ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД УКООПСПІЛКИ
«ПОЛТАВСЬКИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТОРГІВЛІ»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ БІЗНЕСУ ТА СУЧАСНИХ ТЕХНОЛОГІЙ

ФОРМА НАВЧАННЯ ДЕННА
КАФЕДРА МАТЕМАТИЧНОГО МОДЕЛЮВАННЯ ТА СОЦІАЛЬНОЇ ІНФОРМАТИКИ

Допускається до захисту

Завідувач кафедри _____О.О. Ємець
(підпис)

«_____»_____2021 р.

ПОЯСНЮВАЛЬНА ЗАПИСКА ДО БАКАЛАВРСЬКОЇ РОБОТИ

на тему

Ігрові задачі комбінаторного типу: програмне забезпечення і дослідження

зі спеціальності 122 «Комп'ютерні науки»

Виконавець роботи Деркач Роман Анатолійович _____ «___»_____2021р.
(підпис)

Науковий керівник д.ф.-м.н., проф., Ємець Олег Олексійович
_____ «___»_____2021р.
(підпис)

ПОЛТАВА 2021 р.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Умовні позначення, символи, одиниці, скорочення, терміни	**Пояснення умовних позначень, символів,одиниць, скорочень, термінів**
ІМТБР	Ітераційний метод типу Брауна-Робінсон
A	Платіжна матриця
p^*	Наближене значення мішаних стратегій першого гравця
q^*	Наближене значення мішаних стратегій другого гравця
v^*	Наближення значення ціни гри
X	Обмеження-перестановка для першого гравця
Y	Обмеження-перестановка для другого гравця

ВСТУП

Актуальність теми пов'язана з тим, що серед задач комбінаторної оптимізації все популярнішими стають задачі розв'язування конфліктних ситуацій, котрі відносяться до класу задач комбінаторної оптимізації ігрового типу з обмеженнями на стратегії гравців, що визначаються комбінаторними конфігураціями, в тому числі множинами перестановок та розміщень. Для подібних задач є необхідними дослідження уже існуючих методів розв'язування, можливостей їхньої модифікації та проведення різноманітних експериментів з метою визначення їх практичної ефективності.

Об’єкт розробки – програмне забезпечення для методу Брауна-Робінсон.

Предмет розробки – консольний додаток, котрий реалізує ІМТБР.

Мета бакалаврської роботи – створити програму, яка реалізує ітераційний метод типу Брауна-Робінсон для розв’язування ігрових задач з обмеженнями-перестановками у обох гравців.

Задачі бакалаврської роботи: 1)здійснити огляд та аналіз робіт, подібних до теми бакалаврської роботи. 2)реалізувати програмно ІМТБР.3)Розробити блок-схему ІМТБР. 4) Протестувати на різних 3 тестових випадках та проаналізувати результати програми.5)Зробити висновки.

Методи розробки – мова програмування С++, середовище програмування IDE Code::Blocks 13.12 та ІМТБР.

Нові практичні розробки – створено консольний додаток, котрий реалізує ІМТБР.

Ступінь готовності до використання – програма повністю готова до використання.

Структура роботи: бакалаврська робота складається з чотирьох частин.

Перша частина – постановка задачі – містить технічне завдання дороботи.

Друга частина – інформаційний огляд – містить характеристику та аналіз робіт, котрі схожі за темою до бакалаврської роботи, що викладені в мережі Інтернет.

Третя частина – теоретична – містить алгоритм ІМТБР для програми, блок-схему алгоритму.

Четверта частина – практична – містить опис програмної реалізації ІМТБР та інструкції по користуванню програмою, а також результати числових експериментів.

Обсяг пояснювальної записки: 68 сторінок, в т.ч. основна частина 41 стор., джерел – 15.

1.ПОСТАНОВКА ЗАДАЧІ

Розглянемо комбінаторну оптимізаційну задачу ігрового типу на множині перестановок, в якій комбінаторні обмеження накладаються на стратегії одного гравця [1].

Нехай елементи P_i^x мультимножини $P^x = \left\{P_1^x,...,P_m^x\right\}$ $0 \le P_i^x \le 1$, $i \in J_m = \left\{1,2,...,m\right\}$, сума їх $\sum_{i=1}^{m} P_i^x = 1$. У векторі $X = (x_1, x_2,..., x_m)$ елемент x_i – ймовірність застосування стратегії з номером i, належить P^x, $x_i \in P^x$, а сам вектор X належить множині $E_{mv}\left(P^x\right)$ m-перестановок з елементів мультимножини P^x, тобто $X = \left(x_1, x_2,..., x_m\right) \in E_{mv}\left(P^x\right)$. Очевидно, що $\sum_{i=1}^{m} x_i = 1$

.

Гра полягає в тому, що перший гравець обирає стратегію-вектор $X = \left(x_1, x_2,..., x_m\right) \in E_{mv}\left(P^x\right)$, а другий обирає стратегію-число $j \in J_n$. Ці стратегії назвемо чистими. При цьому другий гравець платить першому платежі $a'_{1j},...,a'_{mj}$ з ймовірностями $x_1,...,x_m$, де a'_{ij} – задані дійсні числа $\forall i \in J_m$ $\forall\ j \in J_n$.

Позначимо A' матрицю з елементами a'_{ij}. Середній платіж (математичне сподівання) другого гравця першому (при виборі стратегії $x^i = \left(x_{1i},...,x_{mi}\right) \in E_{mv}\left(P^x\right)$ і стратегії $j \in J_n$ відповідно першим та другим гравцями, $i \in J_k$) виражається функцією:

$$F\left(x^i, j\right) = \sum_{t=1}^{m} a'_{tj} x_{it} = a_{ij}, \qquad (1.1)$$

де $k = \left|E_{mv}\left(P^x\right)\right| = \dfrac{m!}{\eta_1! \cdot ... \cdot \eta_v!}$.

Модель 1. Необхідно знайти стратегії гравців X^* і j^*, які дозволять першому гравцю максимізувати свій виграш, а другому – мінімізувати програш:

$$X^* = \arg \max_{X \in E_{mv}\left(P^x\right)} \left(\min_{j \in J_n} F\left(X, j\right) \right),$$

$$j^* = \arg \min_{j \in J_n} \left(\max_{X \in E_{mv}\left(P^x\right)} F\left(X, j\right) \right),$$

де функція $F\left(\mathrm{X}, j\right)$ має вигляд (1.1).

Якщо виконується умова

$$\min_{j \in J_n} \max_{X \in E_{mv}\left(P^x\right)} F\left(X, j\right) = \max_{X \in E_{mv}\left(P^x\right)} \min_{j \in J_n} F\left(X, j\right) = F\left(X^*, j^*\right), \qquad (2.2)$$

то очевидно, що задача розв'язана з ціною гри $\upsilon = F\left(X^*, j^*\right)$ та оптимальними чистими стратегіями X^*, j^* першого та другого гравців відповідно. При цьому кажуть, що існує сідлова точка гри $\left(X^*, j^*\right)$.

Якщо умова (2.2) не виконується, то, очевидно, що використання кожним з граців його фіксованої чистої стратегії дозволяють іншому отримувати переваги. З огляду на це, кожен з гравців для того, щоб при багатократному повторенні гри досягти своєї мети, повинен застосувати свої чисті стратегії з певною частотою (ймовірністю).

У цьому випадку для пошуку оптимальних стратегій, введемо поняття мішаних стратегії. Позначимо

$$S_k = \left\{ p = (p_1, p_2, ..., p_k),\ \sum_{i=1}^{k} p_i = 1,\ p_i \geq 0\ \ \forall i \in J_n \right\},$$

$$S_n = \left\{ q = (q_1,..., q_n),\ \sum_{j=1}^{n} q_j = 1,\ q_j \geq 0\ \ \forall\, j \in J_m \right\},$$

де k – кількість елементів в $E_{mv}\left(P^x\right)$, $p \in S_k$ – мішана стратегія першого гравця, $p = (p_1, p_2,..., p_k)$, $p_i \geq 0$, $\sum_{i=1}^{k} p_i = 1$. Аналогічно, мішаною стратегією другого гравця є елемент $q \in S_n$, $q = (q_1,..., q_n)$, $q_j \geq 0$, $\sum_{j=1}^{n} q_j = 1$. Числа p_i, q_j є ймовірностями застосування стратегій $x^i \in E_{mv}\left(P^x\right)$ та $j \in J_n$ першого та другого гравців відповідно.

Очікуваною платою другого гравця першому є величина $F(p,q)$ – математичне сподівання випадкової величини, яка реалізується при одночасному настанні випадкових подій: вибір стратегії x^i першим гравцем та вибір стратегії j – другим. Ця випадкова величина приймає значення a_{ij} $\forall i \in J_k$, $\forall j \in J_n$ з ймовірністю $p_i q_j$:

$$F\left(p,q\right) = \sum_{j=1}^{n}\sum_{i=1}^{k}\sum_{t=1}^{m} a'_{tj} x_{it}\, p_i q_j = \sum_{j=1}^{n}\sum_{i=1}^{k} a_{ij}\, p_i q_j\,, \qquad (1.3)$$

де p_i – ймовірність вибору $x^i = \left(x_{i1},..., x_{im}\right)$, а q_j – ймовірність вибору j.

Природно, що очікуваний програш другого гравця обчислюється аналогічно відповідно формули (1.3), оскільки маємо гру з нульовою сумою.

Не важко бачити, що перший гравець може забезпечити собі виграш не менше $\max\limits_{p \in S_k} \min\limits_{q \in S_n} \sum_{j=1}^{n}\sum_{i=1}^{k} a_{ij}\, p_i q_j$, а другий гравець може забезпечити собі програш

не більше $\min\limits_{p\in S_k}\max\limits_{q\in S_n}\sum\limits_{j=1}^{n}\sum\limits_{i=1}^{k}a_{ij}p_iq_j$. Якщо$\left(p^*,q^*\right)$ − сідлова точка функції $F\left(p,q\right)$, що визначається (2.3), тобто виконуються нерівності

$$F\left(p^*,q\right)\le F\left(p^*,q^*\right)\le F\left(p,q^*\right),$$

то p^*, q^* називають оптимальними мішаними стратегіями першого та другого гравців відповідно. У цьому випадку, як відомо,

$$\upsilon = F\left(p^*,q^*\right)=\min_{q\in S_n}\max_{p\in S_k}F\left(X,Y\right)=\max_{p\in S_k}\min_{q\in S_n}F\left(p,q\right).$$

При цьому будемо казати, що задача комбінаторної оптимізації ігрового типу на перестановках (ЗКОІТП) має розв'язок в мішаних стратегіях, а $F\left(p^*,q^*\right)$ – ціна гри.

Модель 2. Розглянемо випадок, коли і на стратегії другого гравця накладаються обмеження, що визначені перестановками, тобто вектор $P^y=(P_1^y,P_2^y,...,P_l^y)$ – вектор, для якого виконується: $Y=(y_1,y_2,...,y_l)\in E_{L\mu}(P^y)$, $P_j^y\ge 0\ \ \forall j\in J_l;\ \sum\limits_{j=1}^{l}P_j^y=1.$ Необхідно знайти ймовірності стратегій гравців X^* і Y^*, які дозволять першому гравцю максимізувати свій виграш, а другому – мінімізувати програш:

$$X^*=\arg\max_{X\in E_{mv}\left(P^x\right)}\left(\min_{Y\in E_{L\mu}(P^y)}F\left(X,Y\right)\right),$$

$$Y^*=\arg\min_{Y\in E_{L\mu}(P^y)}\left(\max_{X\in E_{mv}\left(P^x\right)}F\left(X,Y\right)\right),$$

де $F(X,Y)=\sum_{i=1}^{k}x_i\sum_{j=1}^{l}a'_{ij}y_j=\sum_{i=1}^{k}\sum_{j=1}^{l}a'_{ij}x_iy_j$, $k=\left|E_{mv}\left(P^x\right)\right|=\frac{m!}{\eta_1!\cdot\ldots\cdot\eta_v!}$,

$l=\left|E_{L\mu}\left(P^x\right)\right|=\frac{L!}{\lambda_1!\cdot,\ldots,\cdot\lambda_v!}$.

Ця математична модель описує задачу комбінаторної оптимізації ігрового типу з обмеженнями-перестановками на стратегії обох гравців.

В бакалаврській роботі необхідно розробити програму типу «консольний додаток» з теми «Ігрові задачі комбінаторного типу».

Програма призначена для розв’язку ігрових задач комбінаторного типу з обмеженнями-перестановками для обох гравців, котрі базуються на використанні ІМТБР.

В якості першого прикладу, взяти приклад з джерела [1].

Наступні два приклади розробити самостійно.

У програмі передбачити вивід відповіді у вихідний файл. Відповідь повинна бути від першої до останньої ітерації алгоритму. Відповідь доцільно записати у вигляді таблиці у текстовий файл.

2.ІНФОРМАЦІЙНИЙ ОГЛЯД

2.1. Огляд робіт

Розглянемо подібні роботи, до теми бакалаврської роботи.

1)Робота «Розв'язування комбінаторних задач ігрового типу з обмеженнями-переставленнями у обох гравців: ітераційний метод» (автори: О.О.Ємець, О.В.Ольховська) [2].

В даній роботі розглянуто постановку та математичну модель ігрової задачі сільськогосподарського виробництва з обмеженнями-перестановками для обох гравців. Також в ній наведено ІМТБР, ілюстративний приклад роботи даного алгоритму, результати числових експериментів, а також оцінка складності алгоритму. На рис. 2.1 та рис.2.2 показана частина даної роботи.

РЕЗУЛЬТАТИ ЧИСЛОВИХ ЕКСПЕРИМЕНТІВ

За допомогою розробленої програмної реалізації цього методу, проведено числовий експеримент з метою виявлення залежності часу обчислень від вимірності задачі. Результати експерименту внесено в табл. 2, де: t — загальний час серії в секундах (с); $t_{сер}$ — середній час задачі в секундах (с) або мілісекундах (мс); t_{im} — середній час ітерації в мілісекундах (мс); $\Delta = \min \bar{v} - \max \underline{v}$; $\delta = \dfrac{\Delta}{(1/2)(|\min \bar{v}| + |\max \underline{v}|)}$; $\delta_{\min}$ — мінімальне δ в серії обчислень; $\delta_{\max}$ — максимальне δ в серії обчислень.

Таблиця 2. Результати першої серії експериментів

№	$m=n$	$[a,b]$	t	$t_{сер}$	t_{im} (мс)	Δ сере	$\delta_{\min}$	$\delta_{\max}$	$\delta_{сер}$	10^{-4}	10^{-5}
1	10	[1,200]	< 1 с	470мс	<1	0,0119	1,06E-5	0,000354	0,000129	100	39
2	20	[1,400]	1 с	14 мс	<1	0,0177	2,52E-5	0,000164	8,99E-5	100	66
3	30	[1,600]	3 с	30 мс	<1	0,0246	3,27E-5	0,000164	8,26E-5	100	82
4	40	[1,800]	5 с	50 мс	<1	0,0279	3,8E-5	0,000114	7,02E-5	100	94
5	50	[1,1000]	7 с	74 мс	<1	0,0306	4,22E-5	9,85E-5	6,12E-5	100	100
6	60	[1,1200]	10 с	101 мс	<1	0,0334	3,37E-5	8,33E-5	5,58E-5	100	100
7	70	[1,1400]	13 с	136 мс	<1	0,0366	3,19E-5	7,12E-5	5,24E-5	100	100
8	80	[1,1600]	17 с	167мс	<1	0,0394	3,44E-5	6,76E-5	4,93E-5	100	100
9	90	[1,1800]	21 с	214 мс	<1	0,0419	3,27E-5	6,66E-5	4,65E-5	100	100
10	100	[1,2000]	25 с	255 мс	<1	0,0451	3,12E-5	6E-5	4,51E-5	100	100
11	200	[1,4000]	3 хв,21с	1 с	1	0,0651	2,38E-5	4,13E-5	3,26E-5	100	100
12	300	[1,6000]	5 хв,4 2с	4 с	4	0,0831	2,21E-5	3,49E-5	2,77E-5	100	100
13	400	[1,8000]	10 хв,51с	6 с	6	0,106	2,18E-5	3,83E-5	2,65E-5	100	100
14	500	[1,10000]	17 хв,32с	10 с	10	0,129	2,06E-5	4,24E-5	2,6E-5	100	100
15	600	[1,12000]	24 хв,44с	14 с	15	0,206	2,07E-5	7,1E-5	3,48E-5	100	100
16	700	[1,14000]	35 хв,24с	21 с	21	0,201	2,05E 5	5,73E 5	2,88E 5	100	100

Рис. 2.1 – Результати числових експериментів

ОЦІНКА СКЛАДНОСТІ АЛГОРИТМУ

Для розрахунку складності алгоритму [16] складемо табл. 3, де у стовпці «Алгоритм» записано програмну реалізацію ітераційного методу (одна ітерація) для розв'язування комбінаторних оптимізаційних задач ігрового типу на перестановках, в яких накладаються обмеження, що визначаються переставленнями, на стратегії обох гравців. Час виконання різних рядків алгоритму різний, але один і той рядок i виконується за час c_i, де c_i — константа. Позначимо через τ_j — кількість раз виконання умови.

Під час розрахунку складності алгоритму потрібно визначити асимптотичну верхню границю з точністю до постійного множника [16]. Для функції $g(n)$ позначка $O(g(n)) = f(n)$ з [16] означає множину функцій таких, що існує додатня константа c і n_0 така, що $0 \le f(n) \le cg(n)$ для всіх $n \ge n_0$.

Підрахуємо $T = \sum_{i=1}^{39} c_j \tau_j$:

$$T = 1(c_1 + c_8 + c_{11} + c_{15} + c_{16} + c_{25} + c_{30} + c_{31} + c_{32} + c_{33}) +$$

$$+ n(c_2 + c_3 + c_6 + c_7 + c_9 + c_{10} + c_{12} + c_{13} + c_{14}) +$$

$$+ m(c_{17} + c_{18} + c_{21} + c_{22} + c_{23} + c_{24} + c_{26} + c_{27} + c_{28} + c_{29}) +$$

$$+ nm(c_4 + c_5 + c_{19} + c_{20}) + O(n \log n) + O(m \log m) + T_0(m) + T_1(n)$$

або

$$T = O(1) + O(n) + O(m) + O(nm) + O(n \log n) + O(m \log m) + T_0(m) + T_0(m),$$

якщо $T_0(m) = O(m)$, $T_1(n) = O(n)$, то

$$T = O(1) + O(n) + O(m) + O(nm) + O(n \log n) + O(m \log m) + O(m) + O(n).$$

Враховуючи, що $\forall c > 0 : cO(f(n)) = O(f(n))$, то

$$T = O(n) + O(m) + O(nm) + O(n \log n) + O(m \log m)$$

Рис.2.2 – Оцінка складності алгоритму ІМТБР.

2)Стаття «Програмний комплекс, що реалізує методи розв'язування задач комбінаторної оптимізації ігрового типу»(автори: О.О.Ємець, Д.М.Ольховський, О.В.Ольховська)[3]

В даній статті наведений опис програмного комплексу, для розв'язування задач комбінаторної оптимізації ігрового типу, програмно реалізовано ІМТБР. Окрім цього програмний комплекс реалізує монотонний ітераційний метод, генерування початкових даних, симплекс-метод. Програмний продукт має підтримку на 32-х та 64-х бітну версію Windows. Дані можна вводити як з клавіатури так і з файлу. Вихідні дані(відповідь) можна зберігати у файл або виводити на екран.

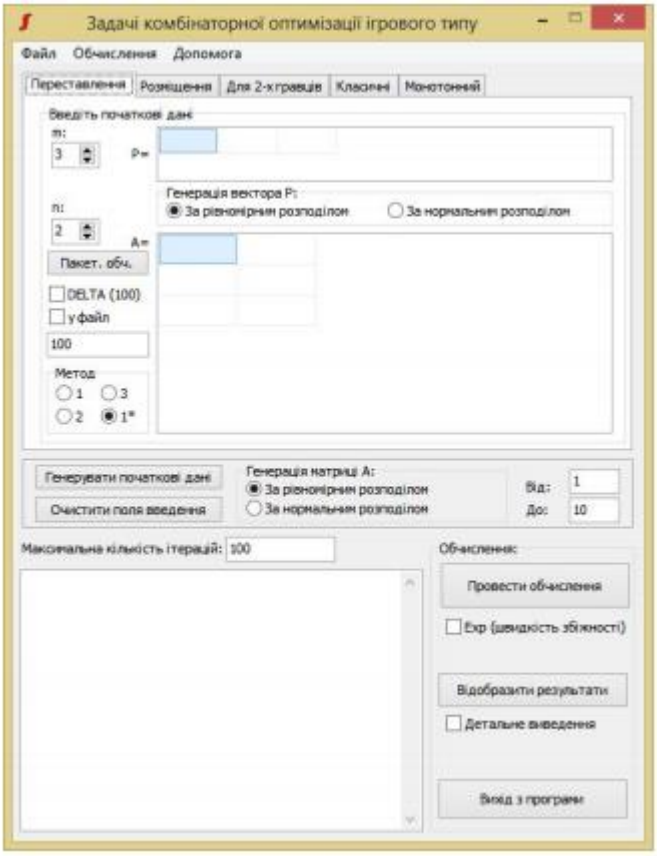

Рис.2.3 – Графічний інтерфейс програмного комплексу.

3)Стаття Теоретична оцінка складності алгоритмів розв'язування задач комбінаторної оптимізації ігрового типу(автори: О.О.Ємець, Д.М.Ольховський, О.В.Ольховська)[4]

В даній роботі наведено аналіз основних досліджень та публікацій, теоретична оцінка складності алгоритму ітераційного методу(з обмеженнями-розміщеннями на стратегії одного гравця), а також наведений код на мові програмування Pascal.

№ кроку	Алгоритм	Час c_j	Кількість раз τ_j
0	Встановлення номеру ітерації N	c_1	1
1	Визначення першої стратегії першим гравцем	c_2	n
2	Обчислення скалярних добутків векторів стратегій другого гравця на вектор стратегії першого гравця	1	$O(n \cdot m)$
3	Обчислення накопичених сум скалярних добутків	1	$O(m)$
4	Вибір стратегії другого гравця	1	$O(m)$
5	Вибір стратегії першого гравця	1	$S(m,n)$
6	Обчислення значень $\overline{v}$, $\underline{v}$ та v^*	c_3	1
7	Перевірка критерію завершення роботи алгоритму	c_4	1

Враховуючи, що $\forall c > 0 : cO(f(n)) = O(f(n))$, то $T = O(n \cdot m) + O(m) + S(m,n)$.

Відомо [15], що якщо $\lim\limits_{n \to \infty} \frac{f(n)}{g(n)} = \infty$, то $O(f(n)) + O(g(n)) = O(f(n))$. З огляду на це, складність алгоритму набуде вигляду: $T = O(n \cdot m) + S(m,n)$.

Рис.2.4 – Оцінка складності алгоритму

2.2Позитивні аспекти оглянутих робіт

В роботі [2]:

1) Наявна конкретна постановка задачі;
2) Побудована математична модель до задачі;
3) ІТМБР показаний на прикладі;
4) Наявні результати числових експериментів.

В статті [3]:

1) Програмний комплекс розв’язує достатньо велику кількість різних задач;

2) Є генерування початкових даних;

3) Вхідні та вихідні дані можна записувати як з клавіатури(з консолі) так і з файла;

В статті [4]:

1) Наведена оцінка складності алгоритму;

2) Є в наявності програмна реалізація на мові програмування Pascal;

2.3. Недоліки оглянутих робіт

В роботі [2]:

1) Відсутній код, котрий реалізує програмно ІМТБР.

В статті [3]:

1) Відсутній ілюстративний приклад роботи з програмним комплексом;

2) Немає достатнього числового експерименту, щоб побачити швидкість роботи програмного комплексу.

В статті [4]:

1) Немає достатнього числового експерименту, щоб побачити швидкість роботи алгоритму.

2.4 Необхідність та актуальність теми

При застосуванні робіт, котрі розглянуті в інформаційному огляді, виникає декілька проблем:

1) Відсутність програм, котрими можна було скористатися при розв’язуванні поставленої задачі;автору, не вдалося відшукати в глобальній мережі Інтернет доступних програм, котрі реалізують ІМТБР для поставленої задачі;
2) Немає достатнього числового експерименту, щоб побачити

швидкість роботи ІМТБР програмно;

3) Невелика кількість робіт, котра подібна за тематикою до бакалаврської роботи;
4) Не у всіх роботах описаний сам алгоритм.

Таким чином, потрібно врахувати всі недоліки розглянутих робіт з інформаційного огляду та створити консольний додаток, який повинен дотримуватися вимогам:

1. Простий інтерфейс;
2. Швидка дія виконання програми;
3. Запис результатів до файлу;
4. Введення вхідних даних як з файлу так і з клавіатури.

3.ТЕОРИТИЧНА ЧАСТИНА

3.1Алгоритм програми

Використаємо метод з джерела[1] для розв'язування задачі. Опишемо алгоритм методу.

Крок 0. Встановлюємо номер N ітерації рівний 1: $N=1$.

Крок 1. Обираємо першу стратегію (перестановки X) першого гравця випадковим чином з множини перестановок $E_m\left(P^x\right)$. Вона записується покоординатно в стовпець X таблиці.

Крок 2. У таблиці в стовпці B_j з елементами b_j $\forall j \in J_n$ записується вектор-стовпець з номером j з матриці A. Обчислюються скалярні добутки векторів-стратегій B_j другого гравця і вектора обраної стратегії першого гравця. У таблиці це зручно оформлювати, ввівши стопці: B_jX – вектор, що складається з поелементних добутків векторів X та B_j, $j \in J_n$. Ці вектори (X, B_j, B_jX) займають m рядків таблиці. У наступному рядку sum_l в стовпцях B_jX записуємо скалярні добутки векторів B_j та X (як сума елементів стовпця B_jX таблиці).

Крок 3. Знаходяться значення SUM_L – накопичені суми скалярних добутків (в лівій частині таблиці). У рядку SUM_L таблиці записується сума значень елементів рядка sum_l та рядка SUM_L з попереднього ($(N-1)$-го) етапу. При $N=1$ рядок SUM_L збігається з рядком sum_l цього етапу.

Крок 4. Стратегія $NextY$ другого гравця обирається з умови отримання ним максимального сумарного за N етапів платежу, як розв'язок задачі максимізації цільової функції на множині перестановок. Практично це означає

впорядкування елементів вектора P^x відповідно до порядку елементів рядка SUM_L.

Крок 5. Обчислюється значення $N\bar{v}$ – максимальний накопичений виграш другого гравця як скалярний добуток рядка SUM_R та стратегії $NextY$, воно записується в цьому ж рядку в стовпці $N\bar{v}$.

Крок 6. Обчислюється значення $\bar{v}$ за формулою $\bar{v} = \frac{N\bar{v}}{N}$ та записується в цьому ж рядку в стовпці $\bar{v}$.

Крок 7. У стовпець Y записується значення стратегії другого гравця. У стовпець Y покоординатно заноситься значення рядка $NextY$ лівої частини таблиці.

Крок 8. У праву частину таблиці у стовпці A_i $\forall i \in J_m$, записується вектор стратегії першого гравця – рядок i з матриці A. Обчислюється скалярні добутки вектора-стратегії другого гравця на вектори стратегій першого гравця – стовпці A_i $\forall i \in J_m$. У таблиці це зручно оформлювати, ввівши стопці: A_iY – вектор, що складається з поелементних добутків векторів Y та A_i, $i \in J_m$. Вектори Y, A_i, A_iY займають n рядків таблиці.

Крок 9. Обчислюються значення sum_r та SUM_R. У наступному рядку sum_r в стовпцях A_iY записуються скалярні добутки векторів A_i та Y (як сума елементів стовпця A_iY таблиці). У наступному рядку SUM_R таблиці записується сума значень елементів рядка sum_r та рядка SUM_R з попереднього (($N-1$)-го) етапу.

Крок 10. Стратегія $Next\,X$ (права частина таблиці) першого гравця обирається з умови отримання ним мінімального сумарного за N етапів платежу (програшу), тобто розв'язок задачі мінімізації цільової функції на

множині перестановок аналогічно до кроку 4. Обрана стратегія записується в рядок $NextX$ правої частини таблиці та стовпець X.

Крок 11. Знаходиться значення $N\underline{v}$ – мінімальний накоплений програш другого гравця. Мінімальне значення $N\underline{v}$ правої частини таблиці обчислюється як скалярний добуток рядка SUM_R та стратегії $NextX$ (правої частини таблиці) та записується у цьому ж рядку в стовпці $N\underline{v}$.

Крок 12. За формулою $\underline{v} = \frac{N\underline{v}}{N}$ обчислюється $\underline{v}$ та заноситься у стовпець $\underline{v}$ таблиці.

Крок 13. За формулою $\upsilon^* = \frac{\overline{v} + \underline{v}}{2}$ обчислюється υ^* та заноситься у стовпець υ^* таблиці.

Крок 14. Перевіряється критерій завершення роботи алгоритму – проведення заданої кількості ітерацій. Якщо критерій N = MAX_ITERATION виконується, то зупинка алгоритму, інакше – перехід на крок 2 алгоритму, обравши за стратегію першого гравця стратегію $NextX$. N збільшуємо на одиницю, N=N+1. За ціну гри приймається значення $\upsilon^* = \overline{v} = \underline{v}$.

Наближені значення p^*, q^* (оптимальні мішані стратегії першого та другого гравців) – це вектори частот застосування чистих стратегій–перестановок гравцями. Неважко помітити, що для забезпечення більшої точності знаходження чистих стратегій гравцями, необхідно проводити якомога більшу кількість ітерацій.

Ілюстративний приклад. Розглянемо роботу ІМТБР розв’язування комбінаторних оптимізаційних задач ігрового типу на перестановках, коли комбінаторні обмеження накладаються на стратегії обох гравців.

Нехай задана платіжна матриця $A=\begin{pmatrix}7 & 3\\ 1 & 5\\ 3 & 4\end{pmatrix}$, на стратегії першого гравця накладаються обмеження, що визначаються перестановками з множини $P^x=\{0.1;0.3;0.6\}$, тобто

$$E_3(P^x)=\{(0.1;0.3;0.6),(0.3;0.1;0.6),(0.3;0.6;0.1),(0.6;0.3;0.1),$$
$$(0.1;0.6;0.3),(0.6;0.1;0.3)\},$$

на стратегії другого гравця накладаються комбінаторні обмеження з множини $P^y=\{0.2;0.8\}$, що визначені перестановками: $E_2(P^y)=\{(0.2;0.8),(0.8;0.2)\}$.

Всі розрахунки занесемо в таблицю (табл. 3.1), в якій N – номер ітерації, X – вибрана стратегія першого гравця, $B_1,B_2,...,B_n$ – стратегії другого гравця, $B_1X,B_2X,...,B_nX$ – скалярний добуток векторів, $N\bar{\nu}$ – максимальний накопичений виграш (максимальний з накопичених скалярних добутків SUM_L і $Next\,X$), $\bar{\nu}=\frac{N\bar{\nu}}{N}$, Y – вибрана стратегія другого гравця, $A_1,A_2,...A_m$ – стратегії другого гравця; числа в рядку з назвою SUM_L – суму значень елементів рядка sum_l та рядка SUM_L з попереднього $((N-1)$-го) етапу, SUM_R – сума значень елементів рядка sum_r та рядка SUM_R з попереднього $((N-1)$-го) етапу, $Next\,X$ – вибрана стратегія першого гравця, $Next\,Y$ – вибрана стратегія другого гравця, $N\underline{\nu}$ – мінімальний накопичений платіж (програш), $\bar{\nu}=\frac{N\bar{\nu}}{N}$ $\underline{\nu}=\frac{N\underline{\nu}}{N}$ $\upsilon^*=\frac{\bar{\nu}+\underline{\nu}}{2}$.

Розв’язування.

Крок 0. Встановлюємо номер ітерації $N=1$.

Крок 1. Перша чиста стратегія першого гравця обирається випадковим чином: $(0.1;0.3;0.6)$.

Крок 2. Обчислюються значення скалярних добутків $B_j X$:

$$B_1 X = (0.1 * 7; 0.3 * 1; 0.6 * 3) = (0.7; 0.3; 1.8)$$

$$B_2 X = (0.1 * 3; 0.3 * 5; 0.6 * 4) = (0.3; 1.5; 2.4)$$

та значення $sum\_l = (0.7 + 0.3 + 1.8; 0.3 + 1.5 + 2.4) = (2.8; 4.2)$.

Крок 3. Знаходяться значення, на першій ітерації вони дорівнюють значенням вектора sum_l: $SUM\_L = (2.8; 4.2)$.

Крок 4. Визначається стратегія другого гравця, серед двох значень вектора SUM_L – більшому ставиться у відповідність більше значення вектора-перестановок другого гравця, меншому – менше: $Next Y = (0.2; 0.8)$.

Крок 5. Обчислюється значення $N\bar{v}$:

$$N\bar{v} = \max(2.8 \cdot 0.2; 4.2 \cdot 0.8) = \max(0.56; 3.92) = 3.92.$$

Крок 6. Обчислюється значення $\bar{v}$: $\bar{v} = \frac{3.92}{1} = 3.92$.

Крок 7. Записуються значення вектора Y відповідно до обраної стратегії другого гравця: $Y = \begin{pmatrix} 0.2 \\ 0.8 \end{pmatrix}$.

Крок 8. Обчислюються скалярні добутки $A_j Y$ векторів стратегії другого гравця на стратегії першого гравця:

$$A_1 Y = (0.2 \cdot 7; 0.8 \cdot 3) = (1.4; 2.4),$$

$A_2Y = (0.2 \cdot 1; 0.8 \cdot 5) = (0.2; 4)$,

$A_3Y = (0.2 \cdot 3; 0.8 \cdot 4) = (0.6; 3.2)$.

Крок 9. Обчислюються значення sum_r та SUM_R:

$sum\_r = (1.4 + 2.4; 0.2 + 4; 0.6 + 3.2) = (3.8; 4.2; 3.8)$,

SUM_R на першій ітерації дорівнює вектору sum_r.

Крок 10. Визначається стратегія $Next\,X$ першого гравця шляхом впорядкування значень компонентів вектору SUM_R за спаданням з подальшим встановленням відповідності максимальним значенням компонентів вектору SUM_R мінімальних значень вектора-перестановки першого гравця: $Next\,X = (0.3; 0.1; 0.6)$.

Крок 11. Обчислюється значення $N\underline{v}$ як скалярний добуток векторів SUM_R та $Next\,X$: $N\underline{v} = 3.8 \cdot 0.3 + 4.2 \cdot 0.1 + 3.8 \cdot 0.6 = 1.14 + 0.42 + 2.28 = 3.84.$

Крок 12. Обчислюється значення $\underline{v} = \frac{3.84}{1} = 3.84$.

Крок 13. Обчислюється значення $\upsilon^* = \frac{3.92 + 3.84}{2} = 3.88$.

Крок 14. Виконується перевірка критерію завершення роботи алгоритму:N=MAX_ITERATION :1≠20. Умова зупинки алгоритму не задовольняється, тому виконується перехід на крок 2 алгоритму, обравши стратегію першого гравця та збільшивши номер N ітерації на 1: $Next\,X = (0.3; 0.1; 0.6)$, $N = 1 + 1 = 2$.

Табл.3.1 - 1, 2 та 20 ітерації розв’язування ЗКОТП з обмеженнями на стратегії обох гравців ІМТБР

N	X	B_1	B_1X	B_2	B_2X	$N\overline{v}$	$\overline{v}$	γ	A_1	A_1Y	A_2	A_2Y	A_3	A_3Y	$N\underline{v}$	$\underline{v}$	υ^*
1	0,1	7	0,7	3	0,3			0,2	7	1,4	1	0,2	3	0,6			
	0,3	1	0,3	5	1,5			0,8	3	2,4	5	4	4	3,2			
	0,6	3	1,8	4	2,4			*sum _*r		3,8		4,2		3,8			
	sum _l		2,8		4,2			*SUM _ R*		3,8		4,2		3,8			
	SUM _L		2,8		4,2			*Next X*		0,3		0,1		0,6	3,84	3,84	3,9
	Next Y		0,2		0,8	3,92	3,92										
2	0,3	7	2,1	3	0,9			0,2	7	1,4	1	0,2	3	0,6			
	0,1	1	0,1	5	0,5			0,8	3	2,4	5	4	4	3,2			
	0,6	3	1,8	4	2,4			*sum _*r		3,8		4,2		3,8			
	sum _l		4		3,8			*SUM _ R*		7,6		8,4		7,6			
	SUM _L		6,8		8			*Next X*		0,3		0,1		0,6	7,68	3,84	3,86
	Next Y		0,2		0,8	7,76	3,88										
… … …																	
20	0,3	7	2,1	3	0,9			0,2	7	1,4	1	0,2	3	0,6			
	0,1	1	0,1	5	0,5			0,8	3	2,4	5	4	4	3,2			
	0,6	3	1,8	4	2,4			*sum _*r		3,8		4,2		3,8			
	sum _l		4.0		4,2			*SUM _ R*		78,4		81,6		75,4			
	SUM _L		72,8		78,5			*Next X*		0,3		0,1		0,6	76,9	3,85	3,86
	Next Y		0,2		0,8	77,4	3,87										

3.2 Блок-схема алгоритму

На рисунках 3.1-3.3 показана блок-схема алгоритму.

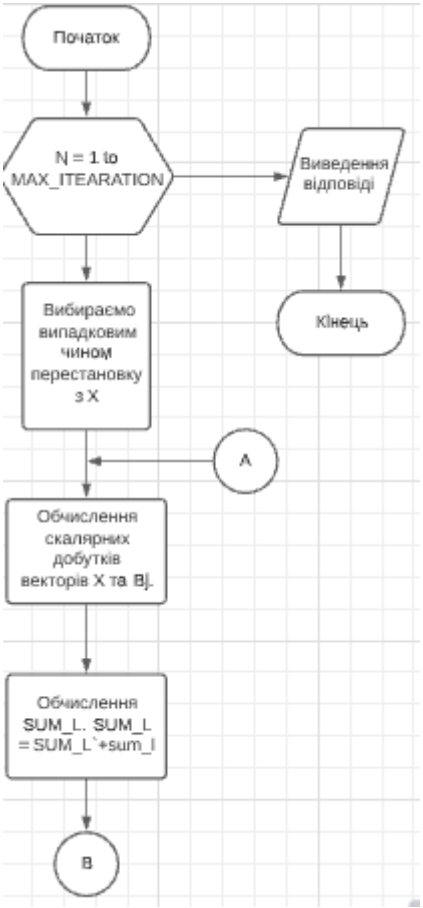

Рис.3.1 – Блок-схема алгоритму.

Рис.3.2 – Перше продовження блок-схеми алгоритму.

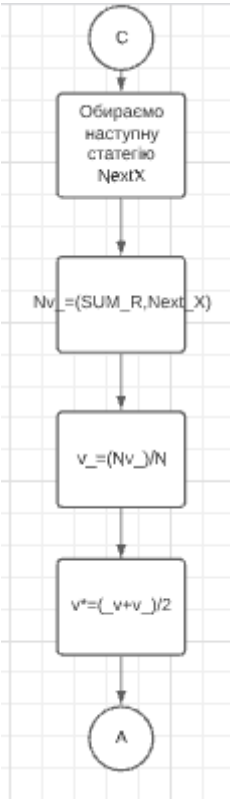

Рис.3.3 – Друге продовження блок-схеми алгоритму.

4.ПРАКТИЧНА ЧАСТИНА

4.1 Опис програми

Програма була розроблена у середовищі Code::Blocks 13.12 з використанням мови програмування C++.

На рис. 4.1-4.10 зображений частково код програми (лише найголовніші моменти). Для зручності код написаний в стилі процедурного програмування (тобто код поділений при можливості на процедури, для зручності в тестуванні та програмуванні самої програми).

Процедури createA та createB (рис.4.2) будують платіжні матриці відповідно до гравців (для другого гравця, стартова платіжна матриця – транспонована).

Процедура InputData(рис.4.3) реалізує зчитування вхідних даних з файлу.

Одним з найскладніших моментів при побудові програми, було вивести читабельно саму відповідь. Для цього була створена процедура writeLineTable яка будуємо порядково (ітерація за ітерацією) нашу вихідну таблицю. Дана процедура представлена на рисунках 4.4 – 4.7.

На рисунках 4.8-4.10 зображена сама процедура Algorithm, котра реалізує ІМТБР.

```
#include<bits/stdc++.h>

#define int long long

using namespace std;

const int MAX_N = 8,
          MAX_M = 8,
          MAX_FACTORIAL = 2e6,
          MAXX_ITERATION = 10000;
int MAX_ITERATION;

double a[MAX_N][MAX_M];
double P_X[MAX_N];
double P_Y[MAX_M];
double eps = 1e-9;
int N = 1;
int n;//row
int m;//column
vector<double> E1[MAX_FACTORIAL];
vector<double> E2[MAX_FACTORIAL];
double B_X[MAX_N][MAX_N];//B*X
double A_Y[MAX_M][MAX_M];//A*Y
int cntE1;
int cntE2;
vector<double> X;//now permutation player 1
vector<double> Y;//now permutation player 2
vector<double> NextX;//next permutation player 1
vector<double> NextY;//next permutation player 2
double sum_l[MAX_N];
double sum_r[MAX_N];
double SUM_L[MAXX_ITERATION];
double SUM_R[MAXX_ITERATION];
double B[MAX_N][MAX_N];
double A[MAX_N][MAX_N];
int used1[MAX_N],used2[MAX_N];
```

Рис.1 – Об’явлення змінних

```
double Q[MAX_N];
double N_v;
double _v;
double _vMin = 2e9;
double Nv_;
double v_;
double v_Max = -2e9;
double v[MAXX_ITERATION];
map<vector<double>,int> mp[10],used_1,used_
int Constant = 6;
int formula;

inline int MyRand(int n){
    return(rand() % n) + 1;
}

void createA(){
    for(int i = 1; i <= n; ++i){
        for(int j = 1; j <= m; ++j){
            A[i][j] = a[i][j];
        }
    }
}

void createB(){
    for(int i = 1; i <= m; ++i){
        for(int j = 1; j <= n; ++j)
            B[i][j] = a[j][i];
    }
}
```

Рис4.2 - Об’явлення змінних та процедури createA та createB

```
void inputData(){
    cin >> n >> m;
    for(int i = 1; i <= n; ++i){
        for(int j = 1; j <= m; ++j){
            cin >> a[i][j];
        }
    }

    for(int i = 1; i <= n; ++i){
        cin >> P_X[i];
    }

    for(int i = 1; i <= m; ++i){
        cin >> P_Y[i];
    }

    cin >> MAX_ITERATION;

    createA();
    createB();

}
```

Рис.4.3 – процедура InputData

```
void writeLineTable(int N){
    cout << "|  " << N;
    int del = 3;
    if(N > 9)++del;
    if(N > 99)++del;
    writeCntSpace(Constant - del);
    for(int i = 1; i <= max(n, m); ++i){
        if(i > 1){
          cout << "|";
          writeCntSpace(Constant);
        }
        if(i <= n){
            cout << setw(Constant) << X[i - 1];
            cout << '|';
            for(int j = 1; j <= m; ++j){
                cout << setw(Constant) << B[j][i];
                cout << '|';

                cout << setw(Constant) << B_X[j][i];
                cout << '|';
            }
        }else{
            for(int j = 1; j <= 2*m+1; ++j){
                 cout << setw(Constant + 1) << '|';
            }
        }
        writeCntSpace(Constant);
        writeCntSpace(Constant);
        if(i <= m){
            cout << setw(Constant) << Y[i - 1];
            cout << '|';
            for(int j = 1; j <= n; ++j){
                cout << setw(Constant) << A[j][i];
                cout << '|';
```

Рис.4.4 – Процедура writeLineTable

```
                cout << setw(Constant) << A_Y[j][i];
                cout << '|';
            }
        }else{
            for(int j = 1; j <= 2*n+1; ++j){
                 cout << setw(Constant + 1) << '|';
            }
        }
         cout << '\n';
    }
//  cout << endl;

    //sum_l & sum_r
    cout << '|';
    cout << setw(7) << '|';
    cout << "sum_l |";
    for(int i = 1; i <= m; ++i){
        cout << setw(Constant + 1) << '|';
        cout << setw(Constant) << sum_l[i] << '|';
    }
    cout << setw(Constant + 1) << '|';
    cout << setw(Constant + 1) << '|';
    cout << "sum_r |";
    for(int i = 1; i <= n; ++i){
        cout << setw(Constant + 1) << '|';
        cout << setw(Constant) << sum_r[i] << '|';
    }
    cout << endl;
```

Рис.4.5 – процедура WriteLineTable

```
//SUM_L & SUM_R
cout << '|';
cout << setw(7) << '|';
cout << "SUM_L |";
for(int i = 1; i <= m; ++i){
    cout << setw(Constant + 1) << '|';
    cout << setw(Constant) << SUM_L[i] << '|';
}
cout << setw(Constant + 1) << '|';
cout << setw(Constant + 1) << '|';
cout << "SUM_R |";
for(int i = 1; i <= n; ++i){
    cout << setw(Constant + 1) << '|';
    cout << setw(Constant) << SUM_R[i] << '|';
}
cout << endl;

//Last elements add in table(NextY,N_v,_v,NextX,Nv_,v_,v*)
cout << '|';
cout << setw(7) << '|';
cout << "NextY |";
for(int i = 1; i <= m; ++i){
    cout << setw(Constant + 1) << '|';
    cout << setw(Constant) << NextY[i - 1] << '|';
}

cout << setw(Constant) << N_v << '|';
cout << setw(Constant) << _v << '|';
cout << "NextX |";
for(int i = 1; i <= n; ++i){
    cout << setw(Constant + 1) << '|';
    cout << setw(Constant) << NextX[i - 1] << '|';
}
```

Рис.4.6 – процедура WriteLineTable

```
cout << setw(Constant) << Nv_ << '|';
cout << setw(Constant) << v_ << '|';
cout << setw(Constant) << v[N] << '|';

cout << endl;


writeCntEmphasis(formula);
```

Рис.4.7 – процедура WriteLineTable

```
void Algorithm(){
    writeShapka();
    formula = (1+Constant)*(8 + 2 * (n + m)) + 1;
    writeCntEmphasis(formula);
    buildE1();
    buildE2();
    for(int N = 1; N <= MAX_ITERATION; ++N){
           if(N == 1){
                int ind = MyRand(cntE1);
                ind = 1;
                X = E1[ind];
            } else{
                X = NextX;
            }
            hashing(X, 1);
          for(int i = 1; i <= m; ++i){
                sum_l[i] = 0;
                for(int j = 1; j <= n; ++j){
                    B_X[i][j] = B[i][j] * X[j - 1];
                    sum_l[i] += B_X[i][j];
                    SUM_L[i] += B_X[i][j];
                }
            }
            for(int i = 1; i <= m; ++i){
                double mn = 2e9;
                int ind = 0;
                for(int j = 1;j <= m; ++j){
                    if(SUM_L[j] < mn && used1[j] < N){
                        mn = SUM_L[j];
                        ind = j;
                    }
                }
                Q[ind] = P_Y[i];
                used1[ind] = N;
            }
```

Рис. 4.8 – процедура Algorithm

```
        NextY.clear();
        for(int i = 1; i <= m; ++i)
            NextY.push_back(Q[i]);
    Y = NextY;
    hashing(Y, 2);
    N_v = 0.0;
    for(int i = 1; i <= m; ++i){
        N_v += SUM_L[i] * Y[i - 1];
    }
    _v = N_v / N;
    _vMin = min(_v, _vMin);
         for(int i = 1; i <= n; ++i){
            sum_r[i] = 0;
            for(int j = 1; j <= m; ++j){
                A_Y[i][j] = A[i][j] * Y[j - 1];
                sum_r[i] += A_Y[i][j];
                SUM_R[i] += A_Y[i][j];
            }
        }
        for(int i = 1; i <= n; ++i){
            double mx = -1;
            int ind = 0;
            for(int j = 1;j <= n; ++j){
                if(SUM_R[j] > mx && used2[j] < N){
                    mx = SUM_R[j];
                    ind = j;
                }
            }
            Q[ind] = P_X[i];
            used2[ind] = N;
        }

        NextX.clear();
        for(int i = 1; i <= n; ++i)
            NextX.push_back(Q[i]);
```

Рис.4.9 – процедура Algorithm

```
        Nv_ = 0.0;
        for(int i = 1; i <= n; ++i){
            Nv_ += SUM_R[i] * NextX[i - 1];
        }
        v_ = Nv_ / N;
        v_Max = max(v_Max, v_);
        v[N] = (v_ + _v) / 2;
        writeLineTable(N);
    }
```

Рис.4.10 – процедура Algorithm

4.2 Необхідна користувачу програми інструкція

Робота програми показана на рисунках 4.11-4.17

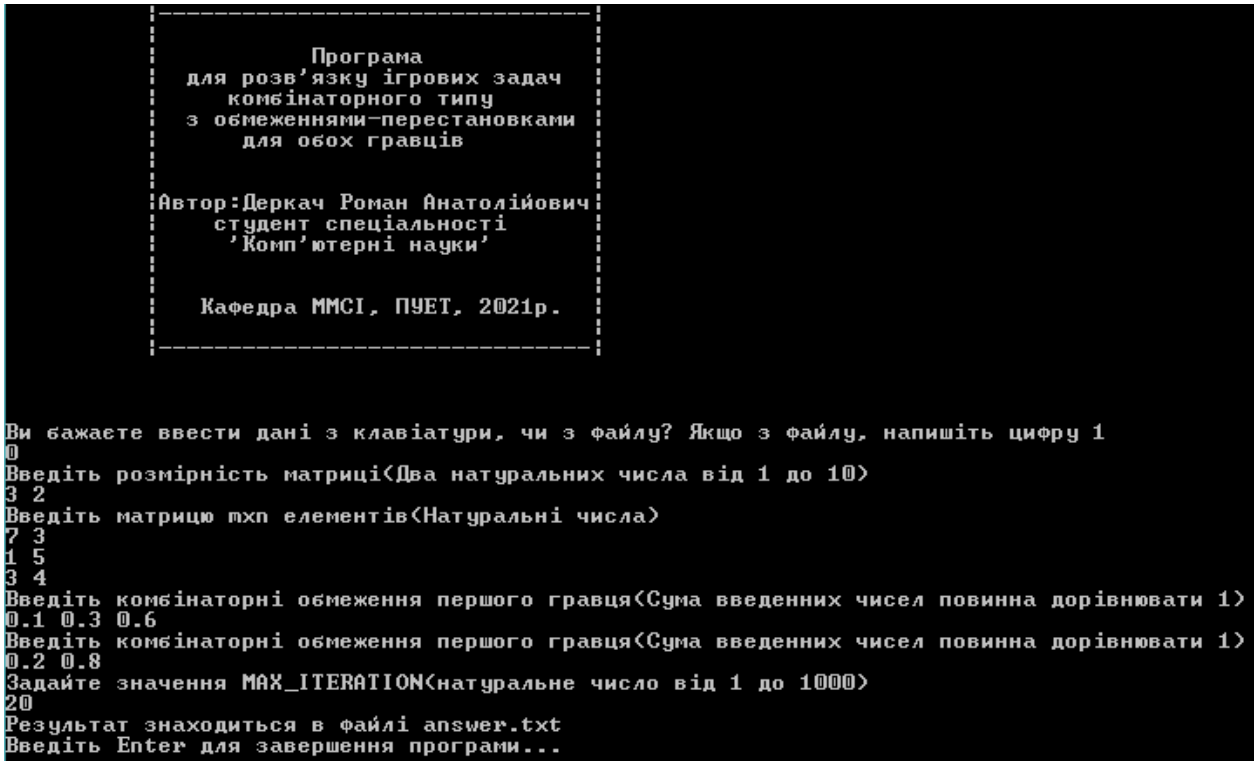

Рис.4.11 – робота програми на прикладі з джерела[1]

Програма має інтерактивне спілкування з користувачем. Перше питання «Ви бажаєте ввести дані з клавіатури, чи з файлу? Якщо з файлу напишіть цифру 1». Воно для того, щоб зрозуміти програмі звідки брати вхідні дані.

Якщо ми обираємо ввід даних з клавіатури, то наступною дією ми повинні ввести вимірність матриці(для зручності користувачеві вказано обмеження на числа). Наступним потрібно ввести саму матрицю. Потім – мультимножини ймовірностей для обмеженнь першого гравця та другого гравця відповідно.

Останнє, що просить ввести програма, це значення MAX_ITERATION – максимальну кількість ітерацій методу. Щоб задати вхідні дані з файлу, потрібно в такій же послідовності задавати дані, як у прикладі з введенням даних з клавіатури.

```
|  N   |   X  |B(1)  |B(1)X |B(2)  |B(2)X |N_v   |v_    |   Y  |A(1)  |A(1)Y |A(2)  |A(2)Y |A(3)  |A(3)Y |Nv_   |v_    |v*    |
-------------------------------------------------------------------------------------------------------------------------------------
|  1   | 0.100| 7.000| 0.700| 3.000| 0.300|      |      | 0.200| 7.000| 1.400| 1.000| 0.200| 3.000| 0.600|
|      | 0.300| 1.000| 0.300| 5.000| 1.500|      |      | 0.800| 3.000| 2.400| 5.000| 4.000| 4.000| 3.200|
|      | 0.600| 3.000| 1.800| 4.000| 2.400|      |      |      |      |      |      |      |      |      |
|      |sum_l |      | 2.800|      | 4.200|      |      |sum_r |      | 3.800|      | 4.200|      | 3.800|
|      |SUM_L |      | 2.800|      | 4.200|      |      |SUM_R |      | 3.800|      | 4.200|      | 3.800|
|      |NextY |      | 0.200|      | 0.800| 3.920| 3.920|NextX |      | 0.300|      | 0.100|      | 0.600| 3.840| 3.840| 3.880|
-------------------------------------------------------------------------------------------------------------------------------------
|  2   | 0.300| 7.000| 2.100| 3.000| 0.900|      |      | 0.200| 7.000| 1.400| 1.000| 0.200| 3.000| 0.600|
|      | 0.100| 1.000| 0.100| 5.000| 0.500|      |      | 0.800| 3.000| 2.400| 5.000| 4.000| 4.000| 3.200|
|      | 0.600| 3.000| 1.800| 4.000| 2.400|      |      |      |      |      |      |      |      |      |
|      |sum_l |      | 4.000|      | 3.800|      |      |sum_r |      | 3.800|      | 4.200|      | 3.800|
|      |SUM_L |      | 6.800|      | 8.000|      |      |SUM_R |      | 7.600|      | 8.400|      | 7.600|
|      |NextY |      | 0.200|      | 0.800| 7.760| 3.880|NextX |      | 0.300|      | 0.100|      | 0.600| 7.680| 3.840| 3.860|
-------------------------------------------------------------------------------------------------------------------------------------
|  20  | 0.300| 7.000| 2.100| 3.000| 0.900|      |      | 0.200| 7.000| 1.400| 1.000| 0.200| 3.000| 0.600|
|      | 0.100| 1.000| 0.100| 5.000| 0.500|      |      | 0.800| 3.000| 2.400| 5.000| 4.000| 4.000| 3.200|
|      | 0.600| 3.000| 1.800| 4.000| 2.400|      |      |      |      |      |      |      |      |      |
|      |sum_l |      | 4.000|      | 3.800|      |      |sum_r |      | 3.800|      | 4.200|      | 3.800|
|      |SUM_L |      |72.800|      |78.500|      |      |SUM_R |      |78.400|      |81.600|      |75.400|
|      |NextY |      | 0.200|      | 0.800|77.360| 3.868|NextX |      | 0.300|      | 0.100|      | 0.600|76.920| 3.846| 3.857|
-------------------------------------------------------------------------------------------------------------------------------------
V*(Кінцеве) = 3.853
Частота(P)
permutation = (0.100, 0.300, 0.600) = 0.350
permutation = (0.100, 0.600, 0.300) = 0.000
permutation = (0.300, 0.100, 0.600) = 0.600
permutation = (0.300, 0.600, 0.100) = 0.000
permutation = (0.600, 0.100, 0.300) = 0.050
permutation = (0.600, 0.300, 0.100) = 0.000

Частота(Q)
permutation = (0.200, 0.800) = 0.950
permutation = (0.800, 0.200) = 0.050
```

Рис.4.12 – результат роботи(1,2 та 20 ітерації) програми на прикладі з джерела[1], котрий був введений з клавіатури

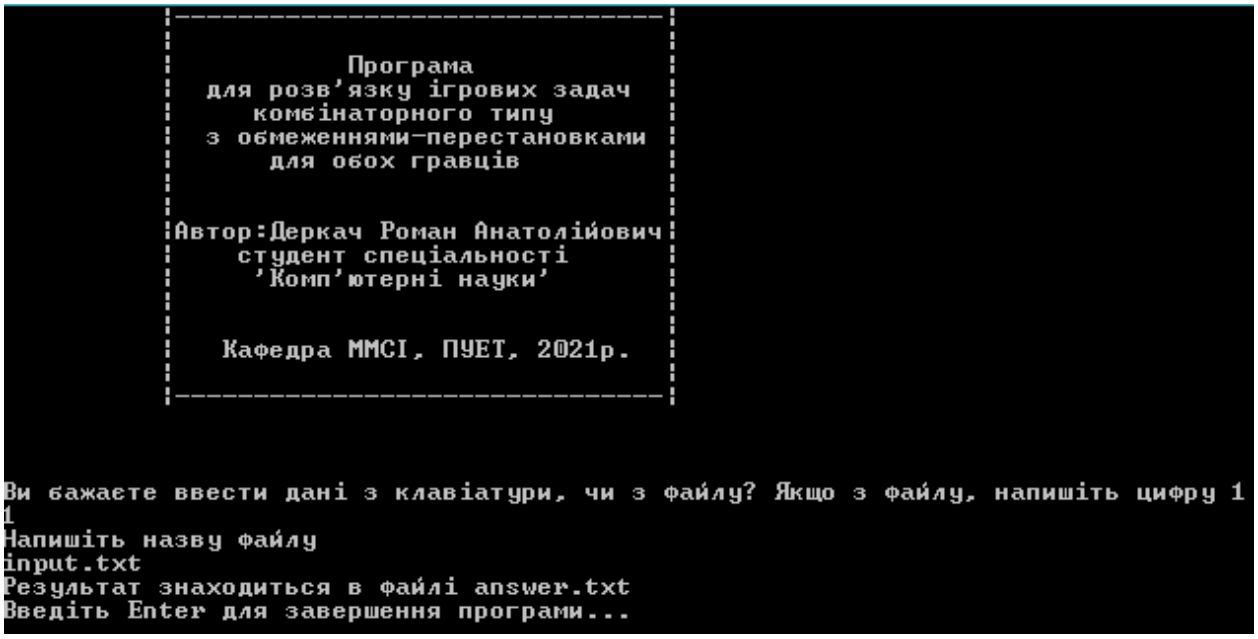

Рис.4.13 – Введення даних(прикладу 2) за допомогою файлу

```
5 5
9 10 1 5 12
1 2 12 8 5
10 3 2 11 1
10 3 4 11 11
2 4 1 7 1
0.17 0.13 0.11 0.31 0.28
0.85 0.12 0 0 0.03
93
```

Рис.4.14 – Вміст файлу input.txt(приклад 2)

```
|  91  | 0.310| 9.000| 2.790|10.000| 3.100| 1.000| 0.310| 5.000| 1.550|12.000| 3.720|      |      | 0.030| 9.000| 0.270| 1.000| 0.030|10.000| 0.300|10.000|
|      | 0.170| 1.000| 0.170| 2.000| 0.340|12.000| 2.040| 8.000| 1.360| 5.000| 0.850|      |      | 0.000|10.000| 0.000| 2.000| 0.000| 3.000| 0.000| 3.000|
|      | 0.130|10.000| 1.300| 3.000| 0.390| 2.000| 0.260|11.000| 1.430| 1.000| 0.130|      |      | 0.000| 1.000| 0.000|12.000| 0.000| 2.000| 0.000| 4.000|
|      | 0.110|10.000| 1.100| 3.000| 0.330| 4.000| 0.440|11.000| 1.210|11.000| 1.210|      |      | 0.850| 5.000| 4.250| 8.000| 6.800|11.000| 9.350|11.000|
|      | 0.280| 2.000| 0.560| 4.000| 1.120| 1.000| 0.280| 7.000| 1.960| 1.000| 0.280|      |      | 0.120|12.000| 1.440| 5.000| 0.600| 1.000| 0.120|11.000|
|      |sum_l |      | 5.920|      | 5.280|      | 3.330|      | 7.510|      | 6.190|      |      |sum_r |      | 5.960|      | 7.430|      | 9.770|      |1
|      |SUM_L |      |539.040|      |479.150|      |303.140|      |684.610|      |562.630|      |      |SUM_R |      |541.550|      |675.050|      |891.500|
|      |NextY |      | 0.030|      | 0.000|      | 0.000|      | 0.850|      | 0.120|665.605| 7.314|NextX |      | 0.310|      | 0.170|      | 0.130|      |
-----------------------------------------------------------------------------------------------------------------------------------------------------------
|  92  | 0.310| 9.000| 2.790|10.000| 3.100| 1.000| 0.310| 5.000| 1.550|12.000| 3.720|      |      | 0.030| 9.000| 0.270| 1.000| 0.030|10.000| 0.300|10.000|
|      | 0.170| 1.000| 0.170| 2.000| 0.340|12.000| 2.040| 8.000| 1.360| 5.000| 0.850|      |      | 0.000|10.000| 0.000| 2.000| 0.000| 3.000| 0.000| 3.000|
|      | 0.130|10.000| 1.300| 3.000| 0.390| 2.000| 0.260|11.000| 1.430| 1.000| 0.130|      |      | 0.000| 1.000| 0.000|12.000| 0.000| 2.000| 0.000| 4.000|
|      | 0.110|10.000| 1.100| 3.000| 0.330| 4.000| 0.440|11.000| 1.210|11.000| 1.210|      |      | 0.850| 5.000| 4.250| 8.000| 6.800|11.000| 9.350|11.000|
|      | 0.280| 2.000| 0.560| 4.000| 1.120| 1.000| 0.280| 7.000| 1.960| 1.000| 0.280|      |      | 0.120|12.000| 1.440| 5.000| 0.600| 1.000| 0.120|11.000|
|      |sum_l |      | 5.920|      | 5.280|      | 3.330|      | 7.510|      | 6.190|      |      |sum_r |      | 5.960|      | 7.430|      | 9.770|      |1
|      |SUM_L |      |544.960|      |484.430|      |306.470|      |692.120|      |568.820|      |      |SUM_R |      |547.510|      |682.480|      |901.270|
|      |NextY |      | 0.030|      | 0.000|      | 0.000|      | 0.850|      | 0.120|672.909| 7.314|NextX |      | 0.310|      | 0.170|      | 0.130|      |
-----------------------------------------------------------------------------------------------------------------------------------------------------------
|  93  | 0.310| 9.000| 2.790|10.000| 3.100| 1.000| 0.310| 5.000| 1.550|12.000| 3.720|      |      | 0.030| 9.000| 0.270| 1.000| 0.030|10.000| 0.300|10.000|
|      | 0.170| 1.000| 0.170| 2.000| 0.340|12.000| 2.040| 8.000| 1.360| 5.000| 0.850|      |      | 0.000|10.000| 0.000| 2.000| 0.000| 3.000| 0.000| 3.000|
|      | 0.130|10.000| 1.300| 3.000| 0.390| 2.000| 0.260|11.000| 1.430| 1.000| 0.130|      |      | 0.000| 1.000| 0.000|12.000| 0.000| 2.000| 0.000| 4.000|
|      | 0.110|10.000| 1.100| 3.000| 0.330| 4.000| 0.440|11.000| 1.210|11.000| 1.210|      |      | 0.850| 5.000| 4.250| 8.000| 6.800|11.000| 9.350|11.000|
|      | 0.280| 2.000| 0.560| 4.000| 1.120| 1.000| 0.280| 7.000| 1.960| 1.000| 0.280|      |      | 0.120|12.000| 1.440| 5.000| 0.600| 1.000| 0.120|11.000|
|      |sum_l |      | 5.920|      | 5.280|      | 3.330|      | 7.510|      | 6.190|      |      |sum_r |      | 5.960|      | 7.430|      | 9.770|      |1
|      |SUM_L |      |550.880|      |489.710|      |309.800|      |699.630|      |575.010|      |      |SUM_R |      |553.470|      |689.910|      |911.040|
|      |NextY |      | 0.030|      | 0.000|      | 0.000|      | 0.850|      | 0.120|680.213| 7.314|NextX |      | 0.310|      | 0.170|      | 0.130|      |
-----------------------------------------------------------------------------------------------------------------------------------------------------------
V*(Кінцеве) = 7.309
Частота(P)
permutation = (0.110, 0.130, 0.170, 0.280, 0.310) = 0.011
permutation = (0.310, 0.170, 0.130, 0.110, 0.280) = 0.989

Частота(Q)
permutation = (0.030, 0.000, 0.000, 0.850, 0.120) = 0.968
permutation = (0.120, 0.000, 0.000, 0.850, 0.030) = 0.032
```

Рис.4.15 – Вміст файлу answer.txt - результат для прикладу 2

При великій кількості обмежень-перестановок для гравців, для зручності у файлі-відповіді будемо виводити частоти лише для тих, котрі не дорівнюють нулю (тобто лише для тих, котрі зустрічалися хоча б один раз).

```
7 10
9 10 4 6 1 10 6 11 3 1
10 11 6 6 1 8 7 12 2 12
5 7 11 9 2 4 1 11 7 3
6 5 10 12 11 9 10 1 11 12
6 12 5 10 6 8 1 1 2 8
6 12 4 11 2 10 12 5 7 3
2 3 8 10 8 5 3 8 1 8
0.48 0.13 0.26 0.07 0.04 0 0.02
0.93 0.05 0.01 0 0 0 0 0 0 0.01
78
```

Рис.4.16 – Вміст файлу input.txt(приклад 3)

```
--------------------------------------------------------------------------------------------------
V*(Кінцеве) = 8.128
Частота(P)
permutation = (0.000, 0.020, 0.040, 0.070, 0.130, 0.260, 0.480) = 0.013
permutation = (0.070, 0.040, 0.020, 0.480, 0.130, 0.000, 0.260) = 0.038
permutation = (0.130, 0.040, 0.020, 0.070, 0.260, 0.000, 0.480) = 0.013
permutation = (0.130, 0.040, 0.020, 0.260, 0.070, 0.000, 0.480) = 0.026
permutation = (0.130, 0.040, 0.020, 0.260, 0.480, 0.000, 0.070) = 0.026
permutation = (0.130, 0.040, 0.020, 0.480, 0.070, 0.000, 0.260) = 0.026
permutation = (0.260, 0.040, 0.020, 0.070, 0.130, 0.000, 0.480) = 0.026
permutation = (0.260, 0.040, 0.020, 0.070, 0.480, 0.000, 0.130) = 0.026
permutation = (0.260, 0.040, 0.020, 0.480, 0.070, 0.000, 0.130) = 0.013
permutation = (0.260, 0.070, 0.020, 0.130, 0.040, 0.000, 0.480) = 0.026
permutation = (0.260, 0.070, 0.040, 0.130, 0.020, 0.000, 0.480) = 0.013
permutation = (0.260, 0.130, 0.000, 0.070, 0.480, 0.040, 0.020) = 0.013
permutation = (0.260, 0.130, 0.020, 0.040, 0.070, 0.000, 0.480) = 0.026
permutation = (0.260, 0.130, 0.020, 0.070, 0.040, 0.000, 0.480) = 0.051
permutation = (0.260, 0.130, 0.040, 0.070, 0.020, 0.000, 0.480) = 0.026
permutation = (0.260, 0.130, 0.070, 0.040, 0.020, 0.000, 0.480) = 0.013
permutation = (0.480, 0.040, 0.020, 0.070, 0.130, 0.000, 0.260) = 0.026
permutation = (0.480, 0.040, 0.020, 0.130, 0.070, 0.000, 0.260) = 0.013
permutation = (0.480, 0.040, 0.020, 0.130, 0.260, 0.000, 0.070) = 0.013
permutation = (0.480, 0.040, 0.020, 0.260, 0.130, 0.000, 0.070) = 0.013
permutation = (0.480, 0.070, 0.020, 0.040, 0.130, 0.000, 0.260) = 0.038
permutation = (0.480, 0.070, 0.020, 0.040, 0.260, 0.000, 0.130) = 0.026
permutation = (0.480, 0.070, 0.020, 0.130, 0.040, 0.000, 0.260) = 0.051
permutation = (0.480, 0.130, 0.020, 0.040, 0.070, 0.000, 0.260) = 0.051
permutation = (0.480, 0.130, 0.020, 0.040, 0.260, 0.000, 0.070) = 0.051
permutation = (0.480, 0.130, 0.020, 0.070, 0.040, 0.000, 0.260) = 0.026
permutation = (0.480, 0.130, 0.020, 0.070, 0.260, 0.000, 0.040) = 0.013
permutation = (0.480, 0.130, 0.070, 0.040, 0.020, 0.000, 0.260) = 0.013
permutation = (0.480, 0.260, 0.020, 0.040, 0.070, 0.000, 0.130) = 0.167
permutation = (0.480, 0.260, 0.020, 0.040, 0.130, 0.000, 0.070) = 0.051
permutation = (0.480, 0.260, 0.020, 0.070, 0.040, 0.000, 0.130) = 0.026
permutation = (0.480, 0.260, 0.040, 0.020, 0.070, 0.000, 0.130) = 0.013
permutation = (0.480, 0.260, 0.070, 0.040, 0.020, 0.000, 0.130) = 0.013
permutation = (0.480, 0.260, 0.130, 0.000, 0.040, 0.020, 0.070) = 0.026
```

Рис.4.17 – результат програми для прикладу 3

```
Частота(Q)
permutation = (0.000, 0.010, 0.000, 0.010, 0.000, 0.050, 0.000, 0.930, 0.000, 0.000) = 0.038
permutation = (0.000, 0.010, 0.000, 0.050, 0.000, 0.010, 0.000, 0.930, 0.000, 0.000) = 0.128
permutation = (0.000, 0.010, 0.000, 0.050, 0.000, 0.930, 0.000, 0.010, 0.000, 0.000) = 0.013
permutation = (0.000, 0.010, 0.000, 0.930, 0.000, 0.010, 0.000, 0.050, 0.000, 0.000) = 0.282
permutation = (0.000, 0.010, 0.000, 0.930, 0.000, 0.050, 0.000, 0.000, 0.000, 0.010) = 0.013
permutation = (0.000, 0.010, 0.000, 0.930, 0.000, 0.050, 0.000, 0.010, 0.000, 0.000) = 0.244
permutation = (0.000, 0.050, 0.000, 0.010, 0.000, 0.010, 0.000, 0.930, 0.000, 0.000) = 0.064
permutation = (0.000, 0.050, 0.000, 0.010, 0.000, 0.930, 0.000, 0.010, 0.000, 0.000) = 0.038
permutation = (0.000, 0.050, 0.000, 0.930, 0.000, 0.010, 0.000, 0.010, 0.000, 0.000) = 0.064
permutation = (0.000, 0.930, 0.000, 0.010, 0.000, 0.010, 0.000, 0.050, 0.000, 0.000) = 0.013
permutation = (0.000, 0.930, 0.000, 0.010, 0.000, 0.050, 0.000, 0.010, 0.000, 0.000) = 0.064
permutation = (0.000, 0.930, 0.000, 0.050, 0.000, 0.010, 0.000, 0.010, 0.000, 0.000) = 0.038
```

Рис.4.18 – результат програми для прикладу 3

ВИСНОВКИ

В бакалаврській роботі досліджено тему «Ігрові задачі комбінаторного типу» та побудована програма, котра розв’язує ігрові задачі комбінаторного типу з обмеженнями-перестановками для обох гравців. Основним теоретичним матеріалом була інформація з робіт, багатьох з яких співавтором є Ємець О.О.

Було здійснено огляд робіт, схожих за тематикою до теми бакалаврської роботи. На базі прикладу з [1] побудований та протестований зроблений консольний додаток. Також було побудовано два додаткові приклади, для перевірки швидкості роботи програми. Виявлено, що для прикладу №3, швидкість програми сповільнюється (при зростанні обмежень на стратегії гравців швидкість програми падає). Також створено блок-схему алгоритму ІМТБР.

Програма є працюючою та була зроблена перевірка на відсутність помилок. В пояснюючій записці описано, як створювалась програма, та як вона працює.

Програма передана ПУЕТ для використанні в дистанційному курсі «Елементи комбінаторної оптимізації» кафедри ММСІ для спеціальності «Комп’ютерні науки». Результати опубліковані в тезах семінару КНіПМ [5].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1.Ольховська О.В. Комбінаторні задачі ігрового типу на множині розміщень/ О.В. Ольховська .– Дис. ... канд. фіз.-мат. наук (01.05.01). – Полтава: ПУЕТ, 2014.

2. Ємець О. О. Розв’язування комбінаторних задач ігрового типу з обмеженнями-переставленнями у обох гравців: ітераційний метод / О. О. Ємець, О. В. Ольховська // Системні дослідження та інформаційні технології. – 2012. – №4. – С. 80-93.

3. Ємець О.О. Програмний комплекс, що реалізує методи розв’язування задач комбінаторної оптимізації ігрового типу / О.О. Ємець, Д. М. Ольховський, О. В. Ольховська // Вісник Черкаського університету. Серія Прикладна математика. Інформатика. – № 18 (351). – 2015. – С. 96-101.

4. Ємець О.О. Теоретична оцінка складності алгоритмів розв’язування задач комбінаторної оптимізації ігрового типу / О.О. Ємець, Д. М. Ольховський, О. В. Ольховська // Вісник Черкаського університету. Серія Прикладна математика. Інформатика. – № 18 (351). – 2015. – С. 11-18.

5.Деркач Р.А. «Ігрові задачі комбінаторного типу: програмне забезпечення і дослідження» / Р.А.Деркач, О.О.Ємець // Комп’ютернінауки і прикладна математика (КНіПМ-2021): матеріали наук.-практ. семінару. Випуск 6. / За ред. Ємця О. О. – Полтава: Кафедра ММСІ ПУЕТ, 2021. –3 с.– Режим доступу: http://dspace.puet.edu.ua/handle/123456789/10566

6. Емец О. А. Исследование задач комбинаторной оптимизации игрового типа на размещениях / О. А. Емец, Н. Ю. Устьян // Проблемы управления и информатики. – 2007. – № 1. – С. 26-36.

7. Емец О.А. Исследование математических моделей и методов решения задач на перестановках игрового типа / О.А. Емец, Н.Ю. Устьян // Кибернетика и сист. анализ. — 2007. — № 6. — С. 103–114

8. Ємець О. О. Монотонний ітераційний метод для розв'язування задач комбінаторної оптимізації ігрового типу на переставленнях / О. О. Ємець, О. В. Ольховська // Доповіді Національної академії наук України – 2014. – №8. – С. 48-52.

9. Ємець О.О. Ігрова комбінаторна модель однієї задачі сільськогосподарського виробництва/ О.О. Ємець, Н.Ю. Устьян // Матеріали VII Міжнародної науково-практичної конференції "Наука і освіта - 2004" (10-25 лютого 2004 року). Том 70. Математика. – Дніпропетровськ: Наука і освіта, 2004. - С.46-49.

10. Ємець О.О. Ігрові комбінаторні задачі на переставленнях / О.О. Ємець, Н.Ю. Устьян // В кн.: Матеріали VIII Міжнародної науково-практичної конференції „Наука і освіта `2005". – Т.22. Математика. – Дніпропетровськ: Наука і освіта, 2005. – С. 26-32.

11. Ємець О.О. Моделювання і розв'язування деяких ігрових задач комбінаторної оптимізації економічного змісту / О.О. Ємець, Н.Ю. Устьян // Економіка: проблеми теорії та практики. Зб. наук. праць. Вип. 207, т. 1. – Дніпропетровськ: ДНУ, 2005. - С. 82-99.

12. Емец О. А. Задачи на перестановках игрового типа / О.А. Емец, Н.Ю. Устьян // В кн.: Материалы XIV Международной конференции «Проблемы теоретической кибернетики» (23-28 мая 2005 г., Пенза) – М.: Изд-во мех.-метем. ф-та МГУ, 2005.– С. 46.

13. Емец О. А. Решение некоторых задач комбинаторной оптимизации на размещениях и перестановках игрового типа / О.А. Емец, Н.Ю. Устьян // Проблемы управления и информатики. – 2006. – №3. С.37–47.

14. Емец О. А. Исследование задач комбинаторной оптимизации игрового типа на размещениях / О.А. Емец, Н.Ю. Устьян // Проблемы управления и информатики. – 2007 – №1. – С. 26-36.

15. Ємець О. О. Методичні рекомендації до виконання бакалаврської роботи для студентів спеціальності 122 «Комп`ютерні науки та інформаційні технології» / О. О. Ємець. – Полтава: ПУЕТ, 2018.

ДОДАТОК А. Код програми

```
#include<bits/stdc++.h>


#define int long long


using namespace std;


const int MAX_N = 11,
        MAX_M = 11,
        MAX_FACTORIAL = 3628801,
        MAXX_ITERATION = 10000;
int MAX_ITERATION;


double a[MAX_N][MAX_M];
double P_X[MAX_N];
double P_Y[MAX_M];
double eps = 1e-9;
int N = 1;
int n;//row
int m;//column
vector<double> E1[MAX_FACTORIAL];
vector<double> E2[MAX_FACTORIAL];
double B_X[MAX_N][MAX_N];//B*X
```

```
double A_Y[MAX_M][MAX_M];//A*Y
int cntE1;
int cntE2;
vector<double> X;//now permutation player 1
vector<double> Y;//now permutation player 2
vector<double> NextX;//next permutation player 1
vector<double> NextY;//next permutation player 2
double sum_l[MAX_N];
double sum_r[MAX_N];
double SUM_L[MAXX_ITERATION];
double SUM_R[MAXX_ITERATION];
double B[MAX_N][MAX_N];
double A[MAX_N][MAX_N];
int used1[MAX_N],used2[MAX_N];
double Q[MAX_N];
double N_v;
double _v;
double _vMin = 2e9;
double Nv_;
double v_;
double v_Max = -2e9;
double v[MAXX_ITERATION];
```

```
map<vector<double>,int> mp[10],used_1,used_2;
int Constant = 6;
int formula;

inline int MyRand(int n){
   return(rand() % n) + 1;
}

void createA(){
   for(int i = 1; i <= n; ++i){
      for(int j = 1; j <= m; ++j){
         A[i][j] = a[i][j];
      }
   }
}

void createB(){
   for(int i = 1; i <= m; ++i){
      for(int j = 1; j <= n; ++j)
         B[i][j] = a[j][i];
   }
}
```

```
void inputData(){
    cin >> n >> m;
    for(int i = 1; i <= n; ++i){
        for(int j = 1; j <= m; ++j){
            cin >> a[i][j];
        }
    }

    for(int i = 1; i <= n; ++i){
        cin >> P_X[i];
    }

    for(int i = 1; i <= m; ++i){
        cin >> P_Y[i];
    }

    cin >> MAX_ITERATION;

    createA();
    createB();
```

```
}

void buildE1(){
    int fac = 1;
    for(int i = 1; i <= n; ++i){
        fac *= i;
    }
    sort(P_X + 1, P_X + n + 1);
    while(fac-->0){
        ++cntE1;
        for(int i = 1; i <= n; ++i){
            E1[cntE1].push_back(P_X[i]);
        }
        next_permutation(P_X + 1,P_X + n + 1);
    }

    //cout << P_X[3] << endl;

}

void writeE1(){
    cout << "count = " << cntE1 << endl;
```

```
    for(int i = 1; i <= cntE1; ++i){
        assert((int)E1[i].size() == n);
        for(int j = 0; j < E1[i].size(); ++j)
            cout << E1[i][j] << ' ';
        cout << endl;
    }

}

void buildE2(){
    int fac = 1;
    for(int i = 1; i <= m; ++i){
        fac *= i;
    }
    sort(P_Y + 1, P_Y + m + 1);

    while(fac-->0){
        ++cntE2;
        for(int i = 1; i <= m; ++i){
            E2[cntE2].push_back(P_Y[i]);
        }
```

```
        next_permutation(P_Y + 1,P_Y + m + 1);
    }
}

void writeE2(){
    cout << "count = " << cntE2 << endl;

    for(int i = 1; i <= cntE2; ++i){
        assert((int)E2[i].size() == m);
        for(int j = 0; j < E2[i].size(); ++j)
            cout << E2[i][j] << ' ';
        cout << endl;
    }

}

void writeLine(){
    cout << '\n';
}

void writeVector(vector<double> &a){
    for(int i = 0; i < a.size(); ++i){
```

```
        cout << a[i] << ' ';
    }
    cout << '\n';
}

inline void hashing(vector<double> &x,int pos){
    ++mp[pos][x]; //used map
}

inline void writeCntSpace(int cntSpace){
    while(cntSpace-->0){
        cout << ' ';
    }
    cout << "|";
}

inline void writeCntEmphasis(int cntEmphasis){
    while(cntEmphasis-->0){
        cout << "-";
    }
    cout << endl;
}
```

```
void writeShapka(){
    cout << "| ";
    cout << " N";
    writeCntSpace(Constant - 3);

    //X
    cout << "   X";
    writeCntSpace(Constant - 4);
    for(int i = 1;i <= m; ++i){
        cout << "B(" << i << ")";
        writeCntSpace(Constant - 4);
        cout << "B(" << i << ")X";
        writeCntSpace(Constant - 5);
    }

    cout << "N_v";
    writeCntSpace(Constant - 3);

    cout << "v_";
    writeCntSpace(Constant - 2);
```

```
//Y
cout << "   Y";

writeCntSpace(Constant - 4);
for(int i = 1;i <= n; ++i){
    cout << "A(" << i << ")";
    writeCntSpace(Constant - 4);
    cout << "A(" << i << ")Y";
    writeCntSpace(Constant - 5);
}

cout << "Nv_";
writeCntSpace(Constant - 3);

cout << "v_";
writeCntSpace(Constant - 2);

cout << "v*";
writeCntSpace(Constant - 2);

cout << endl;
```

```
}

void writeLineTable(int N){
    cout << "|  " << N;
    int del = 3;
    if(N > 9)++del;
    if(N > 99)++del;
    writeCntSpace(Constant - del);
    for(int i = 1; i <= max(n, m); ++i){
        if(i > 1){
          cout << "|";
          writeCntSpace(Constant);
        }
        if(i <= n){
            cout << setw(Constant) << X[i - 1];
            cout << '|';
            for(int j = 1; j <= m; ++j){
                cout << setw(Constant) << B[j][i];
                cout << '|';


                cout << setw(Constant) << B_X[j][i];
                cout << '|';
```

```
    }
}else{
    for(int j = 1; j <= 2*m+1; ++j){
        cout << setw(Constant + 1) << '|';
    }
}
writeCntSpace(Constant);
writeCntSpace(Constant);
if(i <= m){
    cout << setw(Constant) << Y[i - 1];
    cout << '|';
    for(int j = 1; j <= n; ++j){
        cout << setw(Constant) << A[j][i];
        cout << '|';


        cout << setw(Constant) << A_Y[j][i];
        cout << '|';
    }
}else{
    for(int j = 1; j <= 2*n+1; ++j){
        cout << setw(Constant + 1) << '|';
    }
```

```
        }
         cout << '\n';
    }
//  cout << endl;


  //sum_l & sum_r
  cout << '|';
  cout << setw(7) << '|';
  cout << "sum_l |";
  for(int i = 1; i <= m; ++i){
      cout << setw(Constant + 1) << '|';
      cout << setw(Constant) << sum_l[i] << '|';
  }
  cout << setw(Constant + 1) << '|';
  cout << setw(Constant + 1) << '|';
  cout << "sum_r |";
  for(int i = 1; i <= n; ++i){
      cout << setw(Constant + 1) << '|';
      cout << setw(Constant) << sum_r[i] << '|';
  }
  cout << endl;
```

```
//SUM_L & SUM_R
cout << '|';
cout << setw(7) << '|';
cout << "SUM_L |";
for(int i = 1; i <= m; ++i){
    cout << setw(Constant + 1) << '|';
    cout << setw(Constant) << SUM_L[i] << '|';
}
cout << setw(Constant + 1) << '|';
cout << setw(Constant + 1) << '|';
cout << "SUM_R |";
for(int i = 1; i <= n; ++i){
    cout << setw(Constant + 1) << '|';
    cout << setw(Constant) << SUM_R[i] << '|';
}
cout << endl;

//Last elements add in table(NextY,N_v,_v,NextX,Nv_,v_,v*)
cout << '|';
cout << setw(7) << '|';
cout << "NextY |";
for(int i = 1; i <= m; ++i){
```

```
        cout << setw(Constant + 1) << '|';
        cout << setw(Constant) << NextY[i - 1] << '|';
    }

    cout << setw(Constant) << N_v << '|';
    cout << setw(Constant) << _v << '|';
    cout << "NextX |";
    for(int i = 1; i <= n; ++i){
        cout << setw(Constant + 1) << '|';
        cout << setw(Constant) << NextX[i - 1] << '|';
    }

    cout << setw(Constant) << Nv_ << '|';
    cout << setw(Constant) << v_ << '|';
    cout << setw(Constant) << v[N] << '|';

    cout << endl;


    writeCntEmphasis(formula);

}
```

```
void Algorithm(){
    writeShapka();
    formula = (1+Constant)*(8 + 2 * (n + m)) + 1;
    writeCntEmphasis(formula);
    buildE1();
    buildE2();
    for(int N = 1; N <= MAX_ITERATION; ++N){
        if(N == 1){
            int ind = MyRand(cntE1);
            ind = 1;
            X = E1[ind];
        } else{
            X = NextX;
        }
        hashing(X, 1);
      for(int i = 1; i <= m; ++i){
            sum_l[i] = 0;
            for(int j = 1; j <= n; ++j){
                B_X[i][j] = B[i][j] * X[j - 1];
                sum_l[i] += B_X[i][j];
                SUM_L[i] += B_X[i][j];
```

```
        }
    }
    for(int i = 1; i <= m; ++i){
        double mn = 2e9;
        int ind = 0;
        for(int j = 1;j <= m; ++j){
            if(SUM_L[j] < mn && used1[j] < N){
                mn = SUM_L[j];
                ind = j;
            }
        }
        Q[ind] = P_Y[i];
        used1[ind] = N;
    }

    NextY.clear();
    for(int i = 1; i <= m; ++i)
        NextY.push_back(Q[i]);
Y = NextY;
hashing(Y, 2);
N_v = 0.0;
for(int i = 1; i <= m; ++i){
```

```
    N_v += SUM_L[i] * Y[i - 1];
}
_v = N_v / N;
_vMin = min(_v, _vMin);
     for(int i = 1; i <= n; ++i){
        sum_r[i] = 0;
        for(int j = 1; j <= m; ++j){
            A_Y[i][j] = A[i][j] * Y[j - 1];
            sum_r[i] += A_Y[i][j];
            SUM_R[i] += A_Y[i][j];
        }
    }
    for(int i = 1; i <= n; ++i){
        double mx = -1;
        int ind = 0;
        for(int j = 1;j <= n; ++j){
            if(SUM_R[j] > mx && used2[j] < N){
                mx = SUM_R[j];
                ind = j;
            }
        }
        Q[ind] = P_X[i];
```

```
            used2[ind] = N;
        }

        NextX.clear();
        for(int i = 1; i <= n; ++i)
            NextX.push_back(Q[i]);
    Nv_ = 0.0;
    for(int i = 1; i <= n; ++i){
        Nv_ += SUM_R[i] * NextX[i - 1];
    }
    v_ = Nv_ / N;
    v_Max = max(v_Max, v_);
    v[N] = (v_ + _v) / 2;
    writeLineTable(N);
  }

}

void WriteStrategyP(){
  int fac = 1;
  for(int i = 1; i <= n; ++i){
    fac *= i;
```

```
    }
    sort(P_X + 1, P_X + n + 1);
    vector<double> vec;
    while(fac-->0){
        vec.clear();

        for(int i = 1; i <= n; ++i)
        vec.push_back(P_X[i]);

        if(mp[1][vec]!=0 && !used_1[vec]){
            used_1[vec] = 1;
        cout << "permutation = (";
        for(int i = 1; i <= n; ++i){
            cout << P_X[i];
            if(i < n)cout << ", ";
        }

        cout << ") = " << mp[1][vec]*1.0/MAX_ITERATION << '\n';
        }
        next_permutation(P_X + 1,P_X + n + 1);
    }
}
```

```
void WriteStrategyQ(){
    int fac = 1;
    for(int i = 1; i <= m; ++i){
        fac *= i;
    }
    sort(P_X + 1, P_X + m + 1);
    vector<double> vec;
    while(fac-->0){
        vec.clear();

        for(int i = 1; i <= m; ++i)
        vec.push_back(P_Y[i]);

        if(mp[2][vec]!=0 && !used_2[vec]){
        used_2[vec] = 1;
        cout << "permutation = (";
        for(int i = 1; i <= m; ++i){
            cout << P_Y[i];
            if(i < m)cout << ", ";
        }
```

```
        cout << ") = " << mp[2][vec]*1.0 / MAX_ITERATION << '\n';


        }


        next_permutation(P_Y + 1,P_Y + m + 1);
    }
}


void outputData(int x = 0){
    cout << "V*(Кінцеве) = " << (_vMin + v_Max) / 2.0 << '\n';
    cout << "Частота(P)\n";
    WriteStrategyP();
    cout << '\n';
    cout << "Частота(Q)\n";
    WriteStrategyQ();


    if(x == 1){
    cerr << "Результат знаходиться в файлі answer.txt\n";
    cerr << "Введіть Enter для завершення програми...\n";
    int it = 0;
    while(it <= 100000){
    char ch = getchar();
```

```
    if(ch == '\n')break;
    }
    }

    //system("pause");

}

void inputData(string s){
    cerr << "Введіть розмірність матриці(Два натуральних числа від 1 до 10)\n";
    cin >> n >> m;

    cerr << "Введіть матрицю mxn елементів(Натуральні числа)\n";

    for(int i = 1; i <= n; ++i){
        for(int j = 1; j <= m; ++j){
            cin >> a[i][j];
        }
    }

    cerr << "Введіть комбінаторні обмеження першого гравця(Сума введенних чисел повинна дорівнювати 1)\n";
```

```
    for(int i = 1; i <= n; ++i){
        cin >> P_X[i];
    }

    cerr << "Введіть комбінаторні обмеження першого гравця(Сума введенних чисел повинна дорівнювати 1)\n";

    for(int i = 1; i <= m; ++i){
        cin >> P_Y[i];
    }

    cerr << "Задайте значення MAX_ITERATION(натуральне число від 1 до 1000)\n";
    cin >> MAX_ITERATION;

    createA();
    createB();

}

void writeStart(){
    cout << "          |------------------------------|\n";
```

```
    cout << "         |                          |\n";
    cout << "         |          Програма        |\n";
    cout << "         |  для розв'язку ігрових задач  |\n";
    cout << "         |     комбінаторного типу       |\n";
    cout << "         |  з обмеженнями-перестановками |\n";
    cout << "         |      для обох гравців         |\n";
    cout << "         |                          |\n";
    cout << "         |                          |\n";
    cout << "         |Автор:Деркач Роман Анатолійович|\n";
    cout << "         |    студент спеціальності      |\n";
    cout << "         |     'Комп'ютерні науки'       |\n";
    cout << "         |                          |\n";
    cout << "         |                          |\n";
    cout << "         |   Кафедра MMCI, ПУЕТ, 2021р.  |\n";
    cout << "         |                          |\n";
    cout << "         |------------------------------|\n";
    cout << endl;
    cout << endl;
    cout << endl;
}

signed main(){
```

```
    setlocale (LC_CTYPE, "ukr");
    srand(time(NULL));

    //freopen("input.txt","r",stdin);

    writeStart();

    freopen("answer.txt","w",stdout);

    cout.setf(ios::fixed);  // вывод в фиксированном формате
    cout.precision(3);

    cerr << "Ви бажаєте ввести дані з клавіатури, чи з файлу? Якщо з файлу, напишіть цифру 1\n";
    string type;
    cin >> type;
    if(type == "1"){
        cerr << "Напишіть назву файлу\n";
        string C;
        cin >> C;
        freopen(C.c_str(),"r",stdin);
        inputData();
    }else{
```

```
        inputData("console");
    }

    Algorithm();
    outputData(1);//частота

    return 0;
}
```