Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра математики факультету інформатики

КУРСОВА РОБОТА

на тему:

ОПТИМІЗАЦІЯ ВИПЛАТ ПРАВА НА ЗАБРУДНЕННЯ

Керівник курсової роботи

к. ф.-м. н. Чорней Руслан Костянтинович

_____

(підпис)

Виконав студент 3-го року навчання

спеціальності 113 “Прикладна математика”

Бутенко Антон В’ячеславович

“____” _____ 2020 р.

Київ 2020

Тема: Оптимізація виплат права на забруднення.

Календарний план:

Номер	Назва етапу курсової	Термін виконання етапу	Примітка
1.	Отримання теми курсової роботи.	10.10.2019	
2.	Ознайомлення з темою курсової	13.10.2019	
3.	Розробка плану та структури роботи.	30.10.2019	
4.	Робота з науковою літературою, опис основних означень дилеми мандрівника.	15.11.2019	
5.	Проведення експериментальних ігор.	10.03.2020	
6.	Підсумок та обробка даних.	01.04.2020	
7.	Перевірка гіпотез.	10.04.2020	
8.	Робота над текстовим оформленням результатів.	20.04.2020	
9.	Попередній аналіз курсової. Виправлення помилок.	10.05.2020	
10.	Захист курсової роботи.	11.05.2020	

ЗМІСТ

1. ВСТУП

Пропонується та досліджується задача мінімізації витрат економіки на екологічну складову при умові, що кожний вид продукції виробляється, а кожний вид забруднювачів знищується декількома способами. Вважається, що за випуск кожного виду забруднювача береться плата (право на забруднення).

Внаслідок антропогенних впливів в сучасних умовах суспільство потребує розробки та дослідження математичних моделей взаємодії економіки та навколишнього середовища (повітряного та водного басейнів, рослинного та тваринного світів, тощо). Ці дослідження потрібні не тільки для створення матеріальних благ, а й для зменшення забруднення навколишнього середо-вища та відновлення природних ресурсів. Еколого-економічний підхід як на-уковий напрям має метою узгодження екологічних та економічних критеріїв розвитку системи „природа-виробництво”.

Особливо це актуально для України, в якій рівень „забруднення” в 10–15 разів вищий, ніж у Канаді та в 5–10 разів, ніж в ЄС.

2. ПОСТАНОВКА ЗАДАЧІ

2.1. МІЖГАЛУЗЕВА МОДЕЛЬ ЛЕОНТЬЄВА-ФОРДА

Першою міжгалузевою моделлю, що описує взаємозв'язок економіки та навколишнього середовища, є модель Леонтьєва — Форда, що складається із двох груп галузей: основне виробництво (галузі матеріального виробництва) і допоміжне виробництво (галузі, що знищують шкідливі викиди).

Математично ця модель виражається системою рівнянь:

$$\begin{cases} x^1 = A_{11}x^1 + A_{12}x^2 + y^1, \\ x^2 = A_{21}x^1 + A_{22}x^2 - y^2, \end{cases}$$

де $x^{1T} = \left(x_j^1\right)_{j\in J}$, $x^{2T} = \left(x_i^2\right)_{i\in I}$ — вектори розв'язків задачі (відповідно обсяги виробництва продукції та знищених забруднювачів), $J = \{1,2,\text{K},n\}$, $I = \{1,2,\text{K},m\}$ — множини індексів змінних, T — знак транспонування; $A_{11} = \left\| a_{ji}^{11} \right\|_{i\in J}^{j\in J}$ — квадратна матриця прямих затрат продукції на одиницю вироблення продукції , $A_{12} = \left\| a_{ji}^{12} \right\|_{i\in I}^{j\in J}$ — прямокутна матриця затрат продукції на одиницю знищення забруднювачів , $A_{21} = \left\| a_{ij}^{21} \right\|_{j\in J}^{i\in I}$ — прямокутна матриця випуску забруднювачів на одиницю вироблення продукції , $A_{22} = \left\| a_{ji}^{22} \right\|_{i\in I}^{j\in I}$ — квадратна матриця випуску забруднювачів на одиницю знищення забруднювачів), всі $a_{ij} \in [0;1]$, $A_{21}y^1 \geq y^2$ (достатня умова існування невід'ємних розв'язків), $y^{1T} = \left(y_j^1\right)_{j\in J}$, $y^{2T} = \left(y_i^2\right)_{i\in I}$ — вектори правих частин (відповідно обсяги кінцевого споживання та гранична норма незнищування забруднювачів).

2.2. ПОБУДОВА ОПТИМІЗАЦІЙНОЇ МОДЕЛІ НА ОСНОВІ МОДЕЛІ ЛЄОНТЬЄВА-ФОРДА

Основна проблема, яку покликані вирішувати економіко-математичні методи

та моделі, полягає в розв'язку задачі планування виробництва, стимулюванні пріоритетних галузей і способів виробництва. Тому серед інших постає проблема розвитку моделі Леонтьєва — Форда в напряму побудови на її основі оптимізаційних моделей. При цьому виникають оптимізаційні міжгалузеві моделі, в які обов'язково входять основні умови балансових моделей. Більше того, балансові моделі можуть інтерпретуватися та досліджуватись як частинний випадок оптимізаційних моделей.

У цій роботі розглядається оптимізаційна еколого-економічна модель із виробничими способами та цільовою функцією, що виражає витрати економіки на виробництво та знищення забруднювачів. Нехай кожний вид продукції $i \in I$ виробляється декількома способами $\varphi_i \in \mathbf{P}_i$, але кожним способом випускається лише один продукт. Аналогічно, кожний вид забруднювачів $j \in J$ знищується декількома способами $\psi_i \in \mathbf{Q}_i$, але кожним способом знищується лише один забруднювач. Уведемо такі позначення: $x^1_{i\varphi_i}$ — обсяг виробництва продукції i способом φ_i; $x^2_{j\psi_j}$ — обсяг знищення забруднювача j способом ψ_j; $a^{11}_{i\,j\varphi_j}$ — коефіцієнт прямих затрат продукції i на виробництво одиниці продукції j способом φ_j; $a^{12}_{i\,j\psi_j}$ — коефіцієнт прямих затрат продукції i на знищення одиниці забруднювача j способом ψ_j; $a^{21}_{i\,j\varphi_j}$ — коефіцієнт викиду забруднювача i при виробництві одиниці продукції j способом φ_j; $a^{22}_{i\,j\psi_j}$ — коефіцієнт викиду забруднювача i при знищенні одиниці забруднювача j способом ψ_j; y^1_i, y^2_j — відповідно обсяги кінцевого споживання продукції i та гранична норма незнищування забруднювача j. Уважається, що за кожну одиницю забруднювача, що не знищується, береться плата c_j (так зване право на забруднення), а вартість знищення способом ψ_j одиниці забруднювача j дорівнює $c_{j\psi_j}$. Маємо:

$$\sum_{k\in J} c_k \left(\sum_{j\in I} \sum_{\varphi_j \in \mathbf{P}_j} a^{21}_{k\,j\,\varphi_j} x^1_{j\,\varphi_j} + \sum_{g\in J} \sum_{\psi_g \in \mathbf{Q}_g} \sigma_{k\,g\,\psi_g} x^2_{k\,\psi_g} \right) \to \min$$

(перший доданок — обсяг викинутих забруднювачів при виробництві продукції, другий — обсяг викинутих забруднювачів при їх знищенні мінус обсяг знищених забруднювачів плюс витрати на їх знищення, усе у вартісному вигляді), де

$$\sigma_{k\,g\,\psi_g} = \begin{cases} a^{22}_{k\,g\,\psi_g}, & g \neq k, \\ a^{22}_{k\,k\,\psi_k} - 1 + c_{k\,\psi_k} / c_k, & g = k; \end{cases}$$

при таких обмеженнях

$$\sum_{j\in I} \sum_{\varphi_j \in \mathbf{P}_j} \left(\delta_{i\,j} - a^{11}_{i\,j\,\varphi_j} \right) x^1_{j\,\varphi_j} - \sum_{g\in J} \sum_{\psi_g \in \mathbf{Q}_g} a^{12}_{i\,g\,\psi_g} x^2_{g\,\psi_g} \geq y^1_i, \quad i \in I;$$

$$x^1_{j\,\varphi_j} \geq 0,\ j \in I,\ \varphi_j \in \mathbf{P}_j;$$

(обсяг виробництва кінцевої продукції не повинен бути меншим обсягу кінцевого споживання)

$$-\sum_{j\in I} \sum_{\varphi_j \in \mathbf{P}_j} a^{21}_{k\,j\,\varphi_j} x^1_{j\,\varphi_j} + \sum_{g\in J} \sum_{\psi_g \in \mathbf{Q}_g} \left(\delta_{k\,g} - a^{22}_{i\,g\,\psi_g} \right) x^2_{g\,\psi_g} \geq -y^2_k, \quad k \in J;$$

$$x^2_{g\,\psi_g} \geq 0,\ g \in J,\ \psi_g \in \mathbf{Q}_g.$$

(обсяг незнищених забруднювачів не повинен перевищувати граничної норми).

3. РЕАЛІЗАЦІЯ ЗАДАЧІ

3.1. ДАНІ ДЛЯ ДЕМОНСТРАЦІЇ РОБОТИ АЛГОРИТМУ

Нехай маємо 3 продукти та 2 забруднювачі. Продукт номер 1 виробяється трьома способами. Продукт номер 2 виробяється двома способами. Продукт номер 3 виробяється двома способами. Забруднювач номер 1 знищується двома способами. Забруднювач номер 2 знищується трьома способами.

Зімітуємо відповідні дані для тестування програми:

```
A11=np.array([
  [[0.2, 0.1, 0.1], [0.1, 0.1], [0.1, 0.4]],
  [[0.1, 0.3, 0.1], [0.2, 0.1], [0.1, 0.1]],
  [[0.2, 0.1, 0.1], [0.1, 0.3], [0.1, 0.1]],
])

A12=np.array([
    [[0.1, 0.1], [0.1, 0.2, 0.1]],
    [[0.2, 0.1], [0.1, 0.2, 0.1]],
    [[0.1, 0.1], [0.1, 0.1, 0.2]]
])

A21=np.array([
 [[0.1, 0.2, 0.1], [0.1, 0.1], [0.3, 0.1]],
 [[0.2, 0.1, 0.1], [0.2, 0.1], [0.1, 0.1]]
])

A22=np.array([
    [[0.1, 0.1], [0.2, 0.1, 0.2]],
    [[0.3, 0.1], [0.1, 0.2, 0.1]]
])

y1=np.array([
   [1000.],
   [2056.],
   [3037.]
])

y2=np.array([
   [50.],
   [49.]
])
```

```
c=np.array([5,7])

c_psi=np.array([[1,2],
                [3,4,5]
])
```

3.2. ПРОГРАМНИЙ КОД АЛГОРИТМУ

```
def resultsIsNotNegative(A21, y1, y2):
    z = []
    for i in range(len(A21)):
        z.append(np.array(np.meshgrid(*A21[i])).T.reshape((-1,3)))

    z = np.apply_along_axis(lambda x: x.reshape((len(A21), -1)), 1, np.hstack(z))

    return np.all((z @ y1 >= y2).flatten())

if(resultsIsNotNegative(A21,y1,y2)):

  def objectiveFunctionCoefs (c, c_psi, A21, A22):
    z=[]

    for j in range (0,len(A21[0])):
      sum=np.array(A21[0][j])*c[0]
      for k in range(1,len(A21)):
        sum=sum+np.array(A21[k][j])*c[k]
      z=z+list(sum)

    def sigma(A22, c, c_psi, g, k, x):
      if g != k:
          return A22[k][g][x]
      else:
          return A22[k][k] - 1 + c_psi[k][x] / c[k]

    sigma = A22.copy()

    for k in range(len(sigma)):
        for g in range(len(sigma[k])):
            for psi_g in range(len(sigma[k][g])):
                if g == k:
```

```
                sigma[k][g][psi_g] = A22[k][g][psi_g] - 1 + c_psi[k][psi_g] / c[k]
            else:
                sigma[k][g][psi_g] = A22[k][g][psi_g]

  for g in range (0,len(sigma[0])):
    sum=np.array(sigma[0][g])*c[0]
    for k in range(1,len(sigma)):
      sum=sum+np.array(sigma[k][g])*c[k]
    z=z+list(sum)

  return z

objectiveFunctionCoefs=objectiveFunctionCoefs (c, c_psi, A21, A22)

def b_ub(y1,y2,A11,A22):
  z=[]

  for i in range(len(y1)):
    z.append(-1*y1[i][0])

  for k in range(len(y2)):
    z.append(y2[k][0])

  for j in range(len(A11[0])):
    for fi_g in range(len(A11[0][j])):
      z.append(0.0)

  for g in range(len(A22[0])):
    for psi_g in range(len(A22[0][g])):
      z.append(0.0)

  return z

b_ub=b_ub(y1,y2, A11, A22)

def A_ub(A11,A12,A21,A22,y1,y2,b_ub,objectiveFunctionCoefs):

  def kronecker(m,n):
    if(m==n):
      return 1
    else:
      return 0

  Z=np.zeros((len(b_ub), len(objectiveFunctionCoefs)))
########################################
```

```
 for N in range(len(y1)):

  z=[]

  for j in range (0,len(A11[0])):
   sum=np.array(A11[0][j])*-1

   for n in range(len(sum)):
    sum[n]=kronecker(0,j)+sum[n]

   for i in range(1,len(A11)):
    sum=sum+np.array(A11[i][j])*-1

    for n in range(len(sum)):
     sum[n]=kronecker(i,j)+sum[n]

   z=z+list(sum)

  for g in range (0,len(A12[0])):
   sum=np.array(A12[0][g])

   for i in range(1,len(A12)):
    sum=sum+np.array(A12[i][g])

   z=z+list(sum)

  Z[N]=z
################################################
 for N in range(len(y1),len(y1)+len(y2)):
  z=[]

  for g in range (0,len(A21[0])):
   sum=np.array(A21[0][g])

   for i in range(1,len(A21)):
    sum=sum+np.array(A21[i][g])

   z=z+list(sum)

  for j in range (0,len(A22[0])):
   sum=np.array(A22[0][j])*-1

   for n in range(len(sum)):
    sum[n]=kronecker(0,j)+sum[n]
```

```
        for i in range(1,len(A22)):
          sum=sum+np.array(A22[i][j])*-1

          for n in range(len(sum)):
            sum[n]=kronecker(i,j)+sum[n]

        z=z+list(sum)

      Z[N]=z
  ################################################
    start_point=(len(y1)+len(y2))

    for N in range(start_point,len(Z)):

      Z[N][N-start_point] = -1
  ################################################
    return(Z)

  A_ub = A_ub(A11,A12,A21,A22,y1,y2,b_ub,objectiveFunctionCoefs)


  print('-----------------------------------------------')
  print('Коефіцієнти цільової функції')
  print(objectiveFunctionCoefs)
  print('-----------------------------------------------')
  print('Коефіцієнти нерівностей')
  print(A_ub)
  print('-----------------------------------------------')
  print('Обмеження')
  print(b_ub)
  print('-----------------------------------------------')
  print('Результат роботи алгоритму')
  print(linprog(objectiveFunctionCoefs, A_ub=A_ub, b_ub=b_ub, A_eq=None, b_eq
=None))

else:
  print("Дані не пройшли перевірку на існування невід'ємних розв'язків")
```

3.3. РЕЗУЛЬТАТ РОБОТИ АЛГОРИТМУ

Для розв’язання задачі лінійного програмування я обрав бібліотеку scipy.optimize та в ній функцію linprog.[1]

Вивід програми зображено на мал(1)

```
--------------------------------------------------
Результат роботи алгоритму
     con: array([], dtype=float64)
     fun: 12.72029462627774
 message: 'The algorithm terminated successfully and determined that the problem is infeasible.'
     nit: 5
   slack: array([-1.00729877e+03, -2.06329877e+03, -3.04429877e+03,  4.23077537e+01,
        4.13077537e+01,  1.81118925e+00,  1.81133648e+00,  6.64827852e-01,
        1.58842198e+00,  1.54250967e+00,  1.48344086e+00,  1.99023806e+00,
        8.24342729e-01,  7.04606635e-01,  6.87191546e-01,  7.98558376e-01,
        7.91909309e-01])
  status: 2
 success: False
       x: array([1.81118925, 1.81133648, 0.66482785, 1.58842198, 1.54250967,
       1.48344086, 1.99023806, 0.82434273, 0.70460664, 0.68719155,
       0.79855838, 0.79190931])
```

мал(1)

4. ВИСНОВОК

СПИСОК ЛІТЕРАТУРИ

1. SciPy.org [електронний ресурс]
   https://docs.scipy.org/doc/scipy-0.15.1/reference/generated/scipy.optimize.linprog.html