60
3 1   20
3 2   40
3 3    2.25829e-08
3 4   20
;
```

Результат виконання на AMPL:

```
MINOS 5.5: optimal solution found.
7 iterations, objective 31987.7
Nonlin evals: obj = 18, grad = 17.
z1 = 31987.7
z1 - sum{i in I, j in J}
eps*(c[i,j]/(2*eps))*(c[i,j]/(2*eps)) -
  sum{i in I, j in J} eps*x[i,j]*x[i,j] =
720

x :=
1 1   90
1 2    0
1 3   60
1 4    0
2 1    0
2 2    0
2 3    1.94588e-15
2 4   60
3 1   20
3 2   40
3 3    0
3 4   20
;
```

3.4.6. Транспортні задачі для розв'язання

У пунктах постачання A_1, A_2, A_3 є однорідний вантаж в обсязі a_1, a_2, a_3 одиниць відповідно; цей вантаж треба транспортувати у пункти B_1, B_2, B_3, B_4, B_5 в обсязі відповідно b_1, b_2, b_3, b_4, b_5 одиниць. Потреби замовника (в умовних одиницях), запаси вантажу в кожному пункті постачання (у тих же одиницях) і тарифи (вартість

перевезення одиниці вантажу з даного пункту постачання даному замовнику) вказані в таблиці.

Потрібно спланувати перевезення так, щоб загальна сума вартості перевезень була найменшою.

Задача 1.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	2	3	4	2	4	140
A_2	8	4	1	4	1	180
A_3	3	7	3	1	2	160
Потреби	60	70	120	130	100	480

Задача 2.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	4	5	2	8	6	115
A_2	3	1	9	7	3	175
A_3	9	6	7	2	1	130
Потреби	70	220	40	30	60	420

Задача 3.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	3	8	7	11	15	260
A_2	14	3	1	8	6	400
A_3	9	5	16	7	12	240
Потреби	180	200	190	230	100	900

Задача 4.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	2	10	15	14	4	150
A_2	3	7	12	5	8	170
A_3	21	18	6	13	16	260
Потреби	100	90	160	150	80	580

Задача 5.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	14	8	17	5	3	120
A_2	21	10	7	11	6	180
A_3	3	5	8	4	9	230
Потреби	70	120	105	125	110	530

Задача 6.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	12	9	7	11	6	175
A_2	4	3	12	2	8	165
A_3	5	17	9	4	11	180
Потреби	90	120	110	130	70	520

Задача 7.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	3	8	7	11	15	260
A_2	14	3	1	8	6	400
A_3	9	5	16	7	12	240
Потреби	180	200	190	230	100	900

Задача 8.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	2	4	11	5	3	250
A_2	8	17	13	7	6	300
A_3	14	10	5	8	9	270
Потреби	120	200	190	230	80	820

Задача 9.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	21	18	14	3	6	370
A_2	7	11	10	5	12	450
A_3	4	8	12	8	13	430
Потреби	300	230	330	290	100	1250

Задача 10.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	3	10	15	17	9	560
A_2	2	16	3	15	4	570
A_3	8	5	12	14	7	620
Потреби	350	300	350	500	250	1750

Задача 11.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	11	4	15	1	2	350
A_2	20	9	7	14	5	350
A_3	18	10	3	8	6	300
Потреби	180	220	230	270	100	1000

Задача 12.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	2	4	5	11	3	400
A_2	12	8	6	14	11	370
A_3	10	15	7	9	18	380
Потреби	250	200	290	260	150	1150

Задача 13.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	20	3	9	15	35	300
A_2	14	10	12	20	46	150
A_3	25	11	16	19	48	250
Потреби	150	100	150	100	200	700

Задача 14.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	7	3	9	15	35	180
A_2	14	10	12	20	46	100
A_3	15	11	14	17	39	120
Потреби	100	60	90	70	80	400

Задача 15.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	12	42	15	17	9	250
A_2	16	16	13	15	4	125
A_3	16	34	22	14	10	225
Потреби	120	110	80	190	100	600

Задача 16.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	7	20	3	15	35	200
A_2	3	16	10	20	41	100
A_3	15	25	11	19	40	200
Потреби	80	100	70	130	120	500

Задача 17.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	1	20	3	9	35	220
A_2	3	14	10	12	42	100
A_3	15	31	21	16	47	180
Потреби	70	110	80	100	140	500

Задача 18.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	7	20	3	9	15	210
A_2	3	14	10	12	20	140
A_3	15	25	11	16	19	150
Потреби	80	120	90	110	100	500

Задача 19.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	11	7	3	9	15	170
A_2	12	3	10	12	20	120
A_3	19	15	11	16	19	110
Потреби	90	60	100	80	70	400

Задача 20.

Пункти постачання	Пункти призначення					Запаси
	B_1	B_2	B_3	B_4	B_5	
A_1	11	1	20	9	15	250
A_2	12	3	14	12	20	200
A_3	13	15	25	16	19	150
Потреби	75	125	150	50	200	600

IV. Двоетапна транспортна задача

При формулюванні класичної двоетапної транспортної задачі [13, 14] мається на увазі, що вантаж перевозиться від постачальників до споживачів тільки через проміжні пункти. Схема функціонування перевезень вантажу наведена на рис. 4.1. В якості проміжних пунктів можуть виступати посередницькі фірми та різного роду сховища (склади).

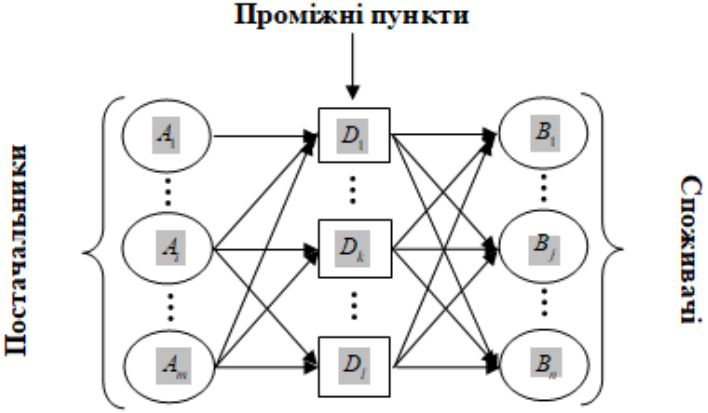

Рис. 4.1. Система «постачальники – проміжні пункти – споживачі» у двоетапній транспортній задачі

У розділі IV розглянуто достатньо універсальний спосіб розв'язання двоетапної транспортної задачі з використанням сучасного програмного забезпечення [15]. Він використовує опис задачі на мові моделювання AMPL [4] та залежить тільки від того, яку із відомих програм буде вибрано для розв'язання задачі лінійного програмування.

Матеріал розділу викладено в такому порядку. У підрозділі 4.1 описано формулювання двоетапної транспортної задачі та наведено її властивості. У підрозділі 4.2 наведено опис задачі на мові моделювання AMPL та результати розрахунків для тестового прикладу. У підрозділі

4.3 описано квадратичну двоетапну транспортну задачу та доведено єдиність її розв'язку.

4.1. Формулювання задачі та її властивості

Нехай в m пунктах постачання $A_1, ..., A_m$ є $a_1, ..., a_m$ одиниць продукції, яку потрібно перевезти до n споживачів $B_1, ..., B_n$, задовольнивши їх потреби $b_1, ..., b_n$. Для транспортування продукції від постачальників до споживачів можна задіяти l проміжних пунктів $D_1, ..., D_l$. Витрати на перевезення одиниці продукції з пункту постачання A_i до проміжного пункту D_k позначимо c_{ik}, з проміжного пункту D_k до споживача B_j – c_{kj}. Кількість продукції, яка перевозиться від постачальника A_i до проміжного пункту D_k, позначимо x_{ik}, кількість продукції від проміжного пункту до споживача – y_{kj}.

Двоетапна транспортна задача має такий вигляд: знайти

$$f^* = \min_{x,y} \left\{ f(x,y) = \sum_{i=1}^{m} \sum_{k=1}^{l} c_{ik} x_{ik} + \sum_{k=1}^{l} \sum_{j=1}^{n} c_{kj} y_{kj} \right\} \qquad (4.1)$$

за обмежень

$$\sum_{k=1}^{l} x_{ik} = a_i, \quad i = 1, \ldots, m, \qquad (4.2)$$

$$\sum_{k=1}^{l} y_{kj} = b_j, \quad j = 1, \ldots, n, \qquad (4.3)$$

$$\sum_{i=1}^{m} x_{ik} - \sum_{j=1}^{n} y_{kj} = 0, \quad k = 1, \ldots, l, \qquad (4.4)$$

$$x_{ik} \geq 0,\ y_{kj} \geq 0, \quad i = 1, \ldots, m,\ k = 1, \ldots, l,\ j = 1, \ldots, n. \qquad (4.5)$$

Задача (4.1) – (4.5) є задачею лінійного програмування, яка містить $m \times l + l \times n$ змінних x_{ik}, y_{kj} та $m + n + l$ обмежень загального виду, не враховуючи обмежень (4.5) – умов на

невід'ємність усіх змінних. Цільова функція (4.1) задає сумарні витрати на транспортування продукції від постачальників до споживачів через проміжні пункти.

Обмеження (4.2) означають транспортування усієї продукції $a_1, ..., a_m$ із пунктів постачання до проміжних пунктів, а обмеження (4.3) – що споживачам потрібно доставити необхідну продукцію $b_1, ..., b_n$ з проміжних пунктів. Обмеження (4.4) задають умови на те, щоб вся продукція яка приходить від постачальників до кожного проміжного пункту, була обовязково відправлена споживачам. Це визначає умови на сумісність системи обмежень (4.2) – (4.5) – системи лінійних рівностей та лінійних нерівностей. Звідси випливає наступне твердження.

Лема 4.1 [15]. Система обмежень (4.2) – (4.5) є несумісною, якщо $\sum_{i=1}^{m} a_i \neq \sum_{j=1}^{n} b_j$.

Лема 4.1 дає можливість перевірити вхідні дані на предмет їх коректності для сформульованої двоетапної транспортної задачі. Якщо не виконується умова леми, то тоді виконується умова $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$. У цьому випадку система (4.2) – (4.5) є сумісною і можна переходити до розв'язання задачі (4.1) – (4.5).

4.2 AMPL-реалізація задачі та тестовий приклад

AMPL-код для опису двоетапної транспортної задачі (4.1) – (4.5) має наступний вигляд:

```
param m>=2; #Кількість постачальників (пункти A)
param l>=1; #Кількість проміжних пунктів (пункти D)
param n>=2; #Кількість споживачів (пункти B)
#Вартості перевезення одиниці продукції:
param cik{i in 1..m, k in 1..l} >= 0; #від A до D
param ckj{k in 1..l, j in 1..n} >= 0; #від D до B
```

```
#Інші дані
param a{i in 1..m} >= 0; #Продукція в A
param b{j in 1..n} >= 0; #Потреби в B
check: sum{i in 1..m} a[i] = sum{j in 1..n}
b[j];#умова леми 4.1

#Невідомі (продукція, яку потрібно перевезти)
var x{i in 1..m, k in 1..l} >=0; #від A до D
var y{k in 1..l, j in 1..n} >=0; #від D до B
minimize f_opt: #Мінімізувати витрати на
перевезення продукції:
sum{i in 1..m, k in 1..l} cik[i,k]*x[i,k]+ #від A
до D
sum{k in 1..l, j in 1..n} ckj[k,j]*y[k,j]; #від D
до B
subject to #за обмежень
con2 {i in 1..m}: #перевезення продукції з A до D
sum{k in 1..l}x[i,k] = a[i];
con3 {j in 1..n}: #задоволення потреб B з D
sum{k in 1..l}y[k,j] = b[j];
con4 {k in 1..l}: #всю продукцію потрібно доставити
в B
sum{i in 1..m}x[i,k]-sum{j in 1..n}y[k,j]=0;
```

Тут оператор **param** використано для опису розмірів та даних задачі; оператор **var** – для опису змінних задачі, де враховано їх невід'ємність; оператор **check** – для перевірки сумісності обмежень (4.2) – (4.5), яка визначається лемою 4.1 та вимагає, щоб уся продукція постачальників була відправлена споживачам.

Якщо цей AMPL-код доповнити необхідними даними для конкретної задачі, то його можна використовувати для розв'язання двоетапних транспортних задач за допомогою стандартного програмного забезпечення для розв'язання задач лінійного програмування. Це можна зробити як за допомогою тих програм NEOS-сервера [2] із розділу «Linear Programming», для яких підтримуються вхідні формати даних

на AMPL, так і за допомогою комерційних або з вільним доступом версій AMPL.

Тестовий приклад [14]. *Виробниче об'єднання складається з трьох філіалів: A_1, A_2, A_3, які виготовляють однорідну продукцію в обсягах відповідно 1000, 1500 та 1200 одиниць на місяць. Ця продукція відправляється на два склади D_1 і D_2, а потім – до п'яти споживачів B_1, B_2, ..., B_5, попит яких становить відповідно 900, 700, 1000, 500 і 600 одиниць. Вартості перевезень одиниці продукції (в умовних одиницях) від виробників на склади, а потім – зі складів до споживачів наведені в наступних таблицях:*

$A \setminus D$	D_1	D_2
A_1	2	8
A_2	3	5
A_3	1	4

$D \setminus B$	B_1	B_2	B_3	B_4	B_5
D_1	1	3	8	5	4
D_2	2	4	5	3	1

Для цього прикладу опис вхідних даних задачі (4.1) – (4.5) реалізує AMPL-код:

```
data; #Блок вхідних даних, де задаємо:
param m := 3; #Кількість постачальників A
param l := 2; #Кількість проміжних пунктів D
param n := 5; #Кількість споживачів B
#Витрати на перевезення одиниці продукції:
param cik: 1 2 := #від A (↓) до D (→)
          1 2 8
          2 3 5
          3 1 4;
param ckj: 1 2 3 4 5 := #від D (↓) до B (→)
          1 1 3 8 5 4
          2 2 4 5 3 1;
param a:= #Продукція в A
1 1000 2 1500 3 1200; #A_1, A_2, A_3
param b:= #Потреби B
1 900 2 700 3 1000 #B_1, B_2, B_3
4 500 5 600; #B_4, B_5
```

```
solve; #Розв'язати задачу (4.1)-(4.5)
#Роздрукувати значення цільової функції та час розв'язання
display f_opt, _solve_time;
#Роздрукувати значення оптимальних змінних
display x; display y;
```

Результат розрахунку для тестового прикладу за допомогою відомої програми Gurobi [7] на NEOS-сервері має наступний вигляд:

```
16 variables, all linear
10 constraints, all linear; 32 nonzeros
        10 equality constraints
1 linear objective; 16 nonzeros.

Gurobi 8.1.0: optimal solution; objective 22100
1 simplex iterations
f_opt = 22100
_solve_time = 0.003998

x :=
1 1   1000
1 2      0
2 1      0
2 2   1500
3 1   1200
3 2      0
;

y :=
1 1   900
1 2   700
1 3   100
1 4   500
1 5     0
2 1     0
2 2     0
2 3   900
2 4     0
```

```
2 5   600
;

Gurobi 8.1.0: threads=4
outlev=1
Optimize a model with 10 rows, 16 columns and 32
nonzeros
Coefficient statistics:
  Matrix range     [1e+00, 1e+00]
  Objective range  [1e+00, 8e+00]
  Bounds range     [0e+00, 0e+00]
  RHS range        [5e+02, 2e+03]
Iteration    Objective       Primal Inf.    Dual
Inf.      Time
       0    2.2100000e+04   0.000000e+00
0.000000e+00      0s

Solved in 0 iterations and 0.00 seconds
Optimal objective  2.210000000e+04
Gurobi 8.1.0: optimal solution; objective 22100
```

Для тестового прикладу задача (4.1) – (4.5) має багато розв'язків. Один із них представлений на рис. 4.2. Для нього значення цільової функції дорівнює 22100 умовних одиниць, склад D_1 завантажений на 2200 одиниць, а склад D_2 – на 1500 одиниць.

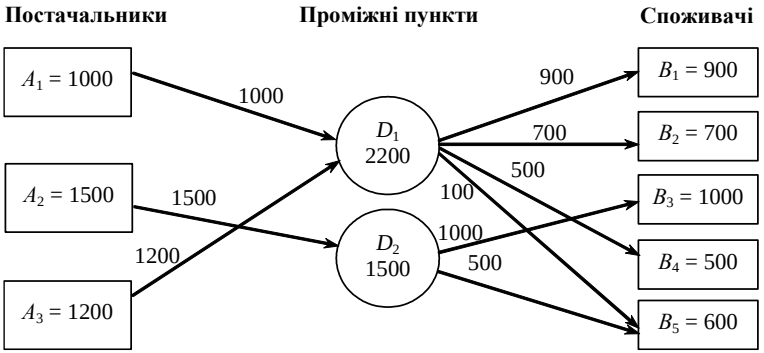

Рис. 4.2. Перший оптимальний план транспортування продукції

Другий розв'язок представлений на рис. 4.3 та характеризується більш рівномірною завантаженістю складів. Для нього значення цільової функції дорівнює 22100 умовних одиниць, склад D_1 завантажений на 2100 одиниць, а склад D_2 – на 1600 одиниць.

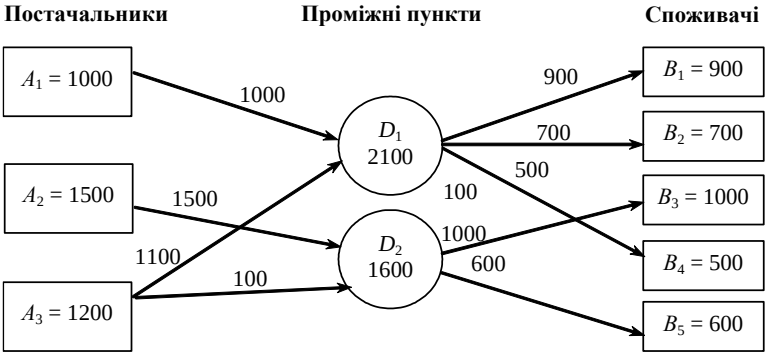

Рис. 4.3. Другий оптимальний план транспортування продукції

Неоднозначності оптимального розв'язку легко уникнути, якщо від задачі лінійного програмування (4.1) – (4.5) перейти

до задачі квадратичного програмування, де цільова функція буде строго опуклою квадратичною функцією. Таку задачу розглянемо нижче.

4.3. Квадратична двоетапна транспортна задача

Квадратичну двоетапну транспортну задачу розглянемо у такому вигляді: знайти

$$F^* = \min_{x,y} \left\{ \begin{array}{l} F(x,y) = \sum\limits_{i=1}^{m} \sum\limits_{k=1}^{l} (\varepsilon_{ik} x_{ik}^2 + c_{ik} x_{ik}) \\ + \sum\limits_{k=1}^{l} \sum\limits_{j=1}^{n} (\varepsilon_{kj} y_{kj}^2 + c_{kj} y_{kj}) \end{array} \right\} \qquad (4.6)$$

за обмежень

$$\sum_{k=1}^{l} x_{ik} = a_i, \quad i = 1,\ldots,m, \qquad (4.7)$$

$$\sum_{k=1}^{l} y_{kj} = b_j, \quad j = 1,\ldots,n, \qquad (4.8)$$

$$\sum_{i=1}^{m} x_{ik} - \sum_{j=1}^{n} y_{kj} = 0, \quad k = 1,\ldots,l, \qquad (4.9)$$

$$x_{ik} \geq 0,\ y_{kj} \geq 0, \quad i = 1,\ldots,m,\ k = 1,\ldots,l,\ j = 1,\ldots,n. \qquad (4.10)$$

Якщо $\varepsilon_{ik} > 0$ та $\varepsilon_{jk} > 0$, то цільова функція (4.6) є опуклою сепарабельною квадратичною функцією. Відповідна їй задача (4.6) – (4.10) є задачею опуклого квадратичного програмування. Якщо для всіх $i = 1,\ldots,m$, $k = 1,\ldots,l$, $j = 1,\ldots,n$, значення $\varepsilon_{ik} = 0$ та $\varepsilon_{jk} = 0$, то задача (4.6) – (4.10) переходить у задачу лінійного програмування (4.1) – (4.5). Тут лінійні обмеження (4.7) означають постачання усієї продукції $a_1, ..., a_m$ до проміжних пунктів, а лінійні обмеження (4.8) – доставку необхідної продукції $b_1, ..., b_n$ з проміжних пунктів. Лінійні обмеження (4.9) задають умови на те, щоб вся продукція яка приходить від постачальників до

кожного проміжного пункту, була обов'язково відправлена споживачам. Лінійні обмеження (4.10) задають умови на невід'ємність змінних x_{ik} та y_{kj}, $i=1,\ldots,m$, $k=1,\ldots,l$, $j=1,\ldots,n$. Для обмежень (4.7) – (4.10) справедлива лема 1.1, яка дає можливість перевірити коректність вхідних даних $a_1,\ldots,a_m$ та $b_1,\ldots,b_n$. Якщо виконується умова $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$, то система (4.7) – (4.10) є сумісною і можна переходити до розв'язання задачі (4.6) – (4.10).

Лема 1.2 Якщо $\varepsilon_{ik} > 0$, $\varepsilon_{jk} > 0$ для всіх $i=1,\ldots,m$, $k=1,\ldots,l$, $j=1,\ldots,n$ та виконується умова $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$, то задача (4.6)-(4.10) має єдиний розв'язок.

Доведення. Доведення проведемо методом від супротивного. Нехай $x^* = \left\{x_{ik}^*\right\}_{i=1,\ldots,m}^{k=1,\ldots,l}$, $y^* = \left\{y_{kj}^*\right\}_{k=1,\ldots,l}^{j=1,\ldots,m}$ та $x^{**} = \left\{x_{ik}^{**}\right\}_{i=1,\ldots,m}^{k=1,\ldots,l}$, $y^{**} = \left\{y_{kj}^{**}\right\}_{k=1,\ldots,l}^{j=1,\ldots,m}$ – два неспівпадаючі розв'язки задачі (4.6) – (4.10). Їм відповідає оптимальне значення цільової функції $f^* = f(x^*, y^*) = f(x^{**}, y^{**})$. Обидва розв'язки x^*, y^* та x^{**}, y^{**} задовольняють обмеженням (4.7) – (4.10), тобто

$$\sum_{k=1}^{l} x_{ik}^* = a_i, \quad \sum_{k=1}^{l} x_{ik}^{**} = a_i, \quad i=1,\ldots,m, \tag{4.11}$$

$$\sum_{k=1}^{l} y_{kj}^* = b_j, \quad \sum_{k=1}^{l} y_{kj}^{**} = b_j, \quad j=1,\ldots,n, \tag{4.12}$$

$$\sum_{i=1}^{m} x_{ik}^* - \sum_{j=1}^{n} y_{kj}^* = 0, \quad \sum_{i=1}^{m} x_{ik}^{**} - \sum_{j=1}^{n} y_{kj}^{**} = 0, \quad k=1,\ldots,l, \tag{4.13}$$

$$\begin{gathered} x_{ik}^* \geq 0,\ y_{kj}^* \geq 0,\ x_{ik}^{**} \geq 0,\ y_{kj}^{**} \geq 0, \\ i=1,\ldots,m,\ k=1,\ldots,l,\ j=1,\ldots,n. \end{gathered} \tag{4.14}$$

Точки $x^{***} = \lambda x^{*} + (1-\lambda)x^{**}$ та $y^{***} = \lambda y^{*} + (1-\lambda)y^{**}$, де $0 < \lambda < 1$, задовольняють системі обмежень (4.7) – (4.10). Дійсно, враховуючи рівності (4.11), для всіх $i = 1,\ldots,m$ отримуємо

$$\sum_{k=1}^{l} x_{ik}^{***} = \sum_{k=1}^{l} \left(\lambda x_{ik}^{*} + (1-\lambda) x_{ik}^{**} \right) =$$

$$= \lambda \sum_{k=1}^{l} x_{ik}^{*} + (1-\lambda) \sum_{k=1}^{l} x_{ik}^{**} = \lambda a_i + (1-\lambda) a_i = a_i,$$

що означає, що точка x^{***} задовольняє рівності (4.7). Враховуючи рівності (4.12), для всіх $j = 1,\ldots,n$ отримуємо

$$\sum_{k=1}^{l} y_{kj}^{***} = \sum_{k=1}^{l} \left(\lambda y_{kj}^{*} + (1-\lambda) y_{kj}^{**} \right) =$$

$$= \lambda \sum_{k=1}^{l} y_{kj}^{*} + (1-\lambda) \sum_{k=1}^{l} y_{kj}^{**} = \lambda b_j + (1-\lambda) b_j = b_j,$$

що означає, що точка y^{***} задовольняє рівностям (4.8). Аналогічно, використовуючи рівності (4.13) та нерівності (4.14), легко показати, що точки x^{***} та y^{***} задовольняють рівностям (4.9) та нерівностям (4.10).

Якщо $\varepsilon_{ik} \geq 0$ та $\varepsilon_{jk} \geq 0$, то для всіх $i = 1,\ldots,m$, $k = 1,\ldots,l$, $j = 1,\ldots,n$, квадратичні функції $F_{ik}(x_{ik}) = \varepsilon_{ik} x_{ik}^2 + c_{ik} x_{ik}$ та $F_{kj}(y_{kj}) = \varepsilon_{kj} y_{kj}^2 + c_{kj} y_{kj}$ є строго опуклими, а значить квадратична функція

$$F(x,y) = \sum_{i=1}^{m} \sum_{k=1}^{l} (\varepsilon_{ik} x_{ik}^2 + c_{ik} x_{ik}) + \sum_{k=1}^{l} \sum_{j=1}^{n} (\varepsilon_{kj} y_{kj}^2 + c_{kj} y_{kj}) \quad (4.15)$$

є також строго опуклою. Тому, якщо $0 < \lambda < 1$, то для $f(x^{***}, y^{***})$ – значення цільової функції в точці (x^{***}, y^{***}) справедливі наступні співвідношення

$$F(x^{***}, y^{***}) = F(\lambda x^{*} + (1-\lambda)x^{**}, \lambda y^{*} + (1-\lambda)y^{**}) <$$

$$< \lambda F(x^{*}, y^{*}) + (1-\lambda) F(x^{**}, y^{**}) = \lambda F^{*} + (1-\lambda) F^{*} = F^{*},$$

з яких випливає нерівність $F(x^{***}, y^{***}) < F^*$. Вона суперечить тому, що (x^*, y^*) та (x^{**}, y^{**}) є розв'язками задачі (4.6) – (4.10), так як в точці (x^{***}, y^{***}), яка задовольняє обмеженням (4.7) – (4.10), значення цільової функції $F(x^{***}, y^{***})$ є меншим за мінімальне значення F^*. Лема 4.2 доведена.

Частковим випадком задачі (4.13) – (4.17) є задача лінійного програмування (4.1) – (4.5). Їй відповідають $\varepsilon_{ik} = 0$ і $\varepsilon_{jk} = 0$ для всіх постачальників $i = 1,\ldots,m$, проміжних пунктів $k = 1,\ldots,l$ та споживачів $j = 1,\ldots,n$. Для задачі (4.1) – (4.5) точка мінімуму не завжди є єдиною (див. тестовий приклад, підрозділ 4.2). Якщо вибирати достатньо малими значення величин ε_{ik} та ε_{jk}, то це дозволяє наблизити розв'язок задачі (4.6) – (4.10) до одного з розв'язків задачі (4.1) – (4.5), якщо остання має багато розв'язків. При цьому як оптимальний розв'язок буде вибрана одна з внутрішніх точок множини оптимальних розв'язків задачі лінійного програмування (4.1) – (4.5), яка визначається вибором величин ε_{ik} та ε_{jk} у задачі квадратичного програмування (4.6) – (4.10).

4.4. Висновки до розділу IV

У даному розділі описано властивості двоетапної транспортної задачі (4.1) – (4.5) та наведено AMPL-код для її розв'язання за допомогою сучасного програмного забезпечення для задач лінійного програмування. Наведено результати розрахунку для задачі з багатьма розв'язками.

Розроблений AMPL-код та його модифікації планується використати для визначення оптимальних енергоємностей накопичувачів електричної енергії.

При цьому легко переконатися, що розв'язок задачі є єдиним, а якщо він не є єдиним, то знайти відповідний

єдиний граничний розв'язок за допомогою квадратичної задачі (4.6) – (4.10) з малими значеннями величин ε_{ik} і ε_{jk} для постачальників $i = 1,\ldots,m$, накопичувачів $k = 1,\ldots,l$ та споживачів $j = 1,\ldots,n$ електричної енергії.

Другий спосіб отримати граничний розв'язок задачі лінійного програмування (4.1) – (4.5) пов'язаний із заміною цільової функції (4.6) на цільову функцію у такому вигляді

$$\varphi^* = \min_{x,y}\left\{\varphi(x,y) = \sum_{i=1}^{m}\sum_{k=1}^{l}\varepsilon_{ik}x_{ik}^2 + \sum_{k=1}^{l}\sum_{j=1}^{n}\varepsilon_{kj}y_{kj}^2\right\} \tag{4.16}$$

та доповненням обмежень (4.13) – (4.17) одним лінійним обмеженням

$$\sum_{i=1}^{m}\sum_{k=1}^{l}c_{ik}x_{ik} + \sum_{k=1}^{l}\sum_{j=1}^{n}c_{kj}y_{kj} \le f^*, \tag{4.17}$$

де f^* – мінімальні сумарні витрати на транспортування продукції від постачальників до споживачів через проміжні пункти. Відмітимо, що для розробленого AMPL-коду легко врахувати нижні та верхні межі на кількість продукції, що перевозиться. Для цього достатньо обмеження (4.5), або (4.17) замінити на таке обмеження

$$\begin{aligned} &x_{ik}^{low} \le x_{ik} \le x_{ik}^{up},\ y_{kj}^{low} \le y_{kj} \le y_{kj}^{up}, \\ &i = 1,\ldots,m,\ k = 1,\ldots,l,\ j = 1,\ldots,n, \end{aligned} \tag{4.18}$$

де x_{ik}^{low}, x_{ik}^{up} та y_{kj}^{low}, y_{kj}^{up} – нижні та верхні межі на кількість продукції, яка перевозиться від постачальника A_i до проміжного пункту D_k, та на кількість продукції від проміжного пункту D_k до споживача B_j, відповідно.

IV. Дві ЛП-задачі з булевими змінними для відмовостійкої мережі

Нехай $N = (V, A)$ – орієнтована мережа з множиною вершин V та множиною дуг A. Пропускну здатність дуги $a = (i, j) \in A$, спрямованої з вершини $i \in V$ у вершину $j \in V$, позначимо y_a. Поломкою дуги $a \in A$ назвемо зменшення її пропускної здатності за правилом: $y'_a = \mu y_a$, де $\mu \in [0,1)$. Якщо $\mu = 0$, то $y'_a = 0$ і це рівносильно відмові дуги $a \in A$.

Нехай для орієнтованої мережі $N = (V, A)$ задані:

1. **Мережевий трафік:** множина трійок $\{d_k, s_k, r_k : k \in K\}$, де d_k – об'єм потоку від вершини-відправника $s_k \in V$ до вершини-одержувача $r_k \in V$, $K = \{1, 2, ..., k_{\max}\}$.
2. **Сценарій відмов:** множина поломок T, де кожна поломка $t \in T$ визначається пропускними здатностями дуг $y_a^t = \mu_{at} y_a$, $\mu_{at} \in [0,1], \forall a \in A$.

Мережу $N = (V, A)$ назвемо відмовостійкою, якщо значення пропускних здатностей дуг $y = \{y_a : a \in A\}$ такі, що **мережевий трафік** можна виконати завжди, якщо відбудеться тільки одна, але довільна, поломка з сценарію відмов.

Нехай для кожної дуги $a \in A$ задані: y_a^0 – значення вже існуючої пропускної здатності, y_a^{low}, y_a^{up} – границі знизу і зверху на додану до існуючої пропускну здатність, c_a – вартість пересилання одиничного потоку, F_a – витрати на створення доданої пропускної здатності.

Потрібно знайти не більше, ніж N_u дуг, таких, що, якщо їх пропускні здатності додати до вже існуючих, то мережа

буде відмовостійкою, а сумарні витрати на створення цих дуг та пересилку потоків по них будуть мінімальними.

Розглянемо дві задачі лінійного програмування (ЛП-задачі) з булевими змінними [12]. В **задачі А** для пересилання потоків може використовуватися будь-який можливий шлях мережі. **Задача Р** визначається тим, що для пересилання потоків використовуються тільки шляхи із заданої множини шляхів $P = \cup_{k \in K} P_k$, де P_k – множина шляхів для потоку k.

Нехай змінні $y = \{y_a : a \in A\}$ – невідомі значення пропускних здатностей доданих дуг, їм відповідають булеві змінні $u = \{u_a : a \in A\}$, де u_a дорівнює одиниці, якщо дуга $a \in A$ додається до мережі, і нулю, в іншому випадку; змінні $x = \{x_{akt}, \forall a, k, t\}$ – невідомі обсяги потоків в мережі, де x_{akt} – обсяг потоку k по дузі a для поломки t.

Нехай A_i^+ та A_i^- – множини вхідних та вихідних дуг із вер-шини $i \in N$. Тоді формулювання **задачі А має** вигляд: знайти

$$f_A^* = \min_{x,y,u} \sum_{a \in A} c_a y_a + \sum_{a \in A} F_a u_a \qquad (1A)$$

при обмеженнях:

$$\sum_{k \in K} x_{akt} \le \mu_{at} (y_a^0 + y_a), \quad \forall a \in A, t \in T, \qquad (2A)$$

$$\sum_{a \in A_i^+} x_{akt} - \sum_{a \in A_i^-} x_{akt} = \begin{cases} d_k, & \text{якщо } i = s_k, \\ -d_k, & \text{якщо } i = r_k, \\ 0, & \text{інакше,} \end{cases} \quad \forall i \in V, k \in K, t \in T, \qquad (3A)$$

$$y_a^{low} u_a \le y_a \le y_a^{up} u_a, \quad \forall a \in A, \quad \sum_{a \in A} u_a \le N_u, \qquad (4A)$$

$$x_{akt} \ge 0, \forall a \in A, k \in K, t \in T, \quad u_a = 0 \vee 1, \forall a \in A. \qquad (5A)$$

Цільова функція (1А) дорівнює сумі затрат на пересилку потоків по доданих дугах і витрат на їх створення.

Обмеження (2А) означають, що сумарні потоки по кожній дузі не перевищують її пропускної здатності при довільній поломці з сценарію відмов; обмеження (3А) відповідають за виконання мережевого трафіку; обмеження (4А) відповідають за вибір не більше N_u дуг та гарантують, що їх пропускні здатності вибираються з діапазону $[y_a^{low}, y_a^{up}]$; обмеження (5А) відповідають за невід'ємність потокових змінних x та булевих змінних u.

Формулювання **задачі Р** має такий вигляд:
знайти

$$f_P^* = \min_{z,y,u} \sum_{a\in A} c_a y_a + \sum_{a\in A} F_a u_a \qquad (1P)$$

при обмеженнях:

$$\sum_{k\in K}\sum_{p\in P_k} \delta_{kpa} z_{kpt} \le \mu_{at}(y_a^0 + y_a), \quad \forall a \in A, t \in T, \quad (2P)$$

$$\sum_{p\in P_k} z_{kpt} = d_k \quad \forall k \in K, t \in T, \qquad (3P)$$

$$y_a^{low} u_a \le y_a \le y_a^{up} u_a, \quad \forall a \in A, \quad \sum_{a\in A} u_a \le N_u, \quad (4P)$$

$$z_{kpt} \ge 0, \forall\, k \in K,\ p \in P, t \in T, \quad u_a = 0 \vee 1, \forall a \in A, \qquad (5P)$$

де

$$\delta_{kpa} = \begin{cases} 1, & \text{якщо шлях } p \in P_k \text{ використовує дугу } a, \\ 0, & \text{інакше.} \end{cases}$$

Тут інші змінні $z = \{z_{kpt}, \forall k, p, t\}$ (z_{kpt} – обсяг потоку $k \in K$ по шляху $p \in P_k$ для поломки $t \in T$), а зміст обмежень (2Р)–(5Р) такий же, як і в **задачі А**.

Задачі А та **Р** належать до задач великих розмірів, в яких кількість змінних та обмежень вимірюється мільйонами навіть для мереж середніх розмірів. Так, у **задачі А** кількість змінних дорівнює $N_A = |A|*|K|*|T| + 2|A|$, а кількість обмежень – $M_A = |V|*|K|*|T| + |A|*|T| + |A| + 1$. Для середніх розмірів мережі задача (1А)-(5А) є задачею великих розмірів.

Наприклад, якщо $|V| \approx 40$, $|A| \approx 50$, $|T| \approx 50$ та $|K| \approx 1000$, то $N_A \approx 2.500.000$, $M_A \approx 2.000.000$. У **задачі Р** кількість змінних дорівнює $N_P = |A| + |P| * |T| + 2|A|$, а кількість обмежень – $M_P = |K| * |T| + |A| * |T| + 3|A| + 1$. Якщо $|A| \approx 50$, $|T| \approx 50$, $|P| \approx 2000$, $|K| \approx 1000$, то задача (1Р)-(5Р) має $N_P \approx 100.000$ змінних та $M_P \approx 50.000$. обмежень.

Наведені ЛП-задачі з булевими змінними узагальнюють ЛП-задачі для знаходження пропускних здатностей дуг відмовостійкої орієнтованої мережі [11, ст. 99–110].

Для середнього розміру мереж їх можна ефективно розв'язувати за допомогою NEOS-сервера та програми Gurobi. Це підтверджують обчислювальні експерименти для **задачі А**, коли неорієнтована мережа net-48-56 включала 48 вершин та 56 ребер. У таблиці 1 наведено розміри ЛП-задач без поломок та для однієї, двох і трьох поломок.

Таблиця 1. Розміри задач для мережі net-48-56

Задача	Поломки	Змінні	Обмеження	Ненул. коеф.
net-48-56	0	252.784	108.400	758.128
net-48-56-1	1	500.944	216.798	1.502.718
net-48-56-2	2	749.104	325.196	2.247.308
net-48-56-3	3	997.264	433.594	2.991.898

У таблиці 2 наведено затрати програми Gurobi (кількість ітерацій, час роботи програми) на розв’язання задач із таблиці 1.

Таблиця 2. Програма Gurobi у задачах Нурмінського

Задача	Ітерації	Час (сек)
net-48-56	47.099	1.74
net-48-56-1	175.646	264.31

net-48-56-2	267.152	935.75
net-48-56-3	382.015	498.68

Із таблиці 2 бачимо, що розв’язок ЛП-задачі з мільйоном змінних та п’ятистам обмежень отримано менше, ніж за десять хвилин. Тому застосування NEOS-інтерфейсу є перспективним, якщо потрібно досить швидко розв’язати ту чи іншу оптимізаційну задачу для мережі середніх розмірів.

ЛІТЕРАТУРА

1. NEOS Server [електронний ресурс]: http://www.neos-server.org. Режим доступу: вільний.
2. NEOS Solver [електронний ресурс]: http://www.neos-server.org/neos/solvers/. Режим доступу: вільний.
3. Щербина О.А. Краткое введение в AMPL – современный алгебраический язык моделирования. Препринт. 2012. 329 с.
4. Fourer R. AMPL, A Modeling Language for Mathematical Programming, Second Edition / R. Fourer, D. Gay, B. Kernighan. Belmont: Duxburry Press, 2003. 517 p.
5. Dantzig, G.B. Linear Programming and Extensions. Princeton University Press, 1991. 656 p.
6. Бейко І.В., Зінько П.М., Наконечний О.Г. Задачі, методи і алгоритми оптимізації. Навчальний посібник. Рівне: Національний університет водного господарства та природокористування (НУВГП), 2011. 624 с.
7. Gurobi Optimization, Inc., Gurobi Optimizer Reference Manual, 2014, http://www.gurobi.com.
8. Murtagh, B.A. Advanced Linear Programming: Theory and Practice. McGraw-Hill Inc.,US, 1981. 202 p.
9. Polyak, B.T. Introduction to Optimization. Optimization Software, 1987. Series: Translations series in mathematics and engineering. 438 p.
10. Михалевич В.С., Шор Н.З., Галустова Л.А., Журбенко Н.Г., Момот А.И., Сибирко А.Т., Трубин В.А., Юн Г.Н. Вычислительные методы выбора оптимальных проектных решений. К.: Наук. думка, 1977. 178 с.
11. Шор Н.З., Сергієнко І.В., Шило В.П., Стецюк П.І. та ін. Задачі оптимального проектування надійних мереж. К.: Наукова думка, 2005. 230 с.
12. Сергиенко И.В., Стецюк П.И. Две ЛП-задачи с булевыми переменными для отказоустойчивой сети. Інформатика та системні науки (ІСН-2014): матеріали V Всеукр. наук.-

практ. конф. (м. Полтава, 13-15 березня 2014 року). Полтава: ПУЕТ, 2014. С. 284–287.
13. Карагодова О.О., Кігель В.Р., Рожок В.Д. Дослідження операцій: Навч. посіб. К.: Центр учбової літератури, 2007. 256 с.
14. Наконечний С.І., Савіна С.С. Математичне програмування: Навч. посіб. К.: КНЕУ, 2003. 452 с.
15. Стецюк П.І., Ляшко В.І., Мазютинець Г.В. Двоетапна транспортна задача та її AMPL-реалізація. Наукові записки НаУКМА. Комп'ютерні науки. 2018. Т. 1. С. 14–20.
16. Біла Г.Д., Корчинський О.О., Стецюк П.І., Хом'як О.М., Шеховцов С.Б. Використання NEOS-сервера для розв'язання двох класів оптимізаційних задач. Cybernetics and Computer Technologies. 2022. 4. С. 56–81.

ІНФОРМАЦІЯ ПРО АВТОРІВ

Стецюк Петро Іванович – член-кореспондент НАН України, доктор фізико-математичних наук, старший науковий співробітник, завідувач відділу методів негладкої оптимізації Інституту кібернетики імені В.М. Глушкова НАН України, професор кафедри інформаційних управляючих систем та технологій Ужгородського національного університету.

Міца Олександр Володимирович – доктор технічних наук, професор, завідувач кафедри інформаційних управляючих систем та технологій Ужгородського національного університету.

Стовба Віктор Олександрович – доктор філософії за спеціальністю «Прикладна математика», науковий співробітник відділу методів негладкої оптимізації Інституту кібернетики імені В.М. Глушкова НАН України.