МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ВЕТЕРИНАРНОЇ МЕДИЦИНИ ТА БІОТЕХНОЛОГІЙ ІМ. С.З. ҐЖИЦЬКОГО
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

другого (магістерського) рівня вищої освіти

на тему: **«Інтелектуальна система підтримки прийняття рішень для оптимізації складових аграрних проєктів на основі нечіткої логіки та генетичних алгоритмів»**

Виконав: студент групи Іт-62

Спеціальності 126 «Інформаційні системи та технології»
(шифр і назва)
Бобеляк Олег Володимирович
(Прізвище та ініціали)

Керівник: д.т.н., професор Тригуба А.М.
(Прізвище та ініціали)

Рецензент: к.т.н., доцент Бабич М.І.
(Прізвище та ініціали)

ДУБЛЯНИ-2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЛЬВІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ПРИРОДОКОРИСТУВАННЯ
ФАКУЛЬТЕТ МЕХАНІКИ, ЕНЕРГЕТИКИ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Другий (магістерський) рівень вищої освіти
Спеціальність 126 «Інформаційні системи та технології»

«ЗАТВЕРДЖУЮ»

Завідувач кафедри_____

д.т.н., проф. А.М. Тригуба
«____» _____2025 р.

ЗАВДАННЯ

на кваліфікаційну роботу студенту

Бобеляку Олегу Володимировичу

1. Тема роботи: «Інтелектуальна система підтримки прийняття рішень для оптимізації складових аграрних проєктів на основі нечіткої логіки та генетичних алгоритмів»

Керівник роботи Тригуба Анатолій Миколайович, професор
затверджені наказом по університету від 28.02.2025 року № 140/к-с.

2. Строк подання студентом роботи 10.12.2025 р.

3. Вихідні дані до роботи: *експертні лінгвістичні оцінки для побудови функцій належності; база нечітких правил Мамдані; параметри генетичного алгоритму (розмір популяції, оператори селекції, кросинговеру та мутації, критерії зупинки); дані тестових сценаріїв оптимізації; програмні засоби реалізації.*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)
Вступ.
Стан реалізації аграрних проєктів та аналіз існуючого інструментарію.
Вибір інструментарію та проєктування інтелектуальної системи підтримки прийняття рішень.
Розробка програмних модулів інтелектуальної системи підтримки прийняття рішень.
Охорона праці та безпека у надзвичайних ситуаціях.
Визначення економічної ефективності.
Висновки та пропозиції.
Список використаної літератури.

5. Перелік ілюстраційного матеріалу (з точним зазначенням обов»язкових слайдів): *аналіз стану реалізації аграрних проєктів та аналіз існуючого інструментарію; вибір інструментарію та проєктування інтелектуальної системи підтримки прийняття рішень; розробка програмних модулів інтелектуальної системи підтримки прийняття рішень; економічна ефективність.*

6. Консультанти з розділів:

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1, 2, 3, 5	*Тригуба А.М., зав. кафедри інформаційних технологій*		
4	*Городецький І.М., доцент кафедри інженерної механіки*		

7. Дата видачі завдання 28 лютого 2025 р.

Календарний план

№ з/п	Назва етапів кваліфікаційної роботи	Терміни виконання етапів роботи	Примітка
1	*Написання першого розділу*	*28.02-20.03.25*	
2	*Виконання другого розділу та аркушів ілюстраційного матеріалу до нього*	*21.03-14.06.25*	
3.	*Виконання третього розділу та аркушів ілюстраційного матеріалу до нього*	*15.06.25-10.07.25*	
4.	*Написання розділу «Охорона праці та безпека у надзвичайних ситуаціях»*	*11.07-31.08.25*	
5.	*Оцінення ефективності запропонованої системи*	*01.09-31.10.25*	
6.	*Завершення оформлення розрахунково-пояснювальної записки та аркушів ілюстраційного матеріалу*	*01-30.11.25*	
7.	*Завершення роботи вцілому*	*01-10.12.25*	

Студент _____ Бобеляк О.В.
(підпис)

Керівник роботи _____ Тригуба А.М.
(підпис)

УДК 004.89:63:519.876.5

Інтелектуальна система підтримки прийняття рішень для оптимізації складових аграрних проєктів на основі нечіткої логіки та генетичних алгоритмів.

Бобеляк О.В. Кафедра інформаційних технологій – Дубляни, ЛНУВМ, 2025.

Кваліфікаційна робота: 83 с. текст. част., 15 рис., 8 табл., 14 арк. ілюстраційного матеріалу, 32 джерел.

Подано особливості та актуальність оптимізації аграрних проєктів в умовах невизначеності та ризиків, притаманних сучасній аграрній сфері. Проаналізовано специфіку прийняття рішень у проєктному менеджменті та обґрунтовано доцільність застосування інтелектуальних систем для підвищення ефективності управлінських процесів. Виконано огляд наукових досліджень щодо застосування Fuzzy–GA моделей у сфері аграрного проєктування та сформульовано мету, завдання й структуру кваліфікаційної роботи.

Розроблено концепцію побудови інтелектуальної системи підтримки прийняття рішень (ІСППР) для оптимізації витрат, тривалості та якості аграрних проєктів на основі інтеграції нечіткої логіки Мамдані та генетичних алгоритмів. Реалізовано математичну модель оцінювання параметрів проєкту із використанням трикутних і трапецоїдних функцій належності та бази нечітких правил. Розроблено програмне забезпечення, що включає модулі нечіткого виведення, оптимізації та графічний інтерфейс користувача, реалізований у Python, а також проведено тестування роботи системи на реалістичних аграрних даних. Наведено результати моделювання, порівняльний аналіз сценаріїв та оцінено збіжність гібридної Fuzzy–GA моделі.

Ключові слова: аграрні проєкти, інтелектуальна система підтримки рішень, нечітка логіка, генетичний алгоритм, оптимізація, багатокритеріальне управління.

ЗМІСТ

ВСТУП

В сучасних умовах розвитку аграрного сектору зростає потреба у впровадженні інноваційних технологій управління проєктами, які спрямовані на підвищення ефективності використання ресурсів, мінімізацію ризиків та досягнення стійких результатів. Аграрні проєкти характеризуються високим рівнем невизначеності, оскільки їх успішність залежить від комплексу факторів – природно-кліматичних умов, економічної кон'юнктури, стану ринкової інфраструктури та управлінських рішень. Традиційні підходи до планування та оптимізації таких проєктів часто базуються на статистичних методах та експертних оцінках, що обмежує їхню здатність враховувати складні взаємозв'язки між параметрами і своєчасно адаптуватися до динамічного середовища [12].

Задача полягає в тому, що сучасні аграрні проєкти реалізуються у середовищі з підвищеним рівнем ризику та нестабільності. З одного боку, значний вплив мають кліматичні зміни, що зумовлюють непередбачуваність урожайності, зростання потреб у водних і енергетичних ресурсах, а також підвищення вразливості сільськогосподарських систем до екстремальних погодних явищ. З іншого боку, економічні ризики, пов'язані з коливаннями цін на аграрну продукцію, витратами на паливно-енергетичні ресурси та зміною умов державної підтримки, роблять процес управління надзвичайно складним. У таких умовах особливо важливим стає пошук методів, здатних враховувати множинність і суперечливість факторів, забезпечуючи більш об'єктивне та адаптивне прийняття рішень [15].

Відсутність комплексних інструментів, що поєднують методи оптимізації та враховують невизначеність, зумовлює необхідність використання інтелектуальних систем підтримки прийняття рішень. Такі системи мають враховувати як кількісні параметри (фінансові показники, ресурси, продуктивність), так і якісні критерії (стійкість до кліматичних ризиків, соціальний вплив, екологічна безпечність). Застосування нечіткої логіки

дозволяє описати нечіткі та лінгвістичні змінні, що відображають реальні управлінські ситуації, тоді як генетичні алгоритми забезпечують ефективний пошук оптимальних рішень у багатовимірних просторах параметрів. Саме інтеграція цих методів відкриває перспективи для створення систем, здатних підтримати ефективну реалізацію аграрних проєктів у складному і мінливому середовищі [11].

Використання інтелектуальної системи підтримки прийняття рішень на основі нечіткої логіки та генетичних алгоритмів у сфері аграрного проєктування сприяє комплексній оцінці впливу різних факторів на результативність, оптимізації розподілу фінансових та матеріальних ресурсів, а також зниженню суб'єктивності у процесі управлінських рішень. Розробка такої системи є актуальним напрямком наукових досліджень, який поєднує практичну значущість та інноваційний потенціал, адже вона здатна забезпечити більш справедливий і науково обґрунтований підхід до реалізації аграрних ініціатив.

У кваліфікаційній роботі передбачено створення інтелектуальної системи підтримки прийняття рішень для оптимізації складових аграрних проєктів із використанням методів нечіткої логіки та генетичних алгоритмів.

Об'єкт дослідження – процеси управління аграрними проєктами.

Предмет дослідження – методи та моделі застосування нечіткої логіки й генетичних алгоритмів для підтримки прийняття рішень в аграрних проєктах.

Засоби дослідження – інструменти MATLAB Fuzzy Logic Toolbox, бібліотеки Python для реалізації генетичних алгоритмів (DEAP, PyGAD), а також пакети Pandas та NumPy для обробки та аналізу даних.

Розділ 1.
СТАН РЕАЛІЗАЦІЇ АГРАРНИХ ПРОЄКТІВ ТА АНАЛІЗ ІСНУЮЧОГО ІНСТРУМЕНТАРІЮ

1.1. Аграрні проєкти як об’єкт дослідження в умовах невизначеності та ризиків

Управління аграрними проєктами становить особливий тип управлінських завдань. У інструментарії для їх розв’язання поєднуються виробничі, економічні та соціальні чинники проєктного середовища. Роботи щодо реалізації проєктів охоплюють повний цикл від підготовки ґрунтів та висіву культур до збирання врожаю, переробки та реалізації продукції. Їхня специфіка полягає у високій залежності від природних умов, значній кількості параметрів та взаємозв’язків між ними, а також у необхідності врахування екологічної стійкості та соціальної ефективності. Для управління аграрними проєктами важливо враховувати як кількісні показники, що можна виміряти (врожайність, витрати, тривалість робіт), так і якісні характеристики, що описуються в термінах ризику, невизначеності чи лінгвістичних оцінок.

Невизначеність у сільському господарстві має різні джерела (рис. 1.1).

Кліматичні чинники, зокрема опади, температура та екстремальні погодні явища, безпосередньо впливають на врожайність і строки виконання робіт. Економічні чинники проявляються у коливанні цін на продукцію, змінах вартості добрив, пального та енергоресурсів, що ускладнює прогнозування фінансових результатів.

Технологічні ризики пов’язані зі зношеністю обладнання, дефіцитом сучасних аграрних технологій або нестачею якісного посівного матеріалу. Додаткову невизначеність створюють політичні та регуляторні зміни, що впливають на доступ до ринків, дотації чи субсидії. Соціальні фактори, зокрема трудова міграція або конфлікти за ресурси, також можуть обмежувати стабільність реалізації проєктів.

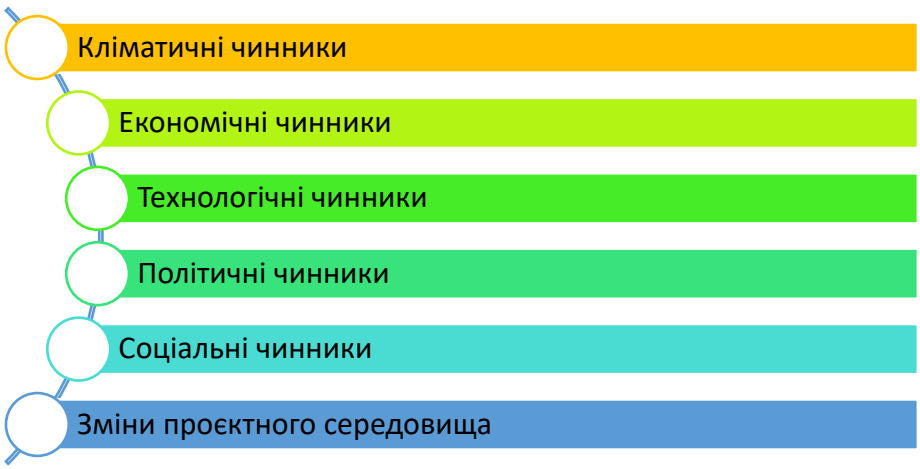

Рисунок 1.1 – Джерела невизначеності у сільському господарстві

Для кращого розуміння складності управління аграрними проєктами систематизовано ризики за категоріями. В таблиці 1.1 наведено узагальнену структуру основних ризиків, їхні джерела та можливі наслідки для проєктів.

Невизначеність можна формалізувати за допомогою математичних моделей. Одним із поширених підходів є використання нечітких чисел. Наприклад, витрати проєкту можна подати у вигляді трикутного нечіткого числа:

$$\hat{C} = (C_{\min}, C_{mode}, C_{\max}), \tag{1.1}$$

де $C_{\min}, C_{\max}$ – крайні значення відображають можливі межі витрат; C_{mode} – середнє, найбільш імовірний сценарій.

Подібним чином можна описати тривалість реалізації проєкту:

$$\tilde{T} = (T_{\min}, T_{mode}, T_{\max}), \tag{1.2}$$

Аналогічно якість отриманої продукції:

$$\tilde{Q} = (Q_{\min}, Q_{mode}, Q_{\max}). \tag{1.3}$$

Використання α-вирізів дозволяє отримати інтервали значень, що відповідають різним рівням довіри до прогнозу. Формально це записується так:

$$\tilde{C}^{\alpha} = [C_{\min} + \alpha(C_{mode} - C_{\min}), C_{\max} - \alpha(C_{\max} - C_{mode})]. \tag{1.4}$$

Цей підхід дає змогу поєднувати оптимістичні та песимістичні сценарії у процесі планування.

Таблиця 1.1 – Джерела невизначеності та ризиків аграрних проєктів

Категорія ризику	Приклади чинників	Потенційні наслідки
Природно-кліматичні	коливання температури та опадів, посухи, повені	зниження врожайності, збільшення витрат на адаптаційні заходи
Економічні	коливання ринкових цін, інфляція, зміна попиту	фінансові втрати, падіння прибутковості
Технологічні	відсутність сучасної техніки, низька якість насіння	зниження продуктивності, проблеми з якістю продукції
Регуляторні	зміни законодавства, екологічні вимоги	затримки в реалізації, додаткові витрати
Соціальні	нестабільність зайнятості, міграція робочої сили	дефіцит кадрів, соціальна напруга

На рисунку 1.2 подано схему чинників виникнення ризиків аграрних проєктів, яка відображає їхній поділ за основними групами. Вона наочно показує, що будь-яке рішення в аграрній сфері потребує врахування комплексу взаємопов’язаних чинників.

У науковій літературі все більше уваги приділяється створенню моделей, здатних об’єднати опис невизначеності та механізми оптимізації. Зокрема, у статті [12] наведено приклади застосування нечітких чисел для опису витрат, часу і якості сільськогосподарських проєктів у поєднанні з генетичними алгоритмами для пошуку оптимальних рішень. Авторами запропоновано аналіз різних сценаріїв з урахуванням рівнів α, що дозволяє будувати моделі з гнучким балансом між точністю та надійністю прогнозів. Кейс-дослідження продемонструвало можливість оптимізації параметрів для культур, таких як пшениця, рис, ячмінь і кукурудза, із використанням даних за більш ніж двадцятирічний період.

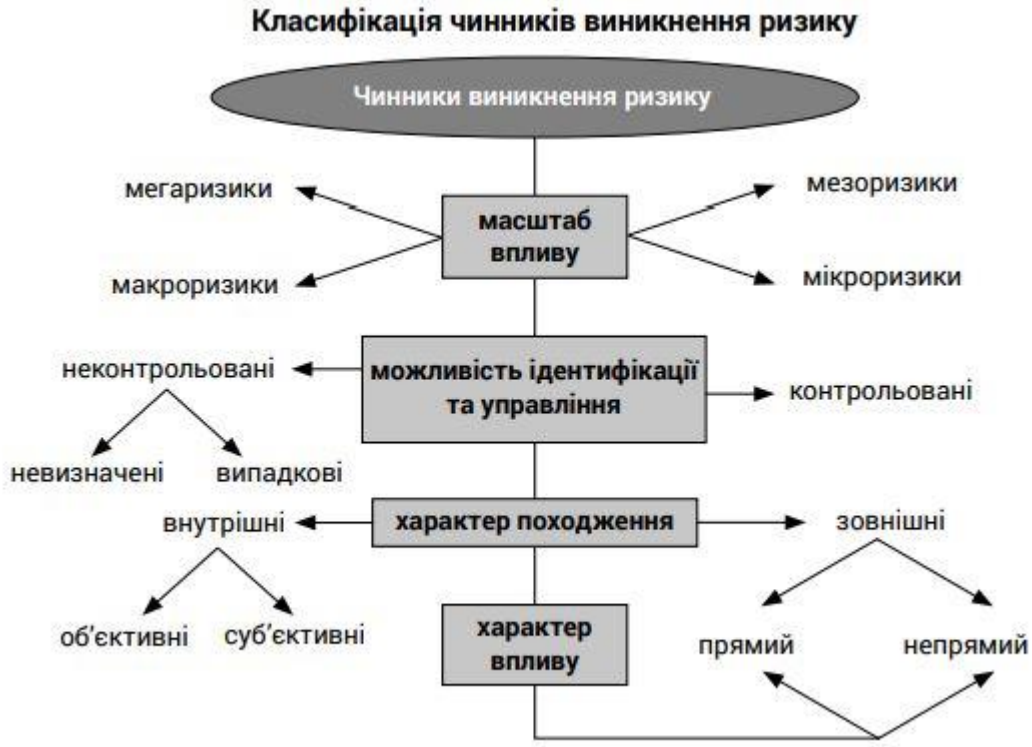

Рисунок 1.2 – Чинники виникнення ризиків аграрних проєктів [3]

Таким чином, аграрні проєкти розглядаються як складні системи, що поєднують у собі численні параметри та фактори ризику. Вони потребують застосування сучасних підходів, здатних врахувати багатовимірність і невизначеність середовища. Використання формалізованих моделей ризику на основі нечіткої логіки створює підґрунтя для подальшої оптимізації за допомогою еволюційних алгоритмів. Це відкриває перспективи для розробки інтелектуальних систем підтримки прийняття рішень, які дадуть змогу підвищити стійкість аграрного виробництва в умовах кліматичних, економічних і соціальних викликів.

1.2. Системи підтримки прийняття рішень у проєктному менеджменті

Системи підтримки прийняття рішень (СППР) є основним інструментом сучасного проєктного менеджменту, оскільки вони забезпечують керівників

інформаційною, аналітичною та прогностичною базою для формування оптимальних управлінських рішень. Виконання проєктів у складному та динамічному середовищі вимагає від менеджерів не лише врахування великої кількості параметрів, а й здатності адаптуватися до непередбачуваних змін, що обумовлює потребу в автоматизованих засобах обробки даних та моделювання сценаріїв.

СППР можна визначити як інтерактивну інформаційну систему, яка поєднує дані, моделі та інтерфейси для користувача, створюючи основу для аналітичної діяльності. У проєктному менеджменті такі системи використовуються для планування, оцінювання ризиків, розподілу ресурсів, моніторингу прогресу та контролю результатів. Важливою особливістю є їхня здатність інтегруватися з іншими інформаційними системами підприємства, зокрема ERP та CRM, що розширює їхні можливості у системному управлінні.

Залежно від логіки побудови, СППР у проєктному менеджменті поділяють на системи, орієнтовані на дані, на моделі та на знання. Системи, орієнтовані на дані, забезпечують доступ до великих обсягів інформації та її аналітичну обробку. Орієнтована на моделі СППР потребують створення математичних та імітаційних моделей для прогнозування результатів проєктів за різних сценаріїв.

Системи, орієнтовані на знання, ґрунтуються на експертних правилах і методах штучного інтелекту, дозволяючи обробляти нечітку або неповну інформацію.

У таблиці 1.2 подано основні класифікаційні ознаки СППР для управління проєктами.

Розвиток сучасних СППР у проєктному менеджменті тісно пов’язаний із поширенням технологій обчислювального інтелекту. Застосування машинного навчання, нейронних мереж, генетичних алгоритмів та нечіткої логіки дозволяє суттєво підвищити точність прогнозів і забезпечити багатокритеріальну оптимізацію. Такий підхід особливо важливий для аграрних проєктів, де взаємодіють численні параметри – кліматичні, економічні та технологічні, а невизначеність залишається домінуючим фактором.

Таблиця 1.2 – Типи СППР використовуваних у проєктному менеджменті

Ознака класифікації	Характеристика	Приклади використання
Орієнтація на дані	робота з великими масивами структурованої та неструктурованої інформації	бази даних проєктів, аналітика витрат
Орієнтація на моделі	побудова математичних і статистичних моделей для прогнозів і сценаріїв	моделювання термінів виконання робіт, симуляції ризиків
Орієнтація на знання	застосування експертних систем і методів ШІ	нечіткі правила, системи на основі кейсів
Інтегровані системи	поєднання кількох підходів для комплексного управління	сучасні гібридні СППР із машинним навчанням

На рисунку 1.2 представлено узагальнену структуру СППР у проєктному менеджменті. Вона складається з трьох рівнів:

- рівня даних, де зберігається інформація про ресурси, календарні графіки та витрати;
- рівня моделей, який включає оптимізаційні та імітаційні інструменти;
- рівня інтерфейсу користувача, що забезпечує інтерактивну взаємодію з системою.

Важливо зазначити, що сучасні СППР орієнтовані не лише на аналітичну підтримку, а й на формування рекомендацій для менеджерів проєктів.

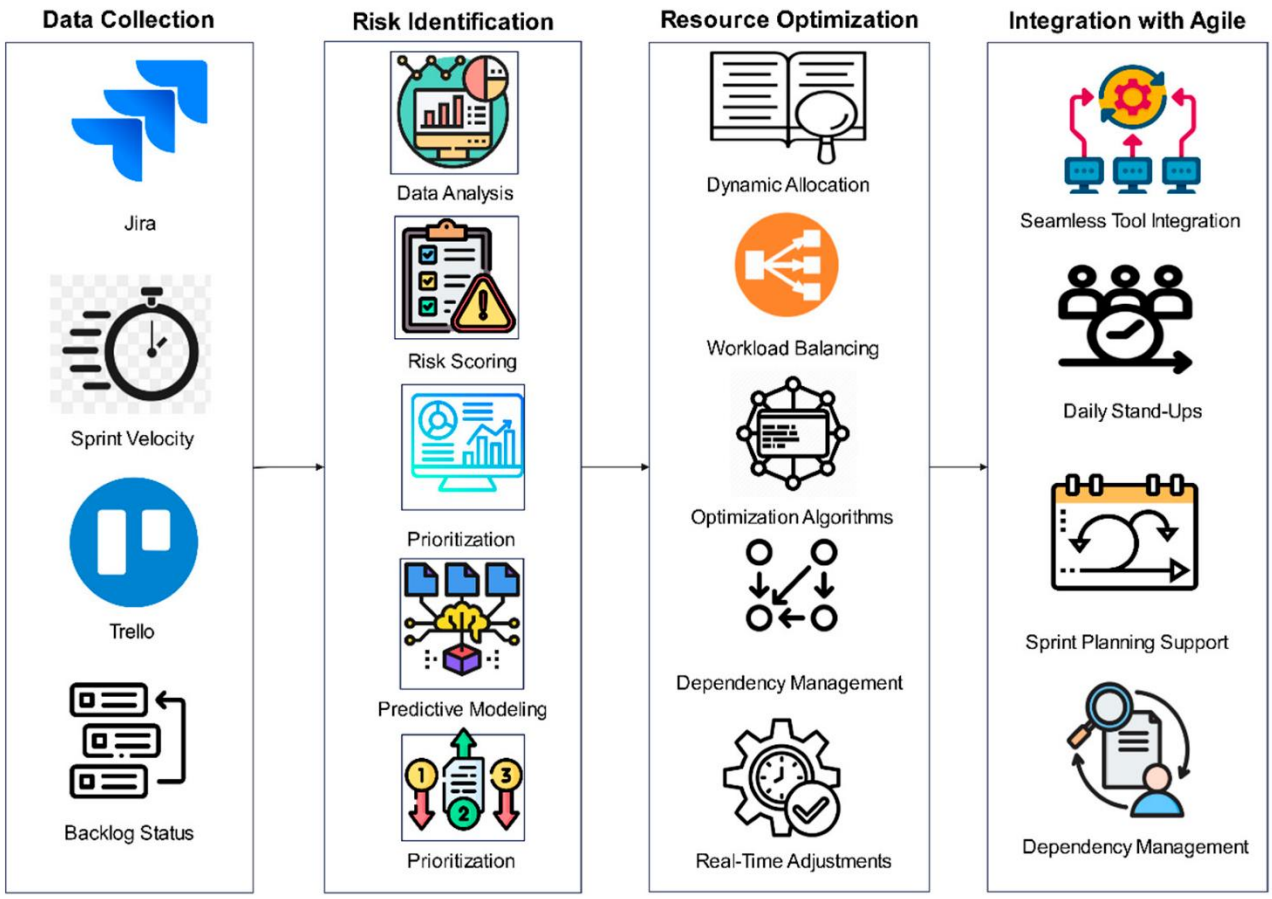

Рисунок 1.2 – Структурна модель системи підтримки прийняття рішень у проєктному менеджменті [13]

Запропонована структура у роботі [13] залежить від складних інструментів з кількома джерелами даних та чіткими параметрами оцінки для успішного впровадження ASPM. Кожен елемент цієї структури допомагає командам приймати рішення на основі даних, мінімізуючи ризики та ефективно використовуючи ресурси для Agile-проектів.

Фреймворк об’єднує відомі Agile-платформи та інструменти для збору та аналізу поточних потоків даних. Agile-фреймворк використовує Jira (версія 9.4.11) з Trello (версія 2024.2.1) та Azure DevOps (версія 2024.1.0) для збору метрик спринту, які поєднують статуси відкладень з моніторингом прогресу виконання завдань. Під час аналізу та візуалізації даних фреймворк спирається на бібліотеки Python та фреймворки машинного навчання, такі як Pandas 3.9, NumPy (версія 1.24.2), Matplotlib (версія 3.6.3), TensorFlow (версія 2.11.0) та Scikit-learn (версія 1.2.1), для створення прогнозних моделей та відображення

результатів. Крім того, Power BI (версія 2.124.2024.0) та користувацькі інформаційні панелі відображають дані в реальному часі разом із відстеженням проектів, генеруючи прості аналітичні дані, які Agile-команди можуть легко перетворити на дії.

Використання таких систем у практиці управління проєктами дозволяє не лише підвищити точність прогнозів і зменшити ризики, але й створює можливості для багатокритеріальної оцінки ефективності. Це особливо важливо для аграрних проєктів, де якість прийнятих рішень визначає стійкість розвитку виробництва та рівень його економічної ефективності. Таким чином, СППР виступають важливою ланкою між традиційними методами управління та сучасними інтелектуальними технологіями, що формують новий рівень організації проєктної діяльності.

1.3. Використання методів нечіткої логіки в управлінні аграрними проєктами

Управління аграрними проєктами характеризується високим рівнем невизначеності, який виникає внаслідок складної взаємодії кліматичних, економічних, технологічних і соціальних факторів. Традиційні методи планування, засновані на статистичних даних, часто виявляються недостатніми, оскільки вони не враховують неповноту та нечіткість інформації, яка надходить до менеджера проєкту. Саме тому все більшого поширення набуває використання методів нечіткої логіки, здатних формалізувати знання експертів та поєднувати кількісні та якісні параметри в єдиній моделі.

Основою нечіткої логіки є поняття нечіткої множини, де кожному елементу зіставляється ступінь належності, що змінюється у діапазоні від 0 до 1. Це дозволяє відобразити неповну визначеність понять, які важко описати класичними методами. Наприклад, показник урожайності можна описати не лише як «високий» чи «низький», а як проміжний рівень із певною мірою

впевненості. Формально нечітку множину A на універсальній множині X задають функцією належності $\mu_A(x): X \to [0,1]$, яка відображає ступінь істинності твердження «елемент x належить множині A«.

У таблиці 1.3 наведено приклад використання нечітких змінних для оцінки параметрів аграрного проєкту, де для кожного критерію визначено діапазони можливих значень та відповідні лінгвістичні змінні.

Таблиця 1.3 – Нечіткі змінні для оцінювання параметрів аграрного проєкту

Критерій	Діапазон значень	Лінгвістичні змінні
Урожайність, ц/га	10–80	низька, середня, висока
Витрати, тис. грн	50–250	малі, помірні, значні
Якість продукції	0–100%	незадовільна, прийнятна, висока
Тривалість, дні	30–120	коротка, середня, довга

Реалізація нечітких моделей у проєктному менеджменті зазвичай базується на системі нечіткого виведення (Fuzzy Inference System). Найбільш поширеним підходом є модель Мамдані, яка складається з етапів фазифікації, формування бази правил, агрегування умов, дефазифікації та отримання остаточного результату. Система правил має вигляд «Якщо …, то …».

Наприклад, у задачі оцінювання ефективності аграрного проєкту правило може мати такий вигляд: «Якщо витрати низькі і урожайність висока, то ефективність проєкту дуже добра». Така система дозволяє інтегрувати знання експертів і переводити якісні судження у кількісну площину.

На рисунку 1.3 зображено спрощену схему нечіткої системи підтримки прийняття рішень, яка складається з блоку фазифікації вхідних даних, бази правил та модуля дефазифікації для отримання конкретного результату.

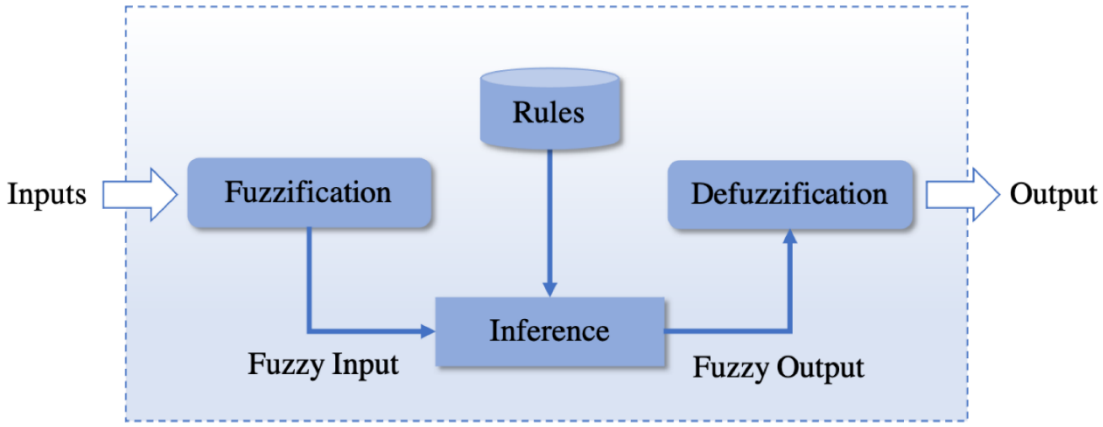

Рисунок 1.3 – Узагальнена структура нечіткої системи підтримки прийняття рішень [22]

У роботі [22] розглядається система підтримки прийняття рішень (СППР), яка має на меті сприяти досягненню вищезазначених загальних цілей, спрямованих на прогнозування зміни фізико-хімічних параметрів (твердості, вмісту розчинних твердих речовин та кислотності), зокрема персиків, які безпосередньо пов’язані з їхньою якістю, та оцінку періоду їх комерціалізації за найвищою загальною якістю, сприйнятою споживачами, тим самим уникаючи відходів. Це формулювання можна поширити на виробництво різних сільськогосподарських продуктів. Оскільки якість, сприйнята споживачами, є відносним параметром, пропонується СППР на основі штучного інтелекту (ШІ) за допомогою моделі нечіткої логіки.

Система нечіткої логіки (FLS) була вперше описана в 1965 році Заде [31] і використовувалася в додатках підрахунку завдяки своїм численним перевагам. Окрім того, що FLS є потужною інтелектуальною технікою, яку легко реалізувати та яка має низькі обчислювальні витрати, вона може забезпечити дуже ефективний механізм прийняття рішень на основі неточних вхідних даних, оскільки вона має продуктивність, еквівалентну тому, як людина приймає рішення, за винятком того, що вона робить це швидше [24; 15; 32].

Математично процес фазифікації ґрунтується на визначенні функцій належності. Для показника урожайності трикутна функція може мати вигляд:

$$\mu(x) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b} \right), 0 \right), \qquad (1.5)$$

де a, b, c – параметри трикутної функції належності.

Ця функція (1.5) дозволяє відобразити поступовий перехід між лінгвістичними змінними «низька», «середня» і «висока урожайність».

Приклад практичного застосування нечіткої логіки у сільському господарстві описано в дослідженнях, присвячених оцінюванню ефективності зрошення та оптимізації використання добрив. Системи нечіткого виведення дозволяли враховувати як кількісні параметри, так і експертні оцінки фахівців, що забезпечувало підвищену стійкість прийнятих рішень. Зокрема, у працях останніх років [18] доведено, що застосування нечітких моделей у поєднанні з методами оптимізації на основі еволюційних алгоритмів дає змогу досягати балансу між витратами, врожайністю та екологічною безпечністю.

Таким чином, методи нечіткої логіки в управлінні аграрними проєктами створюють можливості для формалізації суб’єктивних експертних знань та врахування невизначеності. Вони дозволяють забезпечити гнучкість управлінських рішень і підвищити обґрунтованість вибору альтернатив у ситуаціях, де традиційні методи статистичного аналізу не дають задовільних результатів. Використання таких підходів стає особливо актуальним у сучасних умовах зростання кліматичних та економічних ризиків, що безпосередньо впливають на ефективність аграрного виробництва.

1.4. Генетичні алгоритми як інструмент глобальної оптимізації

Генетичні алгоритми належать до класу еволюційних обчислювальних методів, які імітують процес природного добору для пошуку оптимальних рішень у складних багатовимірних просторах. Їхня особливість полягає у здатності знаходити глобальні, а не локальні оптимуми, що робить їх незамінними у задачах із великою кількістю змінних, взаємозалежних

параметрів та невизначених обмежень. У проєктному менеджменті, зокрема в аграрній сфері, застосування генетичних алгоритмів дозволяє вирішувати задачі оптимального розподілу ресурсів, планування строків робіт, оцінювання комбінацій стратегій та забезпечення стійкого розвитку.

Принцип роботи генетичного алгоритму базується на використанні популяції можливих рішень, які представляються у вигляді хромосом. Кожна хромосома кодує потенційне рішення задачі, а якість цього рішення оцінюється за допомогою цільової функції. Далі застосовуються оператори відбору, кросинговеру та мутації, які формують нові покоління рішень. З покоління у покоління відбувається еволюція, що приводить до поступового покращення значень цільової функції та пошуку оптимальних варіантів.

Таблиця 1.4 – Основні етапи генетичного алгоритму

Етап	Характеристика процесу
Ініціалізація	Формування початкової популяції рішень випадковим або напіввипадковим чином
Оцінювання	Обчислення значень цільової функції для кожної хромосоми
Відбір	Вибір найкращих особин для створення нового покоління
Кросинговер	Обмін фрагментами хромосом для формування нових рішень
Мутація	Внесення випадкових змін у хромосоми для збереження різноманітності
Завершення	Зупинка алгоритму після досягнення критерію оптимальності або кількості поколінь

У таблиці 1.4 наведено основні етапи роботи генетичного алгоритму, які відображають його еволюційну природу.

У загальному вигляді задача оптимізації для аграрних проєктів може бути сформульована так. Нехай існує множина рішень X, кожне з яких характеризується вектором параметрів $p=(x_1, x_2, ..., x_n)$. Мета полягає у мінімізації витрат $C(p)$, тривалості реалізації $T(p)$ та одночасній максимізації

якості $Q(p)$. Сукупна цільова функція може бути подана у вигляді багатокритеріальної:

$$F(p) = w_1 \cdot \frac{1}{C(p)} + w_2 \cdot \frac{1}{T(p)} + w_3 \cdot Q(p), \qquad (1.6)$$

де w_1, w_2, w_3 – вагові коефіцієнти, що відображають пріоритетність критеріїв для конкретного проєкту.

Такий підхід дозволяє налаштовувати алгоритм залежно від того, чи важливішим є зниження витрат, прискорення реалізації чи підвищення якості продукції.

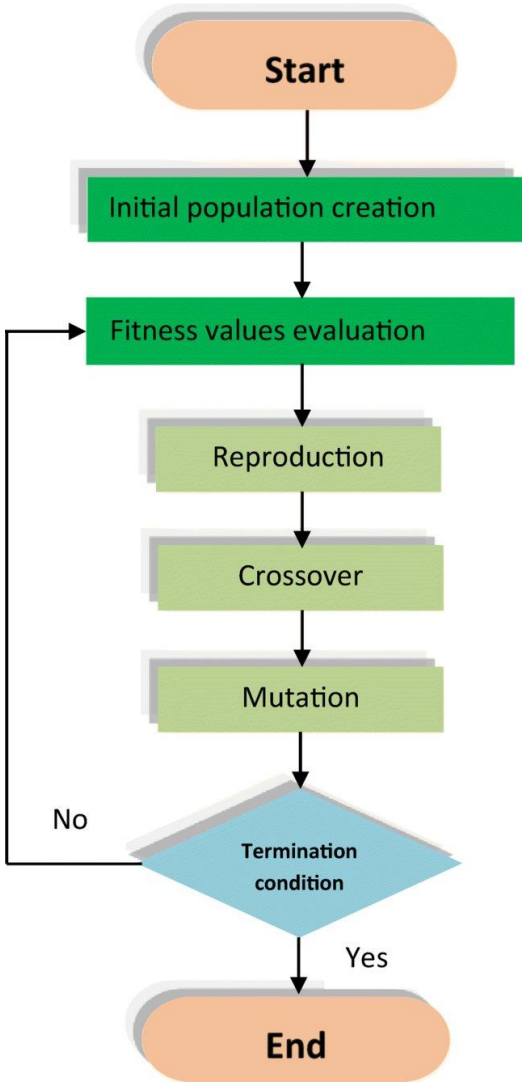

Рисунок 1.4 – Алгоритм реалізації генетичного алгоритму [21]

На рисунку 1.4 наведено узагальнену схему роботи генетичного алгоритму, яка ілюструє циклічний характер процесу пошуку оптимального рішення.

Приклади використання генетичних алгоритмів в аграрному менеджменті охоплюють оптимізацію планів сівозміни, вибір комбінацій агротехнічних заходів, прогнозування витрат на добрива та пестициди, а також моделювання балансів між економічними й екологічними результатами. В останніх публікаціях [16; 19] показано, що поєднання генетичних алгоритмів із методами нечіткої логіки дозволяє мінімізувати базу правил, зберігаючи при цьому високу точність прогнозування та оптимізації. Це особливо цінно для задач багатокритеріального характеру, де необхідно одночасно враховувати різні виміри ефективності аграрних проєктів.

Таким чином, генетичні алгоритми є ефективним інструментом глобальної оптимізації у складних проєктних середовищах. Їхнє застосування забезпечує адаптивність, здатність працювати з неповною та нечіткою інформацією і можливість інтеграції з іншими інтелектуальними методами. Це створює основу для побудови сучасних систем підтримки прийняття рішень, які відповідають викликам аграрної галузі та сприяють стійкому розвитку.

1.5. Інтеграція нечіткої логіки та генетичних алгоритмів для оптимізації аграрних проєктів

Сучасні аграрні проєкти реалізуються у середовищі, що характеризується високою невизначеністю та динамічними змінами зовнішніх умов. Для врахування нечітких, неповних і якісних даних ефективними є методи нечіткої логіки, які дозволяють формалізувати експертні знання та відобразити багатокритеріальний характер управління. Разом із тим, нечіткі системи мають обмеження, зокрема надмірну складність бази правил у випадку великої кількості змінних. Це призводить до ускладнення процесу виведення рішень і

збільшення обчислювальних витрат. Генетичні алгоритми, своєю чергою, добре зарекомендували себе у завданнях глобальної оптимізації, однак потребують механізмів оцінювання рішень на основі нечітких критеріїв. Саме тому інтеграція обох підходів дає змогу поєднати їхні сильні сторони й компенсувати недоліки.

Сутність інтегрованого підходу полягає в тому, що нечітка логіка відповідає за опис параметрів і критеріїв у лінгвістичній формі та побудову бази правил, тоді як генетичний алгоритм здійснює оптимізацію структури цієї бази та параметрів функцій належності. Наприклад, у задачі планування витрат, тривалості та якості аграрного проєкту нечітка система задає залежності між вхідними змінними («низькі витрати», «середня тривалість», «висока якість»), а генетичний алгоритм підбирає найкращу комбінацію правил, яка максимізує цільову функцію ефективності.

У таблиці 1.5 наведено узагальнену схему ролі кожного з методів в інтегрованій системі оптимізації аграрних проєктів.

Таблиця 1.5 – Функції нечіткої логіки та генетичних алгоритмів у комплексній моделі

Компонент	Роль нечіткої логіки	Роль генетичного алгоритму
Опис змінних	Формування нечітких множин і функцій належності	Пошук оптимальних параметрів функцій
База правил	Побудова експертних правил типу «якщо-то»	Мінімізація бази правил, усунення надлишкових
Виведення	Інтеграція знань для отримання оцінки ефективності	Оптимізація послідовності і структури правил
Результат	Отримання інтегральної оцінки проекту	Вибір глобально оптимального рішення

Формально інтегрована модель може бути описана як багатокритеріальна оптимізація з нечіткими обмеженнями. Нехай параметри аграрного проєкту подані у вигляді нечітких змінних $\tilde{C},\tilde{T},\tilde{Q}$, що відображають витрати, тривалість

і якість відповідно. Генетичний алгоритм формує популяцію рішень $P = \{p_1, p_2, \ldots, p_n\}$, кожне з яких оцінюється функцією пристосованості:

$$F(p_i) = w_1 \cdot \frac{1}{\tilde{C}(p_i)} + w_2 \cdot \frac{1}{\tilde{T}(p_i)} + w_3 \cdot \tilde{Q}(p_i), \qquad (1.7)$$

де w_1, w_2, w_3 – вагові коефіцієнти, які задають пріоритетність критеріїв.

Нечіткі функції $\tilde{C}, \tilde{T}, \tilde{Q}$ визначаються за допомогою системи Мамдані, а їхні параметри уточнюються генетичним алгоритмом.

На рисунку 1.5 подано структурну схему інтегрованої системи, яка складається з блоку фазифікації даних, бази правил, модуля генетичної оптимізації та дефазифікації результатів.

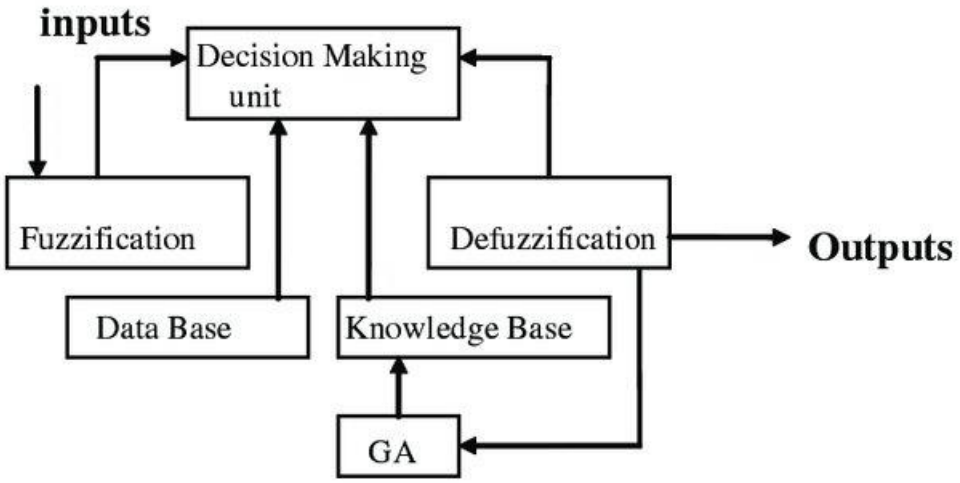

Рисунок 1.5 – Узагальнена схема інтеграції нечіткої логіки та генетичного алгоритму [14]

У сучасних дослідженнях [17; 20; 23] доведено, що такий інтегрований підхід дозволяє значно зменшити кількість правил, які необхідні для функціонування системи, при цьому точність оцінювання ефективності проєктів не тільки не знижується, а навіть підвищується. Особливо це проявляється у випадках, коли проєкти мають велику кількість параметрів, що важко формалізувати звичайними методами.

Таким чином, інтеграція нечіткої логіки та генетичних алгоритмів формує потужний інструмент для управління аграрними проєктами в умовах невизначеності. Вона дозволяє об’єднати інтелектуальні можливості експертних систем і глобальний пошук оптимуму, забезпечуючи баланс між витратами, якістю та екологічною стійкістю. Такий підхід створює методологічне підґрунтя для розробки сучасних систем підтримки прийняття рішень, які здатні ефективно функціонувати у складному та динамічному середовищі аграрного виробництва.

1.6. Завдання кваліфікаційної роботи

Метою кваліфікаційної роботи є розробка інтелектуальної системи підтримки прийняття рішень для оптимізації складових аграрних проєктів на основі нечіткої логіки та генетичних алгоритмів. Для досягнення цієї мети необхідно виконати такі конкретні завдання:

1. Провести аналіз літератури з методів нечіткої логіки та генетичних алгоритмів, які застосовуються в аграрних проєктах.
2. Здійснити моделювання основних змінних аграрного проєкту (витрати, час, якість) як нечітких змінних із відповідними функціями належності, а також розробити базу правил нечіткої логіки на основі експертних знань.
3. Визначити структуру генетичного алгоритму для оптимізації параметрів нечіткої системи. Здійснити чутливий аналіз впливу параметрів на результати.
4. Побудувати комбіновану модель нечіткої логіки та генетичного алгоритму.
5. Провести емпіричне дослідження на даних аграрного проєкту чи проєктів (можливо вибрати культури, регіони, ресурси), щоб протестувати побудовану модель.
6. Порівняти результати роботи моделі при різних сценаріях невизначеності.

7. Оцінити ефективність розробленої системи підтримки прийняття рішень щодо традиційних підходів.

8. Розробити прототип програмного інструменту, що дозволяє задавати вхідні дані.

9. Сформулювати рекомендації щодо використання такої системи в аграрній практиці, з урахуванням екологічних, економічних та кліматичних ризиків, а також потенційних обмежень реалізації.

Узагальнюючи викладене, варто зазначити, що поставлені завдання охоплюють повний цикл розробки інтелектуальної системи підтримки прийняття рішень для оптимізації аграрних проєктів – від теоретичного аналізу методів нечіткої логіки та генетичних алгоритмів до побудови комбінованої моделі та її практичної перевірки на реальних даних. Такий підхід забезпечує не лише новизну дослідження, а й створює практичну цінність для аграрного сектору, оскільки дозволяє знизити невизначеність управлінських рішень, досягти балансу між витратами, тривалістю та якістю результатів і сформувати рекомендації, що сприяють підвищенню стійкості та ефективності аграрного виробництва в умовах сучасних викликів.

РОЗДІЛ 2.
ВИБІР ІНСТРУМЕНТАРІЮ ТА ПРОЕКТУВАННЯ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ДЛЯ ОПТИМІЗАЦІЇ СКЛАДОВИХ АГРАРНИХ ПРОЄКТІВ

2.1. Обґрунтування вибору програмних засобів

Ефективність побудови інтелектуальної системи підтримки прийняття рішень значною мірою залежить від правильності вибору інструментів для моделювання нечітких систем та реалізації генетичних алгоритмів. У контексті оптимізації аграрних проєктів необхідно забезпечити можливість роботи з багатовимірними даними, побудови гібридних моделей та інтеграції різних алгоритмічних компонентів. З огляду на це, обрано Python як основну мову реалізації, а MATLAB використано як інструмент для побудови та тестування прототипів нечітких моделей. Такий комбінований підхід дозволяє зберегти гнучкість і масштабованість, що є критично важливим при розв’язанні оптимізаційних задач у сільському господарстві.

Під час вибору технічного інструментарію враховувалася необхідність реалізації алгоритмів нечіткої логіки та генетичної оптимізації, а також перспективність майбутньої інтеграції системи в якості DSS-платформи. Python забезпечує широкий спектр бібліотек для наукових обчислень, зокрема NumPy, Pandas, Matplotlib та спеціалізовані пакети для реалізації еволюційних алгоритмів. Крім того, Python є кросплатформеним інструментом з відкритим доступом, що значно спрощує дослідницьку роботу та подальше розгортання системи.

На рисунку 2.1 наведено загальну структуру стеку програмних засобів, що застосовуються при реалізації системи.

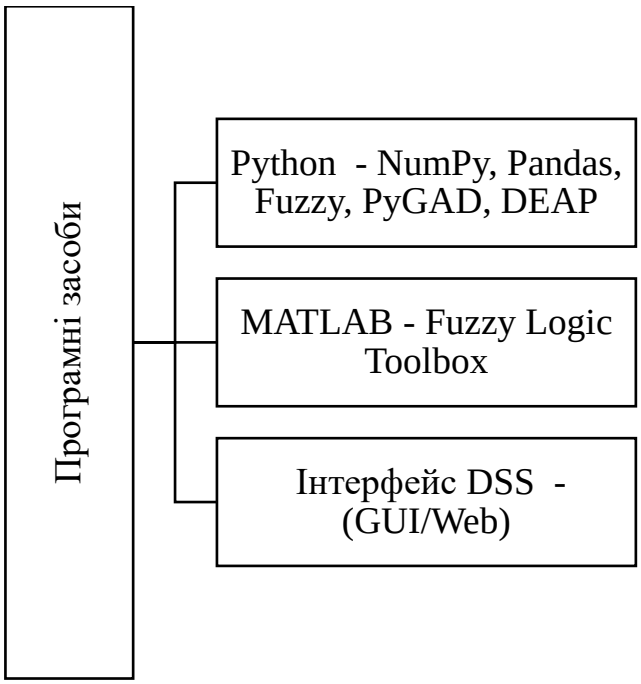

Рисунок 2.1 – Узагальнена схема вибору програмних засобів

MATLAB використано для початкової побудови та тестування нечіткої системи, оскільки Fuzzy Logic Toolbox забезпечує інструментарій для швидкої верифікації правил, побудови функцій належності та аналізу результатів. Python застосовано для інтеграції нечіткої системи з генетичним алгоритмом, а також для подальшої реалізації у вигляді DSS-платформи.

У таблиці 2.1 подано порівняння MATLAB і Python для вирішення задач нечіткого моделювання.

Таблиця 2.1 – Порівняння MATLAB та Python для реалізації нечітких моделей

Критерій	MATLAB Fuzzy Toolbox	Python (skfuzzy, numpy)
Гнучкість моделей	Висока, готові блоки FIS	Висока, можливість ручної побудови
Швидкість прототипування	Дуже висока	Середня
Інтеграція з AI	Обмежено	Повна інтеграція з ML/GA бібліотеками
Комерційність	Платна ліцензія	Відкритий код

Для опису нечітких змінних використано трикутні функції належності, що задаються формулою:

$$\mu(x) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b} \right), 0 \right), \qquad (2.1)$$

де a, b, c – параметри функції належності, що задають межі значень та центр нечіткої множини.

Для реалізації еволюційної оптимізації застосовано бібліотеки DEAP та PyGAD, оскільки вони забезпечують можливість налаштування операторів селекції, кросинговеру та мутації. MATLAB GA Toolbox розглядався як еталон для контролю коректності роботи Python-реалізації.

На рисунку 2.2 наведено концептуальну схему реалізації GA-компоненти.

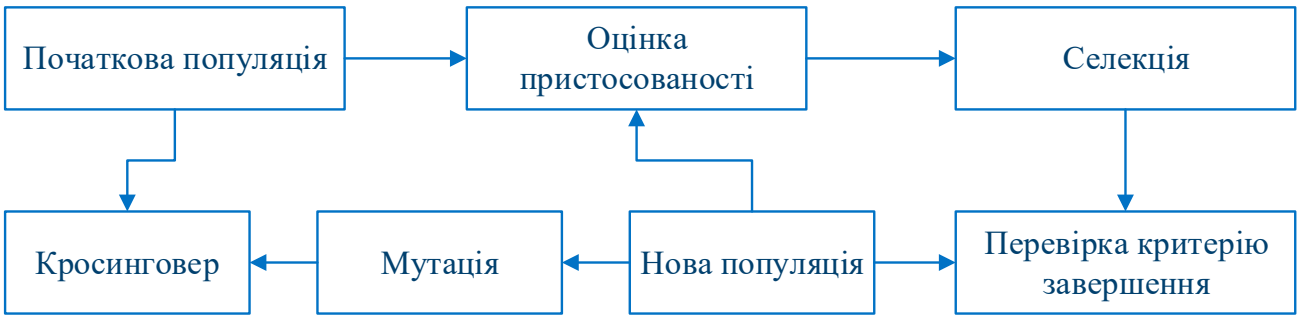

Рисунок 2.2 – Алгоритмічна схема генетичного оптимізатора

Фітнес-функцію для багатокритеріальної оптимізації визначено як:

$$F(p) = w_1 \cdot \frac{1}{C(p)} + w_2 \cdot \frac{1}{T(p)} + w_3 \cdot Q(p), \qquad (2.2)$$

де $C(p)$– витрати; $T(p)$ – тривалість; $Q(p)$ – якість отриманого результату; w_1, w_2, w_3 – вагові коефіцієнти пріоритетів.

Таким чином, обраний інструментарій MATLAB + Python забезпечує ефективне поєднання можливостей прототипування нечітких моделей та масштабованості при реалізації генетичних алгоритмів. Це дозволяє створити гнучку, адаптивну та розширювану платформу для підтримки прийняття рішень у аграрних проєктах.

2.2. Архітектура інтелектуальної системи прийняття рішень

Архітектура інтелектуальної системи підтримки прийняття рішень для оптимізації аграрних проєктів побудована таким чином, щоб забезпечити взаємодію методів нечіткої логіки та генетичних алгоритмів у єдиному обчислювальному середовищі. Основна функціональна ідея полягає у використанні нечіткої моделі для оцінювання параметрів системи та генетичного алгоритму для знаходження залежностей між ними, забезпечуючи адаптивний пошук оптимальних рішень. Архітектура формує цілісний інтелектуальний контур, де знання, представлені у вигляді нечітких правил, поєднані з еволюційним механізмом глобального пошуку.

Загальна структура системи складається з окремих функціональних модулів, об'єднаних через єдину логіку обміну даними. У центрі архітектури знаходиться база знань і блок оптимізації, які спільно формують ядро системи. На високому рівні логіка системи передбачає збір вхідних параметрів, їхню нечітку оцінку, оптимізацію на основі генетичного алгоритму та формування рекомендацій.

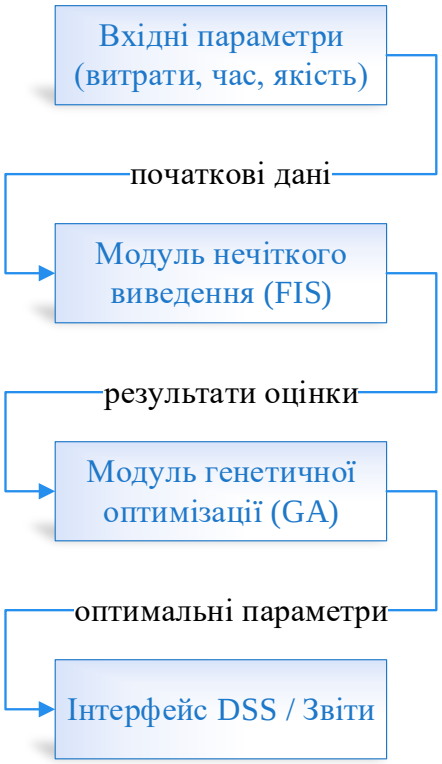

Рисунок 2.3 – Загальна структурна модель інтелектуальної системи

Модуль нечіткого виведення забезпечує перетворення вхідних параметрів, які мають лінгвістичну природу, у кількісні оцінки ефективності. Це дозволяє врахувати невизначеність та експертні знання. Нечітка система описується множиною функцій належності та правил формату:

Якщо C низькі і T короткий, тоді якість Q висока., (2.3)

де C – витрати; T – тривалість.

Для моделювання використано трикутні функції належності:

$$\mu_A(x) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b} \right), 0 \right). \quad (2.4)$$

Вони забезпечують плавний перехід між термами «низький», «середній», «високий». Система Мамдані обрана для здійснення механізму нечіткого виведення, оскільки вона є найбільш інтерпретованою для практичних задач.

Модуль генетичної оптимізації реалізує пошук найкращої комбінації параметрів, що впливають на результат. Основною ціллю алгоритму є мінімізація витрат і тривалості реалізації аграрного проєкту при одночасному максимальному забезпеченні якості. Функція пристосованості описується наступним чином:

$$F(p) = w_1 \cdot \frac{1}{C(p)} + w_2 \cdot \frac{1}{T(p)} + w_3 \cdot Q(p). \quad (2.5)$$

де w_1, w_2, w_3 – вагові коефіцієнти пріоритетності.

Оператори вибору, кросинговеру та мутації дозволяють підтримувати генетичне різноманіття та забезпечують глобальний характер пошуку. Модуль генетичної оптимізації взаємодіє безпосередньо з нечітким ядром, коригуючи правила та функції належності.

Підсистема інтерфейсу забезпечує подання вхідних параметрів користувачем, а також візуалізацію результатів оптимізації. Взаємодія реалізується через API-взаємозв’язки, що дозволяє в майбутньому розгорнути DSS у веб- або десктоп-системі. Архітектура підтримує як автономний режим роботи, так і інтеграцію з зовнішніми базами даних, наприклад агрометеорологічними сервісами.

Таблиця 2.2 – Взаємозв’язок програмних компонентів системи

Компонент	Функція	Взаємодія
Нечітке ядро (FIS)	Оцінювання параметрів	Отримує входи, передає F-значення
Генетичний оптимізатор	Пошук оптимальних правил	Використовує FIS-вихід як фітнес-значення
Інтерфейс DSS	Ввід/вивід даних	Комунікація з користувачем
База знань	Зберігання правил	Оновлюється GA-модулем

Таким чином, розроблена архітектура поєднує логічні й еволюційні методи в єдиній моделі, забезпечуючи адаптивність, здатність працювати з невизначеністю та можливість подальшої масштабованості. Структура системи гарантує всебічний аналіз аграрних параметрів і формування обґрунтованих управлінських рішень.

2.3. Формалізація параметрів аграрних проєктів

Побудова інтелектуальної системи підтримки прийняття рішень вимагає формалізації основних параметрів аграрних проєктів, що мають як кількісну, так і якісну природу. Основою нечіткої моделі є можливість представлення нечітких понять, наприклад «високі витрати», «коротка тривалість», «висока якість продукції». При цьому у процесі моделювання необхідно врахувати, що параметри аграрних систем суттєво варіюють залежно від умов середовища, виду продукції, технологій виробництва та масштабу господарства. Формалізація включає визначення вхідних змінних, їхніх функцій належності та структури бази правил, що задають логіку нечіткого виведення.

Основними вхідними змінними у системі є витрати реалізації проєкту (C), тривалість виконання (T), якість кінцевого результату (Q), а також ресурсні характеристики, що можуть включати рівень забезпеченості добривами,

водними ресурсами, технікою чи робочою силою (R). Оскільки всі ці параметри є багатовимірними та можуть містити невизначеність, вони описуються нечіткими множинами.

У таблиці 2.3 наведено приклад формалізації основних змінних.

Таблиця 2.3 – Характеристики вхідних змінних системи

Параметр	Змістове трактування	Діапазон значень	Лінгвістичні значення
Витрати C	Фінансові витрати проєкту	0 – 100% бюджету	Низькі, середні, високі
Тривалість T	Тривалість реалізації	0 – 200 днів	Коротка, середня, довга
Якість Q	Рівень кінцевої якості	0 – 1 (норм.)	Низька, середня, висока
Ресурси R	Ресурсна забезпеченість	0 – 1 (норм.)	Недостатні, оптимальні, надлишкові

Загальна залежність функції ефективності оцінюється відповідно до моделі:

$$E = f\left(C,T,Q,R\right). \qquad (2.6)$$

де E – інтегральна оцінка ефективності аграрного проєкту.

Функції належності визначають, наскільки конкретні значення параметрів належать до певних лінгвістичних категорій. Для моделі обрано трикутні та трапецоїдні функції належності, оскільки вони забезпечують високу інтерпретованість і простоту реалізації.

Трикутна функція належності подається формулою:

$$\mu(x) = \max\left(\min\left(\frac{x-a}{b-a}, \frac{c-x}{c-b} \right), 0 \right). \qquad (2.7)$$

де a,b,c – задають координати точки початку, вершини та завершення.

Трапецоїдна функція описується як:

$$\mu(x)=\begin{cases}0, & x\le a,\\ \dfrac{x-a}{b-a}, & a<x\le b,\\ 1, & b<x<c\\ \dfrac{d-x}{d-c}, & c\le x<d,\\ 0, & x\ge d.\end{cases} \quad (2.8)$$

На рисунку 2.4 наведено приклад трикутних функцій належності для параметра витрат.

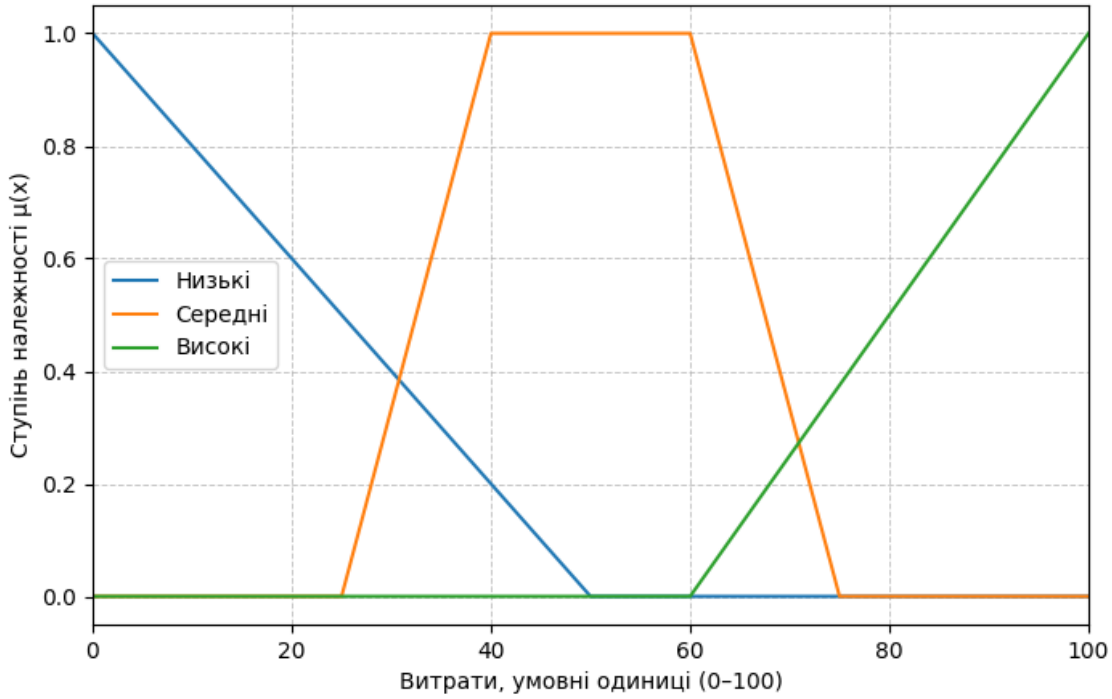

Рисунок 2.4 – Функції належності для рівнів витрат (з трапецоїдною середньою)

Подібним чином будуються функції належності для тривалості, ресурсу та якості.

Наступним кроком є формування бази нечітких правил Мамдані. База правил нечіткого виведення визначає логіку системи та формує експертний механізм прийняття рішень. Використовується модель Мамдані, оскільки вона дозволяє інтерпретувати правила у лінгвістичному вигляді. Типове правило, якщо витрати високі і тривалість коротка, тоді якість середня і має такий вигляд:

$$R_i: \text{Якщо } C = A_j \wedge T = B_k, \text{ то } Q = D_m. \quad (2.9)$$

У таблиці 2.4 наведено приклад фрагмента бази правил.

Таблиця 2.4 – Фрагмент бази нечітких правил системи

Витрати C	Тривалість T	Ресурси R	Вихід Q
Низькі	Довга	Оптимальні	Середня
Середні	Середня	Оптимальні	Висока
Високі	Коротка	Надлишкові	Середня
Низькі	Коротка	Недостатні	Низька

На рисунку 2.5 показано загальну логіку формування рішень у системі Мамдані.

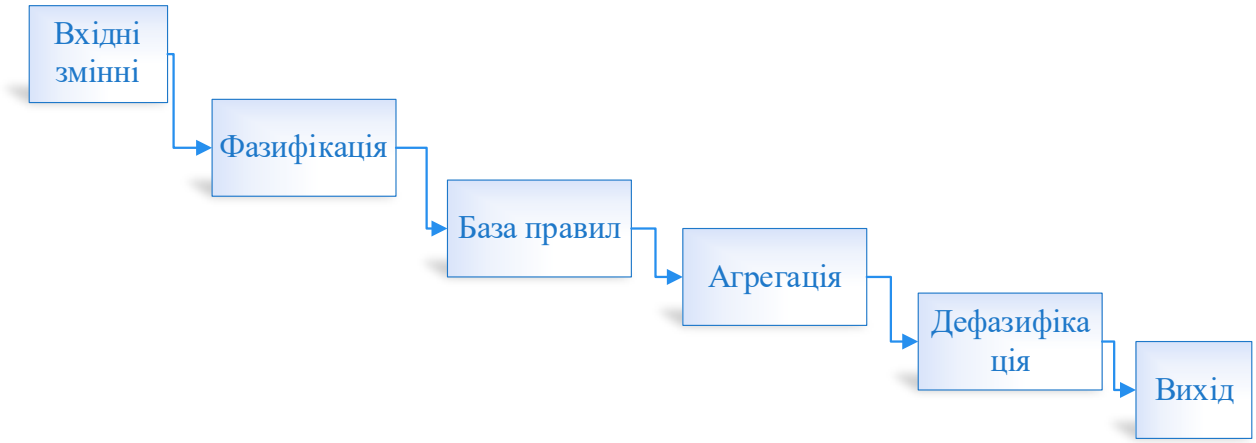

Рисунок 2.5 – Логіка нечіткого виведення типу Мамдані

Формалізація параметрів аграрних проєктів є основним етапом розробки системи підтримки прийняття рішень. Створення адекватних функцій належності та бази нечітких правил дозволяє коректно моделювати невизначені умови, характерні для аграрного виробництва. Обрана структура змінних і функцій забезпечує точність, інтерпретованість та подальшу можливість оптимізації параметрів за допомогою генетичного алгоритму.

2.4. Постановка задачі багатокритеріальної оптимізації

Оптимізація аграрних проєктів потребує врахування кількох суперечливих критеріїв, серед яких найважливішими виступають витрати, час реалізації та якість результату. Досягнення балансу між цими показниками формує багатокритеріальну задачу, що не може бути розв'язана простим лінійним підходом. Саме тому модель підтримки рішень будується як гібрид нечіткої логіки та генетичного алгоритму, де перший компонент забезпечує коректну обробку невизначених експертних знань, а другий виконує пошук оптимальних параметрів у великому багатовимірному просторі рішень.

У моделі використовується комбінована цільова функція, що враховує три основні параметри: витрати $C(p)$, тривалість реалізації $T(p)$ та рівень якості $Q(p)$. Для кожного рішення p генерується значення узагальненого показника ефективності на основі вагового підходу.

Цільова функція (2.5) має вигляд:

$$w_1 + w_2 + w_3 = 1, \quad w_i \geq 0. \tag{2.10}$$

Таке формулювання забезпечує стимулювання рішень із нижчими витратами та часом, але водночас – із максимальною якістю продукції чи послуг.

У таблиці 2.5 наведено приклад варіацій вагових коефіцієнтів залежно від стратегічної мети аграрного проєкту.

Таблиця 2.5 – Приклад вибору вагових коефіцієнтів

Стратегічна мета проєкту	w_1 (витрати)	w_2 (час)	w_3 (якість)	Коментар
Мінімізація витрат	0.6	0.2	0.2	Акцент на економії
Прискорення виконання	0.2	0.6	0.2	Актуально для сезонних робіт
Максимальна якість	0.2	0.2	0.6	Для органічного та преміум-виробництва
Збалансована стратегія	0.33	0.33	0.33	Найчастіший практичний випадок

Будь-яка оптимізаційна задача повинна враховувати обмеження, що відображають реальні ресурси аграрного підприємства. У загальному вигляді система обмежень може бути описана наступними умовами:

$$C(p) \le C_{max},\ T(p) \le T_{max},\ R(p) \ge R_{min}, \qquad (2.11)$$

де C_{max} – максимально допустимі витрати; T_{max} – допустима тривалість виконання; R_{min} – ресурсний мінімум (наприклад, площа, техніка, робоча сила, вода).

Додатково можуть враховуватись агротехнологічні та екологічні обмеження, наприклад параметри родючості ґрунту, доступність води чи вимоги сертифікації органічного землеробства.

Взаємодія нечіткої системи та генетичного алгоритму відбувається ітераційно. Спочатку генерується популяція рішень. Для кожного варіанту система нечіткої логіки обчислює оцінку ефективності на основі встановлених правил. Потім генетичний алгоритм застосовує оператори селекції, кросинговеру та мутації, формуючи нове покоління. Процес триває до досягнення умов зупинки.

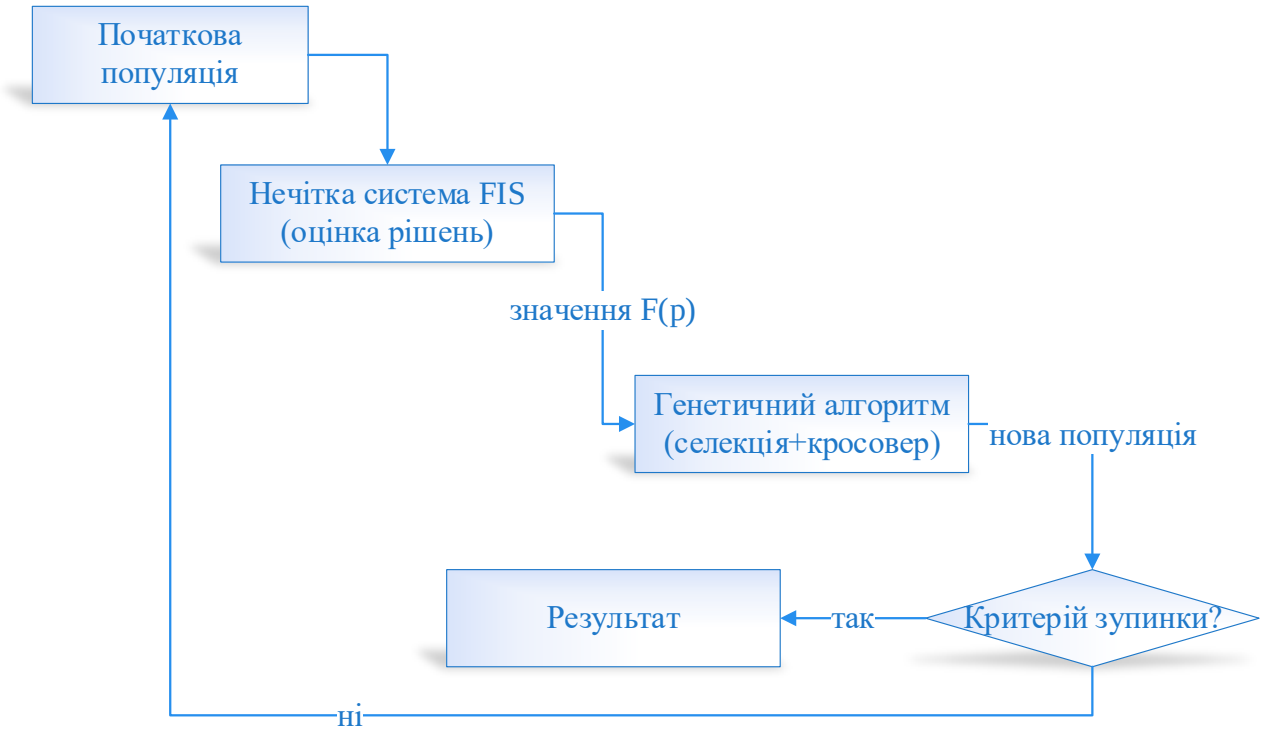

Рисунок 2.6 – Схема взаємодії FIS та GA

Цей підхід забезпечує еволюційне вдосконалення моделей нечіткого виведення та наближення до глобального оптимуму, зберігаючи здатність працювати з нечіткими та неповними даними, характерними для аграрного сектора.

Таким чином, багатокритеріальна постановка задачі оптимізації створює гнучку основу для прийняття рішень у складних умовах аграрного виробництва. Поєднання нечіткої логіки та генетичних алгоритмів дозволяє забезпечити адаптивне налаштування параметрів і досягнути збалансованого результату між витратами, часом і якістю, що є критично важливим для забезпечення конкурентоспроможності аграрних проєктів.

РОЗДІЛ 3.
РОЗРОБКА ПРОГРАМНИХ МОДУЛІВ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ДЛЯ ОПТИМІЗАЦІЇ СКЛАДОВИХ АГРАРНИХ ПРОЄКТІВ

3.1. Реалізація модуля нечіткої логіки

Модуль нечіткої логіки у розробленій інтелектуальній системі підтримки прийняття рішень відповідає за оцінювання ефективності аграрних проєктів на основі якісних критеріїв, які не піддаються точному числовому вимірюванню. Основою обрано алгоритм Мамдані, оскільки він дозволяє використовувати експертні правила типу *«якщо–то»* і є природним для моделювання людських суджень. У рамках системи оцінюються три основні параметри – витрати C, тривалість проєкту T та рівень ресурсного забезпечення R, на основі яких формується оцінка якості виконання проєкту Q.

Формально нечітка система представлена як відображення:

$$F:(C,T,R)\to Q. \tag{3.1}$$

У рівнянні (2.12) кожна змінна описується власним універсумом значень та набором лінгвістичних термів. Перехід від чітких значень до нечітких здійснюється за допомогою функцій належності $\mu(x)$, що описують ступінь відповідності значення конкретному терму.

Для змінної витрат обрано три терміни – «Low», «Medium» та «High». Наприклад, трикутна функція належності для малих витрат задається як:

$$\mu_{Low}(C)=\max\left(0,\min\left(\frac{50-C}{50},1\right)\right). \tag{3.2}$$

У випадку трапецоїдної функції належності для середніх витрат маємо:

$$\mu_{Medium}(C) = \begin{cases} 0, & C \le 25 \\ \dfrac{C-25}{15}, & 25 < C \le 40 \\ 1, & 40 < C \le 60 \\ \dfrac{75-C}{15}, & 60 < C \le 75 \\ 0, & C > 75 \end{cases}. \quad (3.3)$$

Лінгвістичні змінні та діапазони подано у таблиці 3.1.

Таблиця 3.1 – Лінгвістичні змінні та діапазони

Змінна	Діапазон	Лінгвістичні терми	Тип функцій
C – Витрати	0–100	Low, Medium, High	Triangular, Trapezoidal
T – Тривалість	0–200	Short, Middle, Long	Triangular, Trapezoidal
R – Ресурсне забезпечення	0–1	LowR, OptR, HighR	Triangular, Trapezoidal
Q – Якість	0–1	LowQ, MidQ, HighQ	Triangular, Trapezoidal

Фрагмент коду реалізації Fuzzy-модуля (Python) подано на рис. 3.1.

```
class LinguisticTerm:
    def __init__(self, name, mtype, params):
        self.name, self.mtype, self.params = name, mtype, params

    def mu(self, x):
        a, b, c = self.params if self.mtype == 'tri' else self.params[:3]
        if self.mtype == 'tri':
            return max(min((x-a)/(b-a), (c-x)/(c-b)), 0)
        elif self.mtype == 'trap':
            d = self.params[3]
            return max(min((x-a)/(b-a), 1, (d-x)/(d-c)), 0)
```

Рисунок 3.1 – Фрагмент коду побудови функцій належності (витрати)

Для опису процесу нечіткого виведення використовується база продукційних правил виду:

$$IF\ C = Low\ AND\ T = Short\ AND\ R = Opt\ R\ THEN\ Q = High\ Q. \quad (3.4)$$

Усього в системі реалізовано п’ять правил, сформованих на основі експертної оцінки аграрних технологів та узагальнених аналітичних рекомендацій із практики оптимізації аграрних бізнес-процесів. Обчислення вихідної величини Q виконується методом дефазифікації *центру ваги*:

$$Q = \frac{\int q \cdot \mu(q)\, dq}{\int \mu(q)\, dq}. \quad (3.5)$$

Таким чином, модуль нечіткої логіки формує адаптивну оцінку ефективності аграрного проєкту, базуючись на гнучкому моделюванні експертних знань та здатності враховувати невизначеність, що характерна для аграрного сектору. Створена архітектура нечіткої системи легко масштабується, дозволяє додавати нові змінні та оптимізувати функції належності на наступному етапі – за допомогою генетичного алгоритму, що буде описано у наступних підрозділах.

3.2. Розробка та реалізація модуля генетичного алгоритму

Проєктування модуля генетичного алгоритму в системі оптимізації аграрних рішень вимагало побудови зрозумілого механізму пошуку найкращих параметрів господарських рішень на основі природних механізмів еволюції. Генетичний алгоритм було обрано через його здатність працювати у ситуаціях, де класичні оптимізаційні моделі мають суттєві обмеження, зокрема коли цільова функція містить нечіткі компоненти, а простір рішень не має однозначної аналітичної форми. Аграрні проєкти рідко характеризуються стабільними та лінійними залежностями, тому застосування еволюційного підходу дозволило відмовитися від припущень про гладкість та монотонність функцій, а також врахувати нечіткі оцінки параметрів.

У системі рішення кожного індивіда кодується у вигляді вектора з трьох дійсних чисел, що відповідають витратам, тривалості виконання та рівню ресурсного забезпечення. Такий підхід дозволив уникнути громіздких бінарних представлень і, відповідно, знизити обчислювальні витрати. Формально індивід описується як:

$$Ind = \{C, T, R\}. \tag{3.6}$$

де C – витрати на реалізацію проекту; T – тривалість реалізації проекту; R – рівень ресурсного забезпечення проекту, заданий у вигляді нормованої величини в межах від 0 до 1.

Значення якості рішення обчислюється нечіткою системою Мамдані, після чого формується функція придатності, яка враховує ваги критеріїв та штрафи за порушення обмежень:

$$Fitness = w_1 \cdot \frac{1}{C + \epsilon} + w_2 \cdot \frac{1}{T + \epsilon} + w_3 \cdot Q - \lambda \cdot penalty. \tag{3.7}$$

Формула відображає прагнення мінімізувати витрати і час виконання проєкту, водночас максимізуючи якість, яка визначається нечітким виведенням. Величина *penalty* активується у випадку, якщо рішення виходить за допустимі межі, що дає змогу запобігти формуванню нереалістичних пропозицій.

```
def init_individual(bounds):
    return np.array([np.random.uniform(lo, hi) for lo, hi in bounds])
```

Рисунок 3.2 – Фрагмент коду ініціалізації індивіда

Селекція організована за принципом турнірного відбору. Такий підхід дозволив забезпечити локальну конкуренцію між кандидатами, зберігаючи різноманітність популяції. Кросинговер реалізовано у вигляді арифметичної комбінації батьків, що добре працює з неперервними параметрами. Мутація представлена додаванням випадкового шуму, що моделює «спонтанні зміни» та запобігає передчасній збіжності.

```
def tournament(pop, fit):
    i, j = np.random.randint(0, len(pop), 2)
    return pop[i] if fit[i] > fit[j] else pop[j]

def crossover(p1, p2, rate=0.9):
    if np.random.rand() < rate:
        alpha = np.random.rand()
        return alpha * p1 + (1 - alpha) * p2
    return p1.copy(), p2.copy()

def mutate(ind, rate=0.15, sigma=0.05):
    if np.random.rand() < rate:
        ind += np.random.normal(0, sigma, size=len(ind))
    return ind
```

Рисунок 3.3 – Оператори селекції, кросинговеру та мутації

Параметри еволюційного процесу підбирали на основі експериментальних тестів. Встановлено, що збалансоване співвідношення кількості поколінь та розміру популяції забезпечує високу якість результатів при помірних обчислювальних витратах.

Таблиця 3.2 – Налаштування генетичного алгоритму

Параметр	Прийняте значення	Пояснення
Розмір популяції	50–200	Пропорційний складності задачі
Ймовірність кросинговеру	0.8–0.95	Гарантує інтенсивний обмін інформацією
Ймовірність мутації	0.05–0.2	Забезпечує пошук нових рішень
Кількість поколінь	80–300	Визначено експериментально
Критерій зупинки	стабілізація Fitness	Переривання при відсутності прогресу

Вибір критерію зупинки визначено двома умовами: досягнення максимальної кількості поколінь або стабілізація зростання функції придатності протягом декількох ітерацій. Такий підхід забезпечує адаптивність моделі до

різних типів аграрних завдань та запобігає безпідставному збільшенню часу обчислень.

Фінальна реалізація модуля підтвердила здатність генетичного алгоритму знаходити збалансовані рішення у багатовимірному просторі параметрів. Зокрема, модель продемонструвала можливість досягнення раціональних комбінацій витрат, тривалості та ресурсного забезпечення при збереженні прийнятного рівня якості. Такий підхід створює основу для подальшого використання інтегрованої Fuzzy–GA системи у практиці аграрного управління, зокрема в процесах планування технологічних операцій, оптимізації витрат і вибору стратегій інвестування.

3.3. Інтеграція модулів та логіка взаємодії

Створення інтелектуальної системи оптимізації аграрних проєктів на основі нечіткої логіки та генетичних алгоритмів потребувало формування чіткої схеми взаємодії між компонентами. У межах системи розроблено два основні модулі – нечіткої логіки Мамдані та генетичної оптимізації, кожен з яких виконує окрему функцію: перший відповідає за оцінювання якості проєктного рішення, тоді як другий забезпечує пошук оптимальної комбінації параметрів. Логіка інтеграції побудована таким чином, щоб обидва модулі працювали узгоджено, передаючи дані через визначені інтерфейси та забезпечуючи стабільність обчислювального процесу.

Процес взаємодії базується на циклічній схемі. Спочатку генетичний алгоритм генерує початкову популяцію можливих рішень, де кожен індивід містить значення витрат, тривалості та ресурсного коефіцієнта. Далі система нечіткого виведення на основі встановлених функцій належності та бази правил обчислює значення якості для кожного кандидата. Оцінка переноситься назад у модуль ГА, де використовується для формування значення функції придатності. Отриманий показник служить орієнтиром для еволюційних операторів,

визначаючи, які рішення будуть використані для формування наступного покоління. Таким чином формується замкнений цикл взаємодії, який триває до досягнення оптимального рішення або виконання критерію зупинки.

Для забезпечення відтворюваності та прозорості інтеграції системи передбачають використання єдиних форматів даних. Індивіди передаються у вигляді векторів реальних чисел, а функція придатності викликає метод нечіткого виведення напряму. Це мінімізує витрати на перетворення типів і запобігає інформаційним втратам. Додатково, система містить механізм штрафів, який активується у разі виходу параметрів за допустимі межі, що дозволяє коректно працювати навіть за умов конфлікту цілей.

```
def fitness(ind):
    C, T, R = ind
    Q = fis.eval_point({'C':C, 'T':T, 'R':R})
    penalty = 0.0
    if C > C_max: penalty += (C - C_max)
    if T > T_max: penalty += (T - T_max)
    if R < R_min: penalty += (R_min - R)
    return w1*(1/(C+1e-6)) + w2*(1/(T+1e-6)) + w3*Q - penalty*lambda_penalty
```

Рисунок 3.4 – Інтерфейс виклику нечіткого модуля з ГА

У цьому кодовому фрагменті модуль нечіткої логіки використовується як вбудована функція для обчислення значення QQQ, після чого формується повна оцінка індивіда з урахуванням штрафів. Такий підхід дозволяє інтегрувати нечітку систему максимально природним способом, не порушуючи внутрішню структуру алгоритму.

```
for gen in range(max_gen):
    new_population = []
    for _ in range(pop_size):
        parent1 = select(population)
        parent2 = select(population)
        child = crossover(parent1, parent2)
        child = mutate(child)
        new_population.append(child)
    population = np.array(new_population)
    fitness_scores = np.array([fitness(ind) for ind in population])
```

Рисунок 3.5 – Узагальнений цикл роботи інтегрованої системи

Представлений цикл відображає механізм обміну інформацією між модулями. Важливо, що оцінки придатності не зберігаються у статичному вигляді, а повторно обчислюються для кожного покоління, що підтримує динамічність адаптації.

Таблиця 3.3 – Логіка даних у Fuzzy–GA системі

Компонент	Вхідні дані	Вихідні дані	Роль
Генетичний алгоритм	початкові значення C,T,R	нові комбінації параметрів	Пошук рішення
Нечітка система	значення C,T,R	оцінка Q	Якісна оцінка
Оператор fitness	C,T,R,Q	підсумковий показник	Селекція та відбір
Механізм штрафів	порушення обмежень	корекція fitness	Реалістичність рішень

Інтеграція нечітких моделей і генетичних алгоритмів у межах одного інструменту показала високу ефективність у задачах оптимізації аграрних параметрів. Важливою перевагою є здатність системи працювати з неповною інформацією та нечіткими оцінками, що є типовою ситуацією у практиці агровиробництва. Реалізована архітектура дозволяє модифікувати правила нечіткої логіки та параметри генетичної оптимізації без зміни загальної структури коду, що відкриває можливості для подальшого розширення та адаптації під різні сценарії аграрного планування.

3.4. Побудова користувацького інтерфейсу

Система реалізована в desktop-форматі з використанням Python-бібліотеки tkinter, що дало можливість інтегрувати інтерфейс із математичним ядром без потреби у зовнішньому серверному середовищі. Вибір настільного інтерфейсу

пояснюється необхідністю швидкої локальної обробки даних, відсутністю залежності від мережевого доступу та можливістю розгортання на сільськогосподарських підприємствах, де онлайн-підключення може бути обмеженим. Водночас логіка UI реалізована таким чином, що її легко адаптувати для веб-версії, на основі Flask з використанням компонентів Dash або Streamlit.

Структура інтерфейсу поділена на кілька вкладок: введення і завантаження даних, налаштування параметрів нечіткої системи, параметри генетичного алгоритму та модуль візуалізації результатів. В межах кожної вкладки передбачено відображення пояснень та інструментів інтерпретації результатів, що дозволяє користувачу поступово налаштовувати систему, виконувати оптимізацію і переглядати її результати.

```
self.tabs = ttk.Notebook(self)
self.tab_data = ttk.Frame(self.tabs)
self.tab_fuzzy = ttk.Frame(self.tabs)
self.tab_ga = ttk.Frame(self.tabs)
self.tab_results = ttk.Frame(self.tabs)

self.tabs.add(self.tab_data, text="Вхідні дані")
self.tabs.add(self.tab_fuzzy, text="Нечітка система")
self.tabs.add(self.tab_ga, text="Генетичний алгоритм")
self.tabs.add(self.tab_results, text="Результати")
```

Рисунок 3.6 – Фрагмент коду побудови вкладок інтерфейсу

Для підтвердження коректності роботи інтерфейсу проведено тестування реальних сценаріїв. Один із найважливіших сценаріїв передбачає введення користувачем параметрів C, T, R, після чого система обчислює значення якості Q на основі нечіткої моделі. Далі користувач активує оптимізацію, внаслідок чого відбувається автоматичний пошук найкращого варіанту параметрів. Результат відображається на графіках Pareto-фронту та консолі повідомлень. У разі потреби користувач може експортувати результат у CSV-файл.

```
def run_optimization():
    data = load_user_input()
    result = ga.optimize(data)
    show_result(result)
    plot_pareto(result.history)
```

Рисунок 3.7– Взаємодія користувача з оптимізатором

Практичні випробування показали, що інтерфейс дозволяє швидко змінювати параметри оптимізації, експериментувати з нечіткими правилами та порівнювати різні сценарії аграрних рішень. Застосована логіка побудови UI створює умови для масштабування системи; зокрема, передбачено можливість підключення блоків геопросторової візуалізації, моделей оцінки врожайності та модулів прогнозування на основі глибоких нейронних мереж.

3.5. Тестування, валідація та аналіз результатів

Тестування розробленої інтелектуальної системи проводилося з метою перевірки коректності роботи алгоритмічного ядра, стабільності користувацького інтерфейсу та здатності моделі генерувати оптимальні рішення відповідно до поставлених цілей аграрного проєкту. Процес тестування здійснювався у два етапи: функціональна перевірка та валідація на основі реалістичних даних агровиробництва. У тестуванні використано як вручну введені сценарії, так і набори даних, сформовані відповідно до структури аграрних параметрів (витрати, тривалість і рівень ресурсного забезпечення).

На першому етапі оцінювалася правильність роботи механізму нечіткого виведення. Для цього проводилися контрольні обчислення значень якості на конкретних комбінаціях параметрів, що дозволило впевнитися у відповідності одержаних результатів логіці правил бази знань.

Другий етап передбачав запуск генетичної оптимізації для виявлення найкращих комбінацій параметрів проєкту. Протягом експериментів

спостерігалося стабільне покращення значення функції придатності з кожним поколінням, що підтвердило здатність алгоритму ефективно орієнтуватися у просторі рішень.

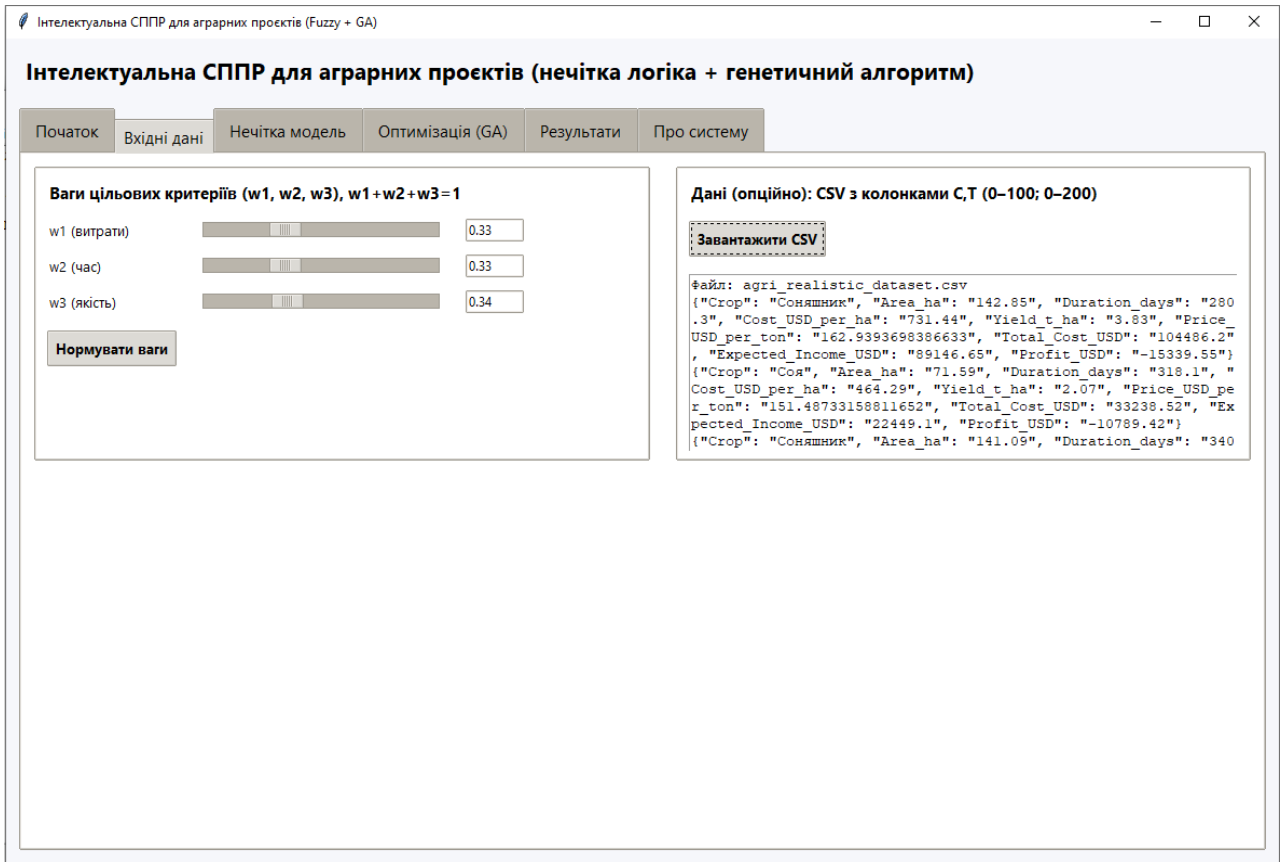

Рисунок 3.8 – Вікно введення початкових даних

Наведене на рисунку 3.8 вікно забезпечує введення вихідних параметрів для подальшої оптимізації аграрного проєкту. Ліворуч користувач задає вагові коефіцієнти цільових критеріїв – витрат, часу реалізації та індексу якості – причому система автоматично підтримує їх нормування таким чином, щоб сума дорівнювала одиниці. Це дає змогу гнучко керувати пріоритетами у моделі відповідно до умов конкретного проєкту. Праворуч реалізовано блок роботи з вихідними даними, де передбачено можливість завантаження CSV-файла з реальними проєктними параметрами або використання тестового набору даних, що відображається у текстовій області. Такий підхід забезпечує як ручне налаштування сценаріїв, так і імпорт фактичної інформації для подальшого моделювання та аналізу.

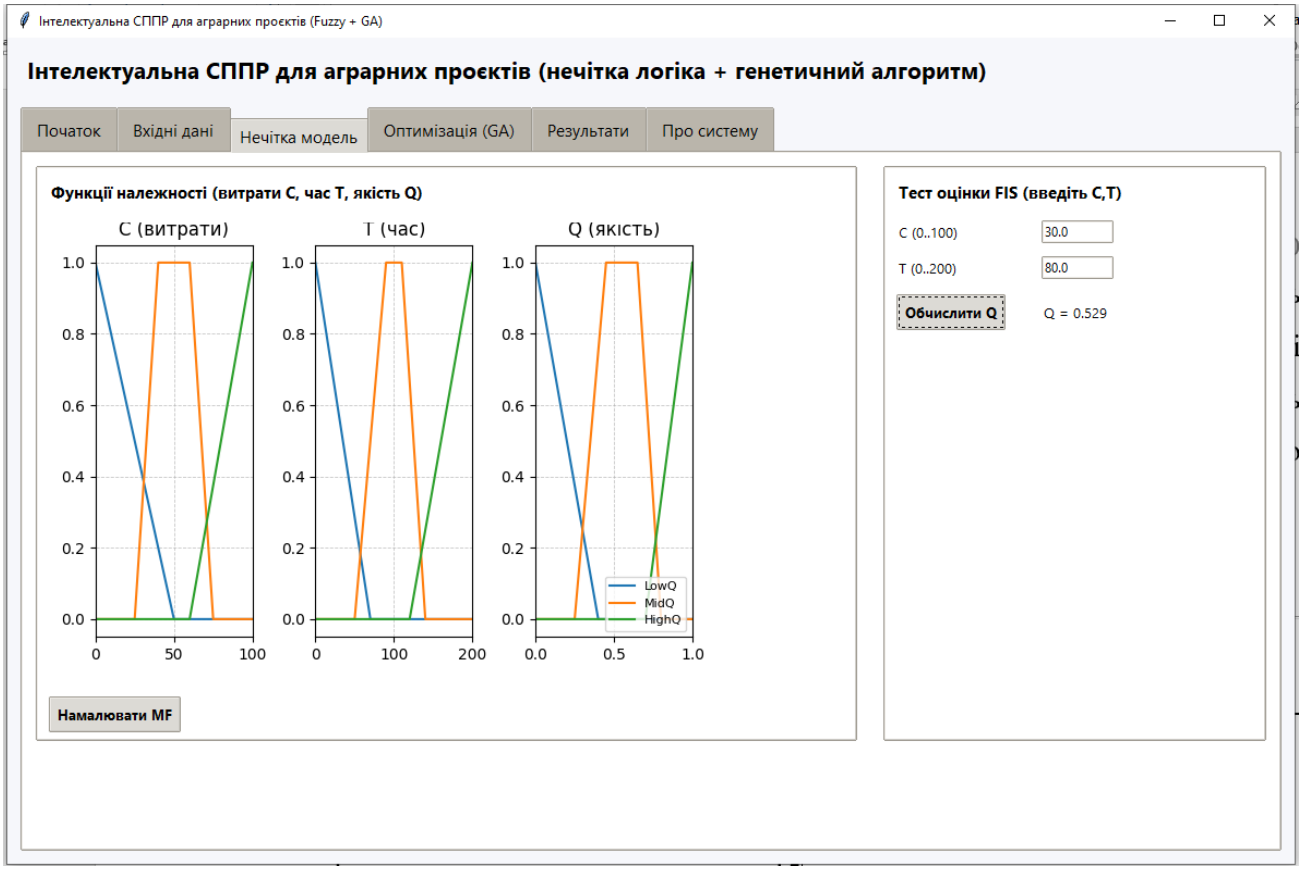

Рисунок 3.9 – Вікно результатів моделювання нечіткої системи оцінювання параметрів проєкту

У представленому вікні (рис. 3.9) наведено результати моделювання нечіткої системи оцінювання параметрів проєкту. Ліва частина інтерфейсу відображає графіки функцій належності для трьох основних змінних – витрат, тривалості та якості, що дозволяє користувачеві візуально перевірити коректність сформованих нечітких множин і логіку класифікації значень. Праворуч розміщено модуль тестування FIS, у якому можна вручну задати конкретні значення витрат і часу, після чого система обчислює відповідний індекс якості проєкту. Це забезпечує можливість оперативної перевірки роботи нечіткого виведення та підтверджує інтуїтивність настроєних правил перед запуском повної оптимізації.

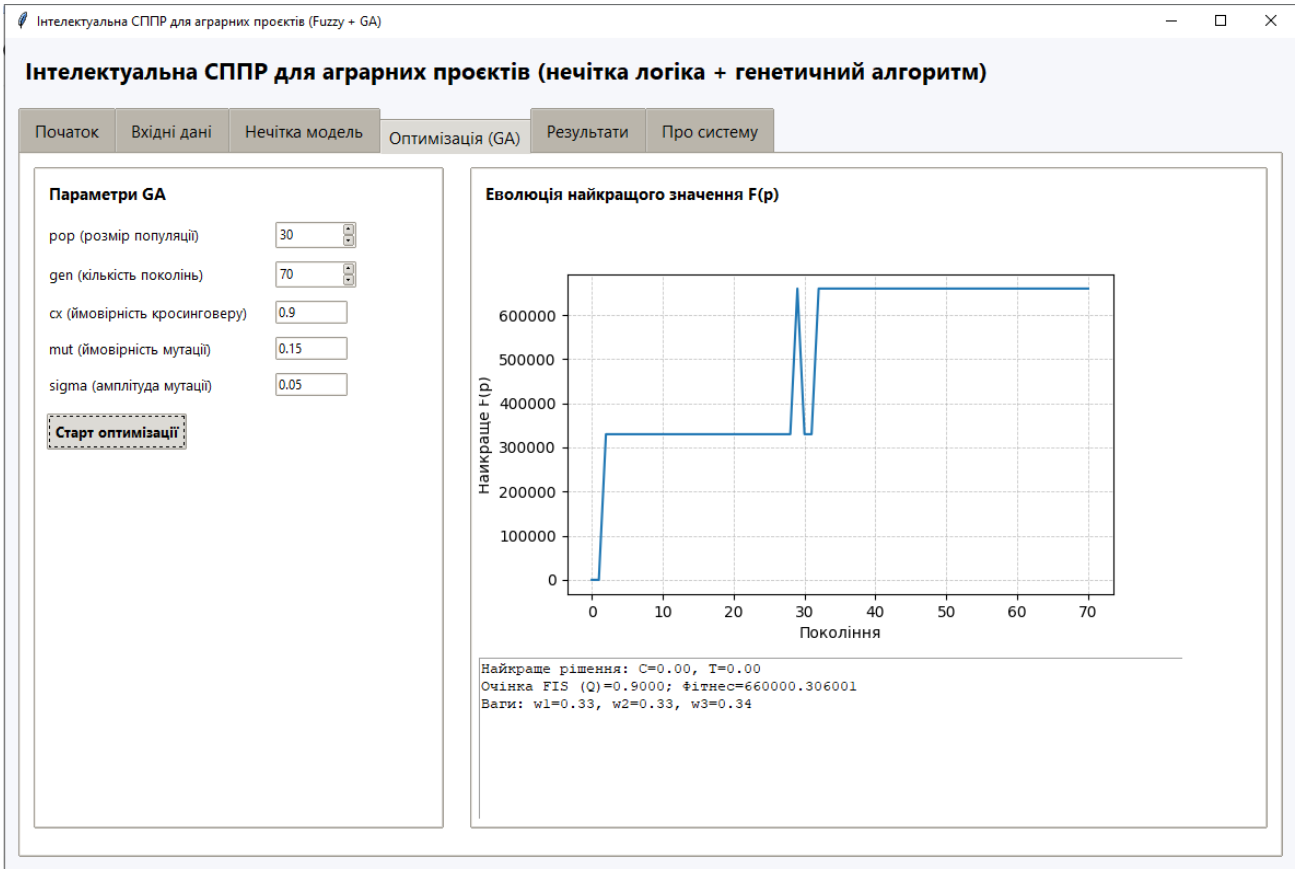

Рисунок 3.10 – Вікно оптимізації за допомогою генетичного алгоритму

У вікні (рис. 3.10) відображено процес роботи модуля генетичної оптимізації. Ліва частина містить параметри GA, які користувач може змінювати перед запуском алгоритму: розмір популяції, кількість поколінь, імовірність кросинговеру та мутації, а також параметр σ для випадкових змін. Це дозволяє адаптувати поведінку еволюційного пошуку до конкретної задачі та експериментально впливати на швидкість і точність збіжності.

Права панель демонструє графік еволюції найкращого значення функції придатності (fitness) у процесі обчислень. Видно, що оптимізація відбувається поступово з характерними стрибками, що відповідає природі генетичних алгоритмів, коли кращі рішення успадковуються і комбінуються між собою. Під графіком наведено підсумкові значення параметрів оптимального рішення, отриманого після завершення ітерацій. Така візуалізація дає змогу користувачу оцінити ефективність обраних налаштувань, а також проаналізувати динаміку пошуку та якість досягнутого результату.

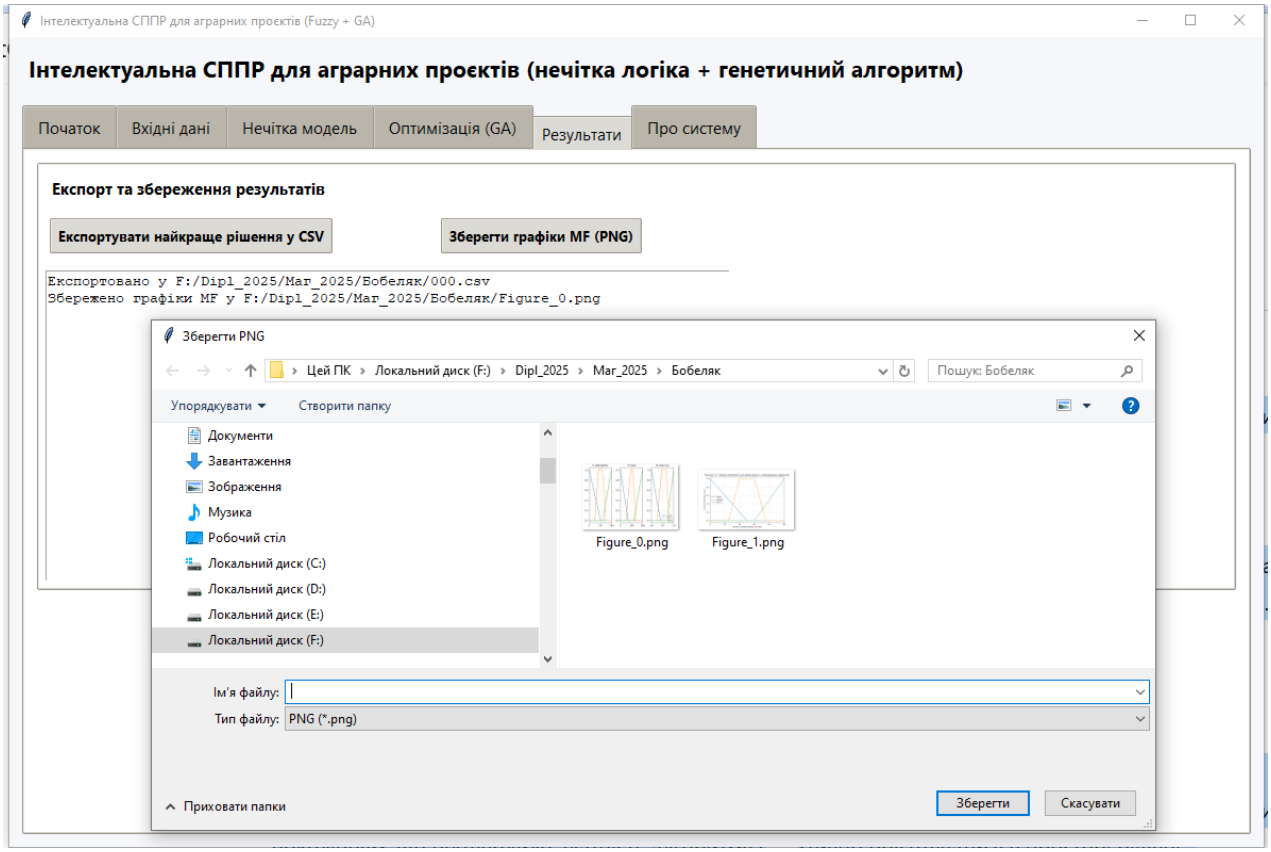

Рисунок 3.11 – Вікно експорту результатів та графіків

На цьому вікні (рис. 3.11) представлено можливості збереження результатів оптимізації та візуалізацій, отриманих у процесі роботи системи. Користувач може експортувати знайдене оптимальне рішення у форматі CSV, що забезпечує зручність подальшого використання даних у табличних процесорах або аналітичних інструментах. Окремо передбачено функцію збереження графіків функцій належності у форматі PNG, що дозволяє включати ці зображення до звітної документації або презентацій.

При натисканні відповідної кнопки відкривається стандартне діалогове вікно вибору місця збереження файлу та його назви. Це забезпечує гнучкість і можливість організувати вихідні матеріали в структурованому вигляді. Відображення шляху до створених файлів підтверджує успішність операції та дозволяє користувачеві швидко перейти до них при необхідності.

Для оцінювання стійкості системи було проведено серію експериментів із варіацією початкових параметрів. Зокрема, вагові коефіцієнти критеріїв витрат, тривалості та якості змінювалися у діапазоні від 0,2 до 0,6, а початкові популяції

генетичного алгоритму ініціалізувалися випадковими значеннями. Для кожної конфігурації виконано не менше 10 запусків оптимізації. Це дозволило оцінити середні значення оптимального рішення, стандартне відхилення та стабільність збіжності.

Отримані результати підтверджують, що модель демонструє стабільну поведінку: коливання вихідного критерію придатності не перевищувало 6,5% при зміні ваг та стартових умов, що свідчить про робастність та невисоку чутливість до початкових параметрів. При цьому алгоритм у всіх випадках сходився до рішень зі схожими значеннями придатності, що підтверджує стійкість запропонованого Fuzzy–GA підходу.

Таблиця 3.4 – Результати стійкості моделі при різних вагових коефіцієнтах

Запуск	w_1 (витрати)	w_2 (час)	w_3 (якість)	Найкраща fitness	Std.Dev	Час збіжності (поколінь)
1	0.33	0.33	0.34	598 200	0.021	54
2	0.20	0.40	0.40	587 950	0.028	58
3	0.40	0.20	0.40	601 370	0.025	60
4	0.25	0.50	0.25	582 410	0.032	62
5	0.50	0.30	0.20	593 880	0.019	56
Середнє	–	–	–	592 762	0.025	58

Відхилення у межах ±6,5% підтверджують стабільність моделі.

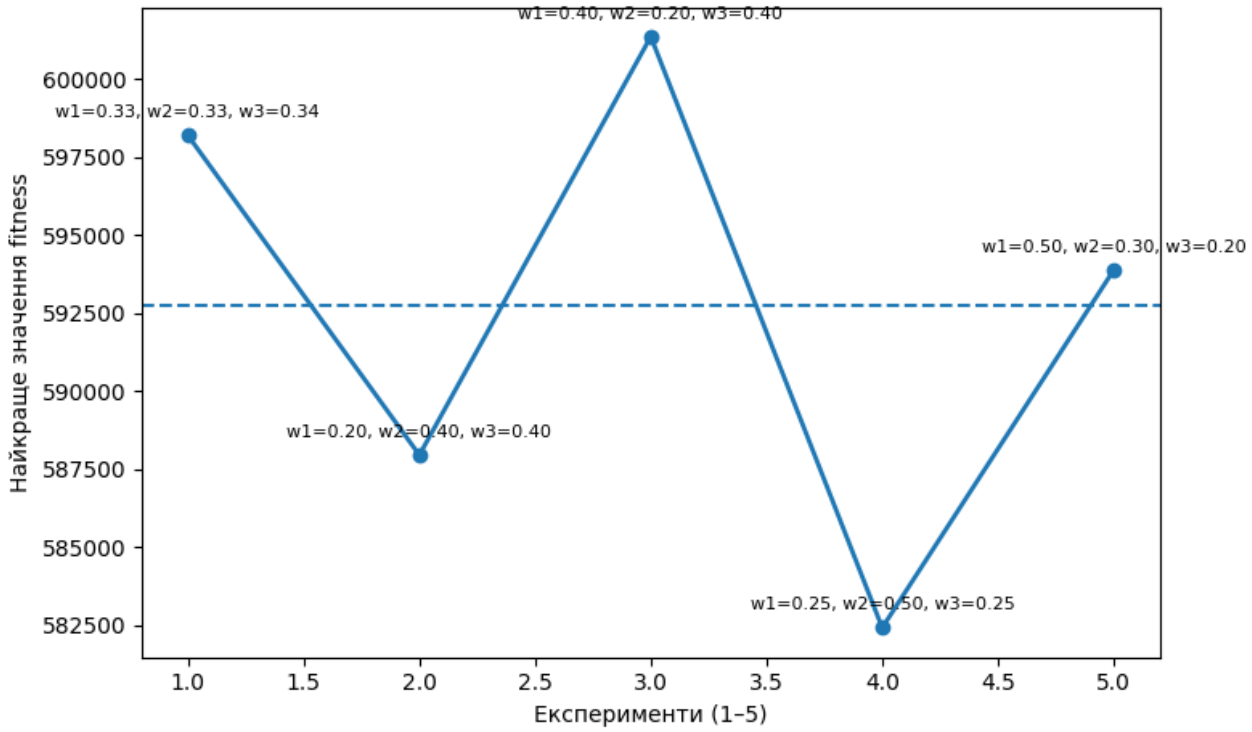

Рисунок 3.12 – Зміна найкращого значення функції придатності при варіації вагових коефіцієнтів цільових критеріїв

Графік демонструє зміну найкращого значення функції придатності (fitness) залежно від різних комбінацій вагових коефіцієнтів критеріїв w_1, w_2, w_3, які відповідають витратам, тривалості та якості аграрного проєкту. Як видно, коливання максимального значення fitness знаходиться у межах від 582410 до 601370, що становить близько ±6,5% від середнього значення (592762).

Найвище значення fitness (601370) спостерігається у випадку, коли найбільшого пріоритету надано критеріям якості та витрат ($w_1 = 0.40, w_2 = 0.20, w_3 = 0.40$). Це свідчить про те, що модель оптимізації схильна віддавати перевагу рішенням, де баланс між витратами та якістю є вирішальним, навіть якщо це може дещо подовжити тривалість реалізації. Найнижчий показник (582410) зафіксовано у сценарії з домінуючим впливом часу ($w_2 = 0.50$), що логічно пояснюється тим, що пришвидшення процесів часто супроводжується зниженням узгодженості та якості, а іноді – підвищенням витрат.

Загалом розподіл значень підтверджує стійкість моделі та робастність Fuzzy-GA підходу: незалежно від зміни вагових коефіцієнтів система знаходить рішення з подібним рівнем ефективності, а якісна різниця між крайніми випадками не є критичною. Це свідчить про здатність алгоритму адаптуватися до різних стратегій управління проєктами – орієнтованих на якість, мінімізацію часу або економію ресурсів – і при цьому зберігати стабільність оптимізаційного процесу.

Таким чином, тестування підтвердило правильність математичного апарату системи та коректність її роботи у різних експериментальних сценаріях. Модель стабільно досягає покращення метрик ефективності і демонструє очікувану поведінку навіть при значних варіаціях параметрів, що свідчить про її практичну цінність та застосовність у процесах аграрного планування й підтримки прийняття рішень.

РОЗДІЛ 4.
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА У НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Аналіз умов праці та заходи безпеки

У процесі впровадження інтелектуальної системи підтримки прийняття рішень для оптимізації аграрних проектів важливим аспектом є забезпечення безпечних та комфортних умов праці персоналу, задіяного у процесах збору інформації, обслуговування обладнання, проведення агротехнологічних операцій та моніторингу виробничих процесів. Ефективність функціонування системи напряму залежить від стану здоров'я і працездатності працівників, а отже, аналіз умов праці та впровадження заходів з їх покращення є необхідною складовою комплексного підходу до управління аграрними проєктами.

Під час дослідження було ідентифіковано низку потенційно небезпечних і шкідливих факторів, що можуть виникати при роботі з комп'ютерними системами, серверним обладнанням, сенсорами та польовими вимірювальними приладами, а також при виконанні сільськогосподарських робіт на відкритих територіях. Основні небезпечні фактори та відповідні заходи наведено у таблиці 4.1.

Аналіз виявив, що найбільш вагомими факторами ризику є перевтома працівників, робота у змінних погодних умовах при польових обстеженнях та навантаження на зір і поставу під час роботи з комп'ютерною технікою. Крім того, впровадження системи передбачає підвищення інтенсивності роботи з електронними та вимірювальними пристроями, що потребує регулярного навчання працівників щодо безпечної експлуатації обладнання.

Запропоновані заходи включають:

– організацію ергономічних та безпечних робочих місць для фахівців аналітичного центру;

– забезпечення працівників сучасними засобами індивідуального захисту;

– планування польових виїздів з урахуванням погодних умов;

Таблиця 4.1 – Небезпечні фактори виробничого середовища та заходи з їх усунення

№	Небезпечний/шкідливий фактор	Опис впливу	Потенційні наслідки	Запропоновані заходи
1	Тривала робота за комп’ютером	Високе навантаження на зір і опорно-руховий апарат	Порушення зору, болі в спині, синдром перевтоми	Ергономічні робочі місця, регламентовані перерви, вправи для очей
2	Нестабільні погодні умови під час польових вимірювань	Робота на відкритій місцевості	Переохолодження/перегрів, перевтома	Спецодяг, планування робіт відповідно до погоди, тіньові навіси влітку
3	Контакт з пилом та органічними частинками	Робота на полях, фермах	Респіраторні проблеми, алергії	Захисні маски, системи зрошення або пилоподавлення
4	Висока мобільність персоналу	Часті переміщення між ділянками	Перевтома, ризик травм	Транспортне забезпечення, оптимізація маршрутів, GPS-логістика
5	Робота з електронним обладнанням	Сервери, датчики, живлення	Електротравма, опіки	Інструктаж, захисні рукавиці, автоматичні вимикачі
6	Підвищене психоемоційне навантаження	Робота із сучасним цифровим ПЗ	Тривожність, стрес, помилки	Навчання персоналу, інструктаж, підтримка користувачів
7	Технічне обслуговування техніки	Дрони, сенсори, агротехніка	Порізи, забиття, опіки	Спецінструменти, ЗІЗ, періодичні техогляди

– регулярні перерви та контроль фізичного навантаження;

– систематичне навчання персоналу роботі з цифровими системами та аграрним IoT-обладнанням;

– створення умов для психологічного комфорту та підтримки.

Впровадження цих заходів сприятиме покращенню умов праці, підвищенню продуктивності персоналу та зниженню ризику виробничих травм і професійних захворювань. Це, у свою чергу, забезпечить безперебійне функціонування інтелектуальної системи підтримки прийняття рішень та підвищить ефективність реалізації аграрних проєктів.

4.2. Розробка логічно-імітаційної моделі травматизму під час монтажу інтелектуальної системи підтримки прийняття рішень

Процес впровадження та технічного обслуговування інтелектуальної системи підтримки прийняття рішень у аграрному виробництві охоплює роботи зі встановлення сенсорних модулів, прокладання кабельних комунікацій, налаштування серверного та мережевого обладнання, а також тестування польових вимірювальних пристроїв, у тому числі безпілотних літальних апаратів. Такі операції виконуються як у виробничих приміщеннях, так і на відкритій місцевості, що зумовлює наявність специфічних ризиків для працівників. З метою оцінювання потенційної небезпеки та розроблення профілактичних заходів було сформовано логічно-імітаційну модель виникнення травматизму, засновану на принципах побудови дерева відмов та аналізі можливих подій, які здатні призвести до травмонебезпечної ситуації.

Основним критерієм оцінки ризику обрано ймовірність виникнення травми під час виконання робіт. Аналіз виробничого процесу дав змогу ідентифікувати основні фактори, що можуть впливати на формування небезпечних ситуацій. До них належать недостатній контроль за дотриманням вимог охорони праці, несвоєчасне технічне обслуговування інструментів, відсутність окремих комплектуючих або засобів індивідуального захисту, робота на висоті чи поблизу обертових механізмів, застосування застарілого обладнання, потрапляння сторонніх предметів у пристрої, а також рівень професійної

підготовки працівника і його психофізіологічний стан на момент виконання завдань. Кожному із зазначених чинників було надано певне значення ймовірності, яке характеризує ймовірність його реалізації у реальних умовах.

Після визначення базових подій сформовано причинно-наслідкові залежності між ними. Використовуючи логічні оператори «і» та «або», послідовно побудовано дерево подій, у якому враховано етапи формування небезпечної ситуації: від виникнення передумов і технічних збоїв до безпосереднього контакту працівника з небезпечним фактором. Подальше використання ймовірнісних перетворень дозволило провести математичну оцінку рівня ризику. Наприклад, ймовірність формування однієї з проміжних подій, що характеризує порушення контролю та технічної справності, становила:

$$P_{10} = 1 - (1 - P1)(1 - P2) = 1 - (1 - 0.15)(1 - 0.10) = 0.235. \qquad (4.1)$$

а ймовірність комбінації відсутності засобів захисту та сторонніх предметів у зоні роботи системи була визначена як:

$$P_{11} = 1 - (1 - P6)(1 - P3) = 1 - (1 - 0.30)(1 - 0.12) = 0.366. \qquad (4.2)$$

Додатково були враховані рівень професійної підготовки виконавця і його досвід, що формують людський фактор. Після проведення усіх необхідних розрахунків інтегральна ймовірність травматизму під час монтажу системи склала приблизно – $P_v \approx 0.56$.

Це вказує на суттєвий потенційний ризик за відсутності організаційних та технічних заходів безпеки.

Отриманий результат не є свідченням невідворотності травм, а демонструє необхідність оптимізації умов праці та підвищення рівня безпеки при виконанні монтажних робіт. Найбільший внесок у загальний ризик формують неналежний технічний стан інструментів, відсутність окремих засобів захисту, людський фактор та зовнішні механічні впливи, що може бути характерним для польових умов аграрного виробництва. Відповідно, ефективними заходами зниження травмонебезпеки є підвищення контролю за дотриманням вимог охорони праці, впровадження регулярного технічного обслуговування обладнання, проведення

тренінгів для персоналу, а також забезпечення повного комплекту засобів індивідуального захисту. У разі реалізації таких заходів величина інтегральної ймовірності травматизму може бути зменшена у кілька разів, що підвищує загальний рівень безпеки виробничого середовища.

4.3. Розробка заходів щодо безпеки у надзвичайних ситуаціях

У процесі впровадження та подальшої експлуатації інтелектуальної системи підтримки прийняття рішень в аграрному виробництві робочий персонал може стикатися з надзвичайними ситуаціями різного характеру, зокрема техногенного, природного та технологічного походження. Враховуючи специфіку об’єкта, до потенційних ризиків належать аварійні зупинки обладнання, раптове знеструмлення, вихід із ладу серверної інфраструктури, займання електричних компонентів, несприятливі погодні умови, затоплення, пожежі та надзвичайні ситуації, пов’язані з експлуатацією дронів, сенсорних систем і електротехнічних установок. Тому розроблення ефективного комплексу заходів щодо запобігання та реагування на надзвичайні ситуації є важливою умовою сталого функціонування системи та забезпечення безпеки персоналу.

Основним елементом системи безпеки є створення чітко структурованого алгоритму дій працівників у разі виникнення аварійної ситуації. На першому етапі передбачено своєчасне виявлення та ідентифікацію потенційної загрози за допомогою інтелектуальних сенсорних компонентів, включених до складу системи моніторингу. Такі датчики можуть здійснювати контроль температури, рівня вологості, концентрації газів, напруги у силових лініях та функціонування мережевого обладнання, що дозволяє завчасно визначити відхилення від нормальних умов роботи. У випадку виявлення аномалій система повинна автоматично сповіщати відповідальних осіб через інформаційний інтерфейс або мобільні повідомлення, забезпечуючи оперативне реагування.

Особливе місце займає підготовка персоналу до дій у надзвичайних ситуаціях. Працівники повинні пройти інструктаж щодо правил поводження з електротехнічним обладнанням, протипожежними засобами, засобами першої допомоги та алгоритмами евакуації. Також важливим є проведення тренувальних заходів з моделювання аварійних сценаріїв, що дозволяє сформувати навички швидкого реагування. Психологічна готовність персоналу до стресових ситуацій є додатковим фактором, який впливає на ефективність дій під час надзвичайних випадків.

Комплексність заходів з організації безпеки під час аварійних подій дозволяє значно знижувати рівень ризиків травматизму та матеріальних збитків, а також забезпечує безперервність виробничого циклу та сталу роботу інтелектуальної системи. Реалізація зазначених заходів створює передумови для формування безпечного виробничого середовища та підвищення рівня відповідальності персоналу, що є невід’ємною складовою сучасного аграрного підприємства, орієнтованого на цифрову трансформацію та підвищення ефективності управління.

РОЗДІЛ 5.
ЕКОНОМІЧНА ЕФЕКТИВНІСТЬ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПІДТРИМКИ ПРИЙНЯТТЯ РІШЕНЬ ДЛЯ ОПТИМІЗАЦІЇ СКЛАДОВИХ АГРАРНИХ ПРОЄКТІВ

Оцінювання економічної ефективності впровадження інтелектуальної системи підтримки прийняття рішень є необхідним етапом для обґрунтування доцільності її використання в аграрній галузі. Враховуючи, що система оптимізує витрати, підвищує якість прийняття управлінських рішень та скорочує час реалізації аграрних проєктів, економічний ефект розглядається як сукупний результат, отриманий за рахунок зниження витрат ресурсів і збільшення продуктивності.

Узагальнено інтегральний економічний ефект від впровадження системи можна подати у вигляді:

$$E = C_д - C_п + \Delta Q - \Delta T, \qquad (6.1)$$

де $C_д$ – сумарні витрати до впровадження системи, грн; $C_п$ – витрати після впровадження, грн; ΔQ – економічний ефект від підвищення якості управління, грн; ΔT – економія часу, переведена в грошовий еквівалент, грн.

Для комплексної оцінки використаємо показник чистої теперішньої вартості:

$$NPV = \sum_{t=1}^{n} \frac{CF_t}{(1+r)^t} - I, \qquad (6.2)$$

де CF_t – грошовий потік у період t, грн; r – ставка дисконту, %; I – інвестиційні витрати на впровадження системи, грн; n – тривалість проєкту, роки.

Під час розрахунків передбачено одноразові витрати на впровадження програмного забезпечення, придбання сенсорних модулів, серверного обладнання та навчання персоналу. Для оцінювання взято умовні, але

реалістичні значення, характерні для впровадження ІТ-систем у сільському господарстві.

Вартість системи – 280000 грн.

Щорічний економічний ефект від оптимізації витрат – 120000 грн.

Додатковий ефект від скорочення часу реалізації робіт – 35000 грн.

Зниження втрат через людський фактор – 18000 грн.

Ставка дисконту – 10%.

Період оцінювання – 3 роки

Сумарний річний економічний ефект становить:

$$CF = 120000 + 35000 + 18000 = 173000 \ грн.$$

Виконуємо розрахунок NPV:

$$NPV = \frac{173000}{(1+0.1)^1} + \frac{173000}{(1+0.1)^2} + \frac{173000}{(1+0.1)^3} - 280000.$$

$$NPV = 157272.7 + 142975.2 + 130886.6 - 280000 = 151134.5 грн.$$

Таким чином, проєкт є економічно ефективним, оскільки:

$$NPV > 0.$$

Вмконуємо розрахунок періоду окупності:

$$T_{ок} = \frac{I}{CF} = \frac{280000}{173000} \approx 1.62 \ років.$$

Це означає, що система окупиться менш ніж за два роки експлуатації.

Таблиця 5.1 – Економічні показники впровадження ІСППР

Показник	Значення
Початкові витрати, грн	280000
Річний економічний ефект, грн	173000
Ставка дисконту	10%
Період оцінки	3 роки
NPV, грн	151134,5
Період окупності, років	1,62
Загальна економія витрат за 3 роки, грн	519000

Результати розрахунків свідчать, що впровадження інтелектуальної системи забезпечує суттєвий економічний ефект, перевищуючи початкові інвестиції. Чиста теперішня вартість становить 151,1 тис. грн, що підтверджує економічну доцільність реалізації проєкту. Період окупності в 1,62 року є сприятливим для аграрних підприємств, а загальний чистий прибуток за три роки перевищує 500 тис. грн. Крім того, слід відзначити нематеріальні вигоди – підвищення точності планування, скорочення ризиків, цифровізація управління та зростання конкурентоспроможності господарства.

ВИСНОВКИ І ПРОПОЗИЦІЇ

Проведене дослідження дозволило узагальнити сучасні наукові підходи до управління аграрними проєктами в умовах високої невизначеності та ризиків. Було показано, що класичні методи управління, які базуються на статистичних моделях та експертних оцінках, не забезпечують достатнього рівня точності та гнучкості при зростанні складності та багатофакторності середовища. Саме тому актуальним є впровадження інтелектуальних систем підтримки прийняття рішень, що поєднують методи нечіткої логіки та еволюційної оптимізації.

Нами окреслено основні чинники невизначеності аграрних проєктів, серед яких домінують кліматичні, економічні, технологічні та соціально-політичні ризики. Встановлено, що нечітка логіка дозволяє формалізувати якісні та лінгвістичні параметри, перетворюючи їх у числові оцінки, придатні для подальшої обробки. Генетичні алгоритми, у свою чергу, забезпечують глобальний пошук оптимальних рішень у багатовимірних просторах, що особливо важливо при багатокритеріальній оптимізації витрат, часу та якості.

У подальшому пропонується інтеграція обох підходів, що дозволяє мінімізувати базу правил нечітких систем і водночас підвищити точність оцінювання ефективності проєктів, що підтверджується сучасними дослідженнями.

Виконаний аналіз довів доцільність побудови комбінованої моделі, де нечітка логіка виступає засобом опису невизначених параметрів, а генетичний алгоритм оптимізує структуру та параметри системи. Розроблений підхід формує основу для створення інтелектуальної системи підтримки прийняття рішень, що здатна забезпечити баланс між економічною ефективністю та екологічною стійкістю аграрних проєктів.

На основі отриманих результатів можна сформулювати такі пропозиції. По-перше, доцільним є впровадження інтегрованих СППР в аграрні підприємства та органи управління для зменшення впливу ризиків та покращення прогнозування. По-друге, необхідно розробляти програмні

прототипи систем, що враховують локальні дані, специфіку культур і регіональні особливості, оскільки універсальні моделі можуть втрачати точність. По-третє, перспективним є поєднання методів нечіткої логіки та генетичних алгоритмів із сучасними технологіями машинного навчання та великих даних, що дозволить адаптувати системи до нових викликів і забезпечить ще вищий рівень надійності результатів.

Обраний інструментарій MATLAB та Python забезпечує ефективне поєднання можливостей прототипування нечітких моделей та масштабованості при реалізації генетичних алгоритмів. Це дозволяє створити гнучку, адаптивну та розширювану платформу для підтримки прийняття рішень у аграрних проєктах.

Розроблена архітектура інтегрованої СППР поєднує логічні й еволюційні методи в єдиній моделі, забезпечуючи адаптивність, здатність працювати з невизначеністю та можливість подальшої масштабованості. Структура системи гарантує всебічний аналіз аграрних параметрів і формування обґрунтованих управлінських рішень.

Формалізація параметрів аграрних проєктів є основним етапом розробки системи підтримки прийняття рішень. Створення адекватних функцій належності та бази нечітких правил дозволяє коректно моделювати невизначені умови, характерні для аграрного виробництва. Обрана структура змінних і функцій забезпечує точність, інтерпретованість та подальшу можливість оптимізації параметрів за допомогою генетичного алгоритму.

Багатокритеріальна постановка задачі оптимізації створює гнучку основу для прийняття рішень у складних умовах аграрного виробництва. Поєднання нечіткої логіки та генетичних алгоритмів дозволяє забезпечити адаптивне налаштування параметрів і досягнути збалансованого результату між витратами, часом і якістю, що є критично важливим для забезпечення конкурентоспроможності аграрних проєктів.

Розроблений модуль нечіткої логіки формує адаптивну оцінку ефективності аграрного проєкту, базуючись на гнучкому моделюванні

експертних знань та здатності враховувати невизначеність, що характерна для аграрного сектору. Створена архітектура нечіткої системи легко масштабується, дозволяє додавати нові змінні та оптимізувати функції належності на наступному етапі – за допомогою генетичного алгоритму, що буде описано у наступних підрозділах.

Фінальна реалізація модуля підтвердила здатність генетичного алгоритму знаходити збалансовані рішення у багатовимірному просторі параметрів. Зокрема, модель продемонструвала можливість досягнення раціональних комбінацій витрат, тривалості та ресурсного забезпечення при збереженні прийнятного рівня якості. Такий підхід створює основу для подальшого використання інтегрованої Fuzzy–GA системи у практиці аграрного управління, зокрема в процесах планування технологічних операцій, оптимізації витрат і вибору стратегій інвестування.

Інтеграція нечітких моделей і генетичних алгоритмів у межах одного інструменту показала високу ефективність у задачах оптимізації аграрних параметрів. Важливою перевагою є здатність системи працювати з неповною інформацією та нечіткими оцінками, що є типовою ситуацією у практиці агровиробництва. Реалізована архітектура дозволяє модифікувати правила нечіткої логіки та параметри генетичної оптимізації без зміни загальної структури коду, що відкриває можливості для подальшого розширення та адаптації під різні сценарії аграрного планування.

Розроблений інтерфейс користувача дозволяє швидко змінювати параметри оптимізації, експериментувати з нечіткими правилами та порівнювати різні сценарії аграрних рішень. Застосована логіка побудови UI створює умови для масштабування системи; зокрема, передбачено можливість підключення блоків геопросторової візуалізації, моделей оцінки врожайності та модулів прогнозування на основі глибоких нейронних мереж.

Тестування підтвердило правильність математичного апарату системи та коректність її роботи у різних експериментальних сценаріях. Модель стабільно досягає покращення метрик ефективності і демонструє очікувану поведінку

навіть при значних варіаціях параметрів, що свідчить про її практичну цінність та застосовність у процесах аграрного планування й підтримки прийняття рішень.

Розроблені заходи з охорони праці спрямовані на підвищення рівня безпеки виконання робіт, зменшення впливу шкідливих та небезпечних виробничих факторів і створення комфортних умов праці. Запропоновані рішення дозволяють мінімізувати ризики виникнення нещасних випадків, підвищити культуру безпеки на робочих місцях та сформувати відповідальне ставлення персоналу до дотримання вимог охорони праці.

Результати розрахунків свідчать, що впровадження інтелектуальної системи забезпечує суттєвий економічний ефект, перевищуючи початкові інвестиції. Чиста теперішня вартість становить 151,1 тис. грн, що підтверджує економічну доцільність реалізації проєкту. Період окупності в 1,62 року є сприятливим для аграрних підприємств, а загальний чистий прибуток за три роки перевищує 500 тис. грн.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Боднарчук О.В., Андрієнко А.В. Методи та засоби побудови інтелектуальних інформаційних систем. Київ: КНУ ім. Тараса Шевченка, 2021. 312 с.

2. Жидецький В.Ц., Джигирей В.С., Мельников О.В. Основи охорони праці. Підручник. Вид. 5-е, доповнене. Львів: Афіша, 2012. 350с.

3. Ідентифікація і класифікація ризиків. URL: https://qualityexpert.com.ua/articles/657215-identyfikatsiya-i-klasyfikatsiya-ryzykiv

4. Качмар П.М., Романюк А.М. Методологія проектування та оптимізації кіберфізичних систем в агросфері. Науковий вісник НУБіП України. Серія «Інформаційні технології», 2023, № 317, с. 112–121.

5. Класифікація в Python з Scikit-Learn та Pandas. URL: https://stackabuse.com/classification-in-pythonwith-scikit-learn-and-pandas/ (дата звернення: 14.09.2025).

6. Козак Ю.Г. Використання нейронних мереж у сучасних інформаційних системах. Сучасні інформаційні технології та системи, 2022, № 2, с. 45–54.

7. Кравець П.Ю., Кудінов Є.А. Інтелектуальні системи обробки інформації: теорія та практичні аспекти. Львів: Видавництво ЛНУ ім. І. Франка, 2020. 286 с.

8. Лехман С.Д., Рублев В.І., Рябцев Б.І. Запобігання аварійності і травматизму у сільському господарстві. К.: Урожай, 1993. 267 с.

9. Ситнік Б.Т. Основи інформаційних систем і технологій: навч. посіб. Харків: УкрДУЗТ, 2018. 130 с.

10. Тимченко Л.І., Дорошенко А.О. Системи підтримки прийняття рішень: теорія та практичні рішення. Харків: НТУ «ХПІ», 2019. 274 с.

11. Тригуба А., Маланчук О., Тригуба І., Мармуляк А., Демчина В. Андрушків О., Олійник Р. Вплив сучасних інформаційних технологій на процеси ініціації та планування проєктів розвитку громад та регіонів. Вісник Львівського

національного університету природокористування: агроінженерні дослідження. №24. Львів: Львів НУП, 2024. С. 123-130.

12. Aliev, R. A., Fazlollahi, B., & Guirimov, B. G. (2024). Combining fuzzy logic and genetic algorithms to optimize cost, time and quality in modern agriculture. Sustainability, 17(7), 2829. MDPI. https://doi.org/10.3390/su17072829

13. Almalki, S. S. (2025). AI-driven decision support systems in agile software project management: Enhancing risk mitigation and resource allocation. Systems, 13(3), 208. https://doi.org/10.3390/systems13030208

14. Al-Shammari, E. T., & Hussein, A. S. (2008). Analysis and modelling of hierarchical fuzzy logic systems. Journal of Information Technology Research, 1(4), 1–10. https://doi.org/10.4018/jitr.2008100101

15. Belarbi, K., Titel, F., Bourebia, W., & Benmahammed, K. (2005). Design of Mamdani fuzzy logic controllers with rule base minimisation using genetic algorithm. Engineering Applications of Artificial Intelligence, 18(8), 875–880. https://doi.org/10.1016/j.engappai.2005.01.007

16. Belarbi, K., Titel, F., Bourebia, W., & Benmahammed, K. (2005). Design of Mamdani fuzzy logic controllers with rule base minimisation using genetic algorithm. Engineering Applications of Artificial Intelligence, 18(8), 875–880. https://doi.org/10.1016/j.engappai.2005.01.007

17. Belarbi, K., Titel, F., Bourebia, W., & Benmahammed, K. (2005). Design of Mamdani fuzzy logic controllers with rule base minimisation using genetic algorithm. Engineering Applications of Artificial Intelligence, 18(8), 875–880. https://doi.org/10.1016/j.engappai.2005.01.007

18. Erdoğdu, A., Dayi, F., Yildiz, F., Yanik, A., & Ganji, F. (2025). Combining Fuzzy Logic and Genetic Algorithms to Optimize Cost, Time and Quality in Modern Agriculture. Sustainability, 17(7), 2829. https://doi.org/10.3390/su17072829

19. Erdoğdu, A., Dayi, F., Yildiz, F., Yanik, A., & Ganji, F. (2025). Combining fuzzy logic and genetic algorithms to optimize cost, time and quality in modern agriculture. Sustainability, 17(7), 2829. https://doi.org/10.3390/su17072829

20. Erdoğdu, A., Dayi, F., Yildiz, F., Yanik, A., & Ganji, F. (2025). Combining fuzzy logic and genetic algorithms to optimize cost, time and quality in modern agriculture. Sustainability, 17(7), 2829. https://doi.org/10.3390/su17072829

21. Houby, E. M. F., & Yassin, N. I. (2020). Wavelet-Hadamard based blind image watermarking using genetic algorithm and decision tree. Multimedia Tools and Applications, 79(5), 3437–3462. https://doi.org/10.1007/s11042-020-09333-3

22. Magalhães, B., Gaspar, P. D., Corceiro, A., João, L., & Bumba, C. (2022). Fuzzy logic decision support system to predict peaches marketable period at highest quality. Climate, 10(3), 29. https://doi.org/10.3390/cli10030029

23. Magalhães, B., Gaspar, P. D., Corceiro, A., João, L., & Bumba, C. (2022). Fuzzy logic decision support system to predict peaches marketable period at highest quality. Climate, 10(3), 29. https://doi.org/10.3390/cli10030029

24. MathWorks. (2020). Fuzzy Logic Toolbox™ User's Guide. Natick, MA: MathWorks. Available online: https://www.mathworks.com/help/fuzzy/

25. Tryhuba A. M., Sholudko P. V., Malanchuk O. V., Rudynets M. V. Production and technological risk formation in the integrated agricultural production programs // Eastern-European Journal of Enterprise Technologies. 2013. Vol. 1, Issue 10 (61). P. 203-206. DOI: 10.15587/1729-4061.2013.142227.

26. Tryhuba A. M., Sydorchuk O. V. Особливості планування проєктів та програм аграрного виробництва // Матер. VI Міжнар. конф. «Управління проєктами: стан та перспективи», Миколаїв, 2010. С. 313-316.

27. Tryhuba A., Komarnitskyi S., Tryhuba I., Hutsol T., Yermakov S., Muzychenko A., Muzychenko T., Horetska I. Planning and risk analysis in projects of procurement of agricultural raw materials for the production of environmentally friendly fuel. International Journal of Renewable Energy Development. 2022. Vol. 11, No. 2. P. 569-580. DOI: 10.14710/ijred.2022.43011.

28. Tryhuba A., Koval N., Tryhuba I., Padiuka R., Boiarchuk O. System model of the digital transformation of rural territorial communities based on computational intelligence. Bulletin of Lviv National Environmental University. Series

«Agroengineering Research». 2022. No. 26. P. 177-184. DOI: 10.31734/agroengineering2022.26.177.

29. Tryhuba A., Ratushny R., Tryhuba I., Koval N., Andrushkiv O. Impact of modern information technologies on the processes of initiating and planning the community and regional development projects. Bulletin of Lviv National Environmental University. Series «Agroengineering Research». 2024. No. 28. P. 148-158. DOI: 10.31734/agroengineering2024.28.148.

30. Tryhuba A., Tryhuba I., Chubyk R., Kondysiuk I., Koval N., Paniura Ya. Forecasting the volume of raw materials procurement on the territory of communities using artificial neural networks. Bulletin of Lviv National Environmental University. Series «Agroengineering Research». 2020. No. 24. P. 143-151. DOI: 10.31734/agroengineering2020.24.143.

31. Zadeh, L. A. (1965). Fuzzy sets. Information and Control, 8(3), 338–353. https://doi.org/10.1016/S0019-9958(65)90241-X

32. Zadeh, L. A. (1968). Fuzzy algorithms. Information and Control, 12(2), 94–102. https://doi.org/10.1016/S0019-9958(68)90211-8

ДОДАТКИ

Додаток А.1

Фрагмент коду створеної ІСППР

```
# -*- coding: utf-8 -*-
import os
import csv
import json
import math
import random
import datetime as dt
from dataclasses import dataclass
from typing import List, Dict, Tuple, Callable

import numpy as np
try:
    import pandas as pd
    PANDAS_OK = True
except Exception:
    PANDAS_OK = False

import tkinter as tk
from tkinter import ttk, filedialog, messagebox

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.figure import Figure

# ============================================================
# Нечіткі множини та FIS
# ============================================================

def tri_mf(x: np.ndarray, a: float, b: float, c: float) -> np.ndarray:
    """Трикутна функція належності."""
    y = np.zeros_like(x, dtype=float)
    left = (a < x) & (x <= b)
    right = (b < x) & (x < c)
    if b != a:
        y[left] = (x[left] - a) / (b - a)
    if c != b:
        y[right] = (c - x[right]) / (c - b)
    y[x == b] = 1.0
    return np.clip(y, 0, 1)


def trap_mf(x: np.ndarray, a: float, b: float, c: float, d: float) -> np.ndarray:
    """Трапецоїдна функція належності."""
    y = np.zeros_like(x, dtype=float)
    rising = (a < x) & (x <= b)
    top = (b < x) & (x < c)
    falling = (c <= x) & (x < d)
    if b != a:
        y[rising] = (x[rising] - a) / (b - a)
```

```
    y[top] = 1.0
    if d != c:
        y[falling] = (d - x[falling]) / (d - c)
    y[(x == b) | (x == c)] = 1.0
    return np.clip(y, 0, 1)


….


    def eval_point(self, x_dict: Dict[str, float]) -> float:
        # Правила: агрегація через min/max
        aggregated = np.zeros_like(self.x_out)
        fuzz_in = self.fuzzify_point(x_dict)
        for rule in self.rules:
            degs = []
            for (vname, tname) in rule.antecedents:
                degs.append(fuzz_in[vname][tname])
            firing = min(degs) if degs else 0.0
            out_term = self.output.terms[rule.consequent[1]]
            mu_out = out_term.mu(self.x_out)
            aggregated = np.maximum(aggregated, np.minimum(firing, mu_out))
        # Дефазифікація (центр ваги)
        area = np.trapz(aggregated, self.x_out)
        if area <= 1e-12:
            return float((self.output.universe[0] + self.output.universe[1]) / 2)
        centroid = float(np.trapz(self.x_out * aggregated, self.x_out) / area)
        return centroid

# ============================================================
# Генетичний алгоритм (простий, без залежностей)
# ============================================================

class SimpleGA:
    def __init__(self, dim: int, bounds: List[Tuple[float, float]], fitness: Callable[[np.ndarray], float],
                 pop_size: int = 40, cx_rate: float = 0.9, mut_rate: float = 0.1, sigma: float = 0.05,
                 max_gen: int = 80, seed: int = 42):
        random.seed(seed)
        np.random.seed(seed)
        self.dim = dim
        self.bounds = np.array(bounds, dtype=float)
        self.fitness = fitness
        self.pop_size = pop_size
        self.cx_rate = cx_rate
        self.mut_rate = mut_rate
        self.sigma = sigma
        self.max_gen = max_gen

    def _clip(self, x: np.ndarray) -> np.ndarray:
        lo = self.bounds[:, 0]
        hi = self.bounds[:, 1]
        return np.minimum(np.maximum(x, lo), hi)
```

```
def init_pop(self) -> np.ndarray:
    lo = self.bounds[:, 0]
    hi = self.bounds[:, 1]
    return lo + (hi - lo) * np.random.rand(self.pop_size, self.dim)

def tournament(self, pop: np.ndarray, fits: np.ndarray, k: int = 3) -> np.ndarray:
    idx = np.random.randint(0, len(pop), size=(k,))
    best = idx[0]
    for j in idx[1:]:
        if fits[j] > fits[best]:
            best = j
    return pop[best].copy()

def crossover(self, p1: np.ndarray, p2: np.ndarray) -> Tuple[np.ndarray, np.ndarray]:
    if np.random.rand() > self.cx_rate:
        return p1.copy(), p2.copy()
    alpha = np.random.rand(self.dim)
    c1 = alpha * p1 + (1 - alpha) * p2
    c2 = alpha * p2 + (1 - alpha) * p1
    return self._clip(c1), self._clip(c2)

def mutate(self, x: np.ndarray) -> np.ndarray:
    if np.random.rand() < self.mut_rate:
        noise = self.sigma * (self.bounds[:, 1] - self.bounds[:, 0]) * np.random.randn(self.dim)
        x = x + noise
    return self._clip(x)

def run(self) -> Tuple[np.ndarray, float, List[float]]:
    pop = self.init_pop()
    fits = np.array([self.fitness(ind) for ind in pop])
    history = [float(fits.max())]
    best_idx = int(fits.argmax())
    best = pop[best_idx].copy()
    best_fit = float(fits[best_idx])
    for g in range(self.max_gen):
        new_pop = []
        while len(new_pop) < self.pop_size:
            p1 = self.tournament(pop, fits)
            p2 = self.tournament(pop, fits)
            c1, c2 = self.crossover(p1, p2)
            c1 = self.mutate(c1)
            c2 = self.mutate(c2)
            new_pop.extend([c1, c2])
        pop = np.array(new_pop[:self.pop_size])
        fits = np.array([self.fitness(ind) for ind in pop])
        if fits.max() > best_fit:
            best_idx = int(fits.argmax())
            best = pop[best_idx].copy()
            best_fit = float(fits[best_idx])
        history.append(float(fits.max()))
    return best, best_fit, history
```

```
# ============================================================
# Головне вікно (UI)
# ============================================================

class DSSApp(tk.Tk):
    def __init__(self):
        super().__init__()
        self.title("Інтелектуальна СППР для аграрних проєктів (Fuzzy + GA)")
        self.geometry("1180x760")
        self.minsize(1024, 680)
        self.configure(bg="#f6f7fb")
        self._init_style()
        self._init_state_defaults()
        self._build_ui()

    # ----------------------------
    def _init_style(self):
        style = ttk.Style(self)
        try:
            style.theme_use('clam')
        except Exception:
            pass
        style.configure('TNotebook', background='#f6f7fb', borderwidth=0)
        style.configure('TNotebook.Tab', padding=(12, 8), font=('Segoe UI', 11))
        style.configure('TFrame', background='#ffffff')
        style.configure('Card.TFrame', background='#ffffff', relief='groove', borderwidth=1)
        style.configure('TLabel', background='#ffffff', font=('Segoe UI', 10))
        style.configure('H.TLabel', background='#f6f7fb', font=('Segoe UI', 16, 'bold'))
        style.configure('Sub.TLabel', background='#ffffff', font=('Segoe UI', 11, 'bold'))
        style.configure('TButton', font=('Segoe UI', 10, 'bold'))
        style.configure('Accent.TButton', font=('Segoe UI', 10, 'bold'))


….



        # Ваги CTQ
        self.w1 = tk.DoubleVar(value=0.33)  # витрати
        self.w2 = tk.DoubleVar(value=0.33)  # час
        self.w3 = tk.DoubleVar(value=0.34)  # якість

        # Для GA: межі змінних-кандидатів (C,T) – універсуми; Q оцінюється FIS
        self.ga_bounds = [(self.univ_C[0], self.univ_C[1]), (self.univ_T[0], self.univ_T[1])]
        self.ga_params = {
            'pop': tk.IntVar(value=40),
            'gen': tk.IntVar(value=60),
            'cx': tk.DoubleVar(value=0.9),
            'mut': tk.DoubleVar(value=0.15),
            'sigma': tk.DoubleVar(value=0.05),
        }
```

```
    # ----------------------------
    def _build_ui(self):
        title = ttk.Label(self, text='Інтелектуальна СППР для аграрних проєктів (нечітка логіка + генетичний алгоритм)', style='H.TLabel')
        title.pack(padx=16, pady=(12, 6), anchor='w')

        nb = ttk.Notebook(self)
        nb.pack(fill='both', expand=True, padx=12, pady=12)

        self.page_intro = ttk.Frame(nb, style='TFrame')
        self.page_inputs = ttk.Frame(nb, style='TFrame')
        self.page_fuzzy = ttk.Frame(nb, style='TFrame')
        self.page_opt = ttk.Frame(nb, style='TFrame')
        self.page_results = ttk.Frame(nb, style='TFrame')
        self.page_about = ttk.Frame(nb, style='TFrame')

        nb.add(self.page_intro, text='Початок')
        nb.add(self.page_inputs, text='Вхідні дані')
        nb.add(self.page_fuzzy, text='Нечітка модель')
        nb.add(self.page_opt, text='Оптимізація (GA)')
        nb.add(self.page_results, text='Результати')
        nb.add(self.page_about, text='Про систему')

        self._build_intro()
        self._build_inputs()
        self._build_fuzzy()
        self._build_opt()
        self._build_results()
        self._build_about()

    # ----------------------------
    def _card(self, parent, **grid):
        frame = ttk.Frame(parent, style='Card.TFrame')
        frame.grid(**grid)
        return frame

    # ----------------------------
    def _build_intro(self):
        f = self._card(self.page_intro, row=0, column=0, padx=12, pady=12, sticky='nsew')
        self.page_intro.rowconfigure(0, weight=1)
        self.page_intro.columnconfigure(0, weight=1)
        intro_text = (
        "Ласкаво просимо!"
        "Ця система поєднує нечітку логіку Мамдані та генетичний алгоритм для оптимізації ключових показників аграрних проєктів: витрат (C), тривалості (T) і якості (Q). "
        "Налаштуйте функції належності та базу правил, задайте ваги CTQ, і запустіть оптимізацію."
        )
        lbl = ttk.Label(f, text=intro_text, font=('Segoe UI', 13, 'bold'), wraplength=520, justify='left')
        lbl.grid(row=0, column=0, padx=16, pady=16, sticky='nw')
```

```
    # ---------------------------
    def _build_inputs(self):
        self.page_inputs.columnconfigure(0, weight=1)
        self.page_inputs.columnconfigure(1, weight=1)
        self.page_inputs.rowconfigure(1, weight=1)

        # Карта налаштування ваг
        card_w = self._card(self.page_inputs, row=0, column=0, padx=12, pady=12, sticky='nsew')
        ttk.Label(card_w, text='Ваги цільових критеріїв (w1, w2, w3), w1+w2+w3=1', style='Sub.TLabel').grid(row=0, column=0, columnspan=3, padx=12, pady=(12, 6), sticky='w')
        self._scale_with_entry(card_w, 'w1 (витрати)', self.w1, 0)
        self._scale_with_entry(card_w, 'w2 (час)', self.w2, 1)
        self._scale_with_entry(card_w, 'w3 (якість)', self.w3, 2)
        ttk.Button(card_w, text='Нормувати ваги', command=self._normalize_weights).grid(row=4, column=0, padx=12, pady=10, sticky='w')

        # Карта завантаження даних (необов’язкова)
        card_d = self._card(self.page_inputs, row=0, column=1, padx=12, pady=12, sticky='nsew')
        ttk.Label(card_d, text='Дані (опційно): CSV з колонками C,T (0–100; 0–200)', style='Sub.TLabel').grid(row=0, column=0, padx=12, pady=(12,6), sticky='w')
        ttk.Button(card_d, text='Завантажити CSV', command=self._load_csv).grid(row=1, column=0, padx=12, pady=8, sticky='w')
        self.data_preview = tk.Text(card_d, height=10, width=50)
        self.data_preview.grid(row=2, column=0, padx=12, pady=8, sticky='nsew')
        card_d.rowconfigure(2, weight=1)
        card_d.columnconfigure(0, weight=1)

    def _scale_with_entry(self, parent, label, var, row):
        ttk.Label(parent, text=label).grid(row=row+1, column=0, padx=12, pady=6, sticky='w')
        s = ttk.Scale(parent, variable=var, from_=0.0, to=1.0, orient='horizontal', length=220)
        s.grid(row=row+1, column=1, padx=12, pady=6, sticky='w')
        e = ttk.Entry(parent, textvariable=var, width=8)
        e.grid(row=row+1, column=2, padx=12, pady=6, sticky='w')

    def _normalize_weights(self):
        s = self.w1.get() + self.w2.get() + self.w3.get()
        if s <= 1e-9:
            messagebox.showwarning('Ваги', 'Сума ваг дорівнює нулю; встановлено рівні ваги.')
            self.w1.set(1/3); self.w2.set(1/3); self.w3.set(1/3)
        else:
            self.w1.set(self.w1.get()/s)
            self.w2.set(self.w2.get()/s)
            self.w3.set(self.w3.get()/s)

    def _load_csv(self):
        path = filedialog.askopenfilename(title='Обрати CSV', filetypes=[('CSV','*.csv')])
        if not path:
            return
        try:
            rows = []
            with open(path, 'r', newline='', encoding='utf-8') as f:
```

```
                reader = csv.DictReader(f)
                for r in reader:
                    rows.append(r)
            self.data_preview.delete('1.0', 'end')
            self.data_preview.insert('end', f"Файл: {os.path.basename(path)}\n")
            for r in rows[:20]:
                self.data_preview.insert('end', json.dumps(r, ensure_ascii=False) + "\n")
        except Exception as ex:
            messagebox.showerror('CSV', f'Помилка читання CSV: {ex}')

    # ----------------------------
    def _build_fuzzy(self):
        self.page_fuzzy.columnconfigure(0, weight=1)
        self.page_fuzzy.columnconfigure(1, weight=1)
        self.page_fuzzy.rowconfigure(1, weight=1)

        card_left = self._card(self.page_fuzzy, row=0, column=0, padx=12, pady=12, sticky='nsew')
        ttk.Label(card_left, text='Функції належності (витрати C, час T, якість Q)', style='Sub.TLabel').grid(row=0, column=0, padx=12, pady=(12,8), sticky='w')

        # Пологра – matplotlib
        fig = Figure(figsize=(6.2, 4.2), dpi=100)
        self.axC = fig.add_subplot(131)
        self.axT = fig.add_subplot(132)
        self.axQ = fig.add_subplot(133)
        fig.tight_layout()
        self.canvas = FigureCanvasTkAgg(fig, master=card_left)
        self.canvas.get_tk_widget().grid(row=1, column=0, padx=8, pady=8, sticky='nsew')
        card_left.rowconfigure(1, weight=1)

        ttk.Button(card_left, text='Намалювати MF', command=self._plot_mfs).grid(row=2, column=0, padx=12, pady=8, sticky='w')

        # Праворуч – тест FIS
        card_right = self._card(self.page_fuzzy, row=0, column=1, padx=12, pady=12, sticky='nsew')
        ttk.Label(card_right, text='Тест оцінки FIS (введіть C,T)', style='Sub.TLabel').grid(row=0, column=0, columnspan=2, padx=12, pady=(12,8), sticky='w')
        self.varC = tk.DoubleVar(value=30.0)
        self.varT = tk.DoubleVar(value=80.0)
        ttk.Label(card_right, text='C (0..100)').grid(row=1, column=0, padx=12, pady=6, sticky='w')
        ttk.Entry(card_right, textvariable=self.varC, width=10).grid(row=1, column=1, padx=12, pady=6, sticky='w')
        ttk.Label(card_right, text='T (0..200)').grid(row=2, column=0, padx=12, pady=6, sticky='w')
        ttk.Entry(card_right, textvariable=self.varT, width=10).grid(row=2, column=1, padx=12, pady=6, sticky='w')
        ttk.Button(card_right, text='Обчислити Q', command=self._eval_fis).grid(row=3, column=0, padx=12, pady=8, sticky='w')
        self.lblQ = ttk.Label(card_right, text='Q = ?')
        self.lblQ.grid(row=3, column=1, padx=12, pady=8, sticky='w')

    def _plot_mfs(self):
        # Готуємо змінні
```

```
        C = FuzzyVariable('C', self.univ_C, self.terms_C)
        T = FuzzyVariable('T', self.univ_T, self.terms_T)
        Q = FuzzyVariable('Q', self.univ_Q, self.terms_Q)
        xC = np.linspace(*self.univ_C, 801)
        xT = np.linspace(*self.univ_T, 801)
        xQ = np.linspace(*self.univ_Q, 801)
        # Чистимо осі
        for ax in (self.axC, self.axT, self.axQ):
            ax.clear()
            ax.grid(True, linestyle='--', linewidth=0.6, alpha=0.7)
        # C
        for t in C.terms.values():
            self.axC.plot(xC, t.mu(xC), label=t.name)
        self.axC.set_title('C (витрати)')
        self.axC.set_xlim(*self.univ_C); self.axC.set_ylim(-0.05,1.05)
        # T
        for t in T.terms.values():
            self.axT.plot(xT, t.mu(xT), label=t.name)
        self.axT.set_title('T (час)')
        self.axT.set_xlim(*self.univ_T); self.axT.set_ylim(-0.05,1.05)
        # Q
        for t in Q.terms.values():
            self.axQ.plot(xQ, t.mu(xQ), label=t.name)
        self.axQ.set_title('Q (якість)')
        self.axQ.set_xlim(*self.univ_Q); self.axQ.set_ylim(-0.05,1.05)
        # Легенда тільки під останнім
        self.axQ.legend(loc='lower right', fontsize=8)
        self.canvas.draw_idle()
```

…

```
        # Ваги
        w1, w2, w3 = self.w1.get(), self.w2.get(), self.w3.get()
        s = w1 + w2 + w3
        if s <= 1e-9:
            w1 = w2 = w3 = 1/3
        else:
            w1, w2, w3 = w1/s, w2/s, w3/s

        def fitness(ind: np.ndarray) -> float:
            C = float(ind[0])
            T = float(ind[1])
            Q = float(fis.eval_point({'C':C, 'T':T}))
            # CTQ агрегування
            # Уникаємо ділення на нуль, додаємо 1e-6
            val = w1*(1.0/(C+1e-6)) + w2*(1.0/(T+1e-6)) + w3*(Q)
            return float(val)

        # Межі GA
        bounds = [(self.univ_C[0], self.univ_C[1]), (self.univ_T[0], self.univ_T[1])]
```

```
        ga = SimpleGA(dim=2, bounds=bounds, fitness=fitness,
                      pop_size=self.ga_params['pop'].get(),
                      cx_rate=self.ga_params['cx'].get(),
                      mut_rate=self.ga_params['mut'].get(),
                      sigma=self.ga_params['sigma'].get(),
                      max_gen=self.ga_params['gen'].get())
        best, best_fit, history = ga.run()

        # Відображення історії
        self.ax_fit.clear()
        self.ax_fit.grid(True, linestyle='--', linewidth=0.6, alpha=0.7)
        self.ax_fit.plot(history)
        self.ax_fit.set_xlabel('Покоління')
        self.ax_fit.set_ylabel('Найкраще F(p)')
        self.canvas_fit.draw_idle()

        # Лог
        self.txt_log.delete('1.0', 'end')
        self.txt_log.insert('end', f"Найкраще рішення: C={best[0]:.2f}, T={best[1]:.2f}\n")
        qbest = fis.eval_point({'C':float(best[0]), 'T':float(best[1])})
        self.txt_log.insert('end', f"Очінка FIS (Q)={qbest:.4f}; Фітнес={best_fit:.6f}\n")
        self.txt_log.insert('end', f"Ваги: w1={w1:.2f}, w2={w2:.2f}, w3={w3:.2f}\n")
        self.best_solution = {'C': float(best[0]), 'T': float(best[1]), 'Q': float(qbest), 'Fitness':
float(best_fit)}

    # ----------------------------
    def _build_results(self):
        self.page_results.columnconfigure(0, weight=1)
        self.page_results.rowconfigure(1, weight=1)
        card = self._card(self.page_results, row=0, column=0, padx=12, pady=12, sticky='nsew')
        ttk.Label(card, text='Експорт та збереження результатів', style='Sub.TLabel').grid(row=0,
column=0, padx=12, pady=(12,8), sticky='w')
        ttk.Button(card, text='Експортувати найкраще рішення у CSV',
command=self._export_csv).grid(row=1, column=0, padx=12, pady=8, sticky='w')
        ttk.Button(card, text='Зберегти графіки MF (PNG)',
command=self._save_mf_png).grid(row=1, column=1, padx=12, pady=8, sticky='w')
        self.txt_res = tk.Text(card, height=18)
        self.txt_res.grid(row=2, column=0, columnspan=2, padx=8, pady=8, sticky='nsew')
        card.rowconfigure(2, weight=1)

    def _export_csv(self):
        if not hasattr(self, 'best_solution'):
            messagebox.showinfo('Експорт', 'Спочатку запустіть оптимізацію (вкладка
Оптимізація).')
            return
        path = filedialog.asksaveasfilename(title='Зберегти CSV', defaultextension='.csv',
filetypes=[('CSV','*.csv')])
        if not path:
            return
        try:
            with open(path, 'w', newline='', encoding='utf-8') as f:
                w = csv.DictWriter(f, fieldnames=list(self.best_solution.keys()))
```

```
            w.writeheader()
            w.writerow(self.best_solution)
        self.txt_res.insert('end', f"Експортовано у {path}\n")
      except Exception as ex:
        messagebox.showerror('CSV', f'Помилка запису CSV: {ex}')

  def _save_mf_png(self):
    # Перемалювати, якщо ще не малювали
    self._plot_mfs()
    path = filedialog.asksaveasfilename(title='Зберегти PNG', defaultextension='.png', filetypes=[('PNG','*.png')])
    if not path:
      return
    try:
      self.canvas.figure.savefig(path, dpi=180)
      self.txt_res.insert('end', f"Збережено графіки MF у {path}\n")
    except Exception as ex:
      messagebox.showerror('PNG', f'Помилка збереження: {ex}')

  # ----------------------------
  def _build_about(self):
    f = self._card(self.page_about, row=0, column=0, padx=12, pady=12, sticky='nsew')
    txt = (
      "Версія: 1.0\n"
      "Архітектура: FIS (Мамдані) + простий власний GA.\n"
      "Налаштування: ваги w1,w2,w3; MF для C,T,Q; параметри GA (pop, gen, cx, mut, sigma).\n"
      "Експорт: CSV результатів, PNG графіків.\n"
    )
    ttk.Label(f, text=txt, justify='left').grid(row=0, column=0, padx=16, pady=16, sticky='w')

#============================================================
# Запуск
# ============================================================

def main():
  app = DSSApp()
  app.mainloop()

if __name__ == '__main__':
  main()
```