**Міністерство освіти і науки України**

­

**Національний Технічний Університет України**

**“Київський Політехнічний Інститут ім. Ігоря Сікорського”**

**Навчально-науковий комплекс «Інститут прикладного системного аналізу»**

Курсова робота

з дисципліни

**"Теорія прийняття рішень"**

Тема: *«* *ПОБУДОВА ПЛАНА ВИРОБНИЦТВА ВИРОБУ»*

Керівник Виконала

Зайченко Юрій Петрович Житанська Дар’я студентка 4 курса

“\_\_\_ ”\_\_\_\_\_\_\_\_\_\_\_2019р. групи КА-66

Захищено з оцінкою

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Київ-2019

РЕФЕРАТ

Мета роботи: дослідження задачі оптимізації побудови плана виробництва виробу, розподілу випуску виробів по підприємствах та мінімізацїї сумарних витрат.

Побудувати відповідну модель завдання НМП. Знайти різення чіткої задачі, а також нечіткої задачі зі ступенем, зокрема для стратегій оптиміста і песиміста, і порівняти отримані рішення.

  Написання програмного продукту, що дозволяє вирішувати такі завдання при різних вихідних даних. При вирішенні задачі враховувалися обмеження на кількість наявних підприємств задіяних у випуску виробів, а також обмеження на комплекти.

# Зміст

|  |  |  |
| --- | --- | --- |
|  | Вступ | 4 |
| 1 | Постановка задачі | 5 |
| 2 | Аналітичне рішення | 7 |
| 2.1  2.2 | Чітка постановка задачі  Знаходження недомінованих альтернатив зі ступенем **.** | 7  9 |
| 2.3 | Вихідні дані | 10 |
| 3 | Алгоритм розв'язку задачі | 11 |
| 3.1 | Вибір метода для розв'язку задачі лінійного програмування | 11 |
| 3.2 | Метод ПАВ(послідовного аналізу та відсіву варіантів) | 11 |
| 4 | Опис програми | 14 |
| 4.1 | Функції,які реалізують методи | 14 |
| 4.2  4.3 | Програма, для розв'язку задачі  Інструкція користувача | 14  15 |
| 5  5.1  5.2 | Аналіз отриманих результатів Аналіз роботи програми  Аналіз результатів | 15  15  20 |
|  | Висновки | 20 |
|  | **Список літератури**  **Лістинг програми** | 21  22 |

**Вступ**

##### На сьогоднішній день у виробничій сфері часто виникають завдання прийняття рішення. Кожна організація намагається максимізувати випуск комплектної продукції, а також мінімізувати витрати для її видобування, транспортування та виробництва. І одним з рішень є оптимально розподілити випуск виробів по підприємствах. В іншому випадку керівництво змушене буде витрачати ресурси і кошти на використання високовитратною і малоефективною технологією. У цій роботі проводиться дослідження задач визначення оптимального виробництва виробів, при якому мінімізуються усі витрати, з якими може зіткнутися виробник на кожному кроці обробки продукції.

**1. Постановка задачі**

*ПОБУДОВА ПЛАНА ВИРОБНИЦТВА ВИРОБУ*

В деякий заданий проміжок часу район повинен бути забезпечений виробом в кількості 5700 шт. Цей продукт випускають два підприємства і , що працюють за різними технологіями. Вироби реалізують через чотири бази , , , , потреби вироби в яких визначають такими кількостями: 1000, 1200, 1500, 2000 відповідно.

Два основних види сировини, що йдуть на виготовлення видобувають в пунктах , , , (перший вид) і , , (другий вид).

Передбачається, що для будь-якого плану виробництва вироби наявних потужностей видобутку сировини цілком достатньо, і що на частину нездійсненної потужності не проводиться жодних витрат, а на реалізовані потужності доводяться витрати, прямо пропорційні реалізованої частини:



У табл.19,20 наведені сумарні витрати на виробництво і транспортування першого і другого видів сировини, віднесені до одиниці відповідного виду сировини;

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Предприятие | Пункт добычи сырья | | | |
|  |  |  |  |  |
|  | 30 | 40 | 70 | 60 |
|  | 80 | 50 | 30 | 40 |

Таблиця 19

|  |  |  |  |
| --- | --- | --- | --- |
| Предприятие | Пункт добычи сырья | | |
|  |  |  |  |
|  | 40 | 80 | 80 |
|  | 60 | 70 | 40 |

Таблиця 20

у табл.21 - витрати на транспортування готового виробу до баз.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Предприятие | Базы | | | |
|  |  |  |  |  |
|  | 4 | 4 | 10 | 8 |
|  | 8 | 3 | 6 | 5 |

Таблиця 21

Витрати на виробництво вироби на кожному підприємстві визначають з наступних співвідношень:



Тут (k=1,2) - відповідні обсяги виробництва вироби на кожному підприємстві.

Введемо позначення: - кількість сировини першого виду, доставлене від i-го джерела j-му підприємству; - кількість сировини другого виду, що доставляється від джерела до підприємства. Технологію виготовлення виробу на k-ому підприємстві визначають наступним співвідношенням:



при наступних значеннях: 

Припустимо, що питомі витрати на видобуток і транспортування сировини першого і другого видів є випадковими, розподіленими рівномірно в інтервалі , а витрати на транспортування готових виробів від k- ого підприємства до j- ї бази - нормально розподіленими випадковими величинами , де дані по параметрах ,  наводяться в табл. 22 - 24 відповідно.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Предприятия | Пункты добычи сырья | | | |
|  |  |  |  |  |
|  | [30;50] | [40;50] | [60;80] | [40;80] |
|  | [60;100] | [30;70] | [25;35] | [35;45] |

Таблиця 22

|  |  |  |  |
| --- | --- | --- | --- |
| Предприятия | Пункты добычи сырья | | |
|  |  |  |  |
|  | [30;50] | [70;90] | [60;100] |
|  | [50;70] | [60;80] | [20;60] |

Таблиця 23

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Предприятия | Базы j | | | |
|  |  |  |  |  |
|  | 40/40 | 25/45 | 30/20 | 80/40 |
|  | 80/40 | 30/15 | 60/30 | 50/50 |

Таблиця 24

Визначити план видобутку і перевозки сировини, обсяг виробництва, вироби на кожному підприємстві, план перевезень виробів до баз, щоб сумарні витрати на видобуток і транспортування сировини, виробництво і транспортування готових виробів були мінімальними.

**2. Аналітичне рішення**

Складемо математичну модель для даного завдання. Дане завдання є завданням дискретного програмування.

**2.1 Чітка постановка задачі**

Отже, введемо такі позначення:

- кількість сировини першого виду, доставлене від i-го джерела k-ому підприємству;

- кількість сировини другого виду, доставлене від j-ого джерела k-ому підприємства.

- витрати на доставку першого виду сировини від i - го джерела k - му підприємству;

- витрати на доставку другого виду сировини від j - го джерела k - му підприємству;

- обсяги виробництва продукту T на кожному підприємстві.

- витрати на виробництво продукту на кожному підприємстві

, - обмеження на кількість видобування сировини

А = [], i = 1,4; B = [] , j = 1,3

- кількість вироби доставляється від k - го підприємства до i – ої бази;

– витрати на транспортування готового продукту з k-ого підприємства до i-ої бази;

k = 1,2;

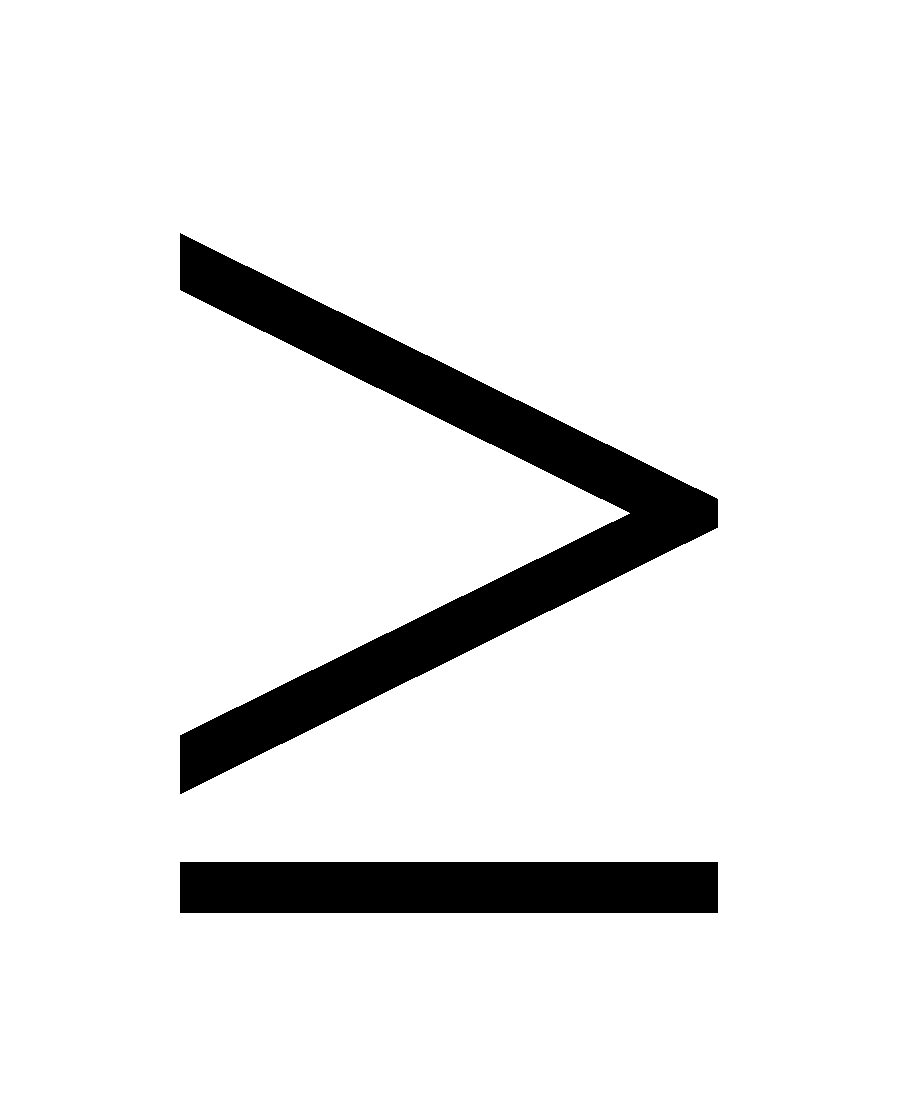
Тоді побудуємо цільову функції даної задачі. Вона буде мати наступний вигляд:

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

Після певних перетворень та спрощення виразу для заданих данних задача буде мати такий вигляд:

при таких обмеженнях :

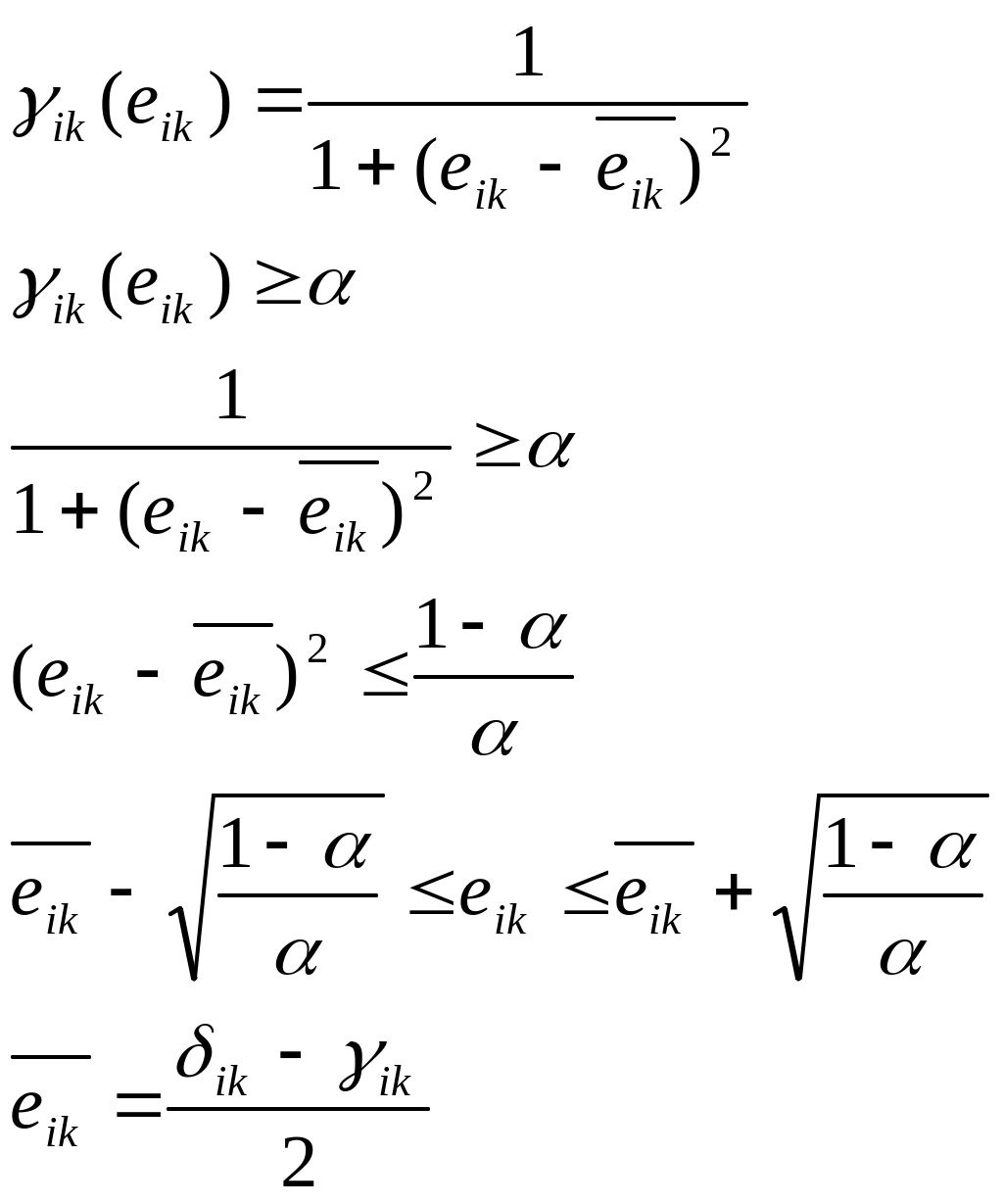
|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |

, , ,   0,

**2.2 Знаходження недомінуючих альтернатив зі ступенем** **.**

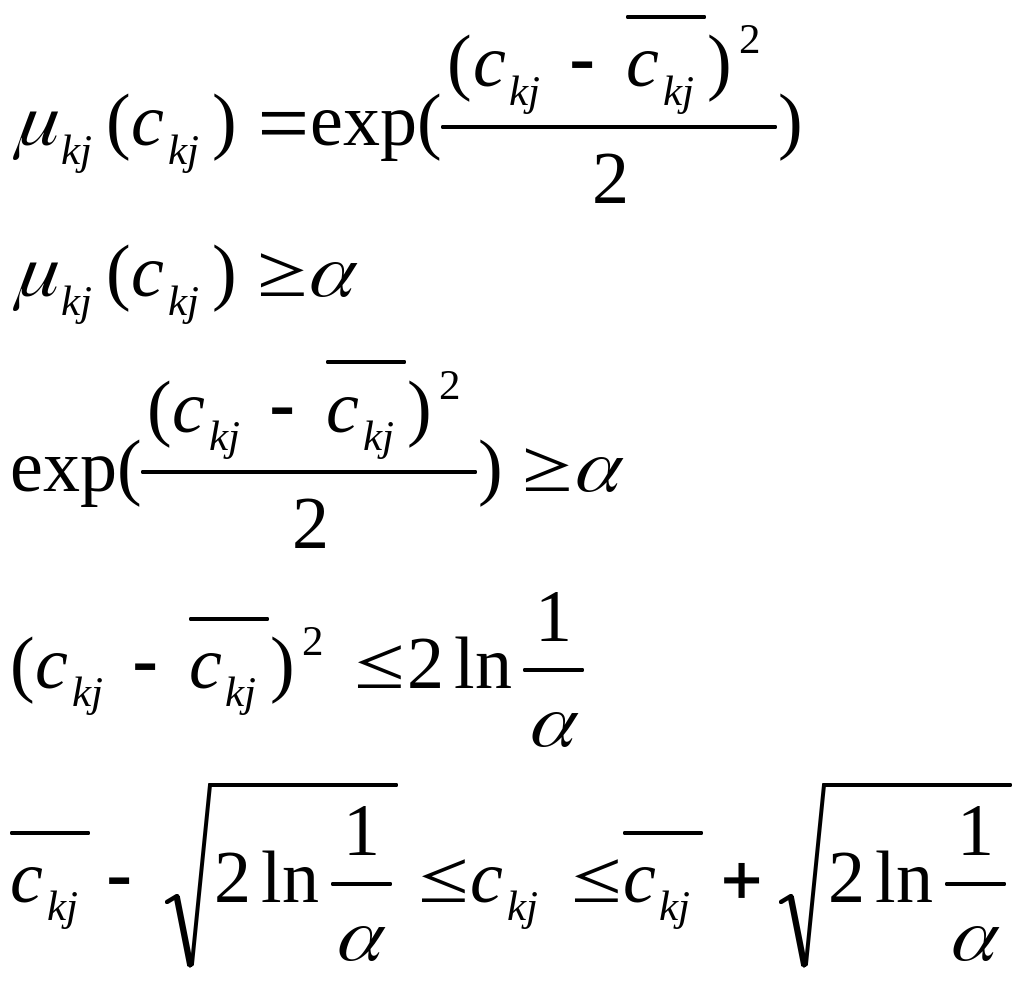
*Виведемо задачі оптиміста та песиміста, взявши рівень 0,8 .*

Величини та розподілені рівномірно у заданому проміжку . Введемо функцію належності:



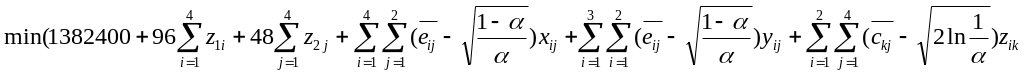
Величина розподілена нормально з заданими параметрами у таблиці 24.

Введемо функцію належності:

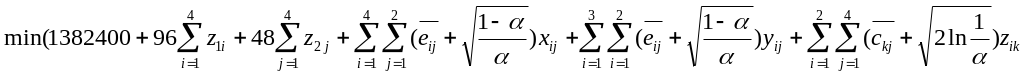


Далі будемо розв’язувати задачі оптиміста та песиміста.

**Задача оптиміста:**

****

**Задача песиміста:**

****

Отже, завдання зводиться до задачі параметричного програмування і може бути вирішена відомими методами.

**2.3 Вихідні дані**

Визначимо дані, які необхідно отримати на виході:

Вироби на кожному підприємстві та обсяг продукції на кожному з підприємств.

# 3. Алгоритм розв'язку задачі

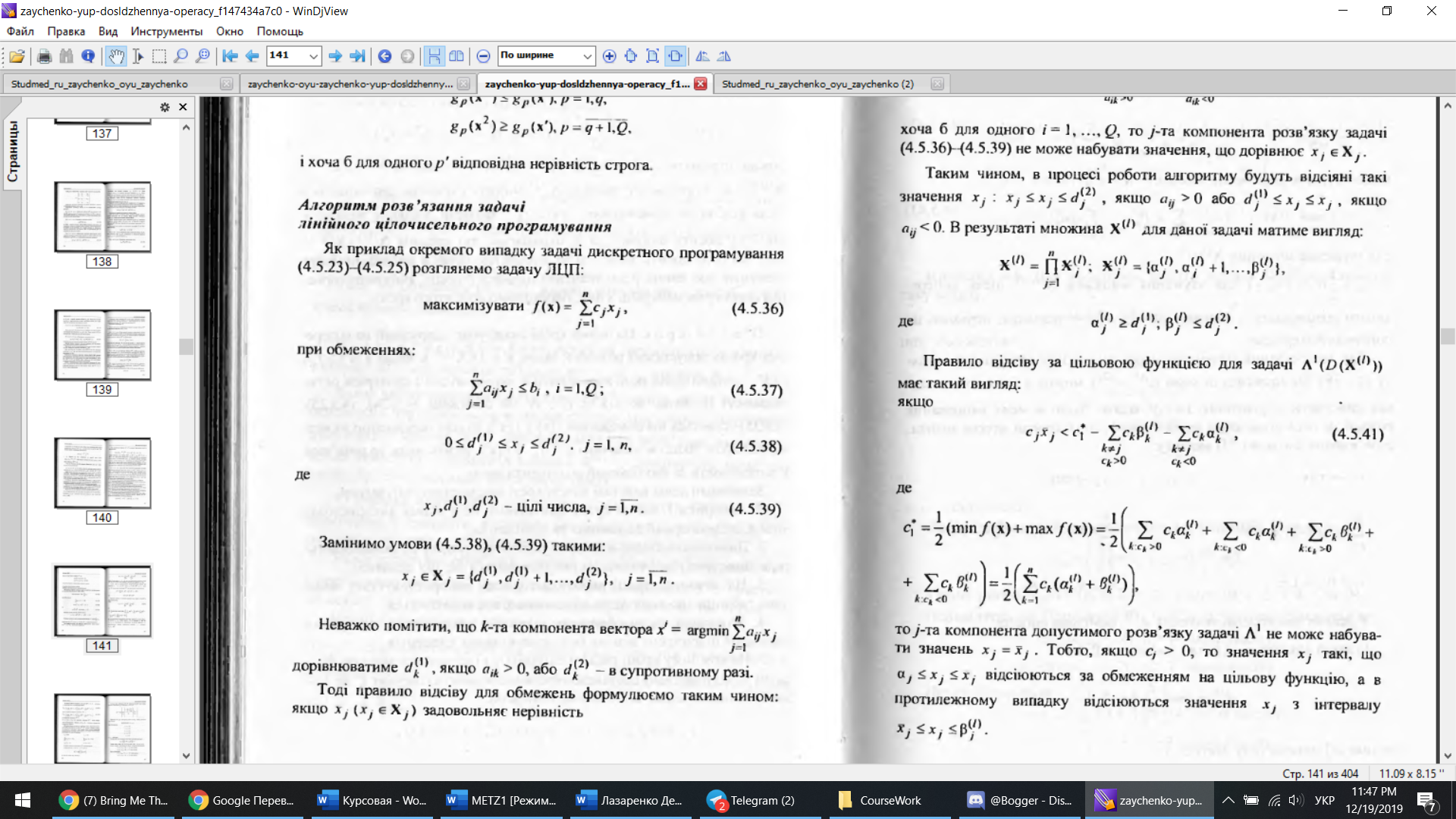
### 3.1 Вибір методу для розв'язку задачі лінійного програмування

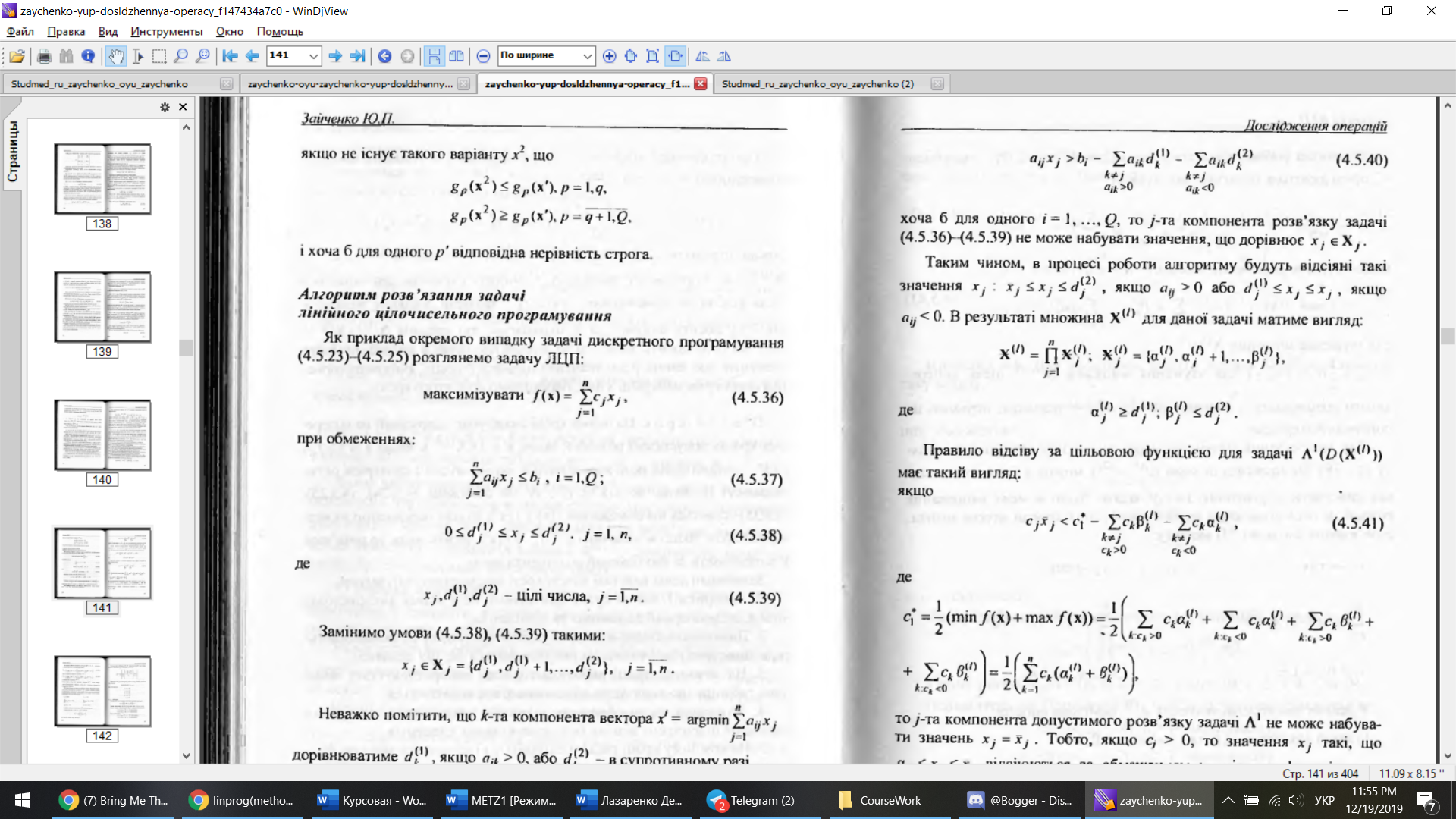
Дана модель містить лінійні обмеження-нерівності і має цілі невідомі. Отже завдання можна віднести до дискретного програмування.

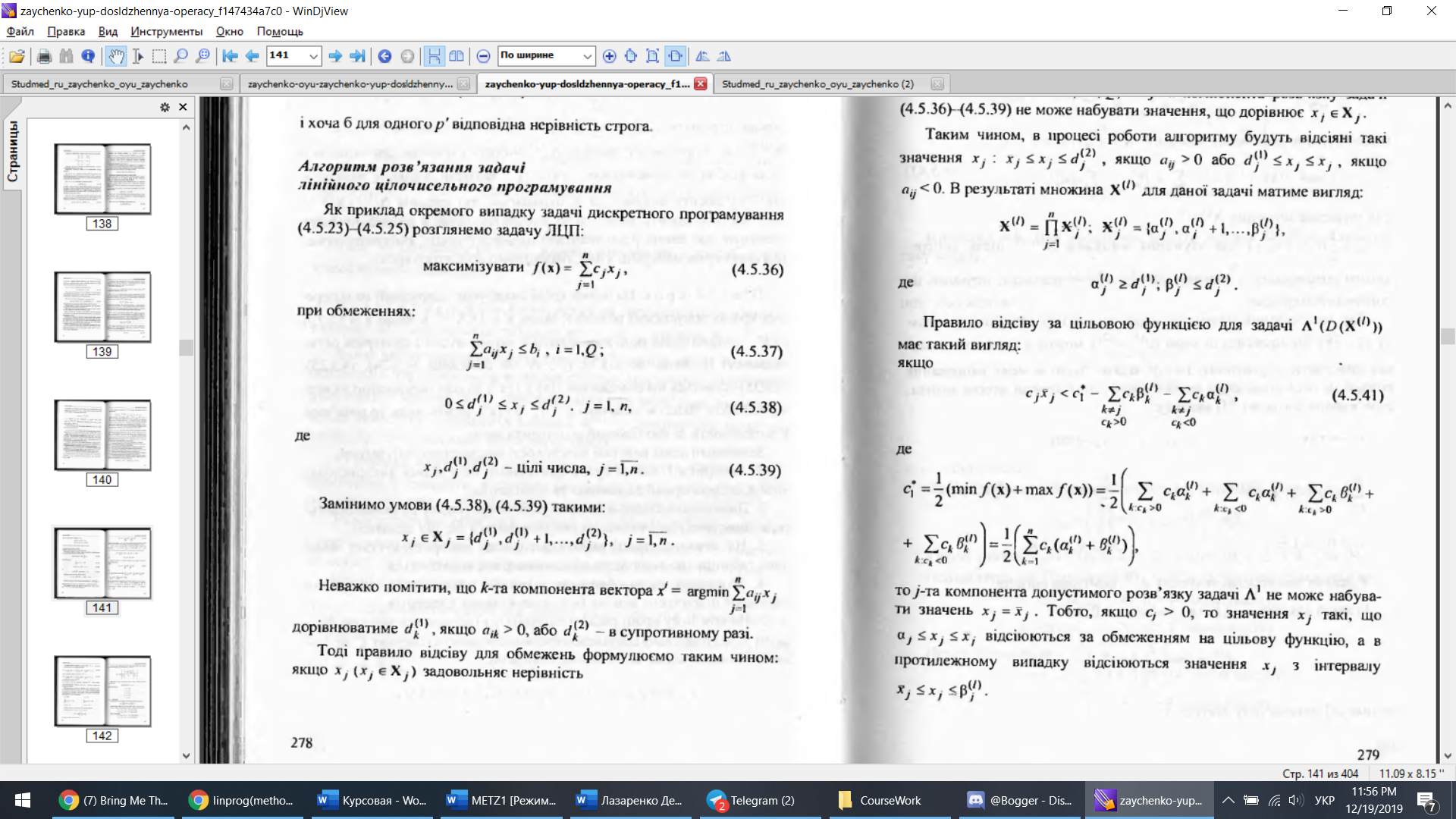
Для вирішення такого завдання можна скористатися симплекс-методом або методом послідовного аналізу та відсіву варіантів.

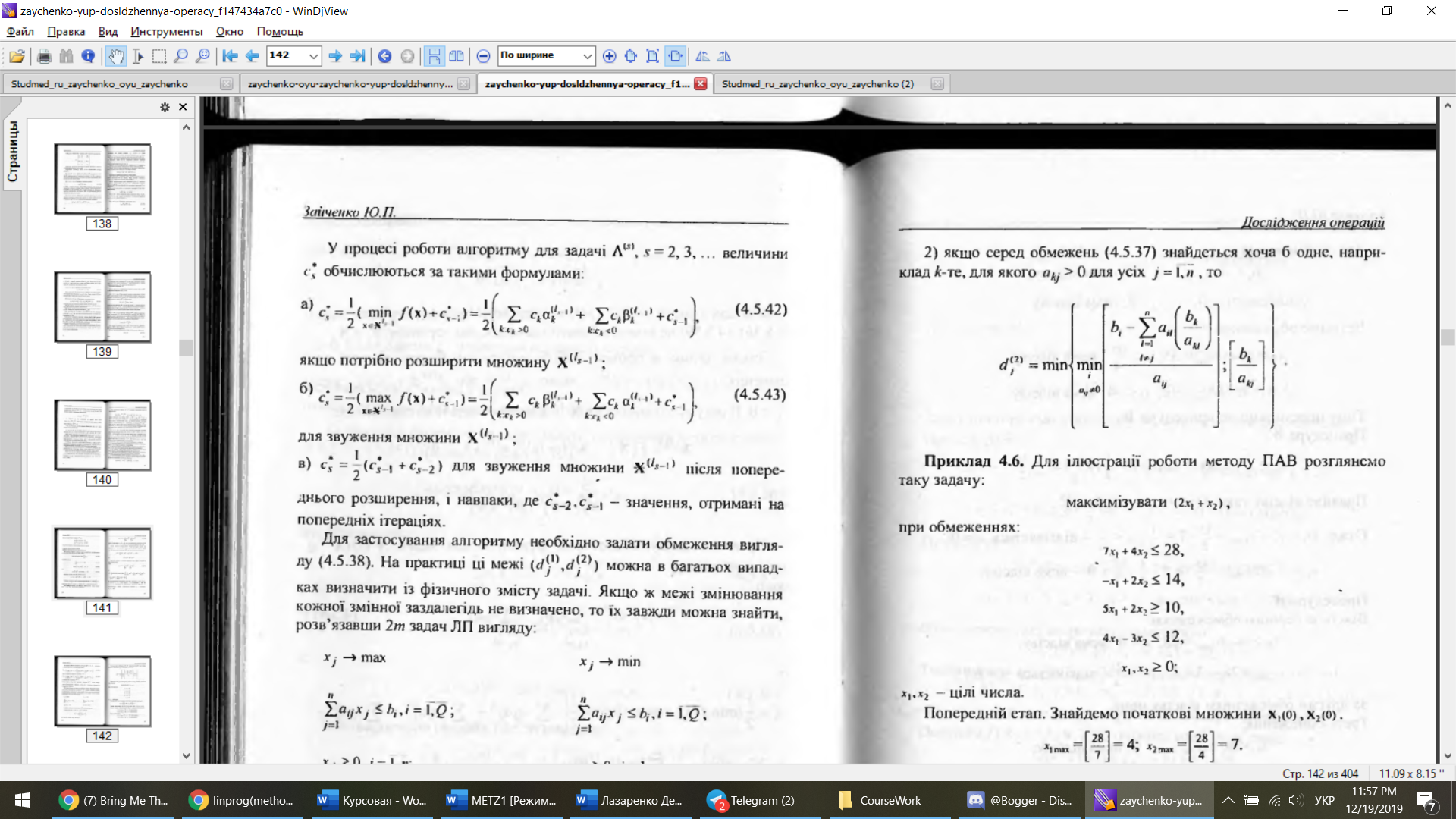
### 

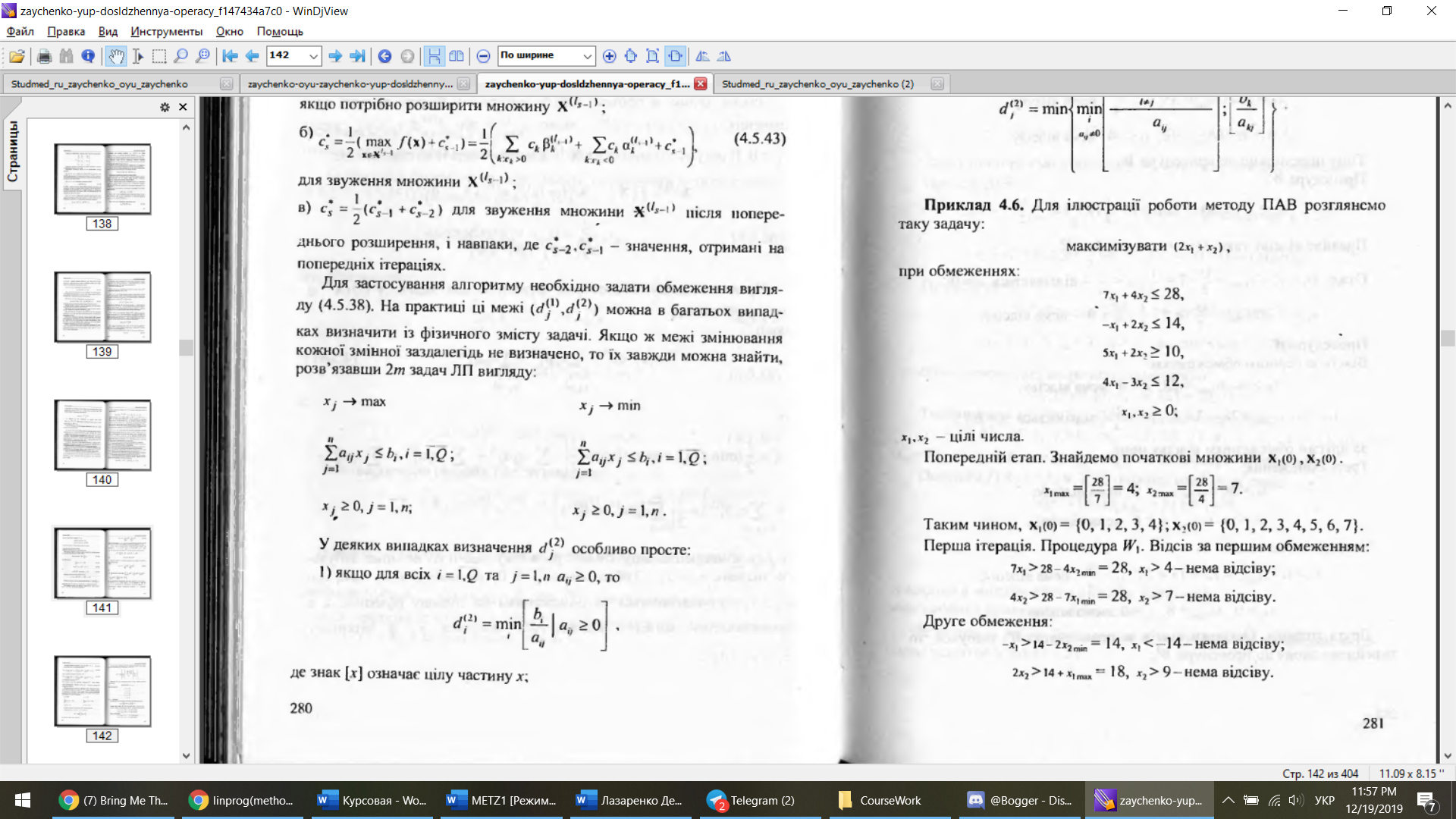
#### 3.2 Метод послідовного аналізу та відсіву варіантів для задачі цілочисельного програмування

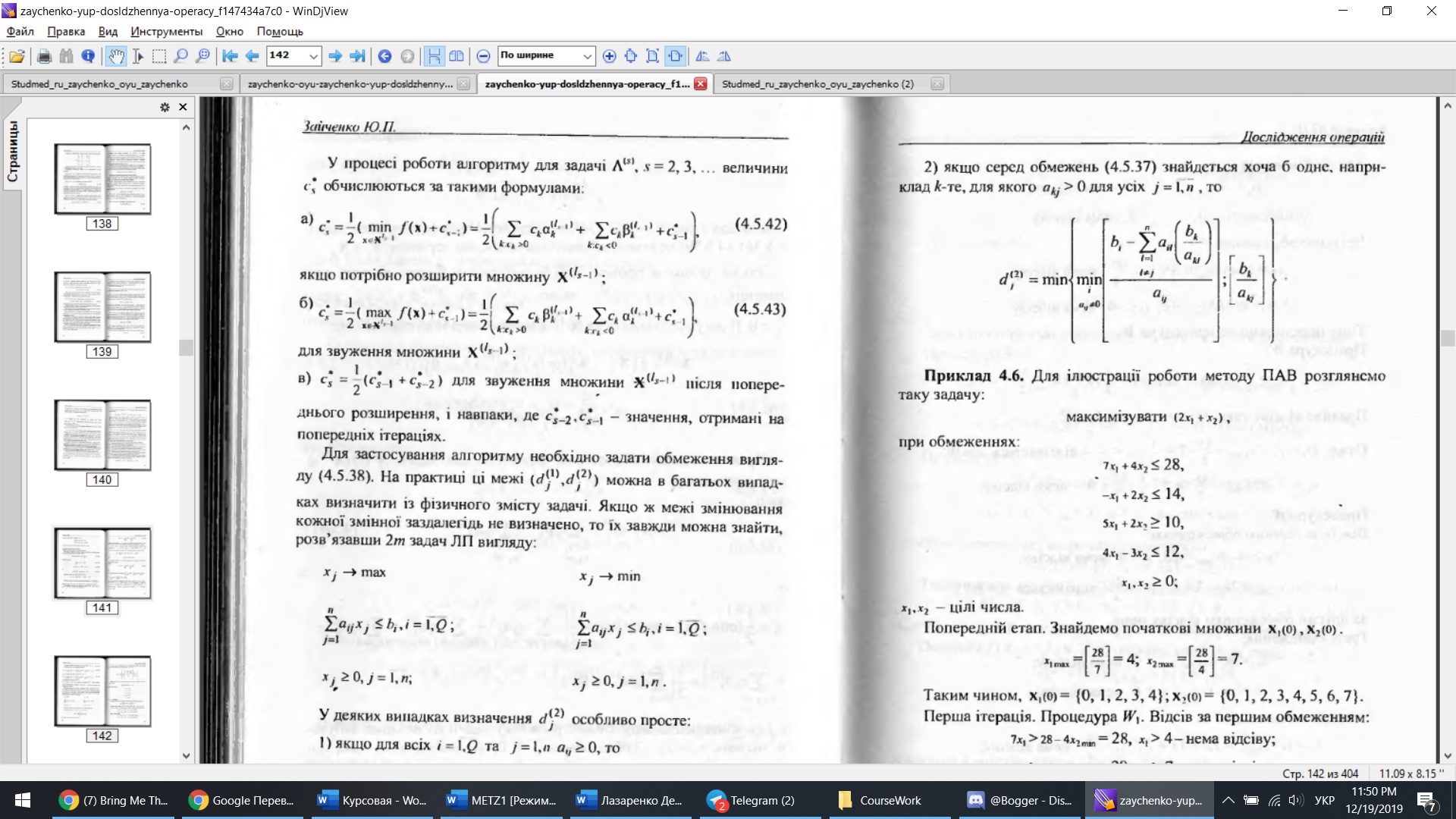










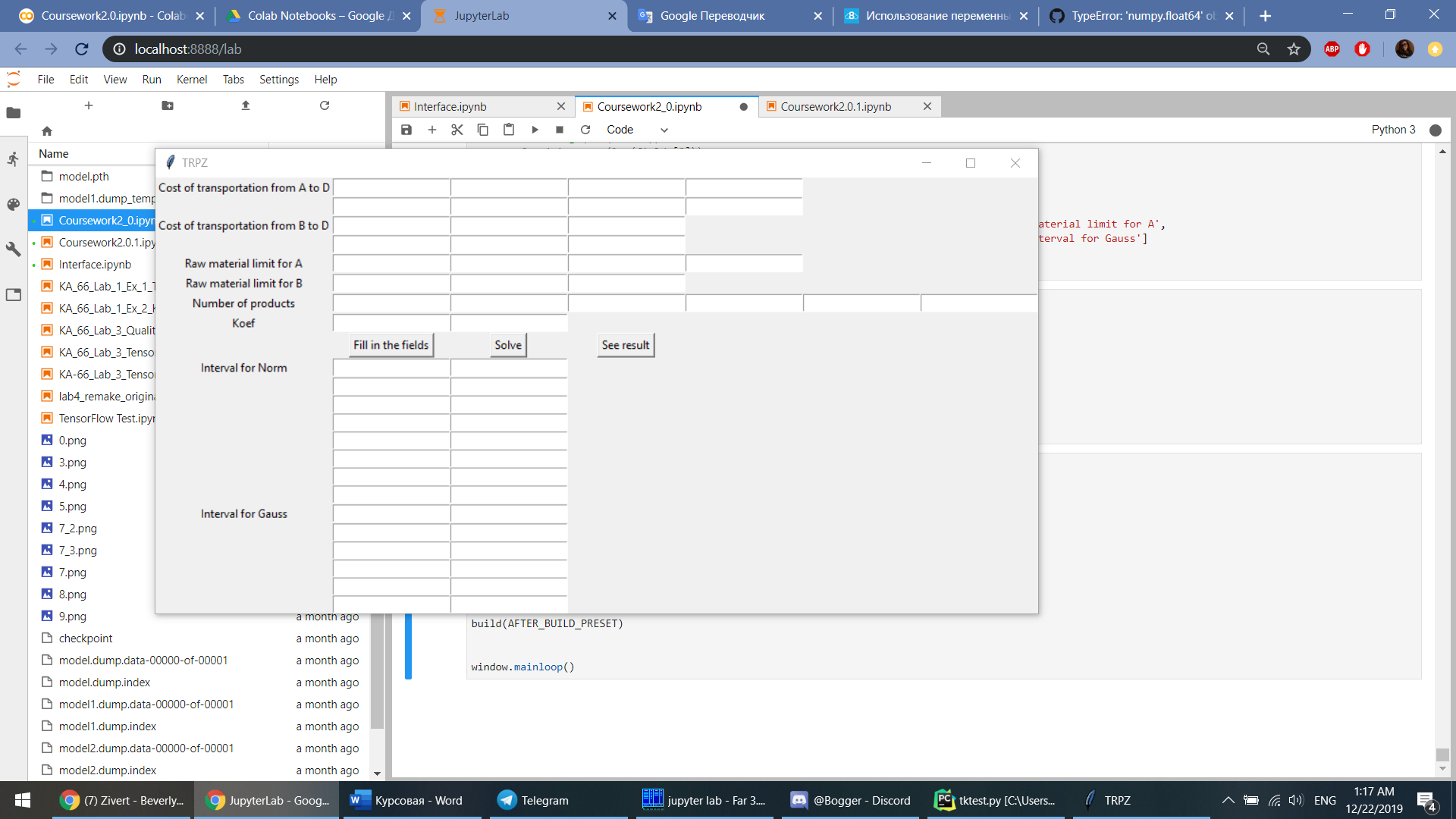


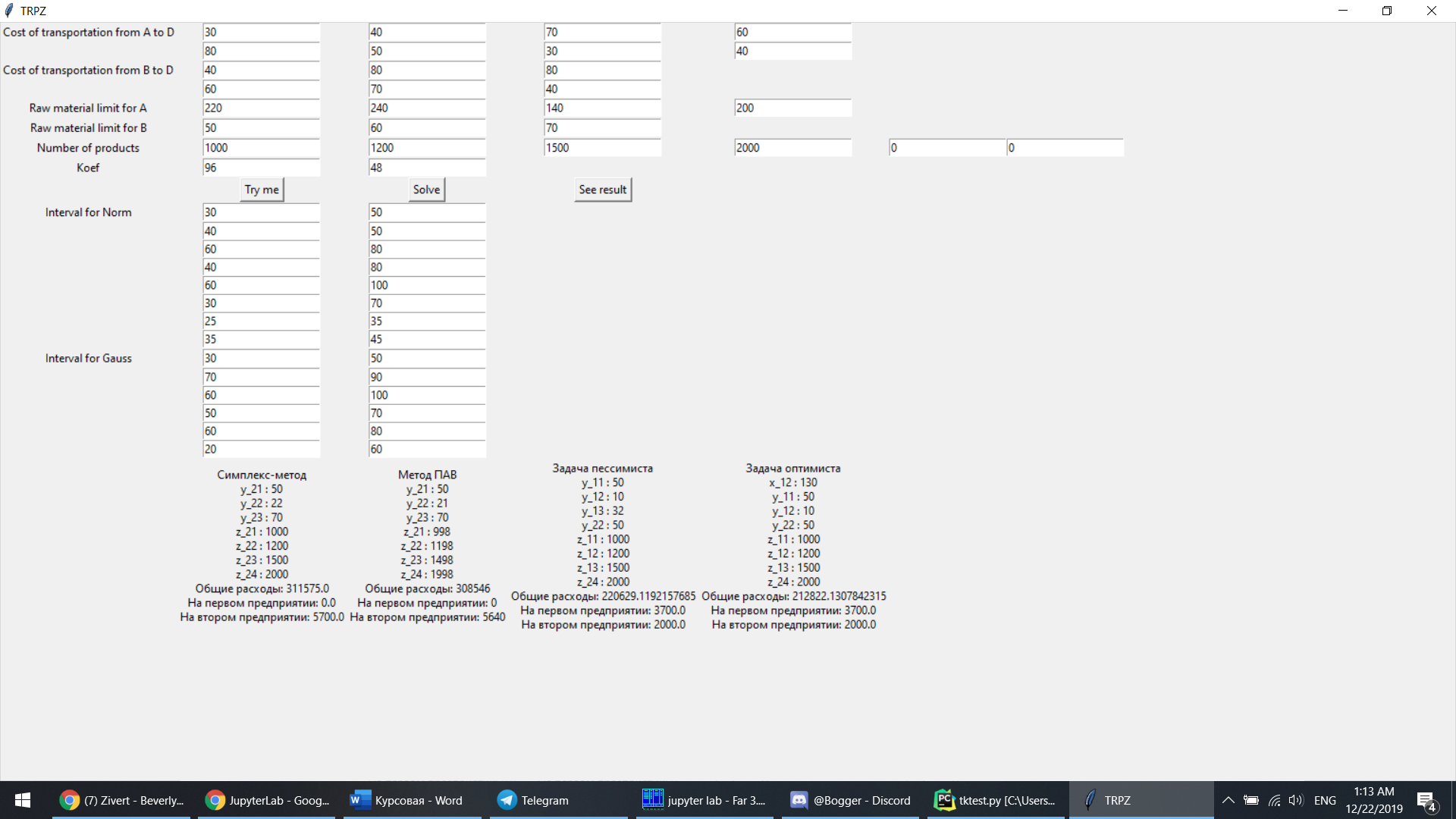
**4. Опис програми**

**4.1 Функції, які реалізують метод послідовного аналізу та відсіву варіантів**

Class PAV() – клас із вбудованими функціями, що реалізують метод ПАВ

Linprog() – вбудована функція мови Python (окремої бібліотеки) для реалізації симплекс-методу

**4.2. Програма для розв'язку задачі**



**4.3 Інструкція користувача**

Програма реалізована за допомогою мови програмування Python в середовищі розробки JupyterNotebook та JupyterLab. Користувач може скористуватися трьома кнопками: “Fill in the fields”, що автоматично заповнює поля, “Solve” – розв’язує задачу с даними параметрами, “See result” – виводить результат на екран

**5. Аналіз отриманих результатів**

**5.1 Аналіз роботи програми**

**Вхідні дані:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Предприятие | Пункт добычи сырья | | | |
|  |  |  |  |  |
|  | 30 | 40 | 70 | 60 |
|  | 80 | 50 | 30 | 40 |

Таблиця 19

|  |  |  |  |
| --- | --- | --- | --- |
| Предприятие | Пункт добычи сырья | | |
|  |  |  |  |
|  | 40 | 80 | 80 |
|  | 60 | 70 | 40 |

Таблиця 20

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Предприятие | Базы | | | |
|  |  |  |  |  |
|  | 4 | 4 | 10 | 8 |
|  | 8 | 3 | 6 | 5 |

Таблиця 21

**Результат:**

Общие расходы: 311575.0

y\_21 : 50

y\_22 : 22

y\_23 : 70

z\_21 : 1000

z\_22 : 1200

z\_23 : 1500

z\_24 : 2000

Объем продукции на первом предприятии: 0.0

Объем продукции на втором предприятии: 5700.0

**Вхідні дані:**

Спробуємо змінити витрати. Наприклад, змінимо витрати на траспортування від пунктів видобучі до підприємтв.

Нехай обмеження будуть такими:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Предприятие | Пункт добычи сырья | | | |
|  |  |  |  |  |
|  | 20 | 40 | 50 | 60 |
|  | 80 | 30 | 10 | 40 |

Таблиця 19

|  |  |  |  |
| --- | --- | --- | --- |
| Предприятие | Пункт добычи сырья | | |
|  |  |  |  |
|  | 80 | 80 | 80 |
|  | 60 | 90 | 50 |

Таблиця 20

**Результат:**

Общие расходы: 311770.0

x\_23 : 140

y\_21 : 44

y\_23 : 70

z\_21 : 1000

z\_22 : 1200

z\_23 : 1500

z\_24 : 2000

Объем продукции на первом предприятии: 0.0

Объем продукции на втором предприятии: 5700.0

**Аналіз:**

Можемо помітити, що результат – змінився. Це показує вплив витрат на транспортування сировини на цільову функцію.

**Вхідні дані:**

Тепер спробуємо змінити витрати на траспортування до баз. Та залишимо такі ж ціни на транспортування до підприємтсв.

Нехай обмеження будуть такими:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Предприятие | Пункт добычи сырья | | | |
|  |  |  |  |  |
|  | 20 | 40 | 50 | 60 |
|  | 80 | 30 | 10 | 40 |

Таблиця 19

|  |  |  |  |
| --- | --- | --- | --- |
| Предприятие | Пункт добычи сырья | | |
|  |  |  |  |
|  | 80 | 80 | 80 |
|  | 60 | 90 | 50 |

Таблиця 20

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Предприятие | Базы | | | |
|  |  |  |  |  |
|  | 8 | 3 | 5 | 2 |
|  | 30 | 10 | 10 | 9 |

Також змінимо відношення між виробництвом продукції. Нехай = 8, =10.

**Результат:**

Общие расходы: 310080.0

x\_22 : 240

x\_23 : 140

x\_24 : 190

z\_21 : 1000

z\_22 : 1200

z\_23 : 1500

z\_24 : 2000

Объем продукции на первом предприятии: 0.0

Объем продукции на втором предприятии: 5700.0

**Аналіз:**

Можемо помітити, що результат – змінився. Бачимо, що вигідніше добувати сировину на першому пункті з іншими об’ємами, та транспортувати другому підприємству. На бази краще транспортувати також з другого підприємства.

**Вхідні дані для нечіткої задачі:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Предприятия | Пункты добычи сырья | | | |
|  |  |  |  |  |
|  | [30;50] | [40;50] | [60;80] | [40;80] |
|  | [60;100] | [30;70] | [25;35] | [35;45] |

Таблиця 22

|  |  |  |  |
| --- | --- | --- | --- |
| Предприятия | Пункты добычи сырья | | |
|  |  |  |  |
|  | [30;50] | [70;90] | [60;100] |
|  | [50;70] | [60;80] | [20;60] |

Таблиця 23

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Предприятия | Базы j | | | |
|  |  |  |  |  |
|  | 40/40 | 25/45 | 30/20 | 80/40 |
|  | 80/40 | 30/15 | 60/30 | 50/50 |

Таблиця 24

**Результат:**

**Задача песиміста**

Общие расходы: 224640.77988269064

y\_11 : 50

y\_12 : 10

y\_13 : 32

y\_22 : 50

z\_11 : 1000

z\_12 : 1200

z\_13 : 1500

z\_24 : 2000

Объем продукции на первом предприятии: 3700.0

Объем продукции на втором предприятии: 2000.0

**Задача оптиміста**

Общие расходы: 208739.80749234871

x\_12 : 130

y\_11 : 50

y\_12 : 10

y\_22 : 50

z\_11 : 1000

z\_12 : 1200

z\_13 : 1500

z\_24 : 2000

Объем продукции на первом предприятии: 3700.0

Объем продукции на втором предприятии: 2000.0

**Аналіз:**

Бачимо, що для задачі оптиміста загальні витрати набагато менші, аніж для задачі песиміста, що є гарним результатом. Також, для задачі оптиміста використовуються інші бази видобування. Для цього варіанту ми використали рівень α = 0,2. Проаналізуємо інші значення α.

**Вхідні дані:**

Нехай α = 0,4

**Результат:**

**Задача песиміста**

Общие расходы: 222627.7263825725

y\_11 : 50

y\_12 : 10

y\_13 : 32

y\_22 : 50

z\_11 : 1000

z\_12 : 1200

z\_13 : 1500

z\_24 : 2000

Объем продукции на первом предприятии: 3700.0

Объем продукции на втором предприятии: 2000.0

**Задача оптиміста**

Общие расходы: 212822.1307842315

x\_12 : 130

y\_11 : 50

y\_12 : 10

y\_22 : 50

z\_11 : 1000

z\_12 : 1200

z\_13 : 1500

z\_24 : 2000

Объем продукции на первом предприятии: 3700.0

Объем продукции на втором предприятии: 2000.0

**Аналіз:**

Бачимо, що результат не змінився, тому спробуємо ще раз змінити α.

**Вхідні дані:**

Нехай α = 0,8

**Результат:**

**Задача песиміста**

Общие расходы: 220629.1192157685

y\_11 : 50

y\_12 : 10

y\_13 : 32

y\_22 : 50

z\_11 : 1000

z\_12 : 1200

z\_13 : 1500

z\_24 : 2000

Объем продукции на первом предприятии: 3700.0

Объем продукции на втором предприятии: 2000.0

**Задача оптиміста**

Общие расходы: 212822.1307842315

x\_12 : 130

y\_11 : 50

y\_12 : 10

y\_22 : 50

z\_11 : 1000

z\_12 : 1200

z\_13 : 1500

z\_24 : 2000

Объем продукции на первом предприятии: 3700.0

Объем продукции на втором предприятии: 2000.0

**Аналіз:**

Результат знову не змінився, це означає, що наш рівень не впливає на задачу.

**5.2 Аналіз результатів**

Рішенням даної задачі є:

* Сумарне значення усіх витрат
* Вектор значень, який показує кількість потрібної для видобування сировини на різних пунктах.
* Вектор значень, який показує кількість продукту, що потрібно транспортувати на базу
* Вектор значень, що показує об’єми продукту на двух підприємствах.

Дослідження завдання дало наступні результати:

 - при будь-якій зміні витрат на транспортування та обробки сировини – цільова функція і план видобутку сировини змінюється

- при зміні відношення між кількістю потрібної сировини – план транспортування та обробки сировини змінюється

- модель не надто чутлива до зміни параметра α, так як в порівнянні з іншими числами є невеликою.

Метод відрізняється досить швидкої збіжністю.

**Висновки**

У цій роботі, було визначено оптимальний план по виробництву певного продукту. Дослідження було проведено за допомогою двух методів дискретного програмування.

Також, так як симлекс-метод іноді не дає цілочисельної відповіді, ми використали метод послідовного аналізу та відсіву варіантів, для ще більшої точності.

При аналізі наших даних та програми, побачили певні змінні у результатах, що показує залежність між частинними витратами та сумарними. Те ж саме можна побачити при змінах відношень виробництва продукту.

Список літератури

1. Конспект лекций по курсу «Исследование операций» Е. Ю. Зайченко.
2. Конспект лекций по курсу «Теория принятия решений в сложных системах» Ю.П.Зайченко.
3. Зайченко Ю. П. Исследование операций. – 3-е изд., перераб. и доп. – К.: Выща шк. Головное изд-во, 1988. – 552 с.

**Лістинг програми**

from \_\_future\_\_ import division

import numpy as np

import scipy

import math

**Затраты и объемы**

C\_A = np.array([[30, 40, 70, 60], [80, 50, 30, 40]]) #затраты на транспортировку 1-ого вида сырья на предприятия D1, D2

C\_B = np.array([[40, 80, 80], [60, 70, 40]]) #затраты на траспортировку 2-ого вида сырья не предприятия D1, D2

b\_ub\_A = np.array([220, 240, 140, 200]) #ограничения в неравенствах

b\_ub\_B = np.array([50, 60, 70])

b\_eq = np.array([1000, 1200, 1500, 2000, 0, 0]) #ограниченияв равенствах

C\_AB = np.concatenate((C\_A, C\_B), axis = None) #все затраты на транспортировку сырья на предприятия

print(C\_AB)

ksi = np.array([96,48]) #коэффициент возле объемов

fm = [1382400] #свободный член

print(ksi)

C\_d1 = np.append((ksi[0] \* np.array([[10, 10, 10, 10]])), ksi[1]\*np.array([8, 8, 8, 8]))

C\_d2 = np.append((ksi[0] \* np.array([[40, 40, 40]])), ksi[1]\*np.array([40, 40, 40]))

print(C\_d1)

#print(C\_d2)

C\_d = np.concatenate((C\_d1, C\_d2), axis = None) #затраты на обработку сырья на двух предприятиях

print(C\_d)

C\_ABD = C\_AB + C\_d

print(C\_ABD) #затраты на добычу,транспортировку и обработку сырья

C\_D1\_E = np.array([4, 4, 10, 8]) #затраты на перевозку продукции с первого предприятия на базы E

C\_D2\_E = np.array([8, 3, 6, 5]) #затраты на перевозку продукции со второго предприятия на базы Е

C\_D\_E = np.concatenate((C\_D1\_E, C\_D2\_E), axis = None) #все затраты на перевозку к базам

print(C\_D\_E)

C = np.append(C\_ABD, C\_D\_E)

print(C) #все коэффициенты целевой функции

**Ограничения**

b\_ub = np.concatenate((b\_ub\_A, b\_ub\_B), axis = None)

A\_ub\_A = np.array([[1, 0, 0, 0, 1, 0, 0, 0], #коэффициенты для переменных в неравенствах для X

[0, 1, 0, 0, 0, 1, 0, 0],

[0, 0, 1, 0, 0, 0, 1, 0],

[0, 0, 0, 1, 0, 0, 0, 1],

[0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0, 0, 0]])

A\_ub\_B = np.array([[0, 0, 0, 0, 0, 0], #коэффициенты для переменных в неравенствах для Y

[0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0],

[0, 0, 0, 0, 0, 0],

[1, 0, 0, 1, 0, 0],

[0, 1, 0, 0, 1, 0],

[0, 0, 1, 0, 0, 1]])

A\_ub\_Z = np.zeros([7,8])

***Общая матрица коэффициентов для неравенств***

A\_ub = np.concatenate((A\_ub\_A, A\_ub\_B, A\_ub\_Z), axis=1)

print(A\_ub)

***Найдём такую же матрицу для равенств***

A\_eq\_XY = np.zeros([4,14])

A\_eq\_z = np.array([[1, 0, 0, 0, 1, 0, 0, 0],

[0, 1, 0, 0, 0, 1, 0, 0],

[0, 0, 1, 0, 0, 0, 1, 0],

[0, 0, 0, 1, 0, 0, 0, 1]]) #коэффициенты для переменных в равенствах для Zkj

A\_eq\_1 = np.concatenate((A\_eq\_XY,A\_eq\_z), axis=1) #первые 4 ограничения в равенствах

#print(A\_eq)

***Определим коэффициенты для объемов***

Z1 = np.array([[10, 10, 10, 10, 0, 0, 0, 0, 40, 40, 40, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]])

Z2 = np.array([[0, 0, 0, 0, 8, 8, 8, 8, 0, 0, 0, 40, 40, 40, 0, 0, 0, 0, 0, 0, 0, 0]])

***Введем коэффициенты для продукции***

A\_Z1 = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 0, 0, 0]) #коэффициенты для ограничений

A\_Z2 = np.array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1])

A\_eq\_Z1 = A\_Z1-Z1 #общие коэффициенты, при переносе всех переменных в левую часть

A\_eq\_Z2 = A\_Z2-Z2

#print(A\_eq\_Z1)

A\_eq\_Z = np.concatenate((A\_eq\_Z1, A\_eq\_Z2), axis = 0)

print(A\_eq\_Z)

***Общая матрица коэффициентов для равенств***

A\_eq = np.concatenate((A\_eq\_1, A\_eq\_Z), axis=0)

print(A\_eq)

print(C)

print(A\_ub)

print(b\_ub)

print(A\_eq)

print(b\_eq)

from scipy.optimize import linprog

res = linprog(C, A\_ub=A\_ub, b\_ub=b\_ub, A\_eq=A\_eq, b\_eq=b\_eq,

method='simplex',

options={"disp": True},

callback = lambda \*x, \*\*kwargs:"tableau" )

print(res)

V1\_Ch = np.dot(res.get('x')[0:22], Z1[0])

V2\_Ch = np.dot(res.get('x')[0:22], Z2[0])

Xvar = np.array([[0, 1],

[0, 1],

[0, 1],

[0, 1],

[0, 1],

[0, 1],

[0, 1],

[0, 1],

[0, 1],

[0, 1],

[0, 1],

[48, 50],

[21, 23],

[68, 70],

[0, 1],

[0, 1],

[0, 1],

[0, 1],

[998, 1000],

[1198, 2000],

[1498, 1500],

[1998, 2000]])

***Метод ПАВ***

class PAV():

def \_\_init\_\_(self, cost, A\_ub, b\_ub, Xvar, A\_eq=None, b\_eq=None):

self.cost = cost

self.f = 0

self.A\_ub = A\_ub

self.b\_ub = b\_ub

self.A\_eq = A\_eq

self.h = b\_eq

self.Xvar = Xvar

self.W1\_flag = 0

self.W2\_flag = 1

def max\_or\_min\_W1(self, i, j, mode, flag=0):

# flag==0 :j

# flag==1 j+1:

if (flag):

return [np.max(self.Xvar[j+1+t]) if self.A\_ub[i][j+1+t] > 0 else mode(self.Xvar[j+1+t]) for t in range(len(self.A\_ub[i,j+1:]))]

else:

return [np.max(self.Xvar[t]) if self.A\_ub[i][t] > 0 else mode(self.Xvar[t]) for t in range(len(self.A\_ub[i,:j]))]

def Ax(self, i, j, mode=None):

return self.A\_ub[i,:j].dot(self.max\_or\_min\_W1(i, j, mode, 0)) + self.A\_ub[i, j+1:].dot(self.max\_or\_min\_W1(i, j, mode, 1)) if j!=0 else self.A\_ub[i, j+1:].dot(self.max\_or\_min\_W1(i, j, mode, 1))

def check\_W1(self, i, j):

b1 = (self.b\_ub[i] - self.Ax(i, j, np.min))/self.A\_ub[i][j]

print('{:.0f}x{:.0f}'.format(self.A\_ub[i][j], j), '<', self.b\_ub[i], '-', self.Ax(i,j,np.min), 'X')

print('x{:.0f}'.format(j), '<', b1)

print('x{:.0f}'.format(j), 'in({:.0f},{:.0f})'.format(self.Xvar[j][0], self.Xvar[j][1]))

if (self.Xvar[j][0] < b1) & (b1>0) & (self.Xvar[j][1]>=b1):

print('Отсев')

self.W1\_flag = 1

if (b1 - int(b1)>0):

self.Xvar[j][0] = int(b1)+1

else:

self.Xvar[j][0] = int(b1)

print("Новые ограниечения", 'x{:.0f}'.format(j), 'in({:.0f},{:.0f})'.format(self.Xvar[j][0], self.Xvar[j][1]))

def filter\_W1(self):

self.W1\_flag = 0

for j in range(self.A\_ub.shape[1]): # переменная по которой идем

print("переменная", j)

for i in range(self.A\_ub.shape[0]): # уравнение, которое берем

print("уравенение", i)

self.check\_W1(i,j)

def W1(self):

if not self.stop\_point():

print('----------------------------W1----------------------------------')

if self.W2\_flag == 1:

self.W2\_flag = 0

self.W1\_flag = 1

else:

raise NameError('Error')

k = 0

while (self.W1\_flag == 1) & (k<1000) & (not self.stop\_point()):

print("КВ-ВО ИТЕРАЦИЙ W1", k)

self.filter\_W1()

k+=1

else:

pass

def max\_func(self):

return sum([self.cost[i]\*Xvar[i][1] if self.cost[i]>0 else self.cost[i]\*Xvar[i][0] for i in range(len(self.cost))])

def min\_func(self):

return sum([self.cost[i]\*Xvar[i][0] if self.cost[i]>0 else self.cost[i]\*Xvar[i][1] for i in range(len(self.cost))])

def max\_or\_min\_W2(self, i, mode, flag=0):

if (flag):

return [np.max(self.Xvar[i+1+t]) if self.cost[i] < 0 else mode(self.Xvar[i+1+t]) for t in range(len(self.cost[i+1:]))]

else:

return [np.max(self.Xvar[t]) if self.cost[i] < 0 else mode(self.Xvar[t]) for t in range(len(self.cost[:i]))]

def Cx(self, i, mode=None):

return self.cost[:i].dot(self.max\_or\_min\_W2(i, mode, 0)) + self.cost[i+1:].dot(self.max\_or\_min\_W2(i, mode, 1)) if i!=0 else self.cost[i+1:].dot(self.max\_or\_min\_W2(i, mode, 1))

def check\_W2(self, i):

b1 = (self.f - self.Cx(i, np.min))/self.cost[i]

print('{:.0f}x{:.0f}'.format(self.cost[i], i), '>', self.f, '-', self.Cx(i, np.min), 'X')

print('x{:.0f}'.format(i), '>', b1)

print('x{:.0f}'.format(i), 'in({:.0f},{:.0f})'.format(self.Xvar[i][0], self.Xvar[i][1]))

if (self.Xvar[i][1] > b1) & (b1>=0) & (self.Xvar[i][0]<=b1):

print('Отсев')

self.W2\_flag = 1

self.Xvar[i][1] = int(b1)

print("Новые ограниечения", 'x{:.0f}'.format(i), 'in({:.0f},{:.0f})'.format(self.Xvar[i][0], self.Xvar[i][1]))

def filter\_W2(self):

for i in range(self.cost.shape[0]): # переменная, которую берем

print("переменная", i)

self.check\_W2(i)

def W2(self):

if not self.stop\_point():

print('----------------------------W2----------------------------------')

if self.W1\_flag == 1:

raise NameError('Error')

self.f = 1/2\*(self.min\_func() + self.max\_func())

k = 0

while (self.W2\_flag == 0) & (k<1000) & (not self.stop\_point()):

print("КВ-ВО ИТЕРАЦИЙ W2", k)

self.filter\_W2()

self.f = 1/2\*(self.f + self.min\_func())

k+=1

else:

pass

def stop\_point(self):

return sum([self.Xvar[i][0] == self.Xvar[i][1] for i in range(self.Xvar.shape[1])]) == self.Xvar.shape[1]

def main(self):

k = 0

while (not self.stop\_point()) & (k<1000):

print("--------------------------ИТЕРАЦИЯ----------------------------------------:", k)

self.W1()

self.W2()

k+=1

X = [Xvar[i][0] for i in range(Xvar.shape[0])]

return X

instance = PAV(C, A\_ub, b\_ub, Xvar)

X\_opt = instance.main()

print(X\_opt)

Z = Z1 +Z2

print(len(Z[0]))

Cost = np.dot(X\_opt, C)

V1 = np.dot(X\_opt[0:22], Z1[0])

V2 = np.dot(X\_opt[0:22], Z2[0])

print(V1, V2)

print(Cost)

E\_A = np.array([[30, 50], [40, 50], [60, 80], [40,80], [60, 100], [30, 70], [25, 35], [35,45]]) #отрезок для равномерного распределения первого пункта добычи сырья

E\_B = np.array([[30, 50], [70, 90], [60, 100], [50, 70], [60, 80], [20, 60]]) #отрезок для равномерного распределения второго пункта добычи сырья

alpha = np.array([0.2, 0.4, 0.6, 0.8]) #уровень альфа

k = 3

print(alpha[0])

scatter = np.array([])

scatter2 = np.array([])

a = np.zeros([4])

b = np.zeros([4])

a = math.sqrt((1-alpha[k])/alpha[k]) #разброс для равномерного

scatter = np.append(scatter, a)

b = math.sqrt(2\*math.log(1/alpha[k], math.e)) #разброс для гауссовского

scatter2 = np.append(scatter2, b)

print(scatter)

print(scatter2)

M = np.array([40, 25, 30, 80, 80, 30, 60, 50]) #мат ожидание для гауссовского распределения

#scatter2 = math.sqrt(2\*math.log(1/alpha, math.e)) #разброс для гауссовского

#print(scatter2)

***Задача пессимиста***

E\_AB = np.concatenate((E\_A, E\_B), axis = 0 )

print(E\_AB)

def koef\_p():

C\_U\_p = np.array([])

C\_u\_p = np.array([])

e\_p = np.zeros([14])

c\_p = np.zeros([8])

#print(e)

for i in range(len(E\_AB)):

e\_p[i] = (E\_AB[i][1] - E\_AB[i][0])/2 + scatter

C\_U\_p = np.append(C\_U\_p, e\_p[i])

for j in range(len(M)):

c\_p[j] = M[j] + scatter2

C\_u\_p = np.append(C\_u\_p, c\_p[j])

c\_p = np.concatenate((C\_U\_p, C\_u\_p), axis = None)

return c\_p

print(koef\_p()) #все коэффициенты целевой функции

from scipy.optimize import linprog

res\_p = linprog(koef\_p(), A\_ub=A\_ub, b\_ub=b\_ub, A\_eq=A\_eq, b\_eq=b\_eq,

method='simplex',

options={"disp": True},

callback = lambda \*x, \*\*kwargs:"tableau" )

print(res\_p)

Xvar\_p = np.array([[0, 1],

[0, 1],

[0, 1],

[0, 1],

[0, 1],

[0, 1],

[0, 1],

[0, 1],

[49, 50],

[9, 10],

[32, 33],

[0, 1],

[49, 50],

[0, 1],

[999, 1000],

[1199, 1200],

[1499, 1500],

[0, 1],

[0, 1],

[0, 1],

[0, 1],

[1999, 2000]])

V1\_p = np.dot(res\_p.get('x')[0:22], Z1[0])

V2\_p = np.dot(res\_p.get('x')[0:22], Z2[0])

***Задача оптимиста***

E\_AB = np.concatenate((E\_A, E\_B), axis = 0 )

def koef\_o():

C\_U\_o = np.array([])

C\_u\_o = np.array([])

e\_o = np.zeros([14])

c\_o = np.zeros([8])

#print(e)

for i in range(len(E\_AB)):

e\_o[i] = (E\_AB[i][1] - E\_AB[i][0])/2 - scatter

C\_U\_o = np.append(C\_U\_o, e\_o[i])

for j in range(len(M)):

c\_o[j] = M[j] - scatter2

C\_u\_o = np.append(C\_u\_o, c\_o[j])

c\_o = np.concatenate((C\_U\_o, C\_u\_o), axis = None)

return c\_o

print(koef\_o()) #все коэффициенты целевой функции

from scipy.optimize import linprog

res\_o = linprog(koef\_o(), A\_ub=A\_ub, b\_ub=b\_ub, A\_eq=A\_eq, b\_eq=b\_eq,

method='simplex',

options={"disp": True},

callback = lambda \*x, \*\*kwargs:"tableau" )

print(res.get('x'))

V1\_o = np.dot(res\_p.get('x')[0:22], Z1[0])

V2\_o = np.dot(res\_p.get('x')[0:22], Z2[0])

***Оформление ответов***

var = ['x\_11', 'x\_12', 'x\_13', 'x\_14', 'x\_21', 'x\_22', 'x\_23', 'x\_24', 'y\_11', 'y\_12', 'y\_13', 'y\_21', 'y\_22', 'y\_23', 'z\_11', 'z\_12', 'z\_13', 'z\_14', 'z\_21', 'z\_22', 'z\_23', 'z\_24']

def answ(answ):

counter\_list = list(enumerate(answ, 1))

for i in counter\_list:

#print(i)

if i[1] != 0:

print(var[i[0]-1], ':', math.floor(i[1]))

***Ответы***

***Чёткая задача***

print('Общие расходы:', res.get('fun'))

answ(res.get('x'))

print('Объем продукции на первом предприятии:', V1\_Ch)

print('Объем продукции на втором предприятии:', V2\_Ch)

print('Общие расходы:', Cost)

answ(X\_opt)

print('Объем продукции на первом предприятии:', V1)

print('Объем продукции на втором предприятии:', V2)

***Нечёткая задача***

**Задача пессимиста**

print('Общие расходы:', res\_p.get('fun'))

answ(res\_p.get('x'))

print('Объем продукции на первом предприятии:', V1\_p)

print('Объем продукции на втором предприятии:', V2\_p)

**Задача оптимиста**

print('Общие расходы:', res\_o.get('fun'))

answ(res\_o.get('x'))

print('Объем продукции на первом предприятии:', V1\_o)

print('Объем продукции на втором предприятии:', V2\_o)

***Интерфейс***

from tkinter import \*

MAIN\_BUILD\_PRESET = [

('Cost of transportation from A to D', 2, 4),

('Cost of transportation from B to D', 2, 3),

('Raw material limit for A', 1, 4),

('Raw material limit for B', 1, 3),

('Number of products', 1, 6),

('Koef', 1, 2),

]

AFTER\_BUILD\_PRESET = [

('Interval for Norm', 8, 2),

('Interval for Gauss', 6, 2),

]

LABELS\_ADDRESS = {}

DATA = {

'Cost of transportation from A to D': [[30, 40, 70, 60], [80, 50, 30, 40]],

'Cost of transportation from B to D': [[40, 80, 80], [60, 70, 40]],

'Raw material limit for A': [[220, 240, 140, 200]],

'Raw material limit for B': [[50, 60, 70]],

'Number of products': [[1000, 1200, 1500, 2000, 0, 0]],

'Koef' : [[96, 48]],

'Interval for Norm': [[30, 50], [40, 50], [60, 80], [40,80], [60, 100], [30, 70], [25, 35], [35,45]],

'Interval for Gauss': [[30, 50], [70, 90], [60, 100], [50, 70], [60, 80], [20, 60]],

}

window = Tk()

window.title("TRPZ")

build(MAIN\_BUILD\_PRESET)

row = window.grid\_size()[1]

Button(window, text="Fill in the fields", command = fill\_fields).grid(column=1, row=row)

Button(window, text="Solve", command = solve).grid(column=2, row=row)

Button(window, text="See result", command = result).grid(column=3, row=row)

build(AFTER\_BUILD\_PRESET)

window.mainloop()

def build(preset):

for p in preset:

start\_row = window.grid\_size()[1]

Label(window, text=p[0]).grid(row=start\_row)

LABELS\_ADDRESS[p[0]] = [[0 for j in range(p[2])] for i in range(p[1])]

for i in range(p[1]):

for j in range(p[2]):

LABELS\_ADDRESS[p[0]][i][j] = Entry(window, text="")

LABELS\_ADDRESS[p[0]][i][j].grid(row=i + start\_row, column=j + 1)

def set\_params(fields, info):

for i in range(len(fields)):

for j in range(len(fields[0])):

fields[i][j].delete(0, END)

fields[i][j].insert(0, str(info[i][j]))

def fill\_fields():

VAR = ['Cost of transportation from A to D', 'Cost of transportation from B to D', 'Raw material limit for A',

'Raw material limit for B', 'Number of products', 'Koef', 'Interval for Norm', 'Interval for Gauss']

for var in VAR:

set\_params(LABELS\_ADDRESS[var], DATA[var])

def result():

def answers(a,b,c,d, name):

result = [name]

counter\_list = list(enumerate(b, 1))

for i in counter\_list:

if i[1] != 0:

result.append(f'{var[i[0]-1]} : {math.floor(i[1])}')

result.append(f'Общие расходы: {a}')

#result.append(f'{answ(b)}')

result.append(f'На первом предприятии: {c}')

result.append(f'На втором предприятии: {d}')

return '\n'.join(result)

row = window.grid\_size()[1]

Label(window, text = answers(res.get('fun'), res.get('x'), V1\_Ch, V2\_Ch, 'Симплекс-метод')).grid(column = 1, row=row)

Label(window, text = answers(Cost, X\_opt, V1, V2, 'Метод ПАВ')).grid(column = 2, row=row)

Label(window, text = answers(res\_p.get('fun'), res\_p.get('x'), V1\_p, V2\_p, 'Задача пессимиста')).grid(column = 3, row=row)

Label(window, text = answers(res\_o.get('fun'), res\_o.get('x'), V1\_o, V2\_o, 'Задача оптимиста')).grid(column = 4, row=row)

def solve():

C\_A1 = get\_label('Cost of transportation from A to D')

C\_B1 = get\_label('Cost of transportation from B to D')

b\_ub\_A1 = get\_label('Raw material limit for A')[0]

b\_ub\_B1 = get\_label('Raw material limit for B')[0]

b\_eq1 = get\_label('Number of products')[0]