Міністерство освіти і науки України

Національний університет «Києво-Могилянська академія»

Факультет інформатики

Кафедра математики

# Магістерська робота

освітній ступінь - магістр

на тему: **«ОПТИМАЛЬНІ СТРАТЕГІЇ В КОАЛІЦІЙНИХ ІГРАХ З ЛОКАЛЬНОЮ ВЗАЄМОДІЄЮ»**

Виконав: студент 2-го року навчання,
спеціальності 124 Системний аналіз

Несисюк Мирослав Олегович

Керівник: Чорней Р. К.,
кандидат фізико-математичних наук,
доцент

Рецензент_____
(прізвище та ініціали)

Магістерська робота захищена
з оцінкою «_____»

Секретар ЕК _____
«____» _____20__ р.

Київ 2018

# ЗМІСТ

# ВСТУП

В цій роботі розглянуто актуальну на даний момент тему оптимальних стратегій в коаліційних іграх з локальною взаємодією. Ця тема є актуальною в зв`язку з її застосуваннях при моделюванні роботи мережі інтернет. Об`єктом є система з локальною взаємодією, а предметом стратегії гравців в цій системі. Метою цієї роботи є розглянути основні способи побудови систем з локальною взаємодією на основі задачі “хижак - жертва”, а завданням створити програму для знаходження отпимальних стратегій жертв для уникнення хижака і перевірити її виконувальність в складніших випадках наприклад декількох хижаків. Методом дослідження є комп`ютерна програма для моделювання задачі.

Структура роботи наступна:

В першому розділі розглянуто теорію стратегічних ігор

У розділі 2 розглянуто поняття коаліційної стратегічної гри та значення Шеплі такої гри.

У розділі 3 визначено кооперативні рішення для TU і NTU стратегічних ігор і показано теореми існування і єдиності.

У розділі 4 розглянуто MAC та її особливості.

Розділ 5 присвячено постановці та розв'язку задачі “хижак-жертва”.

Четвертий та п’ятий розділ створено на основі статі П.Кравця “Ігрова координація та самоорганізація стратегій в мультиагентній моделі “хижак-жертва””, яка розглядає основні властивості MAC та зв`язок задачі “хижак-жертва” з MAC.

# РОЗДІЛ 1. Стратегічні ігри

Стратегічні ігри в загальному випадку, можуть бути описані наступним чином. На кожному етапі, гра знаходиться в одному з кінцевого числа станів. Кожен з n гравців вибирає дію з кінцевої множини можливих дій. Дії гравців і стан спільно визначають виграш для кожного гравця і наступний стан.

Грунтуючись на роботі Неша ([16]), Джон Харсані ([9]), і Шеплі ([22]), ми розглянемо кооперативне рішення для стратегічних ігор і доведемо існування теореми.

Оригінальна ідея з'являється в Неша ([16]), який запропонував рішення, яке враховує як конкурентоспроможну так і кооперативні аспекти стратегічної гри. Нешем визначено таке рішення для двох гравцевої гри і доведено існування і унікальність теореми.

Рішення отримано за допомогою “торгу зі змінними загрозами”. У початковій стадії конкурентності, кожен гравець оголошує “стратегію загроз”, яка застосовується у випадку неможливості переговорів; результат розгортання цих стратегій є точкою “незгоди”. В подальшому кооперативому етапі, гравці координують свої стратегії для досягнення Парето оптимального результату, і розділюють прибуток по точці розбіжності; розділ здійснюється відповідно до принципів справедливості.

Розширення рішення Неша до n-гравцевих стратегічних ігор вимагає кілька ідей. Вони з'являються, більш-менш явно, в роботі Гарсануї ([9]), Шеплі ([22]), Аумана і Курца ([3]), і Неймана ([17]). Проте там не було створено єдиної комплексної теорії. Таким чином, ми розглянемо формальне визначення та доказ існування для того, що можна вважати найпростішим можливим

узагальнення рішення Неша. Ми називемо це *кооперативним рішенням стратегічної гри.*

Першим кроком треба розглянути альтернативний погляд на випадок двох гравців. У припущенні переказної корисності (TU), результат переговорів “зі змінними загрозами” можна описати дуже просто. (Опис випливає з Шеплі ([23]). Він також з'являється в Калай і Калай ([10]), які використовують його для визначення їхньої “ко-ко-вартості".)

Нехай s позначають максимальну суму виграшів гравців в будь-якому записі матриці виграшу, і нехай d буде MINMAX значення гри з нульовою сумою побудоване як різниця між гравцем 1 та 2 по виграшами. Тоді рішення Неша ділить суму S таким чином, що різниця в виграші складає d. В такому разі, виграші гравців 1 і 2, відповідно s/2+d/2 та s/2-d/2

Ця процедура може бути узагальнена на n-гравці TU-гри наступним чином. Нехай s максимальна сума виплат в будь-якому записі платіжної матриці, і нехай $d_s$ бути значенням MINMAX гри з нульовою сумою між підмножиною гравців, S, та його доповнення N\S, де гравці в кожній з цих підмножин співпрацюють як однин гравець, і де виграш це різниця між сумою виплат гравцям в S, та сумою виплат гравцям в N\S.

Кооперативне рішення ділить суму s таким чином, щоб відобразити відносні переваги підмножин S. Зокрема, кооперативне рішення це значення Шеплі з коаліційної гри v, де v(N)=s, і для S $\neq$ N, v(S)= $d_s$.

Оскільки описана вище процедура добре визначена і складає один вектор виграшу, то це означає, що будь-яка n-гравцева TU стратегічна гра має унікальне кооперативне рішення.

Наприклад в трьох-гравцевій стратегічній грі виграш гравця один складає $s/3+[d_1-(d_2+d_3)/2]/3$.

Далі ми розглянемо більш загальний випадок гри непередаваємої користносі (NTU). Тут, здійснення описаної вище процедури є більш складним завданням. У той час як у випадку TU, мета множини гравців яка діє як один гравець, очевидно, щоб максимізувати суму своїх виграшів, в випадку NTU, мета не ясна.

Проте, можна узагальнити рішення Неша для NTU ігор. Основна ідея це метод Шеплі “ лямбда-переказів”. Уявіть собі, що корисність можна передати після множення на деяких фактор масштабування, $\lambda = (\lambda_1, ..., \lambda_n) \geq 0$. Обчислити кооперативне рішення отриманої гри TU. Якщо це кооперативне рішення, насправді, можливе, без фактичних переказів корисності, то воно визначає кооперативне рішення гри NTU.

Легко перевірити, що в двох-гравцевих NTU іграх, “торги зі змінною загрозою” фактично збігаються з методом лямбда-переказів. Таким чином, початковий доказ Неша ([16]) встановлює існування і єдиність кооперативного рішення в будь-якій стратегічній грі з двох гравців.

Доведено, що кооперативне рішення існує в будь-якій n-гравцевій стратегічній грі. В NTU іграх з більш ніж двома гравцями може бути кілька рішеннь.

# РОЗДІЛ 2 Значення Шеплі для Коаліційних Ігор

Коаліційна гра з передаваємою користністю (“коаліційна гра” для стислості), це пара (N; v), де N = {1; ... ; n} скінченна множина гравців та відображення $v : 2^N \rightarrow$ R таке що v(0) = 0.

Будь-яка підмножина (“коаліція”) S N, v(S) загальну користність, яку гравці в S можуть здобути на самоті. Звичайно, така інтерпретація ґрунтується на припущенні, що корисність може бути передана серед гравців.

Шеплі [21] ввів поняття “значення” або апріорної оцінки про те, наскільки хід гри вартий кожному гравцеві. Таким чином, значення є відображенням $\varphi : R^{2\wedge N} \rightarrow R^N$, який присвоює кожній коаліційній грі v вектор окремої користності, φv.

Шеплі запропонував чотири бажані властивості, і довів, що вони визначають унікальне відображення значення. Це відображення -- значення Шеплі — може бути визначено наступним чином:

$$\varphi_i v = \frac{1}{n!} \sum_R \left( v(P_i^R \cup i) - v(P_i^R) \right) \quad (1)$$

де суммування відбувається на n! можливих розташувань множини N та де $P_i^R$ визначає підмножину $j \in N$ яка передує і в розташовуванні R. Таким чином, розподіл гравця і являє собою зважене граничне середнє значення вкладу, $v(S \cup i) - v(S)$, зважене відповідно до частоти, з якою і з'являється відразу після S у випадковому впорядкуванні N.

З формули, зрозуміло, що значення Шеплі має такі властивості. Для всіх коаліційних ігор (N, v), (N, w)

**Ефективність** $\sum_{i \in N} \varphi_i v = v(N)$

(2)

**Лінійність** $\varphi(\alpha v + \beta w) = \alpha \varphi v + \beta \varphi w, \forall \alpha, \beta \in R$

(3)

**Замітка** Це дві з чотирьох властивостей, які характеризують значення Шеплі. Ми визписуємо їх, тому що вони будуть використовуватися в подальшому. Також буде використано три додаткові наслідки (1):

$$\min_{S \subset N, i \notin S} (v(S \cup i) - v(S)) \leqslant \varphi_i v \leqslant \max_{S \subset N, i \notin S} (v(S \cup i) - v(S)), \quad (4)$$

$$\varphi_i v = \sum_{S \subset N, S \ni i} \frac{(s-1)!(n-s)!}{n!} (v(S \cup i) - v(N \setminus S)), \text{ де } s = |S| \quad (5)$$

і

$$\varphi v_i = \frac{1}{n} \sum_{k=1}^{n} \frac{(k-1)!(n-k)!}{(n-1)!} \sum_{S \subset N, S \ni, k=|S|} d_S = \frac{1}{n} \sum_{k=1}^{n} d_{i,k}, \quad (6)$$

де $d_S := v(S) - v(N \setminus S)$ та $d_{i,k}$ позначє середнє $d_S$ для всіх k-гравцевих коаліцій, які містять і.

З (5) випливає

***Наслідок 1.*** Значення Шеплі, φv, визначається різницями v(S)-v(N\S), $S \subset N$ .

Примітки:

- Властивість (4), відома як умова мілнора, було запропоновано Мілнором [14] в якості бажаної властивості рішення коаліційної гри. Це говорить про те, що розподіл гравця повинен перебувати між найменшим і найбільшим граничним вкладами цього гравця. Ясно, що це справедливо для значення Шеплі.

# РОЗДІЛ 3 Кооперативні Рішення Стратегічних Ігор

## 3.1 Стратегічні ігри

Скінченна стратегічна гра це набір G=(N,A,g) де

- N=(1,...,n) скінченний набір гравців
- A скінченна множина чистих стратегій гравців
- $g=(g^i)_{i\in N}$, де $g^i: A^N \to R$ функція прибутку і-го гравця.

**Замітка:** Для спрощення позначень ми вважаємо. що множина стратегій однакова для всіх гравців. Так як ці множини скінчені, то немає втрати загальності.

Використаємо таке саме позначення для g лінійного розширення $g^i: \Delta(A^N) \to R^N$, де для будь-якої множини K, $\Delta(K)$ визначає розподіл ймовірностей на К.

Також визначимо $A^i = A$, та $A^S = \prod_{i\in S} A^i$, а також $X^S = \Delta(A^S)$ (корелативні стратегії гравців в S)

## 3.2 Кооперативні рішення в TU іграх

У TU іграх передбачається, що гравці можуть робити побічні платежі; тобто користність може бути передана від одного гравця до іншого. Таким чином, має сенс розглядати максимальну суму виплат, доступну для всіх гравців:

$$v(N) := \max_{x\in X^N} \sum_{i\in N} g^i(x) \tag{8}$$

Тоді виникає питання, як сума v (N) ділиться між гравцями. Як вже говорилося у вступі, ми фіксуємо цінність будь-якої коаліції, S, за допомогою MinMax значення гри з нульовою сумою між S і його доповненням, де кожна з цих коаліцій виступає в якості одного гравця:

$$v(S)-v(N \setminus S):=\max_{x\in X^S}\min_{y\in X^{N\setminus S}}\left(\sum_{i\in S} g^i(x,y)-\sum_{i\notin S} g^i(x,y)\right) \quad (9)$$

Потім ми застосовуємо значення Шеплі. Це може бути виправдане з точки зору, що значення Шеплі є справедливим розподілом v(N), що відображає силу різних коаліцій, $S\subset N$ .

Визначення 1. Rооперативним рішенням стратегічної гри TU G є значення Шеплі, φv, де v є коаліційної грою, що задовольняє (9).

За наслідком 1, будь-яка коаліційна гра, яка задовольнить (9) має те ж значення Шеплі. Таким чином, описана вище процедура добре визначенна:

Теорема 1. Кожен TU стратегічна гра має унікальне кооперативне рішення.

І, застосовуючи рівняння (5), маємо:

Пропозиція 1. Нехай G стратегічна гра TU. Її кооперативне рішення, $\psi\in R^N$ може бути описано наступним чином:

$$\psi_i=\frac{1}{n}\sum_{k=1}^{n} d_{i,k},$$

де $d_{i,k}$ визначає середнє $d(S):=\max_{x\in X^S}\min_{y\in X^{N\setminus S}}\left(\sum_{i\in S} g^i(x,y)-\sum_{i\notin S} g^i(x,y)\right)$ для всіх к-гравцевих коаліцій S, які включають і.

### 3.3 Кооперативні рішення в NTU іграх

У загальній стратегічній грі, передача корисності не завжди можлива. Тому ми розглянемо розв'язок для таких випадків.

Оскільки в NTU іграх додавання виграшів не має сенс, в центрі уваги більше не максимальна сума виплат, а межа Парето допустимої множини, $F:=\{g(x): x\in X^N\}=conv\{g(a): a\in A^N\}$ яка складається з усіх можливих векторів виграшу, якщо гравці корелюють свої стратегії.

Щоб визначають кооперативний результат стратегічної гри NTU, ми використаємо методи лямбда-передавань Шеплі ([22]). Припускається, що корисність стає передаваємою після відповідного множення на коефіцієнт масштабування $\lambda=(\lambda_1,\dots,\lambda_n)\geqslant 0, \lambda\neq 0$ , після чого ми можемо продовжувати по аналогії з випадком TU гри.

Загальний прибуток для всіх гравців складає:

$$v_\lambda(N):=\max_{x\in X^N}\sum_{i\in N}\lambda_i g^i(x) \tag{10}$$

та умовна сила коаліції S складає

$$v_\lambda(S)-v_\lambda(N\setminus S):=\max_{x\in X^S}\min_{y\in X^{N\setminus S}}\left(\sum_{i\in S}\lambda_i g^i(x,y)-\sum_{i\notin S}\lambda_i g^i(x,y)\right) \tag{11}$$

Помітка: якщо S=N то це те саме що й (10) враховуючи $N\setminus N=\varnothing$

Як і в разі TU, ми визначаємо кооперативне значення стратегічної гри G, приймаючи значення Шеплі, $\varphi v_\lambda$. Тим не менш, є дві проблеми.

По-перше, неясно, як коефіцієнти масштабування λ повинні бути обрані.

По-друге, для досягнення $\varphi v_\lambda$ гравці, можливо, доведеться вдатися до передачі користності, яке виключенно в припущенні NTU.

Ідея, що лежить в основі методу лямбда-передачі це не вказати один λ, а прийняти будь-якого λ, для яких відповідне значення Шеплі може бути реалізований без фактичних передач користності. Таким чином, ми вимагаємо, щоб $\varphi(v_\lambda)$ був λ-перемасштабуванням розподілу в допустимій множині.

Іншими словами, за допомогою позначень $f*g:=(f_i g_i)_{i\in N}\ \forall f, g\in R^N$ ми вимагаємо, що $\varphi v_\lambda=\lambda*\psi$, де $\psi\in F$

Слід зазначити, що в силу лінійності значення Шеплі, для кожного вектора λ з коефіцієнтів масштабування, і для кожного a> 0, якщо $\varphi(v_\lambda) = \lambda * \psi$, то $\varphi(v_{\alpha\lambda}) = \psi * \alpha\lambda$. Отже, ми можемо нормалізувати λ:

$$\Delta:=\{\lambda=(\lambda_1,\ldots,\lambda_n)\in R^N, \lambda\geqslant 0, \sum_{i\in N}\lambda_i=1\}$$

У підсумку:

Визначення 2. $\psi\in F=conv\{g(a): a\in A^N\}$ є кооперативним рішенням стратегічної гри G, якщо $\exists\lambda\in\Delta$, такий, що $\varphi(\lambda)$ є коаліційною грою, що задовольняє умові (11).

Теорема 2. Кожна скінченна стратегічна гра має кооперативне рішення.

Ця теорема тісно пов'язана з результатами Шеплі [22] і [9] Гарсануї. Це особливий випадок Неймана [17].

Примітки: • Порядок, в якому певні обмінні курси - ті, які дозволяють реалізувати рішення без фактичних побічних платежів - виникають з даних гри,

має деяку схожість з образом, в якому ціни на ринку виникають з даних обмінної економіки ,

• Метод лямбда-передачі був застосований і в інших контекстах, в першу чергу у визначенні значення Шеплі для NTU коаліційної гри.

• Філософські підкріплення методу лямбда-передачі викликали жваву дискусію серед теорії ігор, наприклад, Аумана ([1] і [2]) і Рота ([20]).

# РОЗДІЛ 4. Мультиагентні системи

Сучасні мереживні технології створюють необхідність розроблення та впровадження розподілених інформаційних та програмних систем, складність яких унеможливлює будь-яке централізоване керування. Децентралізовані агентно-орієнтовані методи керування забезпечують роботоздатності таких систем в умовах невизначеності. [25–29].

Програмний агент – це автономна програмна система з елементами штучного інтелекту, яка може використовувати ресурси мережі для взаємодії з іншими агентами та людиною,.

Агент виконує поставлену перед ним задачу. Агенти мають такі властивості: 1)автономність: агенти повністю або частково незалежні; 2)інтелектуальність: можливість самостійно приймати рішення щодо своїх наступни дій на основі поточного стану; 3)спеціалізація: як правило, агенти вузькоспеціалізовані; 4)децентралізація: відсутні агенти, які керують усією системою.

Мультиагентна система (МАС) – це система, утворена декількома взаємодіючими агентами. У децентралізованій системі агенти взаємодіють локально. Програмні агенти можуть бути статичними, з фіксованим розміщенням, або мобільними, здатними мігрувати всередині мережі.

Функціонує МАС, як правило, в умовах невизначеності, обумовленої різними факторами. Так, автономність агентів призводить до появи внутрішнього випадкового стану. Невизначеність частково компенсується самонавчанням агентів та адаптивними стратегіями.

Теорія МАС зароджена в другій половині минулого століття на основі моделей, методів та алгоритмів кібернетики, теорії автоматів, інформації,

еволюції та штучного інтелекту. Системотворчий вплив на агентноорієнтоване прийняття рішень мали наукові праці з колективної поведінки автоматів, стохастичних ігор, організації комунікації між агентами.

Сучасні дослідження МАС пов’язані зі складними проблемами штучного інтелекту: розподілене прийняття рішень, керування знаннями, забезпечення взаємодій, структурна організація МАС і багато інших. Сучасні підходи створення МАС регламентуються специфікаціями фахових міжнародних організацій, наприклад, специфікацями FIPA (Foundation for Intelligent Physical Agents).

МАС використовуються для розв’язування задач паралелізації або взаємодії багатьох гравців, наприклад: логістики, електронної комерції, військової справи, подолання наслідків надзвичайних ситуацій, моделювання соціальних подій, керування мережними потоками, пошуку інформації у мережі Internet, організації мобільного зв’язку та інших.

Успішність розподіленого розв’язку задачі залежить від рівня координованості агентів. Координація – це властивість МАС забезпечувати узгоджену, впорядковану роботу усіх структур МАС.

Координація може бути централізованою або децентралізованою. Із зростанням структурної та функціональної складності розподіленої системи ефективність методів децентралізованої координації агентів зростає. Координація необхідна для узгодження індивідуальних цілей і поведінки агентів, коли кожен агент покращує або не погіршує результат функції корисності, а якість розв’язування задачі зростає.

В основі більшості методів координації, явно або неявно лежать поняття „спільних зобов'язань” (commitments) та „суспільних угод” (conventions).

„Спільне зобов’язання” означає виконання агентом його ролі – послідовності дій, для досягнення заданої мети в інтересах колективу агентів. Знання про зобов'язання інших агентів дозволяють агентові планувати свою поведінку в рамках „суспільного контексту”. „Суспільна угода” надає умови, за яких зобов'язання вважається виконаним, і можливі обставини за яких агент може або повинен відмовитися від виконання зобов'язань. У роботі [30] висунуто гіпотезу, яка стверджує, що всі механізми координації в МАС можуть бути виражені в термінах спільних зобов'язань і суспільних угод

Найчастіше поведінку агентів координують такими варіантами [25–29]:

1. Координація задоволенням загальних правил групової поведінки. Цей підхід корисний в системах із заданою структурою, з чіткими правилами групової поведінки. Координація містить два види діяльності: підтримка структури (забезпечення неможливості порушення суспільних правил) і використання правил поведінки для визначення конкретних дій агента, використовуючи його локальні знання.

2. Координація на основі обміну метаінформацією, наприклад, для узгодження зобов'язань і правил вирішення конфліктів. В цьому випадку агенти надають один одному свої локальні плани та узгоджують свої зобов'язання переговорами.

3. Командна робота -- співпраця агентів для досягнення загальної довгострокової мети, в динамічному середовищі невизначеності та протидії з боку суперника (або команди суперників). Кожен агент має обмежену інформацію про власну команду, про зовнішнє середовище та про суперника і реалізує власні наміри за допомогою індивідуальних дій, що виконуються синхронно або асинхронно з діями інших агентів.

4. Координація в умовах конкуренції, коли агенти конкурують між собою. Кожен агент максимізує свою функцію корисності, повністю ігноруючи інтереси інших агентів. В таких МАС агенти взаємодіють переговорами, в процесі яких агенти стараються досягти взаємовигідної угоди. Правила ведення

переговорів повинні бути попередньо встановлені й відомі всім агентам і формалізовані у вигляді протоколу переговорів. Протоколи ґрунтуються або на теоретико-ігровий арбітражної схемі Неша, або на моделі аукціону. Найчастіше використовується стандартний протокол контрактних мереж (FIPA: Contract Net Protocol) або його модифікації.

Фундаментом децентралізованої координації є взаємодія між агентами. Така взаємодія може бути неявною, коли агенти ведуть себе незалежно і впливають один на одного через зміну станів спільного середовища, та явною, комунікативною, коли агенти додатково здійснюють прямий обмін інформацією між собою. Обмін інформації може бути глобальним або локальним. Агенти можуть обмінюватися інформацією про розвідане середовище, даними про стратегії, значеннями отриманих виграшів тощо. Для обміну знаннями агенти використовують спеціальні мови (наприклад, KQML, ACL) та встановлені протоколи взаємодії.

Комунікація є основою кооперації агентів для досягнення спільної мети. Кооперація – це налагодження спільних дій у межах об`єднань агентів для оптимізації спільних виграшів або програшів. В кооперації, агенти частково обмежують ступінь власної свободи, враховуючи керованість з боку інших агентів об`єднання.

Колективні рішення є скоординованими, якщо вони задовольняють вимоги вигідності, стійкості та справедливості для усіх учасників прийняття рішень. На практиці використовують декілька різних критеріїв рівноваги, наприклад: критерій Неша.

Координація є одним з найважливіших факторів самоорганізації системи. Самоорганізація – це цілеспрямований процес створення, відтворення, впорядкування або вдосконалення організації (структури та функцій) складної

динамічної системи на основі внутрішніх факторів, без зовнішнього впливу. Термін самоорганізації вперше використав Р. Декарт. У літературі з кібернетики цей термін вперше зустрічається в роботах У. Р. Ешбі [31], який визначає систему із самоорганізацією як таку, що сама змінює свою структуру. П. Грассе вказав на те, що самоорганізація виникає за рахунок внутрішніх взаємодій між елементами системи через загальне середовище (явище стігмергії) [32]. В термінах термодинаміки самоорганізацію визначив І. Р. Пригожин [33]. Г.Хакен визначив самоорганізацію як явище спонтанного утворення просторових та часових структур у фізичних і хімічних системах, які перебувають далеко від стану рівноваги [34, 35]. Механізм самоорганізації як еволюцію циклів та гіперциклів хімічних реакцій у біологічних системах, які мають властивості самовідтворення та здатність до виживання [36], визначив М. Ейген.

За певних умов самоорганізація може проявлятися в різних природніх (наприклад, фізичних, хімічних, біологічних), організаційних (наприклад, соціальних, екологічних, економічних) або штучних (наприклад, мультиагентних) системах. Класичними прикладами самоорганізації є фізика лазера, хімічна реакція Бєлоусова–Жаботинського, конвективні комірки Бенара, біологічна самоорганізація бактерій, колоній або роїв комах, косяків риб, зграй птахів тощо [37].

Самоорганізація МАС – це здатність об`єднання агентів з локальними зв’язками і цілями досягати стійких стратегій поведінки в умовах невизначеності самонавчанням та забезпечувати виконання глобальної мети розвитку системи. Самоорганізація та складні форми поведінки МАС можуть проявлятися при реалізації найпростіших дій агентів [38, 39].

МАС мають усі необхідні вимоги для забезпечення самоорганізації:

1) автономність – здатність системи до самостійного прийняття рішень;

2) відкритість – здатність сприймати зовнішній світ за допомогою рецепторів і локально впливати на нього за допомогою ефекторів;

3) наявність програмного або фізичного середовища для розподіленої взаємодії агентів;

4) наявність засобів підтримання взаємодії та кооперації агентів;

5) відсутність зовнішнього або централізованого керування;

6) динамічна оптимізація виконується в процесі роботи системи і не вимагає попереднього настрою- вання або планування;

7) простота правил локальної взаємодії, яка веде до складної поведінки системи загалом;

8) можливість повторного використання варіантів дій у часі;

9) адаптивність та стійкість стосовно змін – здатність належного реагування на зміни зовнішнього середовища.

Виділяють такі механізми самоорганізації МАС:

1) прямі взаємодії між агентами за допомогою відповідних протоколів обміну даними;

2) непрямі комунікації через середовище;

3) самоорганізація поведінки агентів на основі використання навчання з підкріпленням (заохоченням);

4) кооперативна поведінка індивідуальних агентів;

5) вибір типової архітектури системи (холонічний підхід з елементами централізації).

Вивченням процесів самоорганізації систем різної природи займається теорія систем та синергетика – міждисциплінарна наука про нелінійні відкриті дисипативні (такі, що розсіюють енергію) системи [34].

Ентропія є мірою оцінювання самоорганізації системи. Зменшенням рівня ентропії повищує ступінь самоорганізації системи. Зовнішніми проявами самоорганізації можуть бути утворення впорядкованих структур, скоординованих дій агентів та властивість емерджентності – набута властивість

системи, не характерна для її складових. Такі або інші прояви самоорганізації дозволяють розглядати і вивчати МАС як єдиний організм.

Система проявляє емерджентність (системний ефект), якщо на макрорівні в ній виникають нові властивості, процеси, поведінка, структури, патерни тощо, які є наслідком локальних взаємодій елементів на мікрорівні, причому ці властивості не описуються в властивостях мікрорівня [38, 39].

Емерджентність виникає за таких умов :

1) наявність принаймні дворівневої організації: мікрорівень, де відбуваються локальні взаємодії, та макрорівень, де може виявитися емерджентний процес;

2) нелінійність: взаємодія на мікрорівні є нелінійною;

3) наявність зворотного зв'язку: локальні взаємодії на мікрорівні повинні містити зворотний зв'язок;

4) динамічна рівновага: емерджентні процеси не мають статичної рівноваги.

Явища емерджентності мають такі властивості:

1) новизна: невисловлювальність у термінах мікрорівневих компонентів;

2) когерентність: узгоджене проходження хвильових або коливних процесів, яке проявляється тоді, коли в системі виникають відповідні взаємодії;

3) емерджентність проявляється на макрорівні стосовно до породжуючих це явище компонентів;

4) динаміка: емерджентність виникає в процесі взаємодії і не є заданою заздалегідь або ззовні;

5) візуалізація: емерджентність явно демонструє себе тим або іншим способом, наприклад, виникненням впорядкованих структур, фазових переходів тощо.

Враховуючи притаманні колективу агентів фактори конкуренції, взаємодії, кооперації, навчання, самоорганізації, для дослідження МАС в умовах невизначеності використаємо модель стохастичної гри [40], яка забезпечує адаптивний пошук точок рівноваги функцій виграшів на комбінованих змішаних стратегіях гравців. Під час стохастичної гри гравці навчаються вибирати оптимальні в середньому чисті стратегії (дії), перебудовуючи власні вектори динамічних змішаних стратегій (умовних імовірностей варіантів дій).

# РОЗДІЛ 5. Задача “хижак-жертва”.

## 5.1 Мета постановки задачі

Процеси координації та самоорганізації МАС вивчатимемо на основі ігрової моделі переслідування „хижак – жертва” [41]. Запозичені від природи моделі переслідування з різноманітними сценаріями (наприклад, переслідування акулою косяка риб, атака коршуном зграї птахів тощо) використовуються для відпрацювання стратегій поведінки агентів у розподілених системах прийняття рішень. У моделях переслідування виділяють агентів-хижаків та агентів-жертв, які мають протилежні цілі. Кінцевою метою агентів-хижаків є досягнення однієї із жертв або мінімізація відстані до жертви. Агенти-жертви стараються зберегти власне життя або максимізувати чи дотримувати безпечну відстань до хижака. Вибір цієї задачі обумовлений її простотою та зручністю візуалізації скоординованих дій агентів як прикладу оптимальних стратегій.

Метою цієї роботи є визначення умови та механізми локальної координації агентів, для самоорганізації МАС. Для досягнення мети необхідно розв’язати такі задачі: побудувати модель гри, розробити метод та алгоритм для розв’язування та виконати програмне комп’ютерне моделювання для виявлення координації та самоорганізації МАС.

## 5.2 Постановка ігрової задачі

Враховуючи розподіленість системи та колективний характер формування рішень, антагонізм (між хижаком та жертвою) та узгодженість (у межах групи хижаків або жертв) цілей, розв’яжемо задачу самоорганізації МАС на основі адаптивних методів стохастичних ігор [40, 41].

Стохастична гра $\Gamma = (D, \{U^i\}_{i \in D}, \{P^i(\xi^i)\}_{i \in D})$ задається множиною

агентів-гравців D $\forall i \in D$ з $N_i \geq 2$ векторами чистих стратегій $u^i[1], u^i[2], \ldots, u^i[N_i]$ та апріорі невідомими розподілами $P^i(\xi^i)$ випадкових програшів $\xi^i \ \forall i \in D$. Нехай один з агентів є хижаком, а решта – жертвами. Чисті стратегії u[k], k=1..N визначають напрямки можливих рухів агентів.

Повторювальна гра розгортається у дискретні моменти часу n =1,2,... . Для цього кожен агент $i \in D$ задає $N_i \geq 2$ власних чистих стратегій $u^i_n = u^i \in U^i$.

Структура МАС визначає локально обумовлений механізм формування випадкових програшів $\xi^i_n = \xi^i_n(u^{Di}_n)$, що є функціями спільних стратегій $u^{D_i} \in U^{D_i} \equiv \prod_{j \in D_i} U^j$ з локальних підмножин агентів $D_i \subseteq D. D \neq \varnothing, \forall i \in D$. Випадкові програші $\{\xi^i_n\}$ є незалежними $\forall n \in U, \forall i \in D, n = 1, 2, \ldots,$ мають постійне математичне сподівання $M\{\xi^i_n(u^{Di})\} = v(u^{Di}) = \text{const}$ та обмежений другий момент $\sup_n M\{[\xi^i_n(u^{D_i})]^2\} = \sigma^2(u^{D_i}) < \infty$. Стохастичні характеристики випадкових програшів $\{\xi^i_n\}$ $\forall i \in D$ апріорі невідомі агентам.

Після вибору варіантів $u^i_n \ \forall i \in D$ гравці отримують поточний програш, який складається з трьох складових:

$$\xi^i_n = \lambda \xi^i_n[1] + (1-\lambda)\xi^i_n[2] + \mu_n, \qquad (1)$$

де $\lambda$ – ваговий коефіцієнт $\in [0,1]$; $\mu_n \sim$ Normal (0,d) – адитивний білий гауссівський шум, нормально розподілена випадкова величина з нульовим математичним сподіванням та дисперсією $d > 0$.

Перша складова (1) визначає штраф за порушення координації руху сусідніх агентів: $\xi^i_n = \sum_{j \in D_i} \chi(u^i_n \neq u^j_n)$, де $\chi(\cdot)$ – індикаторна функція події, $u^i_n \in \{0,1\}$ – поточна стратегія жертви $i \in D$.

Друга складова (1) визначає штраф за порушення умови реплікації (повторення) лівосторонніх або правосторонніх дій у напрямку руху агента: $\xi_n^i[2]=\chi(s\in D_i)\chi(u_n^i\neq u_n^s)+\chi(s\notin D_i)\chi(u_n^i\neq u_n^{Side(i)})$ , де s – ідентифікатор хижака; $u^s_n$ – поточна стратегія хижака; Side(i) – операторна функція для визначення ідентифікатора сусіднього агента, розміщеного ліворуч Side(i) = Left або праворуч Side(i) = Right від напрямку руху жертви.

Якість вибору чистих стратегій у момент часу n визначається середніми програшами

$$Z_n^i=\frac{1}{n}\sum_{t=1}^{n}\zeta_t^i \quad \forall i\in D \quad . \tag{2}$$

Мета гри полягає у мінімізації функцій середніх програшів:

$$\lim_{n\to\infty} Z_n^i \to \min_{u_n^i} \quad \forall i\in D. \tag{3}$$

Отже, взаємодіючи з середовищем, на основі спостереження поточних програшів $\{\zeta^i_n\}$ кожен гравець і $\in D$ пов $\{u^i_n\}$ так, щоби з часом n =1,2,... забезпечити виконання системи критеріїв (3).

Розв’язки ігрової задачі задовольнятимуть одну з умов колективної рівноваги, наприклад, Неша, Слейтера, Парето, залежно від методу формування послідовностей стратегій $\{u^i_n\}$ $\forall i \in D$.

### 5.3 Метод розв’язування задачі

Для вибору варіантів рішень в умовах невизначеності застосуємо марківські адаптивні методи, які на основі опрацювання поточної інформації про систему забезпечують оптимальний у середньому вибір варіантів дій у наступні моменти часу.

Сформуємо послідовність варіантів рішень $\{u^i_n\}$ на основі динамічних векторів змішаних стратегій $p^i_n = (p^i_n(1), p^i_n(2), ..., p^i_n(N))$ $\forall i \in D$ $p^i_n(j)$, j =1..N яких є імовірностями вибору чистих стратегій за умови реалізації передісторії вибору чистих стратегій $\{u^i_t | t=1,2,...,n-1\}$ та отримання відповідних програшів $\{\zeta^i_t | t=1,2,...,n-1\}$. Змішані стратегії набувають значення на N -вимірних одиничних симплексах:

$$S^N = \{p | \sum_{j=1}^{N} p(j) = 1 ; p(j) \geq 0 \quad (j = 1..N)\}.$$

Значення чистих стратегій визначаються з умови:

$$u^i_n = \{u^i(l) | l = arg\min_{l} \sum_{k=1}^{l} p^i_n[k] > \omega \quad (l = 1..N)\}, \qquad (4)$$

де $\omega$ випадкова величина з рівномірним розподілом.

Необхідно визначити метод зміни векторів змішаних стратегій $p^i_n \ \forall i \in D$, який згідно з (4) забезпечуватиме генерування випадкових чистих стратегій $\{u^i_n\}$ так, щоб в асимптотиці часу забезпечити виконання системи критеріїв (3).

Побудуємо метод розв'язування стохастичної гри на основі стохастичної апроксимації умови доповняльної нежорсткості детермінованої гри, справедливої для змішаних стратегій у точці рівноваги за Нешем [42]. Для цього визначимо полілінійну функцію середніх програшів детермінованої гри:

$$V^i(p^{D_i}) = \sum_{u^{D_i} \in U^{D_i}} v^i(u^{D_i}) \prod_{j \in D_i ; u^j \in U^{D_i}} p^j(u^j),$$

де $v(u^{D_i}) = M\{\xi^i_n(u^{D_i})\}$.

Тоді векторна умова доповняльної нежорсткості (CS, Complementary Slackness) матиме вигляд: $CS = \nabla_{p^i} V^i(p^{D_i}) - e^{N_i} V^i(p^{D_i}) = 0 \ \forall i \in D$, де $\nabla_{p^i} V^i(p^{D_i})$ – градієнт функції середніх програшів; $e^N = (1_j \mid j=1..N)$ – вектор, всі компоненти якого дорівнюють 1; $p^{D_i} \in S^{M_i}$ – комбіновані змішані стратегії гравців з локальних множин $D_i$, задані на опуклому одиничному симплексі

$S^{M_i}(M_i=\prod_{j\in D_i}|D_i|)$ .

Для врахування розв'язків у вершинах одиничного симплексу виконаємо зважування умови доповняльної нежорсткості елементами векторів змішаних стратегій:

$$diag(\overrightarrow{p^i})(CS)=0 \quad \forall i\in D \ , \tag{5}$$

де diag($p^i$) – квадратна діагональна матриця порядку $N_i$, побудована з елементів вектора $p_i$.

Враховуючи, що $diag(p^i)[\nabla_{p^i}V^i(p^{D_i})-e^{N_i}V^i(p^{D_i})]=E\{\zeta^i_n[e(u^i_n)-p^i_n]|p^i_n=p^i\}$ , де E{} – функція математичного сподівання, з (5) на основі методу стохастичної апроксимації отримаємо рекурентну залежність:

$$p^i_{n+1}=\pi^N_{\varepsilon_{n+1}}\{p^i_n-\gamma_n\zeta^i_n[e(u^i_n)-p^i_n]\}, \tag{6}$$

де $\pi^N_{\varepsilon_{n+1}}$ – проектор на одиничний ε -симплекс $S^N_{\varepsilon_{n+1}}\subseteq S^N$ [41]; $p^i_n\in S^N_{\varepsilon_n}$ – змішані стратегії і-го агента; $\gamma_n>0$ – монотонно спадна послідовність невід'ємних величин, яка регулює величину кроку методу; $\varepsilon_n>0$– монотонно спадна послідовність невід'ємних величин, яка регулює швидкість розширення ε -симплексу; $\zeta^i_n\in R^1$ – поточний програш агента; $e(u^i_n)$ – одиничний вектор-індикатор вибору варіанта $u^i_n\in U^i$.

Проектування на розширюваний $\varepsilon_n$-симплекс $S^N_{\varepsilon_{n+1}}$ забезпечує виконання умови $p^i_n[j]\geq\varepsilon_n$, j = 1..N, необхідної для повноти статистичної інформації про вибрані чисті стратегії, а параметр $\varepsilon_n\to 0$, n = 1,2,... використовується як додатковий елемент керування збіжністю рекурентного методу.

Стохастична гра розпочинається з ненавчених векторів змішаних стратегій зі значеннями елементів $p^i_0(j) = 1/N$ , де j = 1..N. У наступні моменти часу динаміка векторів змішаних стратегій визначається марківським

рекурентним методом (6).

У момент часу n кожен гравець $i \in D$ на о

вибирає чисту стратегію $u^i_n$, за що до моменту часу n+1 отримує поточний програш $\zeta^i_n$, після чого обчислює змішану стратегію $p^i_{n+1}$ згідно з (6).

Завдяки динамічній перебудові змішаних стратегій на основі опрацювання поточних програшів, метод (6) забезпечує адаптивний вибір чистих стратегій у часі.

Параметри $\gamma_n$ та $\varepsilon_n$ визначають умови збіжності стохастичної гри і можуть бути задані так:

$$\gamma_n = \gamma n^{-\alpha}, \ \varepsilon_n = \varepsilon n^{-\beta}, \tag{7}$$

де $\gamma > 0$ ; $\alpha > 0$ ; $\varepsilon > 0$ ; $\beta > 0$ .

Збіжність стратегій (6) до оптимальних значень з імовірністю 1 та у середньоквадратичному визначається співвідношеннями параметрів $\gamma_n$ та $\varepsilon_n$, які повинні задовольняти базові умови стохастичної апроксимації [44].

Ефективність прийнятих рішень оцінюється:

1) функцією середніх втрат:

$$Z_n = \frac{1}{L} \sum_{i=1}^{L} Z^i_n, \tag{8}$$

де L =| D | – потужність множини гравців;

2) середньою кількістю скоординованих стратегій гравців:

$$K_n = \frac{1}{n} \sum_{t=1}^{n} \sum_{i=1}^{L} \chi(\xi^i_n[1] + \xi^i_n[2] = 0), \tag{9}$$

де $\chi(\cdot) \in \{0,1\}$ – індикаторна функція події;

## 5.4 Алгоритм розв’язування стохастичної гри

1. Задати початкові значення параметрів:

n = 0 – початковий момент часу;

L =|D| – кількість гравців;

N – кількість чистих стратегій гравців;

$U^i$ = ( $u^i[1]$, $u^i[2]$, ... , $u^i[N]$), i=1..L – вектори чистих стратегій гравців;

$p^i_0$= (1/N,...,1/N), i =1..L – початкові змішані стратегії гравців;

$\gamma > 0$ – параметр кроку навчання;

$\alpha \in (0,1]$ – порядок кроку навчання;

$\varepsilon$ – параметр $\varepsilon$ -симплекса;

$\beta > 0$ – порядок швидкості розширення $\varepsilon$ -симплекса;

d > 0 – дисперсія завад;

max n – максимальна кількість кроків методу.

2. Вибрати варіанти дій $u^i_n \in U^i$, i =1..L згідно з (4).

3. Отримати значення поточних програшів $\zeta^i_n$, i =1..L згідно з (1). Поточні значення гауссівського білого шуму обчислюють за формулою:

$$\mu_n = \sqrt{d}\left(\sum_{j=1}^{12} \omega_{j,n} - 6\right),$$

де $\omega \in [0,1]$ – дійсне випадкове число з рівномірним законом розподілу.

4. Обчислити значення параметрів $\gamma_n$, $\varepsilon_n$ згідно з (7).

5. Обчислити елементи векторів змішаних стратегій $p^i_n$, i =1..L згідно з (6). 6. Обчислити характеристики якості прийняття рішень $Z_n$ (8), $K_n$ (9).

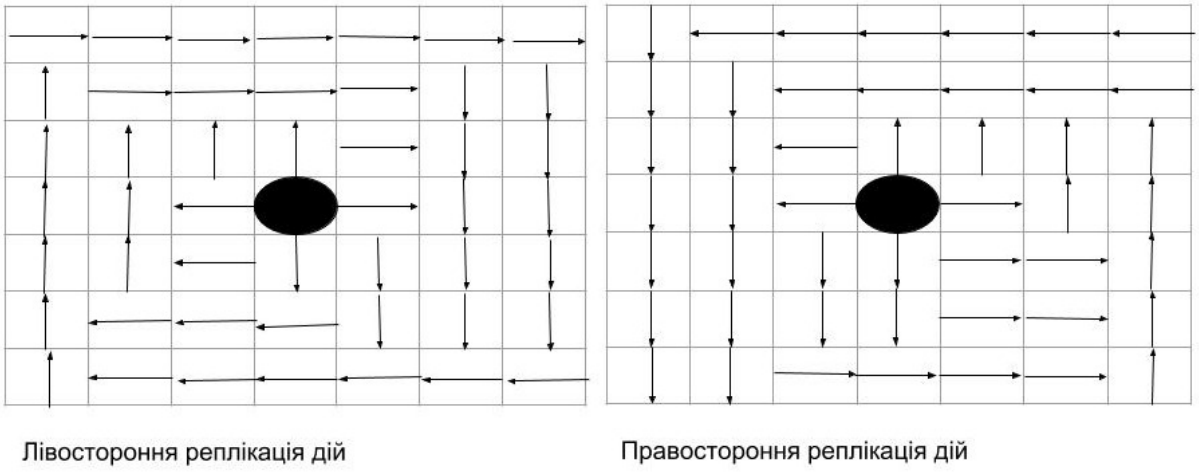
7. Задати наступний момент часу n := n +1.

8. Якщо $n<n_{max}$, то перейти на крок 2, інакше – кінець.

## 5.5 Результати комп'ютерного моделювання

Розв'язування стохастичної гри переслідування „хижак–жертва" виконаємо за допомогою ігрового методу (6) з параметрами: D ={(x,y)}, x =1..m, y =1..m, m = 7, s = (4,4), N = 4, U = (front=0, right=1, behind=2, left=3), λ = 0.5, γ=1, ε = 0.999/N, α = 0.01, β = 2, $n_{max}=10^5$.

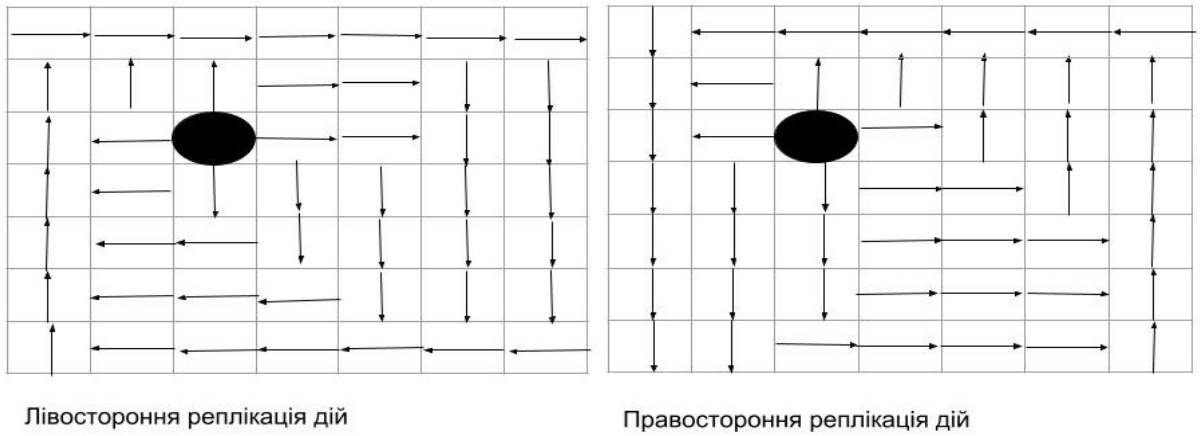
Метод (6) забезпечує розв'язування стохастичної гри у чистих стратегіях. Початкові стратегії агентів є нескоординованими, що визначається вибором довільного напрямку руху u ∈U. У г
свої дії. Отримані варіанти скоординованих стратегій навченої гри зображено на рис. 1.




(рис.1)

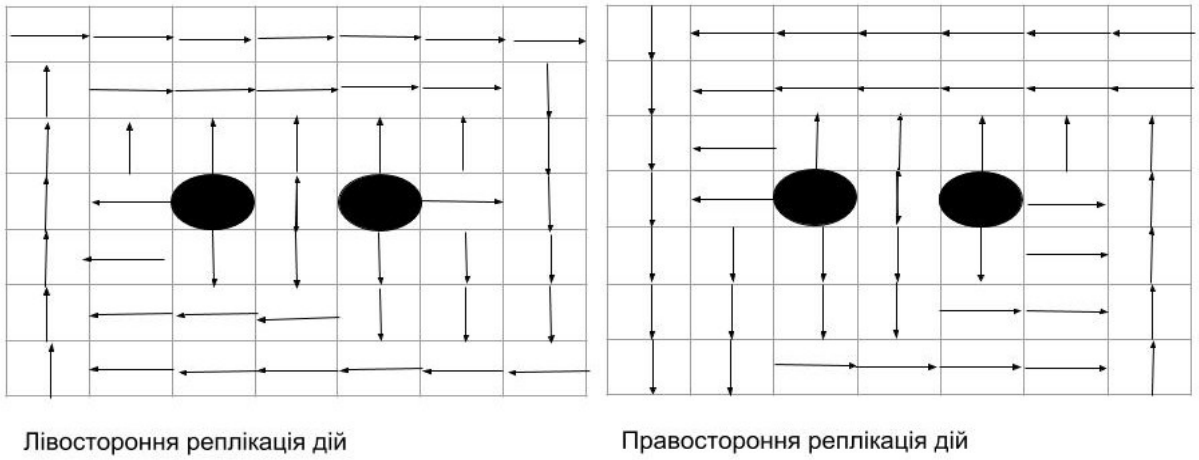
Позицію хижака зображено точкою у центральній частині рисунка. Стрілками позначено напрямки руху агентів-жертв для навченої стохастичної гри. Відслідковуючи напрямок стрілок, побачимо, що при лівосторонньому орієнтуванні (Side(i) = Left) сусідніх агентів має місце правостороння самоорганізація і, навпаки, при правосторонньому орієнтуванні (Side(i) = Right) – лівостороння самоорганізація системи.

У випадку з розміщенням хижака не в центрі картина змінюється, але загалом залишається подібною (рис. 2).



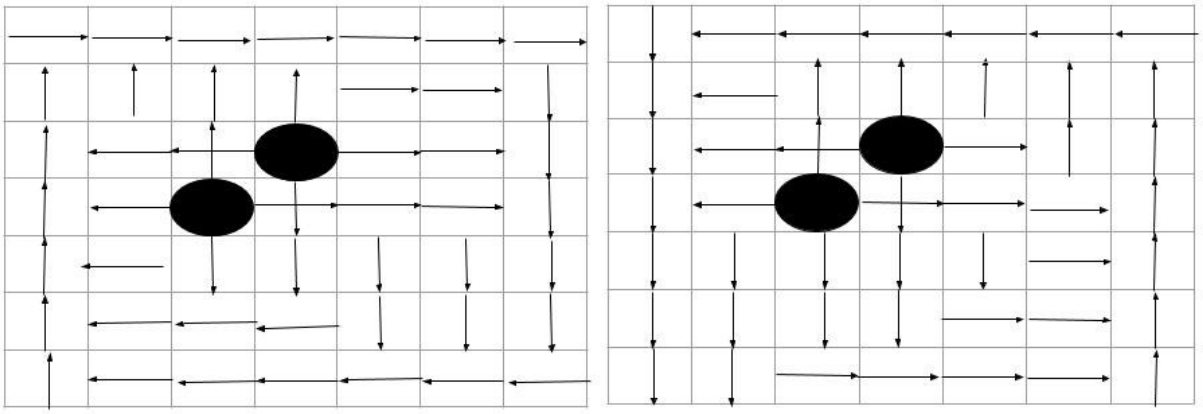

(рис. 2)

Якщо ж хижаків більше одного то агенти які розміщені між хижаками, а бо в дотику не вибирають однієї і тієї самої стратегії кожен раз а розділяють приблизно 50 на 50 напрямок руху.




(рис. 3)

Особливу увагу привертає агент по центру, стратегію якого є рух



Лівостороння реплікація дій

Правостороння реплікація дій

вверх або вниз з шансом 50 на 50 при моделюванні.

(рис. 4)

Знов таки агенти в дотику до декількох хижаків не можуть визначити переважаючий напрямок руху і вибирають обидва напрямки з шансами 50 на 50.

# ВИСНОВКИ

Завдяки локальній координації стратегії агентів даний розв`язок забезпечує самоорганізацію МАС „хижак – жертва”. Кожен гравець спостерігає дії сусідніх і отримує власні програші через не співпадіння, що заставляє його динамічно обирати стратегії з меншими штрафами. Динамічне обирання стратегій перетворює локально скоординовані дії гравців на глобальну координацію гри, як правостороння або лівостороння динаміки МАС, коли колектив гравці - жертви поводяться як цілісний організм.

Достовірність отриманих результатів підтверджується повторювальністю значень розрахованих характеристик стохастичної гри для різних послідовностей випадкових величин.

Також було перевірено збіжність системи для складніших випадків наприклад розміщення не в центрі поля або декількох хижаків.

# Список використаної літератури

[1] Aumann, R.J. (1985), On the Non-transferable Utility Value: A Comment on the Roth–Shafer Examples, Econometrica, 53, 667–677.

[2] Aumann, R.J. (1986), On the Non-transferable Utility Value: Rejoinder, Econometrica, 54, 985–989.

[3] Aumann, R.J. and M. Kurtz (1977), Power and Taxes, Econometrica, 45, 1137–1161.

[4] Bewley, T. and E. Kohlberg (1976), The Asymptotic Theory of Stochastic Games, Mathematics of Operations Research, 1, 197–208.

[5] Bewley, T. and E. Kohlberg (1978), On Stochastic Games with Stationary Optimal Strategies, Mathematics of Operations Research, 3, 104–125..

[6] Blackwell, D. (1962), Discrete Dynamic Programming, The Annals of Mathematical Statistics, 2, 724–738.

[7] Blackwell, D. and T.S. Ferguson (1968), The Big Match, Annals of Mathematical Statistics, 39, 159–168.

[8] Gilette, D. (1957), Stochastic Games with Zero Stop Probabilities, Contributions to the Theory of Games, 3, 179–187.

[9] Harsanyi, J. (1963), A Simplified Bargaining Model for the n-Person Cooperative

Game, International Economic Review, 4, 194–220.

[10] Kalai, A. and E. Kalai (2013), Cooperation in Strategic Games Revisited, Quarterly Journal of Economics, 128, 917–966.

[11] Kohlberg, E. (1974), Repeated Games with Absorbing States, Annals of Statistics, 4, 194–220.

[12] Mertens, J.F. and A. Neyman (1981), Stochastic Games, International Journal of Game Theory, 10, 53–66.

[13] Mertens J.F., S. Sorin, and S. Zamir (1994), Repeated Games, Core Discussion Papers 9420–9421–9422, Universit´e Catholique de Louvain la Neuve, Belgium.

[14] Milnor, J. (1952), Reasonable Outcomes for n-Person Games, The Rand Corporation, 18, 196.

[15] Nash, J. (1950), The Bargaining Problem, Econometrica, 18, 155–162.

[16] Nash, J. (1953), Two-Person Cooperative Games, Econometrica, 21, 128–140.

[17] Neyman, A. (1977), Values for Non-transferable Utility Games with a Continuum of Players, Technical Report No. 351, School of Operations Research and Industrial Engineering, Cornell University.

[18] Neyman, A. (2003), From Markov Chains to Stochastic Games, A. Neyman and S. Sorin (eds.), NATO ASI series 2003, Kluwer Academic Publishers, pp. 9–25.

[19] Neyman, A. (2003), Real Algebraic Tools in Stochastic Games, in Stochastic Games and Applications, A. Neyman and S. Sorin (eds.), NATO ASI series 2003, Kluwer Academic Publishers, pp. 58–75.

[20] Roth, A. (1985), On the Non-transferable Utility Value: A Reply to Aumann, Econometrica, 54, 981–984.

[21] Shapley, L. (1953), Stochastic Games, Proceedings of the National Academy of Sciences, U.S.A., 39, 1095–1100.

[22] Shapley, L. (1969), Utility and the Theory of Games, Editions du Centre Nattional de la Recherche Scientifique, 39, 251–263.

[23] Shapley, L. (1984), Mathematics 147 Game Theory, UCLA Department of Mathematics, 1984, 1987, 1988, 1990.

[24] Young, H.P. (1988), Individual Contribution and Just Compensation, in The Shapley Value: Essays in Honor of Lloyd S. Shapley, A. Roth (ed.), Cambridge University Press, pp. 267–278.

[25] Поспелов Д.А. Многоагентные системы – настоящее и будущее / Д.А. Поспелов // Информационные технологии и вычислительные системы. – 1998. – № 1. – С. 14–21.

[26] Городецкий В.И. Информационные технологии и многоагентные системы / В.И. Городецкий // Проблемы информатизации. – 1998. – Вып. 1. – С. 3–14.

[27] Тарасов В.Б. От многоагентных систем к интеллектуальным организациям: философия, психология, информатика / В.Б.Тарасов. – М.: Эдиториал УРСС, 2002. – 352 с.

[28] Швецов А.Н. Агентно-ориентированные системы: от формальных моделей

к промышленным приложениям / А.Н. Швецов. – Вологодский гос. технич. унив. – 101 с.: [Электрон. ресурс]. – http://www.ict.edu.ru/ft/005656/62333e1-st20.pdf.

[29] Wooldridge M. An Introduction to Multiagent Systems / M. Wooldridge. – John Wiley & Sons, 2002. – 366 pp.

[30] Jennings N. R. Commitments and Conventions: The Foundation of Coordination in Multi-Agent Systems / N. R. Jennings // The Knowledge Engineering Review. – 1993. Vol. 8 (3). – P. 223–250.

[31] Ashby W. R. Principles of the Self-Organizing Dynamic System / W. R. Ashby // Journal of General Psychology. – 1947. – Vol. 37. – P. 125–128.

[32] Grasse P.P. La reconstruction du nid et les coordinations inter-individuelles chez Bellicositermes natalensis et Cubitermes sp. La theorie de la stigmergie: essai d'interpretation des termites constructeurs / P.P. Grasse // Insect Society. – 1959. – Vol. 6. – P. 41–84.

[33] Пригожин И. Самоорганизация в неравновесных системах: От диссипативных структур к упорядоченности через флуктуации / И. Пригожин, Г. Николис. – М.: Мир, 1979. – 512 с.

[34] Хакен Г. Синергетика. Иерархия неустойчивостей в самоорганизующихся системах и устройствах / Г. Хакен. – М.: Мир, 1985.

[35] Хакен Г. Информация и самоорганизация. Макроскопический подход к сложным системам: Пер. с англ. / Г. Хакен. – М.: КомКнига, 2005. – 248 с.

[36] Эйген М. Гиперцикл. Принципы самоорганизации макромолекул / М.

Эйген, П. Шустер. – М.: Мир, 1982.- 260 с.

[37] Omicini A. SelfOrganisation & MAS An Introduction / A. Omicini, L. Gardelli: [Електрон. ресурс]. – http://unibo.lgardelli.com/teaching/2007-selforg-mas.pdf.

[38] Самоорганизация и многоагентные системы. I. Модели многоагентной самоорганизации / В. И. Городецкий // Изв. РАН. Теория и системы управления. – 2012. – N 2. – С. 92–120.

[39] Самоорганизация и многоагентные системы. II. Приложения и технология разработки / В. И. Городецкий // Изв. РАН . Теория и системы управления. – 2012. – N 3. – С. 55 –75.

[40] Доманский В.К. Стохастические игры / В.К. Доманский // Математические вопросы кибернетики. – 1988. – № 1. – С. 26–49.

[41] Вольтерра В. Математическая теория борьбы за существование. Пер. с франц. О. Н. Бондаренко / В. Вольтерра. – М.: Наука, 1976. – 287 с.

[42] Назин А.В. Адаптивный выбор вариантов: Рекуррентные алгоритмы / А.В. Назин, А.С. Позняк. – М.: Наука, 1986. – 288 с.

[43] Мулен Э. Теория игр с примерами из математической экономики / Э. Мулен. – М.: Мир, 1985. – 200 с.

[44] Граничин О.Н. Введение в методы стохастической аппроксимации и оценивания: Учеб. пособие / О.Н. Граничин. – СПб.: Издательство С.-Петербургского университета, 2003. – 131 с.

[45] Кравець П. Ігрова координація та самоорганізація стратегій у

мультиагентній моделі „хижак-жертва” / П. Кравець: [Електрон. ресурс]. – http://ena.lp.edu.ua/bitstream/ntb/34097/1/P.%20Kravets.pdf.

## Додаток

```
using System;
using System.IO;

namespace Game_Coordination
{
   class Program
   {
      static void Main(string[] args)
      {
         Random rnd = new Random();
         // initial time
         int n = 1;
         //field size
         int m = 3;
         // players number
         const int L = 9;
         // pred number;
         const int num = 1;
         // pred identifier;
         int [] s = new int [num];
         // strategies number
         const int N = 4;
         double[][] U = new double[N][]
         {
            new double[] {0}, //up
```

```
            new double[] {1}, //left
            new double[] {2}, //down
            new double[] {3}, //right
        };
        double[][] p = new  double[4][]
            {
              new double [L],
              new double [L],
              new double [L],
              new double [L],
            };
        double gamma = 1;
        double lambda = 0.5;
        // simplex E
        double E = 0.999/N;
        double A = 0.01;
        double B = 2;
        double d = 0.01;
        // max amount of steps
        const int nMax = 100000;
        const Boolean SideLeft = false;
        double[] loss = new double[L];
        double equl=0;
        double[] Z = new double[L];
        double K=0;
 string                                    path                                    =
Directory.GetParent(Directory.GetCurrentDirectory()).Parent.FullName                 +
"output.txt";
        using (StreamWriter file = File.CreateText(path))
            file.WriteLine("Started...");
```

```
for (var i = 0; i < num; i++)
{
    var w = rnd.NextDouble();
    s[i] = (int)Math.Floor((decimal)w * L );
}
for (var pl = 0; pl < L; pl++)
    for (var i = 0; i < N; i++)
        p[i][pl] = (1.0 / N);
for (var pl = 0; pl < L; pl++)
{
    equl = 0;
    for (var i = 0; i < N; i++)
    {
        if (Check(p, pl, i, m))
        { }
        else
        {
            p[i][pl] = 0;
        }
    };
    for (var i = 0; i < N; i++)
    {
        p[i][pl] = Math.Abs(p[i][pl]);
        equl += p[i][pl];
    }
    for (var i = 0; i < N; i++)
    {
        p[i][pl] /= equl;
    }
}
```

```
do
{
   //2.
   int[] playersStrategies = new int[L];
   for (var pl = 0; pl < L; pl++)
   {
      var w = rnd.NextDouble();
      double sum = 0;
      int strategyIndex = 0;
      for (var i = 0; i < N; i++)
      {
         sum += p[i][pl];
         if (sum > w)
         {
            strategyIndex = i;
            break;
         }
      }
      for (var i = 0; i < num; i++)
         if (pl == s[i])
         strategyIndex = 0;
      playersStrategies[pl] = strategyIndex;
   }
   //3.
   for (var pl = 0; pl < L; pl++)
      loss[pl] = 0;
   double mu = WhiteNoise(d);
   for (var pl = 0; pl < L; pl++)
   {
      loss[pl] = lambda * Penalty1(playersStrategies, pl,m) + (1 - lambda) *
```

```
Penalty2(playersStrategies, pl,SideLeft, s,m, num) + mu; // (1)
                for (var i = 0; i < num; i++)
                    if (pl == s[i]) loss[pl] = 0;


            }
            //4.
            double gamman = gamma * Math.Pow(n, -A);
            double En = E * Math.Pow(n, -B);
            //5.
            for (var pl = 0; pl < L; pl++)
            {
                equl = 0;
                p[playersStrategies[pl]][pl] = En * (p[playersStrategies[pl]][pl] -
gamman * loss[pl] * (1 - p[playersStrategies[pl]][pl]));
                p[playersStrategies[pl]][pl] = Math.Abs(p[playersStrategies[pl]][pl]);
                p[playersStrategies[pl]][pl] %= 1;
                for (var i = 0; i < N; i++)
                {
                    p[i][pl] = Math.Abs(p[i][pl]);
                    equl += p[i][pl];
                }
                for (var i = 0; i < N; i++)
                {
                    p[i][pl] /= equl;
                    p[i][pl] = Math.Abs(p[i][pl]);
                    if (n % 1000 == 0)
                    Console.WriteLine("n = {0}, pl = {1}, chance={2}", n, pl, p[i][pl]);
                }
        }
            //6.
```

```
    for (var pl = 0; pl < L; pl++)
        Z[pl] = ( Z[pl]*(n-1) + loss [pl] ) / n;
    double Zn = 0;
    for (var pl = 0; pl < L; pl++)
        Zn += Z[pl];
    Zn /= L;
    if (n % 100 == 0)
        using (StreamWriter file = new StreamWriter(path, append: true))
            file.WriteLine("n = {0},  Zn = {1}", n, Zn);
    // out Zn;
    int cor = 0;
    for (var pl = 0; pl < L; pl++)
        for (var i = 0; i < num; i++)
            cor += Coordinate(playersStrategies, pl, SideLeft, s[i],m,num);
    K = (K*(n-1) + cor) / n;
    if (n % 100 == 0)
        using (StreamWriter file = new StreamWriter(path, append: true))
            file.WriteLine("n = {0}, cor = {1}, K = {2}", n, cor, K);
    // out K
    //7.
    n++;
}
//8.
while (n <= nMax);
using (StreamWriter file = new StreamWriter(path, append: true))
    file.WriteLine("Finished");
for (var pl = 0; pl < L; pl++)
{
   int way = 0;
    for (var i = 0; i < N; i++)
```

```
            if (p[i][pl] > p[way][pl])
                way = i;
            using (StreamWriter file = new StreamWriter(path, append: true))
                file.WriteLine("player ={0}, way = {1}",pl, way);
        }
        using (StreamWriter file = new StreamWriter(path, append: true))
            file.WriteLine("up = 0, left = 1, down = 2, right = 3");
    }

    private static int Penalty1(int[] playersStrategies, int playerIndex, int fieldsize)
    {
        if (playerIndex < fieldsize)
            return NeighbourFR(playersStrategies, playerIndex, fieldsize);
        else if (playerIndex % fieldsize == 0)
            return NeighbourFC(playersStrategies, playerIndex, fieldsize);
        else if ((playerIndex + 1) % fieldsize == 0)
            return NeighbourLC(playersStrategies, playerIndex, fieldsize);
        else if ((playerIndex < Math.Pow(fieldsize, 2) && (playerIndex > (Math.Pow(fieldsize, 2) - fieldsize))))
            return NeighbourLR(playersStrategies, playerIndex, fieldsize);
        else
            return NeighbourC(playersStrategies, playerIndex, fieldsize);
    }

    private static int Penalty2(int[] playersStrategies, int playerIndex, Boolean left, int [] pred, int fieldsize, int num)
    {
        int loss = 0;
        for (var i = 0; i < num; i++)
            loss += (Penalty21(playersStrategies, playerIndex, pred[i], fieldsize) +
```

```
Penalty22(playersStrategies, playerIndex, left, pred[i], fieldsize));
            return loss;
        }

        private static int Penalty21(int[] playersStrategies, int playerIndex, int s, int fieldsize)
        {
           if ((playerIndex == (s - 1)) || (playerIndex == (s + 1)) || (playerIndex == (s - fieldsize)) || (playerIndex == (s + fieldsize)))
               if (playersStrategies[s] == playersStrategies[playerIndex])
                  return 0;
               else
                  return 1;
           else return 0;
        }

        private static int Penalty22(int[] playersStrategies, int playerIndex, Boolean left, int s, int fieldsize)
        {
           if ((playerIndex == (s - 1)) || (playerIndex == (s + 1)) || (playerIndex == (s - fieldsize)) || (playerIndex == (s + fieldsize)))
               return 0;
           else
               if (left == true)
               switch (playersStrategies[playerIndex])
               {
                  case 0:
                    if (playerIndex % fieldsize == 0)
                        return 0;
                     else
```

```
                return NeighbourL(playersStrategies, playerIndex);
            case 1:
                if ((playerIndex <= Math.Pow(fieldsize, 2) && (playerIndex >= (Math.Pow(fieldsize, 2) - fieldsize))))
                    return 0;
                else
                    return NeighbourD(playersStrategies, playerIndex, fieldsize);
            case 2:
                if ((playerIndex + 1) % fieldsize == 0)
                    return 0;
                else
                    return NeighbourR(playersStrategies, playerIndex);
            default:
                if (playerIndex < fieldsize)
                    return 0;
                else
                    return NeighbourU(playersStrategies, playerIndex, fieldsize);
        }
    else switch (playersStrategies[playerIndex])
        {
            case 0:
                if ((playerIndex + 1) % fieldsize == 0)
                    return 0;
                else
                    return NeighbourR(playersStrategies, playerIndex);
            case 1:
                if (playerIndex < fieldsize)
                    return 0;
                else
                    return NeighbourU(playersStrategies, playerIndex, fieldsize);
```

```
            case 2:
               if (playerIndex % fieldsize == 0)
                  return 0;
               else
                  return NeighbourL(playersStrategies, playerIndex);

            default:
               if ((playerIndex < Math.Pow(fieldsize, 2) && (playerIndex >= (Math.Pow(fieldsize, 2) - fieldsize))))
                  return 0;
               else
                  return NeighbourD(playersStrategies, playerIndex, fieldsize);

         }
      }

      private static int Coordinate (int[] playersStrategies, int playerIndex, Boolean left, int s,int m, int num)
      {
         int coor=0;
         int[] pred = { s };
         num =  pred.Length;
         coor = Penalty1(playersStrategies, playerIndex,m) + Penalty2(playersStrategies, playerIndex, left, pred,m,num);
         if (coor == 0)
            return 1;
         else
            return 0;
      }
```

```
    private static double WhiteNoise(double d)
    {
        Random rnd = new Random();
        double sum = 0;
        for (int i = 0; i < 12; i++)
            sum += rnd.NextDouble();
        sum -= 6;
        return Math.Sqrt(d) * sum;
    }
    private static int NeighbourFR(int []playersStrategies, int playerIndex, int
fieldsize)
    {
        if (playerIndex == 0)
            return NeighbourR(playersStrategies, playerIndex) +
NeighbourD(playersStrategies, playerIndex,fieldsize);
        else if (playerIndex == fieldsize - 1)
            return NeighbourL(playersStrategies, playerIndex) +
NeighbourD(playersStrategies, playerIndex,fieldsize);
        else
            return NeighbourR(playersStrategies, playerIndex) +
NeighbourD(playersStrategies, playerIndex,fieldsize)+NeighbourL(playersStrategies,
playerIndex);
    }
    private static int NeighbourFC(int[] playersStrategies, int playerIndex, int
fieldsize)
    {
      if (playerIndex == (fieldsize - 1)*fieldsize)
            return NeighbourL(playersStrategies, playerIndex) +
NeighbourU(playersStrategies, playerIndex,fieldsize);
        else
```

```
        return NeighbourR(playersStrategies, playerIndex) +
NeighbourD(playersStrategies, playerIndex,fieldsize) +
NeighbourU(playersStrategies, playerIndex,fieldsize);
    }
    private static int NeighbourLR(int[] playersStrategies, int playerIndex, int
fieldsize)
    {
      return NeighbourR(playersStrategies, playerIndex) +
NeighbourU(playersStrategies, playerIndex, fieldsize) +
NeighbourL(playersStrategies, playerIndex);
    }
    private static int NeighbourLC(int[] playersStrategies, int playerIndex, int
fieldsize)
    {
      if (playerIndex == Math.Pow(fieldsize, 2) - 1)
        return NeighbourL(playersStrategies, playerIndex) +
NeighbourU(playersStrategies, playerIndex, fieldsize);
      else
        return NeighbourU(playersStrategies, playerIndex,fieldsize) +
NeighbourD(playersStrategies, playerIndex, fieldsize) +
NeighbourL(playersStrategies, playerIndex);
    }
    private static int NeighbourC(int[] playersStrategies, int playerIndex, int
fieldsize)
    {
      return NeighbourU(playersStrategies, playerIndex, fieldsize) +
NeighbourD(playersStrategies, playerIndex, fieldsize) +
NeighbourL(playersStrategies, playerIndex) + NeighbourR(playersStrategies,
playerIndex);
    }
```

```
    private static int NeighbourU(int[] playersStrategies, int playerIndex, int
fieldsize)
    {
      if (playersStrategies[playerIndex] == playersStrategies[playerIndex -
fieldsize])
          return 0;
      else
          return 1;
    }
    private static int NeighbourD(int[] playersStrategies, int playerIndex, int
fieldsize)
    {
      if (playersStrategies[playerIndex] == playersStrategies[playerIndex +
fieldsize])
          return 0;
      else
          return 1;
    }
    private static int NeighbourL(int[] playersStrategies, int playerIndex)
    {
      if (playersStrategies[playerIndex] == playersStrategies[playerIndex - 1])
          return 0;
      else
          return 1;
    }
    private static int NeighbourR(int[] playersStrategies, int playerIndex)
    {
      if (playersStrategies[playerIndex] == playersStrategies[playerIndex + 1])
          return 0;
      else
```

```
        return 1;
    }
    private static Boolean Check(double [][] playerfield, int playerIndex, int Strategy, int fieldsize)
    {
       switch (Strategy)
       {
          case 0:
             if (playerIndex < fieldsize)
                return false;
             else
                return true;

          case 2:
             if ((playerIndex <= Math.Pow(fieldsize, 2) && (playerIndex >= (Math.Pow(fieldsize, 2) - fieldsize))))
                return false;
             else
                return true;
          case 3:
             if ((playerIndex + 1) % fieldsize == 0)
                return false;
             else
                return true;
          default:
             if (playerIndex % fieldsize == 0)
                return false;
             else
                return true;
```

```
            }
        }
    }
}
```