УДК 004.4:519.87
DOI 10.47049/2226-1893-2023-4-207-220

РІШЕННЯ ДВОЕТАПНОЇ ТРАНСПОРТНОЇ ЗАДАЧІ З ВИКОРИСТАННЯМ БІБЛІОТЕКИ PULP МОВИ PYTHON

М.В. Розум
к.ф.-м.н., доцент кафедри «Технічна кібернетика й інформаційні технології ім. проф. Р.В. Меркта»
ORCID:0000-0002-9459-8044
І.Г. Бугаєва
к.т.н, доцент кафедри «Технічна кібернетика й інформаційні технології ім. проф. Р.В. Меркта»
ORCID:0000-0002-2839-9266

Одеський національний морський університет, Одеса, Україна

***Анотація**. У статті розглянуто побудову математичної моделі двоетапної транспортної задачі. Наведений приклад розв'язання двохетапної транспортної задачі з додатковими умовами: можливі безпосередні поставки продукції від виробників до споживачів, перевезення продукції між проміжними базами є недопустимим, в незбалансованій задачі продукція деяких окремих виробників повинна бути вивезена повністю. Наведено рішення такої задачі з використанням вирішувача CBC бібліотеки Python. Показана ефективність отриманого рішення порівняно зі стандартним рішенням транспортних задач методом потенціалів з побудовою початкового опорного плану, наприклад, методом мінімальної вартості.*

***Ключові слова:** транспортна задача, лінійне програмування, математична модель, зовнішній вирішувач, комп'ютерна модель.*

UDC 004.4:519.87
DOI 10.47049/2226-1893-2023-4-207-220

SOLUTION OF THE TWO-STAGE TRANSPORTATION PROBLEM WITH USING THE PULP LIBRARY OF THE PYTHON LANGUAGE

M.V. Rozum
Ph.D., Docent of the Department «Technical Cybernetics and Information Technologies named after prof. R.V. Merkt»
ORCID:0000-0002-9459-8044
I.H. Buhaieva
Ph.D., Docent of the Department «Technical Cybernetics and Information Technologies named after prof. R.V. Merkt»
ORCID:0000-0002-2839-9266

Odessa National Maritime University

***Abstract**. The article considers the construction of a mathematical model of a two-stage transport problem. The given example of solving a two-stage transport problem with additional conditions: direct deliveries of products from manufacturers to consumers are possible, transportation of products between intermediate bases is inadmissible, in an unbalanced problem, the products of some individual manufacturers must be exported in full. The solution of such a problem is presented using the CBC solver of the Python library. The effectiveness of the obtained solution compared to the standard solution of transport problems by the method of potentials with the construction of an initial reference plan, for example, by the method of minimum cost, is shown.*

***Keywords**: transportation problem, linear programming, mathematical model, external solver, computer model.*

Вступ. Вагомою складовою ефективності економічної діяльності є раціональне перевезення сировини, матеріалів, готової продукції тощо. Транспортна задача відноситься до спеціального класу задач лінійного програмування, до яких зводиться аналіз практичних моделей управління і планування. Призначення класичної транспортної задачі – визначення обсягів перевезення з пунктів відправлення в пункти призначення з мінімальною сумарною вартістю перевезення. Двохетапна транспортна задача або задача з проміжними пунктами відповідає ситуації, коли перевезення продукції виконується не безпосередньо від постачальника до споживача, а через деякі проміжні пункти. В якості проміжних пунктів можуть виступати прикордонні митні термінали, морські порти тощо.

Розв'язання задачі можна розбити на п'ять основних етапів: отримання опису проблеми, формулювання економіко-математичної моделі, рішення маремематичної задачі, виконання постоптимального аналізу, подання рішення та аналіз. Але розглянутий процес не є лінійним. Наприклад, після формулювання і рішення задачі часто розглядається обґрунтованість отриманого рішення при консультації з особою, яка надала опис проблеми. Або якщо отримали рішення, яке не задовольняє замовника, ймовірно треба змінити чи оновити формулювання для того, щоб включити до нього нове розуміння реальної проблеми. Цей процес можна зобразити у вигляді схеми методології дослідження операцій (рис. 1).

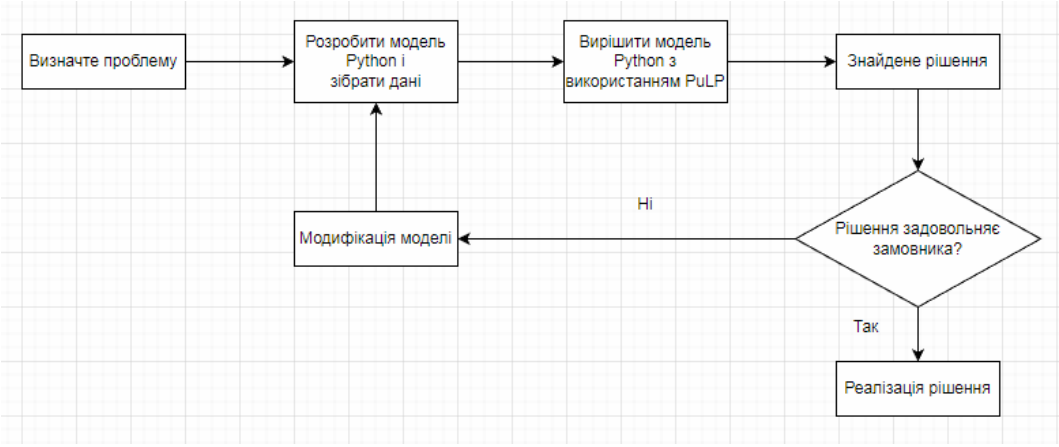

Рис. 1. Методологія дослідження операцій

Процес моделювання починається з чіткого опису моделі, потім за допомогою математичних методів формується математична модель і розробляється програма. Розробник моделі вирішує її з використанням комп'ютерних програм. Отримане рішення перетворюється в рішення з точки зору опису моделі.

Метою статті є дослідження ефективності використання мови Python та її бібліотеки PuLP в процесі моделювання двохетапних транспортних задач.

Аналіз досліджень і публікацій. В роботах [1; 2; 3; 4; 5] розглянута постановка класичної транспортної задачі, знаходження опорних планів, описаний метод потенціалів знаходження розв'язків транспортної задачі, розглянуті альтернативний оптимум та вироджуваність, обмеження на пропускну здатність. В роботі [6] наведено приклад рішення транспортної задачі з обмеженнями перевезення вантажу від постачальника до споживача з отриманням альтернативного рішення. В статті [7] розроблений програмний додаток для рішення транспортної задачі методом потенціалів. Програмне забезпечення реалізоване за допомогою веб-технологій HTML, CSS, Javascript і надає можливість в інтерактивному режимі вивчити всі етапи рішення задачі. В роботі [8] розглядається алгоритм переходу від двохетапної задачі з проміжними пунктами до одноетапної транспортної задачі, для визначення опорного розв'язку застосували метод північно-західного кута, для знаходження рішення – метод потенціалів. В статті [9] наведено демонстраційний приклад з результатами розрахунку двоетапної транспортної задачі з використанням AMPL за допомогою програми gurobi для випадку $\sum_{i=1}^{m} x_{ik} = \sum_{j=1}^{n} x_{kj}$, $k = \overline{1,\ p}$, $i = \overline{1,\ m}$, $j = \overline{1,\ n}$. В роботі [10] на прикладі освоєння вантажних перевезень при оптимізації режиму взаємодії двох видів транспорту показано рішення двохетапної транспортної задачі засобами MS Excel за умовою заборони перевезення безносередньо від постачальників до споживачів. В статті [11] розроблено математичні моделі, методи та алгоритми розв'язання неперервних багатоетапних задач розміщення підприємств з неперервно-розподіленим ресурсом.

Виклад основного матеріалу. Нехай в m пунктах постачання $A_1, A_2, \ldots, A_m$ є відповідно $a_1, a_2, \ldots, a_m$ одиниць продукції, яку необхідно перевезти до проміжних пунктів $D_1, D_2, \ldots, D_p$, місткості сховищ (складських приміщень) яких становлять $d_1, d_2, \ldots, d_p$, а потім доставити її споживачам $B_1, B_2, \ldots, B_n$, потреби яких становлять $b_1, b_2, \ldots, b_n$. Відомі також витрати на перевезення одиниці продукції від кожного постачальника до проміжних пунктів — c'_{jk} ($i = 1, 2, \ldots, m$, $k = 1, 2, \ldots, p$) та від посередників до споживачів — c''_{kj} ($k = 1, 2, \ldots, p$, $j = 1, 2, \ldots, n$). Потрібно визначити оптимальну схему перевезень продукції з мінімальними сумарними витратами.

Математична модель.

Якщо обсяг продукції, що перевозиться від i-го постачальника до k-ої фірми, позначити через x'_{ik} ($i = 1, 2, \ldots, m$, $k = 1, 2, \ldots, p$), а обсяг продукції, що перевозиться від k-ої фірми j-му споживачеві – через x''_{kj} ($k = 1, 2, \ldots, p$, $j = 1, 2, \ldots, n$), то математична модель задачі матиме вигляд

$$Z = \sum_{i=1}^{m}\sum_{k=1}^{p} c'_{ik}x'_{ik} + \sum_{k=1}^{p}\sum_{j=1}^{n} c''_{kj}x''_{kj} \to \min$$

$$\sum_{k=1}^{p} x'_{ik} = a_i, \qquad i = \overline{1, m},$$

$$\sum_{k=1}^{p} x''_{kj} = b_j, \qquad j = \overline{1, n},$$

$$\sum_{i=1}^{m} x'_{ik} = \sum_{j=1}^{n} x''_{kj} = d_k, \; k = \overline{1, p},$$

$$x'_{ik} \geq 0, \qquad x''_{kj} \geq 0, \; i = \overline{1, m}, \; j = \overline{1, n}, \; k = \overline{1, p}.$$

Мова програмування Python має бібліотеки для вирішення задач лінійного програмування, особливим випадком яких є транспортні задачі. Вирішимо цю модель за допомогою бібліотеки PuLP. PuLP [12] − безкоштовне програмне забезпечення з відкритим кодом для моделювання задач лінійного програмування, написане на мові програмування Python. Воно використовується для опису задач оптимізації у вигляді математичних моделей. Потім PuLP може викликати будь-який з численних зовнішніх вирішувачів (CBC, GLPK, CPLEX, Gurobi тощо) для вирішення цієї моделі, а потім використовувати команди Python для управління і відображення рішення. Для установки бібліотеки PuLP в ОС Windows застосовується система управління python-пакетами pip:

pip install pulp.

Для створення нових змінних застосовується клас LpVariable(), наприклад x = LpVariable("x", 0, 100), y = LpVariable("y", 0, 50), для створення нової задачі лінійного програмування – клас LpProblem(): prob = LpProblem("myTask", LpMinimize). Створені обмеження додаються в задачу, наприклад prob += x + y <= 1000. Цільова функція також додається в задачу: prob += 25 * x + 10 * y.

Для вирішення задачі використовуємо вирішувач CBC, який встановлено по замовчанню status = prob.solve().

Приклад проведення вирішення двоетапної транспортної задачі.

Опис проблеми. На ринку є 4 виробники A_1, A_2, A_3, A_4 деякої однорідної продукції. За звітний період підприємства A_1, A_2, A_3, A_4 виготовляють продукцію в обсягах відповідно 75, 15, 40, 15 одиниць. Ця продукція відправляється на дві проміжні бази D_1 і D_2, місткість складських приміщень яких складає 75 од. кожної, а потім − до двох споживачів B_1, B_2, попит кожного з яких становить відповідно 45 і 55 одиниць. Вартості перевезень одиниці продукції (в умовних одиницях) від виробників на проміжні бази, а потім — з проміжних баз до споживачів наведені в табл. 1 і табл. 2.

Таблиця 1

Виробник	Вартість перевезення одиниці продукції від виробника на проміжну базу, ум. од.	
	D_1	D_2
A_1	3	7
A_2	5	6
A_3	4	9
A_4	8	11

Таблиця 2

Проміжна база	Вартість перевезення одиниці продукції з проміжної бази до споживачів, ум. од.	
	B_1	B_2
D_1	3	7
D_2	9	12

Крім того, за індивідуальними контрактами можливі також безпосередні поставки продукції від другого виробника до другого споживача. Вартість транспортування одиниці продукції за маршрутом A_2B_2 складає 2 ум. од. Перевезення продукції з однієї проміжної бази на іншу є недопустимим. Додаткова умова: продукцію першого і третього виробників вивезти повністю.

Визначити оптимальний план перевезення продукції, за якого загальні транспортні витрати були б найменшими.

Математична модель.

Позначимо через x_{ik}^1 ($i=1,2,3,4; k=1,2$) обсяг перевезень продукції від *i*-го виробника до *k*-ї проміжної бази, x_{kj}^2 ($k=1,2; j=1,2$) обсяг перевезень продукції від *k*-ї проміжної бази до *j*-го споживача, x'_{22} обсяг безпосереднього перевезення продукції від другого виробника до другого споживача.

Специфіка даної задачі:

1) $\sum_{i=1}^{4} a_i = 145 > 100 = \sum_{j=1}^{2} b_j$ – маємо незбалансовану задачу, тобто виробники виробляють більше продукції, ніж сумарна потреба в ній, але продукція за умовами задачі першого і третього виробників повинна бути вивезена повністю.

2) $\sum_{i=1}^{4} a_i = 145 < \sum_{k=1}^{2} d_k = 150$, тобто місткість проміжних баз повністю не використовуватиметься.

3) $\sum_{j=1}^{2} b_j = 100 < \sum_{k=1}^{2} d_k = 150$, тобто місткість проміжних баз повністю не використовуватиметься.

Запишемо економіко-математичну модель задачі.

Знайти $x_{ik}^1 \geq 0$, $x_{kj}^2 \geq 0$ $i = 1,2,3,4; k = 1,2; j = 1,2$, що належать області допустимих рішень, визначеної умовами

$$\sum_{k=1}^{2} x_{ik}^1 \leq a_i ,$$

$$\sum_{k=1}^{2} x_{kj}^2 = b_j ,$$

$$\sum_{i=1}^{4} x_{ik}^1 \leq \sum_{j=1}^{2} d_k ,$$

$$\sum_{j=1}^{2} x_{kj}^2 \leq \sum_{j=1}^{2} d_k ,$$

$$\sum_{k=1}^{2} x_{1k}^1 = 75, \sum_{k=1}^{2} x_{3k}^1 = 40,$$

$$i = 2,4; \; j = 1,2; \; k = 1,2$$

і мінімізують сумарні витрати на перевезення продукції

$$Z = \sum_{i=1}^{4} \sum_{k=1}^{2} c_{ik}^1 x_{ik}^1 + \sum_{k=1}^{2} \sum_{j=1}^{2} c_{kj}^2 x_{kj}^2 + 2x'_{22} \to \min ,$$

де $a = (75; 15; 40; 15)$, $b = (45; 55)$, $C^1 = \begin{pmatrix} 3 & 7 \\ 5 & 6 \\ 4 & 9 \\ 8 & 11 \end{pmatrix}$, $C^2 = \begin{pmatrix} 3 & 7 \\ 9 & 12 \end{pmatrix}$, $d = (75; 75)$.

Для зручності матрицю вартості розташуємо у вигляді таблиці 3. Для збалансованості задачі введемо фіктивного третього споживача B_3 з попитом 145-100=45 одиниць продукції. Вартість перевезення до цього споживача дорівнює 0.

Таблиця 3

	D_1	D_2	B_1	B_2	B_3	*Пропозиція*
A_1	3	7				75
A_2	5	6		2		15
A_3	4	9				40
A_4	8	11				15
D_1	0		3	7	0	75
D_2		0	9	12	0	75
Попит	75	75	45	55	45	

Темні клітинки означають, що перевезення між відповідними пунктами заборонене.

Побудова комп'ютерної моделі на мові Python з використанням PuLP

Спочатку імпортуємо всі класи та функції з PuLP
```
from pulp import *
```

Для оцінки швидкості роботи імпортуємо модуль time
```
import time
```

Далі визначимо вузли (виробники, проміжні бази, споживачі) та їх потужності

```
Vur_Bazu = ["A1", "A2", "A3", "A4", "D1", "D2"]
supply = {"A1": 75, "A2": 15, "A3": 40, "A4": 15, "D1": 75, "D2": 75}

Bazu_Spoj = ["D1", "D2", "B1", "B2", "B3"]
demand = {"D1": 75, "D2": 75, "B1": 45, "B2": 55, "B3": 45}
```

Далі введемо матрицю вартості. Для того, щоб показати що перевезення між відповідними пунктами заборонене, введемо в цю матрицю дуже велике число порівняно з іншими числами матриці, наприклад 1000000.

```
costs = [  # Bazu_Spoj
   # D1 D2 B1 B2 B3
   [3, 7, 100000, 1000000, 1000000],  # A1   Vur_Bazu
   [5, 6, 1000000, 2, 1000000],  # A2
   [4, 9, 1000000, 1000000, 1000000],  # A3
   [8, 11, 1000000, 1000000, 1000000],  # A4
   [0, 1000000, 3, 7, 0],  # D1
   [1000000, 0, 9, 12, 0],  # A2
]
```

Списки Vur_Bazu (вузли пропозиції) та Bazu_Spoj (вузли попиту) додаються для створення великого списку (всіх вузлів) та вводяться в функцію makeDict PuLP.

```
costs = makeDict([Vur_Bazu, Bazu_Spoj], costs, 0)
```

Другий параметр − це матриця витрат, а останній параметр встановлює стандартне значення для вартості дуги. Після створення матриці вартості під час виклику costs[“A1”][“D1”] він поверне вартість транспортування 3 від виробника A_1 до проміжної бази D_1. Якщо викликається costs[“A5”][“D2”], він поверне 0, оскільки це значення за замовчуванням.

Розрахуємо швидкість виконання програми. Для цього запустимо таймер:

```
start = time.time()
```

Змінна prob створюється за допомогою функції LpProblem, яка має два параметри: перший – назва задачі (моделі), другий - тип оптимізації (LpMinimize, LpMaximize). Для транспортних задач вказуємо LpMinimize

```
prob = LpProblem("Transport Problem", LpMinimize)
```

Далі створимо список кортежів, в якому зберігаються всі шляхи

```
Routes = [(i, j) for i in Vur_Bazu for j in Bazu_Spoj]
```

За допомогою класу LpVariable створюємо словник vars, що містить змінні нашої задачі x_{ij}. Він має п’ять параметрів: перший – назва, другий – кортеж, який складається з ключем словників Vur_Bazu, Bazu_Spoj, третій – нижня межа - нуль, четвертий − верхня межа – None (відсутня), п’ятий – тип даних (LpContinuous, LpInteger, значення за замовчуванням – LpContinuous), в нашій задачі змінні визначаються як цілі числа. Ключами словника є ім'я виробника, потім ім'я споживача (["A1"]["D1"]) і дані Route_Tuple, наприклад, [“A1”][“D1”] : Шлях_A1_D1.

```
vars = LpVariable.dicts("Шлях", (Vur_Bazu, Bazu_Spoj), 0, None, LpInteger)
```

Цільова функція додається до змінної prob за допомогою списку, другий параметр – рядок, який пояснює, що вона робить

```
prob += (
   lpSum([vars[i][j] * costs[i][j] for (i, j) in Routes]),
   "Sum_of_Transporting_Costs",
)
```

Далі введемо обмеження задачі

```
for i in Vur_Bazu:
```

```
    prob += (
        lpSum([vars[i][j] for j in Bazu_Spoj]) == supply[i],
        f"Sum_of_Products_out_Vur_Bazu{i}",
    )

for j in Bazu_Spoj:
    prob += (
        lpSum([vars[i][j] for i in Vur_Bazu]) == demand[j],
        f"Sum_of_Products_into_Bazu_Spoj{j}",
    )
```

Обмеження типу $x_{ij} \geq 0$ було додано до моделі при визначенні змінних.

Тепер, коли всі дані про задачу введені, функцію writeLP() можна використовувати для копіювання цієї інформації у файл .lp до каталогу, з якого запускається блок коду. Після успішного виконання коду можна відкрити цей .lp-файл у текстовому редакторі, щоб побачити, що зробили попередні кроки.

```
prob.writeLP("Trans_Model.lp")
```

Створена модель вирішується за допомогою вирішувача, який вибирає PuLP. Вхідні дужки після solve()в цьому випадку залишаються порожніми, проте їх можна використовувати для вказівки того, який вирішувач хоче використовувати користувач (наприклад, prob.solve(CPLEX())):

```
prob.solve()
```

Результати виклику вирішувача треба відобразити у вигляді статусу виконання. Статус може приймати значення “Not Solved”, “Infeasible”, “Unbounded”, “Undefined” or “Optimal”. Значення prob.status повертається у вигляді цілого числа, яке необхідно перетворити на його текстове значення за допомогою словника LpStatus:

```
print("Status:", LpStatus[prob.status])
```

Далі друкуємо змінні та їх оптимальні значення

```
print("---- Оптимальний план: ----")
for i in prob.variables():
    print(i.name, "=", i.varValue)
```

Та виводимо мінімальне значення цільової функції за допомогою функції value(). Це гарантує, що отримане число буде виведено в десятковому, а не в експоненційному форматі:

```
print("Транспортні витрати = ", value(prob.objective))
```

Наприкінці підрахуємо та роздрукуємо швидкість виконання програми
stop = time.time()
print("Час виконання програми:", stop - start)
Результат розв’язання задачі показано на рис. 2.
Аналіз розв’язку.

```
Status: Optimal
---- Оптимальний план: ----
Шлях_A1_B1 = 0.0
Шлях_A1_B2 = 0.0
Шлях_A1_B3 = 0.0
Шлях_A1_D1 = 35.0
Шлях_A1_D2 = 40.0
Шлях_A2_B1 = 0.0
Шлях_A2_B2 = 15.0
Шлях_A2_B3 = 0.0
Шлях_A2_D1 = 0.0
Шлях_A2_D2 = 0.0
Шлях_A3_B1 = 0.0
Шлях_A3_B2 = 0.0
Шлях_A3_B3 = 0.0
Шлях_A3_D1 = 40.0
Шлях_A3_D2 = 0.0
Шлях_A4_B1 = 0.0
Шлях_A4_B2 = 0.0
Шлях_A4_B3 = 0.0
Шлях_A4_D1 = 0.0
Шлях_A4_D2 = 15.0
Шлях_D1_B1 = 45.0
Шлях_D1_B2 = 30.0
Шлях_D1_B3 = 0.0
Шлях_D1_D1 = 0.0
Шлях_D1_D2 = 0.0
Шлях_D2_B1 = 0.0
Шлях_D2_B2 = 10.0
Шлях_D2_B3 = 45.0
Шлях_D2_D1 = 0.0
Шлях_D2_D2 = 20.0
Транспортні витрати =  1205.0
Час виконання програми: 0.05000019073486328
```

Рис. 2. Результат розв’язання двоетапної транспортної задачі

Зобразимо рішення задачі графічним способом, де вузли – це виробники, проміжні бази та споживачі, а дуги – кількість продукції, яка перевозиться між відповідними вузлами (рис. 3).

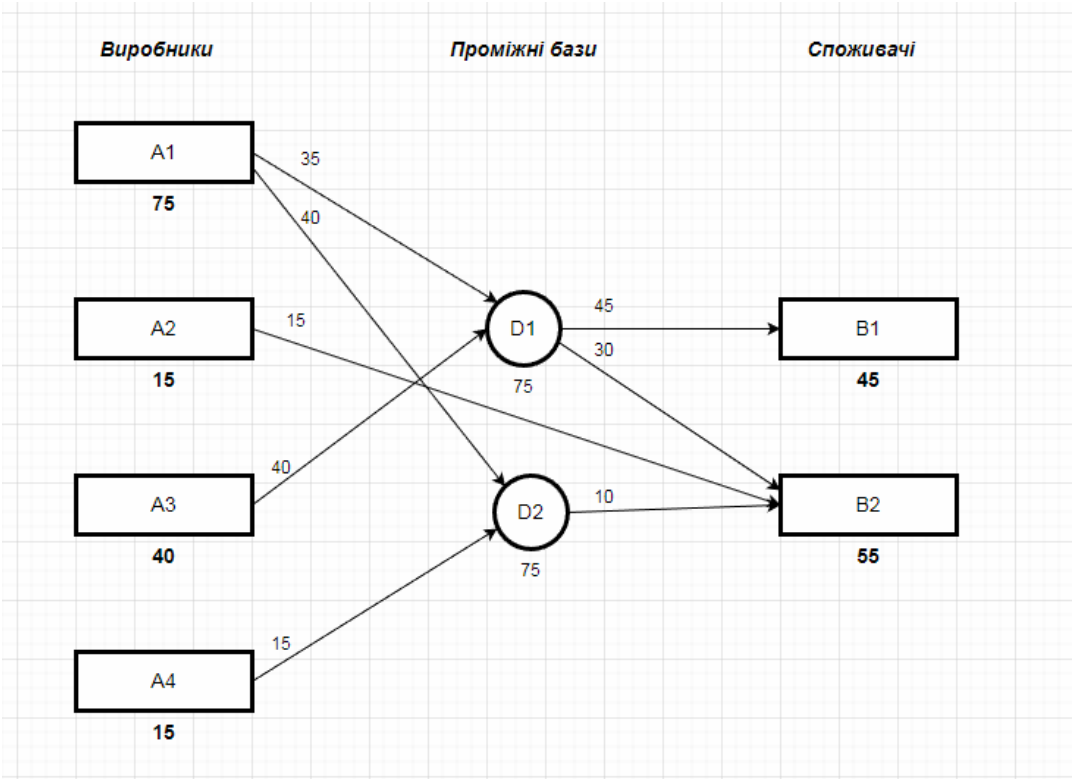

Рис. 3. Графічний запис оптимального плану перевезення

Продукція виробника A_1 (75 од.) вивезена повністю: 35 од. на проміжну базу D_1, 40 од. на проміжну базу D_2. Загальна вартість перевезення від виробника A_1 на проміжні бази D_1 і D_2 складає 385 ум. од.

Продукція виробника A_2 (15 од.) вивезена повністю: 15 од. споживачу В2, здійснений механізм прямої поставки. Вартість перевезення складає 30 ум. од.

Продукція виробника A_3 (40 од.) вивезена повністю: 40 од. на проміжну базу D_1. Вартість перевезення складає 160 ум. од.

Продукція виробника A_4 (15 од.) вивезена повністю: 15 од. на проміжну базу D_2. Вартість перевезення складає 165 ум. од.

Потужність проміжної бази D_1 використовується повністю. Продукція на базу D_1 постачається з пункту A_1 в кількості 35 од., з пункту A_3 в кількості 40 од.

Потужність проміжної бази D_2 використовується не повністю. Продукція на базу D_2 постачається з пункту A_1 в кількості 40 од., з пункту A_4 в кількості 15 од. Складські поміщення бази D_2 в обсязі 20 од. не використані.

Споживачу B_1 треба завозити продукцію з бази D_1 в кількості 45 од., вартість перевезення складає 135 ум. од., споживачу B_2 треба завозити продукцію з бази D_1 в кількості 30 од., з бази D_2 в кількості 10 од., вартість перевезення складає 330 ум. од. Мінімальні транспортні витрати при такому плані перевезення складають 1205 ум. од. З бази D_2 не вивезена продукція у розмірі 45 од. по причині незбалансованості транспортної задачі.

Схеми (рис.1, рис.3) побудовані авторами за допомогою програми draw.io.

Висновки. Побудована математична модель двохетапної транспортної задачі з додатковими умовами:

1) існує пряме перевезення від виробника до споживача;

2) задача незбалансована з надлишком виробленої продукції;

3) продукція першого і третього виробників повинна бути вивезена повністю.

Побудована комп'ютерна модель двоетапної транспортної задачі, наведено її рішення, реалізоване в середовищі PyCharm, на мові Python з використання бібліотеки PuLP. Рішення є ефективним порівняно зі стандартним рішенням транспортних задач методом потенціалів з побудовою початкового опорного плану, наприклад методом мінімальної вартості. Час виконання програми 0,05 секунд.

СПИСОК ЛІТЕРАТУРИ

1. *Дякон В.М., Ковальов Л.Є. Математичне програмування. – К.: Вид-во Європ. ун-ту, 2007. – 497 с.*
2. *Зайченко Ю.П. Дослідження операцій. – К.: ЗАТ «ВІПОЛ», 2000. – 688 с.*
3. *Бех О.В. Математичне програмування: Навч. посібник / О.В. Бех, Т.А. Городня, А.Ф. Щербак. – Львів: Магнолія-2006, 2014. – 200 с.*
4. *Дзюбан І.Ю. Методи дослідження операцій / І.Ю. Дзюбан, О.Л. Жиров, О.Г. Охріменко. – К.: ІВЦ Видавництво «Політехніка», 2005. – 108 с.*
5. *Дослідження операцій в економіці: Підручник / За ред. І.К. Федоренко, О.І. Черняка. – К.: Знання, 2007. – 558 с.*
6. *Розум М.В. Транспортна задача з обмеженнями на пропускну спроможність // М.В. Розум. Зб. тез допов. 75 проф.-виклад. науково-технічна конф. (30 вересня 2022 р.) – Одеса, ОНМУ, 2022. – С.271-274.*
7. *Бугаєва І.Г., Розум М.В. Інтерактивне навчання рішенню транспортної задачі // І.Г. Бугаєва, М.В. Розум. Матеріали 5 міжнар. конф. «Адаптивні технології управління навчанням» Одеса, 23-25 жовтня 2019. – Одеса, 2019. – С.38-41.*
8. *Карагодова О.О., Кігель В.Р., Рожок В.Д. Дослідження операцій. – К.: Центр учбової літератури, 2007. – 256 с.*
9. *П.І. Стецюк, О.П. Бисага, С.С. Трегубенко. Двоетапна транспортна задача з обмеженням на кількість проміжних пунктів // Компьютерна математика. 2018, № 2, С.119-128.*

10. *Леснікова І.Ю., Халіпова Н.В., Терещенко М.В., Харченко Є.М., Єршова Н.М. Дослідження операцій у середовищі електронних таблиць Excel. – К.: Центр учбової літератури, 2007. – 186 с.*
11. *Us S.A., Koriashkina L.S. Stanina O.D. An optimal two-stage allocation of material flows in a transport-logistic system with continuously distributed resource // S.A. Us, L.S. Koriashkina, O.D. Stanina. Радіоелектроніка, інформатика, управління. 2019. № 1, С. 256-271.*
12. *Optimization with PuLP. URL: / https://coin-or.github.io/pulp/ (дата звернення 01.09.2023).*

REFERENCES

1. *Diakon V.M., Kovalov L.Ie. Matematychne prohramuvannia. – K.: Vyd-vo Evrop. un-tu, 2007. – 497 p.*
2. *Zaichenko Yu.P. Doslidzhennia operatsii. – K.:ZAT «VIPOL», 2000.–688p.*
3. *Bekh O.V. Matematychne prohramuvannia: Navch. posibnyk / O.V. Bekh, T.A. Horodnia, A.F. Shcherbak. – Lviv: Mahnoliia-2006, 2014. – 200 p.*
4. *Dziuban I.Iu. Metody doslidzhennia operatsii / I.Iu. Dziuban, O.L. Zhyrov, O.H. Okhrimenko. – K.: IVTs Vydavnytstvo «Politekhnika», 2005. – 108 p.*
5. *Doslidzhennia operatsii v ekonomitsi: Pidruchnyk / Za red. I.K. Fedorenko, O.I. Cherniaka. – K.: Znannia, 2007. – 558 p.*
6. *Rozum M.V. Transportna zadacha z obmezhenniamy na propusknu spromozhnist // M.V. Rozum. Zb. tez dopov. 75 prof.-vyklad. naukovo-tekhnichna konf. (30 veresnia 2022 r.) – Odesa, ONMU, 2022. – P.271-274.*
7. *Buhaieva I.H., Rozum M.V. Interaktyvne navchannia rishenniu transportnoi zadachi. // I.H. Buhaieva, M.V. Rozum. Materialy 5 mizhnar. konf. «Adaptyvni tekhnolohii upravlinnia navchanniam» Odesa, 23-25 zhovtnia 2019. – Odesa, 2019. – P. 38-41.*
8. *Karahodova O.O., Kihel V.R., Rozhok V.D. Doslidzhennia operatsii. – K.: Tsentr uchbovoi literatury, 2007. – 256 p.*
9. *Stetsiuk P.I., Bysaha O.P., Trehubenko S.S. Dvoetapna transportna zadacha z obmezhenniam na kilkist promizhnykh punktiv // Kompiuterna matematyka. 2018, № 2, P.119-128.*
10. *Lesnikova I.Iu., Khalipova N.V., Tereshchenko M.V., Kharchenko Ye.M., Yershova N.M. Doslidzhennia operatsii u seredovyshchi elektronnykh tablyts Excel. − K.: Tsentr uchbovoi literatury, 2007. – 186 p.*
11. *Us S.A., Koriashkina L.S. Stanina O.D. An optimal two-stage allocation of material flows in a transport-logistic system with continuously distributed resource // S.A. Us, L.S. Koriashkina, O.D. Stanina. Radioelektronika, informatyka, upravlinnia. 2019. № 1, P. 256-271.*
12. *Optimization with PuLP. URL: / https://coin-or.github.io/pulp/ (data zvernennia 01.09.2023).*

Стаття надійшла до редакції 12.11.2023

Посилання на статтю: Розум М.В., Бугаєва І.Г.Рішення двоетапної транспортної задачі з використанням бібліотеки PULP мови PYTHON // Вісник Одеського національного морського університету: Зб. наук. праць, 2023. № 4 (71). С. 207-220. DOI 10.47049/2226-1893-2023-4-207-220.

Article received 12.11.2023

Reference a journalartic: Rozum M.V., Buhaieva I.H. Solution of the twostage transportation problem with using the pulp library of the python language // Herald of the Odesa national maritime university: Coll. scient. works, 2023. № 4 (71). P. 207-220. DOI 10.47049/ 2226-1893-2023-4-207-220.