МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАНИ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
«КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Факультет інформатики

Кафедра математики

КУРСОВА РОБОТА

на тему:

«Стохастичні коаліційні ігри на графах»

Виконала студентка 3 р.н.:

Крючкова Анастасія Сергіївна

Науковий керівник:

Кандидат фізико-математичних наук, доцент

Чорней Руслан Констянтинович

Київ – 2021

Календарний план виконання роботи:

№ п/п	Назва етапу	Термін виконання	Примітка
1.	Отримання завдання на курсову роботу	11.10.2020	
2.	Огляд літератури за темою роботи	18.12.2020	
3.	Розбір задачі	25.02.2021	
4.	Програмна реалізація	09.05.2021	
5.	Створення слайдів для доповіді та написання доповіді	14.05.2021	

ЗМІСТ

ВСТУП 4

1. ТЕОРЕТИЧНА ІНФОРМАЦІЯ 7

1.1 Основне припущення 7

1.2 Розподілені гравці, які діють локально, на загальному графі системи 8

1.3 Стратегії розподілених гравців та локальна політика коаліцій 10

1.4 Формалізація математичної моделі стохастичної гри з одним гравцем 11

1.5 Марковський процес з виграшем 13

2. ПРОГРАМНА РЕАЛІЗАЦІЯ 15

2.1 Постановка задачі 15

2.1.1 Ергодичність Марковського процесу. 15

2.1.2 Повний очікуваний виграш 17

2.1.3 Стратегії гравця. 19

2.1.4 Коаліційний вплив. 21

ВИСНОВКИ 27

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 28

ОПИС МЕТОДІВ 29

ДОДАТКИ 30

ВСТУП

Історія ігор сягає давнього людського минулого, перші ігри засвідчені ще в 2600р. до н.е.. Ігри є однією з найдавніших форм соціальної взаємодії людини[1]. Загальні особливості ігор включають невизначеність наперед результату, узгоджені правила, конфлікт інтересів учасників, елементи випадковості та/або наслідки вольових рішень учасників, мету у вигляді отримання виграшу. Теорія ігор - галузь прикладної математики, що бере початок ще з XVIII століття. Її основоположником вважають математика Джона фон Ноймана, який разом з колегою, економістом Оскаром Моргенштерном, розробляли її для вирішення економічних проблем. Теорія ігор забезпечує інструменти для аналізу ситуацій, в яких сторони, які називаються гравцями, приймають взаємозалежні рішення. Ця взаємозалежність змушує кожного гравця враховувати можливі рішення чи стратегії іншого гравця при формуванні власної стратегії. Рішення гри описує оптимальні дії гравців, які можуть мати схожі, протилежні або змішані інтереси, та досягнення максимального виграшу. Підкреслюючи стратегічні аспекти прийняття рішень або аспекти, контрольовані гравцями, а не чистою випадковістю, теорія ігор доповнює і виходить за рамки класичної теорії ймовірності. На сьогодні теорія ігор широко використовується для дослідження багатьох процесів, наприклад, біологічних, інформаційних, штучного інтелекту, економічних, соціальних, у військовому та державному управлінні[3].

Існує багато ознак класифікації ігор, деякі з них: за кількістю гравців, за взаємодією гравців, за рівнем інформації про ситуації прийняття рішень, за характером отримання інформації і т.д.

Метою дослідження даної роботи є стохастичні коаліційні ігри на графах. Стохастичність передбачає те, що відома множина можливих станів системи і вірогіднісний розподіл потрапляння в них. Коаліція передбачає можливість гравцями об'єднуватися задля обміну один з одним інформацією й розподілу прибутку та/або витрат серед всіх учасників коаліції. Ігри, що розглядаються, є дискретними, у яких скінченна послідовність станів представлена у вигляді визначеної кількості вершин, а переходи між вершинами графа обумовлені двома факторами: ймовірністю переходу та обраною гравцем стратегією.

Надаємо перевагу в роботі саме інформаційній коаліції з розподілом витрат без узгодження стратегій між гравцями, оскільки вона дозволяє досліджувати дії одного гравця й абстрагуватись від аналогічних дій інших учасників коаліції, оскільки ми вважаємо їхні дії раціональними з метою максимізації прибутку. Таке спрощення дає можливість скористатися Марковським процесом вирішення. Різниця між стохастичною грою та Марковським процесом прийняття рішень полягає у кількості гравців – Марковський процес описує гру з одним учасником.

Гра з одним учасником передбачає взаємодію гравця з обстановкою гри, тобто сукупністю всіх об'єктів та суб'єктів, які на неї впливають: фактор випадковості, умови й правила гри, інші сторони(наприклад, споживачі), які не є гравцями, але які своїми рішеннями впливають на зміну стану системи.

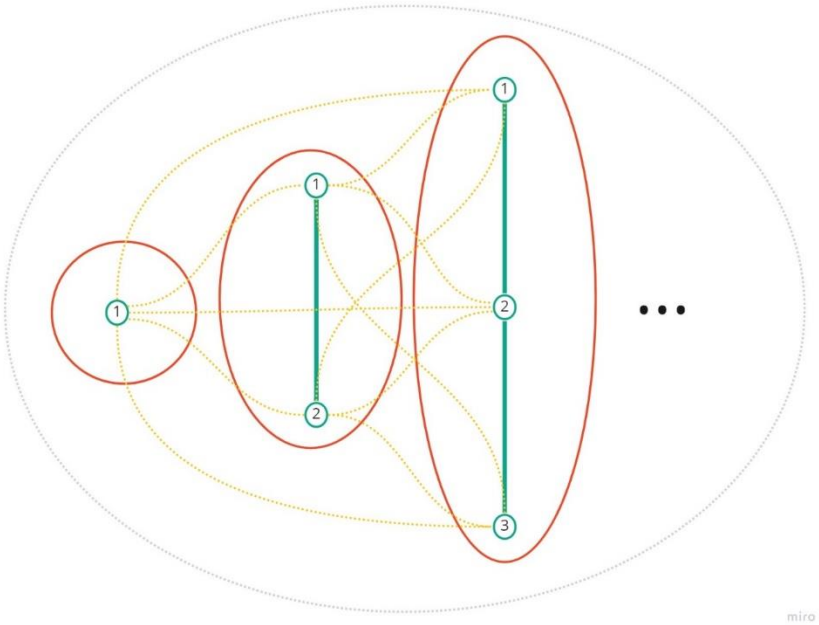
Внесок цієї роботи полягає в застосуванні Марковського процесу прийняття рішень для опису взаємодії n-нної кількості гравців в коаліції, де $n \in \mathbb{N}$, які мають оптимізувати свою поведінку в послідовних стохастичних іграх задля максимізації прибутку. Це

приводить до основних питань пошуку оптимальних стратегій та впливу коаліції на величину прибутку окремого гравця.

1. ТЕОРЕТИЧНА ІНФОРМАЦІЯ

1.1 Основне припущення

Досліджуємо стохастичну коаліційну гру n-нної кількості учасників, де $n \in \mathbb{N}$, вивчаючи математичну модель одного гравця, оскільки припускаємо, що дії кожного гравця раціональні в однакових умовах гри: ймовірності й вартості переходу однакові для всіх учасників гри. За допомогою аналізу дій одного гравця спрогнозуємо вплив коаліції з будь-якою кількістю учасників у вигляді зміни ймовірностей та вартостей переходу для даного гравця.



Модель гри представлена у вигляді графа, де гравці розташовані у вершинах графа взаємодії, ребра, позначені пунктирною жовтою лінією, представляють собою зв'язки між усіма учасниками гри, ребра, позначені зеленою суцільною лінією, представляють собою посилені зв'язки між гравцями в межах однієї коаліції.

Беремо до розгляду скінченну кількість коаліцій, всередині кожної може знаходитись n учасників, де $n \in \mathbb{N}$. Зв'язки між учасниками гри в межах усієї гри представляють собою

конкурентність і обмеженість ресурсів. Посилені зв'язки між гравцями в межах кожної окремої коаліції представляють собою обмін інформацією та розподіл витрат.

Математична модель стохастичної гри одного гравця представлена у вигляді орієнтованого графа, в якому вершини відповідають станам, а ребра - переходами з одного стану в інший. Ребра мають дві характеристики: ймовірність та вартість переходу. Цю модель можемо розмножувати на кількість учасників, які об'єднуються в коаліцію, аби дослідити вплив коаліції на хід гри. Стандартні моделі систем з n-нною кількістю учасників(гравців), які конкурують та / або співпрацюють за спільні ресурси, дозволяють гравцям приймати рішення на основі або всієї історії системи, або настільки ж ефективно на основі знання лише поточного стану системи, тому ми розглянемо Марковський процес прийняття рішень.

1.2 Розподілені гравці, які діють локально, на загальному графі системи

Матеріал цього й наступного підрозділів має реферативний характер. Використовуються теоретичні положення з роботи [3].

Для систем з координатами, які взаємодіють локально, структуру взаємодії визначимо через скінченний неорієнтований граф $\Gamma = (V, B)$, де V – множина вершин, B – множина ребер. Позначимо $\{k, j\}$ як ребро графа, що з'єднує вершини k і j.

Нехай $X_i = \{x_i^1, \dots, x_i^{n_i}\}$ – локальний простір станів у вершині $i \in V$. Тоді $X = \times_{i \in V} X_i$ – глобальний простір станів системи. Для довільної підмножини вершин $K \subset V$ вектор $x_K := (x_k, k \in K) \in X_K = \times_{i \in K} X_i$ позначає маргінальний опис стану у вершинах K.

Надалі всі введені випадкові величини будуть визначені на фіксованому ймовірнісному просторі $(\Omega, \mathcal{F}, P)$.

Означення 1.2.1. (Марковське випадкове поле)

Випадкова величина $\xi: (\Omega, \mathcal{F}, P) \to X$ називається (дискретним) марковським випадковим полем над $\Gamma = (\mathrm{V}, \mathrm{B})$, якщо виконується умова Маркова для всіх $k \in V$:

$$P\left(\xi_k = x_k / \xi_{V-\{k\}} = x_{V-\{k\}}\right) = P\left(\xi_k = x_k / \xi_{N(k)} = x_{N(k)}\right), \forall x \in X \quad (1)$$

де через ξ_k позначаємо випадкові величини, що набувають значень з X_K, зокрема, для $K = \{k\}$ пишемо ξ_k.

Оскільки ми спостерігаємо за поведінкою системи крізь час, то доречно вважати, що її еволюція – зміна станів – описується послідовністю таких випадкових полів ξ_k^t , що позначають випадкове значення в момент часу $\mathrm{t} = 0, 1, \ldots$ для вершини $k \in K$. Інакше кажучи, поведінка системи описується марковським процесом з дискретним часом $\xi = (\xi^t : t = 0,1, \ldots)$

Означення 1.2.2. (Марковське поле з дискретним часом)

Нехай $\xi = \{\xi^t : t = 0,1, \ldots\}$ - марковський процес із дискретним часом і простором станів X. Тоді якщо перехідні ймовірності X задовольняють умовам:

локальності : $\forall k \in V, x^0, \ldots, x^{t+1} \in X$

$$P(\xi_k^{t+1} = x_k^{t+1} / \xi^t = x^t, \ldots, \xi^0 = x^0) = P\left(\xi_k^{t+1} = x_k^{t+1} / \xi_{\tilde{N}(k)}^t = x_{\tilde{N}(k)}^t\right),$$

(2)

і синхронності : $\forall K \subset V, x^t, x^{t+1} \in X$

$$P(\xi_K^{t+1} = x_K^{t+1} / \xi^t = x^t) = \prod_{k \in K} P(\xi_k^{t+1} = x_k^{t+1} / \xi^t = x^t), \quad (3)$$

то процес ξ і граф Γ утворюють Марковське випадкове поле із синхронними компонентами, що взаємодіють локально, коротко – Марковське поле з дискретним часом. Процеси з дискретним часом

називаються стохастичними послідовностями або випадковими ланцюгами.

Наслідок 1.2.2.1. Об'єднавши умови (2) та (3), можемо описати перехідний механізм спостережуваного процесу для будь-якого моменту часу $t \in \mathbb{N}$

$$P(\xi_K^{t+1} = x_K^{t+1} / \xi^t = x^t) = \prod_{k \in K} P\left(\xi_k^{t+1} = x_k^{t+1} / \xi_{\tilde{N}(k)}^t = x_{\tilde{N}(k)}^t\right),$$
$$K \subset V, x^t, x^{t+1} \in X.$$

1.3 Стратегії розподілених гравців та локальна політика коаліцій

Опишемо структуру взаємодії гравців на загальному графі системи, де гравці розташовані у вершинах графа взаємодії Г, ребра представляють собою зв'язки між гравцями, кількість гравців $n = |V|$.

Означення 1.3.1 (Множина дій гравця)

$A = \times_{i \in V} A_i$ над Г = (V, B) – це множина дій, що обираються гравцем в моменти управління, де A_i – це множина можливих дій (рішень) для вершини i.

Для гравця, який приймає рішення, у вершині i в момент часу t в стані $\xi^t = x$ множина контрольних дій обмежена до $A_i^t(x) \subset A_i$, де $A_i^t(x)$ – множина допустимих дій (рішень) у момент часу t у стані x.

Гравці розташовані на вершинах скінченного графа Г = (V, B), тому коаліції, кількість яких скінченна, які створюють гравці, можна описати декомпозицією множини вершин: $V = \bigcup_{i=1}^{N} V_i$, $V_1, V_2, \dots, V_{\mathrm{N}} \neq \phi$, $\bigcap_{i=1}^{N} V_i = \phi$

V_i - множина вершин, де знаходяться гравці коаліції K_i.

Означення 1.3.2 (Стратегія гравця та політика коаліції)

(1) Нехай α_i^t позначає дію, обрану гравцем на вершині i в момент часу

$t, \alpha^t := (\alpha_i^t : i \in V)$ - вектор спільного рішення всіх гравців одночасно в момент часу $t,\ t \in N$.

(2) Стратегія (або політика) π для коаліції K_1 з взаємодіючими компонентами визначена як вектор координуючої політики $\pi = (\pi_i, i \in V_1)$, де для вершини i послідовність ймовірностей переходу: $\pi_i = \{\pi_i^0, \dots, \pi_i^t, \dots\} :\ \pi_i^t = \pi_i^t(\cdot/x^0, a^0, \dots, x^{t-1}, a^{t-1}, x^t)$.

Вважаємо дії в однакових умовах гри усіх гравців раціональними, тож стратегії гравців в межах конкретної коаліції однакові, тому скористаємося спрощенням і визначимо стратегію гравця через стратегію коаліції.

Закон руху системи характеризується множиною інваріантних ймовірностей переходу щодо часу. Щоразу, коли стан системи $\xi^t = x^t$ та приймається рішення $\alpha^t = a^t$, то ймовірність переходу $P\{\xi^{t+1} \in C/\xi^t = x^t, \alpha^t = a^t\} =: Q(C/x^t, a^t)$, а отже незалежна від минулого з огляду на нинішній (узагальнений) стан.

1.4 Формалізація математичної моделі стохастичної гри з одним гравцем

Матеріал цього підрозділу має реферативний характер. Використовуються теоретичні положення з роботи [4].

Кортеж $< X, A, p, r, (T, r_T) >$ визначає стохастичну гру одного гравця, де:

$-$ X є скінченним простором станів, що описують можливі конфігурації системи;

$-A$ є скінченною множиною дій або рішень гравця (A_X є скінченною множиною дій, доступних зі стану X), які контролюють динаміку стану;

$-p$ позначає функцію переходу зі стану в стану та характеризує динаміку стану системи, тобто являє собою ймовірність переходу $P_a(x, x') = P\ (x_{t+1} = x' \mid x_t = x, a_t = a)$ зі стану x в момент часу t при застосуванні дії a до стану x' в момент часу $t + 1$,

$-r$ позначає функцію винагороди, визначену в момент переходу системи: $R_a(x, x')$ є безпосередньою винагородою (або очікуваною безпосередньою винагородою), отримуваною після переходу зі стану x в стан x',

$-$T - часовий горизонт, за яким потрібно приймати рішення, і може бути як скінченним, так і нескінченним. Позначає кількість кроків системи. T потрібно визначити лише у випадку скінченного горизонту, у цьому випадку винагорода $r_T(x_T)$ теж визначена.

Додатковим параметром є дисконт:

$-\gamma \in [0,1]$ є коефіцієнтом знецінювання (англ. discount), який гарантує зближення нескінченної суми винагород до скінченного значення. Дисконт може мати економічну інтерпретацію, пов'язану з обмеженістю ресурсів.

Процеси прийняття рішень Маркова використовуються для моделювання динаміки стану стохастичної системи, коли ця система може контролюватися тим, хто приймає рішення. Стратегія або політика гравця - функція $\pi: X \to A$.

Середні критерії оптимізації на нескінченному горизонті визначають функції значення $V^{\pi}: X \to R$, які оцінюють ефективність політики за умови критерію оптимізації в будь-якому початковому стані системи x.

Оптимальна політика π^*, для якої значення критерію $V^{\pi*}(x)$ набуває найбільшого значення: $\forall x \in X, V^{\pi}(x) \leq V^{\pi*}(x)$.

1.5 Марковський процес з виграшем

Матеріал цього підрозділу має реферативний характер. Використовуються теоретичні положення з роботи [2] та [6].

Означення 1.5.1 (Ергодичний розподіл).

$\pi_j(n)$ - ергодичний розподіл ланцюга Маркова $\{\xi_n\}$, який у цьому разі також є ергодичним, якщо:

$$\forall p^{(0)} \lim_{n\to\infty} p_j^{(0)} = \pi_j, \ \sum_j \pi_j = 1$$

З визначення отримаємо, що

$$\pi_j(n+1) = \Sigma_{i=1}^{N} \pi_i(n) p_{ij}, n = 0,1,2, \dots$$

Вектор ймовірностей $\boldsymbol{\pi}(\boldsymbol{n})$ станів:

$$\pi(n) = \pi(0)\mathbf{P}^n, n = 0,1,2, \dots \tag{1.5.1}$$

p_{ij}^k – ймовірність переходу зі стану i в стан j при виборі стратегії k,

r_{ij}^k – винагорода в момент переходу зі стану i в стан j. Матриця винагород R визначена елементами r_{ij},

q_i^k - очікуваний виграш в момент переходу зі стану i при виборі стратегії k:

$$q_i^k = \sum_{j=1}^{N} p_{ij}^k r_{ij}^k \tag{1.5.2}$$

$d_i(n)$ - рішення в стані і на n-тому кроці. Поведінка гравця визначена, якщо для всіх і та n задано $d_i(n)$.

$v_i(n)$ - повний очікуваний виграш за n кроків за оптимальної поведінки, якщо система на початку знаходиться в стані i:

$$\forall n, v_i(n+1) = \max_k \sum_{j=1}^{N} p_{ij}^k \left[r_{ij}^k + v_j(n) \right], n = 0,1,2, \ldots$$

(1.5.3)

З урахуванням безпосередньо очікуваних виграшів для кожної стратегії рівність 1.5.3 може бути записана у вигляді:

$$v_i(n+1) = \max_k \left[q_i^k + \sum_{j=1}^{N} p_{ij}^k v_j(n) \right]$$

(1.5.4)

Векторна форма рівняння (1.5.4):

$$v_i(n+1) = \max_k \left[q_i^k + \boldsymbol{P} v_j(n) \right]$$

(1.5.5)

Для того, аби скористатися нерівностями вище, необхідно задати граничні виграші процесу $v_j(0)$.

2. ПРОГРАМНА РЕАЛІЗАЦІЯ

Для дослідження гри та реалізації алгоритмів було застосовано мову програмування Python, бібліотеку Numpy, Random та пакет Markov Decision Process (MDP) Toolbox.

Модуль mdp надає класи для вирішення дискретних Марковських процесів прийняття рішень.

2.1 Постановка задачі

Для перевірки правильності власних математичних підрахунків, класів пакету MDP Toolbox та власно розробленого коду розглянемо наступну гру.

Матриця переходів:

$$\mathbf{P} = \begin{bmatrix} 0{,}5 & 0{,}5 \\ 0{,}4 & 0{,}6 \end{bmatrix}$$

Нижче представлений відповідний орієнтований граф системи, вершини якого є станами, ребра - ймовірностями переходів:

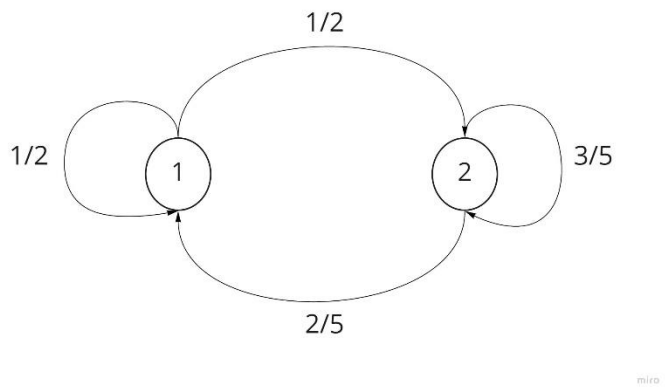


Рис. 1

2.1.1 Ергодичність Марковського процесу.

Ймовірність знаходження гравця в стані 1 через n кроків, за умови, що він знаходився у стані 1 в початковий момент часу t:

Гравець починає зі стану 1: $\pi_1(0) = 1, \pi_2(0) = 0$, $\pi(0) = [1\ 0]$.

При n=1:

$\pi(1) = \pi(0)\mathbf{P} = [1 \quad 0]\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{2}{5} & \frac{3}{5} \end{bmatrix} = \left[\frac{1}{2}\frac{1}{2}\right]$ – рівноймовірно, що гравець опиниться у стані 1 чи 2.

При n=2:

$\pi(2) = \pi(0)\mathbf{P^2} = \left[\frac{9}{20}\frac{11}{20}\right]$

При n=3:

$\pi(3) = \pi(0)\mathbf{P^3} = \left[\frac{89}{200}\frac{111}{200}\right]$ і т.д.

Робимо висновок, що $\pi_1(n) \to \frac{4}{9}, \pi_2(n) \to \frac{5}{9}$ при $n \to \infty$.

Гравець починає з 2 стану: $\pi_1(0) = 0, \pi_2(0) = 1$, $\pi(0) = [0\ 1]$.

При n=1:

$$\pi(1) = \pi(0)\mathbf{P} = [0 \quad 1]\begin{bmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{2}{5} & \frac{3}{5} \end{bmatrix} = \left[\frac{2}{5}\frac{3}{5}\right]$$

При n=2:

$\pi(2) = \pi(0)\mathbf{P^2} = \left[\frac{22}{50}\frac{28}{50}\right]$

При n=3:

$\pi(3) = \pi(0)\mathbf{P^3} = \left[\frac{111}{250}\frac{139}{250}\right]$ і т.д.

З розрахунків робимо висновок, що $\pi_1(n) \to \frac{4}{9}, \pi_2(n) \to \frac{5}{9}$ при $n \to \infty$ незалежно від того, з якого стану гравець починає.

Граничні ймовірності станів цієї системи за рівностями (1.5.1):

$$\pi_1 = \frac{1}{2}\pi_1 + \frac{2}{5}\pi_2,\ \pi_2 = \frac{1}{2}\pi_1 + \frac{3}{5}\pi_2$$

$$\pi_1 + \pi_2 = 1$$

Маємо єдиний розв’язок: $\pi_1 = \frac{4}{9},\ \pi_2 = \frac{5}{9}$.

Створимо симулятор гри з випадковою ймовірністю переходу зі стану в стан та з великою кількістю гравців (100000), які грають кілька кроків, статистично усереднені результати:

Гравець починає зі стану 1:

```
ймовірність потрапити в 1 стан 0.44541
ймовірність потрапити в 2 стан 0.55459
```

Вивід коду з Додатку 1

Гравець починає зі стану 2:

```
ймовірність потрапити в 1 стан 0.445559999999
ймовірність потрапити в 2 стан 0.55444
```

Вивід коду з Додатку 1

Перевірили правильність наших математичних викладок, дійшовши висновку, що гравець у $\frac{4}{9}$ всіх випадків буде знаходитися у стані 1, відповідно в $\frac{5}{9}$ випадках – у стані 2.

2.1.2 Повний очікуваний виграш.

Введемо матрицю виграшів:

$$\mathbf{R} = \begin{bmatrix} 9 & 3 \\ 3 & -7 \end{bmatrix}$$

З рівності (1.5.2) знайдемо, що $\mathbf{q} = \begin{bmatrix} 6 \\ -3 \end{bmatrix}$.

Скориставшись рівностями (1.5.2) й (1.5.5), задавши початкові значення виграшів, складемо таблицю значень повного очікуваного виграшу$v_i(n)$ гравця як функція стану й кількості кроків:

n	0	1	2	3	4	5	..
$v_1(n)$	0	6	7,5	8,55	9,555	10,5555	.
$v_2(n)$	0	−3	−2,4	−1,44	−0,444	0,5556	.

Табл. 1

Перевіримо дані за допомогою програмної реалізації ітераційного пошуку значень повного очікуваного виграшу гравця, маємо вивід:

```
[[ 6.          -3.        ]
 [ 7.5         -2.4       ]
 [ 8.55        -1.44      ]
 [ 9.555       -0.444     ]
 [10.5555       0.5556    ]
 [11.55555      1.55556   ]
 [12.555555     2.555556  ]
 [13.5555555    3.5555556 ]
 [14.55555555   4.55555556]
 [15.55555556   5.55555556]]
```

Вивід коду з Додатку 2

Перевіримо дані за допомогою пакету Markov Decision Process (MDP) Toolbox, маємо вивід:

```
[[10.5555  9.555   8.55    7.5     6.      0.    ]
 [ 0.5556 -0.444  -1.44   -2.4    -3.      0.    ]]
```

Вивід коду з Додатку 3

Як бачимо, дані збіглись.

2.1.3 Стратегії гравця.

Ускладнимо гру, ввівши множину дій(стратегій): 1 та 2. Матриці переходів та виграшів матимуть наступний вигляд:

для стратегії $k = 1$:

$$\mathbf{P^1} = \begin{bmatrix} 0{,}5 & 0{,}5 \\ 0{,}4 & 0{,}6 \end{bmatrix}, \mathbf{R^1} = \begin{bmatrix} 9 & 3 \\ 3 & -7 \end{bmatrix}$$

для стратегії $k = 2$:

$$\mathbf{P^2} = \begin{bmatrix} 0{,}8 & 0{,}2 \\ 0{,}7 & 0{,}3 \end{bmatrix}, \mathbf{R^2} = \begin{bmatrix} 4 & 4 \\ 1 & -19 \end{bmatrix}$$

Безпосередньо очікуваний виграш q_i^k в залежності від стратегії, розрахований за рівністю (1.5.2):

Стани		1	2	1	2
Стратегії		1		2	
Очікувані доходи	n =1	6	-3	4	-5
	n =2	7,5	-2,4	6,2	-3,7
	n =3	8,55	-1,44	8,22	-1,77

Табл. 2

Маємо такі ж результати, запустивши ітераційний цикл підрахунку:

```
[[[ 6.      -3.     ]
  [ 4.      -5.     ]]

 [[ 7.5     -2.4    ]
  [ 6.2     -3.7    ]]

 [[ 8.55    -1.44   ]
  [ 8.22    -1.77   ]]

 [[ 9.555   -0.444  ]
  [10.222    0.223  ]]

 [[10.5555  0.5556]
  [12.2222  2.2223]]]
```

Вивід коду з Додатку 4

Розрахуємо повний очікуваний виграш за допомогою рівності (1.5.2), задавши граничні значення $v_j(0)$ нулями, і скористаємося рівністю (1.5.4) для пошуку оптимальної стратегії:

n	0	1	2	3	4	5	..
$v_1(n)$	0	6	8,2	8,55	10,22	12,222	.
$v_2(n)$	0	−3	−1,7	−1,44	0,23	2,223	
$d_1(n)$	-	1	2	2	2	2	.
$d_2(n)$	-	1	2	2	2	2	

Табл. 3

При n=1 для стану 1:

$\boldsymbol{v}(\mathbf{0}) = 0$, $v_1(1) = \max q_1^k$. Так як $q_1^1 = 6$ та $q_1^2 = 4$, то в стані 1 очікуваний виграш найбільший, застосовуючи стратегію 1, при цьому повний очікуваний виграш $v_1(1) = 6$.

При n=1 для стану 2:

$v_2(1) = \max q_2^k$, враховуючи, що $q_2^1 = -3$ і $q_2^2 = -5$, то і в стані 2 найкращою є 1 стратегія, а повний очікуваний виграш $v_2(1) = -3$.

Результати запуску коду за допомогою пакету MDP демонструють, що процес послідовних наближень, який базується на співвідношенні (1.5.4), буде зводитися до найкращої стратегії для кожного стану, коли n необмежено зростає.

```
[[24.22222222 22.22222222 20.2222222  18.222222   16.22222    14.2222
  12.222      10.22        8.2         6.          0.        ]
 [14.22222222 12.22222223 10.2222223   8.222223    6.22223     4.2223
   2.223       0.23       -1.7        -3.          0.        ]]
[[1 1 1 1 1 1 1 1 1 0]
 [1 1 1 1 1 1 1 1 1 0]]
```

Вивід коду з Додатку 5

Для даної задачі збіжність має місце вже при $n = 2$ і в кожному стані найкращою є друга стратегія.

2.1.4 Коаліційний вплив.

Дослідимо коаліційний вплив на результати учасників гри. Керуємося тим, що участь у коаліції не несе сенсу, якщо вона невигідна для учасників.

Участь в коаліції впливає на дві характеристики: зменшення витрат шляхом їх поділу на кількість учасників та збільшення ймовірності переходу в стан з більшими виграшами, дослідивши ймовірнісні залежності від стану інших гравців.

Скористаємося матрицями переходів та винагород з моделі гри, представленої вище.

Для дослідження впливу коаліції введемо в множину дій гравців третю дію – вступити в коаліцію. Обрахуємо її матрицю переходів. Для цього скористаємося стартовими значеннями, заданими в Табл. 2.

Оптимальна стратегія гравця спрямована на отримання максимального прибутку, стан 1 є найбільш прибутковим.

Для гравців, що починають з 1 стану, межі ймовірностей потрапити в 1 стан наступні:

$p_{low} = 0{,}5$ - нижня межа, застосовуючи дію 1

$p_{high} = 0{,}8$ - верхня межа, застосовуючи дію 2

Покращення ймовірностей завдяки участі в коаліції мають лежати в цих межах: $\Delta = p_{low} - p_{high} = 0{,}3$.

Для гравців, що починають з другого стану, межі ймовірностей потрапити в 1 стан наступні:

$p_{low} = 0{,}4$ - нижня межа, застосовуючи дію 1

$p_{high} = 0{,}7$ - верхня межа, застосовуючи дію 2

Покращення ймовірностей завдяки участі в коаліції мають лежати в цих межах: $\Delta = p_{low} - p_{high} = 0{,}3$.

Оскільки Марковський процес ергодичний, то скориставшись рівністю (1.5.1), знайдемо граничні ймовірності станів цієї системи:

За дії $k = 1$:

$$\pi_1 = \frac{4}{9},\ \pi_2 = \frac{5}{9}$$

За дії $k = 2$:

$$\pi_1 = \frac{7}{9},\ \pi_2 = \frac{2}{9}$$

Для прикладу продемонструємо розрахунок для двох гравців за дії $k = 1$.

Повна група подій:

$$P(A_1) = \frac{4}{9} \cdot \frac{5}{9} = \frac{20}{81}; \quad P(A_2) = \frac{5}{9} \cdot \frac{4}{9} = \frac{20}{81}$$
$$P(A_3) = \frac{4}{9} \cdot \frac{4}{9} = \frac{16}{81}; \quad P(A_4) = \frac{5}{9} \cdot \frac{5}{9} = \frac{25}{81}.$$

$P(A_4) = \frac{25}{81} = 0{,}3086419753$ – ймовірність того, що жоден учасник коаліції не знаходиться в стані 1, тобто не має потрібної інформації для обміну.

$1 - \mathrm{P}(\mathrm{A}_4) = \frac{56}{81} = 0{,}6913580247$ – ймовірність того, що хоча б один учасник коаліції знаходиться в стані 1 з імовірністю $\frac{4}{9}$ і має потрібну інформацію для обміну, тобто стан гравця вплине на ймовірність переходу іншого гравця цієї коаліції.

Ймовірнісна залежність даного гравця від станів інших гравців коаліції з n-нною кількістю учасників для дії $k = 1$:

	1	2	3	4	5	6	7	8	9	10
1 стан	0,4	0,6913580247	0,8285322359	0,9047401311	0,9470778506	0,9705988059	0,9836660033	0,9909255574	0,994958643	0,9971992461
2 стан	0,6	0,3086419753	0,1714677641	0,09525986892	0,0529221494	0,02940119411	0,01633399673	0,009074442627	0,005041357015	0,002800753897
	Δ = 0,3	0,2074074074	0,2485596708	0,2714220393	0,2841233552	0,2911796418	0,295099801	0,2972776672	0,2984875929	0,2991597738

Додаток 7

Ймовірнісна залежність даного гравця від станів інших гравців коаліції з n-нною кількістю учасників для дії $k = 2$:

	1	2	3	4	5	6	7	8	9	10
1 стан	0,5	0,3950617284	0,5294924554	0,6340496875	0,7153719792	0,7786226505	0,827817617	0,8660803688	0,8958402869	0,9189868898
2 стан	0,5	0,6049382716	0,4705075446	0,3659503125	0,2846280208	0,2213773495	0,172182383	0,1339196312	0,1041597131	0,08101311022
	Δ = 0,3	0,1185185185	0,1588477366	0,1902149063	0,2146115938	0,2335867951	0,2483452851	0,2598241106	0,2687520861	0,2756960669

Додаток 7

Бачимо, що зі збільшенням кількості учасників коаліції ймовірність отримати потрібну інформацію, тобто ймовірність того,

що в коаліції виявиться гравець, який знаходиться у стані 1, прямує до 1.

Для відображення покращення ймовірності переходу в стан з найбільшим виграшом - 1 (тобто зміна матриці переходів для дії $k = 3$), нам варто додати до стартових значень дельту $\Delta = 0{,}3$ помножену на отримані ймовірності запозичити потрібну інформацію у членів коаліції (ймовірність знайти серед учасників коаліції тих, хто знаходиться в 1 стані):

$$p_{ij}^{3} = p_{ij}^{(0)} + \Delta \cdot {P_{info}}^{k}$$

Таблиця залежності ймовірностей переходу в стани 1 та 2 від кількості учасників коаліції:

для $k = 1$:

	1	2	3	4	5	6	7	8	9	10
1 стан	0,4	0,6074074074	0,6485596708	0,6714220393	0,6841233552	0,6911796418	0,695099801	0,6972776672	0,6984875929	0,6991597738
2 стан	0,6	0,3925925926	0,3514403292	0,3285779607	0,3158766448	0,3088203582	0,304900199	0,3027223328	0,3015124071	0,3008402262

Додаток 7

Як бачимо, ймовірність потрапити в кращий стан, у даному випадку 1, завдяки інформаційній коаліції для кожного гравця підвищується, але при цьому знаходиться в межах $p_{low} = 0{,}4$ та $p_{high} = 0{,}7$.

для $k = 2$:

	1	2	3	4	5	6	7	8	9	10
1 стан	0,5	0,6185185185	0,6588477366	0,6902149063	0,7146115938	0,7335867951	0,7483452851	0,7598241106	0,7687520861	0,7756960669
2 стан	0,5	0,3814814815	0,3411522634	0,3097850937	0,2853884062	0,2664132049	0,2516547149	0,2401758894	0,2312479139	0,2243039331

Додаток 7

Те ж саме спостерігаємо і для 2 дії: ймовірності переходу в 1 стан в межах $p_{low} = 0{,}5$ та $p_{high} = 0{,}8$.

Дослідимо вплив участі в коаліції на матрицю вартості переходів гравця.

Для цього скористаємося результатами запуску коду з Додатку 5, ввівши відповідні дані в матриці переходу для дії $k = 3$ (доєднатися до коаліції) для двох учасників і знайдемо мінімальний поріг збільшення виграшу, щоб участь у коаліції була вигідною і 3-ю дію метод пошуку оптимальної стратегії пакету MDP обрав як найоптимальніший.

Таким чином, ми досліджуємо мінімальну кількість учасників коаліції, коли участь в ній стає вигідною для всіх гравців. Нижче описано випадок для коаліції з 2 учасників.

Порівнявши з витратами для оптимальної стратегії $k = 2$ (до появи 3-ьої стратегії), мінімальний поріг збільшення виграшу завдяки участі в коаліції і, відповідно поділу витрат, складає $5{,}7 - 4 = -3{,}3 - (-5) = 1{,}7$ грошових одиниць на одного гравця.

Мінімальні витрати, поділені в даному випадку на двох, тобто витрати на коаліцію, складають 3,4.

Виведемо формулу для знаходження виграшів гравця, який є членом коаліції, в залежності від кількості учасників n в коаліції. Для цього скористаємося стартовими значеннями та введемо постійні витрати у вигляді x.

Знайдемо вартості переходу u:

$$u_1 - \frac{3{,}4}{1} = 4,$$

$$u_1 = 7{,}4$$

$$u_2 - \frac{3{,}4}{1} = -5$$

$$u_2 = -1{,}6$$

Розподіл витрат на одного гравця для знаходження в стані 1 знаходиться по формулі $costs_{\ 1} = \ (x + 3{,}4/n) + 4$

Розподіл витрат на одного гравця для знаходження в стані 2 знаходиться по формулі $costs_{\ 2} = \ (x + 3{,}4/n) - 5$

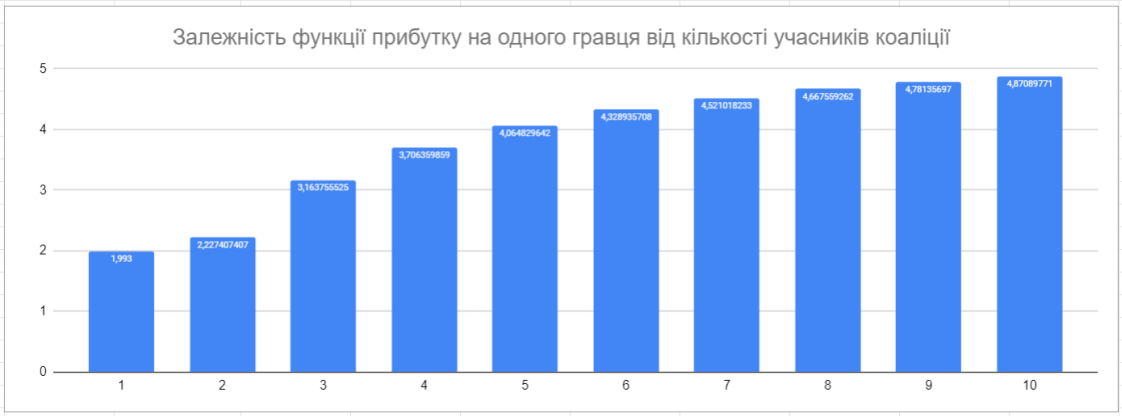
Складемо матрицю виграшів:

n =	1	2	3	4	5	6	7	8	9	10
	4	5,7	6,266666667	6,55	6,72	6,833333333	6,914285714	6,975	7,022222222	7,06
	-5	-3,3	-2,733333333	-2,45	-2,28	-2,166666667	-2,085714286	-2,025	-1,977777778	-1,94
Ваги переходу	7,4	Постійні витрати		x =	0					
	-1,6	Мінімальні спільні витрати на розробку			3,4					

Додаток 7

Застосувавши метод пошуку середньої винагороди з коду Додатку 6, введемо дані для коаліцій з визначеною кількістю гравців, аби проаналізувати її вплив на хід гри. Визначимо середній виграш на одного гравця:

n=	1	2	3	4	5	6	7	8	9	10
r=	1,993	2,227407407	3,163755525	3,706359859	4,064829642	4,328935708	4,521018233	4,667559262	4,78135697	4,87089771



Додаток 7

Вісь X – кількість членів коаліції, вісь У – отримані значення середнього виграшу на одного гравця.

Як бачимо, зі збільшенням учасників коаліції зростає середній виграш кожного її члена в межах від мінімального значення середнього виграшу одного учасника до теоретично можливого максимуму. Межі виграшу можна визначити, застосувавши код з Додатку 6.

ВИСНОВКИ

Використовуючи вже розроблені засоби для аналізу Марковського процесу прийняття рішень та власно реалізований ітераційний алгоритм пошуку й підрахунку повного виграшу, дослідили стохастичну коаліційну гру для багатьох гравців завдяки основному спрощенню, що дії усіх гравців аналогічні в однакових умовах гри. За допомогою аналізу дій одного гравця проаналізували та продемонстрували вплив коаліції з будь-якою кількістю учасників на результати гри окремого гравця.

Представлений код можна легко розширювати, додаючи нові стани та дії для гравця, отже фактично досліджувати не лише дискретні ігри, а й ігри неперервні в часі.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

[1] MacGregor Historic Games. History of Social Games, 2006.

[2] Р. А. Ховард Динамическое программирование и Марковские процессы/ Р. А. Ховард: [Пер. с англ. В.В.Рыкова под ред. Н.П.Бусленко] Издательство ”Советское радио Москва, 1964.

[3] Ruslan K. Chornei, Hans Daduna V. M., and Pavel S. Knopov. Stochastic games for distributed players on graphs, 2004.

[4] Frédérick Garcia, Iadine Chadès, Mj Cros. MDPtoolbox: A multi-platform toolbox to solve stochastic dynamic programming problems, 2014.

[5] Markov Decision Process (MDP) Toolbox documentation URL : https://pymdptoolbox.readthedocs.io/en/latest/api/mdptoolbox.html

[6] Чорней Р.К. Елементи теорії ймовірностей і випадкових процесів, 2018.

ОПИС МЕТОДІВ

Опис методів взятий з документації пакету Markov Decision Process (MDP) [5].

Class `mdptoolbox.mdp.FiniteHorizon`(*transitions, reward, discount, N, h=None*)

A MDP solved using the finite-horizon backwards induction algorithm.

Parameters:

- **transitions** (*array*) – Transition probability matrices.
- **reward** (*array*) – Reward matrices or vectors.
- **discount** (*float*) – Discount factor.
- **N** (*int*) – Number of periods. Must be greater than 0.
- **h** (*array, optional*) – Terminal reward. Default: a vector of zeros.

Functions:

- **V** (*array*) – Optimal value function. Shape = (S, N+1). `V[:, n]` = optimal value function at stage `n` with stage in {0, 1...N-1}. `V[:, N]` value function for terminal stage.
- **policy** (*array*) – Optimal policy. `policy[:, n]` = optimal policy at stage `n` with stage in {0, 1...N}. `policy[:, N]` = policy for stage `N`.

class `mdptoolbox.mdp.RelativeValueIteration`(*transitions, reward, epsilon=0.01, max_iter =1000*)

A MDP solved using the relative value iteration algorithm.

Parameters:

- **transitions** (*array*) – Transition probability matrices.
- **reward** (*array*) – Reward matrices or vectors
- **epsilon** (*float, optional*) – Stopping criterion. Default: 0.01.
- **max_iter** (*int, optional*) – Maximum number of iterations. Default: 1000.

Functions:

- **policy** (*tuple*) – epsilon-optimal policy
- **average_reward** (*tuple*) – average reward of the optimal policy

ДОДАТКИ

Додаток 1

```
import random
import numpy as np
# states matrix
S = ['s0', 's1']
# probability matrix
P = {'s0s0': 0.5, 's1s0': 0.4}
# number of steps
total = 100000
set_of_states = []

# current state [s0 = 0 or s1 = 1]
cur_state = 0
# cur_state = 1
while total > 0:
    # imitation of real situation with random probability
    probability = random.random()
    if cur_state == 0:
        if probability > P['s0s0']:
            prev_state = cur_state
            cur_state = 0
            set_of_states.append(cur_state)
        else:
            prev_state = cur_state
            cur_state = 1
            set_of_states.append(cur_state)
    else:
        if probability >= P['s1s0']:
            prev_state = cur_state
            cur_state = 1
            set_of_states.append(cur_state)
        else:
            prev_state = cur_state
            cur_state = 0
            set_of_states.append(cur_state)
    total -= 1

print("ймовірність потрапити в 1 стан", 1 -
np.sum(set_of_states)/len(set_of_states))
print("ймовірність потрапити в 2 стан",
np.sum(set_of_states)/len(set_of_states))
```

Додаток 2

```
import numpy as np

# default reward matrix
R = np.array([[9, 3],
              [3, -7]])

# default probability matrix
P = np.array([[0.5, 0.5],
              [0.4, 0.6]])

MAX_IT = 10
STATE_NUMBER = len(R)

# coefficient matrix
Q = np.zeros(STATE_NUMBER)

result_matrix = np.zeros((MAX_IT, STATE_NUMBER))
for i in range(MAX_IT):
    if i == 0:
        for j in range(STATE_NUMBER):
            summ = 0
            for k in range(STATE_NUMBER):
                a = P[j][k]*R[j][k]
                summ += a
            result_matrix[i][j] = summ
            Q[j] = summ
    else:
        for j in range(STATE_NUMBER):
            summ = 0
            for k in range(STATE_NUMBER):
                a = P[j][k]*result_matrix[i-1][k]
                summ += a
            result_matrix[i][j] = summ + Q[j]

print(result_matrix)
```

Додаток 3

```
import mdptoolbox
import numpy as np

R = np.array([[[9, 3],
                [3, -7]]])

P = np.array([[[0.5, 0.5],
               [0.4, 0.6]]])

fh = mdptoolbox.mdp.FiniteHorizon(P, R, 1, 5)
fh.run()
print(fh.V)
```

Додаток 4

```
import numpy as np

# default matrix
R = np.array([[[9, 3], [4, 4]],
              [[3, -7], [1, -19]]])

# probability matrix
P = np.array([[[0.5, 0.5], [0.8, 0.2]],
              [[0.4, 0.6], [0.7, 0.3]]])

# example result
# result = [[[6, -3], [4, -5]],
#           [[7.5, - 2.4], [6.2, -3.7]]]
# Q example
# Q_ex = [[6, 4], [-3, -5]]

MAX_IT = 2
STATE_NUMBER = len(R)
ACTION_NUMBER = 2

# coefficient matrix
Q = np.zeros((STATE_NUMBER, ACTION_NUMBER))

result_matrix = np.zeros((MAX_IT, ACTION_NUMBER, STATE_NUMBER))
for i in range(MAX_IT):
    if i == 0:
        for j in range(STATE_NUMBER):
            x = [0, 0]
            for k in range(ACTION_NUMBER):
                summ = 0
                for s in range(STATE_NUMBER):
                    a = P[k][j][s] * R[k][j][s]
                    summ += a
                x[k] = summ
            result_matrix[i][j] = x
        Q = result_matrix[0]
    else:
        for j in range(STATE_NUMBER):
            x = [0, 0]
            for k in range(ACTION_NUMBER):
                summ = 0
                for s in range(STATE_NUMBER):
                    a = P[k][j][s] * result_matrix[i-1][j][s]
                    summ += a
                x[k] = summ + Q[j][k]
            result_matrix[i][j] = x

print(result_matrix)
```

Додаток 5

```
import mdptoolbox
import numpy as np

# transition probabilities is a number of states by number of states matrix
and there`s one for each action

# 2 actions, 2 states, 2 states
prob = np.zeros((2, 2, 2))
# action a0
prob[0] = [[0.5, 0.5],
           [0.4, 0.6]]
# action a1
prob[1] = [[0.8, 0.2],
           [0.7, 0.3]]

# and reward matrix is a number of states(5) by the number of actions(2)
rewards = np.zeros((2, 2))
rewards[0] = [6, 4]
rewards[1] = [-3, -5]

vi = mdptoolbox.mdp.FiniteHorizon(prob, rewards, 1, 10)
vi.run()

print(vi.V)
print(vi.policy)
```

Додаток 6

```
import mdptoolbox.example
import numpy as np

# transition probabilities is a number of states by number of states matrix
and there`s one for each action

# 1 action, 2 states, 2 states
P = np.zeros((1, 2, 2))

# action join coalition
P[0] = [[0.8, 0.2],  # 1 state
        [0.7, 0.3]]  # 2 state

# and reward matrix is a number of states(2) by the number of actions(1)
R = np.zeros((2, 1))

R[0] = [4]
R[1] = [-5]
vi = mdptoolbox.mdp.FiniteHorizon(P, R, 1, 10000)
vi.run()
print(vi.V)
print(vi.policy)

rvi = mdptoolbox.mdp.RelativeValueIteration(P, R)
rvi.run()
print(rvi.average_reward)
```

Додаток 7

[Посилання на таблиці даних]