НАВЧАЛЬО-НАУКОВИЙ КОМПЛЕКС
«ІНСТИТУТ ПРИКЛАДНОГО СИСТЕМНОГО АНАЛІЗУ»
ПРИ НАЦІОНАЛЬНОМУ ТЕХНІЧНОМУ УНІВЕРСИТЕТІ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ»
КАФЕДРА МАТЕМАТИЧНИХ МЕТОДІВ СИСТЕМНОГО АНАЛІЗУ

Практична робота №5
з системного аналізу

«Виявлення пріоритетностей галузей і технологій військово-промислового комплексу на основі методу морфологічного аналізу»

Бригада № 3

Виконали:
студенти 4 курсу
групи КА-71
Бойко Павло
Зінченко Світлана
групи КА-77
Худіков Павло

Київ 2021

Завдання

Приймається рішення щодо пріоритетності галузей і технологій військово-промислового комплексу, опис якого у вигляді морфологічної таблиці заданий в табл. 2. Рішення приймається в умовах різних потенційних обставин у світі і в країні (табл. 1).

Табл. 1. Ситуація в країні і в світі

1. Напруженість в світі	2. Напруженість в країні	3. Стан економіки	4. Терористична загроза
1.1. Відносна стабільність	2.1. Затишшя	3.1. Оптимістичний	4.1. Сильна
1.2. Холодна війна	2.2. Заморожений конфлікт, збройні провокації	3.2. Стабільний	4.2. Помірна
1.3. Локальні конфлікти	2.3. Збройне протистояння	3.3. Спад	4.3. Слабка
1.4. Локальні війни	2.4. Повномасштабна агресія іншої держави	3.4. Криза/дефолт	

Таблиця 2. Морфологічна таблиця рішень щодо ВПК

5. Галузь	6. Технології	7. Експорт/використання
5.1. Піхотне озброєння	6.1. Приладобудування і електроніка	7.1. Переважно внутрішнє використання
5.2. Наземна техніка	6.2. Машинобудування	7.2. Переважно експорт
5.3. Повітряна техніка	6.3. Новітні матеріали	
5.4. Протиповітряна оборона, ракетні війська	6.4. Оптика	
5.5. Розвідка	6.5. Технології зв’язку	
5.6. Зв’язок і радіоелектронна боротьба		

Попередні оцінки ймовірності від експертів задані в табл. 3.

Табл. 3. Попередні оцінки ймовірності альтернатив параметрів

1. Напруженість в світі	2. Напруженість в країні	3. . Стан економіки	4. Терористична загроза
0,05	0,2	0,3	0,35
0,35	0,7	0,65	0,8
0,65	0,4	0,25	0,2
0,2	0,1	0,05	

Матриця взаємозв’язків альтернатив параметрів морфологічної таблиці першого етапу задана в табл. 4.

Табл. 4. Матриця взаємозв’язків альтернатив параметрів морфологічної таблиці першого етапу

		1				2				3			
		1.1	1.2	1.3	1.4	2.1	2.2	2.3	2.4	3.1	3.2	3.3	3.4
2	2.1	0,5	0	–0,3	–0,6								
	2.2	0	0,5	0	0								
	2.3	–0,5	0,3	0,5	0								
	2.4	–0,9	–0,5	–0,2	0								
3	3.1	0,2	0	–0,1	–0,2	0,4	0	–0,2	–0,7				
	3.2	0	0	0	0	0,2	0,2	0	–0,4				
	3.3	0	0	0	0	0	0	0	0				
	3.4	–0,4	–0,2	0	0	0	0	0	0,3				
4	4.1	–0,5	0	0,3	0,5	–0,3	–0,1	0,3	0,6	–0,2	0	0	0,2
	4.2	0	0	0	0	0	0	0	0,2	0	0	0	0
	4.3	0,5	0	–0,3	–0,5	0,3	0,1	–0,5	–0,6	0,2	0	0	–0,2

Матриця зв’язків морфологічних таблиць першого і другого етапів задана в табл. 5.

Табл. 5. Матриця зв’язків морфологічних таблиць першого і другого етапів

		5						6					7	
		5.1	5.2	5.3	5.4	5.5	5.6	6.1	6.2	6.3	6.4	6.5	7.1	7.2
1	1.1	0,2	–0,2	0,1	–0,1	0,4	0,2	0	0	0	0	0	0	–0,2
	1.2	0,1	0	0,2	0,3	0,5	0,2	0	0	0	0	0	0	0,2
	1.3	0,3	0,2	0,3	0,2	0,3	0,1	0	0	0	0	0	0	0,4
	1.4	0,4	0,5	0,5	0,4	0,3	0,2	0	0	0	0	0	0	0,6
2	2.1	0,2	–0,3	0	–0,4	0,2	0,1	0	0	0	0	0	0	0
	2.2	0,3	0	0	–0,3	0,5	0,4	0	0	0	0	0	0,4	0
	2.3	0,4	0,4	0,3	0	0,6	0,6	0	0	0	0	0	0,7	–0,2
	2.4	0,5	0,8	0,6	0,8	0,7	0,7	0	0	0	0	0	0,9	–0,5
3	3.1	0	0,2	0,5	0,5	0,3	0,4	0,5	0,4	0,7	0,6	0,6	0	0
	3.2	0,3	0,2	0,3	0,3	0,3	0,3	0,5	0,3	0,5	0,4	0,3	0	0,1
	3.3	0	0	–0,1	–0,2	0	0	0,3	0,2	0,1	0,1	0,1	0	0,3
	3.4	0	–0,2	–0,5	–0,3	0	0	0	0	–0,3	–0,3	–0,1	–0,1	0,5
4	4.1	0,5	0,2	0	0	0,8	0,5	0	0	0	0	0	0,3	0
	4.2	0,3	0	0	0	0,7	0,4	0	0	0	0	0	0,1	0
	4.3	0,2	0	0	0	0,2	0,1	0	0	0	0	0	0	0

Крім того, параметри морфологічної таблиці другого етапу пов’язані між собою матрицею взаємозв’язків (табл. 6).

Табл. 6. Матриця взаємозв’язків альтернатив параметрів морфологічної таблиці другого етапу

		5						6				
		5.1	5.2	5.3	5.4	5.5	5.6	6.1	6.2	6.3	6.4	6.5
6	6.1	0	0,3	0,7	0,7	0,5	0,8					
	6.2	0	0,7	0,8	0,5	0	0					
	6.3	0,5	0,5	0,7	0,7	0	0					
	6.4	0,5	0,1	0,4	0,7	0,7	0,2					
	6.5	0	0,1	0,5	0,5	0,7	0,8					
7	7.1	0,3	0,5	0,3	0,5	0,7	0,7	0,5	0,5	0,7	0,3	0,4
	7.2	0,4	0,6	0,2	0,5	0,2	0,2	0,6	0,7	0,3	0,2	0,2

Потрібно:

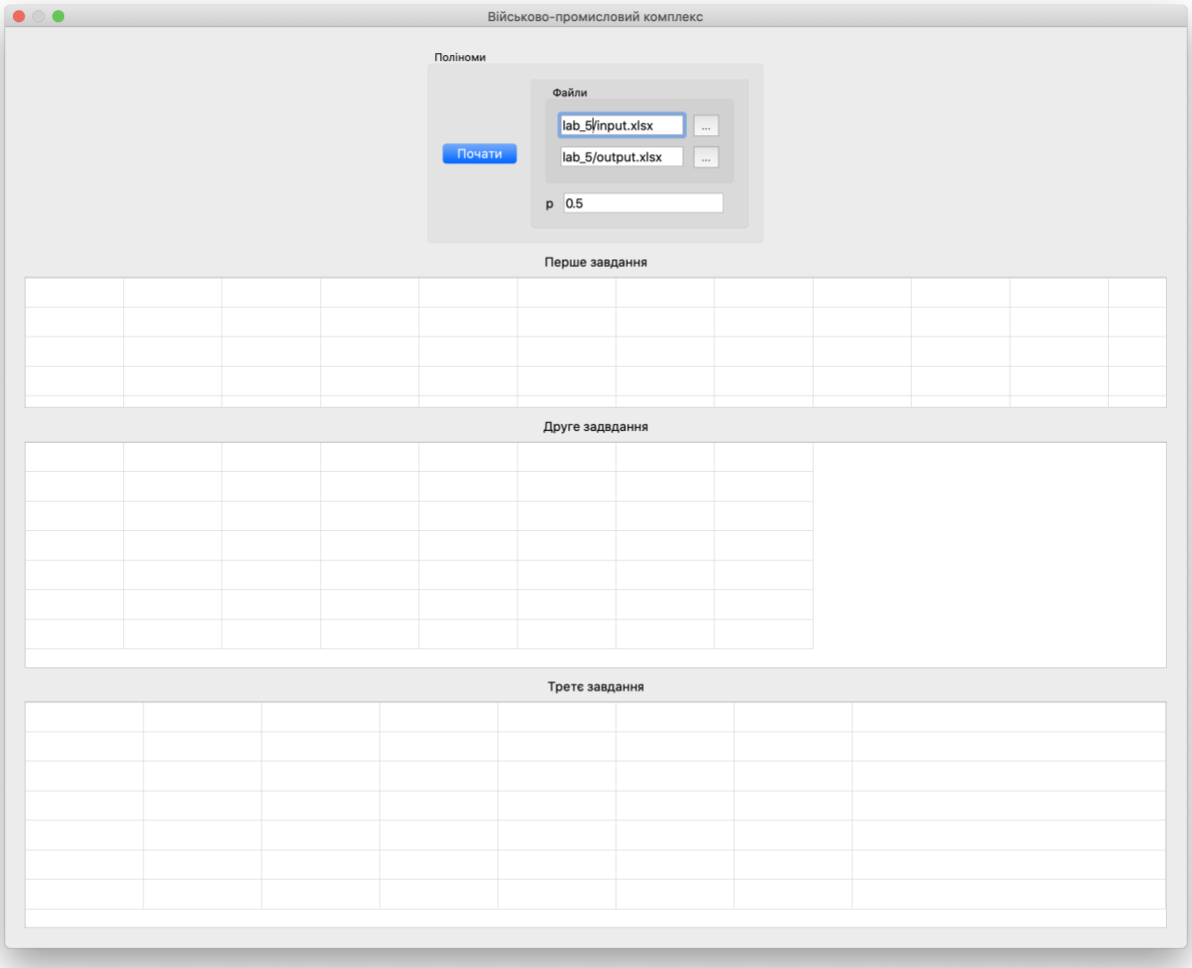
- Знайти розподіл ймовірностей альтернатив параметрів морфологічної таблиці для ситуації в країні і в світі, якщо задані матриця попередніх оцінок альтернатив (табл. 3) і матриця взаємозв’язків (табл. 4).

- Розрахувати незалежні очікувані результативності альтернативи рішень щодо пріоритетності галузей ВПК, виходячи з результатів розв'язку попередньої задачі і матриці зв’язків (табл. 5).

- Розрахувати очікувані результативності альтернатив рішень з урахуванням залежностей між параметрами, заданих матрицею взаємозв’язків (табл. 6), використовуючи в якості незалежних оцінок результати розв'язку попередньої задачі.

Інтерфейс програми

1) Користувач повинен увести файл, з якого буде зчитуватись вхідна інформація (файл з варіантом нашої задачі за замовчанням лежить у lab_6 та називається input.xlsx).
2) Користувач повинен увести файл, куди запишуться результати програми. Також результати можна буде побачити у самих таблицях.
3) Користувач повинен увести значення p (від 0 до 1), яке використовується у подальших формулах.



Завдання 1

Знайти розподіл ймовірностей альтернатив параметрів морфологічної таблиці для ситуації в країні і в світі, якщо задані матриця попередніх оцінок альтернатив (табл. 3) і матриця взаємозв’язків.

Перерахунок будемо виконувати з використання формули повної імовірності, або схеми Байєса:

$$
\begin{cases}
p_1'^{(1)} = \sum\limits_{i_2=1}^{n_2}\sum\limits_{i_3=1}^{n_3}\dots\sum\limits_{i_N=1}^{n_N} P(a_1^{(1)} \mid a_{i_2}^{(2)}, a_{i_3}^{(3)}, \dots, a_{i_N}^{(N)}) \prod\limits_{j=2}^{N} p_{i_j}'^{(j)}, \\
\dots \\
p_{n_1}'^{(1)} = \sum\limits_{i_2=1}^{n_2}\sum\limits_{i_3=1}^{n_3}\dots\sum\limits_{i_N=1}^{n_N} P(a_{n_1}^{(1)} \mid a_{i_2}^{(2)}, a_{i_3}^{(3)}, \dots, a_{i_N}^{(N)}) \prod\limits_{j=2}^{N} p_{i_j}'^{(j)}, \\
\dots \\
p_1'^{(N)} = \sum\limits_{i_1=1}^{n_1}\sum\limits_{i_2=1}^{n_2}\dots\sum\limits_{i_{N-1}=1}^{n_{N-1}} P(a_1^{(N)} \mid a_{i_1}^{(1)}, a_{i_2}^{(2)}, \dots, a_{i_{N-1}}^{(N-1)}) \prod\limits_{j=1}^{N-1} p_{i_j}'^{(j)}, \\
\dots \\
p_{n_N}'^{(N)} = \sum\limits_{i_1=1}^{n_1}\sum\limits_{i_2=1}^{n_2}\dots\sum\limits_{i_{N-1}=1}^{n_{N-1}} P(a_{n_N}^{(N)} \mid a_{i_1}^{(1)}, a_{i_2}^{(2)}, \dots, a_{i_{N-1}}^{(N-1)}) \prod\limits_{j=1}^{N-1} p_{i_j}'^{(j)}, \\
\sum\limits_{i=1}^{n_1} p_i'^{(1)} = 1, \\
\dots \\
\sum\limits_{i=1}^{n_N} p_i'^{(N)} = 1.
\end{cases}
$$

Ми отримали систему нелінійних рінянь (в нашому випадку з 15 невідомими та 19 рівняннями). Невідомими тут є величини $p'^{(K)}_{n_j}$, де K=$\overline{1, N}$, n_j – кількість альтернатив.

Умовні імовірності будемо апроксимувати поліномом 3-го ступеня. Коефіцієнти полінома будемо знаходити виходячи з наступних міркувань:

$$
P(a_j^{(i)} \mid V) = \begin{cases}
0, & (c_{ij,V_1} = -1) \vee \dots \vee (c_{ij,V_{N-1}} = -1), \\
p_j^{(i)}, & (c_{ij,V_1} = 0) \wedge \dots \wedge (c_{ij,V_{N-1}} = 0), \\
1, & (c_{ij,V_1} = 1) \wedge \dots \wedge (c_{ij,V_{N-1}} = 1).
\end{cases}
$$

Де c знаходимо наступним чином:

$$c_i' = 2/(1-c_i) - 1.$$

У випадку більше ніж двох характеристичних параметрів умовна ймовірність залежить від декількох значень з таблиці взаємозв'язків. Вони зводяться до одного за допомогою такої процедури: начення з матриці взаємозв'язків, від яких залежить умовна ймовірність, відобразимо на множину за допомогою перетворення.

Далі обернено відображаємо отримані значення $C = 1 - 2/(C' + 1)$.

Апроксимацію будуємо на основі полінома:

$$
P(a_j^{(i)} \mid a_{j'}^{(i')}) = 3\left(\frac{t(c)+1}{2}\right)^2 - 2\left(\frac{t(c)+1}{2}\right)^3
$$

Де

$$t(c)=\begin{cases}1-2\left(\dfrac{1-c}{2}\right)^{\eta(p)}, \eta(p)\geq 1,\\ 2\left(\dfrac{1+c}{2}\right)^{\frac{1}{\eta(p)}}-1, \eta(p)<1,\end{cases}$$

$$\eta(p)=\begin{cases}-\log_2(r(p)),\ p\geq 0{,}5,\\ -\log_2(1-r(p))^{-1},\ p<0{,}5,\end{cases}$$

$$r(p)=\cos\left(\frac{\arccos(1-2p)+\pi}{3}\right)+\frac{1}{2}.$$

Ці формули підібрані так, щоб виконувалися інтерполяція полінома в точці 0.

Підсталяючи отримані значення умовних імовірностей ми отримали можливість вирішувати системи.

За початкові значення ми взяли імовірності з Таблиці 1. Потім ітераційно підставляючи до системи рівнянь (перших 15), отримували нові значення. Після цього ми їх нормували, щоб виконувалися останні 4 рівняння.

Отже, ми отримали такі нові значення імовірностей з урахуванням взаємозв’язків.

Напруженість в світі		**Напруженість в країні**		**Стан економіки**		**Терористична загроза**	
Відносна стабільність :	0,18075	Затишшя :	0,235385	Оптимістичний:	0,215874	Сильна :	0,409896
Холодна війна :	0,30419	Заморожений конфлікт, збройні провокації :	0,317513	Стабільний :	0,280106	Помірна :	0,350036
Локальні конфлікти :	0,28077	Збройне протистояння :	0,285874	Спад :	0,261273	Слабка :	0,240067
Локальні війни:	0,23429	Повномасштабна агресія іншої держави :	0,161227	Криза/дефолт :	0,242747		

Представимо порівняльний аналіз нових і старих значень імовірностей:

Завдання 2

Розрахувати незалежні очікувані результативності альтернативи рішень щодо пріоритетності галузей ВПК, виходячи з результатів розв'язку попередньої задачі і матриці зв’язків (табл. 5).

Результативність рішень будемо знаходити з наступної формули:

$$E\{a_j^{(N+i)}\}=\sum_k P(a_j^{(N+i)}\mid V^{(k)})P(V^{(k)})$$

Де

$a_j^{(N+i)}$ – j-а альтернатива $(N+i)$-го параметра стратегії;

$V^{(k)} = \{a_{i_{k,1}}^{(1)}; ...; a_{i_{k,N}}^{(N)}\}$ – вектор сценарію, що складається з альтернатив кожного параметра сценарію;

$P(a_j^{(N+i)} | V^{(k)})$ – оцінка альтернативи $a_j^{(N+i)}$ за умов сценарію $V^{(k)}$, яка апроксимується, виходячи зі значень матриці зв'язків для цієї альтернативи параметра стратегії і заданих вектором $V^{(k)}$ альтернатив параметрів сценарію;

$P(V^{(k)})$ – ймовірність сценарію, яка розраховується на основі даних, отриманих на першому етапі морфологічного дослідження.

При підрахунку умовних ймовірностей, ми будемо користуватись тими же методами апроксимації, що і в першому завданні. В точці 0 ми беремо значення результативності за 0.25 з розрахунку, що якщо альтернативи не мають ніякого впливу, то ми можемо взяти їх рівними.

Отже, ми отримали наступну таблицю:

Галузь		Технології		Експорт/ використання	
Розвідка :	0,19609	Приладобудування і електроніка :	0,218744	Переважно внутрішнє використання :	0,552867
Зв'язок і радіоелектронна боротьба :	0,19278	Машинобудування :	0,198778	Переважно експорт :	0,447133
Піхотне озброєння :	0,192105	Новітні матеріали :	0,19774		
Наземна техніка :	0,152471	Технології зв'язку :	0,194837		
Повітряна техніка :	0,148946	Оптика :	0,189901		
Протиповітряна оборона, ракетні війська :	0,117608				

Завдання 3

Розрахувати очікувані результативності альтернатив рішень з урахуванням залежностей між параметрами, заданих матрицею взаємозв'язків (табл. 6), використовуючи в якості незалежних оцінок результати розв'язку попередньої задачі.

Таким самим методом, як і в першому завданні ми розрахуємо результативність, маючи на увазі зв'язки між альтернативами.

Результат:

Галузь		Технології		Експорт/ використання	
Протиповітряна оборона, ракетні війська :	0,178769	Приладобудування і електроніка :	0,209108	Переважно внутрішнє використання:	0,515244
Повітряна техніка :	0,173006	Машинобудування :	0,204337	Переважно експорт:	0,484756
Наземна техніка :	0,171877	Новітні матеріали :	0,201542		
Розвідка :	0,163945	Технології зв'язку :	0,193125		
Зв'язок і радіоелектронна боротьба :	0,161974	Оптика :	0,191888		
Піхотне озброєння :	0,150429				

Висновки:

Приймається рішення щодо пріоритетності галузей ВПК в умовах майбутньої ситуації в країні і в світі. З морфологічного аналізу ми отримали, що найбільш пріорітетними при існуючих ймовірностях впливу зовнішніх факторів та взаємозв'язків є :

Для галузей – Протиповітряна оборона, ракетні війська
Для технологій – Приладобудування і електроніка
Для експорту/використання – Переважно внутрішнє використання

Список літератури:

1. Панкратова Н.Д., Савченко І.О. Застосування методу морфологічного аналізу до задач технологічного передбачення // Наукові праці / Миколаївський держ. гуманітарний ун-т ім. Петра Могили комплексу НаУКМА. Сер. Комп'ютерні технології, системний аналіз, моделювання. — 2008. — 90, вип. 77.

2. Morphological Analysis – A Problem Solving Method By An Astrophysicist Who Discovered Dark Matter https://northstar.greyb.com/morphological-analysis/

3. Ritchey T. Strategic Decision Support using Computerised Morphological Analysis // 9th International Command and Control Research and Technology Symposium, Copenhagen, September 2004. — Copenhagen, 2004.

4. Using Morphological Analysis for Innovation and Resource and Development: An Invaluable Tool for Entrepreneurship https://www.scmspune.ac.in/chapter/pdf/Chapter%203.pdf

5. The Norris Brothers Ltd. morphological approach to engineering design – an early example of applied morphological analysis http://www.amg.swemorph.com/pdf/amg-3-2-2014.pdf

Лістинг:

Main

```
import logging
import os
import sys
from lab_6.solver import Solver
import xlrd
import math
import openpyxl
from copy import deepcopy

from PyQt5.uic import loadUiType

import sys
from PyQt5.QtCore import pyqtSlot, pyqtSignal, QTimer
from PyQt5.QtWidgets import QApplication,QTableWidgetItem, QDialog, QFileDialog, QMessageBox

app = QApplication(sys.argv)
app.setApplicationName('LABA 6')
form_class, base_class = loadUiType('lab_6/main_window_2.ui')

data = {'col1':['1','2','3','4'],
        'col2':['1','2','1','3'],
        'col3':['1','1','2','1']}


class MainWindow_2(QDialog, form_class):
    # signals:
    input_changed = pyqtSignal('QString')
    output_changed = pyqtSignal('QString')


    def __init__(self, parent = None):
        super(MainWindow_2, self).__init__(parent)



        # setting up ui
        self.setupUi(self)



        self.setWindowTitle('Військово-промисловий комплекс')
        self.input_path = self.line_input.text()
        if len(self.input_path) == 0:
            self.input_path = os_path
        self.output_path = './output.xlsx'
```

```
    #set tablewidget
    self.tablewidget.verticalHeader().hide()
    self.tablewidget_first.verticalHeader().hide()
    self.tablewidget_second.verticalHeader().hide()
    self.tablewidget.horizontalHeader().hide()
    self.tablewidget_first.horizontalHeader().hide()
    self.tablewidget_second.horizontalHeader().hide()
    #self.tablewidget.setRowCount(0)
    '''
    column_size = [60, 70, 100, 100,200, 60, 200, 80]
    for index, size in enumerate(column_size):
        self.tablewidget.setColumnWidth(index,size)
    '''
    return

@pyqtSlot()
def input_clicked(self):
    filename = QFileDialog.getOpenFileName(self, 'Відкрити данні', '.', 'Data file (*.xlsx)')[0]
    if filename == '':
        return
    if filename != self.input_path:
        self.input_path = filename
        self.input_changed.emit(filename)
    return

@pyqtSlot('QString')
def input_modified(self, value):
    if value != self.input_path:
        self.input_path = value
    return



@pyqtSlot()
def output_clicked(self):
    filename = QFileDialog.getSaveFileName(self, 'Save data file', '.', 'Spreadsheet (*.xlsx)')[0]
    if filename == '':
        return
    if filename != self.output_path:
        self.output_path = filename
        self.output_changed.emit(filename)
    return

@pyqtSlot('QString')
def output_modified(self, value):
    if value != self.output_path:
        self.output_path = value
```

```
        return


    @pyqtSlot()
    def exec_clicked(self):
        self.exec_button.setEnabled(False)
        try:
            prob = 0.5

            p_in = xlrd.open_workbook(self.input_path)
            p_out = openpyxl.Workbook()

            s = Solver(prob)
            s.load_data(p_in)

            tmp, output1 = s.solve_task1(p_out.create_sheet('Task 1'), self.tablewidget_first)
            tmp, output2 = s.solve_task2(p_out.create_sheet('Task 2'), tmp,self.tablewidget_second)
            tmp, output3 = s.solve_task3(p_out.create_sheet('Task 3'), tmp, self.tablewidget)

            p_out.save(self.output_path)

        except Exception as e:
            QMessageBox.warning(self,'Error!','Error happened during execution: ' + str(e))
        self.exec_button.setEnabled(True)
        return
```

Solver

```
import xlrd
import math
import openpyxl
from copy import deepcopy
import numpy as np
from PyQt5.QtWidgets import QTableWidgetItem


class Solver:
    def __init__(self, p):
        self.AlternativesNames = []
        self.Alternatives = []
        self.OutcomeNames = []
        self.Outcomes = []

        self.Alt_Expert_Prob = []
        self.Alternatives_Dependencies = []
        self.Outcomes_Dependencies = []
        self.Alternatives_Outcomes_Crossdependencies = []
```

```
        self.p = p

    def solve_task1(self, outsheet, table):
        for i in range(len(self.Alternatives)):
            summ = 0
            for j in range(len(self.Alternatives[i])):
                summ = summ + self.Alt_Expert_Prob[i][j]
            for j in range(len(self.Alternatives[i])):
                self.Alt_Expert_Prob[i][j] = self.Alt_Expert_Prob[i][j] / summ
        alt_prob = self.modify_probabilities(self.Alt_Expert_Prob, self.Alternatives_Dependencies)

        row = 0
        column = 0
        for i in range(len(self.Alternatives)):
            outsheet.cell(1, 3*i+1).value = self.AlternativesNames[i]
            for j in range(len(self.Alternatives[i])):
                outsheet.cell(2+j, 3*i+1).value = self.Alternatives[i][j] + ' :'
                outsheet.cell(2+j, 3*i+2).value = str(alt_prob[i][j])
                if 2 + j > row:
                    row = 2 + j
                if 3 * i + 2 > column:
                    column = 3 * i + 2

        result = np.empty((row, column), dtype='object')
        for i in range(len(self.Alternatives)):
            newitem = QTableWidgetItem(str(self.AlternativesNames[i]))
            table.setItem(0, 3*i,newitem)
            result[0, 3*i] = self.AlternativesNames[i]
            for j in range(len(self.Alternatives[i])):
                newitem = QTableWidgetItem(str(self.Alternatives[i][j] + ' :'))
                table.setItem(1+j, 3*i,newitem )
                newitem = QTableWidgetItem(str(alt_prob[i][j]))
                table.setItem(1+j, 3*i+1, newitem)
                result[1+j, 3*i] = self.Alternatives[i][j] + ' :'
                result[1+j, 3*i+1] = str(alt_prob[i][j])
        table.resizeColumnsToContents()
        table.resizeRowsToContents()
        return alt_prob, result

    def solve_task2(self, outsheet, alt_prob, table):
        outcome_prob = self.calculate_outcome_prob(alt_prob)
        outcomes_c = [None for k in range(len(self.Outcomes))]
        for i in range(len(self.Outcomes)):
            outcomes_c[i] = deepcopy(self.Outcomes[i])
            sort = sorted(zip(outcome_prob[i], outcomes_c[i]), reverse=True)
            outcomes_c[i] = [x for y, x in sort]
            outcome_prob[i] = [y for y, x in sort]
```

```
        row = 0
        column = 0
        for i in range(len(self.Outcomes)):
            outsheet.cell(1, 3*i+1).value = self.OutcomeNames[i]
            for j in range(len(self.Outcomes[i])):
                outsheet.cell(2+j, 3*i+1).value = outcomes_c[i][j] + ' :'
                outsheet.cell(2+j, 3*i+2).value = str(outcome_prob[i][j])
                if 2 + j > row:
                    row = 2 + j
                if 3 * i + 2 > column:
                    column = 3 * i + 2

        result = np.empty((row, column), dtype='object')
        for i in range(len(self.Outcomes)):
            newitem = QTableWidgetItem(str(self.OutcomeNames[i]))
            table.setItem(0, 3*i,newitem)
            result[0, 3*i] = self.OutcomeNames[i]
            for j in range(len(self.Outcomes[i])):
                newitem = QTableWidgetItem(str(outcomes_c[i][j] + ' :'))
                table.setItem(1+j, 3*i,newitem )
                newitem = QTableWidgetItem(str(outcome_prob[i][j]))
                table.setItem(1+j, 3*i+1, newitem)
                result[1+j, 3*i] = outcomes_c[i][j] + ' :'
                result[1+j, 3*i+1] = str(outcome_prob[i][j])
        table.resizeColumnsToContents()
        table.resizeRowsToContents()

        return outcome_prob, result

    def solve_task3(self, outsheet, outcome_prob, table):
        outcome_mod_prob = self.modify_probabilities(outcome_prob, self.Outcomes_Dependencies)
        outcomes_c = [None for k in range(len(self.Outcomes))]
        for i in range(len(self.Outcomes)):
            outcomes_c[i] = deepcopy(self.Outcomes[i])
            sort = sorted(zip(outcome_mod_prob[i], outcomes_c[i]), reverse=True)
            outcomes_c[i] = [x for y, x in sort]
            outcome_mod_prob[i] = [y for y, x in sort]

        row = 0
        column = 0
        for i in range(len(self.Outcomes)):
            outsheet.cell(1, 3 * i + 1).value = self.OutcomeNames[i]
            for j in range(len(self.Outcomes[i])):
                outsheet.cell(2 + j, 3 * i + 1).value = outcomes_c[i][j] + ' :'
                outsheet.cell(2 + j, 3 * i + 2).value = str(outcome_mod_prob[i][j])
            outsheet.cell(2, 3*i+1).font = openpyxl.styles.Font(bold=True)
            outsheet.cell(2, 3*i+2).font = openpyxl.styles.Font(bold=True)
            if 2 + j > row:
```

```
            row = 2 + j
        if 3 * i + 2 > column:
            column = 3 * i + 2

    result = np.empty((row, column), dtype='object')
    for i in range(len(self.Outcomes)):
        newitem = QTableWidgetItem(str(self.OutcomeNames[i]))
        table.setItem(0, 3*i,newitem)
        result[0, 3*i] = self.OutcomeNames[i]
        for j in range(len(self.Outcomes[i])):
            newitem = QTableWidgetItem(str(outcomes_c[i][j] + ' :'))
            table.setItem(1+j, 3*i,newitem )
            newitem = QTableWidgetItem(str(outcome_mod_prob[i][j]))
            table.setItem(1+j, 3*i+1, newitem)

            result[1+j, 3*i] = outcomes_c[i][j] + ' :'
            result[1+j, 3*i+1] = str(outcome_mod_prob[i][j])
    table.resizeColumnsToContents()
    table.resizeRowsToContents()



    return outcome_mod_prob, result

def modify_probabilities(self, starting_prob, dep_matrix):
    cur = 0
    l = len(starting_prob)
    res = [None for k in range(2)]
    res[0] = [None for k in range(l)]
    res[1] = [None for k in range(l)]

    for i in range(l):
        res[0][i] = [None for k in range(len(starting_prob[i]))]
        res[1][i] = [None for k in range(len(starting_prob[i]))]

    for i in range(l):
        for j in range(len(starting_prob[i])):
            res[1][i][j] = starting_prob[i][j]

    for iteration in range(500):
        for i in range(l):
            for j in range(len(starting_prob[i])):
                n = [0 for k in range(l)]
                res[cur][i][j] = 0
                while True:
                    # calculating of probabilities
                    cur_dep = [None for k in range(l - 1)]
                    for k in range(l):
                        if k > i:
```

```
                    cur_dep[k - 1] = dep_matrix[i][k][j][n[k]]
                elif k < i:
                    cur_dep[k] = dep_matrix[i][k][j][n[k]]
            p_a = self.calculate_p(cur_dep)

            for k in range(l):
                if k == i:
                    continue
                p_a = p_a * res[1 - cur][k][n[k]]
            res[cur][i][j] = res[cur][i][j] + p_a

            # modification of index array
            cn = l - 1
            while cn >= 0:
                n[cn] = n[cn] + 1
                if n[cn] == len(starting_prob[cn]):
                    n[cn] = 0
                    cn = cn - 1
                else:
                    break
            if cn < 0:
                break

        # norming the probabilities
        summ = 0
        for j in range(len(starting_prob[i])):
            summ = summ + res[cur][i][j]
        for j in range(len(starting_prob[i])):
            res[cur][i][j] = res[cur][i][j] / summ

    cur = 1 - cur
  return res[cur]

def calculate_outcome_prob(self, alt_prob):
  res = [None for k in range(len(self.Outcomes))]
  for i in range(len(res)):
    res[i] = [0 for k in range(len(self.Outcomes[i]))]
  for i in range(len(self.Outcomes)):
    for j in range(len(self.Outcomes[i])):
      n = [None for k in range(len(self.Alternatives))]
      for k in range(len(self.Alternatives)):
        n[k] = 0
      while True:
        # calculating of probabilities
        cur_dep = [None for k in range(len(self.Alternatives))]
        for k in range(len(self.Alternatives)):
          cur_dep[k] = self.Alternatives_Outcomes_Crossdependencies[k][i][n[k]][j]
        p_a = self.calculate_p(cur_dep)
```

```
                for k in range(len(self.Alternatives)):
                    if k == i:
                        continue
                    p_a = p_a * alt_prob[k][n[k]]
                res[i][j] = res[i][j] + p_a

                #modification of index array
                cn = len(self.Outcomes) - 1
                while cn >= 0:
                    n[cn] = n[cn] + 1
                    if n[cn] == len(self.Alternatives[cn]):
                        n[cn] = 0
                        cn = cn - 1
                    else:
                        break
                if cn < 0:
                    break
            summ = 0
            for j in range(len(self.Outcomes[i])):
                summ = summ + res[i][j]
            for j in range(len(self.Outcomes[i])):
                res[i][j] = res[i][j] / summ
        return res

    def calculate_p(self, a_i):
        c = 1
        for i in range(len(a_i)):
            c = c * (2 / (1 - a_i[i]) - 1)
        c = 1 - 2 / (c + 1)

        rp = math.cos((math.acos(1 - 2 * self.p) + math.pi) / 3) + 0.5
        if rp <= 0.5:
            np = -1 * math.log(rp, 2)
        else:
            np = -1 / math.log(1 - rp, 2)
        if np >= 1:
            tc = 1 - 2 * math.pow((1 - c) / 2, np)
        else:
            tc = 2 * math.pow((1 + c) / 2, 1 / np) - 1
        return 3 * ((tc + 1) / 2) * ((tc + 1) / 2) - 2 * ((tc + 1) / 2) * ((tc + 1) / 2) * ((tc + 1) / 2)

    def load_data(self, file):
        sheet = file.sheet_by_index(0)

        # inputting alternatives
        alt_count, out_count = None, None
        for i in range(1000):
            if str(sheet.row_values(0)[i]) == '*':
```

```
        alt_count = i
        break
max_row = 0

for i in range(alt_count):
    self.AlternativesNames.append(sheet.row_values(0)[i])
    max_ind = None
    for j in range(1, 1000):
        if sheet.row_values(j+1)[i] == '*':
            max_ind = j
            break
    self.Alternatives.append([])
    self.Alt_Expert_Prob.append([])
    for j in range(max_ind):
        self.Alternatives[i].append(sheet.row_values(j+1)[i])
        self.Alt_Expert_Prob[i].append(None)
    if max_row < max_ind:
        max_row = max_ind

# inputting outcomes
base_row = max_row + 2
for i in range(1000):
    if sheet.row_values(base_row)[i] == '*':
        out_count = i
        break
max_row = 0
for i in range(out_count):
    self.OutcomeNames.append(sheet.row_values(base_row)[i])
    max_ind = None
    for j in range(1000):
        if sheet.row_values(j + 1 + base_row)[i] == '*':
            max_ind = j
            break
    self.Outcomes.append([])
    for j in range(max_ind):
        self.Outcomes[i].append(sheet.row_values(j + 1 + base_row)[i])
    if max_row < max_ind:
        max_row = max_ind
base_row = base_row + max_row + 3

# inputting alternative probabilities
max_row = 0
for i in range(alt_count):
    for j in range(len(self.Alt_Expert_Prob[i])):
        self.Alt_Expert_Prob[i][j] = float(sheet.row_values(base_row + j)[i])
base_row = 1

# inputting alt-alt dependency matrix
```

```
        sheet = file.sheet_by_index(1)
        self.Alternatives_Dependencies = [[None for h in range(alt_count)] for k in range(alt_count)]
        for i in range(alt_count - 1):
            base_column = 1
            for j in range(1, alt_count):
                if j <= i:
                    base_column = base_column + len(self.Alternatives[j])
                    continue
                li = len(self.Alternatives[i])
                lj = len(self.Alternatives[j])
                self.Alternatives_Dependencies[i][j] = [[None for h in range(lj)] for k in range(li)]
                self.Alternatives_Dependencies[j][i] = [[None for h in range(li)] for k in range(lj)]
                for i1 in range(li):
                    for j1 in range(lj):
                        if sheet.row_values(base_row + i1)[base_column + j1] == '*':
                            self.Alternatives_Dependencies[i][j][i1][j1] = 0
                            self.Alternatives_Dependencies[j][i][j1][i1] = 0
                        else:
                            self.Alternatives_Dependencies[i][j][i1][j1] = float(
                                sheet.row_values(base_row + i1)[base_column + j1])
                            self.Alternatives_Dependencies[j][i][j1][i1] = float(
                                sheet.row_values(base_row + i1)[base_column + j1])
                base_column = base_column + len(self.Alternatives[j])
            base_row = base_row + len(self.Alternatives[i])
        base_row = base_row + 2

        # inputting alt_out dependency matrix
        self.Alternatives_Outcomes_Crossdependencies = [[None for h in range(out_count)] for k in
range(alt_count)]
        for i in range(alt_count):
            base_column = 1
            for j in range(out_count):
                li = len(self.Alternatives[i])
                lj = len(self.Outcomes[j])
                self.Alternatives_Outcomes_Crossdependencies[i][j] = [[None for h in range(lj)] for k in
range(li)]
                for i1 in range(li):
                    for j1 in range(lj):
                        if sheet.row_values(base_row + i1)[base_column + j1] == '*':
                            self.Alternatives_Outcomes_Crossdependencies[i][j][i1][j1] = 0
                        else:
                            self.Alternatives_Outcomes_Crossdependencies[i][j][i1][j1] = float(
                                sheet.row_values(base_row + i1)[base_column + j1])
                base_column = base_column + len(self.Outcomes[j])
            base_row = base_row + len(self.Alternatives[i])
        base_row = base_row + 2

        # inputting out-out dependency matrix
```

```
self.Outcomes_Dependencies = [[None for h in range(out_count)] for k in range(out_count)]
for i in range(out_count):
    base_column = 1
    for j in range(1, out_count):
        if j <= i:
            base_column = base_column + len(self.Outcomes[j])
            continue
        li = len(self.Outcomes[i])
        lj = len(self.Outcomes[j])
        self.Outcomes_Dependencies[i][j] = [[None for h in range(lj)] for k in range(li)]
        self.Outcomes_Dependencies[j][i] = [[None for h in range(li)] for k in range(lj)]
        for i1 in range(li):
            for j1 in range(lj):
                if sheet.row_values(base_row + i1)[base_column + j1] == '*':
                    self.Outcomes_Dependencies[i][j][i1][j1] = 0
                    self.Outcomes_Dependencies[j][i][j1][i1] = 0
                else:
                    self.Outcomes_Dependencies[i][j][i1][j1] = float(
                        sheet.row_values(base_row + i1)[base_column + j1])
                    self.Outcomes_Dependencies[j][i][j1][i1] = float(
                        sheet.row_values(base_row + i1)[base_column + j1])
        base_column = base_column + len(self.Outcomes[j])
    base_row = base_row + len(self.Outcomes[i])
```