УДК 519.85

Стецюк П. І.
https://orcid.org/0000-0003-4036-2543
Ляшко В. І.
Мазютинець Г. В.

ДВОЕТАПНА ТРАНСПОРТНА ЗАДАЧА ТА ЇЇ AMPL-РЕАЛІЗАЦІЯ

Наведено формулювання двоетапної транспортної задачі закритого типу та обґрунтовано умови, за яких система лінійних обмежень задачі є несумісною. Ці умови використано під час розробки AMPL-коду для розв'язання задачі за допомогою сучасного програмного забезпечення для задач лінійного програмування. Наведено демонстраційний приклад із результатами розрахунку за допомогою програми gurobi NEOS-сервера.

Ключові слова: двоетапна транспортна задача, задача лінійного програмування, мова моделювання AMPL, NEOS-сервер, gurobi.

Вступ

Формулювання класичної двоетапної транспортної задачі [1] передбачає, що вантаж перевозиться від постачальників до споживачів тільки через проміжні пункти. Схему функціонування перевезень вантажу наведено на рис. 1. Проміжними пунктами можуть бути посередницькі фірми та різного роду сховища (склади).

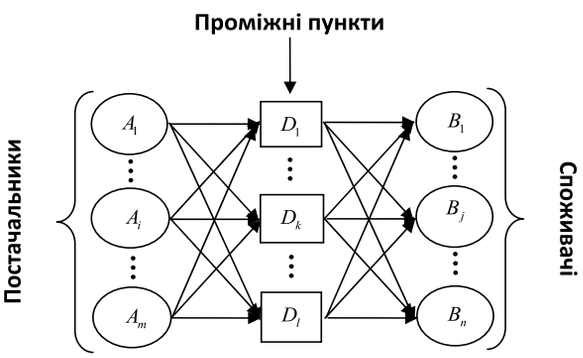

Рис. 1. Система «постачальники – проміжні пункти – споживачі» у двоетапній транспортній задачі

Для розв'язання двоетапної транспортної задачі застосовують різного роду спеціалізовані алгоритми [1; 2]. У статті розглянемо універсальний спосіб розв'язання двоетапної транспортної задачі з використанням сучасного програмного забезпечення. Цей спосіб використовує опис задачі на мові моделювання AMPL [5] та залежить тільки від того, яка з відомих програм використовуватиметься для розв'язання задачі лінійного програмування.

Матеріал статті викладено в такому порядку. У розділі 1 описано формулювання двоетапної транспортної задачі та наведено її властивості. У розділі 2 наведено опис задачі на мові моделювання AMPL [5] та результати розрахунків для тестового прикладу.

1. Двоетапна транспортна задача та її властивості

Нехай у m пунктах постачання $A_1, \ldots, A_m$ є $a_1, \ldots, a_m$ одиниць продукції, яку потрібно перевезти до n споживачів $B_1, \ldots, B_n$, задовольнивши їхні потреби $b_1, \ldots, b_n$. Для транспортування продукції від постачальників до споживачів можна задіяти l проміжних пунктів $D_1, \ldots, D_l$. 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

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

за обмежень

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

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

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

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

Задача (1)–(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 (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 $a_1, \ldots, a_m$ із пунктів постачання до проміжних пунктів, а обмеження (3) – що споживачам потрібно доставити необхідну продукцію $b_1, \ldots, b_n$ з проміжних пунктів. Обмеження (4) задають умови на те, щоб уся продукція, яка надходить від постачальників до кожного проміжного пункту, була обов'язково відправлена споживачам. Це визначає умови на сумісність системи обмежень (2)–(5) – системи лінійних рівностей та лінійних нерівностей. Звідси випливає таке твердження.

Лема 1. Система обмежень (2)–(5) є несумісною, якщо $\sum_{i=1}^{m} a_i \ne \sum_{j=1}^{n} b_j$.

Доведення. Доведення проведемо методом від супротивного. Приймемо, що існують невід'ємні $\bar{x} = \{\bar{x}_{ik}, i = 1,\ldots,m, k = 1,\ldots,l\}$ та $\bar{y} = \{\bar{y}_{kj}, k = 1,\ldots,l, j = 1,\ldots,n\}$, які задовольняють систему обмежень (2)–(4). Тобто для $\bar{x} \ge 0$ та $\bar{y} \ge 0$ справджуються такі рівності:

$$\sum_{k=1}^{l} \bar{x}_{ik} = a_i, \quad i = 1,\ldots,m, \tag{6}$$

$$\sum_{k=1}^{l} \bar{y}_{kj} = b_j, \quad j = 1,\ldots,n, \tag{7}$$

$$\sum_{i=1}^{m} \bar{x}_{ik} - \sum_{j=1}^{n} \bar{y}_{kj} = 0, \quad k = 1,\ldots,l. \tag{8}$$

Якщо всі рівності із (6) просумувати за індексом $i = 1,\ldots,m$, а всі рівності із (7) просумувати за індексом $j = 1,\ldots,n$, то отримаємо такі рівності:

$$\sum_{i=1}^{m}\sum_{k=1}^{l} \bar{x}_{ik} = \sum_{i=1}^{m} a_i, \tag{9}$$

$$\sum_{j=1}^{n}\sum_{k=1}^{l} \bar{y}_{jk} = \sum_{j=1}^{n} b_j. \tag{10}$$

Віднявши від рівності (9) рівність (10) та змінивши порядок індексів у сумах, отримаємо таку рівність:

$$\sum_{k=1}^{l}\sum_{i=1}^{m} \bar{x}_{ik} - \sum_{k=1}^{l}\sum_{j=1}^{n} \bar{y}_{jk} = \sum_{i=1}^{m} a_i - \sum_{j=1}^{n} b_j. \tag{11}$$

Якщо всі рівності із (8) просумувати за індексом $k = 1,\ldots,l$, то отримаємо таку рівність:

$$\sum_{k=1}^{l}\sum_{i=1}^{m} \bar{x}_{ik} - \sum_{k=1}^{l}\sum_{j=1}^{n} \bar{y}_{kj} = 0. \tag{12}$$

За умови $\sum_{i=1}^{m} a_i \ne \sum_{j=1}^{n} b_j$ із рівності (11) випливає $\sum_{k=1}^{l}\sum_{i=1}^{m} \bar{x}_{ik} - \sum_{k=1}^{l}\sum_{j=1}^{n} \bar{y}_{kj} \ne 0$, що суперечить рівності (12). Це протиріччя доводить, що вектори $\bar{x}$ та $\bar{y}$ не можуть бути допустимими для обмежень (2)–(4), отже, при виконанні умови $\sum_{i=1}^{m} a_i \ne \sum_{j=1}^{n} b_j$ система обмежень (2)–(5) є несумісною. **Лему доведено.**

Лема 1 дає можливість перевірити вхідні дані щодо 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) є сумісною і можна переходити до розв'язання задачі (1)–(5).

2. AMPL-реалізація задачі (1)–(5) та тестовий приклад

AMPL-код для опису двоетапної транспортної задачі (1)–(5) має такий 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 [6] із розділу «Linear Programming», для яких підтримуються вхідні формати даних на AMPL, так і за допомогою комерційних або з вільним доступом версій AMPL[1].

Тестовий приклад [3]. *Виробниче об'єднання складається з трьох філій: 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 наведено в таких 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 (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;
```

[1] Безкоштовні версії програми розміщено за посиланням: http://ampl.com/try-ampl/download-a-free-demo/

```
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 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 на NEOS-сервері має такий вигляд:

```
************************************************************

   NEOS Server Version 5.0
   Job#     : 6064547
   Password : AonehGLi
   User     : None
   Solver   : lp:Gurobi:AMPL
   Start    : 2018-05-23 00:49:53
   End      : 2018-05-23 00:49:58
   Host     : NEOS HTCondor Pool

   Disclaimer:

   This information is provided without any express or
   implied warranty. In particular, there is no warranty
   of any kind concerning the fitness of this
   information  for any particular purpose.
************************************************************
File exists
You are using the solver gurobi_ampl.
Checking ampl.mod for gurobi_options...
Executing AMPL.
processing data.
processing commands.
Executing on prod-exec-2.neos-server.org

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

Gurobi 7.5.1: 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 7.5.1: 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 7.5.1: optimal solution; objective 22100
```

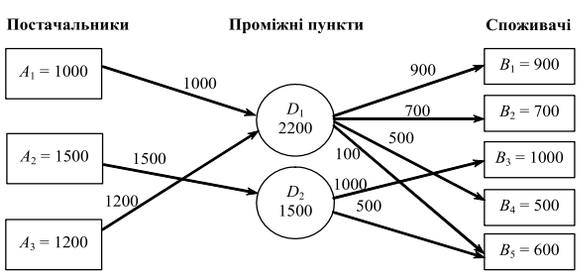

Рис. 2. Оптимальний план перевезень продукції

Розв'язок задачі (1)–(5) для тестового прикладу наведено 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.

Якщо умови леми 1 задовольняються, то система (2)–(5) є несумісною і програма gurobi не переходитиме до розв'язання задачі (1)–(5).

Наприклад, якщо в описі даних оператори

```
param a:= #Продукція в А
1 1000 2 1500 3 1200; #A_1, A_2, A_3
```

замінити на оператори

```
param a:= #Продукція в А
1 1000 2 1500 3 1100; #A_1, A_2, A_3
```

то результат розрахунку на NEOS-сервері має такий вигляд:

```
**********************************************************

   NEOS Server Version 5.0
   Job#     : 6064560
   Password : mMRdIvOn
   User     : None
   Solver   : lp:Gurobi:AMPL
   Start    : 2018-05-23 01:26:09
```

```
   End      : 2018-05-23 01:26:14
   Host     : NEOS HTCondor Pool

   Disclaimer:

   This information is provided without any express or
   implied warranty. In particular, there is no warranty
   of any kind concerning the fitness of this
   information  for any particular purpose.
**************************************************************
check at line 16 of amplin fails:
  check: sum{i in 1 .. m} a[i] == sum{j in 1 .. n} b[j];
xecuting AMPL.
processing data.
processing commands.
Executing on prod-exec-2.neos-server.org
 Error (2) in /opt/ampl/ampl -R amplin
```

Це означає, що кількість продукції у постачальників на 100 одиниць менша, ніж це потрібно споживачам. У цьому випадку система обмежень є несумісною, про що сигналізує AMPL-оператор **check**.

Висновки

У цій статті описано двоетапну транспортну задачу та наведено AMPL-код для її розв'язання за допомогою сучасного програмного забезпечення для задач лінійного програмування. Наведено демонстраційний приклад з результатами розрахунку. Розроблений AMPL-код використовують під час вивчення навчальної дисципліни «Мережеві задачі оптимізації» магістранти спеціальностей 8.122 «Комп'ютерні науки» та 8.121 «Інженерія програмного забезпечення» ДВНЗ «Ужгородський національний університет» [4]. Його планується використати під час вивчення навчальної дисципліни «Системи обробки економічної інформації» для студентів економічного факультету Національного університету «Києво-Могилянська академія».

Модифікації двоетапної транспортної задачі (1)–(5) можуть використовувати студенти вищих навчальних закладів для розроблення власних програмних проектів, пов'язаних із теорією та методами оптимізації. Наприклад, якщо обмеження (5) замінити на обмеження, які враховують нижні та верхні межі на кількість продукції, що перевозиться, то описаний метод не зміниться, а спеціалізовані алгоритми потребуватимуть значної корекції.

Список використаної літератури

1. Карагодова О. О. Дослідження операцій : навч. посіб. / О. О. Карагодова, В. Р. Кігель, В. Д. Рожок. – Київ : Центр учбової літератури, 2007. – 256 с.
2. Наконечний С. І. Математичне програмування : навч. посіб. / С. І. Наконечний, С. С. Савіна. – Київ : КНЕУ, 2003. – 452 с.
3. Стецюк П. І. AMPL-реалізація двоетапної транспортної задачі / П. І. Стецюк, Г. В. Мазютинець, Б. І. Мілешовський // Математичне та програмне забезпечення інтелектуальних систем : Тези доповідей XV Міжнародної науково-практичної конференції MPZIS-2017, 22–24 листопада 2017 р. – Дніпро : ДНУ, 2017. – С. 186–191.
4. Стецюк П. І. Мережні інформаційні технології : методичні рекомендації до вивчення курсу / П. І. Стецюк, О. В. Міца, В. І. Пецко. – Ужгород : Вид-во УжНУ «Говерла», 2014. – 65 с.
5. Fourer R. AMPL: A Modeling Language for Mathematical Programming / R. Fourer, D. Gay, B. Kernighan. – Belmont : Duxbury Press, 2003. – 517 p.
6. NEOS Solver [Electronic resource]. – Mode of access: https://neos-server.org/neos/solvers/. – Title from the screen.

Petro Stetsyuk, Volodymyr Lyashko, Gabriela Mazyutynets

TWO-STAGE TRANSPORTATION PROBLEM AND ITS AMPL-REALIZATION

The paper discusses the two-stage transportation problem of the closed type and investigates its properties. The problem is connected with finding an optimal plan for transportation of products from suppliers to consumers, using intermediate points. The formulation of the two-stage transportation problem is presented in the form of a linear programming problem, and the conditions under which the system of

linear constraints is incompatible are substantiated. These conditions are connected with checking that the total quantity of products from suppliers does not equal the total quantity of products that consumers need. It is convenient for using an AMPL check operator to determine when the two-stage transportation problem has no solution.

The AMPL code for solving the two-stage transportation problem with the help of up-to-date software for linear programming problems is offered. To describe the AMPL model, the code uses a check operator which stops the process of solving the two-stage transportation problem if the linear constraint system is incompatible. Otherwise an optimal plan for the transportation of products is printed, which can be found with the help of any program for solving the linear programming problem.

A demonstration example with the results of the calculations using the gurobi program from the NEOS server is presented. The demonstration example involves the supply of products from three suppliers to five consumers, using only two intermediate points. The AMPL code to describe the input data of the demonstration example is given, and the output files of the program gurobi for two cases are presented: (1) when the check operator allows not to interrupt the process of solving the problem; (2) when the check operator reports the incompatibility of the system of linear equations. An optimal plan for transportation of products found by the program gurobi for the demonstration example is graphically illustrated.

The developed AMPL code can be used in the study of academic disciplines for graduates of the specialities "Computer Science" and "Software Engineering" and students of economic departments. Its modifications for the two-stage transportation problem can be used by students of higher educational institutions to develop their own software projects related to the theory and methods of optimization.

Keywords: two-stage transportation problem, linear programming problem, AMPL, NEOS server, gurobi.

Матеріал надійшов 24.04.2018