Одеський національний університет імені І. І. Мечникова
Факультет математики, фізики та інформаційних технологій
Кафедра оптимального керування і економічної кібернетики

Кваліфікаційна робота

на здобуття ступеня вищої освіти «бакалавр»

«Моделювання неантагоністичних конфліктів»

«Modeling Non-Antagonistic Conflicts»

Виконав: здобувач денної форми навчання
спеціальності 113 Прикладна математика
Освітня програма «Прикладна математика»
Кулік Віталій Олександрович

Керівник: доктор фіз.-мат. наук, проф. Кічмаренко О. Д. ___
Рецензент: канд. фіз.-мат. наук, доц. Страхов Є. М.

Рекомендовано до захисту:

Протокол засідання кафедри

№ ___ від _____ 2025 р.

Завідувач кафедри

_____ _____

Захищено на засіданні ЕК № _____

Протокол № ___ від _____ 2025 р.

Оцінка _____ / _____ / _____

Голова ЕК

_____ _____

Одеса — 2025 р.

ЗМІСТ

ВСТУП

Конфліктні ситуації, в яких інтереси різних сторін перетинаються, є невід'ємною частиною людської взаємодії в економіці, політиці, соціальній сфері та повсякденному житті. Ефективне управління такими ситуаціями та прийняття обґрунтованих рішень вимагають глибокого розуміння їхньої структури та динаміки. Теорія ігор надає потужний математичний апарат для формального аналізу стратегічної взаємодії між раціональними агентами, дозволяючи моделювати та прогнозувати результати конфліктів.

Особливий інтерес становлять неантагоністичні конфлікти, де інтереси гравців не є строго протилежними, і можливі сценарії, вигідні для всіх учасників, або, навпаки, такі, що призводять до загальних втрат. Моделювання саме таких ситуацій є актуальним, оскільки більшість реальних взаємодій (економічна конкуренція з можливістю кооперації, переговори, соціальні дилеми) мають неантагоністичний характер. Проте, класичні моделі теорії ігор часто спираються на припущення про повну інформованість гравців щодо параметрів гри, зокрема функцій виграшу. В реальності ж, економічні системи, соціальні процеси та технологічні взаємодії функціонують в умовах значної невизначеності, спричиненої коливаннями попиту, зміною технологій, непередбачуваною поведінкою ринків чи навіть неточністю вихідних даних. Ігнорування такої невизначеності може призвести до суттєвих помилок у прогнозах та прийнятті неоптимальних рішень. Це зумовлює нагальну потребу в розробці та дослідженні моделей неантагоністичних конфліктів, які здатні адекватно враховувати фактор невизначеності.

Сучасний стан досліджень у теорії ігор охоплює широкий спектр моделей. Починаючи з фундаментальних робіт Дж. фон Неймана, О. Моргенштерна та Дж. Неша, які заклали основи теорії матричних та безкоаліційних ігор і ввели ключове поняття рівноваги, теорія активно розвивалася. Для врахування неповної інформації Дж. Гарсані запропонував модель байєсівських ігор, де невизначеність моделюється за допомогою апріорних імовірнісних розподілів. Однак, на практиці отримати достовірну імовірнісну інформацію часто неможливо. Це стимулювало розвиток підходів, що не вимагають знання точних розподілів. До них належать класичні критерії

прийняття рішень в умовах невизначеності (Вальда, Гурвіца, Севіджа та ін.), а також більш сучасні методи, такі як теорія ігор з інтервальною невизначеністю та, зокрема, робастна оптимізація. Робастний підхід, запропонований та розвинений у працях Агассі, Берцимаса та інших дослідників, дозволяє гравцям оптимізувати свої стратегії відносно найгіршого можливого сценарію в межах заданої множини невизначеності, забезпечуючи гарантований результат. Саме цей напрям представляє значний інтерес для моделювання реальних неантагоністичних конфліктів.

Метою даної дипломної роботи є дослідження теоретичних основ та методів моделювання неантагоністичних конфліктів, з акцентом на аналіз ігрових ситуацій в умовах невизначеності параметрів, зокрема, шляхом застосування критеріїв прийняття рішень та підходів робастної оптимізації для знаходження рівноважних стратегій.

Об'єктом дослідження є процеси стратегічної взаємодії між учасниками в неантагоністичних конфліктах.

Предметом дослідження є математичні моделі безкоаліційних неантагоністичних ігор, методи їх аналізу та знаходження розв'язків в умовах повної інформації та в умовах інтервальної невизначеності параметрів з використанням критеріїв прийняття рішень та принципів робастної оптимізації.

Практичне значення отриманих результатів полягає у можливості застосування розроблених підходів та моделей для аналізу широкого кола реальних економічних, управлінських та соціальних ситуацій, де параметри взаємодії не є точно відомими. Це дозволяє підвищити обґрунтованість прийнятих рішень та стійкість стратегій до непередбачуваних змін у зовнішньому середовищі.

Структура роботи. Основна частина роботи складається з двох розділів. У першому розділі наведено загальні теоретичні відомості з теорії ігор - класифікацію ігор, визначення безкоаліційної гри, відмінності між антагоністичними та неантагоністичними іграми, розглянуто поняття оптимальності, стійкості та рівноваги Неша, а також основи мішаних стратегій та біматричних ігор. У другому розділі представлено підходи до моделювання ігор з неповною інформацією. Розглянуто ігри з інтервальною

невизначеністю виграшів, проаналізовано критерії прийняття рішень для детермінізації невизначеності. Основну увагу приділено робастному підходу: його формалізації, теоремі існування робастно-оптимізаційної рівноваги, обчислювальним аспектам її знаходження та програмній реалізації. Ефективність робастного підходу ілюструється на прикладах гри інспекції та задачі "безбілетника".

РОЗДІЛ 1

ТЕОРЕТИЧНІ ОСНОВИ МОДЕЛЮВАННЯ НЕАНТАГОНІСТИЧНИХ КОНФЛІКТІВ

1.1 Класифікація ігор

У теорії ігор виділяють дві основні частини: теорію безкоаліційних ігор та теорію кооперативних ігор. У безкоаліційних іграх основною одиницею аналізу є індивідуальний учасник, який прагне максимізувати свою вигоду відповідно до чітко визначених правил і можливостей. Теорія безкоаліційних ігор базується на двох припущеннях:

- **максимізації** (кожен економічний агент раціональний і має чітке уявлення про світ);
- **узгодженості** (уявлення агента і його очікування щодо поведінки інших агентів є правильними).

У теорії кооперативних ігор основною одиницею аналізу, як правило, є група учасників або коаліція; якщо гра визначена, то частиною цього визначення є опис того, який виграш може отримати кожна коаліція гравців (без вказівки на те, які дії повинен здійснювати кожен конкретний гравець коаліції для отримання максимального виграшу).

Ігри також можна класифікувати за:

- кількістю гравців;
- кількістю стратегій;
- характером виграшу;
- кількістю ходів;
- станом інформації тощо.

Залежно від кількості стратегій ігри бувають:

- **скінченними** – усі гравці мають скінченну кількість можливих стратегій;
- **нескінченними** – хоча б один з гравців має нескінченну кількість можливих стратегій.

За характером виграшів ігри поділяються на:

- **ігри з нульовою сумою** – загальний капітал усіх гравців не змінюється, а лише перерозподіляється між ними; сума виграшів усіх гравців дорівнює нулю;
- **ігри з ненульовою сумою**.

За видом функцій виграшу ігри поділяються на:

- матричні;
- біматричні;
- неперервні;
- опуклі;
- сепарабельні;
- ігри типу дуелей тощо.

Також розрізняють ігри з досконалою та недосконалою інформацією.

Послідовна гра називається грою з досконалою інформацією, якщо гравці роблять свої ходи по черзі (не одночасно), і кожен гравець знає всі дії гравців, які робили свій хід до нього, у кожній точці. Технічно це означає, що кожна інформаційна множина містить лише один вузол (одну вершину). Усі інші ігри називаються іграми з недосконалою інформацією.

1.2 Безкоаліційні ігри

У практичній діяльності досить часто доводиться розглядати різноманітні явища та ситуації, в яких бере участь кілька сторін, що мають різні інтереси та застосовують певні дії для досягнення своїх цілей. Такі ситуації прийнято називати конфліктами. Типова конфліктна ситуація характеризується трьома основними складовими:

1) зацікавленими сторонами,
2) можливими діями цих сторін,
3) інтересами сторін.

Конфлікти, взяті з реального життя, як правило, досить складні, тому зазвичай будують спрощену математичну модель конфлікту, яку називають

грою. Безкоаліційною грою називають трійку об'єктів

$$\Gamma =< I, \{X_i\}_{i\in I}, \{f_i(x)\}_{i\in I} >, \tag{1.1}$$

де I — скінченна множина, елементи якої називаються гравцями (зазвичай приймають $I = \{1,2,\ldots,n\}$), $X_i = \{x_i\}$ — довільні попарно неперетинні множини, елементи яких називаються стратегіями відповідних гравців $i \in I$. Упорядковані набори $\{x_i\}_{i\in I}$ стратегій гравців, тобто елементи декартового добутку

$$X = \prod_{i\in I} X_i$$

називаються ситуаціями у грі Γ; відповідні кожному гравцю $i \in I$ обмежені функції

$$f_i : X \to \mathbb{R}$$

називаються функціями виграшу цього гравця, а їхні значення на окремих ситуаціях — його виграшами у цих ситуаціях.

Залежно від характеру взаємодії інтересів гравців, безкоаліційні ігри поділяються на антагоністичні та неантагоністичні.

Антагоністичні ігри (також відомі як ігри з постійною сумою, а у випадку, коли ця сума дорівнює нулю – ігри з нульовою сумою) характеризуються тим, що інтереси гравців є прямо протилежними. У таких іграх сума виграшів усіх гравців є сталою величиною для будь-якої ситуації: $\sum_{i\in I} f_i(x) = C$ для всіх $x \in X$. Це означає, що виграш одного гравця (або групи гравців) можливий лише за рахунок програшу іншого гравця (або групи гравців). У випадку гри двох осіб ($I = \{1,2\}$) це означає, що $f_1(x) + f_2(x) = C$. Якщо константа $C = 0$, то $f_1(x) = -f_2(x)$, і гра називається грою з нульовою сумою. В такій грі те, що виграє один гравець, повністю програє інший. Метою кожного гравця є максимізація власного виграшу, що в грі двох осіб з нульовою сумою автоматично означає мінімізацію виграшу суперника. Класичним прикладом є шахи або покер (де виграш одного означає програш іншого на ту ж суму).

Неантагоністичні ігри (або ігри з непостійною/змінною сумою) представляють ширший і більш поширений клас ігор, де інтереси гравців не є строго протилежними. У таких іграх сума виграшів гравців не є

сталою величиною і може змінюватися залежно від обраних ними стратегій ($\sum_{i\in I} f_i(x)$ не є константою). Це означає, що можливі ситуації, коли всі гравці одночасно виграють (наприклад, в результаті успішної співпраці, навіть якщо вона неформальна), або всі програють, або виграш одних не обов'язково означає еквівалентний програш інших. Гравці можуть мати як конфліктні, так і спільні інтереси. Наприклад, у "дилемі в'язня"обидва гравці можуть отримати кращий результат, якщо співпрацюватимуть, але мають стимули зрадити один одного. Більшість реальних економічних, політичних та соціальних взаємодій моделюються саме як неантагоністичні ігри.

Загальна теорія безкоаліційних ігор, що розглядається в цьому розділі, охоплює як антагоністичні, так і неантагоністичні ігри. Однак для антагоністичних ігор, особливо для ігор двох осіб з нульовою сумою, існують специфічні та більш розроблені методи аналізу, такі як теорема про мінімакс. У неантагоністичних іграх поняття розв'язку є складнішим і часто пов'язане з концепцією рівноваги Неша.

На змістовному рівні перелічені тут компоненти безкоаліційної гри Г становлять звід "правил"цієї гри. Гравці можуть об'єднуватися в коаліції — довільні підмножини $K \in I$. Зокрема, коаліція може складатися з одного гравця або з усіх гравців I. Для кожної коаліції $K = \{i_1, \ldots, i_k\}$ можна розглядати її коаліційну стратегію $x_K = (x_{i_1}, \ldots, x_{i_k})$, що представляє собою можливі дії цієї коаліції. У зв'язку з цим стратегії i-го гравця називають його індивідуальними стратегіями. Множину всіх коаліційних стратегій коаліції K у грі Г позначимо через X_K. Вона являє собою декартовий добуток

$$X_K = \prod_{i\in K} X_i.$$

Виграш коаліції K у ситуації $x \in X$ позначимо
$f_K(x) = (f_{i_1}(x), f_{i_2}(x), \ldots, f_{i_k}(x)) \in \mathbb{R}^k$. Введення та використання тут поняття коаліції є суто допоміжним і жодною мірою не означає переходу до коаліційних ігор. Окрема партія гри розвивається наступним чином. Кожен гравець $i \in I$ обирає свою індивідуальну стратегію $x_i \in X_i$. В результаті складається ситуація $x \in X$, $x = (x_1, x_2, \ldots, x_n)$. Значення функції $f_i(x)$ на даній ситуації і становить виграш i-го гравця. Особливість безкоаліційної

гри полягає в тому, що гравці здійснюють вибір своїх стратегій одночасно незалежно один від одного, тоді як у коаліційних іграх вони можуть попередньо домовлятися про вибір своїх стратегій, а також, якщо це дозволено правилами, перерозподіляти виграші між собою (в іграх з побічними платежами). Розглядають також випадок, коли не всі ситуації є допустимими (за правилами гри), тоді говорять про гру із забороненими ситуаціями, в якій $X \subset X_1 \times X_2 \times \ldots \times X_n$.Кожна сторона реального конфлікту прагне реалізувати свої інтереси якнайкраще для себе (з найменшими витратами та найбільшим прибутком). Це прагнення виражається в іграх за допомогою введення поняття оптимальності. Під принципом оптимальності для певного класу безкоаліційних ігор $\mathbb{G}$ (який збігається з класом усіх безкоаліційних ігор або становить його частину) розуміється відображення φ, яке ставить у відповідність кожній грі $\Gamma \in \mathbb{G}$ певну підмножину множини її ситуацій X та виражає в математичній формі ті чи інші змістовні риси (інтуїтивного) уявлення про оптимальність. Ці риси можуть проявлятися у вигляді вигідності, стійкості або справедливості. При цьому ситуації $x \in \varphi\Gamma$ називаються реалізаціями принципу оптимальності φ у грі Γ або розв'язками гри Γ у сенсі принципу оптимальності φ. Якщо $\varphi\Gamma \neq \emptyset$, то принцип φ називається реалізованим у грі Γ, а гра Γ — розв'язною у сенсі принципу φ. На відміну від конфліктів з реального життя, в іграх передбачається, що всі гравці діють найбільш розумно та раціонально, і беззастережно дотримуються "правил"гри. Оскільки інтуїтивні уявлення про оптимальність можуть бути досить різноманітними, то принципи оптимальності можуть вводитися по-різному. Проте, наприклад, для гри з одним гравцем $I = \{1\}$, що являє собою по суті дійсну функцію $f_1 : X = X_1 \to \mathbb{R}$, природно буде вводити принцип оптимальності, що максимізує виграш цього гравця, оскільки жодні інші міркування, якими він міг би керуватися, у такій моделі не представлені. У цьому випадку $\varphi\Gamma = \operatorname{argmax} f_1(x)$. Тоді будь-який розумний принцип оптимальності у безкоаліційній грі Γ повинен спиратися на вищеописаний принцип оптимізації виграшу і перетворюватися на нього, якщо Γ зводиться до гри з одним гравцем. Прикладом слугує принцип φ, такий що $x^* \in \varphi\Gamma$, якщо

$$f_i(x^*) = \max_{x \in X} f_i(x) \qquad \forall i \in I.$$

Практично цей принцип виявляється нереалізованим для більшості безкоа-

ліційних ігор. Водночас він відповідає деяким уявленням про оптимальність, хоча й виражає їх у надмірно посиленій і тому в суперечливій формі. Він відображає ідею вигідності, оскільки в оптимальній ситуації кожен гравець отримає більше, ніж у будь-якій іншій; ідею стійкості, оскільки жоден з гравців не зацікавлений у заміні оптимальної ситуації на будь-яку іншу; та ідею справедливості, оскільки він задовольняє запити всіх гравців рівною мірою. Виходячи з цього, оптимальними вважають такі ситуації, в яких, з урахуванням їхніх стратегічних можливостей, гравці отримують найбільший виграш і не зацікавлені у зміні ситуації. Для формального опису цих ідей вводяться такі позначення та поняття.Нехай $x \in X$ — деяка ситуація у грі Γ, $K \subset I$ — деяка коаліція в ній, x'_K — деяка її коаліційна стратегія. Через $x||x'_K$ позначається ситуація, отримана із ситуації x заміною наявних у ній індивідуальних стратегій гравців з K на їхні індивідуальні стратегії, що беруть участь в утворенні коаліційної стратегії x'_K.

Нехай у грі Γ коаліція $K \subset I$, ситуації x,$y \in X$. Кажуть, що ситуація x K-домінує ситуацію y, якщо виконуються такі умови:

1) Ефективність для K: ситуація y має вигляд $x||y_K$, де $y_K \in X_K$.
2) Перевага для K: $f_i(y) \leq f_i(x) \forall i \in I, f_K(y) \neq f_K(x)$.

Кажуть, що ситуація x строго K-домінує ситуацію y, якщо виконується умова ефективності для K і умова суворої переваги для K: $f_i(y) < f_i(x) \forall i \in I$. Принципи оптимальності, засновані на K-домінуванні, — це принципи стійкості; засновані на строгому K-домінуванні — принципи рівноваги. У безкоаліційній грі Γ ситуація x^* називається K-стійкою, якщо не існує K-домінуючої її ситуації. K-стійкість ситуації часто називається також її оптимальністю для K за Парето. Просто оптимальними за Парето називаються I-стійкі ситуації. K-стійкість ситуації x^* може бути виражена так: $\forall x \in X$ x не K-домінує $x^* \Leftrightarrow$

$$\forall x \in X \quad \overline{((f_i(x) \geq f_i(x^*)) \wedge (f(x) \neq f(x^*)) \wedge (x^* = x||x^*_K))} \quad \Leftrightarrow$$
$$\forall x \in X \quad \overline{(x^* = x||x^*_K)} \vee \overline{(\forall i \in K f_i(x) \geq f_i(x^*))} \vee \overline{(f(x) \neq f(x^*))} \quad \Leftrightarrow$$
$$\forall x \in X \quad ((x^* = x||x^*_K) \Rightarrow ((\exists i \in K(f_i(x) < f_i(x||x^*_K))) \vee (f(x) = f(x||x^*_K))) \quad \Leftrightarrow$$
$$\forall x_K \in X_K \quad (\forall i \in K : f_i(x^*||x_K) = f_i(x^*)) \vee (\exists i \in K : f_i(x^*||x_K) < f_i(x^*)) \quad \Leftrightarrow$$
$$\forall x_K \in X_K \quad (\exists i \in K : f_i(x^*||x_K) > f_i(x^*)) \Rightarrow (\exists j \in K : f_j(x^*||x_K) < f_j(x^*))).$$

Остання нотація означає, що в K-стійкій ситуації який-небудь гравець коаліції K може збільшити свій виграш тільки за рахунок зменшення виграшу іншого гравця цієї коаліції. Ситуація x^* буде i-стійкою (їх ще називають i-прийнятними), якщо $\forall x_i \in X_i : f_i(x^*||x_i) \leq f_i(x^*)$, тобто прийнятною для гравця буде така ситуація, в якій зміна ним своєї стратегії не може збільшити його виграшу. Вводиться також поняття стійкості для множини коаліцій $C = \{K \subset I\}$. Ситуація x^* називається C-стійкою, якщо вона є K-стійкою для будь-якої коаліції $K \in C$. Розглянемо множину коаліцій $I = \{\{1\}, \{2\}, \ldots, \{n\}\}$, що складаються з одного гравця, яка включає всіх гравців. I-стійкі ситуації називаються рівноважними за Нешем. Рівноважність ситуації x^* за Нешем означає, що у грі Г$\forall\ i \in I$ та $\forall\ x_i \in X_i$ виконується нерівність $f_i(x^*||x_i) \leq f_i(x^*)$. Змістовно ситуацію рівноваги за Нешем можна інтерпретувати як таку угоду між гравцями, яку жодному з них не вигідно порушувати. У безкоаліційній грі Г ситуація x^* називається K-рівноважною, якщо не існує строго K-домінуючої її ситуації. K-рівноважність ситуації x^* означає, що для будь-якого $x_K \in X_K$ $\exists\ i \in K : f_i(x^*||x_K) \leq f_i(x^*)$. Таким чином, якщо K складається з єдиного гравця i, поняття стійкості збігатиметься з поняттям рівноваги. В основному у безкоаліційних іграх шукають ситуації рівноваги за Нешем, які називають просто рівноважними. Процес пошуку таких ситуацій і називають розв'язком гри.

1.3 Біматричні ігри

Біматричною грою Г називається безкоаліційна гра двох осіб, у якій множини стратегій першого та другого гравця є скінченними. Нехай гравець 1 має m, а 2 – n стратегій, тобто $X_1 = \{1,2,\ldots,m\},\ X_2 = \{1,2,\ldots,n\}$. Значення виграшів гравців тоді можна записати у вигляді двох матриць

розмірності $m \times n$:

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}, \quad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{pmatrix}.$$

Тобто при виборі першим гравцем стратегії i, а другим гравцем стратегії j (у ситуації (i,j)), гравець 1 отримує виграш a_{ij}, а гравець 2 – b_{ij}. Оскільки матриці $\mathbf{A}$ та $\mathbf{B}$ повністю визначають гру, то її можна записати як $\Gamma =< \mathbf{A},\mathbf{B} >$. Варто зазначити, що при $\mathbf{B} = -\mathbf{A}$ біматрична гра $< \mathbf{A},\mathbf{B} >$ перетворюється на матричну гру $\mathbf{A}$. Рівноважною в грі $< \mathbf{A},\mathbf{B} >$ буде така ситуація (k,l), що

$$\begin{aligned} a_{il} &\le a_{kl} \quad \forall i = \overline{1,m}, \\ b_{kj} &\le b_{kl} \quad \forall j = \overline{1,n}. \end{aligned}$$

Таким чином, для пошуку рівноважних ситуацій необхідно в матриці $\mathbf{A}$ знайти елементи, на яких досягається максимум у стовпці $k_j = \max\limits_i a_{ij},\ j = \overline{1,n}$, а в матриці $\mathbf{B}$ знайти елементи, на яких досягається максимум у рядку $l_i = \max\limits_j b_{ij},\ i = \overline{1,m}$. Якщо існують такі ситуації, в яких ці максимуми досягаються на одних і тих же елементах, тобто $k = i$ та $j = l$, то ситуації (k,l) будуть рівноважними.

1.4 Мішані розширення безкоаліційних ігор

Вже на прикладі матричних ігор було видно, що ситуації рівноваги в чистих стратегіях існують не в усіх іграх, і знайти розв'язок іноді вдається лише в мішаних стратегіях. Тому і для загальних безкоаліційних ігор має сенс шукати ситуації рівноваги в мішаних стратегіях. Розглянемо скінченну

(зі скінченними множинами стратегій) безкоаліційну гру

$$\Gamma =< I,\{X_i\}_{i\in I},\{f_i(x)\}_{i\in I} > .$$

Нехай ξ_i — довільна мішана стратегія гравця i, тобто деякий імовірнісний розподіл на множині чистих стратегій X_i. Ймовірність, яку розподіл ξ_i приписує чистій стратегії x_i, позначимо через $\xi_i(x_i)$. Множину всіх мішаних стратегій гравця i позначимо через Ξ_i. Вважатимемо, що мішані стратегії всіх гравців $i \in I = \{1,\ldots,n\}$ є незалежними в сукупності розподілами, тобто ймовірність появи ситуації $s = (s_1,\ldots,s_n)$ дорівнює добутку ймовірностей вибору стратегій, що її складають $\xi_1(x_1)\xi_2(x_2)\ldots\xi_n(x_n)$. Тоді ситуацією в мішаних стратегіях називають імовірнісний розподіл ξ на множині всіх ситуацій, що задається співвідношенням

$$\xi(x) = \xi(x_1,\ldots,x_n) = \xi_1(x_1)\ldots\xi_n(x_n).$$

Ситуація гри Γ у мішаних стратегіях реалізує різні ситуації з деякими ймовірностями, тому значення функції виграшу кожного з гравців стає випадковою величиною. За виграш i-го гравця в ситуації в мішаних стратегіях приймають математичне сподівання цієї величини, тобто

$$f_i(\xi) = \sum_{x\in X} f_i(x)\xi(x) = \sum_{x_1\in X_1}\ldots\sum_{x_n\in X_n} f_i(x_1,\ldots,x_n)\prod_{i=1}^{n}\xi_i(x_i). \qquad (1.2)$$

Зазначимо також, що

$$f_i(\xi||x_j^0) = \sum_{x_1\in X_1}\ldots\sum_{x_{j-1}\in X_{j-1}}\sum_{x_{j+1}\in X_{j+1}}\ldots\sum_{x_n\in X_n} f_i(\xi||x_j^0)\prod_{i=1,i\neq j}^{n}\xi_i(x_i).$$

Гра

$$\Gamma^* =< I,\{\Xi_i\}_{i\in I},\{f_i(\xi)\}_{i\in I} >,$$

в якій I — множина гравців, Ξ_i — множина стратегій гравця i, а функція виграшу визначається рівністю (1.2), називається мішаним розширенням гри Γ.

Лема 1.1. *Якою б не була ситуація в мішаних стратегіях* $\xi = (\xi_1,\ldots,\xi_n)$,

будь-який гравець i має таку чисту стратегію x_i^0, що одночасно виконуються дві нерівності

$$\xi_i(x_i^0) > 0 \text{ та } f_i(\xi||x_i^0) \leq f_i(\xi).$$

Доведення. Припустимо протилежне, тобто для всіх x_i гравця i, таких що $\xi_i(x_i) > 0$, виконується

$$f_i(\xi||x_i) > f_i(\xi).$$

Тоді для всіх таких стратегій

$$f_i(\xi||x_i)\xi_i(x_i) > f_i(\xi)\xi_i(x_i).$$

Для всіх інших стратегій $\xi_i(x_i) = 0$, отже

$$f_i(\xi||x_i)\xi_i(x_i) = f_i(\xi)\xi_i(x_i) = 0.$$

Тоді

$$\sum_{x_i \in X_i} f_i(\xi||x_i)\xi_i(x_i) > \sum_{x_i \in X_i} f_i(\xi)\xi_i(x_i).$$

Але це означає, що $f_i(\xi) > f_i(\xi)$, чого не може бути. Отримане протиріччя вказує на існування шуканої чистої стратегії гравця i. □

Ситуацією рівноваги гри Γ у мішаних стратегіях називається ситуація рівноваги її мішаного розширення Γ^*. Ситуація ξ^* буде ситуацією рівноваги в Γ^*, якщо для будь-якого гравця i та для будь-якої його мішаної стратегії ξ_i має місце нерівність

$$f_i(\xi^*||\xi_i) \leq f_i(\xi^*).$$

Теорема 1.1. *Для того, щоб ситуація ξ^* у грі Γ була ситуацією рівноваги цієї гри в мішаних стратегіях, необхідно і достатньо, щоб для будь-якого гравця i та будь-якої його чистої стратегії x_i виконувалося*

$$f_i(\xi^*||x_i) \leq f_i(\xi^*). \tag{1.3}$$

Доведення. Необхідність Оскільки чиста стратегія є окремим випадком мішаної, то нерівність (1.3) безпосередньо випливає з визначення рівноважної

ситуації.

Достатність Візьмемо довільну мішану стратегію ξ_i гравця i, домножимо нерівність (1.3) на $\xi_i(x_i)$ та просумуємо по всіх $x_i \in X_i$

$$f_i(\xi^*||x_i) = \sum_{x_i \in X_i} f_i(\xi^*||x_i)\xi_i(x_i) \leq \sum_{x_i \in X_i} \xi_i(x_i) f_i(\xi^*) = f_i(\xi^*) \sum_{x_i \in X_i} \xi_i(x_i) = f_i(\xi^*).$$

З отриманої нерівності випливає рівноважність ситуації ξ^*. □

Теорема 1.2 (Неша)**.** *У кожній безкоаліційній грі*

$$\Gamma =< I, \{X_i\}_{i \in I}, \{f_i(x)\}_{i \in I} >$$

існує хоча б одна ситуація рівноваги.

Доведення. Якщо гравець i має в Γ m_i чистих стратегій, то множина його мішаних стратегій Ξ_i являє собою $(m_i - 1)$-вимірний симплекс. Позначимо його через $S^{(i)}$. Тоді будь-яку ситуацію в мішаних стратегіях $\xi = (\xi_1, \ldots, \xi_n)$ можна розглядати як точку декартового добутку $S^{(1)} \times \ldots \times S^{(n)}$, яке є опуклою замкнутою обмеженою підмножиною $(m_1 + \ldots + m_n - n)$-вимірного простору. Введемо функцію $\varphi_{ij}(\xi) = \max\{0, f_i(\xi||x_i^{(j)}) - f_i(\xi)\}$ для будь-якої ситуації ξ та будь-якої чистої стратегії $x_i^{(j)} \in X_i$ гравця i. Вона показує збільшення виграшу гравця i у ситуації ξ, що відбувається за рахунок зміни його стратегії ξ_i, яка входить до цієї ситуації, на деяку чисту стратегію $x_i^{(j)}$. Складемо тепер для всіх $i = \overline{1,n}$ та $j = \overline{1,m_i}$ числа виду

$$\frac{\xi_i(x_i^{(j)}) + \varphi_{ij}(\xi)}{1 + \sum_{j=1}^{m_i} \varphi_{ij}(\xi)}. \tag{1.4}$$

Оскільки всі $\varphi_{ij}(\xi) \geq 0$, то всі ці числа також є невід'ємними, а кожна сума виду

$$\sum_{j=1}^{m_i} \frac{\xi_i(x_i^{(j)}) + \varphi_{ij}(\xi)}{1 + \sum_{j=1}^{m_i} \varphi_{ij}(\xi)} = 1.$$

Отже, при фіксованих ξ та i дроби (1.4) можна розуміти як ймовірності відповідних чистих стратегій $x_i^{(j)}$ гравця i, а кожен їхній набір для всіх чистих стратегій $x_i^{(j)}$ — як мішану стратегію гравця i. Сукупність дробів

(1.4) для всіх гравців визначає систему мішаних стратегій усіх гравців, тобто ситуацію в грі Г. Ця ситуація є функцією вихідної ситуації ξ, позначимо її через $g(\xi)$. Функція g здійснює перетворення опуклої компактної множини всіх ситуацій Ξ у себе. Крім того, вона є неперервною функцією ξ, оскільки $\varphi_{ij}(\xi)$ неперервна, а знаменник $1 + \sum\limits_{j=1}^{m_i} \varphi_{ij}(\xi) \geq 1$. Це означає, що g задовольняє всі умови теореми про нерухому точку, згідно з якою неперервне перетворення g опуклої підмножини скінченновимірного простору в себе має хоча б одну нерухому точку, тобто таку точку ξ^0, що $g(\xi^0) = \xi^0$. Нехай ξ^0 — нерухома точка функції g. Це означає, що для всіх i та j

$$\xi_i^0(x_i^{(j)}) = \frac{\xi_i^0(x_i^{(j)}) + \varphi_{ij}(\xi^0)}{1 + \sum_{j=1}^{m_i} \varphi_{ij}(\xi^0)}. \tag{1.5}$$

Раніше було показано, що для будь-якого гравця i існує така чиста стратегія x_i^0 цього гравця, що $\xi_i^0(x_i^0) > 0$ та $\varphi_{i0}(\xi^0) = 0$. Для цієї стратегії рівність (1.5) записується як

$$\xi_i^0(x_i^0) = \frac{\xi_i^0(x_i^0) + \varphi_{i0}(\xi^0)}{1 + \sum_{j=1}^{m_i} \varphi_{ij}(\xi^0)},$$

звідки

$$\xi_i^0(x_i^0) + \xi_i^0(x_i^0) \sum_{j=1}^{m_i} \varphi_{ij}(\xi^0) = \xi_i^0(x_i^0) + \varphi_{i0}(\xi^0),$$

$$\xi_i^0(x_i^0) \sum_{j=1}^{m_i} \varphi_{ij}(\xi^0) = \varphi_{i0}(\xi^0) = 0.$$

А оскільки $\xi_i^0(x_i^0) > 0$, то

$$\sum_{j=1}^{m_i} \varphi_{ij}(\xi^0) = 0.$$

А оскільки всі $\varphi_{ij}(\xi^0) \geq 0$, то вони $\varphi_{ij}(\xi^0) = 0$. Це означає, що будь-яка різниця

$$f_i(\xi || x_i^{(j)}) - f_i(\xi) \leq 0,$$

тобто

$$f_i(\xi || x_i^{(j)}) \leq f_i(\xi).$$

За попередньою теоремою отримуємо, що ξ^0 — ситуація рівноваги. $\square$

Зауваження 1.1. Варто зазначити, що теорема Неша лише гарантує існування ситуацій рівноваги, але не дає алгоритмів їх знаходження. Це пов'язано з використанням теореми про нерухому точку, яка також не є конструктивною.

РОЗДІЛ 2

БЕЗКОАЛІЦІЙНІ ІГРИ ЗА УМОВ НЕВИЗНАЧЕНОСТІ

Класична теорія безкоаліційних ігор, зокрема результати Неша щодо існування рівноваги, стосується переважно не-кооперативних, одночасних, одноразових, скінченних ігор з повною інформацією. Це означає, що всі параметри гри, включно з функціями виграшу гравців, є загальновідомими. Однак, у реальних ігрових ситуаціях гравці часто стикаються з невизначеністю щодо деяких аспектів структури гри, таких як функції виграшу.

Гарсані [1] змоделював ці ігри з неповною інформацією як так звані "байєсівські ігри". Він визначив "тип"гравця як кодування інформації, доступної цьому гравцеві, включаючи знання власної функції виграшу та переконання щодо функцій виграшу інших гравців. Таким чином, він використовував простори типів для моделювання ігор з неповною інформацією, в яких деякі гравці можуть мати приватну інформацію. Він припустив, що гравці поділяють спільний апріорний розподіл ймовірностей над простором типів, що є загальновідомим. Гарсані запропонував, щоб кожен гравець використовував цей апріорний розподіл ймовірностей разом зі своїм типом для виведення умовного розподілу ймовірностей для параметрів, що залишаються йому невідомими. Крім того, він припустив, що метою кожного гравця буде, використовуючи таким чином байєсівський підхід, максимізувати свій очікуваний виграш відносно як вищезгаданого умовного розподілу ймовірностей, так і мішаних стратегій гравців.

У цих рамках Гарсані розширив результат Неша на ігри з неповною інформацією. Зокрема, він показав, що будь-яка байєсівська гра еквівалентна грі в розгорнутій формі з повною, але недосконалою інформацією. Ця гра в розгорнутій формі, в свою чергу, має представлення у статичній формі. Використовуючи результат існування рівноваги, більш загальний, ніж у Неша, Гарсані довів існування рівноваг, які він назвав "байєсівськими рівновагами у байєсівських іграх. За свою роботу над іграми з неповною інформацією він отримав Нобелівську премію з економіки 1994 року разом

із Нешем та Зельтеном.

Робота Гарсані послаблює припущення, що всі параметри, які впливають на виграші гравців, відомі з певністю. Його модельна техніка по суті аналогічна підходу стохастичного програмування до невизначеності даних у задачах математичної оптимізації. Як і в стохастичному програмуванні, модель Гарсані передбачає наявність повної апріорної розподільної інформації для всіх невідомих параметрів. Крім того, його аналіз припускає, що всі гравці використовують однаковий апріорний розподіл, і цей факт є загальновідомим. Багато ким (Morris, 1995; Wilson, 1987) ставилися під сумнів аспекти спільного апріорного розподілу та загальновідомості цих припущень. Тим не менш, байєсівська модель Гарсані залишається загальноприйнятою конвенцією для аналізу статичних ігор з неповною інформацією.

Застосовуючи інший підхід, інші автори в теорії ігор запропонували концепції рівноваги для ігор з неповною інформацією, що не залежать від розподілу. Ці аналізи розглядають можливість того, що розподільна інформація недоступна гравцям, або що вони вирішують не використовувати потенційно неточну розподільну інформацію. Поняття рівноваги *ex post* є найпоширенішою концепцією розв'язку, що не залежить від розподілу, і особливо поширене в літературі з теорії аукціонів. Холмстрьом та Майєрсон [2] вперше ввели це поняття під назвою "рівномірна стимулююча сумісність а Кремер та Маклін [3] вперше використали його в контексті аукціонів. Рівновага *ex post* є уточненням байєсівської рівноваги і є привабливою концепцією розв'язку, оскільки вона актуальна навіть тоді, коли гравцям бракує розподільної інформації про невизначені параметри. Однак, це сильна концепція, і рівноваги *ex post* не обов'язково існують в грі з неповною інформацією.

2.1 Інтервальна невизначеність в іграх

В іграх, де невизначеність розглядається як зовнішній фактор, а не як приватна інформація інших гравців, модель може бути спрощена. Часто передбачається, що виграш гравця повністю визначається ситуацією в грі, тобто залежить тільки від дій гравців, виходячи з того, що всілякі пере-

шкоди, збурення та іншого виду невизначеності не справляють на виграші гравців істотного впливу. Однак, можливі випадки, коли невизначеність змінює результат гри, і її ігнорування може призвести до результатів, що не задовольняють гравців. Економічна система, наприклад, піддається важкопрогнозованим збуренням: зміні номенклатури поставок, коливанню попиту, змінам, пов'язаним з появою та впровадженням нових технологій, поломкою та заміною обладнання, тощо. Невизначеність зазвичай не вдається описати у вигляді функції розподілу ймовірностей (як у байєсівському підході), але її можна представити як змінну, що приймає значення з деякої множини. Найчастіше про цю величину відомі лише межі її зміни. Щоб врахувати вплив такої невизначеності на результат гри, необхідно внести зміну в визначення гри.

Безкоаліційною грою в умовах невизначеності (або з природою як пасивним гравцем) називається система

$$\Gamma =< I, \{X_i\}_{i\in I}, Y, \{f_i(x,y)\}_{i\in I} >,$$

де $I = \{1,2,\ldots,n\}$ — множина гравців, X_i — множини стратегій гравців $i \in I$, елементи $x = = (x_1,x_2,\ldots,x_n) \in X = X_1 \times X_2 \times \ldots \times X_n$ — ситуації, $y \in Y \subseteq \mathbb{R}^m$ — фактор невизначеності (стан природи), $f_i(x,y) : X \times Y \to \mathbb{R}$ — функції виграшу гравців $i \in I$. Гра Γ реалізується наступним чином: одночасно та незалежно всі гравці $i \in I$ вибирають свої індивідуальні стратегії $x_i \in X_i$, утворюється ситуація $x = (x_1,x_2,\ldots,x_n)$. Незалежно від дій гравців реалізується деяка невизначеність $y \in Y$. Значення функції виграшу $f_i(x,y)$ на сформованій парі (x,y) визначає результат гри для гравця i — його виграш. При виборі своїх стратегій гравці повинні розраховувати на появу в грі будь-якої, заздалегідь непередбачуваної невизначеності $y \in Y$. Інакше кажучи, при заданій множині невизначеностей Y, виграш i-го гравця для фіксованої ситуації x лежить у множині значень $f_i(x,Y) = \{f_i(x,y) \mid y \in Y\}$.

У зв'язку зі змінами в понятті гри виникає питання про те, як гравцям слід враховувати невизначеність при виборі стратегій для досягнення оптимального результату. Особливістю даної ситуації є те, що гравцям не протистоїть активний супротивник (як інший гравець), що оптимізує свій вибір y, і вони не можуть достеменно знати або передбачити, яким чином

невизначеність y реалізується. Існує кілька підходів (критеріїв прийняття рішень) до вибору стратегій у таких ситуаціях, які й будуть розглянуті далі на прикладі аналога біматричної гри в умовах невизначеності.

Розглянемо гру в наступній формі. Нехай $I = \{1,2\}$. Задані матриці

$$\mathbf{A} = \begin{pmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{m1} & a_{m2} & \dots & a_{mn} \end{pmatrix}, \qquad \mathbf{B} = \begin{pmatrix} b_{11} & b_{12} & \dots & b_{1n} \\ b_{21} & b_{22} & \dots & b_{2n} \\ \dots & \dots & \dots & \dots \\ b_{m1} & b_{m2} & \dots & b_{mn} \end{pmatrix},$$

$$\mathbf{A_1} = \begin{pmatrix} \alpha_{11} & \alpha_{12} & \dots & \alpha_{1n} \\ \alpha_{21} & \alpha_{22} & \dots & \alpha_{2n} \\ \dots & \dots & \dots & \dots \\ \alpha_{m1} & \alpha_{m2} & \dots & \alpha_{mn} \end{pmatrix}, \quad \mathbf{B_1} = \begin{pmatrix} \beta_{11} & \beta_{12} & \dots & \beta_{1n} \\ \beta_{21} & \beta_{22} & \dots & \beta_{2n} \\ \dots & \dots & \dots & \dots \\ \beta_{m1} & \beta_{m2} & \dots & \beta_{mn} \end{pmatrix},$$

де $\alpha_{ij} \geq 0, \beta_{ij} \geq 0 \; \forall i = \overline{1,m}, \forall j = \overline{1,n}$. Множина стратегій першого гравця $X_1 = \{1,2,\dots,m\}$, другого — $X_2 = \{1,2,\dots,n\}$. Нехай перший гравець обрав стратегію i, другий — j. Реалізація невизначеності $y \in Y$ призводить до того, що виграш першого гравця $f_{ij}^1(y)$ лежить у відрізку $[a_{ij} - \alpha_{ij}, a_{ij} + \alpha_{ij}]$, а виграш другого $f_{ij}^2(y)$ — у відрізку $[b_{ij} - \beta_{ij}, b_{ij} + \beta_{ij}]$. Така гра може інтерпретуватися наступним чином. Нехай спочатку побудована спрощена модель конфлікту, яка описується біматричною грою $< \mathbf{A}, \mathbf{B} >$ (номінальні виграші). Для уточнення цієї моделі до уваги беруться деякі фактори невизначеності, які спричиняють коливання виграшів гравців у заданих межах, тобто задаються додатково матриці амплітуд коливань $\mathbf{A_1}$, $\mathbf{B_1}$.

Прикладом гри в такій формі може слугувати модель конкурентної боротьби виробників однакових товарів. Якщо за виграші гравців прийняти кількість виробленого ними товару на день (яка пропорційна прибутку за ці товари), то матриці $\mathbf{A}$ та $\mathbf{B}$ міститимуть плановану кількість виробленого товару, а матриці $\mathbf{A_1}$ та $\mathbf{B_1}$ — межі можливих відхилень від цієї кількості (відхилення можуть виникнути, наприклад, через вихід обладнання з ладу

або нестачу витратних матеріалів). На прикладі першого гравця розглянемо, які існують способи вибору стратегій в умовах невизначеності. За допомогою того чи іншого критерію будемо обчислювати "очікуваний"виграш цього гравця (тобто такий, на отримання якого розраховує гравець, прийнявши відповідне рішення). Позначимо цей виграш, що вже не залежить від невизначеності, через f_{ij}^1 (для гравця 2 — f_{ij}^2). Таким чином зводимо гру до звичайної біматричної гри з матрицями $\mathbf{F_1} = ||f_{ij}^1||_{i,j}$ та $\mathbf{F_2} = ||f_{ij}^2||_{i,j}$, в якій можна шукати ситуації рівноваги стандартним способом. Далі розглянемо декілька таких критеріїв.

2.2 Детермінізація невизначеності критеріями прийняття рішень

В умовах, коли виграші гравців залежать не лише від їхніх стратегій, а й від непередбачуваного фактора невизначеності $y \in Y$, виникає потреба у формальних підходах до вибору оптимальних дій. Цей розділ розглядає декілька критеріїв прийняття рішень, які дозволяють "детермінізувати"цю невизначеність. Кожен критерій пропонує спосіб обчислення "очікуваного"виграшу f_{ij}^k для кожного гравця k у кожній ситуації (i,j), тим самим зводячи вихідну гру в умовах невизначеності до стандартної біматричної гри з матрицями виграшів $\mathbf{F_1} = ||f_{ij}^1||_{i,j}$ та $\mathbf{F_2} = ||f_{ij}^2||_{i,j}$, для якої можна шукати рівноважні ситуації стандартними методами.

2.2.1 MM-критерій (критерій песимізму, Вальда)

Даний критерій ґрунтується на мінімаксному підході. Гравець розраховує, що невизначеність реалізується найгіршим для нього чином, тобто очікуваний виграш у ситуації (i,j) становитиме

$$f_{ij}^1 = \min_{y \in Y} f_{ij}^1(y) = a_{ij} - \alpha_{ij}.$$

Тоді, для кожного фіксованого $j = \overline{1,n}$ стратегія гравця 1 обирається з умови

$$\max_i f_{ij}^1 = \max_i (a_{ij} - \alpha_{ij}).$$

Такий критерій повністю виключає ризик для гравця, який він у будь-якому випадку отримає виграш не менший за очікуваний. Однак, можливі випадки, коли мінімаксний принцип недоцільно застосовувати. Нехай, наприклад, матриці $\mathbf{A}$ та $\mathbf{A_1}$ мають наступний вигляд:

$$\mathbf{A} = \begin{pmatrix} 3 \\ 100 \\ 100 \\ 100 \end{pmatrix}, \quad \mathbf{A_1} = \begin{pmatrix} 1 \\ 99 \\ 99 \\ 99 \end{pmatrix}.$$

Матриця очікуваних виграшів першого гравця

$$\mathbf{F_1} = \begin{pmatrix} 2 \\ 1 \\ 1 \\ 1 \end{pmatrix},$$

тобто, використовуючи ММ-критерій, перший гравець повинен обрати свою першу стратегію. Однак, якщо невизначеність реалізується не "найгіршим"для гравця чином або якщо гра проводиться велику кількість разів, обравши будь-яку іншу свою стратегію, перший гравець отримав би більший виграш (у випадку вибору стратегії 1, його максимально можливий виграш становить 3, тоді як при виборі стратегій 2, 3, 4 максимально можливий виграш дорівнює 199). ММ-критерій варто застосовувати в тих випадках, коли:

1) про можливість появи невизначеності нічого не відомо;
2) потрібно виключити навіть мінімальний ризик;
3) гра реалізується єдиний раз.

2.2.2 Критерій оптимізму (критерій "азартного гравця")

В основі цього критерію лежить принцип максимаксу. Гравець розраховує на реалізацію найкращої для нього невизначеності, тобто для кожного j свою стратегію i він обирає з умови

$$\max_i \max_{y \in Y} f_{ij}^1(y) = \max_i (a_{ij} + \alpha_{ij}).$$

Цей критерій передбачає дуже великий ризик, і його використання є у більшості випадків недоцільним.

2.2.3 BL-критерій (Баєса-Лапласа)

Цей критерій використовується, якщо відомий закон розподілу невизначеності ($g_\eta(y)$) і гра реалізується велику кількість разів. Як значення невизначеності обирається її математичне сподівання, тобто очікуваний виграш гравця становить

$$f_{ij}^1 = a_{ij} + \int_{-\alpha_{ij}}^{\alpha_{ij}} y g_\eta(y) dy,$$

і для будь-якої стратегії другого гравця j перший гравець обирає стратегію i з умови

$$\max_i f_{ij}^1.$$

Якщо кількість реалізацій гри велика, то середнє значення виграшу наближається до значення очікуваного виграшу. Критерій Баєса-Лапласа є більш оптимістичним, ніж мінімаксний критерій, однак його використання передбачає більшу поінформованість, повторення гри велику кількість разів (або допустимість певного ризику, якщо гра повторюється невелику кількість разів). Зазначимо, що якщо щільність розподілу невизначеності є парною функцією, то математичне сподівання (інтеграл від непарної функції на симетричному проміжку) дорівнюватиме нулю. Це означає, що застосування BL-критерію в даному випадку призводить до біматричної гри з вихідними матрицями $\mathbf{A}$ та $\mathbf{B}$.

2.2.4 Критерій мінімаксного жалю (Севіджа)

Вводиться додаткова матриця

$$\mathbf{R}(\mathbf{y}) = ||r_{ij}(y)||_{i,j}, \qquad r_{ij}(y) = \max_i f^1_{ij}(y) - f^1_{ij}(y) \quad \forall i = \overline{1,m}, \quad \forall j = \overline{1,n}.$$

Елементи цієї матриці можна інтерпретувати як втрати (штрафи), що виникають при виборі гравцем i-ї стратегії замість тієї, яка дає найбільший виграш (за деякої невизначеності). Максимізуючи ці втрати за невизначеністю, отримаємо:

$$r_{ij} = \max_{y\in Y} r_{ij}(y) = \max_{y\in Y}(\max_i f^1_{ij}(y) - f^1_{ij}(y)) = \max_{y\in Y}\max_i f^1_{ij}(y) - \min_{y\in Y} f^1_{ij}(y) =$$
$$= \max_{y\in Y}\max_i f^1_{ij}(y) - (a_{ij} - \alpha_{ij}).$$

При використанні даного критерію завдання першого гравця полягає в тому, щоб мінімізувати свої максимально можливі втрати r_{ij}. Це означає, що при будь-якому j він повинен обирати свою стратегію i з умови

$$\min_i r_{ij}.$$

Цей критерій ґрунтується на тому ж підході, що й ММ-критерій: гравець розраховує на реалізацію "найгіршої"невизначеності, і своїми зусиллями домагається зменшення її впливу на свій виграш. Таким чином, цей критерій доцільно використовувати в тих же випадках, що й ММ-критерій, тобто коли нічого не відомо про розподіл невизначеності, необхідно виключити будь-який ризик або гра реалізується єдиний раз. У тому випадку, коли невизначеність розподілена неперервно, критерій мінімаксного жалю взагалі збігатиметься з мінімаксним критерієм. Дійсно,

$$\max_{y\in Y}\max_i f^1_{ij}(y) = c_j,$$

тобто це деяке число для кожної фіксованої стратегії другого гравця j. Тоді

$$\min_i r_{ij} = \min_i(c_j - (a_{ij} - \alpha_{ij})) = c_j - \max_i(a_{ij} - \alpha_{ij}).$$

Це означає, що перший гравець повинен обирати свою стратегію з умови

$$\max_i(a_{ij} - \alpha_{ij}),$$

що збігається з умовою для ММ-критерію.

2.2.5 Критерій песимізму-оптимізму (Гурвіца)

Цей критерій є похідним від критеріїв песимізму та оптимізму. Гравець обирає коефіцієнт довіри критерію песимізму $c \in [0,1]$, тоді елементи матриці очікуваних виграшів вважаються рівними

$$f_{ij}^1 = c\min_{y\in Y} f_{ij}^1(y)+(1-c)\max_{y\in Y} f_{ij}^1(y) = c(a_{ij}-\alpha_{ij})+(1-c)(a_{ij}+\alpha_{ij}) = a_{ij}+(1-2c)\alpha_{ij}.$$

Очевидно, при $c = 1$ критерій Гурвіца перетворюється на ММ-критерій, а при $c = 0$ – на критерій "азартного гравця". Загалом питання вибору вагового коефіцієнта c є досить складним. Зазвичай беруть його середнє значення, що дорівнює $\frac{1}{2}$. Але тоді матриця очікуваних виграшів збігатиметься з вихідною матрицею $\mathbf{A}$, оскільки

$$f_{ij}^1 = a_{ij} + (1 - 2 \cdot \frac{1}{2})\alpha_{ij} = a_{ij} + 0 \cdot \alpha_{ij} = a_{ij}.$$

Потім для будь-якого j стратегія першого гравця i обирається з умови

$$\max_i f_{ij}^1.$$

Принцип Гурвіца представляє собою "урівноважену"позицію, оскільки враховує можливість реалізації як "хороших так і "поганих"для гравця невизначеностей. Даний критерій варто застосовувати, якщо про появу невизначеностей нічого невідомо, допустимий певний ризик і гра повторюється невелику кількість разів.

2.2.6 Критерій Ходжа-Лемана

Цей критерій спирається одночасно на ММ-критерій та критерій Баєса-Лапласа. Обирається коефіцієнт довіри використовуваному закону розподілу невизначеності $c \in [0,1]$. Залежно від його величини, домінує критерій Баєса-Лапласа або ММ-критерій. Елементи матриці очікуваних виграшів матимуть вигляд

$$\begin{aligned} f_{ij}^1 &= c(a_{ij} + \int_{-\alpha_{ij}}^{\alpha_{ij}} y g_\eta(y) dy) + (1-c) \min_y f_{ij}^1(y) = \\ &= c(a_{ij} + \int_{-\alpha_{ij}}^{\alpha_{ij}} y g_\eta(y) dy) + (1-c)(a_{ij} - \alpha_{ij}) = \\ &= a_{ij} + c \int_{-\alpha_{ij}}^{\alpha_{ij}} y g_\eta(y) dy - (1-c)\alpha_{ij}. \end{aligned}$$

Потім аналогічно до попередніх випадків гравець обирає свою стратегію так, щоб за будь-якої фіксованої стратегії іншого гравця максимізувати очікуваний виграш. Вибір коефіцієнта довіри, як і в критерії Гурвіца, залишається на розсуд гравців. Критерій Ходжа-Лемана застосовується в тих випадках, коли

1) закон розподілу невизначеності достеменно невідомий, але деякі припущення щодо нього можливі;
2) гра реалізується велику кількість разів;
3) допустимий певний ризик, якщо гра реалізується невелику кількість разів.

2.3 Робастний підхід

У класичній теорії ігор для задач з повною інформацією припускається, що всі параметри гри, включаючи виграшні функції гравців, є відомими. В умовах неповної інформації, модель Харсані [1] замінює невідомі параметри випадковими величинами з відомими розподілами й запроваджує поняття типів гравців. Цей підхід дозволяє зберігати математичну строгість, але вимагає виконання сильного припущення про існування спільного апріорного розподілу, який є загальновідомим для всіх гравців.

Однак у багатьох реальних сценаріях (наприклад, в економіці, телекомунікаційних мережах або системах безпеки) учасники гри не мають точних і достовірних статистичних даних про невизначені параметри. У таких випадках використання імовірнісної моделі є не тільки непродуктивним, але й потенційно хибним. У відповідь на ці виклики було розроблено *робастний підхід*, який уникає потреби у статистичних припущеннях.

Робастна оптимізація, як узагальнення принципу мінімаксу [4], передбачає, що кожен гравець прагне максимізувати свій *гірший можливий очікуваний виграш* з-поміж усіх допустимих реалізацій невідомих параметрів. Таким чином, робастний підхід природно враховує епістемічну невизначеність (невизначеність через брак знання), що відрізняє його від байєсівського (стохастичного) підходу.

У статті [5] авторами вперше формалізовано модель *робастної гри* як гру з неповною інформацією, в якій замість розподілів задані множини невизначеності параметрів виграшу. Кожен гравець обирає стратегію, максимізуючи свій виграш у найгіршому сценарії в межах цієї множини. Такий підхід забезпечує гарантований результат незалежно від імовірнісної природи невизначеності та дозволяє уникнути надмірно консервативних стратегій, характерних для класичного мінімаксного аналізу.

На відміну від *ex post* рівноваг, які можуть не існувати при відсутності розподільчої інформації [?], у робастних іграх автори доводять існування *робастно-оптимізаційної рівноваги* у випадку скінченного простору дій і обмеженої множини невизначеності. Це дає вагому мотивацію до використання робастного підходу як альтернативи традиційним моделям у складних сценаріях з невизначеними або нечітко визначеними даними.

2.3.1 Робастна лінійна оптимізація

Робастна (стійка) оптимізація — це підхід до задач математичного програмування, який дозволяє враховувати невизначеність параметрів моделі без використання ймовірнісного опису цієї невизначеності. Нехай

розглядається задача оптимізації:

$$\begin{aligned} \max_{x} \quad & f(x) \\ \text{s.t.} \quad & x \in X(\delta_1, \ldots, \delta_\omega), \end{aligned}$$

де x — вектор керованих змінних, $X(\cdot)$ — область допустимих рішень, яка залежить від параметрів δ_i. У номінальній задачі ці параметри фіксовані, але в реальності їх значення можуть бути невідомими та належати деякій множині невизначеності U.

Робастний аналог задачі формулюється так:

$$\begin{aligned} \max_{x} \quad & f(x) \\ \text{s.t.} \quad & x \in X(\delta_1, \ldots, \delta_\omega),\ \forall(\delta_1, \ldots, \delta_\omega) \in U. \end{aligned}$$

Тобто розв'язок має залишатися допустимим для будь-якого допустимого сценарію параметрів з множини U.

Загальна постановка робастної лінійної задачі виглядає так:

$$\begin{aligned} \max_{x} \quad & c^\top x \\ \text{s.t.} \quad & \tilde{A}x \leq b, \quad \forall \tilde{A} \in U, \\ & x \in S, \end{aligned}$$

де U — поліедральна множина невизначеності для матриці коефіцієнтів $\tilde{A}$, S — допустима множина для x. У [6] було доведено, що така задача еквівалентна номінальній задачі з більшим розміром (розширеним простором змінних), але її все одно можна розв'язувати як задачу лінійного програмування.

2.3.2 Формалізація робастної моделі гри

Розглядається гра з N гравцями, кожен з яких має скінченну множину дій $A_i = \{1, 2, \ldots, a_i\}$, $i = 1, \ldots, N$. Гравець i обирає *змішану стратегію*

$x_i \in S_{a_i}$, де

$$S_{a_i} := \left\{ x_i \in \mathbb{R}^{a_i} \;\middle|\; x_i \geq 0,\ \sum_{j=1}^{a_i} x_{ij} = 1 \right\},$$

а повний профіль стратегій позначається через $x = (x_1, \ldots, x_N) \in S := \prod_{i=1}^{N} S_{a_i}$.

Нехай P — багатовимірний тензор виграшів $P^i \in \mathbb{R}^{a_1 \times \cdots \times a_N}$ для кожного гравця i, який є невідомим, але належить до компактної множини невизначеності $\mathcal{U} \subset \mathbb{R}^{N \cdot a_1 \cdots a_N}$, тобто

$$P = (P^1, \ldots, P^N) \in \mathcal{U}.$$

Очікуваний виграш гравця i при профілі стратегій $x = (x_1, \ldots, x_N)$ і фіксованому P визначається як:

$$\pi_i(P; x_1, \ldots, x_N) = \sum_{j_1=1}^{a_1} \cdots \sum_{j_N=1}^{a_N} P^i_{j_1,\ldots,j_N} \cdot \prod_{k=1}^{N} x_{kj_k}.$$

У моделі з неповною інформацією припускаємо, що гравці знають лише множину $\mathcal{U}$ можливих реалізацій параметра P, без жодної ймовірнісної інформації. Кожен гравець дотримується принципу робастної оптимізації: він вибирає таку стратегію, яка максимізує його *найгірший можливий очікуваний виграш*, тобто:

$$\max_{x_i \in S_{a_i}} \inf_{P \in \mathcal{U}} \pi_i(P; x_1, \ldots, x_N).$$

Нехай x_{-i} позначає вектор стратегій усіх гравців, крім i. Тоді найкраща відповідь гравця i у робастному сенсі задається як:

$$\mathcal{B}_i(x_{-i}) := \arg\max_{x_i \in S_{a_i}} \inf_{P \in \mathcal{U}} \pi_i(P; x_{-i}, x_i).$$

Означення 2.1. Профіль стратегій $x^* = (x_1^*, \ldots, x_N^*) \in S$ називається **робастно-оптимізаційною рівновагою**, якщо для кожного $i \in \{1, \ldots, N\}$ має місце:

$$x_i^* \in \mathcal{B}_i(x_{-i}^*).$$

Іншими словами, жоден гравець не може поліпшити свій найгірший очікуваний виграш, односторонньо змінюючи свою стратегію.

На відміну від баєсівської моделі Харсані, де гравці максимізують математичне сподівання $\mathbb{E}_P[\pi_i(P; x)]$ за відомим апріорним розподілом $P \sim \mu$, в представленій моделі не вимагається жодної ймовірнісної інформації. Вся інформація про P обмежується множиною можливих реалізацій $\mathcal{U}$, яка є спільновідомою.

Таким чином, робастна модель гри природно узагальнює класичну модель Неша на випадок невизначеності у виграшах, без потреби в ймовірностях, і водночас не вимагає надзвичайно сильних умов, як у випадку ex post рівноваг. Робастно-оптимізаційна рівновага завжди існує за певних умов (про це детальніше у наступному розділі), що робить її придатним концептом для застосування в практичних задачах стратегічної взаємодії в умовах невизначеності.

На відміну від класичних мінімакс стратегій, які мають конфлікт з концепцією рівноваги (оскільки припускають ворожість інших гравців), в описаній моделі робастність стосується лише невизначеності параметрів, але не стратегічної поведінки опонентів. Гравець припускає найгірший сценарій виграшів при фіксованій поведінці інших гравців, яка є відомою.

Зауваження 2.1. "Природа"(невідомі параметри) не є активним гравцем, а тому не має конфлікту з принципом рівноваги.

Таким чином, поєднання концепцій рівноваги та робастної оптимізації є логічно узгодженим і дозволяє моделювати поведінку гравців у складних ситуаціях без статистичних припущень.

2.3.3 Існування рівноваги в кінцевих робастних іграх

Однією з головних переваг робастної моделі гри є те, що вона дозволяє гарантувати існування рівноваги за досить слабких припущень. Незважаючи на те, що ex post рівноваги можуть не існувати у загальному випадку (оскільки вони вимагають оптимальності при *усіх* реалізаціях параметрів),

робастно-оптимізаційні рівноваги завжди існують для кінцевих ігор з обмеженою множиною невизначеності.

Розглядається гра з N гравцями, кожен з яких має скінченну множину дій $A_i = \{1, \dots, a_i\}$. Гравці обирають змішані стратегії $x_i \in S_{a_i}$, як було визначено раніше. Виграші задаються тензорами P^i, що належать до спільної множини невизначеності $\mathcal{U} \subset \mathbb{R}^{N \cdot a_1 \cdots a_N}$.

У моделі робастної гри кожен гравець прагне максимізувати свій найгірший можливий очікуваний виграш:

$$x_i^* \in \arg\max_{x_i \in S_{a_i}} \inf_{P \in \mathcal{U}} \pi_i(P; x_{-i}^*, x_i).$$

Теорема 2.1 (Існування рівноваги у робастній грі)**.** *[5]*
Нехай:

1) $N < \infty$ — *скінченне число гравців;*
2) $a_i < \infty$ — *скінченне число дій у кожного гравця* i*;*
3) $\mathcal{U} \subset \mathbb{R}^{N \cdot a_1 \cdots a_N}$ — *непорожня, компактна (замкнена та обмежена) множина допустимих тензорів виграшів.*

Тоді існує ***робастно-оптимізаційна рівновага*** $x^* = (x_1^*, \dots, x_N^*) \in \mathcal{S}$.

Інтуїтивно, ця рівновага описує ситуацію, коли кожен гравець обрав таку стратегію, яка максимізує його виграш у найгіршому сценарії невизначеності, з урахуванням стратегій інших гравців.

Важливо підкреслити, що ця теорема забезпечує *універсальне* існування рівноваги у широкому класі ігор з невизначеними виграшами, незалежно від того, наскільки складною є множина $\mathcal{U}$. Це робить робастну модель особливо привабливою для застосування в практичних задачах, де ймовірнісна інформація недоступна або ненадійна.

2.3.4 Обчислення рівноваги у кінцевих робастних іграх

Нехай N – кількість гравців, a_i – кількість чистих стратегій i-го гравця. Кожен гравець i обирає змішану стратегію $x^i \in \Delta^{a_i}$, де Δ^{a_i} – стандартний симплекс:

$$\Delta^{a_i} = \left\{ x^i \in \mathbb{R}^{a_i} \,\middle|\, \sum_{j=1}^{a_i} x_{ij} = 1, x_{ij} \geq 0 \right\}.$$

В класичному випадку, коли виграшні функції відомі точно, очікуваний виграш i-го гравця обчислюється як

$$\pi_i(x) = \sum_{j_1,\ldots,j_N} P_i(j_1,\ldots,j_N) \prod_{k=1}^{N} x_{kj_k},$$

де P_i – задана матриця виграшів гравця i. Профіль змішаних стратегій $(x^1,\ldots,x^N)$ є рівновагою Неша тоді й лише тоді, коли кожен гравець не може покращити свій виграш, однобічно відхилившись до будь-якої чистої стратегії.

У випадку робастної гри матриця виграшів P_i невідома точно; припускається, що вона належить множині допустимих значень U. Типовий приклад – поліедральна множина, наприклад:

$$U = \{P \,|\, G \cdot vec(P) \leq d\}.$$

Визначаємо "виграш у найгіршому випадку" як

$$\rho_i(x) = \inf_{P \in U} \pi_i(P; x),$$

де $\pi_i(P; x)$ означає очікуваний виграш i-го гравця при профілі стратегій x та фіксованій матриці виграшу P.

Гравець i прагне максимізувати $\rho_i(x)$, припускаючи, що природа обере такий $P \in U$, який мінімізує його очікуваний виграш.

Для кожного i визначимо множину вершин поліедра U як $G(1),\ldots,G(k)$. Через лінійність функції $\pi_i(P; x)$ за P, мінімум по U досягається на одній із вершин. Тоді:

$$\inf_{P \in U} \pi_i(P; x) = \min_{\ell=1,\ldots,k} \pi_i(G(\ell); x).$$

Вводимо допоміжну змінну z_i, яка представляє найгірший очікуваний виграш гравця i за профілю x. Для зручності додаємо також змінну ϕ_i з умовою $z_i = \phi_i$. Щоб зберегти лінійність, переписуємо мінімум у вигляді:

$$z_i \leq \pi_i(G(\ell); x), \quad \forall \ell = 1, \ldots, k.$$

Крім того, для кожного гравця вводимо вектор $\theta^i \in \Delta^k$ – барицентричні координати, що дозволяють описати змішану стратегію "природи" у просторі вершин $G(\ell)$:

$$\theta^i \in \left\{ \theta \in \mathbb{R}^k \,\middle|\, \sum_{\ell=1}^{k} \theta_\ell = 1,\, \theta_\ell \geq 0 \right\}.$$

Щоб гарантувати відсутність стимулу до відхилення в будь-яку чисту стратегію e_{j_i}, вимагається:

$$\sum_{\ell=1}^{k} \theta_\ell^i \, \pi_i(G(\ell); x^{-i}, e_{j_i}) \leq \phi_i, \quad \forall j_i = 1, \ldots, a_i.$$

Отже, для кожного $i = 1, \ldots, N$ сформульовано таку систему:

$$\begin{aligned}
z_i &= \phi_i \\
z_i - \pi_i \left(G(\ell; \mathbf{x}^1, \ldots, \mathbf{x}^N) \right) &\leq 0, \quad \ell = 1, \ldots, k \\
\mathbf{e}^\top \mathbf{x}^i &= 1 \\
\mathbf{x}^i &\geq \mathbf{0} \\
\mathbf{e}^\top \boldsymbol{\theta}^i &= 1 \\
\sum_{\ell=1}^{k} \theta_\ell^i \pi_i \left(G(\ell; \mathbf{x}^{-i}, \mathbf{e}_{j_i}^i) \right) - \phi_i &\leq 0, \quad j_i = 1, \ldots, a_i \\
\boldsymbol{\theta}^i &\geq \mathbf{0}
\end{aligned} \tag{2.1}$$

Усі разом ці умови формують систему, рішення якої дають профілі стратегій $x = (x^1, \ldots, x^N)$, що є робастно-оптимізаційною рівновагою. Додаткові змінні z_i, ϕ_i та θ^i не входять у фінальний профіль, але є критичними для його перевірки та побудови.

Змінна θ^i відіграє роль координатного представлення "найгіршого сценарію" для i-го гравця в термінах вершин поліедра U. Це дозволяє уникнути явної мінімізації та забезпечити лінійність усієї системи. Вектор θ^i можна також розглядати як двоїстий розв'язок відповідної задачі природи – тобто лінійної програми $\min_{P \in U} \pi_i(P; x)$.

2.4 Програмна реалізація

Було програмно реалізовано обчислювальний метод для знаходження робастно-оптимізаційних рівноваг у скінченних іграх мовою Python. Відповідний код наведено в додатку А (див. стор. 49).

2.4.1 Система мультилінійних рівностей та нерівностей

Система включає наступні змінні для гри двох гравців з a_R діями для Гравця R та a_C діями для Гравця C, а також k екстремальних точок множини невизначеності:

- $\boldsymbol{x}^R \in \mathbb{R}^{a_R}$: Змішана стратегія (розподіл ймовірностей для дій) для Гравця R.
- $\boldsymbol{x}^C \in \mathbb{R}^{a_C}$: Змішана стратегія для Гравця C.
- $z^R \in \mathbb{R}$: Скалярне значення, що представляє очікувану виплату у найгіршому випадку, яку прагне максимізувати Гравець R.
- $z^C \in \mathbb{R}$: Скалярне значення, що представляє очікувану виплату у найгіршому випадку, яку прагне максимізувати Гравець C.
- $\boldsymbol{\theta}^R \in \mathbb{R}^k$: Розподіл ймовірностей, що представляє переконання Гравця R щодо того, яку екстремальну точку Природа (супротивник) обере, щоб мінімізувати виплату Гравця R, за заданої стратегії супротивника $\boldsymbol{x}^C$.
- $\boldsymbol{\theta}^C \in \mathbb{R}^k$: Аналогічно, розподіл ймовірностей для переконань Гравця C щодо вибору Природи проти $\boldsymbol{x}^R$.
- $\phi^R \in \mathbb{R}$: Допоміжна скалярна змінна для Гравця R, що представляє мінімальну очікувану виплату, яку Гравець R може досягти (вибравши оптимальну чисту стратегію або її суміш) проти $\boldsymbol{x}^C$, коли Природа

обирає найгірший розподіл $\boldsymbol{\theta}^R$ над екстремальними точками.

- $\phi^C \in \mathbb{R}$: Допоміжна скалярна змінна для Гравця C з аналогічною інтерпретацією.

Загальна кількість змінних у задачі оптимізації становить $a_R + a_C + 2k + 4$.

Кортеж $(\boldsymbol{x}^R, \boldsymbol{x}^C, z^R, z^C, \boldsymbol{\theta}^R, \boldsymbol{\theta}^C, \phi^R, \phi^C)$ становить рівновагу надійної оптимізації, якщо він задовольняє наступний набір умов, адаптованих із системи (2.1):

Для кожного гравця $i \in \{R, C\}$:

(i) **Обмеження дійсності стратегій**: Змішані стратегії повинні бути дійсними розподілами ймовірностей:

$$\sum_{j=1}^{a_i} x_j^i = 1 \tag{2.2}$$

$$x_j^i \geq 0 \quad \forall j = 1, \ldots, a_i \tag{2.3}$$

(ii) **Визначення виплати у найгіршому випадку**: Змінна z^i повинна бути меншою або рівною очікуваній виплаті, яку гравець i отримує за будь-якої реалізації невизначеності, що відповідає екстремальній точці $G(s)$, враховуючи змішані стратегії $(\boldsymbol{x}^R, \boldsymbol{x}^C)$:

$$z^i - (\boldsymbol{x}^R)^T \boldsymbol{P}^i(G(s))\boldsymbol{x}^C \leq 0 \quad \forall s = 1, \ldots, k \tag{2.4}$$

Гравець i прагне максимізувати цю z^i. Оскільки функція очікуваної виплати $\pi^i(\cdot; \boldsymbol{x}^R, \boldsymbol{x}^C)$ є лінійною щодо невизначених параметрів (для фіксованих стратегій), її мінімум над гіперпрямокутником U_f досягається в одній з екстремальних точок $G(s)$. Таким чином, задоволення цієї умови для всіх s гарантує $z^i \leq \min_{G(s)} \pi^i(G(s); \boldsymbol{x}^R, \boldsymbol{x}^C)$.

(iii) **Допоміжна змінна ϕ^i та обмеження для ваг Природи $\boldsymbol{\theta}^i$**: Ці умови стосуються оцінки гравцем i своїх чистих стратегій проти змішаної стратегії супротивника $\boldsymbol{x}^{-i}$ та вибору Природи як супротивника для розподілу сценаріїв $\boldsymbol{\theta}^i$. Ваги $\boldsymbol{\theta}^i$ повинні формувати дійсний розподіл

ймовірностей:

$$\sum_{l=1}^{k} \theta_l^i = 1 \tag{2.5}$$

$$\theta_l^i \geq 0 \quad \forall l = 1, \ldots, k \tag{2.6}$$

Для кожної чистої стратегії $j_i \in \{1, \ldots, a_i\}$ гравця i (представленої одиничним вектором $\boldsymbol{e}_{j_i}$), очікувана виплата, усереднена за екстремальними точками з використанням ваг θ_s^i, повинна бути не більшою за ϕ^i:

$$\sum_{s=1}^{k} \theta_s^i \left((\boldsymbol{e}_{j_i})^T \boldsymbol{P}^i (G(s)) \boldsymbol{x}^C \right) - \phi^R \leq 0 \quad \text{(для гравця } i = R\text{)} \tag{2.7}$$

$$\sum_{s=1}^{k} \theta_s^i \left((\boldsymbol{x}^R)^T \boldsymbol{P}^i (G(s)) \boldsymbol{e}_{j_i} \right) - \phi^C \leq 0 \quad \text{(для гравця } i = C\text{)} \tag{2.8}$$

Гравець i обирає $\boldsymbol{\theta}^i$ так, щоб мінімізувати ліву частину цих нерівностей (фактично, мінімізувати свою виплату від чистих стратегій), а ϕ^i є значенням цього мінімуму.

(iv) **Умова узгодженості рівноваги**: Основна умова рівноваги пов'язує z^i та ϕ^i:

$$z^i - \phi^i = 0 \tag{2.9}$$

Ця рівність гарантує, що виплата z^i (досягнута гравцем i за допомогою змішаної стратегії $\boldsymbol{x}^i$ проти $\boldsymbol{x}^{-i}$ та Природи у найгіршому випадку) є дійсно найкращою для гравця i. Тобто, жодна чиста стратегія j_i (чиє значення у найгіршому випадку відображено ϕ^i) не може принести кращу виплату, ніж z^i. Це означає, що $\boldsymbol{x}^i$ є найкращою відповіддю.

2.4.2 Знаходження розв'язку

Система з N_E рівностей $h_m(\boldsymbol{y}) = 0$ та N_I нерівностей $g_n(\boldsymbol{y}) \leq 0$ (де $\boldsymbol{y}$ є вектором усіх змінних) перетворюється на єдину скалярну цільову функцію $H(\boldsymbol{y})$. Це досягається шляхом підсумовування квадратів порушень кожного

обмеження:

$$H(\boldsymbol{y}) = \sum_{m=1}^{N_E} (h_m(\boldsymbol{y}))^2 + \sum_{n=1}^{N_I} (\max(0, g_n(\boldsymbol{y})))^2 \tag{2.10}$$

Рівновага $\boldsymbol{y}^*$ вважається знайденою, якщо $H(\boldsymbol{y}^*) = 0$ (або, на практиці, $H(\boldsymbol{y}^*) < \epsilon$ для деякої малої допустимої похибки ϵ). Зокрема

- Обмеження рівності (2.2), (2.5), (2.9) вносять члени, такі як $((\sum x_j^i) - 1)^2$ та $(z^i - \phi^i)^2$.
- Обмеження нерівності (2.3), (2.6) (переписані як $-x_j^i \leq 0, -\theta_l^i \leq 0$) вносять члени, такі як $(\max(0, -x_j^i))^2$.
- Обмеження нерівності (2.4), (2.7), (2.8) вносять члени, такі як $(\max(0, z^i - \pi^i(\dots)))^2$.

Штрафний множник M_p (наприклад, $M_p = 100$) застосовується до квадратів порушень обмежень нормалізації (2.2), (2.5) та невід'ємності (2.3), (2.6), щоб суворо забезпечити їхнє дотримання під час оптимізації.

Для використання ефективних градієнтних алгоритмів оптимізації потрібен градієнт $\nabla H(\boldsymbol{y})$ штрафної функції. Замість ручного виведення цих складних градієнтів, використано автоматичне диференціювання. Уся штрафна функція $H(\boldsymbol{y})$ будується з використанням операцій, визначених у бібліотеці PyTorch. PyTorch потім може автоматично обчислювати точні часткові похідні $H(\boldsymbol{y})$ щодо кожної компоненти $\boldsymbol{y}$, відстежуючи обчислювальний граф. Це забезпечує точні градієнти, що є важливим для збіжності та продуктивності алгоритму оптимізації.

Безумовна мінімізація $H(\boldsymbol{y})$ виконується за допомогою квазіньютонівського методу, а саме алгоритму L-BFGS-B, реалізованого у 'scipy.optimize.minimize'. Цей алгоритм ітеративно шукає локальний мінімум $H(\boldsymbol{y})$. Метод L-BFGS-B вимагає значення цільової функції $H(\boldsymbol{y})$ та її градієнта $\nabla H(\boldsymbol{y})$ на кожній ітерації. Він також може обробляти прості обмеження типу "коробка"для змінних, які використовуються для безпопосереднього забезпечення $0 \leq x_j^i \leq 1$ та $0 \leq \theta_l^i \leq 1$, доповнюючи штрафні члени для цих обмежень. Оптимізація припиняється, коли покращення в $H(\boldsymbol{y})$ або норма градієнта падає нижче заздалегідь визначеної допустимої похибки, або досягнута максимальна кількість ітерацій. Розв'язок $\boldsymbol{y}^*$ вважається успішним, якщо кінцеве значення штрафної функції $H(\boldsymbol{y}^*)$ дуже близьке до нуля (використано $< 10^{-9}$).

Описана задача оптимізації зазвичай є неопуклою [5], що означає: результати локальної оптимізації, зокрема за допомогою методів типу L-BFGS-B, можуть залежати від вибраної початкової точки $\boldsymbol{y}_0$ і призводити до різних локальних мінімумів. Оскільки в ігрових моделях може існувати множинність рівноваг, у цьому контексті застосовується стратегія мультистартів, яка допомагає охопити більшу частину простору розв'язків.

Ця стратегія передбачає багаторазовий запуск оптимізації з рандомізованими початковими припущеннями. Для кожного запуску початкова точка $\boldsymbol{y}_0$ генерується випадковим чином. Щоб гарантувати, що компоненти векторів ймовірностей (а саме, змішані стратегії $\boldsymbol{x}^i$ та ваги природи $\boldsymbol{\theta}^i$) рівномірно розподілені по ймовірносному симплексі – тобто є невід'ємними та такими, що сумуються до одиниці – використовується рівномірний розподіл Діріхле.

Кожен успішний запуск, при якому виконується критерій збіжності $H(\boldsymbol{y}^*) < \epsilon$, дає кандидат на рівновагу. Щоб побудувати множину унікальних рівноваг, усі знайдені розв'язки порівнюються між собою: новий розв'язок $(\boldsymbol{x}^R, \boldsymbol{x}^C)$ додається до множини лише в тому випадку, якщо він суттєво відрізняється від наявних, зокрема якщо евклідова відстань до будь-якого вже знайденого профілю перевищує наперед заданий поріг подібності (наприклад, 10^{-3}).

2.5 Приклади

2.5.1 Робастна задача інспекції

Постановка задачі

Розглянемо класичну гру інспекції, адаптовану до випадку з невизначеними параметрами, за підходом робастної оптимізації. У грі беруть участь два гравці:

- Рядовий працівник (гравець 1), який може *працювати* або *ухилятися*.
- Роботодавець (гравець 2), який може *перевірятиі* або *не перевірятиі*.

Обидва гравці вибирають свої дії одночасно. Роботодавець може уникнути виплати заробітної плати w працівнику, якщо виявить факт ухиляння. Якщо ж працівник працює, він отримує w незалежно від того, чи була перевірка. Однак роботодавець отримує вигоду v лише в разі, якщо працівник працює. Також враховується:

- g – вартість (опортуністичні втрати) для працівника у випадку роботи,
- h – вартість перевірки для роботодавця.

Усі параметри g, h, v вважаються невизначеними, але відомо, що вони належать обмеженим інтервалам:

$$g \in [\underline{g}, \overline{g}], \quad h \in [\underline{h}, \overline{h}], \quad v \in [\underline{v}, \overline{v}].$$

Таким чином, задача є грою з неповною інформацією, де замість апріорного розподілу гравці мають лише інтервальний опис невизначеності.

Матрична форма гри з невизначеністю

Позначимо дії гравців наступним чином:

- Працівник: **S** – ухиляється, **W** – працює.
- Роботодавець: **I** – перевіряє, **N** – не перевіряє.

Матриця виграшів має вигляд:

$$U = \left\{ \left(\begin{array}{cc} (0,\ -h) & (w,\ -w) \\ (w-g,\ v-w-h) & (w-g,\ v-w) \end{array} \right) \;\middle|\; g \in [\underline{g}, \overline{g}],\ v \in [\underline{v}, \overline{v}],\ h \in [\underline{h}, \overline{h}] \right\}$$

де перший компонент у кожній клітинці – виграш працівника, другий – роботодавця.

Робастна постановка та рівновага

Гравці використовують стратегічний підхід на основі **робастної оптимізації**, тобто:

- кожен гравець припускає, що природа вибирає найгірший для нього сценарій (найгірше g,v,h з точки зору його очікуваної вигоди),
- стратегія вибирається так, щоб максимізувати найгіршу можливу очікувану вигоду.

Формально, стратегія гравця i ($i = 1,2$) є розв'язком задачі:

$$x_i^* \in \arg\max_{x_i \in \Delta_i} \min_{P \in U} \mathbb{E}_x[\text{виграш}_i(P)]$$

де Δ_i – симплекс змішаних стратегій, P – конкретна реалізація матриці виграшів.

Чисельна ілюстрація

Нехай:

$$w = 15, \quad \underline{g} = 8, \overline{g} = 12, \quad \underline{v} = 30, \overline{v} = 40, \quad \underline{h} = 3, \overline{h} = 7.$$

У цьому випадку множина матриць виграшів визначається як опуклий паралелепіпед у просторі 3 параметрів. Існує один робастно-оптимізаційний рівноважний розподіл:

- Працівник вибирає **S** з ймовірністю $p^* = \frac{\overline{h}}{w} = \frac{7}{15} \approx 0.47$.
- Роботодавець вибирає **I** з ймовірністю $q^* = \frac{\overline{g}}{w} = \frac{12}{15} = 0.8$.

Це стратегія, яка максимізує найгірший можливий очікуваний виграш для обох гравців, беручи до уваги всю множину U.

2.5.2 Робастна задача «безбілетника»

Постановка задачі

Розглянемо симетричну версію класичної задачі безбілетника з двома гравцями. Кожен з гравців одночасно приймає рішення:

- **1** – внести вклад у створення суспільного блага;
- **2** – утриматися від внеску.

Якщо принаймні один із гравців вносить вклад, суспільне благо створюється, і обидва отримують виграш 1. Гравець, який робить внесок, несе витрати $\tilde{c}$, які є не точно відомими, але належать відомому інтервалу:

$$\tilde{c} \in [\check{c} - \Delta, \check{c} + \Delta]$$

тобто маємо робастну невизначеність. Це інформація загальновідома обом гравцям.

Множина невизначених виграшів

Множина можливих реалізацій виграшів визначається наступною параметризованою матрицею:

$$U = \left\{ \begin{pmatrix} (1-\tilde{c}, 1-\tilde{c}) & (1-\tilde{c}, 1) \\ (1, 1-\tilde{c}) & (0,0) \end{pmatrix} \;\middle|\; \tilde{c} \in [\check{c} - \Delta,\, \check{c} + \Delta] \right\}$$

Це робастна гра з неповною інформацією, але без приватної інформації.

Чисельна постановка

Для чисельного експерименту використано:

$$\underline{c} = \frac{1}{4},\ \overline{c} = \frac{5}{8},$$

У результаті отримано множину матриць виграшу в інтервалі $c \in [0.125, 0.375]$.

Модель рівноваги

Рівновага визначається як пара стратегій $x_1, x_2 \in [0,1]$, де x_i – ймовірність внеску i-го гравця, що максимізує його найгірший очікуваний виграш:

$$x_i^* \in \arg\max_{x_i \in [0,1]} \min_{c \in [0.125,\ 0.375]} \mathbb{E}[u_i(x_1, x_2, c)]$$

Результат

Було виявлено три робастні рівноваги:

$$(x_1^*, x_2^*) \in \left\{ (1,\ 0), \quad (0,\ 1), \quad (1 - \overline{c},\ 1 - \overline{c}) = \left(\frac{3}{8},\ \frac{3}{8}\right) \right\}$$

Тобто, можливі ситуації:

- лише один гравець стабільно вносить вклад (інший – «безбілетник»),
- обидва вносять вклад із ймовірністю 0.375 – збалансована змішана стратегія.

50 випадкових ініціалізацій привели до цих трьох рівноваг. Функція штрафу при знаходженні рішення опускалася нижче 10^{-9}, що свідчить про точне досягнення умов рівноваги.

Цей приклад демонструє здатність методу знаходити множину змішаних рішень.

ВИСНОВКИ

У ході виконання даної роботи було здійснено дослідження проблематики моделювання неантагоністичних конфліктів, приділяючи особливу увагу ситуаціям, що характеризуються невизначеністю вихідних параметрів. Робота охопила як фундаментальні теоретичні аспекти теорії ігор, так і сучасні підходи до аналізу стратегічної взаємодії в умовах неповної інформації.

Насамперед, було проведено аналіз та систематизацію ключових концепцій теорії безкоаліційних ігор. Це включало вивчення класифікації ігрових моделей, формального визначення стратегій та функцій виграшу, а також розкриття сутності принципів оптимальності. Особливу увагу приділено специфіці неантагоністичних конфліктів, де інтереси гравців не є строго протилежними, та центральній ролі рівноваги Неша як ключового принципу розв'язання таких ігор, разом із застосуванням мішаних стратегій для забезпечення існування рівноважних ситуацій.

Далі було досліджено різні підходи до моделювання ігрових взаємодій в умовах невизначеності. Проаналізовано обмеження класичних моделей з повною інформацією та розглянуто альтернативні концепції, зокрема байєсівські ігри, де невизначеність описується ймовірнісними розподілами. Значний акцент зроблено на іграх з інтервальною невизначеністю параметрів, а також на класичних критеріях прийняття рішень, які дозволяють детермінізувати невизначеність шляхом формування «очікуваних» виграшів на основі певних припущень.

Ключовим етапом дослідження було вивчення робастного підходу до теорії ігор як ефективного інструменту для аналізу конфліктів з невизначеними виграшами. Було розглянуто формальну постановку робастної гри, де кожен учасник прагне максимізувати свій гарантований виграш у найгіршому можливому сценарії, визначеному в межах заданої множини невизначеності. Окремо проаналізовано теоретичні гарантії існування робастно-оптимізаційної рівноваги для класу скінченних ігор з компактною множиною невизначеності.

На основі здобутих теоретичних знань та аналізу наукових публіка-

ції, було програмно реалізовано обчислювальний метод для знаходження робастно-оптимізаційних рівноваг у скінченних іграх. Ця методика передбачає перетворення вихідної задачі пошуку рівноваги на систему мультилінійних рівностей та нерівностей, яка ефективно розв'язується шляхом мінімізації спеціально побудованої штрафної функції. Для оптимізації використано квазіньютонівський алгоритм L-BFGS-B, доповнений стратегією мульти-стартів для підвищення надійності та охоплення можливих рівноваг.

Працездатність та ефективність розробленої програмної реалізації було продемонстровано на канонічних прикладах неантагоністичних ігор з елементами невизначеності, зокрема на задачах інспекції та «безбілетника». Отримані в ході чисельних експериментів результати узгоджуються з теоретичними положеннями та даними, наведеними у відповідних наукових джерелах, що підтверджує коректність імплементації та її придатність для аналізу даного класу ігор.

Таким чином, виконана дипломна робота дозволила всебічно дослідити теоретичні та прикладні аспекти моделювання неантагоністичних конфліктів в умовах невизначеності. Поглиблене вивчення сучасних підходів, зокрема робастної теорії ігор, та їх практична реалізація сприяли формуванню цілісного розуміння методів аналізу складних стратегічних взаємодій у невизначеному середовищі.

СПИСОК ЛІТЕРАТУРИ

1. J. Harsanyi. Games with incomplete information played by 'Bayesian' players // Management Science. — 1967. — Vol. 14. — P. 159–182, 320–334, 486–502.
2. Хольмстром Бенгт, Майерсон Роджер. Ефективні та стійкі правила прийняття рішень за неповної інформації // Econometrica. — 1983. — Vol. 51. — P. 1799–1820. — [англ. Holmström B., Myerson R. Efficient and durable decision rules with incomplete information. Econometrica. 1983. Vol. 51. P. 1799–1820.].
3. Кремер Жан, Маклін Річард. Оптимальні стратегії продажу за невизначеності для дискримінуючого монополіста, коли попити є взаємозалежними // Econometrica. — 1985. — Vol. 53, no. 2. — P. 345–362. — [англ. Crémer J., McLean R. Optimal selling strategies under uncertainty for a discriminating monopolist when demands are interdependent. Econometrica. 1985. Vol. 53(2). P. 345–362.].
4. von Neumann John, Morgenstern Oskar. Theory of Games and Economic Behavior. — 1st edition. — Princeton, NJ : Princeton University Press, 1944. — [англ. Theory of Games and Economic Behavior].
5. Агасси Мілдад, Берцимас Димитріос. Робастна теорія ігор // Mathematical Programming. — 2006. — Vol. 107. — P. 231–273. — [англ. Aghassi M., Bertsimas D. Robust Game Theory. Mathematical Programming. 2006. Vol. 107. P. 231–273.].
6. Берцимас Димитріос та ін. Теорія та застосування робастної оптимізації // SIAM Review. — 2011. — Vol. 53, no. 3. — P. 464–501. — [англ. Bertsimas D., Brown D.B., Caramanis C. Theory and applications of robust optimization. SIAM Review. 2011. Vol. 53(3). P. 464–501.].

Додаток А

Код програми

```
from itertools import product as cartesian_product
import numpy as np
import torch
from scipy.optimize import minimize

# For reproducibility
np.random.seed(42)
torch.manual_seed(42)

class RobustGameSolver:
    def __init__(self, num_actions_R, num_actions_C,
     ↪ uncertain_params_bounds,
                 payoff_func, fixed_params=None, game_name="Game"):
        """
        Initializes the robust game solver for a 2-player game.

        Args:
            num_actions_R (int): Number of actions for the Row player.
            num_actions_C (int): Number of actions for the Column player.
            uncertain_params_bounds (dict): {'param_name': (min, max), ...}
            payoff_func (callable): Takes uncertain_param_values (dict) and
             ↪ fixed_params (dict),
                                    returns (payoff_matrix_R_np,
                                     ↪ payoff_matrix_C_np).
            fixed_params (dict, optional): Fixed parameters for payoff_func.
            game_name (str, optional): Name of the game for printing.
        """
        self.game_name = game_name
        self.num_actions_R = num_actions_R
        self.num_actions_C = num_actions_C
        self.uncertain_params_names = list(uncertain_params_bounds.keys())
        self.uncertain_params_bounds = uncertain_params_bounds
        self.payoff_func = payoff_func
        self.fixed_params = fixed_params if fixed_params else {}

        self.extreme_points_uncertainty = self._generate_extreme_points()
        self.k = len(
```

```
35              self.extreme_points_uncertainty)  # Number of extreme points
36
37          # Define variable sizes
38          self.var_sizes = {'xR': self.num_actions_R, 'xC':
            ↪  self.num_actions_C,
39              'zR': 1, 'zC': 1, 'thetaR': self.k if self.k > 0 else 0,
40              # No theta if no uncertainty
41              'thetaC': self.k if self.k > 0 else 0,  # No theta if no
                ↪  uncertainty
42              'phiR': 1, 'phiC': 1}
43          # Adjust k to 1 if no uncertainty params, for nominal payoff case
44          if not self.uncertain_params_names and self.k == 0:
45              self.k = 1  # No thetas needed if k was truly 0 (no uncertainty
                ↪  at all)  # If k was 0 because uncertain_params_bounds was
                ↪  empty, _generate_extreme_points returns [tuple()] -> k=1  #
                ↪  The var_sizes logic for thetaR/C already handles k=0
                ↪  correctly to make them size 0.  # If k=1 (nominal case),
                ↪  thetaR/C will be size 1.
46
47          self.total_vars = sum(self.var_sizes.values())
48
49          # Store slices for easy unpacking of the flat optimization variable
            ↪  vector
50          self.var_slices = {}
51          current_idx = 0
52          for var_name, size in self.var_sizes.items():
53              self.var_slices[var_name] = slice(current_idx, current_idx +
                ↪  size)
54              current_idx += size
55
56          # Pre-calculate payoff matrices at extreme points (as Torch tensors)
57          self.payoff_matrices_at_extreme_points_torch = []
58          if self.k > 0:  # If there are extreme points (includes nominal case
            ↪  where k=1)
59              for ep_idx, ep_values_tuple in enumerate(
60                      self.extreme_points_uncertainty):
61                  # Create a dictionary of uncertain params for the current
                    ↪  extreme point
62                  params_for_payoff_func = {name: val for name, val in
63                                            zip(self.uncertain_params_names,
64                                                ep_values_tuple)}
65
66                  PR_np, PC_np = self.payoff_func(**params_for_payoff_func,
```

```
67                                                **self.fixed_params)
68                 self.payoff_matrices_at_extreme_points_torch.append((
69                     torch.tensor(PR_np, dtype=torch.float64),
70                     torch.tensor(PC_np, dtype=torch.float64)))
71
72     def _generate_extreme_points(self):
73         """Generates all extreme points of the hyperrectangle uncertainty
           ↪ set."""
74         if not self.uncertain_params_names:
75             return [
76                 tuple()]  # Represents one nominal scenario if no uncertain
                   ↪ params
77
78         param_names = self.uncertain_params_names
79         bounds_list = [self.uncertain_params_bounds[name] for name in
80                        param_names]
81
82         extreme_points_tuples = list(cartesian_product(*bounds_list))
83         return extreme_points_tuples
84
85     def _unpack_variables_torch(self, y_torch):
86         """Unpacks the flat torch tensor y_torch into a dictionary of
           ↪ tensors."""
87         unpacked = {}
88         for var_name, s in self.var_slices.items():
89             val = y_torch[s]
90             # Ensure scalar variables are 0-dim tensors
91             if self.var_sizes[var_name] == 1 and len(val.shape) == 0:
92                 unpacked[var_name] = val
93             elif self.var_sizes[var_name] == 1 and len(
94                     val.shape) > 0:  # Sliced 1 element from 1D
95                 unpacked[var_name] = val[0]
96             else:
97                 unpacked[var_name] = val
98         return unpacked
99
100    def _generate_random_initial_guess(self):
101        """Generates a random initial guess for optimization variables."""
102        initial_guess_np = np.empty(self.total_vars, dtype=np.float64)
103
104        # Initialize probability vectors using Dirichlet distribution
105        # Alpha=1 for all components gives a uniform distribution over the
           ↪ simplex
```

```
106         for var_name in ['xR', 'xC', 'thetaR', 'thetaC']:
107             size = self.var_sizes[var_name]
108             if size > 0:
109                 alpha = np.ones(size)
110                 initial_guess_np[
111                     self.var_slices[var_name]] = np.random.dirichlet(alpha)
112 
113         # Initialize z_i, phi_i (e.g., small random values around 0, or
        ↪  z_i=phi_i)
114         for var_pair in [('zR', 'phiR'), ('zC', 'phiC')]:
115             z_name, phi_name = var_pair
116             if self.var_sizes[z_name] > 0:  # Check if variable exists
117                 val_z = np.random.randn() * 0.1  # Small random number for z
118                 initial_guess_np[self.var_slices[z_name]] = val_z
119                 if self.var_sizes[phi_name] > 0:
120                     initial_guess_np[
121                         self.var_slices[phi_name]] = val_z  # phi_i = z_i
122         return initial_guess_np
123 
124     def penalty_function_torch(self, y_torch):
125         """Computes the penalty function (System 15) using PyTorch
        ↪  tensors."""
126         y_vars = self._unpack_variables_torch(y_torch)
127         xR = y_vars.get('xR', torch.empty(0,
128                                           dtype=torch.float64))  # Default
                                           ↪  to empty if not present
129         xC = y_vars.get('xC', torch.empty(0, dtype=torch.float64))
130         zR = y_vars['zR']
131         zC = y_vars['zC']
132         # Handle thetaR/C: if k=1 (nominal), they are effectively fixed at
        ↪  1.0
133         # If k=0 (no uncertainty vars), they don't exist.
134         thetaR = y_vars.get('thetaR', torch.tensor(
135             [1.0] if self.k == 1 and self.var_sizes['thetaR'] > 0 else [],
136             dtype=torch.float64))
137         thetaC = y_vars.get('thetaC', torch.tensor(
138             [1.0] if self.k == 1 and self.var_sizes['thetaC'] > 0 else [],
139             dtype=torch.float64))
140         phiR = y_vars['phiR']
141         phiC = y_vars['phiC']
142 
143         penalty = torch.tensor(0.0, dtype=torch.float64, requires_grad=True)
144         M = 100.0  # Penalty multiplier for normalization/non-negativity
```

```
145
146         # 1. Normalization constraints (sum to 1)
147         if self.var_sizes['xR'] > 0: penalty = penalty + M * (
148                 torch.sum(xR) - 1.0) ** 2
149         if self.var_sizes['xC'] > 0: penalty = penalty + M * (
150                 torch.sum(xC) - 1.0) ** 2
151         if self.var_sizes['thetaR'] > 0: penalty = penalty + M * (
152                 torch.sum(thetaR) - 1.0) ** 2  # Only if thetaR exists
153         if self.var_sizes['thetaC'] > 0: penalty = penalty + M * (
154                 torch.sum(thetaC) - 1.0) ** 2  # Only if thetaC exists
155
156         # 2. Non-negativity constraints (relu(-val)**2 penalizes negative
            ↪ values)
157         if self.var_sizes['xR'] > 0: penalty = penalty + M * torch.sum(
158             torch.relu(-xR) ** 2)
159         if self.var_sizes['xC'] > 0: penalty = penalty + M * torch.sum(
160             torch.relu(-xC) ** 2)
161         if self.var_sizes['thetaR'] > 0: penalty = penalty + M * torch.sum(
162             torch.relu(-thetaR) ** 2)
163         if self.var_sizes['thetaC'] > 0: penalty = penalty + M * torch.sum(
164             torch.relu(-thetaC) ** 2)
165
166         # 3. Constraints: z_i - pi_i(G(l); xR, xC) <= 0
167         if self.k > 0 and len(self.payoff_matrices_at_extreme_points_torch)
            ↪ > 0:
168             for l_idx in range(self.k):  # self.k is at least 1 here
169                 PR_l, PC_l =
                     ↪ self.payoff_matrices_at_extreme_points_torch[l_idx]
170
171                 if self.var_sizes['xR'] > 0 and self.var_sizes['xC'] > 0:
172                     payoff_R_l = torch.matmul(xR.reshape(1, -1),
173                                               torch.matmul(PR_l,
174                                                            xC.reshape(-1,
                                                              ↪ 1))).squeeze()
175                     penalty = penalty + torch.relu(zR - payoff_R_l) ** 2
176
177                     payoff_C_l = torch.matmul(xR.reshape(1, -1),
178                                               torch.matmul(PC_l,
179                                                            xC.reshape(-1,
                                                              ↪ 1))).squeeze()
180                     penalty = penalty + torch.relu(zC - payoff_C_l) ** 2
181
```

```
        # 4. Constraints: sum_l (theta_l^i * pi_i(G(l); x^-i, e_ji)) - phi_i <= 0
        if self.k > 0 and len(self.payoff_matrices_at_extreme_points_torch) > 0:
            # For player R (Row player)
            if self.var_sizes['xR'] > 0 and self.var_sizes['xC'] > 0:
                for jr in range(self.num_actions_R):
                    e_jr = torch.zeros(self.num_actions_R, dtype=torch.float64)
                    e_jr[jr] = 1.0

                    sum_payoffs_R_pure = torch.tensor(0.0, dtype=torch.float64)
                    for l_idx in range(self.k):
                        PR_l, _ = self.payoff_matrices_at_extreme_points_torch[l_idx]
                        payoff_val = torch.matmul(e_jr.reshape(1, -1), torch.matmul(PR_l, xC.reshape(-1, 1))).squeeze()
                        # Use thetaR[l_idx] if thetaR exists and has multiple elements, else use 1.0 if it's nominal
                        current_thetaR_l = thetaR[l_idx] if self.var_sizes['thetaR'] > 1 else (thetaR[0] if self.var_sizes['thetaR'] == 1 else torch.tensor(1.0, dtype=torch.float64))
                        sum_payoffs_R_pure = sum_payoffs_R_pure + current_thetaR_l * payoff_val
                    penalty = penalty + torch.relu(sum_payoffs_R_pure - phiR) ** 2

            # For player C (Column player)
            if self.var_sizes['xR'] > 0 and self.var_sizes['xC'] > 0:
                for jc in range(self.num_actions_C):
                    e_jc = torch.zeros(self.num_actions_C, dtype=torch.float64)
                    e_jc[jc] = 1.0

```

```
213                     sum_payoffs_C_pure = torch.tensor(0.0,
                        ↪ dtype=torch.float64)
214                     for l_idx in range(self.k):
215                         _, PC_l =
                            ↪ self.payoff_matrices_at_extreme_points_torch[
                            l_idx]
216                         payoff_val = torch.matmul(xR.reshape(1, -1),
217                                                   torch.matmul(PC_l,
218                                                                e_jc.reshape(-1,
                                                                ↪ 1))).squeeze()
219                         current_thetaC_l = thetaC[l_idx] if
                            ↪ self.var_sizes['thetaC'] > 1
220                                                          else (
221                             thetaC[0] if self.var_sizes[
222                                                 'thetaC'] == 1 else
                                                 ↪ torch.tensor(
223                                 1.0, dtype=torch.float64))
224                         sum_payoffs_C_pure = sum_payoffs_C_pure +
                            ↪ current_thetaC_l * payoff_val
225                     penalty = penalty + torch.relu(
226                         sum_payoffs_C_pure - phiC) ** 2
227
228         # 5. Constraints: z_i - phi_i = 0 (ensures phi_i is the worst-case
            ↪ expected payoff z_i)
229         penalty = penalty + (zR - phiR) ** 2
230         penalty = penalty + (zC - phiC) ** 2
231
232         return penalty
233
234     def objective_function_scipy(self, y_np):
235         """Wrapper for SciPy: NumPy in, float out."""
236         y_torch = torch.tensor(y_np, dtype=torch.float64,
            ↪ requires_grad=True)
237         penalty = self.penalty_function_torch(y_torch)
238         return penalty.item()
239
240     def jacobian_function_scipy(self, y_np):
241         """Wrapper for SciPy: NumPy in, NumPy grad out, using PyTorch
            ↪ autodiff."""
242         y_torch = torch.tensor(y_np, dtype=torch.float64,
            ↪ requires_grad=True)
243         penalty = self.penalty_function_torch(y_torch)
244
245
```

```
246             if penalty.requires_grad:
247                 penalty.backward()
248                 grad_np = y_torch.grad.numpy().astype(np.float64)
249             else:
250                 # Should not happen
251                 grad_np = np.zeros_like(y_np, dtype=np.float64)
252             return grad_np
253
254         def solve(self, initial_guess_np=None, method='L-BFGS-B', tol=1e-10,
255                   max_iter=1000, verbose=False):
256             """Solves for a robust equilibrium using a local optimizer."""
257             if initial_guess_np is None:
258                 initial_guess_np = self._generate_random_initial_guess()
259
260             # Define bounds for variables (probabilities [0,1], others
                ↪ unbounded)
261             bounds = []
262             for var_name, s in self.var_slices.items():  # Iterate using
                ↪ var_slices to maintain order
263                 size = self.var_sizes[var_name]
264                 if size == 0: continue  # Skip if variable size is zero
265                 if var_name in ['xR', 'xC', 'thetaR', 'thetaC']:
266                     bounds.extend(
267                         [(1e-9, 1.0)] * size)  # Probabilities > 0 and <= 1
268                 else:  # zR, zC, phiR, phiC
269                     bounds.extend([(None, None)] * size)
270
271             if verbose:
272                 print(f"Total variables for {self.game_name}:
                    ↪ {self.total_vars}")
273                 print(f"Number of extreme points (k): {self.k}")
274
275             options = {'disp': verbose, 'maxiter': max_iter, 'ftol': tol,
276                        'gtol': tol * 1e-1}
277
278             result = minimize(self.objective_function_scipy, initial_guess_np,
279                               method=method, jac=self.jacobian_function_scipy,
280                               bounds=bounds, tol=tol, options=options)
281             return result
282
283         def _solutions_are_close(self, sol1_x_np, sol2_x_np, tol=1e-3):
284             """Checks if two solution vectors (primarily strategies) are
                ↪ close."""
```

```
285         vars1 = self._unpack_variables_torch(torch.from_numpy(sol1_x_np))
286         vars2 = self._unpack_variables_torch(torch.from_numpy(sol2_x_np))
287 
288         # Compare primarily the strategies xR and xC
289         close = True
290         if self.var_sizes['xR'] > 0:
291             dist_xR = torch.norm(vars1['xR'] - vars2['xR']).item()
292             if dist_xR >= tol: close = False
293         if self.var_sizes['xC'] > 0 and close:
294             dist_xC = torch.norm(vars1['xC'] - vars2['xC']).item()
295             if dist_xC >= tol: close = False
296 
297         return close
298 
299     def find_multiple_equilibria(self, num_starts=10, solve_tol=1e-8,
300                                  solution_similarity_tol=1e-3,
301                                  max_iter_per_solve=1500,
302                                  success_penalty_threshold=1e-5,
303                                  verbose_solve=False):
304         """Attempts to find multiple distinct robust equilibria using random
            ↪ starts."""
305         print(
306             f"\n--- Finding Multiple Equilibria for {self.game_name}
                ↪ ({num_starts} starts) ---")
307         found_equilibria_np = []
308 
309         for i in range(num_starts):
310             if verbose_solve or (
311                     i % max(1, num_starts // 10) == 0):  # Print progress
312                 print(
313                     f"Multi-start attempt {i + 1}/{num_starts} for
                        ↪ {self.game_name}...")
314 
315             initial_guess = self._generate_random_initial_guess()
316             result = self.solve(initial_guess_np=initial_guess,
                    ↪ tol=solve_tol,
317                                 max_iter=max_iter_per_solve,
318                                 verbose=verbose_solve)
319 
320             final_penalty = self.objective_function_scipy(result.x)
321 
322             if (
```

```
323                     result.success or final_penalty <
                        ↪ success_penalty_threshold * 10) and final_penalty <
                        ↪ success_penalty_threshold:  # Success OR very low
                        ↪ penalty
324
325                 is_new_solution = True
326                 for existing_sol_x_np in found_equilibria_np:
327                     if self._solutions_are_close(result.x,
                        ↪ existing_sol_x_np,
328
                                                     ↪ tol=solution_similarity_tol):
329                         is_new_solution = False
330                         break
331                 if is_new_solution:
332                     print(
333                         f"  Found a new distinct equilibrium (penalty:
                            ↪ {final_penalty:.2e}).")
334                     found_equilibria_np.append(result.x)
335                 elif verbose_solve:
336                     print(
337                         f"  Found a similar equilibrium (penalty:
                            ↪ {final_penalty:.2e}).")
338             elif verbose_solve:
339                 print(
340                     f"  Start {i + 1} did not yield a good solution
                        ↪ (penalty: {final_penalty:.2e}, success:
                        ↪ {result.success}).")
341
342         print(
343             f"\n--- Summary for {self.game_name}: Found
                ↪ {len(found_equilibria_np)} distinct robust equilibria ---")
344         for idx, sol_x_np in enumerate(found_equilibria_np):
345             print(f"\nEquilibrium {idx + 1}:")
346             self.print_solution(sol_x_np)  # Default verbose_penalty=True
347
348         return found_equilibria_np
349
350     def print_solution(self, result_x_np, verbose_penalty=True):
351         """Prints the solution variables in a readable format."""
352         # Unpack using torch then convert to numpy for printing
353         y_vars_torch = self._unpack_variables_torch(
354             torch.tensor(result_x_np, dtype=torch.float64))
355         y_vars_np = {name: (
```

```
356             val.detach().numpy() if isinstance(val, torch.Tensor) else val)
                ↪  for
357             name, val in y_vars_torch.items()}
358
359         if self.var_sizes['xR'] > 0: print(
360             f"Row Player Strategy (xR): {y_vars_np['xR']}")
361         if self.var_sizes['xC'] > 0: print(
362             f"Col Player Strategy (xC): {y_vars_np['xC']}")
363
364         # Ensure zR, zC, phiR, phiC are scalars for printing if they are
            ↪  0-dim arrays
365         def get_scalar(val):
366             return val.item() if isinstance(val,
367                                             np.ndarray) and val.ndim == 0
                                                ↪  else val
368
369         print(
370             f"Row Player Worst-Case Payoff (zR):
                ↪  {get_scalar(y_vars_np['zR']):.4f} (phiR:
                ↪  {get_scalar(y_vars_np['phiR']):.4f})")
371         print(
372             f"Col Player Worst-Case Payoff (zC):
                ↪  {get_scalar(y_vars_np['zC']):.4f} (phiC:
                ↪  {get_scalar(y_vars_np['phiC']):.4f})")
373
374         if self.var_sizes.get('thetaR', 0) > 0:
375             print(
376                 f"Row Player Theta_R (weights on extreme points):
                    ↪  {y_vars_np['thetaR']}")
377         if self.var_sizes.get('thetaC', 0) > 0:
378             print(
379                 f"Col Player Theta_C (weights on extreme points):
                    ↪  {y_vars_np['thetaC']}")
380
381         # Print sums for verification
382         sums_str = []
383         if self.var_sizes['xR'] > 0: sums_str.append(
384             f"Sum xR: {np.sum(y_vars_np['xR']):.4f}")
385         if self.var_sizes['xC'] > 0: sums_str.append(
386             f"Sum xC: {np.sum(y_vars_np['xC']):.4f}")
387         if self.var_sizes.get('thetaR', 0) > 0: sums_str.append(
388             f"Sum thetaR: {np.sum(y_vars_np['thetaR']):.4f}")
389         if self.var_sizes.get('thetaC', 0) > 0: sums_str.append(
```

```
390             f"Sum thetaC: {np.sum(y_vars_np['thetaC']):.4f}")
391         if sums_str: print(", ".join(sums_str))
392 
393         if verbose_penalty:
394             final_penalty = self.objective_function_scipy(result_x_np)
395             print(f"Final penalty function value: {final_penalty:.2e}")
396 
397 
398 # --- Example Game Definitions ---
399 def inspection_payoff_func(g, v, h, w_fixed):
400     PR = np.array([[0, w_fixed], [w_fixed - g, w_fixed - g]],
    ↪  dtype=np.float64)
401     PC = np.array([[-h, -w_fixed], [v - w_fixed - h, v - w_fixed]],
402                   dtype=np.float64)
403     return PR, PC
404 
405 def freerider_payoff_func(c_tilde_val):  # c_tilde_val is the uncertain cost
406     PR = np.array([[1 - c_tilde_val, 1 - c_tilde_val], [1, 0]],
407                   dtype=np.float64)
408     PC = np.array([[1 - c_tilde_val, 1], [1 - c_tilde_val, 0]],
409                   dtype=np.float64)
410     return PR, PC
411 
412 if __name__ == '__main__':
413     print("--- Robust Inspection Game ---")
414     inspection_uncertain_bounds = {'g': (8, 12), 'v': (16, 24), 'h': (4, 6)}
415     inspection_fixed_params = {'w_fixed': 15}
416     solver_inspection = RobustGameSolver(num_actions_R=2, num_actions_C=2,
417         uncertain_params_bounds=inspection_uncertain_bounds,
418         payoff_func=inspection_payoff_func,
419         fixed_params=inspection_fixed_params, game_name="Inspection Game")
420     # For inspection game, one equilibrium is expected
421     inspection_equilibria = solver_inspection.solve(tol=1e-9, max_iter=500,
422                                                     verbose=True)
423 
424     print("\n\n--- Robust Free-Rider Game ---")
425     # Uncertain cost c_tilde_val is in [1/4, 5/8] = [0.25, 0.625]
426     freerider_uncertain_bounds = {'c_tilde_val': (0.25, 0.625)}
427     solver_freerider = RobustGameSolver(num_actions_R=2, num_actions_C=2,
428         uncertain_params_bounds=freerider_uncertain_bounds,
429         payoff_func=freerider_payoff_func, game_name="Free-Rider Game")
430 
431     freerider_equilibria = solver_freerider.find_multiple_equilibria(
```

```
432         num_starts=100,  # Increase for better chance of finding all
433         solve_tol=1e-10, solution_similarity_tol=5e-2,
            ↪  max_iter_per_solve=2500,
434         success_penalty_threshold=1e-6, verbose_solve=False)
```