# Лабораторная работа №4

Метод анализа иерархий

## Цель работы

Углубление теоретических знаний в области системного анализа, приобретение навыков создания и описания иерархических структур, а также изучение понятий цель, критерий, альтернатива.

## Постановка задачи

Приобрести навыки сравнения и выполнения количественной оценки альтернативных вариантов решения с помощью Метода анализа иерархий (МАИ).

## Вариант задания

(Вариант 1) Предприятие для проектируемых изделий должно выбирать операционную систему по заданным критериям указанных в баллах.

Таблица 4.1 – Критерии альтернатив для выбора операционной системы

|  |  |  |  |
| --- | --- | --- | --- |
| **Критерии** | **Windows 7** | **Windows 8** | **Windows 10** |
| **Стоимость, балл** | 5 | 7 | 5 |
| **Визуальный интерфейс** | 7 | 8 | 6 |
| **Надёжность** | 5 | 5 | 9 |
| **Разрядность кода ОС** | 16 | 32 | 64 |

## Ход выполнения

### Проведение вычислений

Была составлена матрица оценок приоритетов характеристик (Таблица 4.1).

Таблица 4.2 – Матрица попарных сравнений на основе экспертных оценок

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Стоимость, балл** | **Визуальный интерфейс** | **Надёжность** | **Разрядность кода ОС** |
| **Стоимость, балл** | 1 | 3 | 1/5 | 1/2 |
| **Визуальный интерфейс** | 1/3 | 1 | 1/7 | 1/7 |
| **Надёжность** | 5 | 7 | 1 | 2 |
| **Разрядность кода ОС** | 2 | 7 | 1/2 | 1 |

На основе данных из Таблица 4.1 были составлены вектора приоритетов для 2 уровня.

Таблица 4.3 – Вектор приоритетов для уровня 2

|  |  |
| --- | --- |
| **Общее удовлетворение операционной системой** | **Вектор приоритетов** |
| **Стоимость, балл** | 0,133 |
| **Визуальный интерфейс** | 0,052 |
| **Надёжность** | 0,522 |
| **Разрядность кода ОС** | 0,293 |

На основе объективных показателей из Таблица 4.1 были составлены матрицы для 3-го уровня.

Таблица 4.4 – Сравнение вариантов с точки зрения стоимости

|  |  |  |  |
| --- | --- | --- | --- |
| **Стоимость** | **Windows 7** | **Windows 8** | **Windows 10** |
| **Windows 7** | 1 | 5/7 | 1 |
| **Windows 8** | 7/5 | 1 | 7/5 |
| **Windows 10** | 1 | 5/7 | 1 |

Таблица 4.5 – Сравнение вариантов с точки зрения Визуального интерфейса

|  |  |  |  |
| --- | --- | --- | --- |
| **Визуальный интерфейс** | **Windows 7** | **Windows 8** | **Windows 10** |
| **Windows 7** | 1 | 1 | 7/6 |
| **Windows 8** | 1 | 1 | 8/6 |
| **Windows 10** | 6/7 | 6/8 | 1 |

Таблица 4.6 – Сравнение вариантов с точки зрения Надёжности

|  |  |  |  |
| --- | --- | --- | --- |
| **Надёжность** | **Windows 7** | **Windows 8** | **Windows 10** |
| **Windows 7** | 1 | 1 | 5/9 |
| **Windows 8** | 1 | 1 | 5/9 |
| **Windows 10** | 9/5 | 9/5 | 1 |

Таблица 4.7 – Сравнение вариантов с точки зрения Разрядности кода ОС

|  |  |  |  |
| --- | --- | --- | --- |
| **Разрядность кода ОС** | **Windows 7** | **Windows 8** | **Windows 10** |
| **Windows 7** | 1 | 16/32 | 16/64 |
| **Windows 8** | 32/16 | 1 | 32/64 |
| **Windows 10** | 64/16 | 64/32 | 1 |

На основе Таблиц 4.4 – 4.7 были составлены приоритеты 3-го уровня (на примере данных из Таблица 4.4).

Для остальных таблиц получены следующие значения:

После чего были рассчитаны вектора приоритетов для уровня 3 (Таблица 4.8).

Таблица 4.8 – Вектора приоритетов для уровня 3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Вариант решения** | **Стоимость** | **Визуальный интерфейс** | **Надёжность** | **Разрядность кода ОС** |
| **Windows 7** | 0,294 | 0,349 | 0,263 | 0,143 |
| **Windows 8** | 0,412 | 0,365 | 0,263 | 0,256 |
| **Windows 10** | 0,294 | 0,286 | 0,474 | 0,571 |

На основе данных из Таблица 4.3 и Таблица 4.8 были вычислены глобальные приоритеты.

На основе глобальных приоритетов 3 Вариант (Windows 10) получает наивысший вес, а значит является наиболее оптимальным.

### Составление программы

#### Листинг программы

Программа составлена для интерпретатора Python 3.10.0.

#####!/bin/python3

from typing import Iterable

import math

def print\_matrix (a):

for i in range (0, len(a)):

for j in range (0, len(a[i])):

print ("%6.4f" % a[i][j], end=" |")

print ()

def print\_array (a: Iterable) -> None:

for row in range (0, len (a)):

print ("%2d -> %2.4f" % (row + 1, a[row]))

def prod (a: Iterable) -> float:

if len(a) > 0:

return a[-1] \* prod (a[:-1])

else:

return 1.0

def main ():

# Получение минимальных данных

crit\_cnt = int ( input ("Введите количество критериев: ") )

altern\_cnt = int ( input ("Введите кольчество альтернатив: ") )

# Составление необходимых матриц

altern\_matrix = [[0 for i in range (0, altern\_cnt)] for j in range (0, crit\_cnt)]

expert\_score\_matrix = [[0 for i in range (0, crit\_cnt)] for j in range (0, crit\_cnt)]

priorities\_2\_level\_matrix = []

# Заполнение матриц

print ("Ввод значений для матрицы критериев альтернатив: ")

for col in range (0, altern\_cnt):

for row in range (0, crit\_cnt):

altern\_matrix[row][col] = float ( input (" \* Матрица [%2d][%2d]: " % (row + 1, col + 1)) )

print ("Заполнение матрицы альтернатив:")

for col in range (0, crit\_cnt - 1):

for row in range (col + 1, crit\_cnt):

expert\_score\_matrix[row][col] = float ( input (" \* Матрица [%2d][%2d]: " % (row + 1, col + 1)) )

expert\_score\_matrix[col][row] = 1 / expert\_score\_matrix[row][col]

for i in range (0, crit\_cnt):

expert\_score\_matrix[i][i] = 1.0;

# Вывод только что введённых матриц

print ("Введены матрицы:")

print (" \* Матрица альтернатив:")

print\_matrix (altern\_matrix)

print (" \* Матрица оценок:")

print\_matrix (expert\_score\_matrix)

# Определение вектора приоритетов для 2 уровня

priorities\_2\_level\_matrix = [

(

(

math.pow (

prod (expert\_score\_matrix[row]),

1/crit\_cnt

)

) / sum (

[

math.pow (

prod (expert\_score\_matrix[i]),

1/crit\_cnt

) for i in range (0, crit\_cnt)

]

)

) for row in range (0, crit\_cnt)

]

print ()

print ("Вектора приоритетов для второго уровня:")

print\_array (priorities\_2\_level\_matrix)

# Определение вектора приоритетов для 3 уровня

priorities\_3\_level\_matrix = [[0 for j in range (0, crit\_cnt)] for i in range (0, altern\_cnt)]

for table\_num in range (0, crit\_cnt):

tmp\_table = [[0.0 for j in range (0, altern\_cnt)] for i in range (0, altern\_cnt)]

for i in range (0, altern\_cnt):

for j in range (0, altern\_cnt):

tmp\_table[j][i] = altern\_matrix[table\_num][j] / altern\_matrix[table\_num][i]

abcde = [

(

(

math.pow (

prod (tmp\_table[row]),

1/altern\_cnt

)

) / sum (

[

math.pow (

prod (tmp\_table[abcd]),

1/altern\_cnt

) for abcd in range (0, altern\_cnt)

]

)

) for row in range (0, altern\_cnt)

]

for i in range (0, len(abcde)):

priorities\_3\_level\_matrix[i][table\_num] = abcde[i]

print ()

print ("Вектора приоритетов для третьего уровня:")

print\_matrix (priorities\_3\_level\_matrix)

# Вычисление глобальных приоритетов

K = [0.0 for i in range (0, altern\_cnt)]

for alternative\_num in range (0, altern\_cnt):

for criteria\_num in range (0, crit\_cnt):

K[alternative\_num] += priorities\_2\_level\_matrix[criteria\_num] \* priorities\_3\_level\_matrix[alternative\_num][criteria\_num]

print ()

print ("Глобальные приоритеты:")

print\_array (K)

print ("Ответ: %d вариант наиболее оптимален." % (K.index(max(K)) + 1))

main ()

#### Тестовый пример

Рисунок 4.1 демонстрирует выполнение программы. Исходные данные для программы – данные, заданные по варианту.



Рисунок 4.1 – Выполнение программы

# Вывод

При выполнении данной лабораторной работы были получены навыки сравнения и выполнения количественной оценки альтернативных вариантов решения с помощью Метода анализа иерархий. Полученные во время выполнения лабораторной работы навыки помогут в дальнейшей жизни при необходимости выбора наилучшей альтернативы из предложенных.