**ДНІПРОВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ОЛЕСЯ ГОНЧАРА**

Факультет прикладної математики

Кафедра обчислювальної математики та математичної кібернетики

ЗВІТ

про виконання лабораторної роботи №1

з дисципліни «Системи і методи прийняття рішень»

Виконав:

студент групи ПА-18-1

Щербак Роман

Викладач:

канд. фіз.-мат. наук, доцент

Н. Л. Козакова

Дніпро

2022

1. **Постановка задачі**
2. Опрацювати необхідний теоретичний матеріал.
3. Сформулювати задачу, що моделюється на основі першої інформаційної ситуації (). При цьому необхідно визначити:
   1. Множину станів середовища;
   2. Множину можливих рішень;
   3. Функціонал, що оцінює якість рішення в певній ситуації;
   4. Апріорний розподіл імовірности на множині станів середовища.
4. Знайти раціональне рішення, використовуючи критерії, відповідні інформаційній ситуації .
5. Порівняти результати, отримані за допомогою різних критеріїв. Результати оформити у вигляді таблиці.
6. Оформити звіт про виконання роботи.
7. **Опис розв’язку**

Перевірку роботи програми здійснено для таких вхідних даних:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| p |  |  |  |  |  |
| 0.1 |  | 1 | 5 | 3 | 0 |
| 0.3 |  | 2 | 6 | 2 | 4 |
| 0.5 |  | 4 | 2 | 2 | 6 |
| 0.1 |  | 0 | 4 | 3 | 1 |

Задача:

Нехай наявно 4 різні роботи, які необхідно виконати.

Є 4 альтернативні варіанти виконання першої роботи:

* Надати найбільший пріоритет виконанню роботи 1 ();
* Надати найбільший пріоритет виконанню роботи 2 ();
* Надати найбільший пріоритет виконанню роботи 3 ();
* Надати найбільший пріоритет виконанню роботи 4 ().

Можливі ситуації виконання робіт:

* – найважливішою є робота 2, роботи 1 та 4 мають невелику важливість;
* – найважливішою є робота 2, робота 4 досить важлива, роботи 1 та 3 мають однакову важливість;
* – найважливішою є робота 4, робота 1 досить важлива, роботи 2 та 3 мають однакову важливість;

     – найважливішою є робота 2, робота 3 досить важлива, роботи 2 та 4 мають невелику важливість.

1. **Код програми мовою Python**

* Модуль main.py

import criterions as c

import additional\_criterions as ac

import get\_data as gd

import prettytable as pt

criterions\_res = pt.PrettyTable()

criterions\_res.field\_names = ["Критерій", "φ\_1", "φ\_2", "φ\_3", "φ\_4", "Оптимальне рішення"]

probabilities, efficacy = gd.get\_data\_from\_file("variant.txt")

print("Вхідні дані:")

print(probabilities)

print(efficacy)

print()

Bayes\_order, eff\_square, row = c.criteria\_calculation(probabilities, efficacy, "Bayes", [])

criterions\_res.add\_row(row)

Bayes\_values = list(Bayes\_order.values())

min\_disp\_order, row = c.criteria\_calculation(probabilities, efficacy, "min\_disp", Bayes\_values)

criterions\_res.add\_row(row)

dispersion = list(min\_disp\_order.values())

Bayes\_average = (1/len(efficacy[0])) \* sum(Bayes\_values)

row = c.criteria\_calculation(probabilities, efficacy, "min\_disp\_mod", Bayes\_average)

criterions\_res.add\_row(row)

Bayes\_max = max(Bayes\_values)

row = c.criteria\_calculation(probabilities, efficacy, "min\_disp\_mod", Bayes\_max)

criterions\_res.add\_row(row)

row = c.criteria\_calculation(probabilities, efficacy, "max\_prob\_dist", [])

criterions\_res.add\_row(row)

row = ac.modal(probabilities, efficacy)

criterions\_res.add\_row(row)

row = ac.minimal\_entropy(probabilities, efficacy)

criterions\_res.add\_row(row)

row\_1, row\_2, row\_3 = ac.combined(Bayes\_values, eff\_square, dispersion)

criterions\_res.add\_row(row\_1)

criterions\_res.add\_row(row\_2)

criterions\_res.add\_row(row\_3)

print(criterions\_res)

* Модуль criterions.py

import random as r

def check\_for\_equals(arr):

keys\_equal = []

i = 0

while i < 1:

j = i+1

while j < len(arr):

if arr[i][1] == arr[j][1]:

if arr[i][0] not in keys\_equal:

keys\_equal.append(arr[i][0])

if arr[j][0] not in keys\_equal:

keys\_equal.append(arr[j][0])

j += 1

i += 1

return keys\_equal

def criteria\_calculation(p\_arr, eff\_arr, criterion, Bayes\_val):

i = 0

row = []

eff\_square = []

res\_dict = {}

if criterion == "max\_prob\_dist":

min\_arr = []

max\_arr = []

for e in eff\_arr:

min\_arr.append(min(e))

max\_arr.append(max(e))

a\_1 = min(min\_arr)

a\_2 = max(max\_arr)

a = r.randint(a\_1, a\_2)

a = 5

while i < len(eff\_arr[0]):

res = 0

res\_1 = 0

j = 0

while j < len(eff\_arr):

if criterion == "Bayes":

if len(row) == 0:

row.append("Баєс")

arr = Bayes(p\_arr[j], eff\_arr[j][i])

res += arr[0]

res\_1 += arr[1]

elif criterion == "min\_disp":

if len(row) == 0:

row.append("Мінімуму дисперсії")

res += dispersion\_minimum(p\_arr[j], eff\_arr[j][i], Bayes\_val[i])

elif criterion == "min\_disp\_mod":

if len(row) == 0:

row.append("Мінімуму дисперсії модифікація")

res += dispersion\_minimum(p\_arr[j], eff\_arr[j][i], Bayes\_val)

elif criterion == "max\_prob\_dist":

if len(row) == 0:

row.append("Максимізації ймовірности")

res += max\_prob\_dist(p\_arr[j], eff\_arr[j][i], a)

j += 1

row.append(res)

res\_dict[f"φ\_{i+1}"] = res

if criterion == "Bayes":

eff\_square.append(res\_1)

i += 1

if criterion == "Bayes" or criterion == "max\_prob\_dist":

sorted\_dict = sorted(res\_dict.items(), key=lambda x: x[1], reverse = True)

elif criterion == "min\_disp" or criterion == "min\_disp\_mod":

sorted\_dict = sorted(res\_dict.items(), key=lambda x: x[1])

best\_solution = check\_for\_equals(sorted\_dict)

if len(best\_solution) == 0:

best\_solution = sorted\_dict[0][0]

row.append(best\_solution)

if criterion == "Bayes":

return res\_dict, eff\_square, row

elif criterion == "min\_disp":

return res\_dict, row

else:

return row

def Bayes(p, eff):

res = p \* eff

res\_1 = p \* (eff\*\*2)

return res, res\_1

def dispersion\_minimum(p, eff, disp):

res = (eff-disp)\*\*2 \* p

return res

def max\_prob\_dist(p, eff, a):

if eff >= a:

return p

else:

return 0

* Модуль additional\_criterions.py

import math as m

import criterions as c

import random as r

def modal(p\_arr, eff\_arr):

row = []

row.append("Модальний")

theta\_max = max(p\_arr)

ind = p\_arr.index(theta\_max)

best\_solution = f"φ\_{eff\_arr[ind].index(max(eff\_arr[ind]))+1}"

for i in eff\_arr[ind]:

row.append(i)

row.append(best\_solution)

return row

def minimal\_entropy(p\_arr, eff\_arr):

row = []

row.append("Мінімальної ентропії")

sums = []

res\_dict = {}

i = 0

while i < len(eff\_arr[0]):

res\_temp = 0

j = 0

while j < len(eff\_arr):

res\_temp += p\_arr[j] \* eff\_arr[j][i]

j += 1

sums.append(res\_temp)

i += 1

i = 0

while i < len(eff\_arr[0]):

res = 0

j = 0

while j < len(eff\_arr):

temp\_var = (p\_arr[j] \* eff\_arr[j][i]) / sums[i]

try:

res += temp\_var \* m.log(temp\_var)

print(temp\_var \* m.log(temp\_var))

except ValueError:

res += 0

j += 1

res = -res

row.append(res)

res\_dict[f"φ\_{i+1}"] = res

print()

i += 1

sorted\_dict = sorted(res\_dict.items(), key=lambda x: x[1])

best\_solution = c.check\_for\_equals(sorted\_dict)

if len(best\_solution) == 0:

best\_solution = sorted\_dict[0][0]

row.append(best\_solution)

return row

def combined(Bayes\_val, eff\_square, disp):

lambda\_arr = []

i = 0

while i < len(Bayes\_val):

Bayes\_val[i] = Bayes\_val[i]\*\*2

lambda\_arr.append(Bayes\_val[i]/eff\_square[i])

i += 1

lambda\_min = min(lambda\_arr)

lambda\_max = max(lambda\_arr)

row\_1 = combined\_calculate(Bayes\_val, disp, lambda\_min, lambda\_max, 0)

row\_2 = combined\_calculate(Bayes\_val, disp, lambda\_min, lambda\_max, 1)

row\_3 = combined\_calculate(Bayes\_val, disp, lambda\_min, lambda\_max, 2)

return row\_1, row\_2, row\_3

def combined\_calculate(Bayes, d, lambda\_min, lambda\_max, num):

row = []

res\_dict = {}

if num == 0:

lambda\_used = 0 + (r.random() \* (lambda\_min - 0))

elif num == 1:

lambda\_used = lambda\_min + (r.random() \* (lambda\_max - lambda\_min))

elif num == 2:

lambda\_used = lambda\_max + (r.random() \* (1- lambda\_max))

row.append(f"Комбінований за λ = {lambda\_used}")

i = 0

while i < len(Bayes):

res = Bayes[i] \* (1-lambda\_used) - lambda\_used \* d[i]

res\_dict[f"φ\_{i+1}"] = res

row.append(res)

i += 1

sorted\_dict = sorted(res\_dict.items(), key=lambda x: x[1], reverse = True)

best\_solution = c.check\_for\_equals(sorted\_dict)

if len(best\_solution) == 0:

best\_solution = sorted\_dict[0][0]

row.append(best\_solution)

return row

* Модуль get\_data.py

def get\_data\_from\_file(file\_name):

p = []

eff = []

f = open(str(file\_name))

for s in f:

s = s.strip().split(" ")

arr = []

i = 0

while i < len(s):

if i == 0:

p.append(float(s[i]))

else:

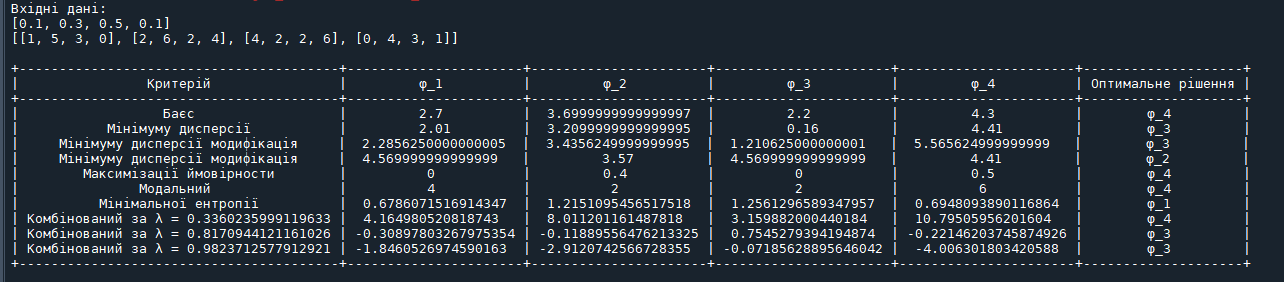
arr.append(int(s[i]))

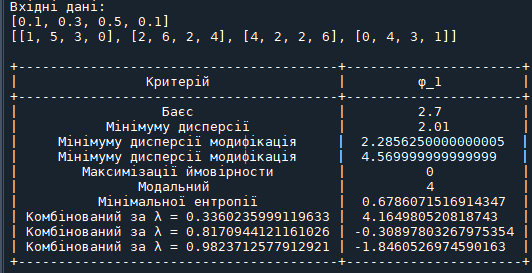
i += 1

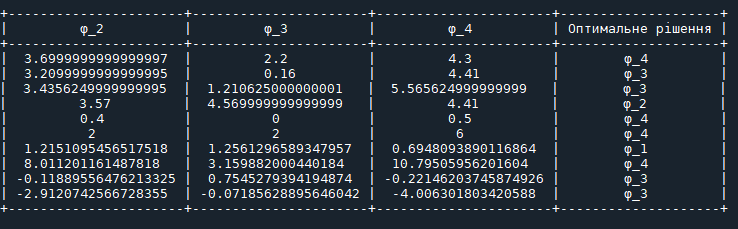
eff.append(arr)

return p, eff

1. **Скриншоти роботи програми**

****

****

****

1. **Висновок**

За більшістю критеріїв найкращими альтернативами є (4 критерії: Баєса, максимізації ймовірности розподілу оцінного функціоналу, модальний та комбінований за = 0.336) та (4 критерії: мінімуму дисперсії оцінного функціоналу, мінімуму дисперсії оцінного функціоналу модифікації 1, комбінований за = 0.817, комбінований за = 0.982). Альтернатива була найкращою за критерієм мінімальної ентропії математичного сподівання оцінного функціоналу. Альтернатива була найкращою за критерієм мінімуму дисперсії оцінного функціоналу модифікації 2.