КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
Факультет комп'ютерних наук та кібернетики
Кафедра системного аналізу та теорії прийняття рішень

Курсова робота
за спеціальністю 124 Системний аналіз
на тему:

ЗАСТОСУВАННЯ ПРИНЦИПУ УЗАГАЛЬНЕННЯ ДЛЯ РОЗВ'ЯЗАННЯ НЕЧІТКОЇ МАТРИЧНОЇ ГРИ

Виконав студент 1-го курсу магістратури
Півень Денис Миколайович _____

Науковий керівник:
професор, доктор фіз.-мат.наук.
Мащенко Сергій Олегович _____

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2023

РЕФЕРАТ

Обсяг роботи 29 сторінок, 1 ілюстрація, 2 таблиці, 15 джерел посилань. ЗАСТОСУВАННЯ ПРИНЦИПУ УЗАГАЛЬНЕННЯ, НЕЧІТКА МАТРИЧНА ГРА, ТЕОРІЯ ІГОР, МАТЕМАТИЧНА ЛОГІКА, ПРИЙНЯТТЯ РІШЕНЬ, МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ, ОПТИМІЗАЦІЯ.

Об'єктом дослідження є ігри з нечіткими параметрами, які знаходяться у центрі уваги дослідників з різних наукових галузей, включаючи економіку, менеджмент, інженерію та інші. Предметом роботи є аналіз задач нечітких матричних ігор та використання відомого алгоритму для вирішення цих задач.

Метою роботи є ознайомлення з сферою нечітких матричних ігор, аналіз алгоритмів вирішення подібних задач для подальшого дослідження та реалізації програмного забезпечення.

Методи розроблення: комп'ютерне моделювання та чисельні методи, розробка програмного продукту розв'язання нечіткої матричної гри використовуючи принцип узагальнення. Інструменти розроблення: безкоштовне, інтегроване середовище розробки IntelliJ IDEA для студентів, бібліотека Apache Commons Math для розв'язання систем лінійних нерівностей за допомогою симплекс методу, мова програмування Java.

Результати роботи: було проведено загальний огляд методів та підходів до розв'язання нечітких матричних ігор, проаналізовано переваги та недоліки використання різних чисельних алгоритмів та комп'ютерного моделювання у процесі вирішення таких завдань. Розроблено програмний продукт «Система розв'язання нечітких матричних ігор за допомогою принципу узагальнення».

Даний програмний продукт сприяє ефективному розв'язання задач у сфері нечітких матричних ігор та може бути корисним для науковців, аналітиків та інших фахівців, що працюють у даній галузі.

ЗМІСТ

ВСТУП

Теорія ігор є методом аналізу взаємодії раціональних агентів, які діють стратегічно, з витоками, що сягають 18 століття. Ця теорія була формалізована в основоположній книзі «Теорія ігор та економічної поведінки» фон Неймана і Моргенштерна в 1944 році. Відтоді теорія ігор швидко розвивалася та отримала широке визнання, застосовуючись в різних сферах, таких як конкуренція, голосування, аукціони, дослідження та розробки, картельна поведінка, штучний інтелект, електронна комерція та біологія. Ігри можна розділити на дві категорії: кооперативні та некооперативні. У кооперативних іграх гравці зосереджені на формуванні коаліцій, тоді як у некооперативних іграх вони зосереджені на виборі стратегій. Ця робота стосується некооперативних ігор у звичайній (або стратегічній) формі.

Теорія ігор традиційно спирається на використання бінарної логіки та припущення, що гравці будуть приймати цілком раціональні рішення. Однак насправді люди часто не роблять повністю раціонального вибору, натомість дотримуючись принципу «обмеженої раціональності». Нечітка логіка, яка дозволяє включати неточну інформацію, може бути використана для кращого представлення цього типу поведінки в теорії ігор. Крім того, традиційна теорія ігор припускає, що всі гравці мають повне знання інформації та результатів гри, що часто не так у сценаріях реального світу. Нечіткі набори, які можуть представляти неточно визначені ситуації, можна використовувати для кращого моделювання цього типу невизначеності в іграх реального світу.

Використання нечітких множин у теорії ігор можна простежити до Обена в 1974 році, який першим представив їх у кооперативній теорії ігор [6]. Пізніше це було розширено Бутнаріу в 1978 році, який представив нечіткі набори в некооперативних іграх [7]. Цю концепцію далі розвинули Біллот та інші дослідники, зокрема Баклі, який використовував принципи прийняття рішень у нечіткому середовищі, Гаразік і Круз, які використовували підхід, заснований на нечітких контролерах, і Кампос, який вивчав некооперативні ігри з нечіткими виграшами [8]. Нещодавно Арфі зробив значний внесок, представивши лінгвістичні нечіткі логічні ігри та підкресливши їх потенціал для використання в соціальних науках [9].

1 ОГЛЯД З ТЕОРІЇ ІГОР З НЕЧІТКИМИ ПАРАМЕТРАМИ

Теорія ігор з нечіткими параметрами є розширенням класичної теорії ігор, яке дозволяє враховувати нечіткість параметрів, що до неї входять, таких як виграш, ймовірності або стратегії гравців.

Однією з основних ідей цієї теорії є використання теорії нечітких множин для представлення нечіткості параметрів гри. У зв'язку з цим, в теорії ігор з нечіткими параметрами застосовуються різні види нечітких множин, такі як трикутні нечіткі множини, трапецієподібні нечіткі множини та інші.

Основні поняття теорії ігор з нечіткими параметрами включають нечітку виграшну функцію, нечітку стратегію гравця, нечітку матрицю виграшів та нечітку оптимальну стратегію гравця.

Теорія ігор з нечіткими параметрами має важливі застосування в різних галузях, таких як економіка, управління та прийняття рішень. Вона дозволяє аналізувати ігри в умовах невизначеності та недостатньої інформації, що дозволяє приймати більш обґрунтовані рішення в таких умовах.

1.1 Нечіткі множини [2]

У традиційній прикладній математиці множина розуміється як сукупність елементів, що мають спільну властивість. Наприклад, множина чисел, що не менше заданого числа, множина векторів, сума компонент яких не перевищує одиницю та інше. При цьому для будь-якого елемента розглядаються лише дві можливості: або цей елемент належить до множини, або ні.

У теорії нечітких множин, що розвивається з 60-х років XX століття, допускається існування елементів, які можуть належати до множини частково або не належати зовсім. Такі множини називаються нечіткими, а елементи, які частково належать до множини, називаються елементами з нечіткою приналежністю.

Однак, як уже зазначалося у вступі, при спробах математичного опису складних систем мова звичайних множин може бути недостатньо точною. Інформація про систему може бути сформульована мовою нечітких понять, які неможливо математично сформалізувати за допомогою звичайних множин.

Поняття нечіткого множини — це спроба математичної формалізації нечіткої інформації з метою її використання при побудові математичних мо-

делей складних систем. В основі цього поняття лежить ідея того, що елементи, які складають дану множину і мають спільну властивість, можуть мати цю властивість в різній мірі, отже, належати до даної множини з різним ступенем. При такому підході висловлювання типу «елемент x належить даній множині» втрачає зміст, оскільки потрібно вказати наскільки сильно або з яким ступенем даний елемент належить до даної множини.

Концепція нечіткої множини — це математичний підхід до формалізації нечіткої або неточної інформації, де елементи в множині можуть мати різні ступені приналежності, представленої функцією приналежності, яка відображається на інтервал [0,1].

$$\mu_C(x) = \begin{cases} 1, & \text{якщо } x \in C \\ 0, & \text{якщо } x \notin C \end{cases}$$

Нечітка множина C у множині X визначається як сукупність пар виду (x, μ(x)), де x належить X, а μ(x) є функцією, яка призначає ступінь приналежності елементу x в C. Таким чином, нечітка множина є більш широким поняттям, ніж звичайна, в тому сенсі, що функція приналежності нечіткої множини може бути, взагалі будь-якою функцією або навіть довільним відображенням.

Нечітка множина називається *пустою*, якщо його функція приналежності рівна нулю на всій множині X, тобто

$$\mu_\emptyset(x) = 0, \forall x \in X,$$

Універсальну множину X також можна описати функцією приналежності виду

$$\mu_X(x) = 1, \forall x \in X.$$

Носієм нечіткої множини A (позначається supp A) з функцією приналежності $\mu_A(x)$ називається множина (у звичайному сенсі) виду

$$\operatorname{supp} A = \{x \mid x \in X, \mu_A(x) > 0\}.$$

Нечітка множина A називається *нормальною*, якщо виконується рівність

$$\sup_{x \in X} \mu_A(x) = 1$$

В іншому випадку нечітка множина називається *субнормальною*.

Нехай A та B – нечіткі множини в X, а $\mu_A(x)$ та $\mu_B(x)$ – їх функції приналежності відповідно. Кажуть, що A *містить у собі* B (тобто $B \subseteq A$), якщо для будь-якого $x \in X$ виконується нерівність $\mu_B(x) \leq \mu_A(x)$. Множини A та B *співпадають*, якщо $\mu_B(x) = \mu_A(x)$ при будь-якому $x \in X$. Якщо нечіткі множини A та B такі, що $B \subseteq A$, то

$$\operatorname{supp} B \subseteq \operatorname{supp} A$$

1.1.1 Операції над нечіткими множинами

Операції над нечіткими множинами, наприклад, об'єднання та перетин, можна визначити різними способами. Нижче наведено декілька таких визначень. Вибір конкретного з них залежить від змісту, який вкладається в відповідну операцію в рамках задачі, що розглядається.

При введенні операцій над нечіткими множинами слід пам'ятати, що клас нечітких множин охоплює і множини у звичайному розумінні. Тому введення визначень в окремому випадку звичайних множин є необхідним у теорії множин. Звичайно, це не стосується до технічних питань, пов'язаних з операціями над нечіткими множинами які для звичайних множин не мають сенсу, наприклад, операції концентрування, розтягування та опуклої комбінації.

Операція об'єднання нечітких множин A та B в X визначається як нечітка множина $A \cup B$ з функцією приналежності вигляду:

$$\mu_{A \cup B}(x) = \max \{\mu_A(x), \mu_B(x)\}, \quad x \in X$$

Якщо $\{A_y\}$ - кінцеве або нескінченне сімейство нечітких множин з функціями приналежності $\mu_{A_y}(x, y)$, де $y \in Y$ - параметр сімейства, то об'єднання $C = \bigcup_Y A_y$ множин цього сімейства є нечіткою множиною з функцією при-

належності вигляду

$$\mu_C(x) = \sup_{y \in Y} \mu_{A_y}(x, y), \quad x \in X$$

Об'єднання нечітких множин A і B в X можна визначити за допомогою алгебраїчної суми їх функцій належності:

$$\mu_{A \cup B}(x) = \begin{cases} 1 & \text{при } \mu_A(x) + \mu_B(x) \geq 1, \\ \mu_A(x) + \mu_B(x) & \text{інакше.} \end{cases}$$

Перетином нечітких множин A та B в X називається нечітка множина $A \cap B$ з функцією приналежності виду

$$\mu_{A \cap B}(x) = \min \{\mu_A(x), \mu_B(x)\}, \quad x \in X.$$

Якщо A_y – нескінчене чи скінчене сімейство нечітких множин з функціями приналежності $\mu_{A_y}(x, y)$, де $y \in Y$ – параметр сімейства, то перетином $C = \bigcap_y A_y$ множин цього сімейства є нечітка множина з функцією приналежності виду

$$\mu_C(x) = \inf_{y \in Y} \mu_{A_y}(x, y), \quad x \in X.$$

Ще один спосіб визначення перетину нечітких множин A та B – використання алгебраїчного добутку їх функцій приналежності:

$$\mu_{A \cap B}(x) = \mu_A(x) \mu_B(x), \quad x \in X$$

Майже всі відмінності між цими визначеннями проявляються у випадку нечітких множин A та B, таких, що $B \subseteq A$, тобто $\mu_B(x) \subseteq \mu_A(x)$ для будь-якого $x \in X$. За першим з цих визначень

$$\mu_{A \cap B}(x) = \mu_B(x), \quad x \in X,$$

тобто функція приналежності множини A фактично не бере участь у цьому визначенні у відповідній функції приналежності, тоді як за визначенням добутку функція приналежності завжди містить функції приналежності обох множин.

Корисним може бути наступна властивість нечітких множин:

$$\operatorname{supp}(A \cup B) = (\operatorname{supp} A) \cup (\operatorname{supp} B)$$
$$\operatorname{supp}(A \cap B) = (\operatorname{supp} A) \cap (\operatorname{supp} B)$$

Легко перевірити, що ці рівності виконуються для будь-якого з наведених вище визначень об'єднання та перетину.

Доповненням нечіткої множини A в X є нечітка множина A' з функцією приналежності, визначеною наступним чином:

$$\mu_{A'}(x) = 1 - \mu_A(x), \quad x \in X.$$

Цікаво, що з таким визначенням доповнення, взагалі кажучи, $A \cap A' \neq \emptyset$ (це відрізняє його від стандартних множин).

Різницею нечітких множин A і B в X називається нечітка множина $A \backslash B$ з функцією приналежності виду

$$\mu_{A\backslash B}(x) = \begin{cases} \mu_A(x) - \mu_B(x) & \text{при } \mu_A(x) \geq \mu_B(x), \\ 0 & \text{інакше.} \end{cases}$$

Зауважимо, що приведене вище визначення доповнення нечіткої множини випливає з даного визначення.

Декартовим добутком нечітких множин $A_1, \ldots, A_n$ у $X_1, \ldots, X_n$ відповідно називається нечітка множина A у декартовому добутку $X = X_1 \times \ldots \times X_n$ з функцією приналежності виду:

$$\mu_A(x) = \min\left\{\mu_{A_1}(x_1), \ldots, \mu_{A_n}(x_n)\right\},$$
$$x = (x_1, \ldots, x_n) \in X.$$

Випуклою комбінацією нечітких множин $A_1, \ldots, A_n$ в X називається нечітка множина A з функцією приналежності вигляду:

$$\mu_A(x) = \sum_{i=1}^{n} \lambda_i \mu_i(x),$$

$$\text{де}, \lambda_i \geq 0, \quad i = 1, 2, \ldots, n, \quad \sum_{i=1}^{n} \lambda_i = 1.$$

Випуклі комбінації нечітких множин можуть застосовуватися, наприклад, у задачах прийняття рішень з кількома нечіткими обмеженнями. Зауважимо, що для звичайних множин ця операція не має сенсу.

Операції концентрування (CON) та розтягування (DIL) нечіткої множини A визначаються наступним чином:

$$\mathrm{CON}\, A = A^2, \quad \mathrm{DIL}\, A = A^{0.5},$$
$$\text{де } \mu_{A^\alpha}(x) = \mu_A^\alpha(x), \quad x \in X, \alpha > 0.$$

Застосування операції концентрування до заданої нечіткої множини фактично означає зменшення «нечіткості» цієї множини. У реальній ситуації це може означати отримання нової інформації, що дозволяє більш точно (більш чітко) описати поняття нечіткої множини. Аналогічним чином, операцію розтягування можна застосовувати для моделювання ситуації, пов'язаної з втратою інформації.

1.1.2 Множини рівня та декомпозиція нечіткої множини

Множина рівня α нечіткої множини A в X називається множиною у звичайному розумінні, складеною з елементів $x \in X$, ступінь належності яких до нечіткої множини A не менше числа α. Таким чином, якщо A_α - множина рівня α нечіткої множини A, то

$$A_a = \{x \mid x \in X, \quad \mu_A(x) \geq \alpha\}.$$

Як буде показано нижче, множинами рівня зручно користуватися при формулюванні та аналізі деяких задач прийняття рішень.

Нехай $(A \cup B)_\alpha$ та $(A \cap B)_\alpha$ – множини рівня α об'єднання та перетину нечітких множин A та B відповідно. Розглянемо зв'язок цих множин з множинами рівня A_α та B_α. Якщо прийняти перші визначення об'єднання та перетину, то, як легко бачити, цей зв'язок такий:

$$(A \cup B)_\alpha = A_\alpha \cup B_\alpha, \quad (A \cap B)_\alpha = A_\alpha \cap B_\alpha.$$

У випадку других визначень маємо лише:

$$(A \cup B)_\alpha \supset A_\alpha \cup B_\alpha, \quad (A \cap B)_\alpha \subset A_\alpha \cap B_\alpha.$$

Якщо $(A_1 \times \ldots \times A_n)_\alpha$ – множина рівня α декартового добутку нечітких множин $A_1, \ldots, A_n$, то з визначення декартового добутку легко бачити, що

$$(A_1 \times \ldots \times A_n)_\alpha = (A_1)_\alpha \times \ldots \times (A_n)_\alpha ,$$

тобто множина рівня α декартового добутку є декартовим добутком множин рівня α його нечітких множин.

Множина (A_α) рівня α будь-якої опуклої комбінації нечітких множин $A_1, \ldots, A_n$ містить перетин множин рівня α усіх її множин, тобто

$$\bigcap_{i=1}^{n} (A_i)_\alpha \subset A_\alpha$$

В деяких випадках зручно використовувати розклад нечіткої множини по її множинам рівня, тобто представленням її у вигляді

$$A = \bigcup_\alpha \alpha A_\alpha,$$

де $\mu_{\alpha A_\alpha}(x) = \alpha \mu_{A_\alpha}(x)$, а об'єднання нечітких множин αA_α береться відповідно до визначенням по всім α від 0 до 1.

1.2 Некооперативні ігри з нечіткими виплатами [3]

Такі ігри мають наступний вигляд:

$$G = \left(X_i, \tilde{f}_i(x) \right)_{i \in I}$$

де $\tilde{f}_i(x), i \in I$ – нечіткі числа. В літературі немає значних результатів щодо цієї загальної форми. Найважливіші результати отримані для двох гравців з кінцевим набором стратегій. Далі представляються основні результати щодо нечітких матричних ігор.

1.2.1 Матрична гра

Матрична гра є скінченною грою двох осіб з нульовою сумою. У такій грі є два гравці: Гравець I та Гравець II. Гравець I максимізує виграш, а Гравець II мінімізує його. Оскільки це гра з нульовою сумою, її можна формально представити наступним чином:

$$G = (X_1, X_2, A)\,,$$

де $X_1 = \{1, 2, \ldots, m\}$ та $X_2 = \{1, 2, \ldots, n\}$ – це множини чистих стратегій Гравців I та II відповідно; A – це матриця $m \times n$, яка представляє виграші Гравця I.

$$A = \begin{pmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{m1} & \ldots & a_{mn} \end{pmatrix}$$

Гравець I вибирає рядки, а Гравець II вибирає стовпці матриці. Якщо гравці обирають пару чистих стратегій (i, j), то Гравець I отримує винагороду a_{ij}, а Гравець II отримує $-a_{ij}$.

1.2.2 Нечіткі матричні ігри

У цьому розділі представляється основна ідея найважливіших підходів для вирішення матричних ігор з нечіткими виграшами. Матрична гра з нечіткими виграшами – це кінцева гра з нульовою сумою для двох осіб із нечіткими виграшами. Її можна представити грою, подібною до звичайної матричної гри, з тією лише різницею, що матриця виграшів нечітка

$$G = \left(S^n, S^m, \tilde{A}\right),$$

$$\tilde{A} = (\tilde{a}_{ij})_{\substack{1 \le i \le m \\ 1 \le j \le n}} = \begin{pmatrix} \tilde{a}_{11} & \ldots & \tilde{a}_{1n} \\ \vdots & \ddots & \vdots \\ \tilde{a}_{m1} & \ldots & \tilde{a}_{mn} \end{pmatrix}.$$

1.2.3 Ранжування підходу нечітких чисел: модель Кампоса

Першу модель, що використовує ранжування нечітких чисел, представив Кампос. Кампос трансформує задачу пошуку рішення нечіткої матричної гри у наступну пару задач нечіткого лінійного програмування:

$$\begin{array}{ll} \text{Max}\, v & \text{Min}\, w \\ p^T \tilde{A} \tilde{\geq} ew & \tilde{A} q \tilde{\leq} ew \\ e^T p = 1, \quad q \geq 0 & e^T q = 1, \quad q \geq 0. \end{array}$$

Тут і далі e представляє вектор відповідної розмірності, компоненти якого є одиницями. Використовуючи метод Ягера для вираження подвійних нечітких обмежень у термінах адекватності $\tilde{t}$ та $\tilde{k}$, пара задач оптимізації нечіткості перетворюється на наступну пару задач лінійного програмування:

$$\begin{aligned} &\text{Min} \sum_{i=1}^{m} u_i \\ &\sum_{i=1}^{m} \left(a_{ij}^L + a_{ij} + a_{ij}^U\right) u_i \geq 3 - \left(t_j^L + t_j + t_j^U\right)(1-\alpha), \quad j = 1, \ldots, n, \\ &\alpha \in [0,1], \quad u_i \geq 0, \quad i = 1, \ldots, m \end{aligned}$$

та

$$\begin{aligned} &\text{Max} \sum_{j=1}^{n} s_j \\ &\sum_{j=1}^{n} \left(a_{ij}^L + a_{ij} + a_{ij}^U\right) s_j \geq 3 - \left(k_i^L + k_i + k_i^U\right)(1-\beta), \quad i = 1, \ldots, m \\ &\beta \in [0,1], \quad s_j \geq 0, \quad j = 1, \ldots, n \end{aligned}$$

де $\tilde{a}_{ij} = \left(a_{ij}^L, a_{ij}, a_{ij}^U\right)$, $\tilde{t} = \left(t_j^L, t_j, t_j^U\right)$ і $\tilde{k} = \left(k_i^L, k_i, k_i^U\right)$ є трикутними нечіткими числами (TFN).

1.2.4 Багатоцільовий підхід: модель Лі

Лі використовував наступний порядок ранжування TFN. Нехай $\tilde{a} = \left(a^L, a, a^U\right)$ та $\tilde{b} = \left(b^L, a, b^U\right)$ – два TFN. Тоді $\tilde{a} \tilde{\leq} \tilde{b}$ якщо $a^L \leq b^L, a \leq b$ та $a^U \leq b^U$.

Нехай $\tilde{v} = \left(v^L, v, v^U\right)$ та $\tilde{w} = \left(w^L, w, w^U\right)$ – два TFN. Тоді $(\tilde{v}, \tilde{w})$ називається розумним рішенням гри, якщо існує $(\bar{p}, \bar{q}) \in S^m \times S^n$ такі, що

(i) $\bar{p}^T \tilde{A} q \geq \tilde{v}, \quad \forall q \in S^n$

(ii) $p^T \tilde{A} \bar{q} \leq \tilde{w}, \quad \forall p \in S^m$.

Позначимо $V = \{\tilde{v} / \tilde{v}$ виконує (i) $\}$ та $W = \{\tilde{w} / \tilde{w}$ виконує (ii) $\}$.

Елемент $(\tilde{v}, \tilde{w}) \in V \times W$ називається рішенням гри, якщо

(1) $\tilde{v}^* \tilde{\geq} \tilde{v}, \quad \forall \tilde{v} \in V$

(2) $\tilde{w}^* \tilde{\leq} \tilde{w}, \quad \forall \tilde{w} \in W$

Проблему пошуку рішення гри можна тоді перетворити на задачу вирішення наступної пари задач багатокритеріального програмування:

$$
\begin{aligned}
&\text{Max } \left(v^L, v, v^U\right) \\
&\sum_{i=1}^{m} a_{ij}^L p_i \geq v^L, \quad j = 1, \ldots, n, \\
&\sum_{i=1}^{m} a_{ij} p_i \geq v, \quad j = 1, \ldots, n, \\
&\sum_{i=1}^{m} a_{ij}^U p_i \geq v^U, \quad j = 1, \ldots, n, \\
&e^T p = 1, \quad p \geq 0
\end{aligned}
$$

та

$$
\begin{aligned}
&\text{Min } \left(w^L, w, w^U\right) \\
&\sum_{j=1}^{n} a_{ij}^L q_j \leq w^L, \quad i = 1, \ldots, m, \\
&\sum_{j=1}^{n} a_{ij} q_j \leq w, \quad i = 1, \ldots, m, \\
&\sum_{j=1}^{n} a_{ij}^U q_j \leq w^U, \quad i = 1, \ldots, m, \\
&e^T q = 1, \quad q \geq 0.
\end{aligned}
$$

1.2.5 Підхід функції ранжирування: Модель Бектора

Бектор використовує подвійність у нечіткому лінійному програмуванні та перший індекс Ягера, щоб представити рішення для матричної гри з нечі-

ткими виграшами. Вони визначають два типи розв'язків для гри: розумний та простий.

Нехай $\tilde{v}, \tilde{w}$ — два нечітких числа. Тоді $(\tilde{v}, \tilde{w})$ називається **розумним розв'язком гри**, якщо існує пара $x^* \in S^m, y^* \in S^n$ що задовольняє

(i) $x^{*T}\tilde{A}y \succeq \tilde{v}, \quad \forall y \in S^n,$

(ii) $x^T\tilde{A}y^* \preceq \tilde{w}, \quad \forall x \in S^m.$

Якщо $(\tilde{v}, \tilde{w})$ є розумним розв'язком, то $\tilde{v}$ (відповідно, $\tilde{w}$) називається **прийнятним значенням** для Гравця I (відповідно, Гравеця II).

Нехай T_1 і T_2 – набори всіх розумних значень $\tilde{v}$ і $\tilde{w}$ для Гравців I та II відповідно, де $\tilde{v}, \tilde{w}$ є нечіткими числами. Нехай існують $\tilde{v}^* \in T_1$ та $\tilde{w}^* \in T_2$ такі, що

(i) $F(\tilde{v}^*) \geq F(\tilde{v}), \quad \forall \tilde{v} \in T_1$

(ii) $F(\tilde{w}^*) \leq F(\tilde{w}), \quad \forall \tilde{w} \in T_2.$

$F : N(R) \to R$ – **функція дефазифікації**, де $N(R)$ – набір дійсних нечітких чисел. Далі Вводиться наступна пара задач нечіткого лінійного програмування для гравців I та II:

$$\text{Max } F(\tilde{v})$$
$$x^T\tilde{A}y \succeq \tilde{p}\tilde{v}, \quad \forall y \in S^n, x \in S^m$$

та

$$\text{Min } F(\tilde{w})$$
$$x^T\tilde{A}y \preceq \tilde{q}\tilde{w}, \quad x \in S^m, \quad y \in S^n$$

Де $x^T\tilde{A}y \succeq \tilde{p}\tilde{v}$ і $x^T\tilde{A}y \preceq \tilde{q}\tilde{w}$ – подвійні нечіткі нерівності з адекватністю $\tilde{p}$ та $\tilde{q}$ відповідно.

Потім задачі перетворюється на наступну чітку пару задач:

$$\text{Max } V$$
$$\begin{cases} \sum_{i=1}^m F(\tilde{a}_{ij})\, x_i \geq V + (1-\gamma)F(\tilde{p}), \quad j = 1, \ldots, n, \\ \sum_{i=1}^m x_i = 1, x \geq 0, 0 \leq \gamma \leq 1 \end{cases}$$

та

$$\text{Min } W$$
$$\begin{cases} \sum_{j=1}^n F(\tilde{a}_{ij})\, y_j \leq W + (1-\rho)F(\tilde{q}), \quad i = 1, \ldots, m, \\ \sum_{j=1}^n y_j = 1, y \geq 0, 0 \leq \rho \leq 1 \end{cases}$$

2 ЗАСТОСУВАННЯ ПРИНЦИПУ УЗАГАЛЬНЕННЯ [4]

Лю та Као розробляють метод вирішення для матричних ігор з нечіткими виграшами на основі принципу розширення. Розглянемо нечітку матричну гру. Тоді завдання для Гравця I (Гравець II) полягає в тому, щоб знайти оптимальний розв'язок задачі. Використовуючи принцип розширення, Лю та Као визначають цінність гри як нечітке число $\tilde{Z}$ з такою функцією належності:

$$\mu_{\tilde{Z}}(z) = \mathrm{Sup}_{a_{ij}} \mathrm{Min}\left\{\mu_{\tilde{a}_{ij}}(a_{ij}), \quad \forall i,j/z = Z(a)\right\}.$$

Для фіксованого значення рівня зрізу α верхня межа $\tilde{Z}_\alpha^U$ і нижня межа $\tilde{Z}_\alpha^L$ з $\tilde{Z}$ обчислюються шляхом вирішення наступної пари дворівневих задач, відповідно,

$$\tilde{Z}_\alpha^U = \mathrm{Max}_{(a_{ij})_\alpha^L \le a_{ij} \le (a_{ij})_\alpha^U, \quad \forall i,j} \mathrm{Max}_p\, v$$
$$\begin{cases} \sum_{i=1}^{m} a_{ij} p_i \ge v, \quad j = 1,\ldots,n, \\ \sum_{i=1}^{m} p_i = 1, \\ p_i \ge 0, \quad i = 1,\ldots,m \end{cases}$$

та

$$\tilde{Z}_\alpha^L = \mathrm{Min}_{(a_{ij})_\alpha^L \le a_{ij} \le (a_{ij})_\alpha^U, \forall i,j} \mathrm{Max}_p\, v$$
$$\begin{cases} \sum_{i=1}^{m} a_{ij} p_i \ge v, \quad j = 1,\ldots,n, \\ \sum_{i=1}^{m} p_i = 1, \\ p_i \ge 0, \quad i = 1,\ldots,m \end{cases}$$

Аналогічну пару дворівневих задач можна сформулювати для Гравця II. Тоді використовуючи заміну змінної $t_{ij} = a_{ij}p_i$ та $l_{ij} = a_{ij}q_j$ отримуємо

таку пару задач лінійного програмування:

$$\tilde{Z}_\alpha^U = \operatorname{Max} v,$$
$$\begin{cases} \sum_{i=1}^{m} t_{ij} \geq v, \quad j = 1, \ldots, n, \\ \sum_{i=1}^{m} p_i = 1, \\ (a_{ij})_\alpha^L p_i \leq t_{ij} \leq (a_{ij})_\alpha^U p_i, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n, \\ p_i \geq 0, t_{ij}, v \quad i = 1, \ldots, m, \quad j = 1, \ldots, n. \end{cases}$$

та

$$\tilde{Z}_\alpha^L = \operatorname{Min} u,$$
$$\begin{cases} \sum_{j=1}^{n} l_{ij} \leq u, \quad i = 1, \ldots, m, \\ \sum_{j=1}^{n} q_j = 1, \\ (a_{ij})_\alpha^L q_j \leq l_{ij} \leq (a_{ij})_\alpha^U q_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n, \\ q_j \geq 0, l_{ij}, u \quad i = 1, \ldots, m, \quad j = 1, \ldots, n. \end{cases}$$

Цільове значення $\tilde{Z}_\alpha^L$ разом із $\tilde{Z}_\alpha^U$, становить α-переріз $\left[\tilde{Z}_\alpha^L, \tilde{Z}_\alpha^U\right]$, де значення $\tilde{Z}$ нечіткої гри лежить на зазначеному α-рівні. Подібне перетворення можна застосувати до пари задач, пов'язаних з Гравцем II. Це показує, що нечітке значення, розраховане для двох гравців, однакове. Зверніть увагу, що більшість підходів до розв'язання гри не забезпечують однакові нечіткі або чіткі значення для обох гравців. Нехай $R(z) = \left(Z_\alpha^U\right)^{-1}$ та $L(z) = \left(Z_\alpha^L\right)^{-1}$ – права та ліва функції форми нечіткого значення $\tilde{Z}$, тоді $\tilde{Z}$ можна отримати як

$$\mu_{\tilde{z}}(z) = \begin{cases} L(z), & z_{\alpha=0}^L \leq z \leq z_{\alpha=1}^L, \\ 1, & z_{\alpha=1}^L \leq z \leq z_{\alpha=1}^U, \\ R(z), & z_{\alpha=1}^U \leq z \leq z_{\alpha=0}^L. \end{cases}.$$

У більшості випадків рівняння $R(z) = \left(Z_\alpha^U\right)^{-1}$ і $L(z) = \left(Z_\alpha^L\right)^{-1}$ неможливо знайти аналітично. Однак для різних рівнів можливостей $\alpha, \tilde{Z}_\alpha^L$ і $\tilde{Z}_\alpha^U$ можна обчислити для наближення форм $R(z)$ і $L(z)$ відповідно.

2.1 Формулювання проблеми

В реальному світі існують випадки, коли виграші невідомі або відомі лише приблизно, що може бути прикладом конкуренції між двома компаніями з різними стратегіями реклами або політичними кандидатами, які обирають різні рішення для своєї кампанії. В таких випадках застосовуються нечіткі числа для кількісного опису невизначеності при прийнятті рішень.

У цьому розділі пропонується метод рішення для двоособової гри з нульовою сумою, де виграші є лише приблизно відомими та можуть бути представлені нечіткими числами. Через те, що виграші є нечіткими, значення гри також є нечітким.

Розглянемо матричну гру двох гравців, де гравець 1 має m стратегій для вибору, а гравець 2 має n стратегій для вибору в кожному раунді гри. Якщо в одному раунді гравець 1 обирає стратегію i, а гравець 2 обирає стратегію j, то гравець 1 отримує a_{ij} одиниць винагороди, а гравець 2 втрачає a_{ij}. Матриця $A = |a_{ij}|$ відома як матриця виграшів гри.

Позначимо ймовірність, з якою гравець 1 використовує стратегію i, як x_i, а ймовірність, з якою гравець 2 використовує стратегію j, як y_j. Вектори ймовірностей $\boldsymbol{x} = (x_1, \ldots, x_m)$ та $\boldsymbol{y} = (y_1, \ldots, y_n)$ відомі як змішані стратегії гравця 1 та гравця 2 відповідно. Гравець 1 прагне знайти змішану стратегію $\boldsymbol{x}$, яка приносить найбільшу очікувану виграш. Коли гравець 2 використовує стратегію j, очікуваний виграш для гравця 1 дорівнює $a_{1j}x_1 + \cdots + a_{mj}x_m$. Оскільки гравець 2 може використовувати будь-яку стратегію j, гравець 1 намагається забезпечити мінімальний очікуваний виграш v через обмеження $a_{1j}x_1 + \cdots + a_{mj}x_m \geq v$ для всіх $j = 1, \ldots, n$. Відповідно до критерію мінімакс, оптимальна стратегія для гравця 1 може бути сформульована за допомогою наступної лінійної програми:

$$
\begin{aligned}
Z = \operatorname{Max} v \\
&\sum_{i=1}^{m} a_{ij}x_i \geq v, \quad j = 1, \ldots, n \\
&\sum_{i=1}^{m} x_i = 1 \\
&x_i \geq 0, i = 1, .., m
\end{aligned}
$$

З точки зору гравця 2, його очікуваний втрати становлять $a_{i1}y_1 + \cdots + a_{in}y_n$, коли гравець 1 використовує стратегію i. Оскільки гравець 1 може використовувати будь-яку стратегію i, гравець 2 намагається мінімізувати очікувані втрати u за допомогою обмеження $a_{i1}y_1 + \cdots + a_{in}y_n \leq u$ для $i = 1, \ldots, m$. Відповідно, лінійна програма для гравця 2 для знаходження оптимальної стратегії $\boldsymbol{y}$ та відповідного значення гри u є:

$$
\begin{aligned}
Z = \text{Min } u & \\
& \sum_{j=1}^{n} a_{ij}y_j \leq u, \quad i = 1, \ldots, m \\
& \sum_{j=1}^{n} y_j = 1 \\
& y_j \geq 0, j = 1, .., n
\end{aligned}
$$

Очевидно, ця модель є двоїстою до попередньої моделі отже, оптимальні стратегії для обох гравців можна вирішити з будь-якої задачі.

Припустимо, що виграш a_{ij} приблизно відомий і може бути представлений опуклою нечіткою множиною $\tilde{A}ij$. Нехай $\mu\tilde{A}_{ij}$ позначає її функцію належності; ми маємо:

$$
\tilde{A}_{ij} = \left\{ \left(a_{ij}, \mu_{\tilde{A}_{ij}} \left(a_{ij} \right) \right) \mid a_{ij} \in S\left(\tilde{A}_{ij} \right) \right\}
$$

де $S\left(\tilde{A}ij\right)$ - це підтримка $\tilde{A}ij$, яка позначає універсальний набір виграшів. Без обмеження загальності, усі виграші в цьому дослідженні припускаються нечіткими числами, оскільки чіткі значення можуть бути представлені здегенерованими функціями належності, які мають лише одне значення в своїх областях. Коли виграші є нечіткими числами, ліва частина обмеження $\sum_{i=1}^{m} a_{ij}x_i \geq v$ у моделі (1) стає $\sum_{\tilde{i}=1}^{m} \tilde{A}ijxi$, яке є нечітким числом. Різне значення a_{ij} у нечіткому множині $\tilde{A}_{ij}$ призводить до різного значення цільової функції. Відповідно, значення цільової функції стає нечітким числом $\widetilde{Z}$ замість чіткого числа. Метою цієї статті є похід функції належності значення гри $\tilde{Z}$. У випадку, коли значення виграшів є нечіткими, гравці повинні вибирати свої стратегії, враховуючи нечіткість. Це може призвести до більш складного аналізу та потреби в модифікації лінійних програм для врахування нечіткості. Однак, концепція нечіткого програмування допомагає розшири-

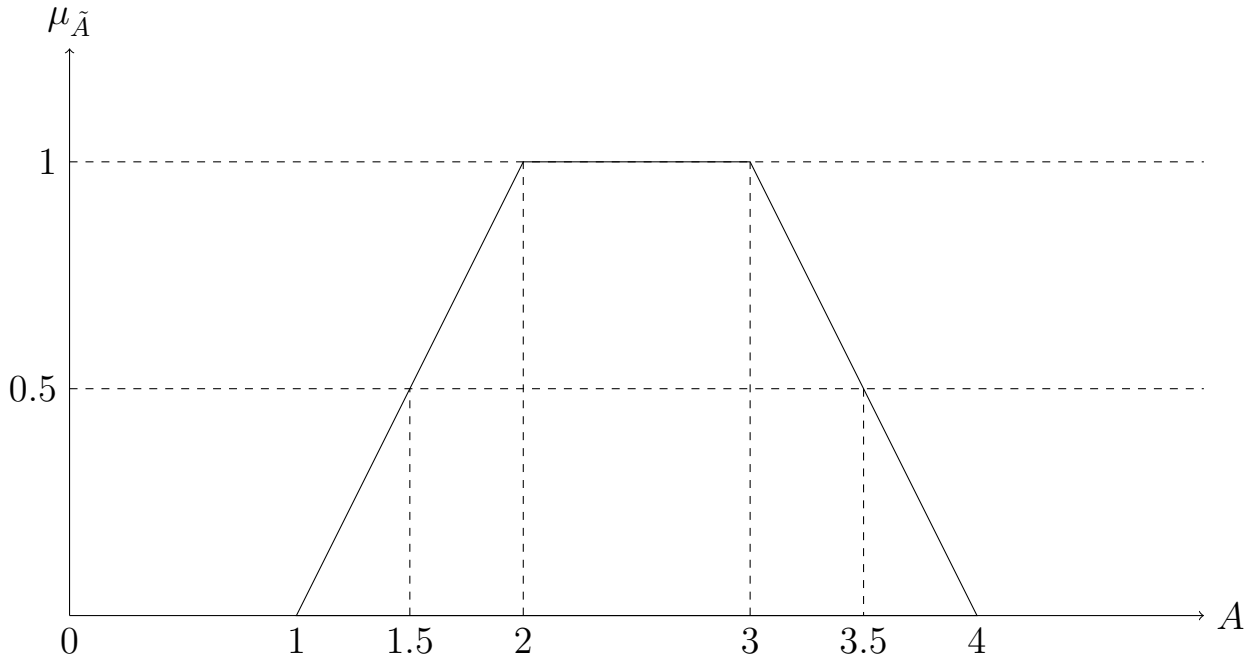
ти існуючі методи розв'язання задач гри на випадок, коли виграші мають нечіткі значення.

$$\mu_{\tilde{Z}}(z) = \mathrm{Sup}_{a_{ij}} \min \left\{ \mu_{\tilde{A}_{ij}}(a_{ij}), \quad \forall i,j \mid z = Z(\mathbf{a}) \right\}$$

де $Z(\mathbf{a})$ визначено в Моделі (1). Застосування принципу розширення до $\tilde{Z}$ можна розглядати як застосування цього принципу розширення до α-перерізу $\tilde{Z}$.[17] У рівнянні (4) задіяно кілька функцій належності. Отримати $\mu_{\tilde{Z}}$ від $\mu_{\tilde{A}ij}$ безпосередньо у закритій формі майже неможливо. Згідно з (4), $\mu\tilde{Z}(z)$ - це мінімум тих $\mu_{\tilde{A}ij}, \forall i,j$, таких, що $z = Z(\mathbf{a})$. На конкретному рівні можливості α, нам потрібно $\mu\tilde{A}ij\,(aij) \geq \alpha$ та принаймні одне $\mu_{\tilde{A}ij}(aij), \forall i,j$, рівне α, таким чином, що $z = Z(\mathbf{a})$, для задоволення $\mu_{\tilde{Z}}(z) = \alpha$. Щоб знайти функцію належності $\mu_{\tilde{Z}}$, достатньо знайти праву функцію форми та ліву функцію форми $\mu_{\tilde{Z}}$. Це еквівалентно знаходженню верхньої межі Z_α^{U} та нижньої межі Z_α^{L} α-перерізів $\widetilde{Z}$.

Позначимо α-переріз $\tilde{A}_{ij}$ як

$$\begin{aligned}(A_{ij})_\alpha &= \left[(A_{ij})_\alpha^{\mathrm{L}}, (A_{ij})_\alpha^{\mathrm{U}}\right] \\ &= [\min_{a_{ij}} \left\{ a_{ij} \in S\left(\tilde{A}_{ij}\right) \mid \mu_{\tilde{A}_{ij}}(a_{ij}) \geq \alpha \right\}, \\ &\quad \max_{a_{ij}} \left\{ a_{ij} \in S\left(\tilde{A}_{ij}\right) \mid \mu_{\tilde{A}_{ij}}(a_{ij}) \geq \alpha \right\}]\end{aligned}$$



Розглянемо α-переріз для $\alpha = 0.5$ на нечіткому числі $\tilde{A} = (1, 2, 3, 4)$

для кращого розуміння. При $\alpha = 0.0$, $(A)_0^{\mathrm{L}} = 1$ та $(A)_0^{\mathrm{U}} = 4$. При $\alpha = 0.5$, $(A)_{0.5}^{\mathrm{L}} = 1.5$ та $(A)_{0.5}^{\mathrm{U}} = 3.5$. При $\alpha = 1.0$, $(A)_1^{\mathrm{L}} = 2$ та $(A)_1^{\mathrm{U}} = 3$. На кожному α-перерізі нечітке число $\tilde{A}$ може приймати значення на відповідному інтервалі $\left[(A)_\alpha^{\mathrm{L}}, (A)_\alpha^{\mathrm{U}}\right]$.

Графік показує функцію належності нечіткого числа на інтервалі $[1, 4]$ з максимальним значенням 1 при 2 та 3. Пунктирні лінії показують інтервали для кожного α-перерізу.

Оскільки Z_α^{U} є максимумом $Z(\mathbf{a})$ та Z_α^{L} є мінімумом $Z(\mathbf{a})$, вони можуть бути виражені як

$$Z_\alpha^{\mathrm{U}} = \max\left\{Z(\mathbf{a}) \mid (A_{ij})_\alpha^{\mathrm{L}} \le a_{ij} \le (A_{ij})_\alpha^{\mathrm{U}}, \quad \forall i, j\right\}$$
$$Z_\alpha^{\mathrm{L}} = \min\left\{Z(\mathbf{a}) \mid (A_{ij})_\alpha^{\mathrm{L}} \le a_{ij} \le (A_{ij})_\alpha^{\mathrm{U}}, \quad \forall i, j\right\}$$

Значення a_{ij}, яке досягає найбільшого значення для v, може бути визначено за допомогою такої дворівневої математичної програми:

$$Z_\alpha^{\mathrm{U}} = \underset{\substack{(A_{ij})_\alpha^{\mathrm{L}} \le a_{ij} \le (A_{ij})_\alpha^{\mathrm{U}} \\ \forall i,j}}{\mathrm{Max}} \begin{cases} \mathrm{Max}_x\, v \\ \sum_{i=1}^m a_{ij} x_i \ge v, \quad j = 1, \ldots, n \\ \sum_{i=1}^m x_i = 1 \\ x_i \ge 0, \qquad\qquad \text{for } i = 1, \ldots, m \end{cases}$$

Принаймні одне значення a_{ij} повинно відповідати межі свого α-розрізу, щоб задовольнити $\mu_{\tilde{Z}}(z) = \alpha$. У моделі (7) внутрішня програма обчислює очікувану винагороду для кожного a_{ij}, заданого зовнішньою програмою, тоді як зовнішня програма визначає значення a_{ij}, яке генерує найбільшу винагороду. Значення цілі Z_α^{U} є верхньою межею очікуваної винагороди для α-рівневого розрізу для гравця 1.

Аналогічно, щоб знайти значення a_{ij}, яке дає найменшу винагороду, дворівневу математичну програму можна сформулювати, просто замінивши зовнішню програму моделі (7) з «Max» на «Min»:

$$Z_\alpha^{\mathrm{L}} = \underset{\substack{(A_{ij})_\alpha^{\mathrm{L}} \le a_{ij} \le (A_{ij})_\alpha^{\mathrm{U}} \\ \forall i,j}}{\mathrm{Min}} \begin{cases} \mathrm{Max}_x\, v \\ \sum_{i=1}^m a_{ij} x_i \ge v, \quad j = 1, \ldots, n \\ \sum_{i=1}^m x_i = 1 \\ x_i \ge 0, \qquad\qquad \text{for } i = 1, \ldots, m \end{cases}$$

У цій моделі внутрішня програма обчислює очікувану винагороду для кожного заданого значення a_{ij}, тоді як зовнішня програма визначає значення a_{ij}, яке дає найменшу винагороду. Значення цілі Z_α^{L} є нижньою межею очікуваної винагороди для α-рівневого розрізу для гравця 1. Коли нечітка виплата $\tilde{A}ij$ дегенерується до точкової виплати aij, зовнішня програма моделей та зникає. У цьому випадку моделі та дегенеруються до звичайної моделі матричних ігор. Ця властивість показує, що розроблена тут модель на основі нечітких значень є узагальненням традиційної моделі з точковими значеннями.

Ми також можемо сформулювати верхню межу та нижню межу очікуваної винагороди для гравця 2 на основі моделі:

$$Z_\alpha^{\mathrm{U}} = \underset{\substack{(A_{ij})_\alpha^{\mathrm{L}} \le a_{ij} \le (A_{ij})_\alpha^{\mathrm{U}} \\ \forall i,j}}{\mathrm{Max}} \begin{cases} \mathrm{Min}_y\, u \\ \sum_{j=1}^n a_{ij} y_j \le u, \quad i = 1, \ldots, m \\ \sum_{j=1}^n y_j = 1 \\ y_j \ge 0, \qquad\qquad \text{для } j = 1, \ldots, n \end{cases}$$

$$Z_\alpha^{\mathrm{L}} = \underset{\substack{(A_{ij})_\alpha^{\mathrm{L}} \le a_{ij} \le (A_{ij})_\alpha^{\mathrm{U}} \\ \forall i,j}}{\mathrm{Min}} \begin{cases} \mathrm{Min}_y\, u \\ \sum_{j=1}^n a_{ij} y_j \le u, \quad i = 1, \ldots, m \\ \sum_{j=1}^n y_j = 1 \\ y_j \ge 0, \qquad\qquad \text{для } j = 1, \ldots, n \end{cases}$$

Хоча дворівнева модель чітко виражає проблему, її не можна використовувати для знаходження рішення. Потрібне перетворення. У наступному розділі ми перетворимо дворівневу математичну програму на звичайну однорівневу лінійну програму, з якої можна отримати нечітке значення нечіткої гри.

2.2 Однорівнева трансформація

Моделі є двома математичними програмами для обчислення верхньої та нижньої меж α-перерізу розподіленого значення матричної гри з точки зору гравця 1, тоді як вони розглядаються з точки зору гравця 2. Оскільки як внутрішня, так і зовнішня програми мають однакову операцію максимізації, їх можна об'єднати в один рівень з урахуванням обмежень обох програм

одночасно. Тобто,

$$
\begin{aligned}
Z_\alpha^{\mathrm{U}} = & \operatorname{Max} v \\
\text{s.t.} & \sum_{i=1}^{m} a_{ij} x_i \geq v, \quad j = 1, \ldots, n \\
& \sum_{i=1}^{m} x_i = 1 \\
& (A_{ij})_\alpha^{\mathrm{L}} \leq a_{ij} \leq (A_{ij})_\alpha^{\mathrm{U}}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \\
& x_i \geq 0, \quad \text{for } i = 1, \ldots, m
\end{aligned}
$$

Ця модель є нелінійною через нелінійні члени $a_{ij}x_i$. Можна застосувати заміну змінних $p_{ij} = a_{ij}x_i$, щоб перетворити нелінійну програму на лінійну:

$$
\begin{aligned}
Z_\alpha^{\mathrm{U}} = & \operatorname{Max} v \\
\text{s.t.} & \sum_{i=1}^{m} p_{ij} \geq v, \quad j = 1, \ldots, n \\
& \sum_{i=1}^{m} x_i = 1 \\
& (A_{ij})_\alpha^{\mathrm{L}} x_i \leq p_{ij} \leq (A_{ij})_\alpha^{\mathrm{U}} x_i, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \\
& x_i \geq 0, \quad \text{for } i = 1, \ldots, m
\end{aligned}
$$

Третій набір обмежень є результатом множення кожного доданка на x_i та заміни $a_{ij}x_i$ на p_{ij}. Розв'язання цієї лінійної програми дозволяє отримати верхню межу α-перерізу нечіткої величини нечіткої гри. Після того, як знайдені оптимальні рішення p_{ij}^* та x_i^*, обчислюється оптимальне значення a_{ij}^* за формулою $a_{ij}^* = p_{ij}^*/x_i^*$. Якщо x_i дорівнює 0, тоді a_{ij}^* може приймати будь-яке значення у діапазоні між $(A_{ij})\,\alpha^{\mathrm{L}}$ та $(Aij)_\alpha^{\mathrm{U}}$.

Розв'язати другу модель не так просто, як розв'язати першу, оскільки її зовнішня та внутрішня програми мають різні напрямки оптимізації – одна для мінімізації, а інша для максимізації. Для отримання рішення потрібно змінити напрямок програми. Враховуючи теорему дуальності, дуальна задача має таке саме оптимальне значення цільової функції, як і її примітивна задача, коли існує оптимальний розв'язок. Отже, ми можемо замінити внутрішню програму другої моделі її дуальною задачею, щоб перетворити її на проблему мінімізації. Після цієї заміни і зовнішня, і внутрішня програми

мають однаковий напрямок для оптимізації:

$$Z_\alpha^{\mathrm{L}} = \underset{\substack{(A_{ij})_\alpha^{\mathrm{L}} \le a_{ij} \le (A_{ij})_\alpha^{\mathrm{U}} \\ \forall i,j}}{\mathrm{Min}} \begin{cases} \mathrm{Min}_y\, u \\ \sum_{j=1}^n a_{ij} y_j \le u, \quad i = 1, \ldots, m \\ \sum_{j=1}^n y_j = 1 \\ y_j \ge 0, \qquad\qquad \text{для } j = 1, \ldots, n \end{cases}$$

Тепер, коли і внутрішня програма, і зовнішня програма мають однакову операцію мінімізації, їх можна об'єднати в однорівневу програму, враховуючи обмеження на двох рівнях одночасно. Нелінійні члени $a_{ij}y_j$ можуть бути лінеаризовані аналогічно попередній моделі шляхом заміни на q_{ij} та обмеження $A_{ij}^{\mathrm{L}} \le a_{ij} \le A_{ij}^{\mathrm{U}}$, помножені на y_j, відповідно. Отримана лінійна програма:

$$\begin{aligned} & Z_\alpha^{\mathrm{L}} = \mathrm{Min}\, u \\ & \sum_{j=1}^n q_{ij} \le u, i = 1, \ldots, m \\ & \sum_{j=1}^n y_j = 1 \\ & (A_{ij})_\alpha^{\mathrm{L}} y_j \le q_{ij} \le (A_{ij})_\alpha^{\mathrm{U}} y_j, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \\ & y_j \ge 0, \quad q_{ij}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n \end{aligned}$$

Коли отримані оптимальні значення q_{ij}^* та y_j^*, оптимальне значення a_{ij}^* обчислюється як $a_{ij}^* = q_{ij}^*/y_j^*$. Значення цільової функції Z_α^{L}, разом з Z_α^{U}, отриманим з моделі (12), $\left[Z_\alpha^{\mathrm{L}}, Z_\alpha^{\mathrm{U}}\right]$, складає α-переріз, в якому значення нечіткої гри знаходиться на конкретному рівні α.

Для двох рівнів можливостей α_1 та α_2 таких, що $0 < \alpha_2 < \alpha_1 \le 1$, допустимі області, визначені α_1, менші, ніж ті, що визначені α_2. Відповідно, ми маємо $Z_{\alpha_1}^{\mathrm{U}} \le Z_{\alpha_2}^{\mathrm{U}}$ та $Z_{\alpha_1}^{\mathrm{L}} \ge Z_{\alpha_2}^{\mathrm{L}}$. Ця властивість, заснована на визначенні «опуклої нечіткої множини», гарантує опуклість $\tilde{Z}$. Якщо обидві Z_α^{U} та Z_α^{L} є оберненими щодо α, то можна отримати праву функцію форми $R(z) = \left(Z_\alpha^{\mathrm{U}}\right)^{-1}$ та ліву функцію форми $L(z) = \left(Z_\alpha^{\mathrm{L}}\right)^{-1}$. Від $R(z)$ та $L(z)$ функція належності $\mu_{\tilde{Z}}$ будується як

$$\mu_{\tilde{Z}} = \begin{cases} L(z), & Z_{\alpha=0}^{\mathrm{L}} \le z \le Z_{\alpha=1}^{\mathrm{L}} \\ 1, & Z_{\alpha=1}^{\mathrm{L}} \le z \le Z_{\alpha=1}^{\mathrm{U}} \\ R(z), & Z_{\alpha=1}^{\mathrm{U}} \le z \le Z_{\alpha=0}^{\mathrm{U}} \end{cases}$$

У більшості випадків значення Z_α^{U} та Z_α^{L} можуть не мати аналітичних розв'язків. Однак чисельні розв'язки для Z_α^{U} та Z_α^{L} на різних рівнях можливості α можуть бути зібрані для апроксимації форм правої $R(z)$ та лівої $L(z)$ функцій, відповідно.

2.3 Приклад

Розглянемо матричну гру, в якій обидва гравці мають чотири стратегії на вибір. З 16 виплат, показаних у таблиці I, 8 є нечіткими числами. Ті, що мають три компоненти, є трикутними, а ті, що мають чотири компоненти, є трапецієподібними. Концептуально, верхня межа Z_α^{U} та нижня межа Z_α^{L} нечіткого значення гри на конкретному рівні α можуть бути сформульовані як

$$\begin{aligned}
&Z_\alpha^{\mathrm{U}} = \operatorname{Max} v \\
&(26,27,28,30)x_1 - 28x_2 + (-27,-25,-22)x_3 + 30x_4 \ge v \\
&-10x_1 + 20x_2 + (10,11,12,14)x_3 + (-38,-36,-35,-34)x_4 \ge v \\
&-20x_1 - 10x_2 + 20x_3 + (-34,-32,-30)x_4 \ge v \\
&(-3,0,3)x_1 + (32,34,35,37)x_2 - 30x_3 + (32,33,34,36)x_4 \ge v \\
&x_1 + x_2 + x_3 + x_4 = 1 \\
&x_1, x_2, x_3, x_4 \ge 0 \\
&Z_\alpha^{\mathrm{L}} = \operatorname{Min} u \\
&(26,27,28,30)y_1 - 10y_2 - 20y_3 + (-3,0,3)y_4 \le u \\
&-28y_1 + 20y_2 - 10y_3 + (32,34,35,37)y_4 \le u \\
&(-27,-25,-22)y_1 + (10,11,12,14)y_2 + 20y_3 - 30y_4 \le u \\
&30y_1 + (-38,-36,-35,-34)y_2 + (-34,-32,-30)y_3 \\
&+ (32,33,34,36)y_4 \le u \\
&y_1 + y_2 + y_3 + y_4 = 1 \\
&y_1, y_2, y_3, y_4 \ge 0
\end{aligned}$$

	1	2	3	4
1	$(26, 27, 28, 30)$	-10	-20	$(-3, 0, 3)$
2	-28	20	-10	$(32, 34, 35, 37)$
3	$(-27, -25, -22)$	$(10, 11, 12, 14)$	20	-30
4	30	$(-38, -36, -35, -34)$	$(-34, -32, -30)$	$(32, 33, 34, 36)$

Таблиця I. Приклад виграшів гри.

На основі моделей верхня межа Z_α^{U} та нижня межа Z_α^{L} на різних рівнях α розраховуються відповідно. Таблиця II містить α-перерізи гри на 11 різних значеннях α: $0, 0.1, 0.2, \ldots, 1.0$. α-переріз $\widetilde{Z}$ відображає ймовірність того, що значення гри з'явиться у пов'язаному діапазоні. Чим більше значення α, тим нижчий ступінь невизначеності. Оскільки розподілене значення гри знаходиться в діапазоні, різні α-перерізи показують різні інтервали та рівень невизначеності значення гри. У цьому прикладі, хоча значення гри нечітке, його найбільш ймовірне значення знаходиться між -2.8213 та -2.6405, і неможливо, щоб його значення вийшло за межі діапазону -3.6105 та -1.5959.

α	0.0	0.1	0.2	0.3	0.4	0.5
Z_α^{U}	-1.5959	-1.7009	-1.8058	-1.9105	-2.0151	-2.1196
Z_α^{L}	-3.6105	-3.5317	-3.4530	-3.3742	-3.2953	-3.2164

α	0.6	0.7	0.8	0.9	1.0
Z_α^{U}	-2.2240	-2.3283	-2.4325	-2.5365	-2.6405
Z_α^{L}	-3.1375	-3.0585	-2.9795	-2.9005	-2.8213

Таблиця II. α-перерізи нечіткого значення гри на 11 α значень.

2.4 Програмна реалізація

Програма спрямована на застосування принципу узагальнення для вирішення нечітких матричних ігор з використанням реалізації на мові Java. Основною метою є розв'язання задач лінійного програмування з нечіткими числами. Для досягнення цієї мети ми використовуємо можливості оптимізації бібліотеки Apache Commons Math.

Встановлення

- Клонуйте репозиторій з GitHub.
- Завантажте бібліотеку Apache Commons Math.
- Розпакуйте завантажену папку і скопіюйте commons-math3-3.6.1 в папку проєкту.
- Відкрийте проєкт у вашому IDE (наприклад, Eclipse, IntelliJ IDEA).
- Додайте папку commons-math3-3.6.1 до проєкту.

Використання

- Створіть файл з назвою input.txt і помістіть його в папку input_output.
- У файлі input.txt введіть матрицю з нечіткими числами, дотримуючись прикладу, наведеного в проєкт.
- Запустіть проєкт.
- Програма буде зчитувати input.txt, формувати нечітку матрицю, створювати системи лінійних нерівностей з альфа-перерізами та розв'язувати їх (кількість альфа-перерізів можна налаштовувати).
- Результат буде записано у файл з назвою output.txt у папці input_output.
- Крім того, буде створено графік точок розв'язку у форматі SVG з назвою image.svg.

ВИСНОВКИ

У цій курсовій роботі було проаналізовано принцип узагальнення для розв'язання нечіткої матричної гри. Було розглянуто теоретичний підхід до цього принципу та його застосування у практичних задачах.

Для реалізації цього підходу була написана програма, яка дозволяє вирішувати нечіткі матричні ігри з використанням принципу узагальнення. У програмі було враховано різні варіації таких ігор, включаючи ігри з різними рівнями нечіткості та різними розмірами матриць. Для цього було використано мову програмування Java та вбудовані функції для розв'язування лінійних нерівностей.

У результаті виконання курсової роботи було отримано досвід у розробці програм для розв'язання нечітких задач та показано, як принцип узагальнення може бути застосований у різних сферах. Було також розглянуто можливості для подальшого вдосконалення програми та застосування принципу узагальнення у складніших задачах.

Для покращення роботи програми та її можливостей у майбутньому, можна додати функції для відображення графічних результатів, які б допомогли користувачам більш ефективно аналізувати результати гри. Також можна розглянути можливості використання більш складних математичних методів для вирішення нечітких матричних ігор.

Як студент першого курсу магістратури, я вперше зіткнувся з нечіткою теорією та матричними іграми, тому ця курсова робота була важливим кроком у моєму навчанні та розумінні цих понять. Під час виконання цієї роботи я отримав досвід у розробці програм на Java та навчився застосовувати теоретичні знання на практиці. Також я розумію, як цей принцип може бути застосований в інших галузях та буду використовувати ці знання в майбутньому для розв'язання складних задач.

ЛІТЕРАТУРА

[1] Мулен Э. Теория игр с примерами из математической экономики. — М., 1985;

[2] Орловский С. А. Проблемы принятия решения при нечеткой исходной информации. — М., 1981;

[3] Larbani, M.: Non cooperative fuzzy games in normal form: A survey.

[4] Shiang-Tai L., Chiang K.: Solution of Fuzzy Matrix Games: An Application of the Extension Principle

[5] J. Koshal, A. Nedic, and U. V. Shanbhag, "Distributed algorithms for aggregative games on graphs," Operations Research, vol. 64, no. 3, pp. 680–704, 2016.

[6] Owen, G. (1974). Cooperative games using fuzzy sets. Theory and Decision, 4(4), 307-312.

[7] Butnariu, D. (1978). Game theory and fuzzy sets. BUSEFAL, 1(1), 139-151.

[8] Buckley, J.J. (1986). Fuzzy hierarchical analysis. Fuzzy Sets and Systems, 17(3), 233-247.

[9] Arfi, B. (2009). Linguistic fuzzy-logic games: Modeling social interactions in social sciences. Journal of Economic Interaction and Coordination, 4(1), 1-20.

[10] Shapley, L. S. (1953). A value for n-person games. Annals of Mathematics, 28(1), 307-317.

[11] Nash, J. F. (1950). Equilibrium points in n-person games. Proceedings of the National Academy of Sciences, 36(1), 48-49.

[12] Binmore, K. (2007). Game theory: A very short introduction. Oxford, England: Oxford University Press.

[13] C. De Persis, S. Grammatico «Continuous-time integral dynamics for aggregative game equilibrium seeking» [Електронний ресурс]: arXiv preprint arXiv:1803.10448v1, 2018

URL: `https://arxiv.org/pdf/1803.10448.pdf`

[14] R. Zhu, J. Zhang, K. You, and T. Basar, "Asynchronous networked aggregative games" [Електронний ресурс]: arXiv preprint arXiv:2101.08973, 2021.

URL: `https://arxiv.org/pdf/2101.08973.pdf`

[15] R. Zhu, J Zhang and K You «Networked Aggregative Games with Linear Convergence» [Електронний ресурс]: arXiv preprint arXiv:2105.05465v1, 2021

URL: `https://arxiv.org/pdf/2105.05465.pdf`

ДОДАТОК А

У цьому проекті були використані такі технології, як Java та бібліотека Apache Commons Math для оптимізації та розв'язування задач лінійного програмування з нечіткими числами.

Посилання на GitHub: https://github.com/PivDen2000/Fuzzy-games

Лістинг 1: input.txt

```
a=11
n=4
m=4
[26,27,28,30], [-10], [-20], [-3,0,3]
[-28], [20], [-10], [32,34,35,37]
[-27,-25,-22], [10,11,12,14], [20], [-30]
[30], [-38,-36,-35,-34], [-34,-32,-30], [32,33,34,36]
```

Лістинг 2: output.txt

```
0,0000  0,1000  0,2000  0,3000  0,4000  0,5000  0,6000  0,7000  0,8000  0,9000  1,0000
-1,5959 -1,7009 -1,8058 -1,9105 -2,0151 -2,1196 -2,2240 -2,3283 -2,4325 -2,5365 -2,6405
-3,6105 -3,5317 -3,4530 -3,3742 -3,2953 -3,2164 -3,1375 -3,0585 -2,9795 -2,9005 -2,8213
```

Лістинг 3: Main.java

```
1
2  import java.io.IOException;
3  import java.nio.file.Paths;
4
5  public class Main {
6      public static void main(String[] args) {
7          try {
8              var inputOutputFolder = Paths.get("./input_output/");
9              var inputPath =
                   inputOutputFolder.resolve("input.txt");
10             var outputPath =
                   inputOutputFolder.resolve("output.txt");
11
12             var numCuts = FuzzyReader.readNumCuts(inputPath);
13             var matrix = FuzzyReader.readMatrix(inputPath);
```

```
14              var result =
                    FuzzySolver.calcAlphaCutsFuzzyValue(matrix,
                    numCuts);
15              FuzzyWriter.writeResult(result,
                    outputPath.toString());
16              FuzzyDrawer.draw(result, outputImage.toString());
17
18          } catch (IOException e) {
19              e.printStackTrace();
20          }
21      }
22  }
```

Лістинг 4: FuzzyNumber.java

```
1
2   import java.util.Arrays;
3
4   public class FuzzyNumber {
5       public Double[] arr;
6
7       public FuzzyNumber(Double[] arr) {
8           this.arr = arr;
9       }
10
11      @Override
12      public String toString() {
13          return Arrays.toString(arr);
14      }
15  }
```

Лістинг 5: FuzzyReader.java

```
1
2   import java.io.IOException;
3   import java.nio.charset.Charset;
4   import java.nio.file.Files;
5   import java.nio.file.Path;
6   import java.util.Arrays;
7   import java.util.stream.Stream;
8
9   public class FuzzyReader {
```

```
10      public static int readNumCuts(Path inputPath) throws
           IOException {
11          return readVar(inputPath, 'a');
12      }
13
14      private static int readVar(Path inputPath, char var) throws
           IOException {
15          try (Stream<String> lines = Files.lines(inputPath,
               Charset.defaultCharset())) {
16              return lines.filter(line -> line.startsWith(var +
                   "="))
17                      .map(line -> line.substring(2))
18                      .findFirst()
19                      .map(Integer::parseInt)
20                      .orElseThrow(() -> new
                           IllegalArgumentException("Invalid␣input␣
                           file:␣cannot␣find␣'" + var + "='"));
21          }
22      }
23
24
25      public static FuzzyNumber[][] readMatrix(Path inputPath)
           throws IOException {
26          var m = readVar(inputPath, 'm');
27          var n = readVar(inputPath, 'n');
28          var matrix = new FuzzyNumber[m][n];
29
30          var input = Files.readString(inputPath,
               Charset.defaultCharset());
31          var lines = Arrays.stream(input.split("\r\n"))
32                  .filter(line -> !line.isEmpty())
33                  .toArray(String[]::new);
34
35          for (int i = 3; i < m + 3; i++) {
36              var values = lines[i].replaceAll("\\s+",
                   "").split(",\\[");
37              for (int j = 0; j < n; j++) {
38                  var value = values[j].replaceAll("[\\[\\]]", "");
39                  var fuzzyValues = value.split(",");
40                  var numbers = new Double[fuzzyValues.length];
41                  for (int k = 0; k < fuzzyValues.length; k++) {
```

```
42                      numbers[k] =
                           Double.parseDouble(fuzzyValues[k]);
43                  }
44                  Arrays.sort(numbers);
45                  matrix[i - 3][j] = new FuzzyNumber(numbers);
46              }
47          }
48          return matrix;
49      }
50  }
```

Лістинг 6: FuzzySolver.java

```
1
2   import org.apache.commons.math3.linear.MatrixUtils;
3   import org.apache.commons.math3.optim.MaxIter;
4   import org.apache.commons.math3.optim.linear.*;
5   import org.apache.commons.math3.optim.nonlinear.scalar.GoalType;
6
7   import java.util.ArrayList;
8   import java.util.Arrays;
9   import java.util.stream.IntStream;
10
11  public class FuzzySolver {
12      public static Double[][]
           calcAlphaCutsFuzzyValue(FuzzyNumber[][] matrix, int
           numCuts) {
13          var cuts = IntStream.range(0, numCuts)
14                  .mapToDouble(i -> i * 1.0 / (numCuts - 1))
15                  .boxed()
16                  .toArray(Double[]::new);
17
18          return new Double[][]{
19                  cuts,
20                  Arrays.stream(cuts)
21                          .map(cut -> solve(fuzzyToNum(matrix,
                               cut, true), false))
22                          .toArray(Double[]::new),
23                  Arrays.stream(cuts)
24                          .map(cut -> solve(fuzzyToNum(matrix,
                               cut, false), true))
25                          .toArray(Double[]::new)
```

```
26          };
27      }
28
29      private static double[][] fuzzyToNum(FuzzyNumber[][] matrix,
           double cut, boolean isMax) {
30          var result = new double[matrix.length][matrix[0].length];
31
32          for (int i = 0; i < matrix.length; i++) {
33              for (int j = 0; j < matrix[0].length; j++) {
34                  var values = matrix[i][j].arr;
35
36                  if (values.length == 1) {
37                      result[i][j] = values[0];
38                      continue;
39                  }
40
41                  var max = values[values.length - 1];
42                  var min = values[0];
43                  var mid = Arrays.copyOfRange(values, 1,
                       values.length - 1);
44
45                  var maxCutValue = max - cut * (max -
                       mid[mid.length - 1]);
46                  var minCutValue = min + cut * (mid[0] - min);
47
48                  result[i][j] = isMax ? maxCutValue : minCutValue;
49              }
50          }
51
52          return result;
53      }
54
55      public static Double solve(double[][] matrix, boolean isMax)
           {
56          var m = matrix.length;
57          var n = matrix[0].length;
58
59          var c = new double[n + 1];
60          c[n] = 1;
61          LinearObjectiveFunction f = new
               LinearObjectiveFunction(c, 0);
```

```
62
63          if (isMax) {
64              matrix = MatrixUtils.createRealMatrix(matrix)
65              .transpose()
66              .getData();
67          }
68
69          var A = new double[2 * m + 1][n + 1];
70          for (int i = 0; i < m; i++) {
71              System.arraycopy(matrix[i], 0, A[i], 0, n);
72              A[i][n] = -1;
73          }
74          Arrays.fill(A[m], 1);
75          A[m][n] = 0;
76          Arrays.fill(A, m + 1, A.length,
77                  IntStream.range(0, m)
78                          .mapToDouble(i -> i == m - 1 ? 1 : 0)
79                          .toArray());
80
81          var b = new int[2 * m + 1];
82          b[m] = 1;
83
84          Relationship[] r = new Relationship[2 * m + 1];
85          Arrays.fill(r, m + 1, 2 * m + 1, Relationship.GEQ);
86          r[m] = Relationship.EQ;
87          Arrays.fill(r, 0, m, isMax ? Relationship.GEQ :
               Relationship.LEQ);
88
89          var constraints = new ArrayList<LinearConstraint>();
90          for (int i = 0; i < A.length; i++) {
91              constraints.add(new LinearConstraint(A[i], r[i],
                   b[i]));
92          }
93
94          var solver = new SimplexSolver();
95          var optSolution = solver.optimize(new MaxIter(100), f,
               new LinearConstraintSet(constraints),
96                    isMax ? GoalType.MAXIMIZE : GoalType.MINIMIZE,
                         new NonNegativeConstraint(false));
97
98          return optSolution.getValue();
```

```
99      }
100 }
```

Лістинг 7: FuzzyWriter.java

```
1
2  import java.io.BufferedWriter;
3  import java.io.FileWriter;
4  import java.io.IOException;
5  import java.util.Arrays;
6
7  public class FuzzyWriter {
8      public static void writeResult(Double[][] matrix, String
           fileName) {
9          try (BufferedWriter writer = new BufferedWriter(new
               FileWriter(fileName))) {
10             for (var row : matrix) {
11                 writer.write(String.join("\t", Arrays.stream(row)
12                         .map(val -> String.format("%.4f", val))
13                         .toArray(String[]::new)));
14                 writer.newLine();
15             }
16             System.out.println("Matrix␣has␣been␣written␣to␣
                   file.");
17         } catch (IOException e) {
18             System.err.println("Error␣writing␣matrix␣to␣file:␣"
                   + e.getMessage());
19         }
20     }
21 }
```

Лістинг 8: FuzzyDrawer.java

```
1
2  import java.io.FileWriter;
3  import java.io.IOException;
4  import java.io.Writer;
5  import java.util.ArrayList;
6
7  public class FuzzyDrawer {
8
9      public record Point(double x, double y, int intX, int intY) {
10     }
```

```
11
12      public static void draw(Double[][] matrix, String path) {
13
14          final int width = 400;
15          final int height = 400;
16
17          double zero = 0.0;
18          double min = Double.POSITIVE_INFINITY;
19          double max = Double.NEGATIVE_INFINITY;
20
21          var data = new Double[matrix.length][matrix[0].length];
22          for (int i = 0; i < matrix.length; i++) {
23              System.arraycopy(matrix[i], 0, data[i], 0,
                    matrix[i].length);
24          }
25
26          for (int i = 1; i < data.length; i++) {
27              for (var value : data[i]) {
28                  min = Math.min(min, value);
29                  max = Math.max(max, value);
30              }
31          }
32
33          min = Math.min(min, zero);
34          max = Math.max(max, zero);
35          zero = (zero - min) / (max - min);
36
37          for (int i = 1; i < data.length; i++) {
38              for (int j = 0; j < data[i].length; j++) {
39                  data[i][j] = (data[i][j] - min) / (max - min);
40              }
41          }
42
43          var minX = (int) (0.125 * width);
44          var minY = (int) (0.875 * height);
45          var maxX = (int) (0.875 * width);
46          var maxY = (int) (0.125 * height);
47          var zeroX = (int) ((0.125 + zero * 0.75) * width);
48          var zeroY = (int) (0.875 * height);
49
50          var points = new ArrayList<Point>();
```

```
51
52          StringBuilder polygon = new StringBuilder("<polygon␣
               points=\"");
53          for (int i = 0; i < data[0].length; i++) {
54              points.add(new Point(matrix[1][i], matrix[0][i],
55                      (int) (minX + data[1][i] * (maxX - minX)),
56                      (int) (minY + data[0][i] * (maxY - minY))));
57          }
58          for (int i = data[0].length - 1; i >= 0; i--) {
59              points.add(new Point(matrix[2][i], matrix[0][i],
60                      (int) (minX + data[2][i] * (maxX - minX)),
61                      (int) (minY + data[0][i] * (maxY - minY))));
62          }
63          points.forEach(point ->
               polygon.append(point.intX).append(",").append(point.intY).a
               "));
64          polygon.append("\"␣fill=\"none\"␣stroke=\"black\"␣/>");
65
66          points.add(new Point(0.0, 0.0, zeroX, zeroY));
67          points.add(new Point(max, 0.0, maxX, minY));
68          points.add(new Point(max, 1.0, maxX, maxY));
69
70          StringBuilder svgContent = new StringBuilder("<svg␣
               xmlns=\"http://www.w3.org/2000/svg\"␣width=\"" +
               width + "\"␣height=\"" + height + "\">"
71                  + "<line␣x1=\"0\"␣y1=\"" + zeroY + "\"␣x2=\"" +
                       width + "\"␣y2=\"" + zeroY + "\"␣
                       stroke=\"black\"␣/>"
72                  + "<line␣x1=\"" + zeroX + "\"␣y1=\"0\"␣x2=\"" +
                       zeroX + "\"␣y2=\"" + height + "\"␣
                       stroke=\"black\"␣/>");
73
74          points.forEach(point -> svgContent
75                  .append("<text␣x=\"").append(point.intX + 3)
76                  .append("\"␣y=\"").append(point.intY - 3)
77                  .append("\"␣font-family=\"Arial\"␣
                       font-size=\"3\"␣fill=\"black\">(")
78                  .append(Math.round(point.x * 100.0) /
                       100.0).append(",␣")
79                  .append(Math.round(point.y * 100.0) /
                       100.0).append(")</text>"));
```

```
80          svgContent.append(polygon).append("</svg>");
81
82          try (Writer writer = new FileWriter(path)) {
83              writer.write(svgContent.toString());
84          } catch (IOException e) {
85              e.printStackTrace();
86          }
87      }
88  }
```