МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Комп'ютерних наук
(повна назва)

Кафедра Програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

другий (магістерський)
(рівень вищої освіти)

Дослідження способів застосування генетичних алгоритмів для визначення життєздатності рухомих об'єктів та їх систем
(тема)

Виконав:

студент II курсу, групи ІПЗм-21-1

Кириченко О.В.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-наукова

Керівник доц. Каук В.І.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ З.В. Дудар

(підпис)

2023р.

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Факультет Комп'ютерних наук

Кафедра Програмної інженерії

Рівень вищої освіти другий (магістерський)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва)

Тип програми освітньо-наукова програма

Освітня програма Інженерія програмного забезпечення
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)

«_____»_____ 202__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові *Кириченку Олегу Володимировичу*
(прізвище, ім'я, по батькові)

1. Тема роботи *«Дослідження способів застосування генетичних алгоритмів для визначення життєздатності рухомих об'єктів та їх систем»*

затверджена наказом університету від *«29» березня 2023 р. № 302 Ст*

2. Термін подання студентом роботи до екзаменаційної комісії *« 19 » травня 2023 р.*

3. Вихідні дані до роботи: *Генетичні алгоритми, еволюційні системи, прикладне застосування генетичних алгоритмів у контексті життєздатності рухомих об'єктів та їх систем, програмна реалізація, пояснювальна записка.*

4. Перелік питань, що потрібно опрацювати в роботі: *мета роботи, аналіз предметної галузі, постановка задачі, генетичні алгоритми, методи виявлення найбільш життєздатних конфігурацій рухомих об'єктів та їх систем, дослідження способів застосування генетичних алгоритмів для виявлення найбільш життєздатних конфігурацій рухомих об'єктів та їх систем, формальний опис постанови оптимізаційної задачі, приклади практичного застосування, універсальна програмна реалізація.*

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Формулювання мети роботи, науково-технічної задачі	*23.01.2023 – 29.01.2023*	*Виконано*
2	Аналіз стану вирішення проблеми, обґрунтування мети дослідження	*30.01.2023 – 05.02.2023*	*Виконано*
3	Постановка задачі	*06.02.2023 – 12.02.2023*	*Виконано*
4	Планування експериментальної частини дослідження	*13.02.2023 – 19.02.2023*	*Виконано*
5	Розробка методики дослідження	*20.02.2023 – 26.02.2023*	*Виконано*
6	Проектування програмної системи	*20.02.2023 – 05.03.2023*	*Виконано*
7	Розробка програмної системи	*06.03.2023 – 09.04.2023*	*Виконано*
8	Науковий аналіз	*10.04.2023 – 23.04.2023*	*Виконано*
9	Апробація отриманих результатів	*24.04.2023 – 30.04.2023*	*Виконано*
10	Підготовка звіту	*01.05.2023 – 17.05.2023*	*Виконано*
11	Захист	*20.05.2023*	*Виконано*

Дата видачі завдання *« 23 »* *січня* *2023 р.*

Студент _____
(підпис)

Керівник роботи _____ доц. Каук В. І.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи магістра, 88 с., 6 ч., 49 рис., 6 табл., 5 додатків, 13 джерел.

ВІЗУАЛІЗАЦІЯ, ГЕНЕТИЧНИЙ АЛГОРИТМ, ЕВОЛЮЦІЯ, ЗОВНІШНЄ СЕРЕДОВИЩЕ, НЕЙРОННА МЕРЕЖА, ОПТИМІЗАЦІЙНА ЗАДАЧА, РЕЦЕПТОР, РУХОМИЙ ОБ'ЄКТ.

Об'єктом дослідження є рухомі об'єкти та їх системи, для яких актуальна задача оптимізації поведінки, розв'язки якої мають визначати їх поведінку та характеристики, враховуючи задані обмеження.

Предметом дослідження є способи застосування гентичних алгоритмів для визначення життєздатності рухомих об'єктів та їх систем, оптимізації їх конфігурації та поведінки.

Об'єктом розробки є програмна система для розв'язання задної оптимізаційної задачі із можливостями задання умов задачі у формальний і одночасно наочний спосіб та візуалізації процесу навчання.

Метою дослідження є виявлення ефективних способів використання генетичних алгоритмів для визначення життєздатності рухомих об'єктів та їх систем, формалізація та узагальнення задачі оптимізації їх конфігурації та поведінки за допомогою генетичного алгоритму; створення програмної системи, яка б дозволяла задати умови задачі такого типу у формальній і одночасно наочній формі, розв'язати цю задачу та візуалізувати проміжні та кінцеві результати.

Методи розробки базуються на рушії Unity та бібліотеці GeneticSharp.

У результаті роботи було формалізовано оптимізаційну задачу поведінки рухомих об'єктів та їх систем задля досягнення найвищого рівня життєздатності та якості виконання поставлених цілей; створено програмну систему, що дозволяє задавати умови задачі такого типу у формальній і одночасно наочній формі, розв'язувати цю задачу та візуалізувати проміжні та кінцеві результати.

VISUALIZATION, GENETIC ALGORITHM, EVOLUTION, EXTERNAL ENVIRONMENT, NEURAL NETWORK, OPTIMIZATION PROBLEM, RECEPTOR, MOVING OBJECT.

The object of research is moving objects and their systems whose behavior and characteristics should be determined by solving some optimization problem, taking into account given restrictions.

The subject of research is methods of applying genetic algorithm to determine the viability of moving objects and their systems, optimizing their configuration and behavior.

The object of development is a software system for solving such optimization problems, providing an ability to set the conditions of the problem in a formal and at the same time visual way and visualize the process of an experiment.

The purpose of the research is to identify effective methods of using genetic algorithms to determine the viability of moving objects and their systems, formalize and generalize the problem of optimizing their configuration and behavior using a genetic algorithm; development of a software system that would allow setting the conditions of a problem of this type in a formal and at the same time visual form, solving this problem and visualizing intermediate and final results.

Development methods are based on Unity engine and GeneticSharp library.

As a result of the work, the optimization problem of the behavior of moving objects and their systems was formalized in order to achieve the highest level of viability and quality of the specified goals performance; a software system, which allows user to set the conditions of a problem in a formal and at the same time visual form, solve this problem and visualize intermediate and final results, was created.

Я, Кириченко Олег Володимирович, студент групи ІПЗм-21-1, здобувач вищої освіти на другому (магістерському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Дослідження способів застосування генетичних алгоритмів для визначення життєздатності рухомих об’єктів та їх систем», що буде представлена до ЕК для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElArKhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

НМРО – нейронна мережа розумової частини генома особини;

НМД – нейронна мережа елемента-датчика;

СМТР – суматор.

ВСТУП

Кожен рік інженери із різних галузей кидають нові виклики фізиці: розроблюють новітні моделі літаків, автомобілей, кораблів тощо. Натомість живі організми почали цей шлях ще задовго до появи інженерів та людини як такої. Еволюція – дуже тривалий механізм розвитку та пристосування, проте дуже результативний, про що свідчить людське існування. Науковці та конструктори часто позичають “ідеї” матінки-природи, пристосовуючи їх до своїх потреб.

Наразі такими запозиченнями зумовлені фізичні характеристики деяких винаходів: форма крил літаків, підводних човнів, принцип роботи гвинтокрила або пристроя нічного бачення. Поведінка ж таких засобів повністю або майже повністю керована людиною-оператором. Враховуючи напрямок розвитку найновітніших технологій (автоматизованих марсоходів, автомобілей Tesla тощо), можна зробити висновок, що в майбутньому для більшості таких пристроїв наявність оператора буде необов’язковою.

Та чи можливо дослідити поведінку природних аналогів та відтворити її для відповідного механізму? Зовсім не завжди, адже для цього необхідні дуже глибокі природничі дослідження, провести які у стислі терміни майже неможливо, а їх результати не завжди знайдуть бажане відображення в технічній галузі. Проте існує механізм синтетичної еволюції – генетичний алгоритм, – який можна пристосувати до дуже великої кількості задач.

Перевагою генетичного алгоритму перед іншими методами навчання моделей є природність та відсутність необхідності навчальних даних, адже модифікація генома особин відбувається шляхом мутації та схрещування, а оптимізація – шляхом селекції. Таким чином, множина задач, до яких можна застосувати генетичний алгоритм, обмежується лише можливістю змоделювати середовище, у якому мають існувати особини [1].

Якщо розглядати саме рухомі об’єкти з реального світу, то найчастіше віхдними даними до оптимізаційної задачі будуть:

– закони фізики, що впливають на досліджуваний об’єкт;
– характеристики досліджуваного об’єкта (які і є шуканими параматрами);

- критерії життєздатності об’єкта;
- критерії якості виконання поставлених цілей;
- сторонні сили (якщо вони існують), що перешкоджають або допомагають об’єкту виконати поставлені йому цілі або впливають на його життєздастність.

Наприклад, необхідно виявити оптимальну форму, матеріал снаряду та кількість пороху для пострілу з корабельного фальконета, аби він завдавав найбільше ураження супротивнику за різної відстані. Звісно для цієї задачі необхідно враховувати силу тяжіння, супротив повітря, перетворення енегрії вибуху пороха на кінетичну енергію снаряда та інші фактори, що впливають на траєкторію польоту. Критерієм життєздатності моделі є попадання в ціль, критерієм якості виконання поставлених цілей – площа пошкодження обшивки супротивника. Природньо така задача вимагає деяких обмежень: снаряд у найширшому місці має бути менший за діаметр дула фальконета, об’єм пороху не може бути більше об’єму фальконета та інше.

Більш актуальною задачею може бути виявлення оптимальної конфігурації та поведінки БПЛА, що виконує бойове завдання. Параметрів такої моделі може бути безліч, проте серед основних можна виділити:

- розмір боєзапасу (що впливає на масу та розмір, а отже і на витрати палива та рівень маскування);
- розмір паливного баку;
- кількість камер та їх розташування;
- відображення матриці обзору камер на поведінку щодо скидування боєприпасів або ухилення від атаки.

Критерієм якості виконання поставлених цілей є знищення найбільшої кількості військових цілей. У якості сторонньої сили, що чинить опір, можна використати ППО (ідеальні або такі, які також можна паралельно навчати уражувати БПЛА). Критерієм життєздатності БПЛА може бути кількість часу, проведеного у повітрі, або ж кількість ухилень від пострілів ППО.

Метою роботи є формалізація оптимізаційної задачі поведінки та характеристик рухомих об’єктів та їх систем задля досягнення найвищого рівня їх життєздатності та якості виконання поставлених цілей; створення програмної системи, яка б дозволяла задати умови задачі такого типу у формальній і одночасно наочній формі, розв’язати цю задачу та візуалізувати проміжні та кінцеві результати.

Враховуючи вищеописане, можна сказати, що програмна система для формального та наочного опису оптимізаційних задач про рухомі об’єкти та їх системи має потенціал у якості інструменту для формування оптимальної поведінки самокерованих пристроїв в умовах симуляції, наближеної до реальності. Також може бути використана у якості допоміжного інструменту інженерів-конструкторів.

1 ОГЛЯД НАУКОВОЇ ТА ПАТЕНТНОЇ ЛІТЕРАТУРИ

1.1 Аналіз предметної галузі

Думки щодо можливості відтворення процесу еволюції за допомогою обчислювальної техніки з’явились у науковому суспільстві разом із статтею “COMPUTING MACHINERY AND INTELLIGENCE” [2] авторства Алана Тюрінга, опублікованою журналом “Mind” ще у 1950 році. У своїй публікації Тюрінг поставив питання “Чи може машина мислити?”, одразу перефомулювавши його в інше: “Що станеться, якщо машина візьме на себе роль допитуваного, що бреше, у грі Імітація?”. Ця гра передбачає участь трьох гравців: допитувач (*C*), допитуваний (*X*), що перешкоджає допитувачу та допитуваний (*Y*), що допомагає допитувачу. Допитувані *X* та *Y* належать до різних груп (*a* та *b* відповідно). Кожен з допитуваних намагається переконати *C*, що він належить до групи *b*. Мета *C* – визначити, хто з допитуваних бреше, а хто говорить правду. Питання, поставлене Тюрінгом, передбачає виявлення того, як зміниться частота невірних рішень *C*, якщо *X* – машина, а *Y* – людина, порівняно із випадком, коли *X* та *Y* – люди. Ця концепція більш відома як тест Тюрінга.

Учений вважав, що для створення машини, яка була б здатна успішно пройти його тест, достатньо апаратних можливостей тих часів, а проблема полягає лише в програмуванні цієї машини. Очевидно, що для проходження тесту Тюрінга машина, що бере участь у ньому, мусить мати розумові здібності дорослої людини: освіту та інший досвід. Вирішення цієї проблеми Тюрінг запропонував розділити на два етапи: програмування “дитячого розуму” та процес “надбання освіти та іншого досвіду”. Із таким формулюванням задачі вже можна провести досить очевидну паралель із алгоритмами машинного навчання, які ми знаємо сьогодні. Тюрінг пише: “Ми не можемо розраховувати знайти хорошу дитину-машину з першої спроби. Потрібно поекспериментувати з навчанням однієї такої машини та побачити, наскільки добре вона навчається. Потім можна спробувати іншу і побачити, краща вона чи гірша. Існує очевидний зв’язок між цим процесом і еволюцією: структура дочірньої машини – спадковий матеріал, зміни – мутації, природній відбір – судження експериментатора”.

Цей опис є тотожним генетичному алгоритму в деякій його конфігурації. Звісно, порівняння природного відбору із судженням експериментатора показує, що для такого навчання необхідна наявність великого обсягу навчальних даних або ж монотонна людська робота. Аби уникнити цих недоліків, можна додати до такої моделі зовнішнє середовище, установити мету, задати функцію, що визначає якість досягнення цієї мети, та замінити судження експериментатора на відбір особин за рівнем якості досягнення поставленої мети. Враховуючи таку модифікацію, набір задач, які зможе розв’язувати машина, обмежується тими, які можна промоделювати. При цьому зникає необхідність у навчальних даних і вчителі як такому, проте з’являється потреба у високому рівні обізнаності в предметній області модельованої задачі.

Протягом 50-х років XX століття генетики, біологи та інші науковці роблять перші спроби симуляції еволюції та штучного відбору, публікують численні роботи на цю тему. У 60-ті роки штучна еволюція стає загальновизнаним методом оптимізації, а на початку 70-х завдяки цьому підходу вперше було вирішено складні інженерні проблеми. Натомість ці дослідження у більшості випадків залишались лише теоретичними до середини 80-х років – саме тоді було проведено першу міжнародну конференцію з генетичних алгоритмів.

Наприкінці 80-х років компанія General Electric випустила інноваційний набір промислових обчислювальних засобів, який працював із використанням генетичного алгоритму.

Вивчивши логіку роботи генетичного алгоритму (рис. 1), можна побачити, що він складається з досить простих, природніх назві алгоритму, операцій. Генетичний алгоритм складається з таких етапів:

- створення початкової популяції;
- обчислення цільової функції;
- відбір особин для схрещування;
- схрещування;
- мутація;
- відбір особин для наступного покоління;

– перевірка виконання умови зупинки.

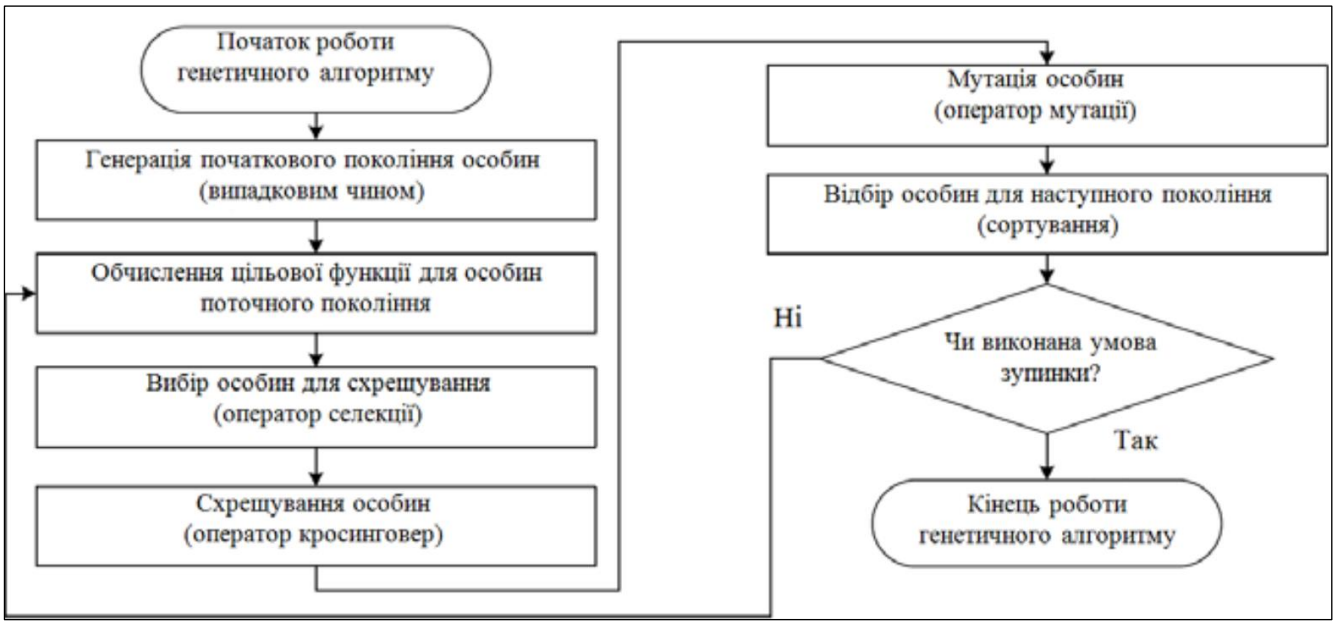

Рисунок 1 – Блок-схема генетичного алгоритму [3]

Першим комерційним продуктом для персональних комп'ютерів, що використовував генетичний алгоритм, став Evolver, випущений компанією Axcelis Inc. у 1989 році. Пізніше цей продукт був придбаний компанією Palisade Corporation, яка продовжує його підтримку в теперішній час. Сьогодні Evolver представлений у вигляді розширення до MS Excel (рис. 2).

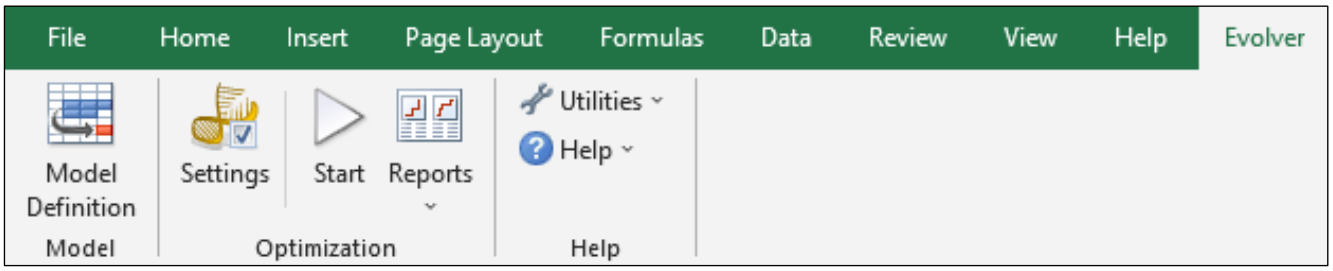

Рисунок 2 – Інтерфейс розширення Evolver

Для використання розширення необхідно вказати комірку цільової функції, мету оптимізації, проміжки комірок параметрів оптимізації, задати обмеження оптимізаційної задачі, якщо вони існують (рис. 3).

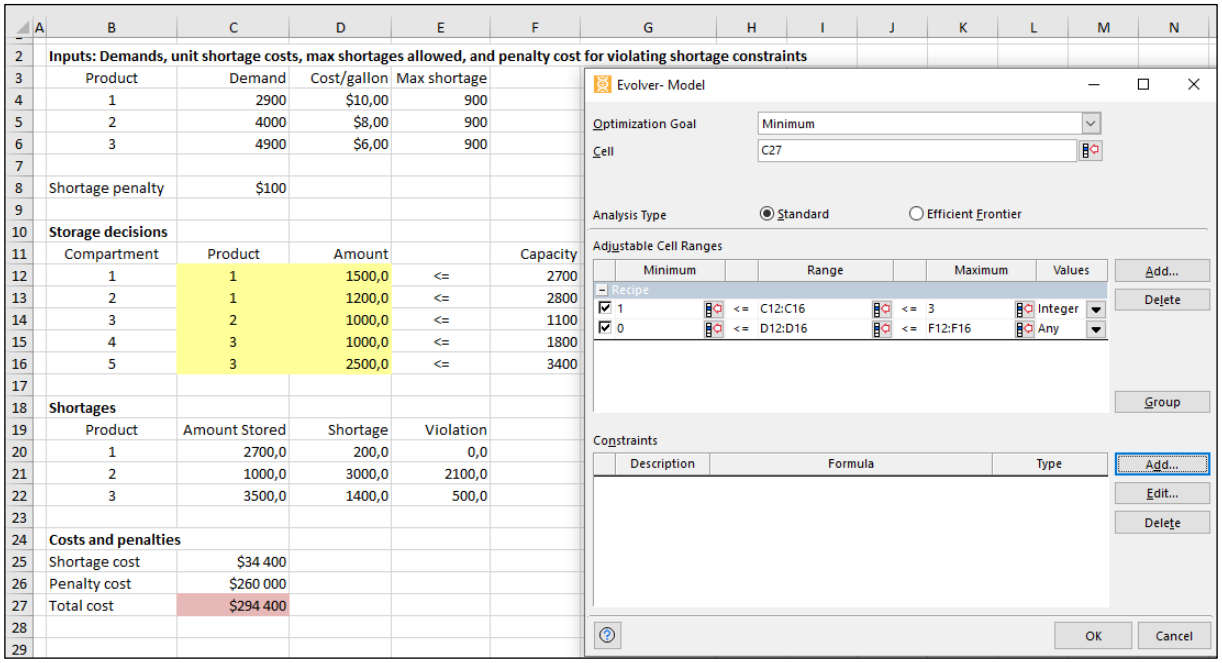

Рисунок 3 – Приклад використання розширення Evolver

Також можна налаштувати параметри (розмір популяції, коефіцієнти схрещування та мутації) та оператори генетичного алгоритму (рис. 4)

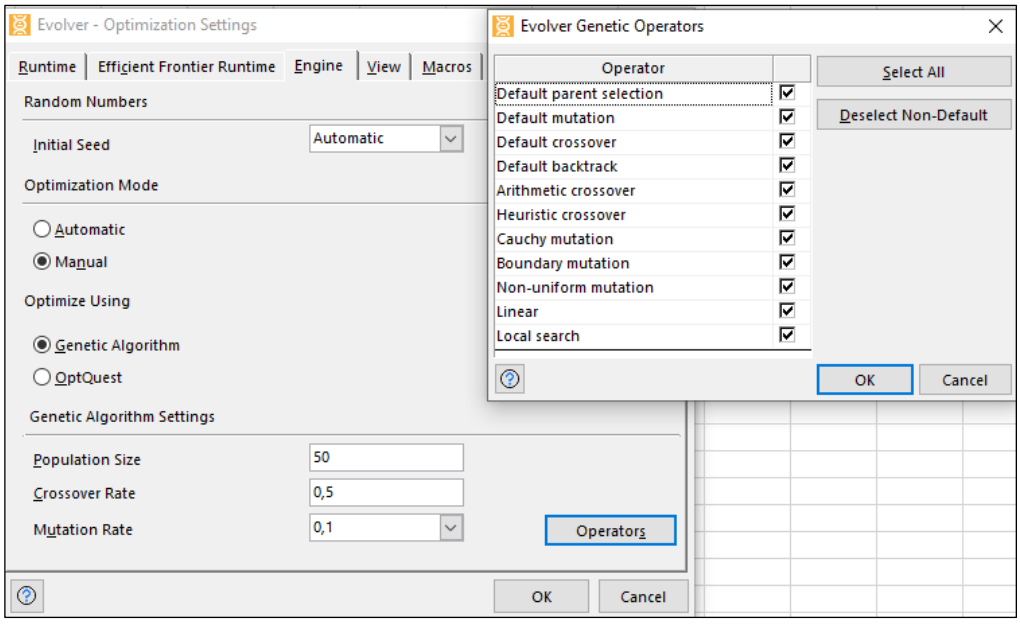

Рисунок 4 – Налаштування генетичного алгоритму

Іншим прикладом програмного забезпечення, що дозволяє працювати із генетичними алгоритмами, є Breve [4] (рис. 5) – програмне забезпечення, що

позиціонується як інтерактивне середовище розробки із широкими можливостями візуалізації. Breve має можливість обробляти скрипти, написані мовами Python та Steve (проста вбудована скриптова мова програмування), аби задати поведінку тривимірних об’єктів та дослідити їх взаємодію. Програмне забезпечення передбачає симуляцію законів фізики, що дозволяє створювати реалістичні об’єкти та механізми.

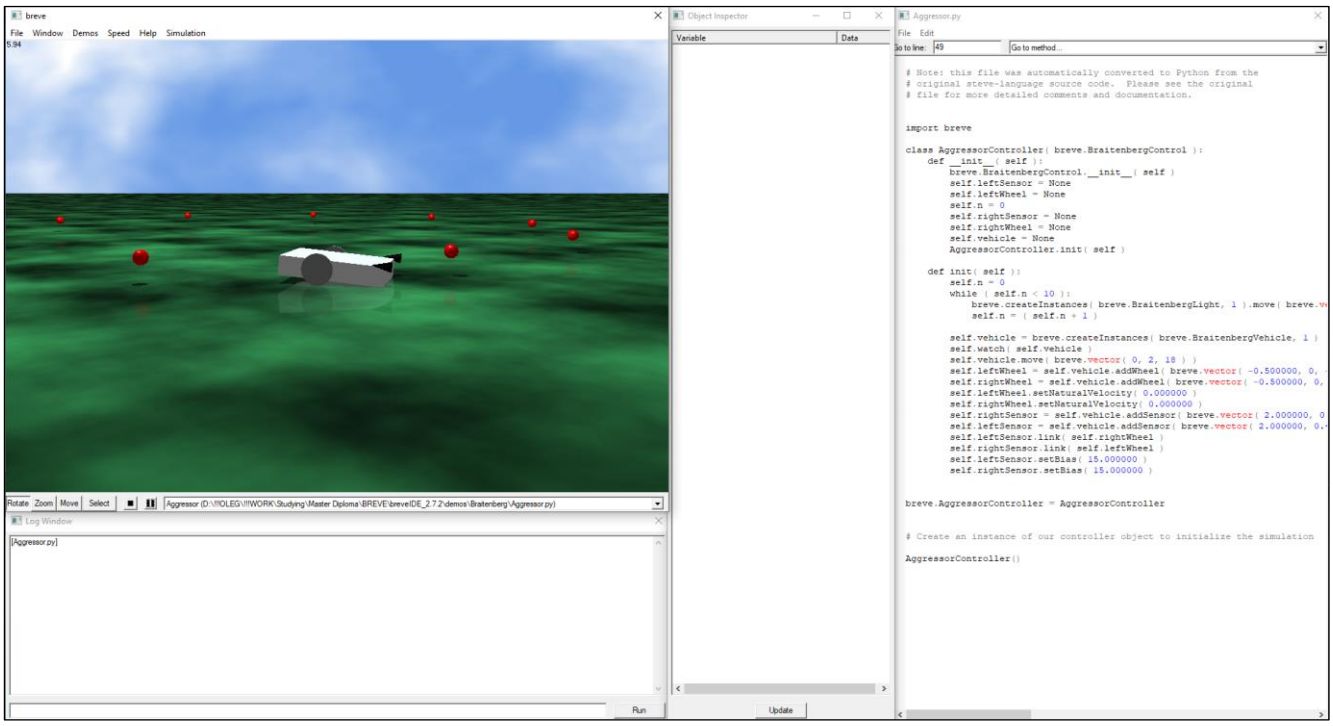

Рисунок 5 – Інтерфейс Breve IDE

На перший погляд може здаватись, що це програмне забезпечення не має прямого відношення до генетичних алгоритмів, проте бібліотека breve для мови Python та мова Steve містять велику кількість класів для симуляції штучного життя: GeneticAlgorithm, NeuralNetwork, PushGP тощо.

На жаль, підтримку як програмного забезпечення Breve, так і його офіційного сайту, було припинено у 2009 році, проте проєкт досі має свою аудиторію, через що у 2015 році автор створив нову сторінку із основними відомостями про проєкт.

У 90-х роках MATLAB включив у себе три евристичних алгоритми оптимізації, що не використовують похідні, серед яких був і генетичний алгоритм. Наразі MATLAB має широкі можливості програмування генетичних алгоритмів за допомогою модуля ga.

Якщо розглядати приклади досліджень використання генетичних алгоритмів для визначення життєздатності саме рухомих об’єктів та їх систем, то за цією темою було проведено багато аматорських експериментів.

Дехто робить спроби навчити модель ходити на двох ногах [5] (рис. 6) і вже на 500-му поколінні отримує задовільний результат.

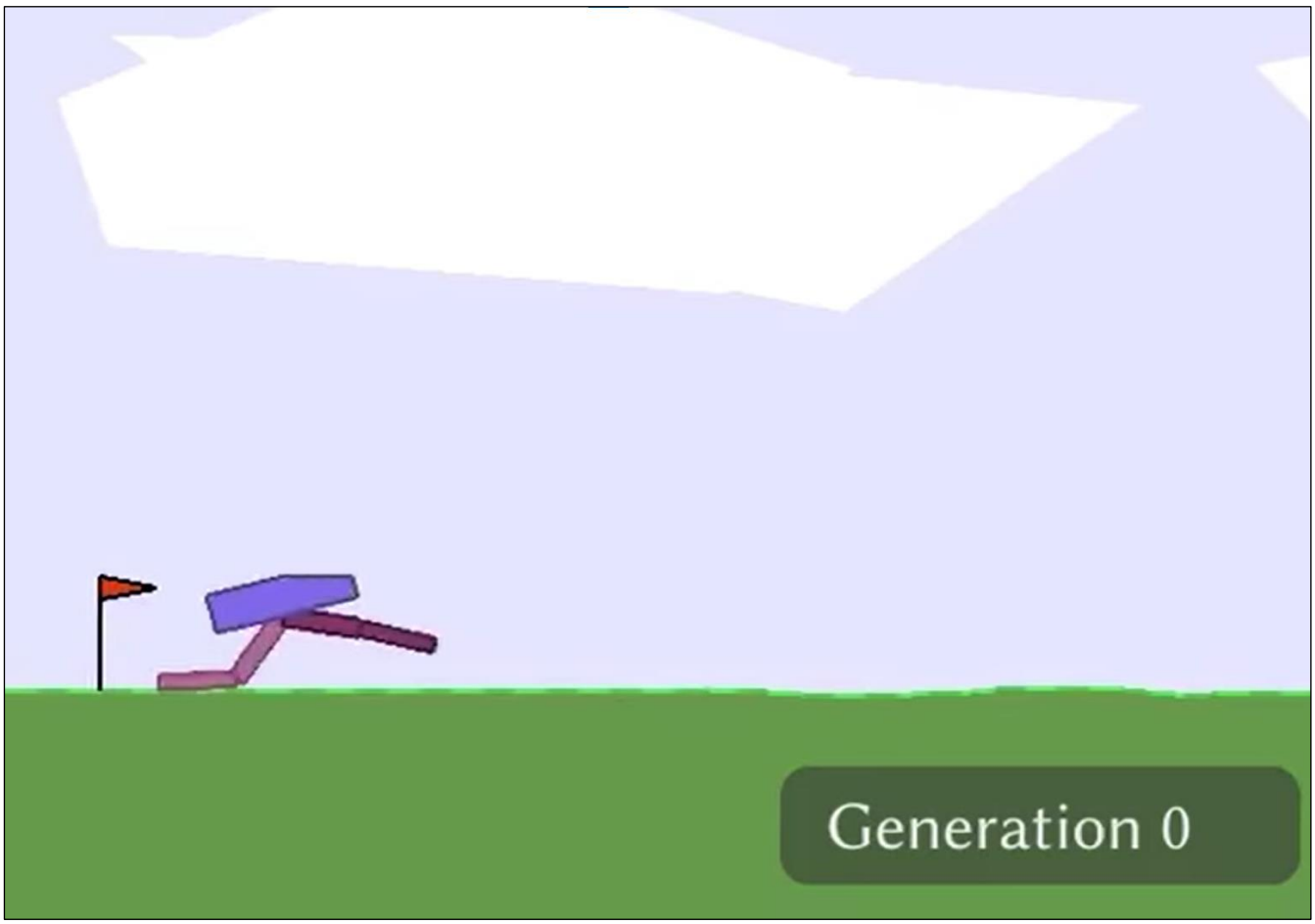

Рисунок 6 – Модель навчається ходити

Інші дослідники роблять спроби навчити модель стріляти та ухилятись від пострілів іншої такої ж моделі [6] (рис. 7). Між моделями наявна сіра зона, обмежена двома лініями, які вони не можуть перетнути. За 55 поколінь такі

моделі навчилась грати на такому високому рівні, що з легкістю перемогли б людину.

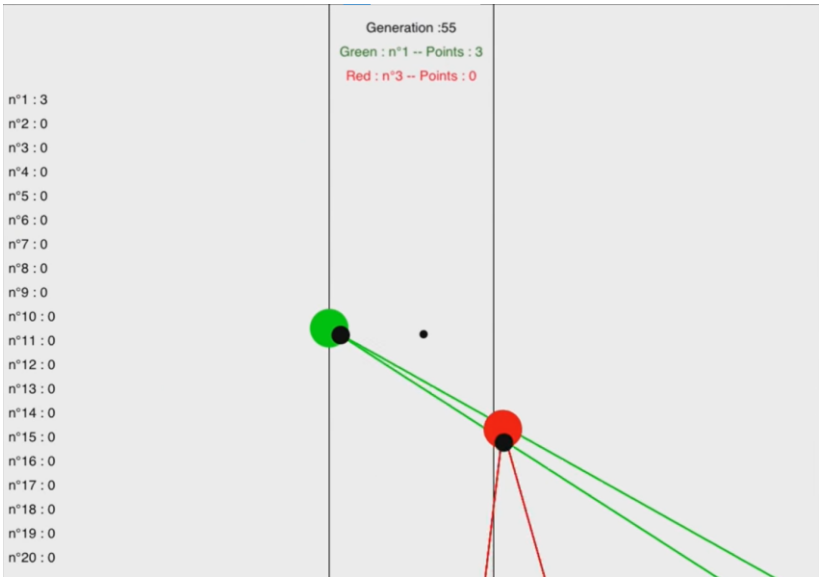

Рисунок 7 – Дві моделі навчаються стріляти одна в іншу та ухилятись

Наявні також дуже амбітні спроби навчити модель керувати дроном у двовимірному просторі за допомогою двох двигунів, збираючі при цьому деякі об’єкти та отримуючи за це винагороду [7] (рис. 8). На 200-му поколінні дрон вперше вдало отримав нагороду, на 2600-му – вперше зібрав усі 12 нагород, а на 5500-му – став робити це дуже швидко і плавно.

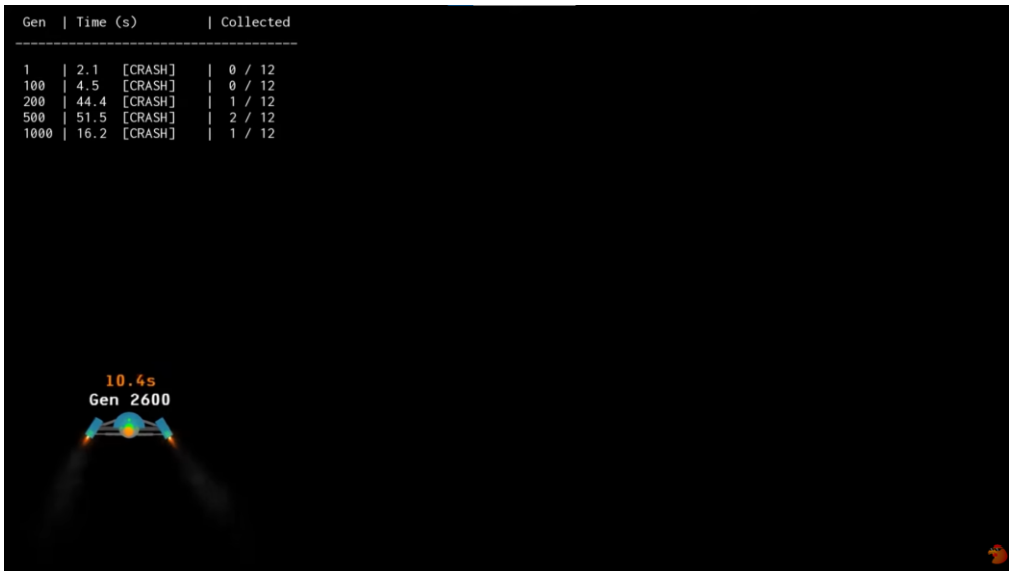

Рисунок 8 – Модель навчається керувати дроном та збирати винагороди

1.2 Виявлення проблем та актуалізація рішень

Програмне забезпечення Evolver є дуже мінімалістичним, простим та зручним у використанні. Воно дозволяє працювати із сухими числовими даними, що далеко не завжди є показовим. Наприклад, може бути поставлено задачу знаходження форми об’єкта, яка найкраще буде відповідати встановленим вимогам до деяких фізичних характеристик цього об’єкта. Так у 2006 році за допомогою генетичного алгоритму NASA створило антену незвичайної форми [8] (рис. 9), яка була здатна передавати радіосигнал набагато краще, ніж антени звичайних форм.

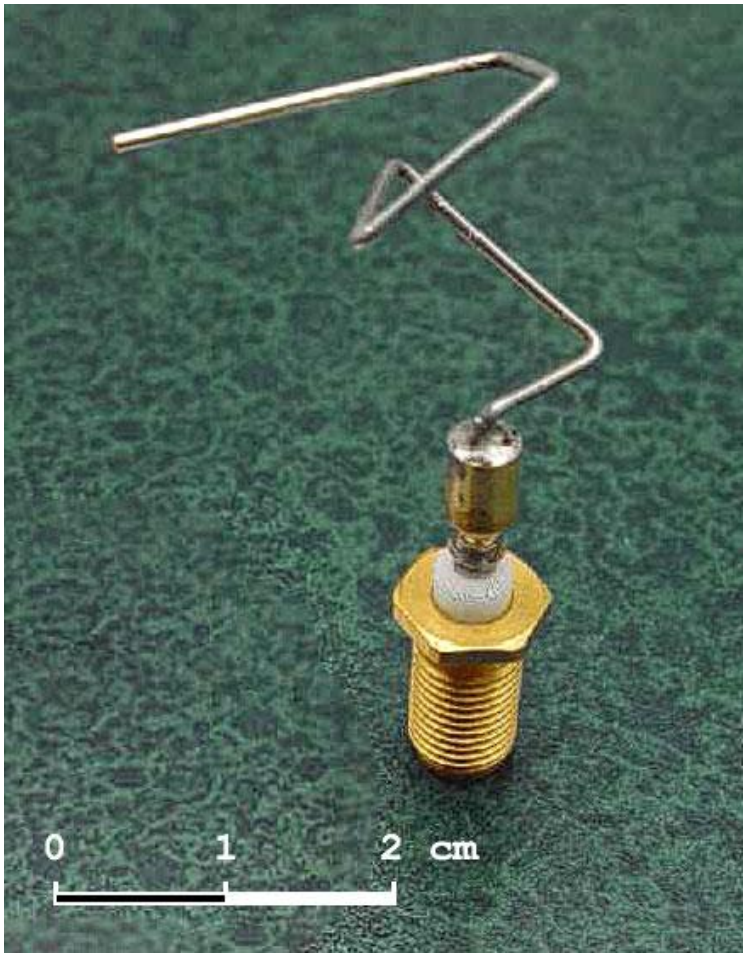

Рисунок 9 – Антена, форму якої було винайдено за допомогою генетичного алгоритму

Breve IDE має вбудовану можливість симуляції законів фізики, широкі можливості візуалізації та застосування генетичного алгоритму та інших підходів.

Проте це програмне забезпечення передбачає необхідність написання програмного коду, що не завжди відповідає можливостям дослідника. Більше того, цей засіб є застарілим та не підтримується.

Модуль ga пакету MATLAB очевидно також передбачає написання програмного коду, при чому, на відміну від Breve, не надає можливостей для візуалізації ходу експеримента.

Аматорські дослідження, розглянуті вище, – цікаві, але мають точковий характер, охоплюють лише деякі конкретні випадки. Вони мають досить високу якість, проте якщо лише трохи змінити умову оптимізаційної задачі, їх доведеться перепрограмовувати.

Урахувавши всі переваги та недоліки описаних вище програмних засобів та окремих експериментів, можна сказати, що програмне забезпечення із можливістю візуалізації ходу експеримента та простим механізмом задання умов оптимізаційної задачі, є затребуваним. Натомість повна відмова від написання програмного коду користувачем системи призведе до значних обмежень можливостей задання умов задачі. Отже необхідно звести необхідність у програмуванні користувачем до мінімуму. Важливо зазначти, що аби результат оптимізації мав інженерну цінність, особини мають складатись із відомих елементів (пропелер, електродвигун, колесо тощо), а не бути повністю випадковими (окрім випадку оптимізації форми).

1.3 Постановка задачі

На основі аналізу предметної галузі та виявлених проблем можна сформувати базові принципи, за якими має бути реалізована програмна система, що вирішуватиме ці проблеми.

Як було описано вище, особини мають складатись із відомих елементів, отже програмна система має містити бібліотеку таких елементів, які мають поділятись на категорії: елементи каркаса, елементи керування (двигун, пушка тощо), датчики (камера, датчик відстані, монітор властивості елемента та інші) та елементи з’єднання. Елемент може мати три види властивостей – метрики,

параметри та константи. Метрика елемента – його властивість, що може змінюватись під час обчислення цільової функції (наприклад об’єм палива). Параметр елемента є невід’ємною його частиною та може змінюватись у ході оптимізації (наприклад об’єм паливного баку), враховуючи задані користувачем обмеження. Значення властивості елемента, що є константою, задається користувачем та є незмінною протягом усього експерименту.

Невід’ємною частиною оптимізаційної задачі є цільова функція, роль якої має грати взаємодія особини із середовищем. Складовими частинами середовища є сцена (набір об’єктів середовища) та контролер. Контролер визначає логіку взаємодії особини з об’єктами сцени, умови надання особині балів пристосованості. Програмна система повинна містити бібліотеку сцен та контролерів, які можна поєднувати у різних комбінаціях.

У загальному випадку геном особини має складатись з двох частин: структурної та розумової. Структурна частина має визначати ієрархією та параметри елементів, з яких складається ця особина.

Оскільки у результаті оптимізації структурної частини генома не завжди можна отримані такі результати, що можуть бути застосовані на практиці, програмна система повинна надавати користувачу можливість власноруч задати структурну частину генома особини, а оптимізувати лише розумову частину.

У випадку оптимізації структурної частини генома користувачу має надаватись можливість встановити обмеження на використання елементів кожного типу.

Розумова частина генома визначає ваги нейронної мережі (НМРО), що відповідає за «мозок особини». Конфігурація вхідного шару нейронної мережі повинна задаватись розмірностями вхідних даних множини елементів-датчиків особини, а вихідного – сумарною кількістю ступенів свободи множини елементів керування.

Програмна система також має надавати можливість встановлення параметрів генетичного алгоритму: реалізацій генетичних операторів, умови завершення оптимізації, максимальний час обчислення цільової функції тощо.

Невід’ємною частиною програмної системи є візуалізація ходу експерименту, що і має бути відображено у її головному вікні. Обчислення цільової функції та візуалізація ходу експерименту може бути виконано паралельно, тому програмна система має надавати можливість встановлювати розмір групи особин, для яких обчислення мають виконуватись паралельно.

2 ПЛАНУВАННЯ ЕКСПЕРИМЕНТАЛЬНОЇ ЧАСТИНИ ДОСЛІДЖЕННЯ

2.1 Визначення об’єкта, предмета та мети дослідження

Об’єктом дослідження є рухомі об’єкти та їх системи, для яких актуальна задача оптимізації поведінки, розв’язки якої мають визначати їх поведінку та характеристики, враховуючи задані обмеження.

Предметом дослідження є способи застосування гентичних алгоритмів для визначення життєздатності рухомих об’єктів та їх систем, оптимізації їх конфігурації та поведінки.

Метою дослідження є виявлення ефективних способів використання генетичних алгоритмів для визначення життєздатності рухомих об’єктів та їх систем, формалізація та узагальнення задачі оптимізації їх конфігурації та поведінки за допомогою генетичного алгоритму; створення програмної системи, яка б дозволяла задати умови задачі такого типу у формальній і одночасно наочній формі, розв’язати цю задачу та візуалізувати проміжні та кінцеві результати.

2.2 Вибір платформи проведення дослідження

Проведення дослідження передбачає його візуалізацію та наявність середовища, у якому діють закони фізики, тому у якості платформи для проведення цього дослідження було обрано рушій Unity. Варто зазначити, що Unity має пакет ML Agents, проте він не передбачає використання генетичного алгоритму.

Для роботи з генетичним алгоритмом було обрано бібліотеку GeneticSharp, що містить готові реалізації генетичних операторів та власне реалізацію генетичного алгоритму.

Як вже було описано вище, розумова частина генома особини задається вагами нейронної мережі. Зазвичай, коли мова йде про роботу з числами, генетичний алгоритм працює на рівні бітів. Тому, не зважаючи на те, що існує низка відомих фреймворків машинного навчання, адаптованих для .NET, враховуючи необхідність нетривіального підходу до роботи із нейронними

мережами, було прийнято рішення не використовувати жодну з готових реалізацій, а створити власну.

2.3 Визначення алгоритмів та метрик

Дослідження передбачає використання генетичного алгоритму, проте окремі його оператори – селекція, схрещування, мутація – можуть змінюватись. Тому користувачу програмної системи має надаватись можливість вибору реалізацій цих операторів. Оскільки варіативність оптимізаційних задач, які можуть бути задані, дуже висока, існує необхіднсть лише в універсальних метриках – рівень пристосованості особини та тривалість роботи. Усі специфічні метрики, якщо вони необхідні, мають керуватись контролером сцени, а в результаті перетворюватись на рівень пристосованості.

2.4 Розробка плану проведення експерименту

У випадку наявності необхідності оптимізації структурної частини генома, проведення експерименту по визначенню життєздатних варіацій рухомих об’єктів можна розділити на наступні етапи:

а) налаштування параметрів експерименту;
б) створення популяції (A) особин з різною структурною частиною генома;
в) створення популяції (b_i) особин з різною розумовою частиною генома для кожної особини з множини A;
г) оптимізація розумової частини генома всередині кожної з популяцій множини B;
д) оптимізація структурної частини генома серед існуючих структур особин;
е) повторення кроків 2.4, б) – 2.4, д), доки не виконається умова виходу;
ж) аналіз результатів.

Етап налаштування експерименту складається з:

– вибору сцени;
– вибору контролера сцени та його параметрів;

- вибору готової структурної частини генома або встановлення обмежень на використання окремих елементів;
- налаштування властивостей елементів;
- налаштування конфігурації НМРО;
- вибору реалізацій генетичних операторів;
- встановлення розміру початкової популяції особин із різною структурною частиною генома та розміру початкової популяції особин із різною розумовою частнию генома;
- встановлення максимального часу обчислення цільової функції, умови завершення оптимізації розумової частини генома, та умови завершення оптимізації структурної частини генома;
- встановлення розміру групи особин для розпаралелювання обчислень.

У випадку, якщо обрано готову структурну частину генома, етапи 2.4, б) та 2.4, д) не виконуються.

2.5 Розробка формальної моделі експеримента

Для формалізації експеримента необхідно ввести умовні позначеннями, що будуть використані (див. табл. 1).

Таблиця 1 – умовні позначення

Умовне позначення	Значення
$A = \{a_0, a_1, \dots, a_n\}$	Популяція особин з різною конструктивною частиною генома.
$B = \{b_0, b_1, \dots, b_m\}$	Множина популяцій особин з різною розумовою частиною генома.
$RandStructPop(n)$	Функція генерації початкової популяції особин з різною конструктивною частиною генома. Розмір популяції задається параметром n.
$RandBrainPop(a, n)$	Функція генерації початкової популяції особин з

Кінець таблиці 1

Умовне позначення	Значення
	різною розумовою частиною генома. Розмір популяції задається параметром n. a – базова особина, від якої наслідується структурна частина генома.
$Experiment(x)$	Цільова функція, що обчислюється для особини x. Результат зберігається всередині особини.
$Select(X)$	Оператор селекції генетичного алгоритма.
$CrossoverB(b_x, b_y)$	Оператор схрещування для розумової частини генома.
$MutationB(b_x)$	Оператор мутації для розумової частини генома.
$BrainEnd(B)$	Умова закінчення оптимізації розумової частини генома.
$CrossoverS(a_x, a_y)$	Оператор схрещування для структурної частини генома.
$MutationS(a_x)$	Оператор мутації для структурної частини генома.
$End(A)$	Умова завершення експерименту.

Увевши умовні позначення, можна формалізувати хід експерименту наступним чином:

$A = RandStructPop(n)$

$B = RandBrainPop(a_i, m) \forall i \geq 0, i \leq |A|$

do

$do\ \forall\ i \geq 0, i < |B|$

$Experiment(b_{i,j})\ \forall j \geq 0, j \leq |b_i|$

$b_i = Select(b_i)$

$while\ not\ BrainEnd(b_i)$

$pairs = \{(X, Y) \mid X, Y\ \in b_i, |X| = |Y| = \frac{|b_i|}{2}, X \cap Y = \emptyset\}$

$$CrossoverB(pairs_{j,0}, pairs_{j,1}) \forall\, j \geq 0, j < \frac{|b_i|}{2}$$

$$MutationB(b_{i,j}) \forall\, j \geq 0, j < |b_i|$$

$$Experiment(b_{i,j})\ \forall j \geq 0, j \leq |b_i|$$

$$b_i = Select(b_i)$$

$$if\ End(B)\ then\ break$$

$$pairs = \{(X,Y) \mid X,Y\ \in B, |X| = |Y| = \frac{|B|}{2}, X \cap Y = \emptyset\}$$

$$CrossoverS(pairs_{i,0}, pairs_{i,1}) \forall\, i \geq 0, i < \frac{|B|}{2}$$

$$MutationS(b_i) \forall i \geq 0, i \leq |B|$$

$$while\ true$$

Для більшої наочності також було розроблено діаграму станів, що описує хід експеримента (рис. 10).

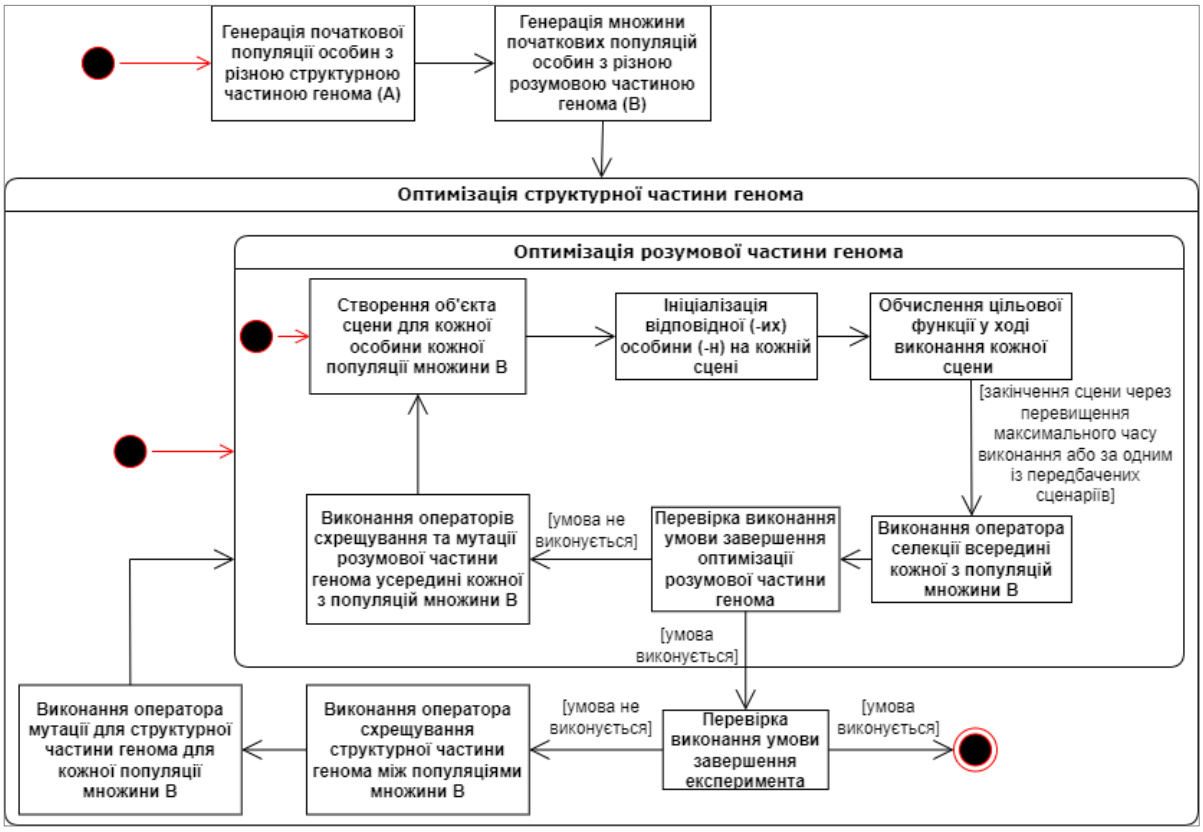

Рисунок 10 – Діаграма станів, що описує хід експеримента

2.6 Огляд способу кодування генома, обмеження генетичних операторів

2.6.1 Структурна частина генома

Структуру особини природньо можна представити у вигляді дерева, де ребра – елементи з’єднання, а вершини – елементи інших типів. Таке представлення є логічним ще й через те, що у Unity об’єкти також мають деревовидне представлення, а елементи з’єднання представлені не у вигляді об’єктів, а у вигляді компонентів, які додаються до об’єкта. Отже перетворення структурної частини генома на об’єкт особини не має створити проблем.

Окрім ієрархії елементів, існує ще низка необхідних параметрів, яка залежить від типу елемента. Елемент з’єднання має містити інформацію про тип з’єднання (жорстке, гнучке), позиції кріплення до елементів у локальних координатах цих елеметів та інші параметри, характерні для конкретного типу з’єднання. Для інших елементів необхідні дані щодо їх позиції, масштабу, обертання та параметрів, специфічних для конкретного елемента.

Варто зазначити, що при застосуванні оператора схрещування параметри елементів одної особини можуть взагалі не мати сенсу для елементів іншої особини. Тому у випадку оптимізації структури особини варто застосовувати оператор схрещування лише до ієрархії елементів, а їх параметри – копіювати. Якщо ж розглядати оператор мутації, то він може бути застосований як до ієрархії елеметів, так і до їх параметрів.

2.6.2 Розумова частина генома

Розумова частина генома задається вагами НМРО, конфігурація якої залежить від структурної частини генома, адже розмірність вхідного шару залежить від загальної розмірності даних елементів-датчиків, а розмірність вихідного – від сумарної кількості ступенів свободи елементів керування.

Аби спростити процес перенесення елементів-датчиків з однієї особини до іншої під час схрещування структурних частин генома та зменшити вплив такої операції на пристосованість окремих елементів-датчиків, кожен елемент-датчик має містити окрему нейронну мережу (НМД), що є часткою НМРО. Конфігурація

НМД має задаватись користувачем для кожного типу елементів-датчиків. Оскільки особина може містити декілька елементів-датчиків та загальна розмірність виходів НМД може не збігатись із кількістю виходів НМРО, існує необхідність у змішуванні виходів усіх НМД. Для цього має використовуватись повнозв’язна нейронна мережа – суматор (СМТР), – розмірність входів якої збігається із загальною розмірністю виходів усіх НМД, а кількість виходів – із кількістю виходів НМРО. Конфігурація прихованих шарів СМТР також має задаватись користувачем. Підсумовуючи, НМРО складається із усіх НМД відповідних елементам-датчикам особини та СМТР.

Ваги усіх складових НМРО можна представити у вигляді звичайного масиву, що спрощує взаємодію із генетичним алгоритмом. На етапі оптимізації розумової частини генома у особин, для яких виконується оператор схрещування, розмір цього масиву є однаковим, адже конфігурації нейронних мереж у цих особин однакові. Отже у цьому випадку обмежень на оператор схрещування і вочевидь на оператор мутації не накладається.

Якщо розглядати етап оптимізації структурної частини генома, то треба знову зауважити, що конфігурація розумової частини генома цілком залежна від структури особини. Таким чином, на етапі схрещуння структурної частини генома можуть відбуватись зміни НМРО шляхом вилучення та додавання окремих НМД.

Оскільки на цьому етапі особини мають різні структури, немає сенсу вираховувати внесок відповідної НМД до виходів СМТР НМРО-донора та інтегрувати цей внесок у СМТР НМРО-реціпієнта – ці СМТР можуть мати різні конфігурації, а їх вихідні нейрони можуть відповідати за різні елементи керування. Тому можна просто перенести НМД з НМРО-донора до НМРО-реціпієнта, видаливши відповідні вхідні нейрони з СМТР НМРО-донора та створивши відповідні вхідні нейрони для СМТР НМРО-реціпієнта.

3 ПРОЄКТУВАННЯ ПРОГРАМНОЇ СИСТЕМИ

3.1 Проєктування архітектури програмної системи

Програмна система не має передбачати фізичний зв’язок з будь-якими сторонніми або власними віддаленими сервісами, отже архітерктура монолітна.

Якщо розглядати шари архітерктури програмної системи, то серед них можна виділити наступні чотири шари:

- представлення – Поєднує у собі сцену, об’єкти сцени та їхню поведінку;
- бізнес-логіка – Уособлює в собі високорівневу логіку роботи програмної системи;
- ML – Містить низькорівневу логіку обробки даних (робота нейронної мережі, генетичного алгоритму тощо);
- ресурси – Статичний шар, який складається з бібліотеки сцен, бібліотеки контролерів сцен, бібліотеки елементів та бібліотеки зразків готових особин.

Шар представлення повинен мати зв’язки лише із шаром бізнес-логіки, а шар бізнес-логіки – із шарами ресурсів та машинного навчання.

3.2 UML-проєктування програмної системи

Визначення сценаріїв використання є одним із базових методів виявлення функціональних вимог до системи. Хоча у даному випадку більшість логіки роботи програмної системи відбувається без втручання користувача, діаграму сценаріїв використання (рис. 11) все ж варто створити аби чітко визначити, у яких випадках користувач здійснює активні дії, а у яких – є лише спостерігачем.

З діаграми сценаріїв використання можна зробити висновки, що основні сценарії користувача пов’язані з налаштуванням ходу експеримента та його запуском.

Діаграма станів, зображена на рисунку 10, у поєднанні з діаграмою сценаріїв використання досконало описує поведінку системи, отже можна перейти до проєктування структури програмної системи – сутностей та зв’язків між ними – розробки діаграми класів (рис. 12).

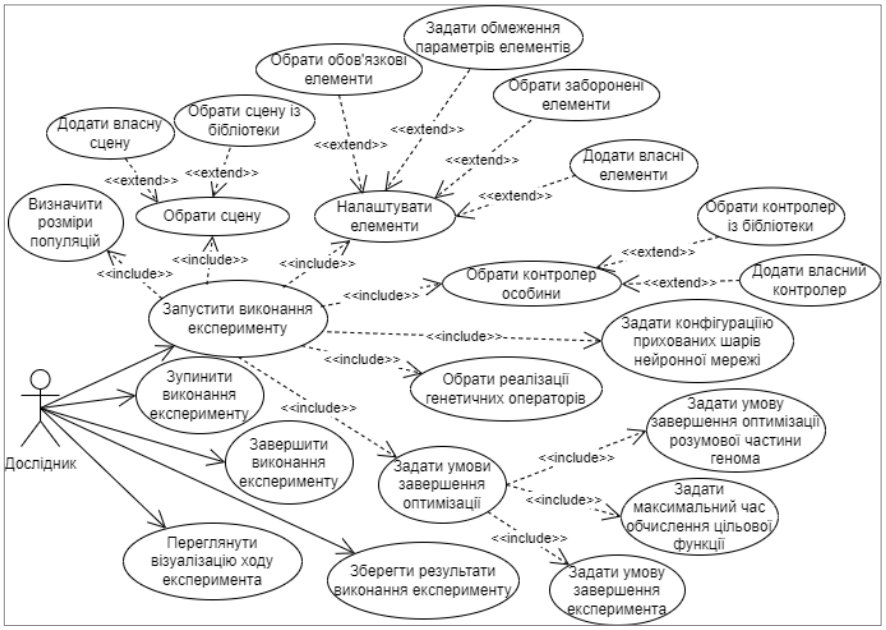

Рисунок 11 – Діаграма сценаріїв використання програмної системи

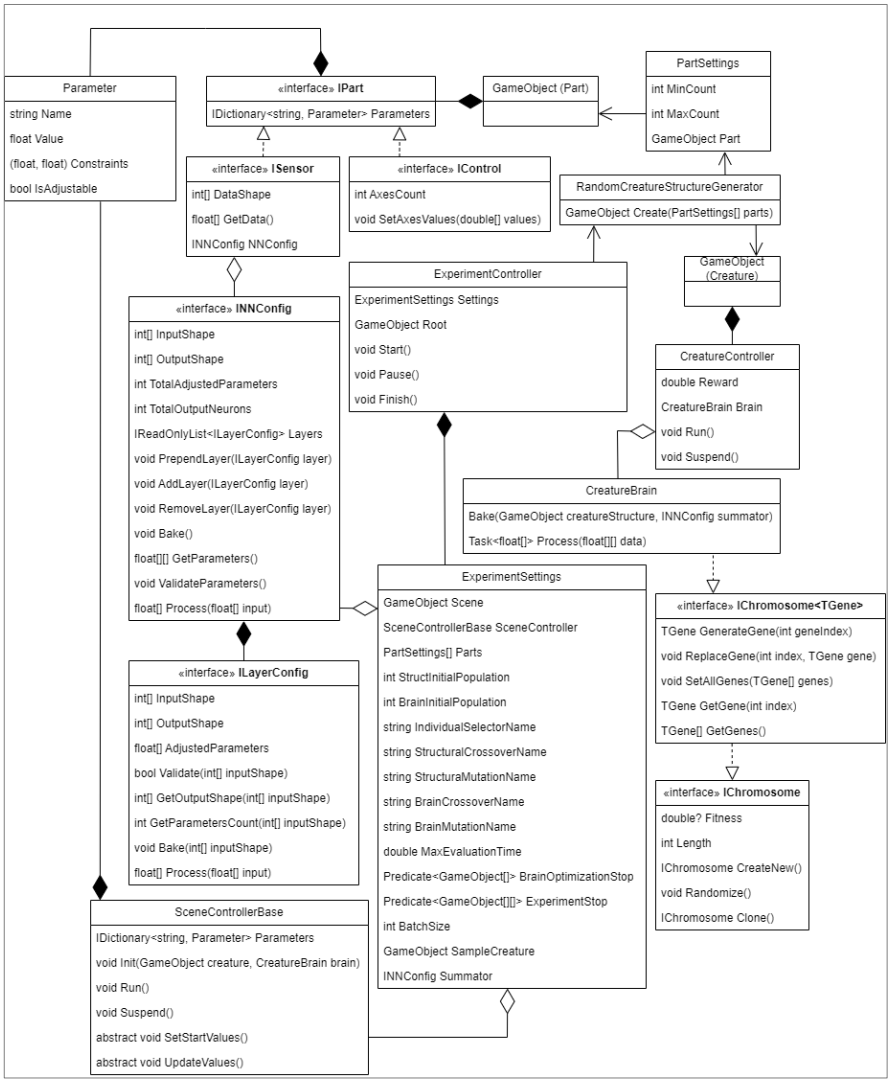

Рисунок 12 – Діаграма класів програмної системи

На діаграмі класів можна побачити, що використовується дві різних сутності GameObject – вбудований клас рушія Unity, який використовується для будь яких об’єктів сцени. Оскільки ці дві сутності семантично дуже різні, на діаграмі їх було зображено окремо.

4 РОЗРОБКА ПРОГРАМНОЇ СИСТЕМИ

4.1 Оптимізація бібліотеки GeneticSharp

У ході спроб інтеграції бібліотеки для роботи із генетичним алгоритмом GeneticSharp, було виявлено, що оригінальний код має великі проблеми з оптимізацією. Аби виявити причину низької продуктивності бібліотеки, було виконано профілювання за допомогою dotTrace. Аналіз показав, що багато часу витрачається на звернення до значень окремих генів особини. Досить логічно, адже структура Gene (див. додаток Б.1) зберігала значення у властивості типу object. Працюючи із чисельними даними, бібліотека оперує окремими бітами, тож за великої кількості параметрів накладні витрати на операції пакування та розпакування виявляються досить великими. Тому було вирішено переробити інтерфейс IChromosome (див. додаток Б.1), розбивши його на дві частини: IChromosome (див. додаток Б.1) – функціонал, незалежний від типу генів, та IChromosome<TGene> (див. додаток Б.1) – функціонал, що безпосередньо працює з генами.

Також було виявлено, що генетичні оператори, реалізовані у бібліотеці, не завжди працюють ефективно. Найчастіший патерн неефективного коду – використання метода ReplaceGene у циклі, що проходиться по всім генам. Цей патерн є неефективним через те, що не завжди можна знайти замінюваний ген за його індексом за час $O(1)$. Наприклад, створена реалізація CreatureBrain містить параметри не усередині себе, а у окремих об’єктах INNConfig, які в свою чергу мають таку ж саму залежність із об’єктами ILayerConfig. Тож, використовуючи бінарний пошук, цей ген за індексом можна знайти за час $O(log\ n)$. Тому було вирішено замінити усі випадки використання неефективного патерна на інший: отримання загального масиву генів за допомогою метода GetGenes, модифікація необхідних генів, оновлення оригінальних масивів генів за допомогою метода SetAllGenes.

4.2 Розробка реалізацій шарів нейронної мережі

У ході розробки реалізацій шарів нейронної мережі (див. додаток Б.2), головною вимогою до програмного коду була його ефективність. Розпаралелювати обчислювання на етапі обчислень у окремому шарі нейронної мережі не має сенсу, адже розпаралелювання вже застосовувалось на більш високому рівні обчислень.

Було розроблено два алгоритми для обчислення вихідних даних згорткового шару: алгоритм з перевіркою границь масиву оригінальних вхідних даних та алгоритм з додаванням відступів до вхідних даних і відсутністю необхідності перевірки границь масиву даних. Для порівняння ефективності алгоритмів було створено низку модульних тестів, які надавали на вхід великі об’єми даних.

Проте на великих об’ємах даних обидва алгоритми працювали однаково повільно: перший втрачав час на перевірках границь масиву, другий – на копіювання оригінального масиву даних у масив із відступами. Оскільки перший алгоритм не потребує виділення додаткової пам’яті, було прийнято рішення використовувати його. Натомість підвищити ефективність більше ніж в 4 рази вдалося шляхом кешування властивостей класу у локальні змінні метода.

5 ОГЛЯД РОЗРОБЛЕНОЇ ПРОГРАМНОЇ СИСТЕМИ

Робота із програмною системою починається із налаштування параметрів експеримента: вибору сцени, контролера сцени, налаштування параметрів контролера сцени, вибору готової структури особини або налаштування мінімальної та максимальної кількості елементів кожного виду, налаштування параметрів елементів, налаштування суматора, вибору генетичних операторів, налаштування інших параметрів генетичного алгоритму та встановлення розміру групи для паралельних обчислень.

Вікно вибору контролера сцени пропонує обрати (рис. 13) один із контролерів з бібліотеки та налаштувати його параметри у окремому модульному вікні (рис. 14).

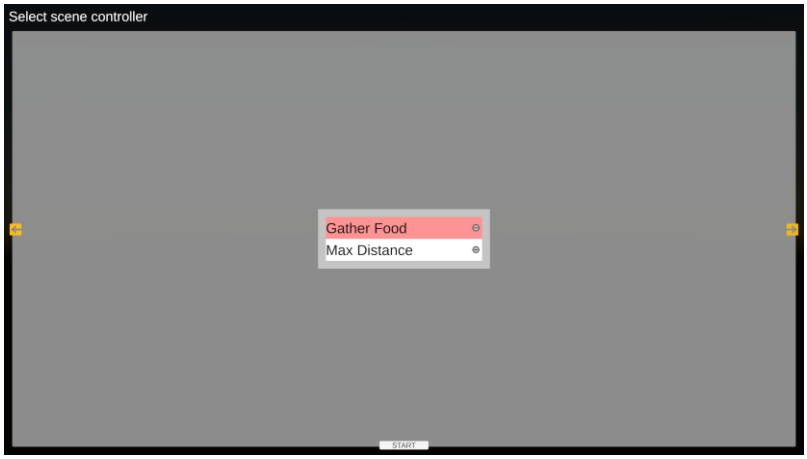

Рисунок 13 – Вікно вибору контролера сцени

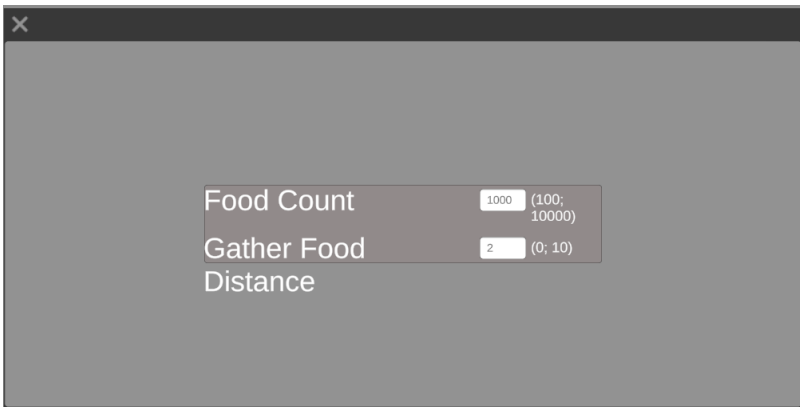

Рисунок 14 – Модульне вікно налаштування параметрів контролера сцени

Кожен вид елементів налаштовується у окремому вікні (рис. 15) задля зручності. Серед усіх видів елементів варто виділити елементи-датчики, для яких, окрім встановлення значень властивостей (рис. 16), користувач має задати конфігурацію НМД (рис. 17).

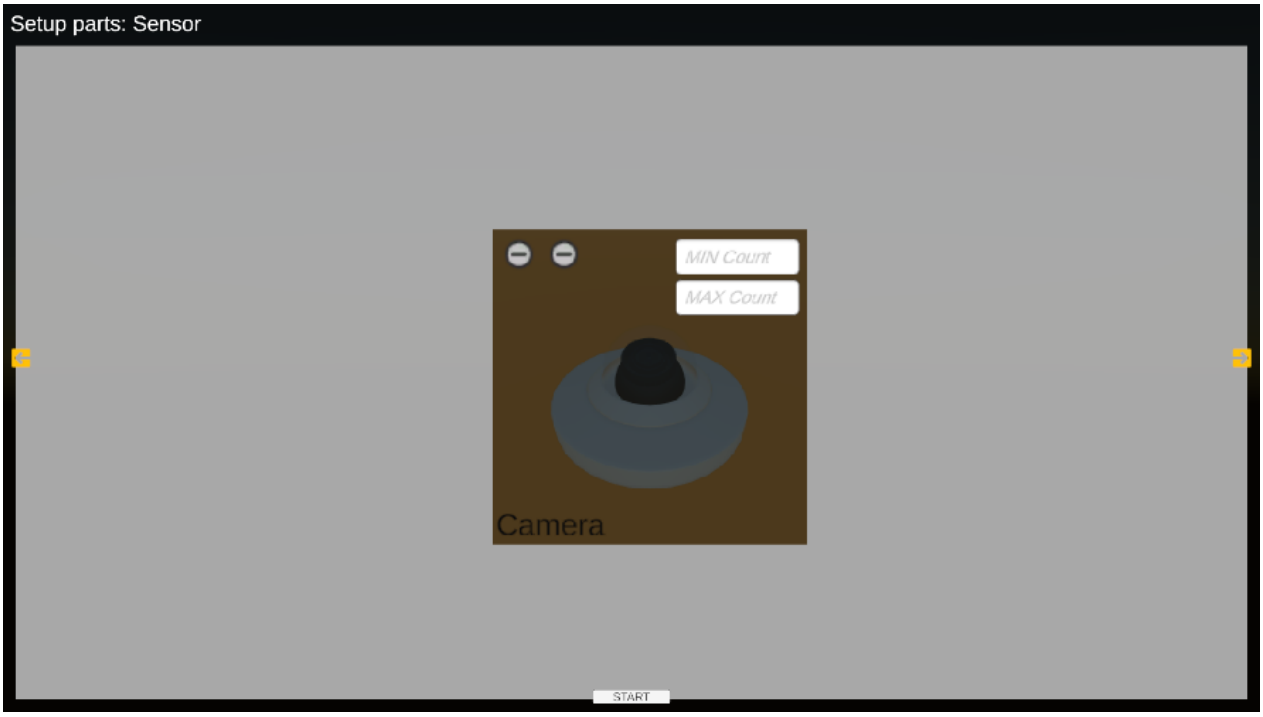

Рисунок 15 – Вікно перегляду бібліотеки елементів-датчиків

Рисунок 16 – Модульне вікно налаштування параметрів елемента

Рисунок 17 – Модульне вікно налаштування НМД

У вікні налаштування конфігурації суматора (рис. 18) користувачу надається можливість додавати лише повнозв’язні шари. Шари, задані користувачем, стануть прихованими шарами суматора, а програмна система автоматично додасть вхідний шар (відповідно до загальної розмірності виходів НМД) и вихідний шар (відповідно до сумарної кількості ступенів свободи елементів керування).

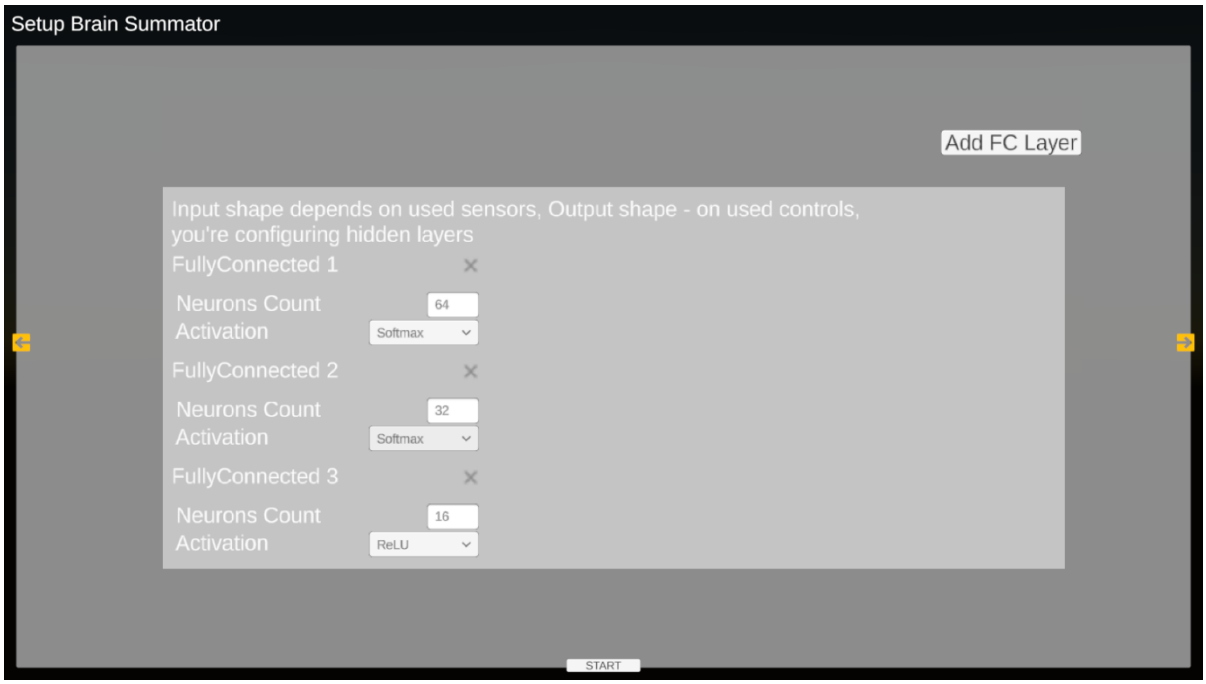

Рисунок 18 – Вікно налаштування конфігурації суматора

Вікно вибору генетичних операторів (рис. 19) дозволяє користувачу обрати такі генетичні оператори як оператор селекції, оператор схрещування, оператор мутації та оператор повторної вставки.

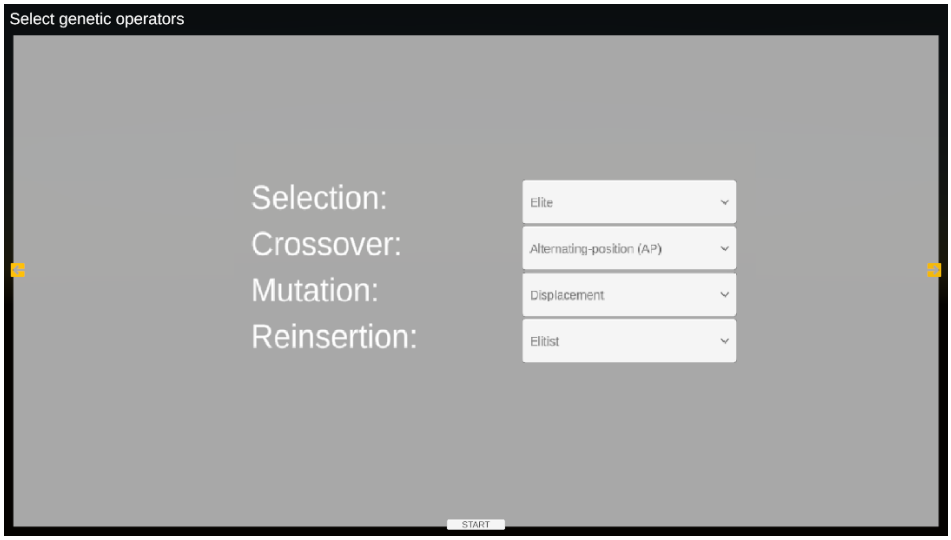

Рисунок 19 – Вікно вибору генетичних операторів

Вікно налаштування додаткових параметрів експеримента (рис. 20) містить поля для вводу початкового розміру популяції, максимального розміру популяції, кількості поколінь для завершення оптимізації розумової частини генома, часу обчислення цільової функції та розміру групи особин для розпаралелювання обчислень.

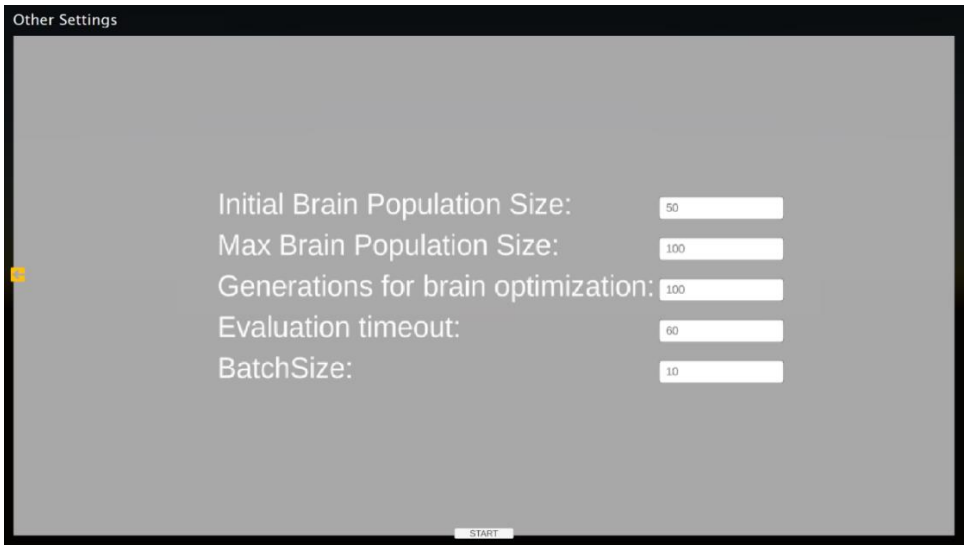

Рисунок 20 – Вікно налаштування додаткових параметрів експеримента

Вікно візуалізації ходу експеримента (рис. 21), окрім самої візуалізації, містить багато іншої інформації: номер поточного покоління, загальну кількість поколінь, номер поточної групи розпаралелювання, загальну кількість груп розпаралелювання, поточний кращий рівень пристосованості, поточний стан генетичного алгоритму.

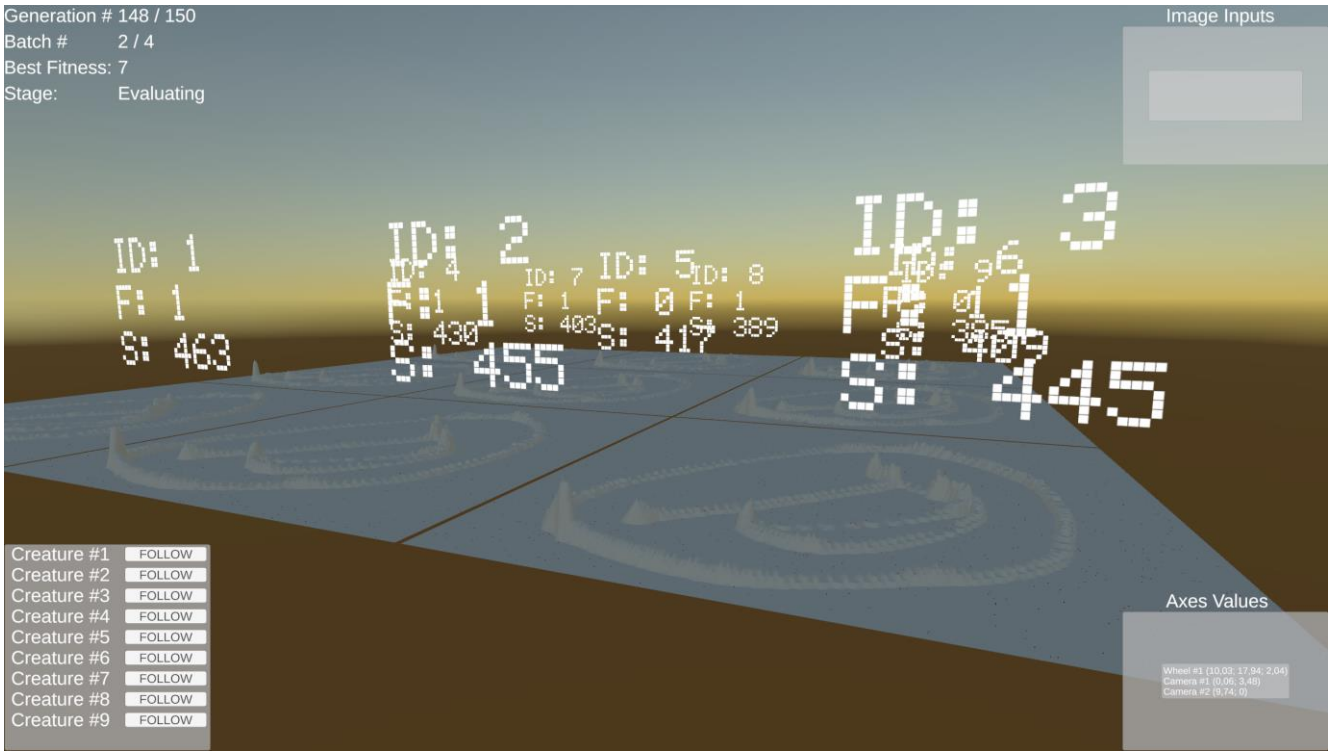

Рисунок 21 – Вікно візуалізації ходу експеримента

Окрім цього, у вікні візуалізації ходу експеримента наявна панель із переліком особин (рис. 22), для яких виконується обчислення цільової функції. Користувач має можливість обрати одну з особин, аби слідкувати за її діями.

Рисунок 22 – Перелік особин

Після вибору особини для слідкування у двох інших панелях – «Image Inputs» (рис. 23) та «Axes Values» (рис. 24) з’являться поточні зображення з камер відповідної особини та значення для ступенів свободи елементів керування цієї особини.

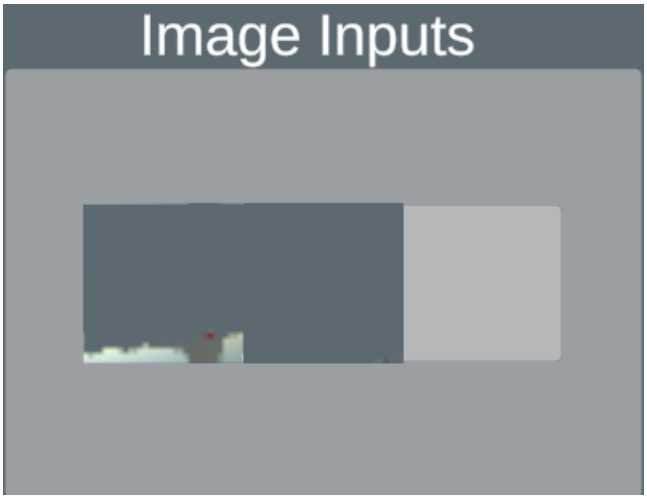

Рисунок 23 – Панель «Image Inputs»

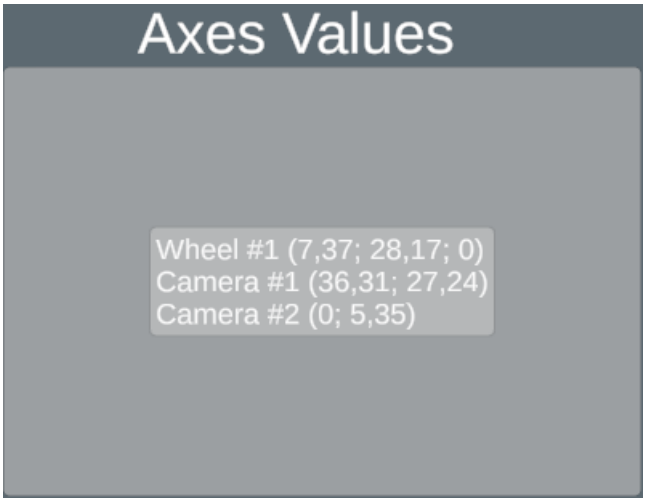

Рисунок 24 – Панель «Axes Values»

Створена програмна система наразі не передбачає оптимізацію структурної частини генома, а підтримує лише вибір готової структури особини і оптимізацію розумової частини генома.

6 ОПИС ПРОВЕДЕНИХ ЕКСПЕРИМЕНТАЛЬНИХ ДОСЛІДЖЕНЬ

6.1 Загальний опис проведених експериментальних досліджень

У ході дослідження для перевірки ефективності використання генетичного алгоритму для виявлення життєздатних конфігурацій рухомих об’єктів та їх систем було використано розроблену програмну систему із різними налаштуваннями експеримента. Оскільки створена програмна система наразі не передбачає оптимізацію структурної частини генома особини, у ході дослідження було використано готові структури особин. Порівняння ефективності застосування генетичного алгоритму має сенс у контексті фіксованого контролера сцени, фіксованої структури особини та фіксованої конфігурації НМРО особин. Оскільки за однакової структури особин конфігурація усіх НМД теж однакова, то критерієм однаковості НМРО особин є однаковість їх СМТР, отже для коректного порівняння ефективності використання генетичного алгоритма у різних його конфігураціях СМТР також має бути фіксованим.

6.2 Опис наявних генетичних операторів

Програмна система містить велику кількість різноманітних генетичних операторів. Серед операторів селекції [9] наявні:

- елітарний метод;
- ранговий метод [10];
- рулеточний метод;
- метод стохастчиної універсальної вибірки [11];
- метод турнірного відбору;
- метод усічення.

Оператори схрещування [12] містять:

- схрещування зміною позиції;
- циклічне схрещування;
- одноточкове схрещування;
- двоточкове схрещування;

– рівномірне схрещування;
– схрещування із голосуванням.

Оператори мутації [13] містять:

– мутація переміщенням гена;
– мутація заміною значення гена на протилежне;
– мутація вставкою;
– мутація частковим перемішуванням генів;
– мутація переворотом послідовності генів;
– мутація обміном двох випадкових генів;
– рівномірна мутація.

Оператори повторної вставки використовується коли поточний розмір популяції менший або більший за встановлені границі розміру популяції. Наявні наступні оператори повторної вставки:

– елітарний метод;
– метод повторної вставки на основі рівня пристосованості.

6.2 Експеримент зі збиранням їжі за фіксований час

Експеримент зі збиранням їжі (рис. 25) передбачає появу на сцені деякої кількості об’єктів їжі у випадкових місцях. Задачею особини є збір найбільшої кількості їжі за фіксований час. Розподіл об’єктів їжі на сцені – рівномірний, кількість їжі та відстань узяття їжі особиною визначаються параметрами контролера сцени, що задаються користувачем (див. додаток Б.3). Встановлена кількість об’єктів їжі – 100 шт, відстань для збору їжі – 2 одиниці.

У ході цього експеримента використовувалась структура особини, що складається з одного колеса та двох камер (рис. 26). Використана конфігурація НМД камери складається з вхідного шару розмірністю 32x32x3, трьох конволюційних шарів, трьох «Max-Pooling» шарів та одного повнозв’язного шару з 32 нейронами.

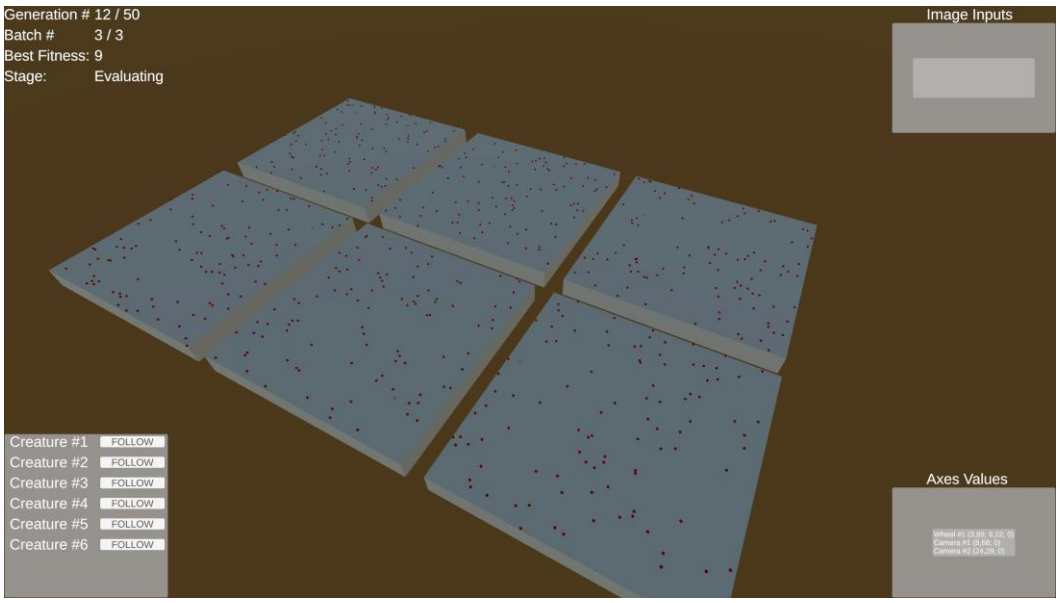

Рисунок 25 – Експеримент зі збирання їжі

Конфігурація суматора наступна: 64 нейрони на вході (відповідно 32 виходам з двох камер), один прихований шар із 32-ма нейронами, один прихований шар із 16-ма нейронами та 7 нейронів на виході (камера є як елементом-датчиком, так і елементом керування із двома ступенями свободи, колесо має три ступеня свободи).

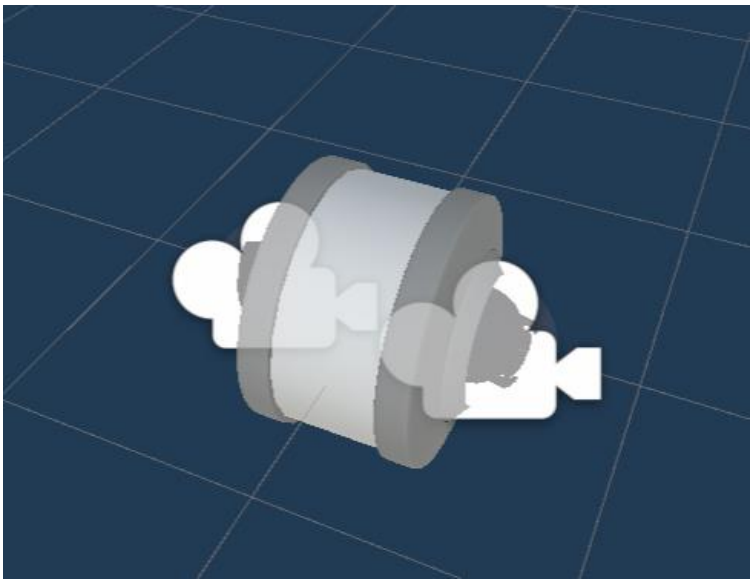

Рисунок 26 – Використана структура особини

Значення додаткових параметрів наведені у таблиці 2.

Таблиця 2 – Значення додаткових параметрів

Назва параметра	Значення
Початковий розмір популяції	30
Кількість поколінь для завершення оптимізації	50
Час обчислення цільової функції	30
Розмір групи розпаралелювання	10

Орієнтовний час проведення експеримента: $\frac{30}{10} * 30 * 50 = 4500$ секунд $=$ 75 хвилин. Аби виявити, які результати експеримента варто вважати прийнятними, а які – ні, за приклад низького рівня пристосованості візьмемо особину, яка рухається за прямолінійною траєкторією. Відстань для збору їжі $h_{\text{зб}} = 2$ од. є сферою із радіусом $R_{\text{зб}} = 2$ од., а об'єкт їжі представлений у вигляді сфери із $R_{\text{їжі}} = 0.5$ од. Тож очікувана кількість об'єктів їжі, зібраних особиною, дорівнюватиме $n_{\text{зібр}} = \frac{N}{S_{\text{сцени}}} S_{\text{тр}}$, де $S_{\text{сцени}}$ – загальна площа сцени; N – загальна кількість об'єктів їжі; $S_{\text{тр}}$ – площа перерізу фігури, утвореної переміщенням сфери збору їжі вздовж траєкторії руху особини, на відстані $h = R_{\text{зб}} - R_{\text{їжі}}$ від центра сфери збору їжі.

Розмір сцени, використаної у даному експерименті – $100\text{од} \times 100\text{од}$, отже $S_{\text{сцени}} = 100 * 100 = 10000$ од^2. Як було зазначено вище, $N = 100$.

Враховуючи траєкторію руху особини, утворена фігура буде складатись із циліндра, висота якого дорівнюватиме шляху $s_{\text{ос}}$, що подолала особина за час обчислення цільової функції t_f, та двох сфер на його основах. Отже $S_{\text{тр}} = 2\frac{\pi\left(\sqrt{R_{\text{зб}}^2 - h^2}\right)^2}{2} + s_{\text{ос}} * 2 * \sqrt{R_{\text{зб}}^2 - h^2} = \pi(R_{\text{зб}}^2 - h^2) + s_{\text{ос}} * 2 * \sqrt{R_{\text{зб}}^2 - h^2} = 1.75\pi + s_{\text{ос}} * 2\sqrt{1.75}$.

Оскільки для останнього прихованого шару обрано функцію активації, що має область значень [-1; 1], кількість нейронів у цьому шарі становить 16, то

максимальне можливе значення для кожного з вихідних нейронів – 16. Тобто максимальна кутова швидкість колеса дорівнює $\omega = 16\frac{\text{од}}{\text{с}}$. За середнє значення кутової швидкості візьмемо $\omega_{\text{ср}} = 8\frac{\text{од}}{\text{с}}$. Тоді, враховуючи радіус колеса $R = 0.5$ од, середня лінійна швидкість колеса складе $v_{\text{ср}} = R\omega_{\text{ср}} = 4\frac{\text{од}}{\text{с}}$. Рухаючись за прямолінійною траєкторією, зі швикістю $4\frac{\text{од}}{\text{с}}$ за час обчислення цільової функції $t_f = 30\text{с}$, особина подолає шлях $s_{\text{ос}} = v_{\text{ср}} * t_f = 4 * 30 = 120$ од.

Отже $S_{\text{тр}} = 1.75\pi + s_{\text{ос}} * 2\sqrt{1{,}75} = 322.98$ од^2. Таким чином, очікувана кількість зібраних об’єктів їжі за таких умов дорівнює $n_{\text{зібр}} = \frac{100}{10000} * 322.98 = 3.23$ шт. Таким чином, особини, що зібрали менше 4 об’єктів їжі можна вважати погано пристосованими.

У таблиці 3 наведено результати проведення експеримента із використанням різних генетичних операторів.

Таблиця 3 – Результати варіацій експеримента зі збору їжі

№	Селекція	Схрещування	Мутація	Повторна вставка	Пристосованість
1	Елітарна	Рівномірне	Рівномірна	Елітарна	10
2	Турнірний метод	Одноточкове	Частковим перемішуванням	Елітарна	10
3	Рулеточний метод	Двоточкове	Переворотом	На основі рівня пристосованості	10

6.3 Експеримент з їжею та отрутою

Недоліком попереднього експеримента є те, що він не є збалансованим: особина може, не відхиляючись відхиляючись від прямолінійної траєкторії,

зібрати досить велику кількість об’єктів їжі. Збалансувати експеримент можна додавши об’єкти, протилежні об’єктам їжі, – об’єкти яду (рис. 27).

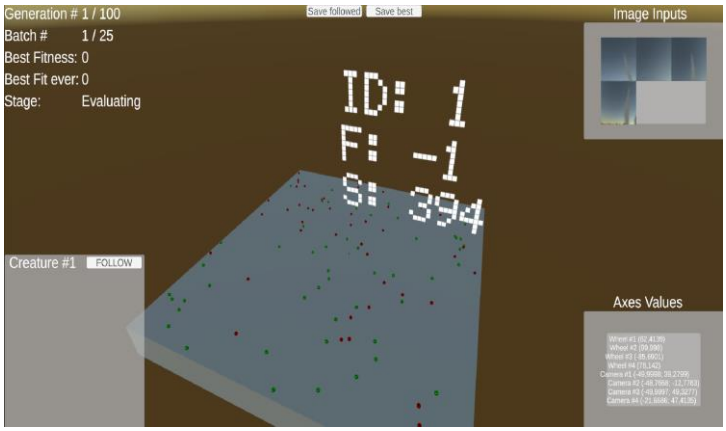

Рисунок 27 – Експеримент з їжею та отрутою

Оскільки за збір об’єкта їжі особина отримує одне очко пристосованості, а за збір яду – одне очко пристосованості вилучається (див. додаток Б.4), у такому експерименті значення пристосованості, що є від’ємним або близьким до нуля, варто вважати поганим рівнем пристосованості.

Усі параметри експеримента, окрім контролера експеримента, збігаються з експериментом зі збору їжі. У таблиці 4 наведено результати експеримента з їжею та отрутою.

Таблиця 4 – Результати варіацій експеримента з їжею та отрутою

№	Селекція	Схрещування	Мутація	Повторна вставка	Пристосованість
1	Елітарна	Рівномірне	Рівномірна	Елітарна	6
2	Турнірний метод	Одноточкове	Частковим перемішуванням	Елітарна	2
3	Рулеточний метод	Двоточкове	Переворотом	На основі рівня пристосованості	3

Інший недолік обох проведених експериментів полягає у структурі особини. Оскільки камери прикріплені до колеса, разом з обертанням колеса обертаються і камери, що створює додаткову складність для навчання моделі. Більше того, камери направлені перпендикулярно основній осі руху особини, тож особина не «бачить», що знаходиться попереду неї.

Через ці обставини було вирішено провести експеримент з їжею та отрутою, використовуючи оновлену структуру особини (рис. 28). Варто зазначити, що, на відміну від структури особини, зображеної на рисунку 26, у новій структурі особини використовуються елементи-колеса із одним ступенем свободи.

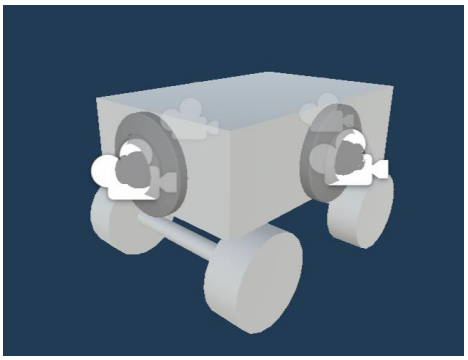

Рисунок 28 – Нова структура особини

Нова структура осбини має 4 камери, 4 колеса та корпус, отже загальна кількість вихідних нейронів НМД дорівнює 128, а сумарна кількість ступенів свободи дорівнює 12. Відповідно було оновлено конфігурацію суматора таким чином, аби він мав 2 прихованих шари із 64 та 32 нейронами відповідно. Інші параметри експеримента наведені у таблиці 5.

Таблиця 5 – Параметри експеримента з оновленою структурою особини

Назва параметра	Значення
Початковий розмір популяції	30
Кількість поколінь для завершення оптимізації	100
Час обчислення цільової функції	30
Розмір групи розпаралелювання	5

У ході цього експеримента було використано елітарний метод селекції, рівномірне схрещування, рівномірну мутацію та елітарний метод повторної вставки. Такий вибір генетичних операторів зумовлений результатами попередніх експериментів. Найбільшим значенням рівня пристосованості виявилось 4.

6.4 Експеримент з відстанню до об’єкта їжі

Експеримент з відстанню до об’єкта їжі принципово відрізняється від двох попередніх експериментів, адже дозволяє оцінити рівень пристосованості особини не залежно від того, чи зібрала вона хоч один об’єкт їжі. Така можливість досягається за рахунок наявності на сцені лише одного об’єкта їжі у будь-який момент часу (рис. 29). Об’єкт їжі з’являється у випадковому місці, а рівень пристосованості особини вимірюється як відношення $\frac{h_{\text{поч}}-h_{\text{пот}}}{h_{\text{поч}}} * \frac{|h_{\text{поч}}-h_{\text{пот}}|}{S}$, де $h_{\text{поч}}$ – початкова відстань між особиною та об’єктом їжі, $h_{\text{пот}}$ – поточна відстань між особиною та об’єктом їжі, S – довжина шляху особини.

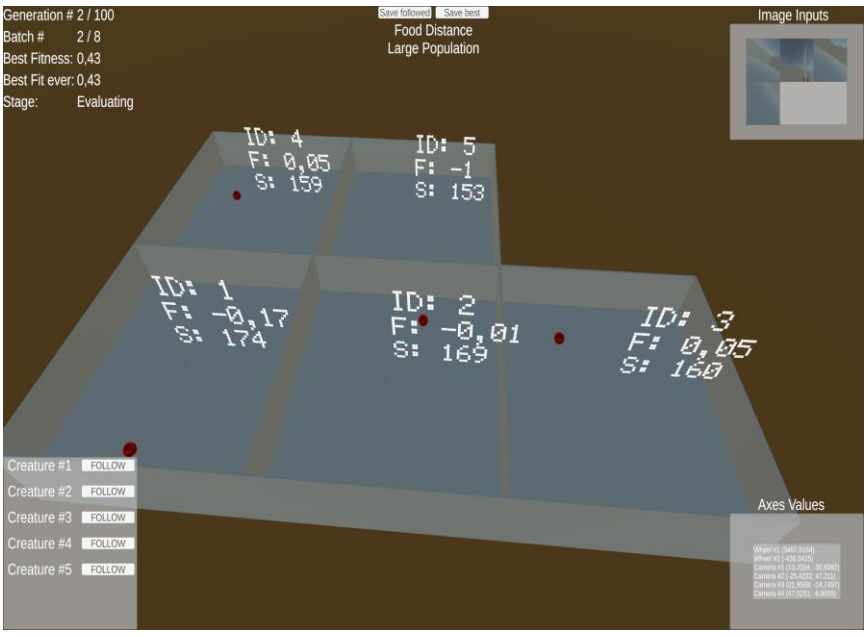

Рисунок 29 – Експеримент з відстанню до об’єктів їжі

У розрахунку рівня пристосованості враховано як міру наближення особини до об’єкта їжі, так і ефективність обраного шляху. У випадку збирання об’єкта їжі особиною, зібраний об’єкт видаляється, рівень пристосованості множиться на

коєфіцієнт, заданий дослідником, після чого створюється новий об’єкт їжі (див. додаток Б.5). Таким чином, максимальне значення рівня пристосованості особини у контексті одного об’єкта їжі дорівнює одиниці. Таке значення показує що особина дісталась об’єкта їжі, обравши при цьому найбільш оптимальний шлях до нього. Значення рівня пристосованості (у контексті одного об’єкта їжі) у діапазоні [0.5; 1] варто вважати показником гарного рівня пристосованості особини.

Для проведення цього експеримента було використано структуру особини, зображену на рисунку 28, елітарну селекцію, одноточкове схрещування, рівномірну мутацію та елітарний оператор повторної вставки. Експеримент має дві варіації, залежні від розміру початкової популяції та кількості поколінь. Результати експеримента наведені у таблиці 6, орієнтовний час виконання кожної з варіацій – 500 хвилин.

Таблиця 6 – Параметри експеримента з відстанню до об’єктів їжі

Назва параметра	Варіація №1	Варіація №2
Початковий розмір популяції	50	10
Кількість поколінь для завершення оптимізації	100	500
Час обчислення цільової функції	30	30
Розмір групи розпаралелювання	5	5
Рівень пристосованості	1.82	0.88

6.5 Аналіз результатів проведених досліджень

Оглядаючи проведені експериментальні дослідження, їх результати можна оцінити як задовільні.

У експерименті зі збором їжі останні 20 поколінь особин у більшості випадків обирали прямолінійну траєкторію, що дозволяло зібрати деяку кількість об’єктів їжі, після чого перетинали межі сцени і падали. Така поведінка пояснюється відсутністю штрафів за перетин меж сцени і тим, що будь-яка – навіть неоптимальна – траєкторія із високою ймовірністю принесе особині деяку

кількість очок пристосованості. Також така поведінка пояснюється невеликою кількістю поколінь особин.

Результати експеримента з їжею та отрутою є кращими, адже сам експеримент більш збалансований. Деякі особини дійсно обирали такі траєкторії руху, які б дозволили зібрати більше об’єктів їжі та оминути об’єкти яду, відхилялись від обраної траєкторії, коли у полі зору з’являвся той чи інший об’єкт. Якщо розглядати варіацію №1, значення пристосованості цілком задовільне, а його відмінність від агалогічної варіації експеримента зі збору їжі пояснюється меншою загальною кількістю об’єктів їжі на сцені експеримента з їжею та отрутою. Якщо розглядати варіації №2 та №3 цього експеримента, було помічено, що особини останніх 20 поколінь збирали 2-3 об’єкти їжі, після чого зупинялись або ж взагалі не зрушували з місця. Такий результат не можна вважати задовільним.

Експеримент з відстанню до об’єкті їжі показав, що залежність між вхідними даними з камер особини та її траєкторією дійсно утворюється і вдосконалюється із ходом експеримента. Також за результатами цього експеримента можна зробити висновок, що вигідніше мати більшу популяцію особин та меншу кількість поколінь, аніж меншу популяцію та більшу кількість поколінь.

Проведені експериментальні дослідження показали, що є досить вдалою конфігурація генетичного алгоритму, де використовується елітарна селекція, рівномірне схрещування, рівномірна мутація та елітарна повторна вставка. Проте для інших вхідних параметрів експеримента (сцени, контролера сцени, структури особини, структури НМРО, додаткових параметрів) інша комбінація генетичних операторів може надати кращий результат.

6.6 Шляхи подальшого розвитку дослідження

Враховуючи результати проведених експериментальних досліджень та проведений аналіз результатів, варто сформулювати шляхи подальшого розвитку дослідження.

Одним із векторов розвитку дослідження є вдосконалення створеної програмної системи: реалізація можливості оптимізації структурної частини генома особини, розширення бібліотеки сцен, контролерів сцен, елементів особин та готових структур особин. Також значним вдосконаленням було б впровадження можливості поєднувати різні контролери сцен у одному експерименті, що дозволило б підвищити рівень можливості їх перевикористання та рекомбінації, знизило б необхідність у написанні власних контролерів сцен.

Перспективним напрямом розвитку дослідження також є впровадження можливості використання готових розумових частин генома з метою їх дообучення у середовищі з іншими умовами.

Оскільки одним із способів отримання особиною інформації з середовища є використання зображень з камери, варто впровадити до програмної системи можливість використання методів передобробки зображень. Програмна система повинна надавати користувачу можливість налаштувати параметри передобробки даних.

У якості іншого, не менш важливого, вектора розвитку дослідження слід розглядати проведення значно більшої кількості експериментів, збільшення об’єму експериментів (кількості поколінь особин), збільшення кількості варіацій, що використовують різні комбінації генетичних операторів.

6.7 Опис можливості використання отриманих результатів у науковій та практичній діяльності

Розроблена програмна система наразі може бути використана у якості інструмента для проведення досліджень з виявлення життєздатності та пристосованості рухомих об’єктів та їх систем. Після реалізації вдосконалень, описаних у попередньому розділі, можливо, що розроблена програмна система буде придатною для розробки реальних інженерних проєктів та проєктів з автоматизації керування рухомими об’єктами та їх системами.

ВИСНОВКИ

У ході дослідження було проведено аналіз наукової та патентної літератури та інших джерел. Аналіз виявив, що не існує універсального програмного засобу для проведення експериентів з виявлення життєздатності та пристосованості рухомих об’єктів та їх систем за допомогою генетичного алгоритму. Також оглянуті дослідження з цієї теми мають досить специфічний характер, є досить вузьконаправленими. Тож для проведення дослідження було вирішено створити власну універсальну програмну систему, що дозволила б задавати умови оптимізаційної задачі, параметри генетичного алгоритму та шукати розв’язки заданої задачі.

Створена програмна система дозволяє налаштовувати параметри середовища, встановлювати порядок взаємодії особин із середовищем, задавати структуру особин та налаштовувати конфігурацію розумової частини генома особин, встановлювати параметри генетичного алгоритму та додаткові параметри експеримента. Після задання усіх необхідних параметрів програмна система надає візуалізацію ходу експеримента із можливістю слідкування за кожною окремою особиною та відстеження вхідних та вихідних даних.

За допомогою створеної програмної системи було проведено низку експериментів із різними параметрами середовища, різними конфігураціями структури особин та різними налаштуваннями гентичного алгоритму. Результати проведених експериментів було проаналізовано та оцінено як задовільні. На основі проведених експериментальних досліджень та їх аналізу було сформульовано шляхи подальшого розвитку програмної системи та дослідження в цілому, описано можливості використання отриманих результатів у науковій та практичній діяльності.

Проведено апробацію результатів дослідження у вигляді публікації статті (див. додаток В) на XXVII-му міжнародному молодіжному форумі «Радіоелектроніка та молодь у XXI столітті».

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Назаров О. С. Використання методів еволюційного моделювання для керування поворотним механізмом антени супутникового зв'язку. *Механіка та машинобудування*. 2004. № 2. С. 254–262.

2. Turing A. COMPUTING MACHINERY AND INTELLIGENCE. *MIND*. 1950. Т. LIX, № 236. С. 433–460. URL: https://academic.oup.com/mind/article/LIX/236/433/986238 (дата звернення: 05.01.2023).

3. ЕВОЛЮЦІЙНИЙ АЛГОРИТМ ДЛЯ АВТОМАТИЗОВАНОГО СТРУКТУРНО- ПАРАМЕТРИЧНОГО СИНТЕЗУ НВЧ ТРАНЗИСТОРНИХ ПІДСИЛЮВАЧІВ / Д. МАКАРИШКІН та ін. *Вісник Хмельницького національного університету*. 2016. № 6. С. 242. URL: http://elar.khmnu.edu.ua/jspui/bitstream/123456789/5416/1/макаришкін2.pdf (дата звернення: 05.01.2023).

4. Klein J. breve – the breve simulation environment. URL: http://www.spiderland.org/s/ (дата звернення: 05.01.2023).

5. Dollar Akshay. Genetic Algorithm. Learning to walk - OpenAI Gym, 2016. *YouTube*. URL: https://www.youtube.com/watch?v=uwz8JzrEwWY (дата звернення: 05.01.2023).

6. Ding Nicolas. A genetic algorithm learns how to fight!, 2013. *YouTube*. URL: https://www.youtube.com/watch?v=u2t77mQmJiY (дата звернення: 05.01.2023).

7. Pezzza's Work. Training a Neural Network to operate drones using Genetic Algorithm, 2020. *YouTube*. URL: https://www.youtube.com/watch?v=hQ4ryudP4j8 (дата звернення: 05.01.2023).

8. Cha S.-H., Tappert C. C. A Genetic Algorithm for Constructing Compact Binary Decision Trees. Journal of Pattern Recognition Research. 2009. Vol. 4, no. 1. P. 1–13. (дата звернення: 05.01.2023).

9. Wieczorek W., Czech Z. J. Selection Schemes in Evolutionary Algorithms. Intelligent Information Systems. 2002. P. 185–194. (дата звернення: 11.05.2023).

10. Selection - Introduction to Genetic Algorithms - Tutorial with Interactive Java Applets. www.obitko.com. URL: https://www.obitko.com/tutorials/genetic-algorithms/selection.php (дата звернення: 11.05.2023).

11. Selection - Introduction to Genetic Algorithms - Tutorial with Interactive Java Applets. www.obitko.com. URL: https://www.obitko.com/tutorials/genetic-algorithms/selection.php (дата звернення: 11.05.2023).

12. Xinjie Y., Mitsuo G. Encoding and Operators. Introduction to evolutionary algorithms. 2010. P. 40–63. (дата звернення: 11.05.2023).

13. Eiben A., Smith J. Variation Operators (Mutation and Recombination). Introduction to Evolutionary Computing. 2015. P. 31–32. (дата звернення: 11.05.2023).