КІБЕРНЕТИКА та КОМП'ЮТЕРНІ ТЕХНОЛОГІЇ

УДК 519.85 DOI:10.34229/2707-451X.20.4.1

П.І. СТЕЦЮК, О.П. ЛИХОВИД, А.А. СУПРУН

ПРО ЛІНІЙНУ ТА КВАДРАТИЧНУ ДВОЕТАПНІ ТРАНСПОРТНІ ЗАДАЧІ

Робота присвячена математичним моделям двоетапної транспортної задачі та достатньо універсальному способу їх розв'язання за допомогою сучасного програмного забезпечення. Описано формулювання двоетапної транспортної задачі та наведено її властивості, наведено опис задачі на мові моделювання AMPL та результати розрахунків для тестового прикладу, описано квадратичну двоетапну транспортну задачу та доведено єдиність її розв'язку.

Ключові слова: *транспортна задача, задача лінійного програмування, мова моделювання AMPL, програма gurobi, задача квадратичного програмування.*

Вступ. При формулюванні класичної двоетапної транспортної задачі [1, 2] мається на увазі, що вантаж перевозиться від постачальників до споживачів тільки через проміжні пункти. Схема функціонування перевезень вантажу в двоетапній транспортній задачі показана на рис. 1. Як проміжні пункти можуть виступати посередницькі фірми та різного роду сховища (склади).

У роботі розглядаються дві математичні моделі для двоетапної транспортної задачі (задача лінійного програмування та задача квадратичного програмування) та достатньо універсальний спосіб їх розв'язання за допомогою сучасного програмного забезпечення [3]. Він використовує опис задачі на мові моделювання AMPL (**A** **M**athematical **P**rogramming **L**anguage) [4] та залежить від того, яка із відомих програм вибирається для розв’язання задачі лінійного, або квадратичного програмування.

Матеріал викладено в такому порядку. В розділі 1 описано формулювання двоетапної транспортної задачі та наведено її властивості. В розділі 2 наведено опис задачі на мові моделювання AMPL та результати розрахунків для тестового прикладу. В розділі 3 описано квадратичну двоетапну транспортну задачу та доведено єдиність її розв'язку.

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 D_k до споживача B_j позначимо y_{kj}.

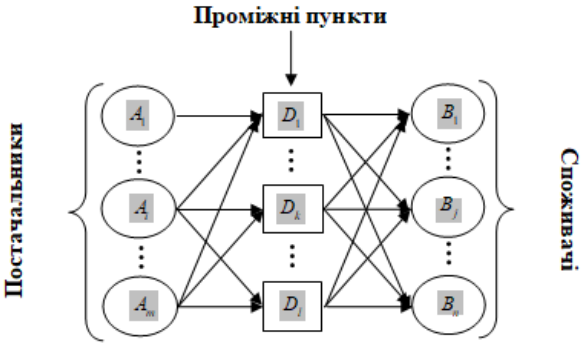

РИС. 1. Система «постачальники – проміжні пункти – споживачі» у двоетапній транспортній задачі

Двоетапна транспортна задача має такий вигляд: знайти

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

за обмежень

$$\sum_{k=1}^{l} x_{ik} = a_i, \quad i = 1,\ldots,m, \tag{2}$$

$$\sum_{k=1}^{l} y_{kj} = b_j, \quad j = 1,\ldots,n, \tag{3}$$

$$\sum_{i=1}^{m} x_{ik} - \sum_{j=1}^{n} y_{kj} = 0, \quad k = 1,\ldots,l, \tag{4}$$

$$x_{ik} \geq 0,\ y_{kj} \geq 0, \quad i = 1,\ldots,m,\ k = 1,\ldots,l,\ j = 1,\ldots,n. \tag{5}$$

Задача (1) – (5) – задача 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 (5) – умов на невід'ємність усіх змінних. Цільова функція (1) задає сумарні витрати на транспортування продукції від постачальників до споживачів через проміжні пункти. Обмеження (2) означають транспортування усієї продукції $a_1, ..., a_m$ із пунктів постачання до проміжних пунктів, а обмеження (3) – що споживачам потрібно доставити необхідну продукцію $b_1, ..., b_n$ з проміжних пунктів. Обмеження (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 (2) – (5) – системи лінійних рівностей та лінійних нерівностей. Звідси випливає наступне твердження.

Лема 1 [3]. Система обмежень (2) – (5) – несумісна, якщо $\sum_{i=1}^{m} a_i \neq \sum_{j=1}^{n} b_j$.

Лема 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 (2) – (5) – сумісна і можна переходити до розв'язання задачі (1) – (5).

2. AMPL-реалізація задачі та тестовий приклад. AMPL-код для опису двоетапної транспортної задачі (1) – (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 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 (2) – (5), яка визначається лемою 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 [5] 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 [2]. 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 табл. 1 і 2.

ТАБЛИЦЯ 1

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

ТАБЛИЦЯ 2

$D \setminus B$	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 (1) – (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 (1)-(5)
#Роздрукувати цільову функцію 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 [6] на NEOS-сервері має наступний вигляд:

```
NEOS Server Version 6.0
   Solver   : lp:Gurobi:AMPL
   Start    : 2020-12-19 07:51:02
   End      : 2020-12-19 07:51:06
   Host     : prod-sub-1.neos-server.org

************************************************************
You are using the solver gurobi_ampl.

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Checking ampl.mod for gurobi_options...
Executing AMPL.
processing data.
processing commands.
Executing on prod-exec-3.neos-server.org

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

Gurobi 9.0.1: threads=4
Gurobi 9.0.1: optimal solution; objective 22100
1 simplex iterations
f_opt = 22100
_solve_time = 0.004989

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
;
```

Для тестового прикладу задача (1) – (5) має багато розв'язків. Один із них показано на рис. 2. Для нього значення цільової функції дорівнює 22100 умовних одиниць, склад D_1 завантажений на 2200 одиниць, а склад D_2 – на 1500 одиниць.

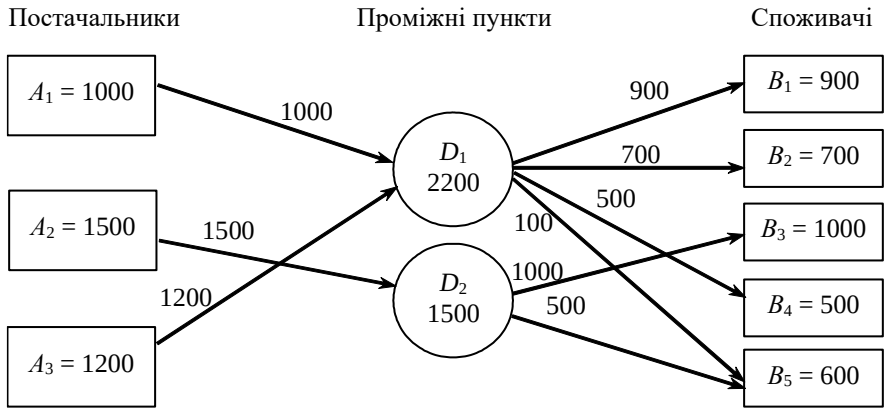

РИС. 2. Перший оптимальний план транспортування продукції

Другий розв'язок показано на рис. 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.

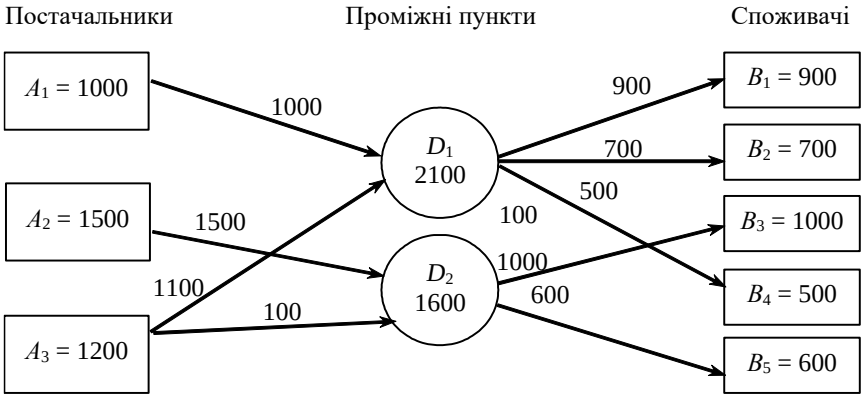

РИС. 3. Другий оптимальний план транспортування продукції

Неоднозначності оптимального розв'язку легко уникнути, якщо від задачі лінійного програмування (1) – (5) перейти до задачі квадратичного програмування, де цільова функція буде строго опуклою квадратичною функцією. Таку задачу розглянемо далі.

3. Квадратична двоетапна транспортна задача. Її будемо розглядати у такому вигляді: знайти

$$F^* = \min_{x,y}\left\{F(x,y) = \sum_{i=1}^{m}\sum_{k=1}^{l}(\varepsilon_{ik}x_{ik}^2 + c_{ik}x_{ik}) + \sum_{k=1}^{l}\sum_{j=1}^{n}(\varepsilon_{kj}y_{kj}^2 + c_{kj}y_{kj})\right\} \quad (6)$$

за обмежень

$$\sum_{k=1}^{l} x_{ik} = a_i, \quad i = 1,\ldots,m, \tag{7}$$

$$\sum_{k=1}^{l} y_{kj} = b_j, \quad j = 1,\ldots,n, \tag{8}$$

$$\sum_{i=1}^{m} x_{ik} - \sum_{j=1}^{n} y_{kj} = 0, \quad k = 1,\ldots,l, \tag{9}$$

$$x_{ik} \geq 0,\ y_{kj} \geq 0, \quad i = 1,\ldots,m,\ k = 1,\ldots,l,\ j = 1,\ldots,n. \tag{10}$$

Якщо $\varepsilon_{ik} > 0$ та $\varepsilon_{jk} > 0$, то цільова функція (6) – це опукла сепарабельна квадратична функція. Відповідна їй задача (6) – (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 (6) – (10) переходить у задачу лінійного програмування (1) – (5). Тут лінійні обмеження (7) означають постачання усієї продукції $a_1, ..., a_m$ до проміжних пунктів, а лінійні обмеження (8) – доставку необхідної продукції $b_1, ..., b_n$ з проміжних пунктів. Лінійні обмеження (9) задають умови на те, щоб вся продукція яка приходить від постачальників до кожного проміжного пункту, була обов'язково відправлена споживачам. Лінійні обмеження (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 (7) – (10) справедлива лема 1, яка дає можливість перевірити коректність вхідних даних $a_1, ..., a_m$ та $b_1, ..., b_n$. Якщо виконується умова $\sum_{i=1}^{m} a_i = \sum_{j=1}^{n} b_j$, то система (7) – (10) є сумісною і можна переходити до розв'язання задачі (6) – (10).

Лема 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 (6) – (10) має єдиний розв'язок.

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 (6) – (10). Їм відповідає оптимальне значення цільової функції $f^* = f(x^*, y^*) = f(x^{**}, y^{**})$. Обидва розв'язки x^*, y^* та x^{**}, y^{**} задовольняють обмеженням (7) – (10), тобто

$$\sum_{k=1}^{l} x_{ik}^* = a_i, \quad \sum_{k=1}^{l} x_{ik}^{**} = a_i, \quad i = 1,\ldots,m, \tag{11}$$

$$\sum_{k=1}^{l} y_{kj}^* = b_j, \quad \sum_{k=1}^{l} y_{kj}^{**} = b_j, \quad j = 1,\ldots,n, \tag{12}$$

$$\sum_{i=1}^{m} x_{ik}^* - \sum_{j=1}^{n} y_{kj}^* = 0, \quad \sum_{i=1}^{m} x_{ik}^{**} - \sum_{j=1}^{n} y_{kj}^{**} = 0, \quad k = 1,\ldots,l, \tag{13}$$

$$x_{ik}^* \geq 0,\ y_{kj}^* \geq 0,\ x_{ik}^{**} \geq 0,\ y_{kj}^{**} \geq 0, \quad i = 1,\ldots,m,\ k = 1,\ldots,l,\ j = 1,\ldots,n. \tag{14}$$

Точки $x^{***} = \lambda x^{*} + (1-\lambda)x^{**}$ та $y^{***} = \lambda y^{*} + (1-\lambda)y^{**}$, де $0 < \lambda < 1$, задовольняють системі обмежень (7) – (10). Дійсно, враховуючи рівності (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 рівностям (7). Враховуючи рівності (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 (8). Аналогічно, використовуючи рівності (13) та нерівності (14), легко показати, що точки x^{***} та y^{***} задовольняють рівностям (9) та нерівностям (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

$$F(x,y) = \sum_{i=1}^{m} \sum_{k=1}^{l} (\varepsilon_{ik} x_{ik}^2 + c_{ik} x_{ik}) + \sum_{k=1}^{l} \sum_{j=1}^{n} (\varepsilon_{kj} y_{kj}^2 + c_{kj} y_{kj}) \tag{15}$$

є також строго опуклою. Тому, якщо $0 < \lambda < 1$, то для $f(x^{***}, y^{***})$ – значення цільової функції у 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 (6) – (10), так як в точці (x^{***}, y^{***}), яка задовольняє обмеженням (7) – (10), значення цільової функції $F(x^{***}, y^{***})$ є меншим за мінімальне значення F^{*}. Лема 2 доведена.

Частковим випадком задачі (6) – (10) є задача лінійного програмування (1) – (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 (1) – (5) точка мінімуму не завжди є єдиною (див. тестовий приклад, розділ 2). Якщо вибирати досить малими значення величин ε_{ik} та ε_{jk}, то це дозволяє наблизити розв'язок задачі (6) – (10) до одного з розв'язків задачі (1) – (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 (1) – (5), яка визначається вибором величин ε_{ik} та ε_{jk} у задачі квадратичного програмування (6) – (10).

Висновки. У роботі описано властивості двох варіантів двоетапної транспортної задачі: задачі лінійного програмування (1) – (5) та задачі квадратичного програмування (6) – (10). Наведено AMPL-код для розв'язання двоетапної транспортної задачі лінійного програмування за допомогою сучасного програмного забезпечення для задач лінійного програмування. Наведено та проаналізовано результати розрахунку за допомогою програми **gurobi** для задачі (1) – (5), яка має багато розв'язків. Сформульовано двоетапну транспортну задачу квадратичного програмування та досліджено умови, при яких вона має єдиний розв'язок.

Розроблений AMPL-код для двоетапної транспортної задачі лінійного програмування та його модифікації можуть бути використані для розв'язання різноманітних транспортних задач при вивченні сучасних дисциплін для підготовки фахівців у галузі логістики та логістичних послуг [7]. Відмітимо, що розроблений AMPL-код легко адаптувати для врахування нижніх та верхніх меж на кількість продукції, що перевозиться. Для цього достатньо обмеження (5), або (10) замінити на таке обмеження

$$x_{ik}^{low} \le x_{ik} \le x_{ik}^{up},\ y_{kj}^{low} \le y_{kj} \le y_{kj}^{up}, \quad i=1,\dots,m,\ k=1,\dots,l,\ j=1,\dots,n, \tag{16}$$

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.

Список літератури

1. Карагодова О.О., Кігель В.Р., Рожок В.Д. Дослідження операцій: Навч. посіб. К.: Центр учбової літератури, 2007. 256 с.
2. Наконечний С.І., Савіна С.С. Математичне програмування: Навч. посіб. К.: КНЕУ, 2003. 452 с.
3. 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. https://doi.org/10.18523/2617-3808.2018.14-20
4. Fourer R., Gay D., Kernighan B. AMPL, A Modeling Language for Mathematical Programming. Belmont: Duxburry Press, 2003. 517 p.
5. NEOS Solver. https://neos-server.org/neos/solvers/
6. Gurobi Optimization, Inc., Gurobi Optimizer Reference Manual, 2014. https://www.gurobi.com
7. Григорак М.Ю. Інтелектуалізація ринку логістичних послуг: концепція, методологія, компетентність: монографія. Київ: Сік Груп Україна, 2017. 513 с.

Одержано 21.10.2020

Стецюк Петро Іванович,
доктор фізико-математичних наук, завідувач відділу
Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
stetsyukp@gmail.com

Лиховид Олексій Петрович,
науковий співробітник Інституту кібернетики імені В.М. Глушкова НАН України, Київ,
o.lykhovyd@gmail.com

Супрун Антон Андрійович,
аспірант Інституту кібернетики імені В.М. Глушкова НАН України, Київ.
anton_s2007@ukr.net

UDC 519.85

P. Stetsyuk, O. Lykhovyd, A. Suprun

On Linear and Quadratic Two-Stage Transportation Problem

V.M. Glushkov Institute of Cybernetics of the NAS of Ukraine, Kyiv
Correspondence: stetsyukp@gmail.com

Introduction. When formulating the classical two-stage transportation problem, it is assumed that the product is transported from suppliers to consumers through intermediate points. Intermediary firms and various kinds of storage facilities (warehouses) can act as intermediate points.

The article discusses two mathematical models for two-stage transportation problem (linear programming problem and quadratic programming problem) and a fairly universal way to solve them using modern software. It uses the description of the problem in the modeling language AMPL (A Mathematical Programming Language) and depends on which of the known programs is chosen to solve the problem of linear or quadratic programming.

The purpose of the article is to propose the use of AMPL code for solving a linear programming two-stage transportation problem using modern software for linear programming problems, to formulate a mathematical model of a quadratic programming two-stage transportation problem and to investigate its properties.

Results. The properties of two variants of a two-stage transportation problem are described: a linear programming problem and a quadratic programming problem. An AMPL code for solving a linear programming two-stage transportation problem using modern software for linear programming problems is given. The results of the calculation using Gurobi program for a linear programming two-stage transportation problem, which has many solutions, are presented and analyzed. A quadratic programming two-stage transportation problem was formulated and conditions were found under which it has unique solution.

Conclusions. The developed AMPL-code for a linear programming two-stage transportation problem and its modification for a quadratic programming two-stage transportation problem can be used to solve various logistics transportation problems using modern software for solving mathematical programming problems. The developed AMPL code can be easily adapted to take into account the lower and upper bounds for the quantity of products transported from suppliers to intermediate points and from intermediate points to consumers.

Keywords: transportation problem, linear programming problem, AMPL modeling language, Gurobi program, quadratic programming problem.