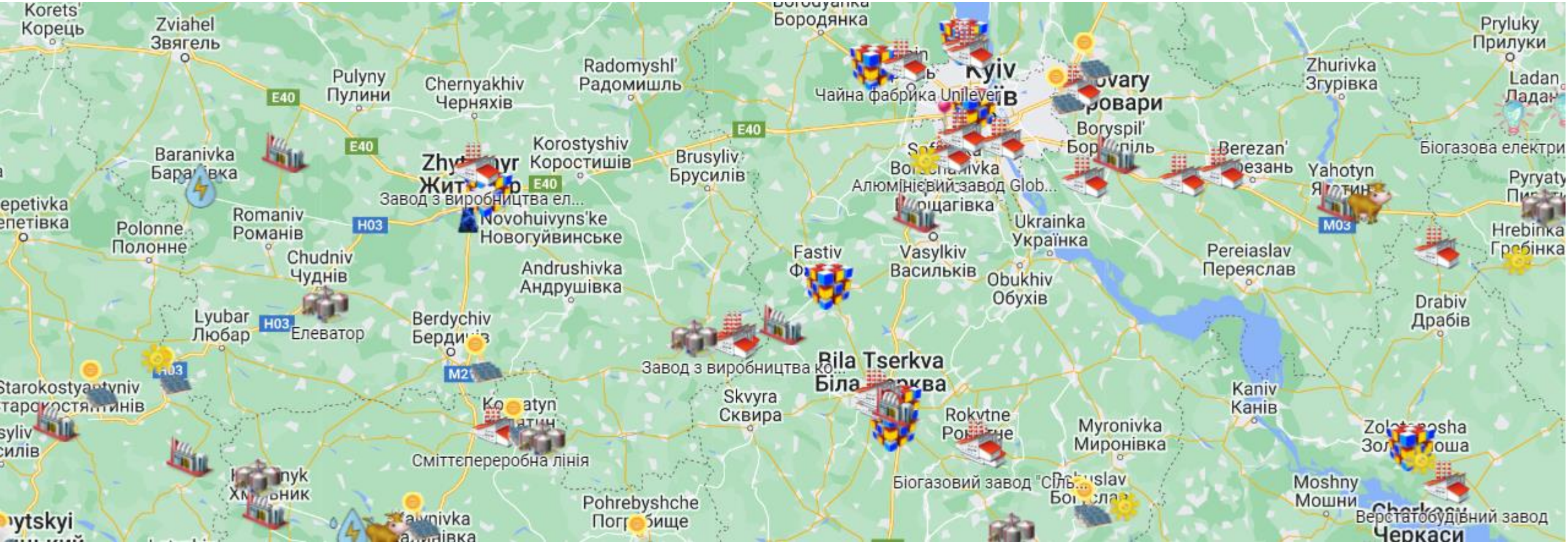
Математичні моделі та методи оцінки ризику на екологічно небезпечних виробництвах

Виконала студентка 3-го курсу Прикладної математики Кржешевська Анастасія

В Україні наразі фіксується складна ситуація екологічної безпеки



Фрагмент інтерактивної карти підприємств, відкритих в Україні з початку 2015 року

Технічний прогрес та все ширше використання інноваційних технологій наштовхнули людство на перехід моделі відповіді на ризик надзвичайної ситуації:

«реагувати та виправляти» → **«передбачати та упереджувати»**

Основною проблемою сучасних виробництв є викиди парникових газів в атмосферу планети

Задля урегулювання даного питання у 1997 році було висунуто Кіотський протокол.

На сучасному етапі економічного розвитку антропогенний вплив на навколишнє середовище загострив конфлікт інтересів між економічною та екологічною системами. Кіотський протокол він зобов’язує брати до уваги «витрати на навколишнє середовище», що впливає на економіку вцілому та, власне, на математично-теоретичну її частину.

Розглянемо модифіковану модель Леонтьєва-Форда, що передбачає оплату забруднень

$$x = Ax + y \quad \rightarrow \quad \begin{cases} x_1 = A_{11}x_1 + A_{12}x_2 + y_1, & x_1 \geq 0 \\ x_2 = A_{21}x_1 + A_{22}x_2 - y_2, & x_2 \geq 0 \end{cases} \quad \rightarrow \quad ?$$

Загальний вигляд моделі Леонтьєва

Загальний вигляд моделі Леонтьєва-Форда

Описує взаємозв'язки між різними секторами економіки, а також потоки ресурсів та продуктів між ними

Використовується для аналізу взаємозв'язків між окремими секторами економіки та екологічними аспектами виробництва; враховуються не лише виробничі відносини, а й використання природних ресурсів та вплив на навколишнє середовище; оцінка екологічних ризиків, пов'язаних з виробництвом та виявлення найбільш уразливих секторів економіки, з погляду їхнього впливу на навколишнє середовище

Модифікована модель Леонтьєва-Форда

Введемо наступні позначення:

$x^1_{i\varphi_i}$ – обсяг виробництва продукції *i* методом φ_i

$x^2_{j\psi_j}$ – обсяг знищення забруднювача *j* методом ψ_j

$a^{11}_{ik\varphi_k}$ – нормативний коефіцієнт витрат продукції *i* на виробництво одиниці продукції $k \in \boldsymbol{I}$ способом φ_k

$a^{12}_{ij\psi_j}$- нормативний коефіцієнт витрат продукції *i* на знищення одиниці забруднювача *j* способом ψ_j

$a^{21}_{ji\varphi_i}$ – нормативний коефіцієнт викиду забруднювача *j* при випуску одиниці продукції *i* способом ψ_j

$a^{22}_{jl\psi_l}$ – нормативний коефіцієнт викиду забруднювача *j* при знищенні одиниці забруднювача $l \in \boldsymbol{J}$ способом ψ_l

y^1_i – обсяг споживання продукції *i*

y^2_j – обсяг максимальної кількості незнищенного забруднювача *j* (викиду)

p_{φ_i} – ймовірність екологічної аварії при виробленні продукції *i* методом φ_i

p_{ψ_j} – ймовірність екологічної аварії при знищенні забруднювача *j* методом ψ_j

$b^1_{ji\varphi_i}$ – коефіцієнт викиду забруднювача *j*, при виробництва продукції *i* методом φ_i, у випадку аварії

$b^2_{jl\psi_l}$ – коефіцієнт викиду забруднювача *j*, при знищенні забруднювача l способом ψ_l, у випадку аварії

c_j – оплата незнищенних викидів забруднення, що береться за кожну одиницю забруднювача *j* («права на забруднення»)

$c_{j\psi_j}$ – вартість знищення одиниці забруднювача l способом ψ_l

Перепишемо деякі значення спираючись на ймовірність того, що екологічна аварія не відбудеться:

$$\widetilde{a^{11}_{ik\varphi_k}} = (1 - p_{\varphi_k}) a^{11}_{ik\varphi_k}$$

$$\widetilde{a^{12}_{ij\psi_j}} = (1 - p_{\psi_j}) a^{12}_{ij\psi_j}$$

$$\widetilde{a^{21}_{ji\varphi_i}} = \left(1 - p_{\varphi_i}\right) a^{21}_{ji\varphi_i} + p_{\varphi_i}\, b^{1}_{ji\varphi_i}$$

$$\widetilde{a^{22}_{jl\psi_l}} = \left(1 - p_{\psi_l}\right) a^{22}_{jl\psi_l} + p_{\psi_l}\, b^{2}_{jl\psi_l}$$

Сформулюємо цільову функцію:

$$\sum_{j\in \boldsymbol{J}} c_j \left(\sum_{i\in I} \sum_{\varphi_i \in \boldsymbol{P}_i} \widetilde{a^{21}_{ji\varphi_i}} x^1_{i\varphi_i} + \sum_{l\in J} \sum_{\psi_j \in \boldsymbol{Q}_j} \sigma_{jl\psi_l} x^2_{l\psi_l} \right) \to \min$$

Де:

$$\sigma_{jl\psi_l} = \begin{cases} \widetilde{a^{22}_{jl\psi_l}}, & j \neq l \\ \widetilde{a^{22}_{jj\psi_l}} - 1 + \dfrac{c_{j\psi_j}}{c_j}, & j = l \end{cases}$$

Беручи до уваги тезу про максимальний прибуток виробництва, за мінімального впливу на навколишнє середовище, формулюємо мету: мінімізувати витрати на виробництво та знищення забруднювачів

Обмеження:

$$x^1_{i\varphi_i}, x^2_{l\psi_l} \geq 0$$

Обсяг кінцевої продукції не може бути меншим обсягів споживання:

$$\sum_{k\in I}\sum_{\varphi_k\in \boldsymbol{P}_k}(\delta_{ik} - \widetilde{a^{11}_{ik\varphi_k}})\, x^1_{k\varphi_k} - \sum_{j\in \boldsymbol{J}}\sum_{\psi_j\in \boldsymbol{Q}_j}\widetilde{a^{12}_{ij\psi_j}} x^2_{j\psi_j} \geq y^1_i$$

Обсяг незнищених забруднювачів не має перевищувати максимальної кількості викиду:

$$-\sum_{i\in I}\sum_{\varphi_i\in \boldsymbol{P}_i}\widetilde{a^{21}_{ji\varphi_i}} x^1_{i\varphi_i} + \sum_{l\in J}\sum_{\psi_j\in \boldsymbol{Q}_j}(\delta_{jl} - \widetilde{a^{22}_{jl\psi_l}}) x^2_{l\psi_l} \geq -y^2_j$$

де $\delta_{jl} = \begin{cases} 1, & j = l \\ 0, & j \neq l \end{cases}$

Реалізація моделі в Python

$$\sum_{j \in J} c_j \left(\sum_{i \in I} \sum_{\varphi_i \in P_i} \widetilde{a_{jl\varphi_l}^{21}} x_{i\varphi_i}^{1} + \sum_{l \in J} \sum_{\psi_j \in Q_j} \sigma_{jl\psi_l} x_{l\psi_l}^{2} \right) \to \min$$

```
def call_solver(prob_a11, prob_a12, prob_a21, prob_a22, prob_a22_sigma, y1, y2, cj, cjf, p_i, p_k, p_j, p_l, b1, b2):
    prob = pulp.LpProblem('MyLP', pulp.LpMinimize)

    prob_a11_num_j, prob_a11_num_i, prob_a11_num_var = prob_a11.shape
    prob_a12_num_j, prob_a12_num_i, prob_a12_num_var = prob_a12.shape
    prob_a21_num_j, prob_a21_num_i, prob_a21_num_var = prob_a21.shape
    prob_a22_sigma_num_j, prob_a22_sigma_num_i, prob_a22_sigma_num_var = prob_a22_sigma.shape

    x1 = pulp.LpVariable.dicts('x1', (range(1), range(prob_a21_num_i), range(prob_a21_num_var)), lowBound=0,
                               cat=pulp.LpContinuous)
    x2 = pulp.LpVariable.dicts('x2', (range(1), range(prob_a22_sigma_num_i), range(prob_a22_sigma_num_var)), lowBound=0,
                               cat=pulp.LpContinuous)

    expr = pulp.lpSum(cj) * \
        (pulp.lpSum([prob_a21[j][i][var] * x1[0][i][var] for j in range(prob_a21_num_j) for i in range(prob_a21_num_i)
                     for var in range(prob_a21_num_var)]) + \
        pulp.lpSum([prob_a22_sigma[j][i][var] * x2[0][i][var] for j in range(prob_a22_sigma_num_j)
                    for i in range(prob_a22_sigma_num_i) for var in range(prob_a22_sigma_num_var)]))

    prob += expr

    exprlimit1 = pulp.lpSum(prob_a11[j][i][var] * x1[0][i][var] for j in range(prob_a11_num_j) for i in range(prob_a11_num_i)
                            for var in range(prob_a11_num_var)) - \
        pulp.lpSum(prob_a12[j][i][var] * x2[0][i][var] for j in range(prob_a12_num_j) for i in range(prob_a12_num_i)
                   for var in range(prob_a12_num_var)) >= pulp.lpSum(y1)

    prob += exprlimit1

    exprlimit2 = -pulp.lpSum(prob_a21[j][i][var] * x1[0][i][var] for j in range(prob_a21_num_j) for i in range(prob_a21_num_i)
                             for var in range(prob_a21_num_var)) + \
        pulp.lpSum(prob_a22[j][i][var] * x2[0][i][var] for j in range(prob_a22_sigma_num_j) for i in range(prob_a22_sigma_num_i)
                   for var in range(prob_a22_sigma_num_var)) >= - pulp.lpSum(y2)
```

```
    prob += exprlimit2

    prob.solve(pulp.PULP_CBC_CMD(msg=0))

    x1_values_arr = []
    for j in range(1):
        for i in range(prob_a21_num_i):
            for var in range(prob_a21_num_var):
                x1_values_arr.append(pulp.value(x1[j][i][var]))
    x1_values_arr = np.array(x1_values_arr).reshape(1, prob_a21_num_i, prob_a21_num_var)

    x2_values_arr = []
    for j in range(1):
        for i in range(prob_a22_sigma_num_i):
            for var in range(prob_a22_sigma_num_var):
                x2_values_arr.append(pulp.value(x2[j][i][var]))
    x2_values_arr = np.array(x2_values_arr).reshape(1, prob_a22_sigma_num_i, prob_a22_sigma_num_var)

    objective = pulp.value(prob.objective)

    return x1_values_arr, x2_values_arr, objective
```

Генерація значень вхідних даних

```
top_rand_shape = 10
top_rand_value = 100

i_num = random.randint(1, top_rand_shape)
k_num = i_num
i_k_methods_num = random.randint(1, top_rand_shape)
i_j_methods_num = random.randint(1, top_rand_shape)

j_num = random.randint(1, top_rand_shape)
l_num = j_num
j_l_methods_num = random.randint(1, top_rand_shape)

a11 = np.random.rand(i_num, k_num, i_k_methods_num)
a12 = np.random.rand(i_num, j_num, i_j_methods_num)
a21 = np.random.rand(j_num, i_num, i_k_methods_num)
a22 = np.random.rand(j_num, l_num, j_l_methods_num)

y1 = np.random.randint(1, top_rand_value, size=(k_num))
y2 = np.random.randint(1, top_rand_value, size=(l_num))

cj = np.random.randint(1, top_rand_value, size=(j_num))
cjf = np.random.randint(1, top_rand_value, size=(j_num))

p_i = np.random.rand(1, i_num, i_k_methods_num)
p_k = np.random.rand(1, k_num, i_k_methods_num)

p_j = np.random.rand(1, j_num, i_j_methods_num)
p_l = np.random.rand(1, l_num, j_l_methods_num)

b1 = np.random.rand(l_num, i_num, i_k_methods_num)
b2 = np.random.rand(j_num, l_num, j_l_methods_num)
```

```
prob_a11 = (1 - p_k) * a11
prob_a12 = (1 - p_j) * a12
prob_a21 = (1 - p_i) * a21 + p_i * b1
prob_a22 = (1 - p_l) * a22 + p_l * b2

prob_a22_sigma = prob_a22.copy()
for i in range(prob_a22_sigma.shape[0]):
    for j in range(prob_a22_sigma.shape[1]):
        for c in range(prob_a22_sigma.shape[2]):
            if prob_a22_sigma[i][j][c] == 1.0:
                prob_a22_sigma[i][j][c] = prob_a22_sigma[i][j][c] - 1 + cjf[i] / cj[i]
```

$$\widetilde{a^{11}_{\iota k \varphi_k}} = (1 - p_{\varphi_k}) a^{11}_{\iota k \varphi_k}$$

$$\widetilde{a^{12}_{\iota j \psi_j}} = (1 - p_{\psi_j}) a^{12}_{\iota j \psi_j}$$

$$\widetilde{a^{21}_{j\iota \varphi_\iota}} = (1 - p_{\varphi_i}) a^{21}_{j i \varphi_i} + p_{\varphi_i} b^{1}_{j i \varphi_i}$$

$$\widetilde{a^{22}_{j\iota \psi_\iota}} = (1 - p_{\psi_l}) a^{22}_{j\iota \psi_\iota} + p_{\psi_l} b^{2}_{j\iota \psi_\iota}$$

$$\sigma_{j\iota\psi_\iota} = \begin{cases} \widetilde{a^{22}_{j\iota\psi_\iota}}, & j \neq \iota \\ \widetilde{a^{22}_{jj\psi_\iota}} - 1 + \dfrac{c_{j\psi_j}}{c_j}, & j = \iota \end{cases}$$

Згенеровані та переписані дані одразу використовуються функцією call_solver, результат роботи алгоритму та дані записуються в CSV файл.

```
counter = 0
data = []
while counter < 20:
    index_num = str(counter + 1)
    a11, a12, a21, a22, y1, y2, cj, cjf, p_i, p_k, p_j, p_l, b1, b2, prob_a11, prob_a12, prob_a21, prob_a22,
    prob_a22_sigma = randomize_params(10, 100)
    x1, x2, objective = call_solver(prob_a11, prob_a12, prob_a21, prob_a22, prob_a22_sigma, y1, y2, cj, cjf, p_i, p_k, p_j,
                                    p_l, b1, b2)

    colslist = [a11, a12, a21, a22, y1, y2, cj, cjf, p_i, p_k, p_j, p_l, b1, b2, prob_a11, prob_a12, prob_a21, prob_a22,
                prob_a22_sigma, x1, x2, objective]

    for i in range(len(colslist)):
        colslist[i] = np.array2string(colslist[i], precision=3, separator=",").replace("\n", "").replace(" ", "")

    datadict = {"index_num": index_num, "a11": a11, "a12": a12, "a21": a21, "a22": a22, "y1": y1, "y2": y2, "cj": cj,
                "cjf": cjf, "p_i": p_i, "p_k": p_k, "p_j": p_j, "p_l": p_l, "b1": b1, "b2": b2, "prob_a11": prob_a11,
                "prob_a12": prob_a12, "prob_a21": prob_a21, "prob_a22": prob_a22, "prob_a22_sigma": prob_a22_sigma,
                "x1": x1, "x2": x2, "objective": objective}

    data.append(datadict)
    counter += 1
    print(counter, end="\r")
    print("x1:")
    print(x1)
    print("x2:")
    print(x2)
    print("Objective:", objective)

df = pd.DataFrame(data)
df.to_csv("gendata.csv", index=False, sep="|")
```

Результат

	A	B	C	D	E	F	G	H	I	J	K	L	M	N
1	index_num|a11|a12|a21|a22|y1|y2|cj|cjf|p_i|p_k|p_j|p_l|b1|b2|prob_a11|prob_a12|prob_a21|prob_a22|prob_a22_sigma|x1|x2|objective													
2	1|"[[[1.02674520e-01 6.28932411e-01 8.53246073e-02 3.02910219e-01]													
3	[6.99326799e-01 9.12399882e-01 2.61691162e-01 6.63291977e-01]													
4	[6.77204470e-01 8.50473143e-01 4.16009849e-01 4.74789049e-01]													
5	[2.71552954e-01 5.62028504e-01 3.25526856e-01 6.90172381e-01]													
6	[1.92513096e-01 2.63043363e-01 9.88619902e-01 4.92020623e-02]													
7	[8.35727356e-01 4.70868286e-01 8.30228330e-01 1.73693348e-02]													
8	[3.29887797e-01 5.56169429e-02 4.03595849e-01 8.64959923e-01]]													
9														
10	[[5.40196316e-01 6.55216051e-02 5.20342022e-01 2.16476219e-01]													
11	[9.33836154e-01 8.83250473e-01 9.87185781e-01 7.37369946e-01]													
12	[1.03858241e-01 2.71761130e-01 9.89019614e-01 4.98280243e-01]													
13	[9.14715879e-01 6.82918699e-01 2.02899088e-01 1.42942161e-01]													
14	[1.97148896e-01 3.44772382e-01 9.44495301e-01 2.12862364e-01]													
15	[3.80291396e-01 4.82695462e-01 5.27729078e-01 5.22479368e-01]													
16	[9.06393062e-01 3.91489323e-01 5.72865381e-01 9.88760721e-01]]													
17														
18	[[7.78114682e-01 7.52121407e-03 5.14744146e-01 1.48501091e-01]													
19	[4.81570229e-01 7.37278501e-01 5.80591004e-01 2.40396606e-01]													
20	[9.42294821e-01 8.20879876e-04 2.01736821e-01 9.01573950e-02]													
21	[5.24649484e-01 1.31493693e-01 7.07049813e-01 8.54363030e-01]													
22	[9.36678861e-01 2.22808476e-01 7.63845413e-01 1.14615299e-01]													
23	[3.87304827e-01 6.75055335e-01 1.07382599e-01 8.47613660e-01]													
24	[5.72234046e-01 2.31330859e-01 9.32952998e-01 8.11498986e-02]]													
25														

Дякую за увагу!

Зображення взято з ресурсів

https://www.google.com/maps/d/u/0/viewer?mid=1kI7M2ngkfqQ78HuA0sCKGEaENzg&hl=en_US&ll=50.48501332653812%2C29.816213565625
07&z=8

https://www.flaticon.com/