

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ
ІНСТИТУТ

Кафедра математичного моделювання та аналізу даних

«До захисту допущено»

В.о. завідувача кафедри

«__» _____ 2023 р.

Дипломна робота
на здобуття ступеня бакалавра

зі спеціальності: 113 Прикладна математика
на тему: «**Модель розвитку простих організмів з
використанням генетичних алгоритмів та глибинного навчання**»

Виконав: студент 4 курсу, групи ФІ-91
Житкевич Іван Олександрович

Керівник: звання, степінь, посада Орехов О.А.

Консультант:

Рецензент: звання, степінь, посада Прізвище І.П.

Засвідчую, що у цій дипломній
роботі немає запозичень з праць
інших авторів без відповідних
посилань.

Студент _____

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені Ігоря СІКОРСЬКОГО»
НАВЧАЛЬНО-НАУКОВИЙ ФІЗИКО-ТЕХНІЧНИЙ
ІНСТИТУТ

Кафедра математичного моделювання та аналізу даних

Рівень вищої освіти — перший (бакалаврський)
Спеціальність (освітня програма) — 113 Прикладна математика,
ОПП «Математичні методи криптографічного захисту інформації»

ЗАТВЕРДЖУЮ

В.о. завідувача кафедри

_____ Сергій ЯКОВЛЄВ

«__» _____ 2023 р.

ЗАВДАННЯ
на дипломну роботу

Студент: Житкевич Іван Олександрович

1. Тема роботи: *«Модель розвитку простих організмів з використанням генетичних алгоритмів та глибинного навчання»*,

керівник: звання, степінь, посада Орехов О.А.,

затверджені наказом по університету №__ від «__» _____ 2023 р.

2. Термін подання студентом роботи: «__» _____ 2023 р.

3. Вихідні дані до роботи: *(впишіть вихідні дані до роботи)*

4. Зміст роботи: *(впишіть теми та задачі, які ви розкриваєте у роботі; можна робити це по пунктно)*

5. Перелік ілюстративного матеріалу (із зазначенням плакатів, презентацій тощо): *(якщо у вас є окремий ілюстративний матеріал окрім власне роботи (креслення, макети тощо), зазначайте; інакше вказуйте «Презентація доповіді»)*

6. Дата видачі завдання: 10 вересня 2022 р.

Календарний план

Студент _____ Житкевич І.О.

Керівник _____ Орехов О.А.

fill

РЕФЕРАТ

ABSTRACT

The English abstract must be the exact translation of the Ukrainian “annotation” (including statistical data and keywords).

ЗМІСТ

Перелік умовних позначень, скорочень і термінів	8
Вступ.....	9
1 Еволюційні Алгоритми	11
1.1 Визначення та принципи роботи еволюційних алгоритмів	12
1.1.1 Популяції	14
1.1.2 Фітнес-функція	15
1.1.3 Відбір	16
1.1.4 Мутація та рекомбінація	17
1.1.5 Елітизм	18
1.2 Застосування	18
1.3 Типи еволюційних алгоритмів.....	19
1.4 Генетичні Алгоритми	19
1.5 Критерії збіжності та завершення в еволюційних алгоритмах	21
1.6 Огляд попередніх робіт	21
1.6.1 Проста реалізація Натана Руя	22
1.6.2 Deep Evolutionary Reinforcement Learning	22
Висновки до розділу 1.....	23
2 Застосування глибинного навчання у моделюванні розвитку простих організмів	24
2.1 Нейронні мережі	24
2.2 Навчання нейронної мережі за допомогою генетичних алгоритмів	27
2.3 Процес моделювання розвитку простих організмів	28
Висновки до розділу 2.....	29
3 Реалізація моделі та аналіз результатів	30
3.1 Середовище	30
3.2 Організми	31
3.3 Кодування	33
3.4 Генетичні оператори	35

3.5	Опис інструментів та збір даних під час моделювання.....	36	7
3.6	Метрики.....	38	
3.7	Аналіз впливу параметрів по метрикам.....	38	
	Висновки до розділу 3.....	46	
	Перелік посилань.....	48	

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

індивід — потенційне рішення

популяція — набір потенційних рішень

генотип або **геном** — структура даних особини, що використовується під час селекції

хромосома — генотип у вигляді векторного гена фіксованої довжини

ген — певна позиція слота в хромосомі

фенотип — фізичні характеристики особини, її дії під час оцінки придатності

(Якщо ви не використовуєте перелік умовних позначень, просто приберіть даний розділ.)

(БУДЬ ЛАСКА, ПРОСЛІДКУЙТЕ, ЩОБ НОМЕР СТОРІНКИ СПІВПАДАВ ІЗ СПРАВЖНІМ! Це залежить від того, наскільки великим є ваш зміст. Номер сторінки пропонується у файлі `thesis.tex`, рядок 35.)

ВСТУП

Актуальність дослідження. Актуальність даного дослідження полягає у тому, що без нього ви не одержите диплом про вищу освіту. Відповідно, ви повинні оформити результати вашого дослідження належним чином.

Вступ є однією із самих формалізованих частин дипломної роботи. На початку ви у двох-трьох абзацах повинні окреслити проблематику та актуальність вашого дослідження, після чого переходити до мети та завдання.

Метою дослідження є певна абстрактна недосяжна річ на кшталт загальнолюдського щастя на горизонті. Для досягнення мети необхідно розв'язати **задачу дослідження**, яка полягає у чомусь суттєво більш конкретному. Для розв'язання задачі необхідно вирішити такі завдання:

- 1) провести огляд опублікованих джерел за тематикою дослідження;
- 2) (наступний пункт, пов'язаний із теоретичним дослідженням);
- 3) (і ще один, наприклад, про експериментальну перевірку результатів);
- 4) (і взагалі, краще із науковим керівником проконсультуйтесь, як ваші завдання правильно писати).

Об'єктом дослідження є якісь процеси або явища загального характеру (наприклад, «інформаційні процеси в системах криптографічного захисту»).

Предметом дослідження є конкретний математичний чи фізичний об'єкт, який розглядається у вашій роботі та який можна трактувати як певну властивість об'єкта дослідження (наприклад, «моделі та методи диференціального криптоаналізу ітеративних симетричних блочних шифрів»).

При розв'язанні поставлених завдань використовувались такі *методи дослідження*: і тут коротенький перелік (наприклад, але не

обмежуючись: методи лінійної та абстрактної алгебри, теорії імовірностей, математичної статистики, комбінаторного аналізу, теорії кодування, теорії складності алгоритмів, методи комп'ютерного та статистичного моделювання)

Наукова новизна отриманих результатів полягає... – тут необхідно перелічити, що саме нового з точки зору науки несе ваша робота. До усіх тверджень, які сюди виносяться, подумки (а іноді й явним чином) потрібно ставити слово «вперше» – і ці твердження повинні залишатись істинними.

Практичне значення результатів полягає... – тут необхідно зазначити практичну користь від результатів вашої роботи. Що саме можна покращити, підвищити (або знизити), зробити гарного (або уникнути поганого) після вашого дослідження.

Апробація результатів та публікації. Наприкінці вступу необхідно зазначити перелік конференцій, семінарів та публікацій, в яких викладено результати вашої роботи. Якщо результати вашої роботи ніде не доповідались, опускайте даний абзац.

1 ЕВОЛЮЦІЙНІ АЛГОРИТМИ

Традиційні детерміновані методи градієнтної оптимізації, такі як градієнтний спуск, покладаються на обчислення градієнтів цільової функції. Вони використовують ці дані для ітеративного коригування змінних в алгоритмі в напрямку зменшення або збільшення значення функції. Таким чином за допомогою таких методів ми знаходимо локальний або глобальний мінімум.

Однак градієнтні методи мають певні недоліки. Не завжди у реальних задачах, які включають оптимізацію, ми можемо мати достатньо інформації про процеси, щоб отримати цільову функцію, над якою проводимо оптимізацію. Як приклад, одну з таких функцій можна інтуїтивно описати як «чорний ящик», що означає, що вона є прихованою для нас, видає лише результат на основі вхідних даних та нічого більше. Також існують мультимодальні задачі, де градієнтні методи мають тенденцію збігатися до локальних мінімумів. Крім того, вони не підтримують багатоцільову оптимізацію за замовчуванням і можуть не справлятися з проблемами високої розмірності.

У тей же час маємо **еволюційні алгоритми**, які використовують стохастичний підхід, працюють з популяцією рішень і не потребують градієнтних обчислень. Основними їхніми перевагами є властивість досліджувати великий простір рішень, знаходити потенційні глобальні оптимуми в складних, мультимодальних задачах. Це пов'язано з тим, що їм притаманне різноманіття популяцій та варіаційні оператори. У їх основі лежать механізми, натхнені біологічною еволюцією, такі як відбір, мутація, рекомбінація. Це підвищує їхню адаптивність і гнучкість, дозволяючи їм працювати з цільовими функціями, які є недиференційованими, розривними або навіть невідомими. Також такі методи вміють обробляти багатоцільові задачі, підтримуючи різноманітний набір рішень представляючи різні компроміси між цілями.

1.1 Визначення та принципи роботи еволюційних алгоритмів

Витоки еволюційних алгоритмів простежуються до основних принципів біологічної еволюції. Процес еволюції організмів завжди був цікавою цікавою темою для науковців. Як біологів цікавили закони еволюції, її принципи так науковців у сфері комп'ютерних наук цікавив механізм розвитку організмів, їх змога пристосовуватися та навчатися. Комп'ютерні вчені прагнули імітувати природний спосіб адаптації до мінливого середовища та еволюції видів протягом мільйонів років, щоб створити надійні та адаптивні алгоритми для розв'язування складних задач. Це призвело до розробки еволюційних алгоритмів, які є набором обчислювальних моделей, що імітують процес природного відбору та генетичної мінливості.

Означення 1.1. Еволюційний алгоритм (ЕА) — це алгоритм, що належить до колекції технік еволюційних обчислень, які є натхненні біологічною еволюцією.

Більшість ЕА можна розділити на генераційні алгоритми, які оновлюють всю вибірку один раз за ітерацію, і стаціонарні алгоритми, які оновлюють вибірку декількома рішеннями-кандидатами за один раз. До найпоширеніших алгоритмів належать генетичний алгоритм (GA) та еволюційні стратегії (ES), причому для кожного з них існують як генеративні, так і стаціонарні версії [?].

Об'єктом цих алгоритмів є популяції індивідів, яких ми інтерпретуємо як потенційні рішення задач. У еволюційних алгоритмах використовуються різні принципи, такі як:

- 1) спадковість
- 2) природний відбір

Впродовж часу організми у популяції еволюціонують за правилами цих принципів, які диктують механізми:

- розмноження
- мутації
- рекомбінації
- відбору

Та не завжди у алгоритмах можуть використовувати усі види таких механізмів. Початковим ступенем є еволюційні стратегії (Evolution Strategies), які є сімейством алгоритмів. Вони мають просту процедуру, що складається з вибору усіканням (Truncation selection) та зазвичай одним з механізмів зміни — мутації [?].

Однією з найпростіших є (μ, λ) стратегія еволюції. Параметри μ та λ у назві індикують розміри вибірок у алгоритмі: μ означає кількість осіб у відборі, а λ — кількість отриманих нащадків. Також зазвичай початкова популяція має саме λ осіб. У підборі параметрів до алгоритму існує важливе правило — λ повинне бути кратним μ . Наприклад, позначають алгоритм при підібраних параметрах як "(10, 20) Еволюційна Стратегія".

Лістинг 1.1: (μ, λ) Еволюційна Стратегія

```
mu := number of selected parents
lambda := number of generate offsprings
P := GenerateInitialPopulation(lambda)
best := null
while best is null or best is ideal solution or we run out of time:
    best := individual for which fitness(individual) > fitness(best)
    S := TruncationSelection(P, mu)
    P := {}
    for each s in S:
        do lambda/mu times:
            P := Union(P, Mutation(Copy(s)))
endwhile
return best
```

У стратегії $(\mu + \lambda)$ символ "+" позначає інший метод відбору особин для наступного покоління у порівнянні із (μ, λ) . Цей метод вносить

елемент елітизму в еволюційний процес. Після процесу відбору популяція не зануляється, а замінюється на результат відбору. Потім відбувається процес мутації як у звичайній (μ, λ) стратегії. Це дозволяє батькам жити в наступному поколінні, якщо вони краще пристосовані, ніж їхні діти.

1.1.1 Популяції

Поняття популяції є ключовим в еволюційних алгоритмах. Популяція у еволюційних алгоритмах це набір потенційних рішень.

При побудові алгоритму так чи інакше можна стикнутися із питанням творення початкової популяції. При початковій популяції важливо досягти різноманітності у рішеннях, що дозволить дослідити різні частини простору розв'язків, яка призведе до підвищення ймовірності знаходження глобального оптимуму. Часто достатньо створювати початкову популяцію випадковим чином за допомогою рівномірного розподілу. Така створена популяція може охопити велику область простору розв'язків, досягаючи варіативності у рішеннях на початку.

За відомою інформацією про простір, такою як можливі підпростори оптимальних рішень, можна створити початкову популяцію для швидшої збіжності алгоритму. Таку інформацію можна отримати з попереднього дослідження простору, коригуючи параметри на дослідження всього простору.

Важливо згадати про два підходи до еволюції популяцій.

- 1) **Генеративний.** Наступна популяція цілком замінюється новою.
- 2) **Стаціонарний.** Зберігається поточна популяція в умовах її ітеративного розмноження (індивіди змінюються). Дозволяє потенційно хорошим рішенням залишатися у популяції надовше.

Стаціонарний підхід має дві особливості:

- 1) Він використовує вдвічі менше пам'яті, ніж традиційний генеративний алгоритм, оскільки одночасно існує лише одна популяція.
- 2) По-друге, він є досить експлуаторським (тобто

перевикористовує наявні рішення для отримання нових, що дає відтінок локального пошуку) порівняно з генеративним підходом: батьки залишаються в популяції, можливо, дуже довго, а отже, подібно до $(\mu + \lambda)$ еволюційної стратегії та елітизму, існує ризик того, що система *передчасно* зведеться до копій кількох дуже пристосованих особин [?].

Ще однією ключовою особливістю еволюційних алгоритмів є розмір популяції. Вона повинна бути достатньо великою, щоб підтримувати різноманітність і дозволяти широке дослідження простору розв'язків, але не настільки великою, щоб стати обчислювально непосильною. Не існує універсального розміру популяції, оскільки він визначається складністю завдання, доступними комп'ютерними ресурсами та конкретним еволюційним алгоритмом, що використовується.

1.1.2 Фітнес-функція

У процесі бере участь основна функція, яка скеровує популяції до кращих рішень.

Означення 1.2. Фітнес-функція — це функція, яка оцінює якість або доречність кожного рішення, що дозволяє скеровувати процес до все кращих рішень.

Чим вищий показник пристосованості індивіда, тим краще він пристосований до вирішення проблеми. Ця оцінка впливає на те, чи пройде індивід відбір в майбутньому поколінні. Як наслідок, хороша фітнес-функція повинна бути здатна відрізнити відмінні рішення від жахливих і забезпечувати чіткий шлях до розвитку.

Побудова фітнес-функції часто є складним завданням, яке значною мірою залежить від конкретної задачі, що вирішується. Вона може бути простим з математичне рівняння, або складною, як система правил. Саме ця функція «повідомляє» алгоритму про суть процесу, над яким проводиться робота. Фітнес-функція повинна відображати обмеження та

цілі проблеми. Якщо вона розроблена неправильно, це може ввести в оману процес пошуку і призвести до хибних результатів.

Варто також підкреслити, що фітнес-функція повинна бути **максимально ефективною** з точки зору обчислень. Обчислювально дорога функція пристосованості може значно сповільнити роботу еволюційного алгоритму, оскільки її потрібно обчислювати для кожної особини в популяції в кожному поколінні.

1.1.3 Відбір

Механізм відбору слугує базовою ланкою процесу формування наступного покоління. На основі різних алгоритмів відбору досягаються різні варіації набору індивідів для рекомбінації, мутації та потрапляння у наступну популяцію. За правилом, особи з вищим показником пристосованості з більшою ймовірністю будуть відібрані для розмноження. Це гарантує, що найбільш перспективні рішення передаються з покоління в покоління. Але у певних алгоритмах відбору існує варіант потрапляння слабких осіб у наступну популяцію для підтримання різноманітності.

Наведемо деякі найвідоміші процеси відбору:

Пропорційний відбір ґрунтується на припущенні, що ймовірність того, що індивід буде обраний, пропорційна його фітнес-функції. Якщо f_i — це фітнес-функція i -ої особи, а N — загальна кількість осіб у популяції, то ймовірність відбору $P(i)$ для i -ої особи обчислюється наступним чином.

$$P(i) = \frac{f_i}{\sum_{j=1}^N f_j}$$

Відбір усіканням (Truncation selection) є дуже простим методом відбору. Для формування наступного покоління обираються лише найкращі n осіб з найкращим рейтингом фітнес-функції. Наприклад,

якщо n дорівнює 10, то буде обрано лише 10 найкращих осіб у популяції.

1.1.4 Мутація та рекомбінація

Мутація та рекомбінація є двома ключовими механізмами в еволюційних алгоритмах, оскільки вони слугують фундаментальними джерелами різноманітності, що дозволяє досліджувати простір розв'язків, виходячи із локальних мінімумів.

Мутація проводить незначні випадкові зміни в особі. Задають певну ймовірність події мутації. Зазвичай таку ймовірність задають досить малою, в межах 0.01 та менше. Така рандомізація слугує для збереження різноманітності популяції та запобігає передчасній збіжності алгоритму до локального оптимуму. В тей же час, велика ймовірність мутації може призвести до сильного розмаїття популяції, що зменшує швидкість збіжності.

Як приклад мутації можна навести додавання шумів до одного обраного індивіда. Генерується аргумент — випадкове значення з гаусового розподілу із заданими параметрами в залежності від задачі. Далі цей аргумент додається до значення індивіда. Змінюючи параметри розподілу, з якого беремо величину шуму, можна керувати процесом знаходження розв'язків: чи будуть вони генеруватися поблизу локальних оптимумів, чи ж зміни приведуть до дослідження нових підпросторів розв'язків.

Рекомбінація (recombination) є ще одним механізмом утворення різноманітності у популяції. Вона ще відома як кросинговер (crossover). Але її механізм потребує не одну особу, а декілька або більше. Існують як рекомбінації, подія яких залежить від ймовірності, так і ті, що відбуваються завжди по отриманню осіб по відборі. Цей механізм об'єднує "батьківських" особин для утворення одного, двох, або й більше нащадків. На меті стоїть утворення кращого нащадка, зливаючи "хороші" компоненти батьків.

Існує безліч варіантів поєднання двох рішень, для створення кращого. Кожен з них показує себе краще у певних своїх задачах. Для створення кращого алгоритму та параметрів під нього потрібно експериментувати та досліджувати простір рішень у задачі. Саме сукупність процесу відбору, рекомбінації та мутації в залежності від обраних варіантів може давати різні результати. Важливо знайти той, що надає гарний баланс між дослідженням простору та пошуком локальних рішень.

1.1.5 Елітизм

Елітизм це проста техніка. Вона полягає у додаванні до наступної популяції найкращих індивідів з попередньої. Вона походить від стратегії $(\mu + \lambda)$, яка зберігає у наступну популяцію індивідів, що пройшли відбір. Все, що наслідує цю еволюційну стратегію має характер елітизму.

Елітизм схожий на відкладення найкращої роботи в безпечну зону, поки надалі проводиться дослідження нового простору. Крім того, елітизм має потенціал для стабілізації еволюційного процесу. Без нього найкращі рішення може бути втрачене, якщо воно не призведе до появи конкурентоспроможних нащадків, що призведе до зміни якості найкращого рішення з плином часу.

До речі, статичний підхід є певною мірою елітизму, бо значна частина осіб переходить у "наступну популяцію".

1.2 Застосування

Завдяки своїй адаптивності та гнучкості еволюційні алгоритми можна застосовувати до широкого спектру проблемних областей. Їх можна використовувати для оптимізації складних математичних функцій, наприклад, там, де традиційні методи оптимізації не

спрацьовують. Вони використовуються в машинному навчанні для вибору функцій, налаштування гіперпараметрів і навіть для навчання нейронних мереж. Вони також знайшли застосування в таких сферах, як складання розкладу, планування маршрутів та ігри, де їх використовують для отримання високоякісних результатів за розумний проміжок часу. Кожна програма застосовує фундаментальні принципи еволюційних алгоритмів, адаптуючи їх до конкретних потреб і обмежень поставленої задачі.

1.3 Типи еволюційних алгоритмів

Більшість еволюційних алгоритмів можна розділити на генераційні алгоритми, які оновлюють всю вибірку один раз за ітерацію, і стаціонарні алгоритми, які оновлюють вибірку декількома рішеннями-кандидатами за один раз. До найпоширеніших алгоритмів належать генетичний алгоритм (GA) та еволюційні стратегії (ES), причому для кожного з них існують як генеративні, так і стаціонарні версії [?].

Однак різноманітність алгоритмів на одним генетичних та еволюційних стратегіях не закінчується. Хоча всі ці алгоритми мають спільні базові концепції, вони відрізняються конкретними процесами, які вони використовують для дослідження простору пошуку задачі. До еволюційних алгоритмів належать: генетичні алгоритми (Genetic Algorithms), стратегії еволюції (Evolution Strategies), генетичне програмування (Genetic Programming), диференціальна еволюція (Differential Evolution) та еволюційне програмування (Evolutionary Programming).

1.4 Генетичні Алгоритми

ГА натхненні природною еволюцією та генетикою і використовують популяцію індивідів (розв'язків), які еволюціонують у часі, для пошуку

оптимальних або близьких до оптимальних розв'язків складних задач. Зображення розв'язків у вигляді двійкових рядків, які можна порівняти з хромосомами, є ключовою особливістю ГА. Кожен байт у рядку можна розглядати як ген, який представляє певну особливість розв'язку.

Хоча ГА зазвичай використовують двійкове кодування, вони не обмежуються цим форматом. Так, наприклад, у генетичному програмуванні використовують коди програм як об'єкти у популяції та кодують ці програми у дерева, або ж репрезентують як послідовності команд [?, ?]. Таким же чином, у ГА можна використовувати кодування з дійсними числами. Тепер кожен ген на хромосомі можна закодувати як число з плаваючою комою. Це особливо корисно в задачах оптимізації зі змінними з дійсними числами. Він забезпечує більш пряме зображення простору розв'язків і може підвищити точність розв'язку. Фенотип буде зображатися більш прямо у генотип: зменшиться спотворення простору фенотипів у простір генотипів.

Можна подумати, що через таке кодування рішень ГА стає стратегією еволюції, які представляють рішення саме через дійсні числа. Але це не є так. Наведемо аргументи чому так.

1) Основна відмінність між ГА та ЕС полягає в тому, як вони поводяться з генетичними операторами. У той час як ГА часто використовують як кросинговер, так і мутацію, причому кросинговер часто є домінуючим оператором, ЕС, як правило, роблять більший акцент на мутації.

2) ЕС часто використовують методи самоадаптації, де сила мутації (кількість змін, спричинених мутацією) також розробляється як частина рішення.

3) При зміні кодування кожного гена на число із плаваючою точкою не змінює поняття генотипу чи гена. Фактично репрезентація чисел через рядки аналогічна до реалізації арифметики у комп'ютерних системах. При зміні кодування змінюються також алгоритми рекомбінації, щоб підлаштуватися під конкретний тип вигляду генотипа.

Генетичні алгоритми та еволюційні стратегії — це дві різні точки в спектрі еволюційних алгоритмів, кожна з яких має свій власний набір якостей і переваг. Використання різних методів кодування, таких як кодування з плаваючою комою, і зміщення акценту на генетичні оператори підкреслюють універсальність і адаптивність цих методів. Вони пристосовуються до поставленого завдання, що робить їх важливими інструментами в оптимізації та машинному навчанні.

1.5 Критерії збіжності та завершення в еволюційних алгоритмах

В еволюційних алгоритмах збіжність означає момент, коли алгоритм знайшов відповідне рішення або досяг точки, де подальші ітерації не дають помітного поліпшення відповіді. Це може вимірюватися шляхом відстеження якості найкращого рішення або середньої якості рішень популяції з плином часу. Коли ці показники більше не покращуються після кількох поколінь, вважатимемо, що алгоритм збігся.

На противагу цьому, критерії завершення - це умови, які визначають, коли алгоритм має бути завершений. Максимальна кількість поколінь є типовою умовою завершення роботи. Іншими критеріями можуть бути міра різноманітності розв'язків або обмеження на обчислювальні ресурси, наприклад, максимальний час роботи процесора.

1.6 Огляд попередніх робіт

З моїх спостережень у області моделювання розвитку організмів у певному середовищі я виділив би два джерела: статтю Натана Руя [?] та DERL.

1.6.1 Проста реалізація Натана Руя

Натан Руй написав детальну статтю з еволюційних обчислень, зосередившись на реалізації простої 2D-симуляції організмів. На його ресурсі міститься детальний опис як організми еволюціонують і пристосовуються до навколишнього середовища, використовуючи генетичний алгоритм для оптимізації [?].

Основними в симуляції є два об'єкта: організм та їжа. Організм включає в себе нейронну мережу та функції для оновлення його курсу, швидкості та положення. Коли організм ініціалізується вперше, його положення, курс, швидкість, прискорення та ваги нейронної мережі генеруються випадковим чином. Їжа — це простий об'єкт, який містить координати x та y та уособлює певну енергетичну цінність. Ця енергетична цінність безпосередньо впливає на виживання організму в середовищі.

1.6.2 Deep Evolutionary Reinforcement Learning

Проект Deep Evolutionary Reinforcement Learning (DERL) досліджує взаємозв'язок між складністю навколишнього середовища, еволюцією морфології та здатністю до навчання інтелектуального управління. Дослідники пропонують обчислювальну платформу під назвою DERL, яка може розвивати різноманітні морфології агентів для навчання складним завданням переміщення та маніпулювання в складних середовищах. DERL імітує переплетені процеси дарвінівської еволюції протягом поколінь для пошуку морфологій і використовує навчання з підкріпленням протягом життя для вивчення інтелектуальної поведінки на основі низькорівневої егоцентричної сенсорної інформації. DERL також використовує розподілений асинхронний еволюційний пошук для розпаралелювання обчислень, що лежать в основі навчання [2].

Дослідження підкреслює важливість еволюційних морфологій для полегшення навчання складних завдань. Однак створення штучних втілених агентів з добре адаптованими морфологіями в різноманітних, складних середовищах є складним завданням через подвійні труднощі пошуку серед комбінаторно великої кількості можливих морфологій і обчислювального часу, необхідного для оцінки придатності через навчання впродовж життя. Ці проблеми призвели до того, що попередні роботи були зосереджені на більш простих завданнях і обмежених морфологічних просторах.

Висновки до розділу 1

Наприкінці кожного розділу ви повинні навести коротенькі підсумки по його результатах. Зокрема, для оглядового розділу в якості висновків необхідно зазначити, які задачі у даній тематиці вже були розв'язані, а саме поставлена вами задача розв'язана не була (або розв'язана погано), тому у наступних розділах ви її й розв'язуєте.

Якщо ваш звіт складається з одного розділу, пропускайте висновок до нього – він повністю включається в загальні висновки до роботи

2 ЗАСТОСУВАННЯ ГЛИБИННОГО НАВЧАННЯ У МОДЕЛЮВАННІ РОЗВИТКУ ПРОСТИХ ОРГАНІЗМІВ

Проілюструємо симбіотичний зв'язок між нейронними мережами та генетичними алгоритмами у моделюванні еволюції видів в екосистемі. Спочатку ми розглянемо нейронні мережі, які є обчислювальними системами, натхненними біологічними нейронними мережами, що складають мозок тварин. Ці мережі в даному контексті слугують мозком наших модельованих організмів, отримуючи інформацію про навколишнє середовище, обробляючи її та обираючи відповідні реакції.

Обрали генетичні алгоритми замість традиційних підходів до навчання, таких як градієнтний спуск. Кожен індивід в нашій популяції представляється набором ваг та зсувів нейронної мережі, які функціонують як ДНК та фенотип організму. Ця нова ідея генетичного представлення нейронних мереж лежить в основі наших зусиль, а ефективність мережі у реагуванні на стимули навколишнього середовища слугує ключовим показником придатності для кожного індивіда.

Дана модель намагається імітувати процеси адаптації, виживання та еволюції, використовуючи принципи глибокого навчання та еволюційні алгоритми. Мета — створити надійну модель еволюції організмів у часі, яка враховує не лише пристосованість окремих організмів, але й динаміку екосистеми, в якій вони живуть.

2.1 Нейронні мережі

Штучні нейронні мережі створені за подібністю до людського мозку, який еволюціонував протягом мільйонів років. Нейронна мережа складається з багатьох взаємопов'язаних нод, які представляють собою штучні нейрони. Всі ноди зі зв'язками між ними утворюють одну

структуру обробки інформації.

Ці мережі є частиною ширшого предмету машинного навчання і досягають успіху у виявленні закономірностей та прогнозуванні на основі складних, багатовимірних даних.

Кожен нейрон отримує вхідні дані, множить їх на певні ваги, застосовує зсув, а потім пропускає через функцію активації. Математична операція нейрона може бути виражена наступним чином:

$$z = wx + b$$

де x - вхідне значення, w - вага, b - зсув, і z - зважена сума. Після отримання зваженої суми застосовується функція активації f , результат якої є результатом роботи нейрону:

$$a = f(z)$$

Ваги та зсуви нейронної мережі є основними параметрами. Ваги визначають силу впливу вхідних даних на вихідні, тоді як зсуви дозволяють переміщати функцію активації по горизонталі.

Функція активації нейрона визначає його вихід при наявності входу або комбінації входів. У нейронних мережах використовуються різні типи функцій активації.

1) **Сигмоїдна функція** стискає вхідні значення в діапазоні від 0 до 1. Вона часто використовується в задачах бінарної класифікації. Сигмоїдна функція активації має наступну формулу:

$$f(z) = \frac{1}{1 + e^{-z}}$$

2) **ReLU** (Rectified Linear Unit) стала більш вживаною функцією активації для нейронних мереж, оскільки при ній мережа навчається швидше та зазвичай видає кращі результати. Для позитивних значень

повертається значення безпосередньо, а для негативних нуль.

$$f(z) = \max(0, z)$$

3) **Гіперболічна функція** також видає результат у певній межі, а саме в межах від -1 до 1. Зазвичай вона використовується саме в прихованих шарах нейронних мереж.

$$f(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

Кожен вузол в шарі пов'язаний з кожним вузлом в наступному шарі. Цим зв'язкам присвоюються ваги, які змінюються в міру того, як мережа навчається на основі даних. Вхідний рівень є початковим рівнем та приймає вхідні дані. Останній шар - це вихідний шар, який відповідає за отримання кінцевого результату. Один або декілька прихованих шарів виконують обчислення і перетворення даних між ними.

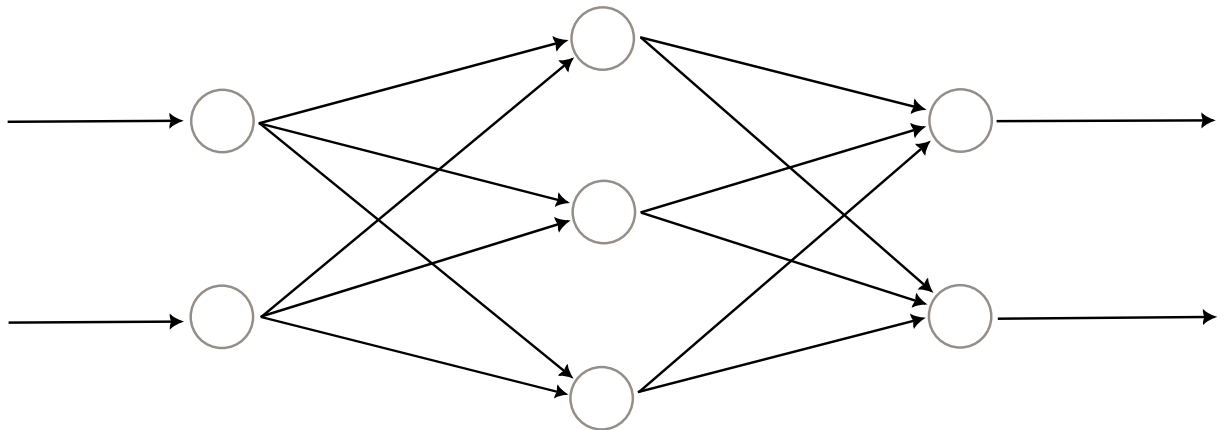


Рисунок 2.1 – Приклад структури нейронної мережі

Щоб ефективно обчислювати вихід нейронної мережі, ми векторизуємо обчислення для нейронів у шарі. Назвемо вагову матрицю

W , вектор зсуву b , вхідний вектор A_p та функцію активації g для даного шару. Вихід шару, A , можна обчислити наступним чином:

$$Z = WA_p + b$$

Глибинне навчання — це підгалузь машинного навчання, у якій застосовуються нейронні мережі із декількома прихованими шарами. Такі мережі також відомі як "глибокі" нейронні мережі.

2.2 Навчання нейронної мережі за допомогою генетичних алгоритмів

У контексті проекту нейронні мережі використовують генетичні алгоритми для навчання замість використання такого типового методу, як градієнтний спуск. Кожна особа у популяції представляє собою набір ваг та зсувів, які кодують геном та фенотип організму. Продуктивність роботи цієї нейронної мережі, як механізм реагування організму на навколишнє середовище, є основним об'єктом для оцінки придатності кожної особини.

У даному випадку використання генетичних алгоритмів для навчання нейронних мереж організмів має декілька переваг:

1) **Глобальний пошук.** Оскільки ГА підтримують популяцію рішень і використовують механізм мутації, вони менш схильні до зациклення на локальних мінімумах.

2) На відміну від градієнтних підходів, генетичні алгоритми не потребують знання похідних функції.

3) ГА забезпечують природний метод кодування і розвитку як архітектури, так і вагових коефіцієнтів у питаннях, де архітектура нейронної мережі (кількість шарів, кількість нейронів на шар і т.д.) також є частиною проблеми оптимізації.

Генетичне представлення нейронних мереж є унікальним

компонентом цієї роботи. «Геном» кожної істоти представлений у вигляді списку всіх вагових коефіцієнтів нейронної мережі. Ці ваги керують поведінкою організму і можуть змінюватися в результаті еволюційних процесів таких як мутації та кросинговеру. Кодування дійсними числами в генетичних алгоритмах представляє нейронні мережі організмів у вигляді вектора дійсних чисел, що надає їм значення генома для процесу еволюції.

Така ідея чудово ілюструє можливості глибокого навчання та генетичних алгоритмів. Глибинне навчання дозволяє простим організмам приймати розумні рішення щодо свого оточення. Генетичні алгоритми, з іншого боку, дозволяють цим організмам еволюціонувати з покоління в покоління, зрештою підвищуючи їхній інтелект та адаптивність.

2.3 Процес моделювання розвитку простих організмів

Нейронна мережа отримує вхідні дані про навколишнє середовище, а на виході - діяльність організму. Повинен існувати певний механізм «зору» таких організмів, що виводив би інформацію про середовище через призму певних обмежень та факторів. Кількість таких механізмів для порівняння не є обмеженою.

Функція пристосованості у генетичному алгоритмі, яка впливає на виживання та розмноження організму, може базуватися на рівні енергії організму та співвідношенні спожитої їжі до пройденої відстані. Це сприяє еволюції видів, які можуть ефективно добувати їжу, витрачаючи при цьому найменшу кількість енергії. Але можна використовувати і простішу реалізацію функції пристосованості, як поточний рівень енергії в організмі.

Модель також повинна включати в себе елітизм та усікаючий відбір, для роботи механізмів рекомбінації та мутації. Елітизм гарантує, що найкращі особини з одного покоління передаються наступному, підтримуючи хороші рішення, тоді як усікаючий відбір видаляє частину

популяції з найнижчими показниками пристосованості перед розмноженням, прискорюючи еволюційний процес.

Крім того, модель використовує ряд операторів кросинговеру та мутації у генетичному алгоритмі: SBX рекомбінація, арифметичний кросинговер та змішаний кросинговер (BLX-0.5) — це методи змішування "геномів" двох батьківських організмів для отримання нащадків. Мутація використовується для внесення випадкових змін у "геном" за допомогою гауссових, рівномірних і нерівномірних мутацій. Ці різні оператори урізноманітнюють популяцію, що дозволяє легше досліджувати простір розв'язків.

Глибинне навчання використовується у проекті для розробки "механізму мислення" для базових організмів. Глибинні нейронні мережі приймають дані про навколишнє середовище як вхідні дані, фільтрують їх через численні взаємопов'язані шари для вилучення та вивчення складних властивостей і видають вихідні дані, які відповідають діяльності організму. Завдяки цьому процесу організм може успішно сприймати навколишнє середовище і приймати розумні рішення щодо свого наступного кроку.

За допомогою методу дискової вибірки Пуассона [1] моделюється поява їжі та організмів в екосистемі. Цей алгоритм генерує випадково розподілені точки, зберігаючи мінімальну відстань між ними, імітуючи природну дисперсію ресурсів та істот у середовищі існування.

По суті, модель являє собою синтез глибокого навчання (представленого нейронними мережами) з еволюційними алгоритмами (представленими генетичним алгоритмом). Мета полягає в тому, щоб побудувати базову, але надійну модель розвитку організму в часі, керуючись принципами виживання, адаптації та еволюції.

Висновки до розділу 2

Наприкінці розділу знову наводяться коротенькі підсумки.

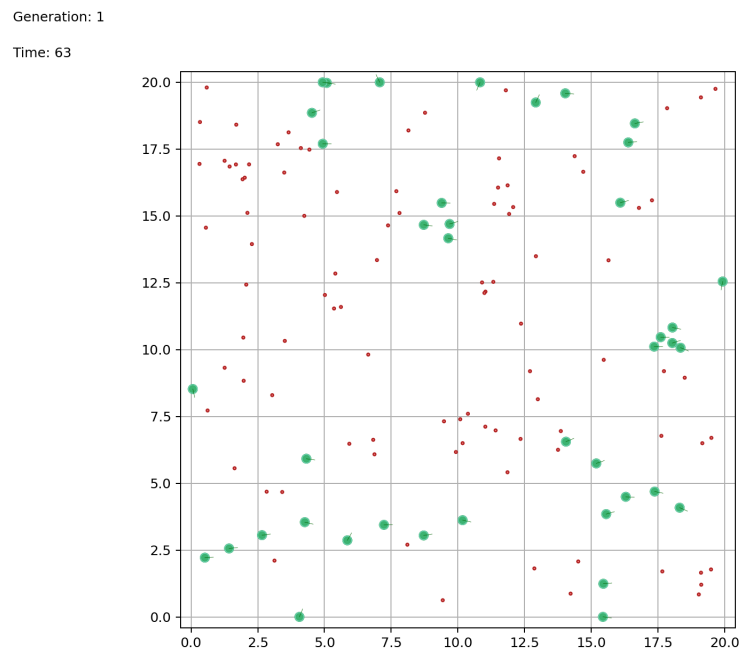
3 РЕАЛІЗАЦІЯ МОДЕЛІ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

Щось про розділ

3.1 Середовище

Як основне середовище для розвитку організмів було створено просту реалізацію двовимірного обмеженого неперервного (в обчислювальному сенсі) середовища. Його ініціалізація проводиться низкою параметрів, таких як розміри середовища, розмір та енергетична цінність їжі, частота появи їжі, та інші. Середовище відповідає за низку процедур для моделювання: обробка результатів організмів, обробка колізій об'єктів, оновлення позицій об'єктів та моделювання плину часу в середовищі.

Рисунок 3.1 – Візуалізація середовища



Для обробки колізій та ефективного зберігання об'єктів обрано просторову структуру даних R-Дерево (R-Tree). Ця структура зазвичай використовується для індексації багатовимірної інформації, такої як географічні координати, прямокутники або багатокутники. Основною ідеєю цієї структури є ідея групування сусідніх об'єктів для представлення їх за допомогою мінімального обмежувального прямокутника на наступному вищому рівні дерева. Тому і має назву R-Tree, як дерево прямокутників. Ця структура є корисною для практичних задач із просторовою взаємодією. В контексті Python існує бібліотека `rtree`, що надає інтерфейс до вже реалізованої бібліотеки на C, що встановлюється на систему. Таким чином ми забезпечені достатньою швидкістю пошуку, зміни та інших маніпуляцій об'єктів у просторі.

Їжа грає ключову роль у процесі виживання та розвитку організмів, надаючи потрібний ресурс, що є розподіленим по середовищу. Сам об'єкт їжі є точкою з певним числом, що задає розмір цієї частинки. Кожна ця частинка має певну кількість енергії, що задається константно при ініціалізації середовища. Сама їжа у середовищі з'являється за ймовірнісним процесом. На старті середовища задається параметр для експоненційного розподілу, за яким і з'являються частинки у рівномірно вибраній точці середовища. Також можливо обрати варіант появи їжі лише у певних областях простору. Це зближує нас із реальною моделлю, оскільки природні ресурси, такі як їжа, часто розташовані у скупченнях через умови навколишнього середовища. Цього можна досягти за допомогою такого методу, як дискова вибірка Пуассона [1].

3.2 Організми

Об'єкт організму репрезентує модель біологічного індивіда, що має певний набір характеристик: геном, розмір, швидкість руху, прискорення, напрямок, енергетичний рівень, вік.

Геном організму задається парами матриць, що слугують вагами та

упередженістю нейронної мережі. Сам клас самостійно оперує цією структурою. Єдина річ, яку потрібно задавати тут це шари нейронної мережі та кількість нейронів у них. Більша кількість шарів та нейронів призводить до більшого геному.

Кожен організм може думати за допомогою цієї мережі. Результат його праці — це команда на пересування. Кількість вихідних даних з мережі організму повинна задаватися типом середовища моделювання. Для тривимірного простору ми можемо задати шість ступенів свободи. Все ж для поточної реалізації двовимірного простору достатньо задати прискорення та зміну кута напрямку.

Але кожна команда впирається у фізичні обмеження цього організму. Ми обмежуємо його пересування у середовищі задаючи максимальну можливу швидкість, прискорення та зміну напрямку. Це є логічним, оскільки і в реальному житті будь-який організм не може змінити свої фізичні характеристики в одну мить. А в моделюванні з дискретним часом нам важливо зберегти якусь подібність до реальності.

На швидкості організм не повинен вміти швидко змінювати напрям. Саме тому додаємо гальмівний ефект від зміни напрямку руху:

$$v^{(t+1)} = \max(-v_{max}, \min(v_{max}, v^{(t)} + a)) \cdot \frac{1}{1 + d}$$

На обробку інформації про зовнішній світ у організма повинна якимось чином змінюватися енергія. так ось і буде вона змінюватися за простим законом. вводимо коефіцієнт зміни енергії μ та будемо задавати зміну енергії так:

$$E^{(t+1)} = E^{(t)} - \mu \cdot |\bar{o}|$$

Де \bar{o} позначаємо як результат роботи нейронної мережі.

Організм повинен якимось чином сприймати навколишнє середовище. У реалізації Натана Руя [5] кожен організм на вхід приймав відстань та напрям до найближчої частинки їжу. Ця модель не зовсім відповідає дійсності, оскільки прорахунок найближчої цілі, напрям та відстань не

отримується просто так. Як і в житті, ми не бачимо точну відстань до предметів, а приблизно оцінюємо ці відстань за допомогою орієнтування у просторі, напруженні м'язів ока або відносній оцінці. Тому потрібно надати найефективніше найближче сприйняття до реальності.

Наразі нейронні мережі сприймають зображення як матрицю з закодованими значеннями кольорів. По суті, зображення є проекцією об'єктів у трьох вимірах, тому й наш механізм зору повинен надавати певну проекцію об'єктів на матрицю. Ідеалістично, фенотип організму не повинен залежати від середовища, у якому він розвивається, але й зір має тісний зв'язок із середовищем та ідейно пов'язаний з об'єктом, у якому він знаходиться. Також наші організми можуть розвиватися як у трьох, так і в двох вимірах. Тому краще відокремити механізм зору від організму. Це дозволяє тестувати будь-яку кількість таких імплементацій, навіть використовуючи один процес моделювання. Кількість нейронів у вхідному шарі мережі буде визначена таким механізмом зору.

Виберемо стратегію, яка повторно використовує концепцію цифрових зображень. Світ навколо організму проектується на матрицю певного розміру. Заповнення цієї матриці також залежить від конкретного способу реалізації ідеї. Зменшимо рівень абстракції «зору» до рівня кодування інформації навколо об'єкта в матрицю фіксованого розміру. Логічно використовувати набір позицій об'єктів у просторі та їх фізичні властивості для створення матриці. Кожен елемент матриці кодує напрямок і відстань відносно «голови» організму, а значення елемента можна подавати як тип об'єкта, де для їжі ми ставимо одне константне значення, а для інших організмів на шляху цільового — інше.

3.3 Кодування

Для генетичних алгоритмів використовують різні методи кодування, щоб представити розв'язки задачі в структурі хромосоми або геному. Типовими прикладами кодувань є: двійкове кодування,

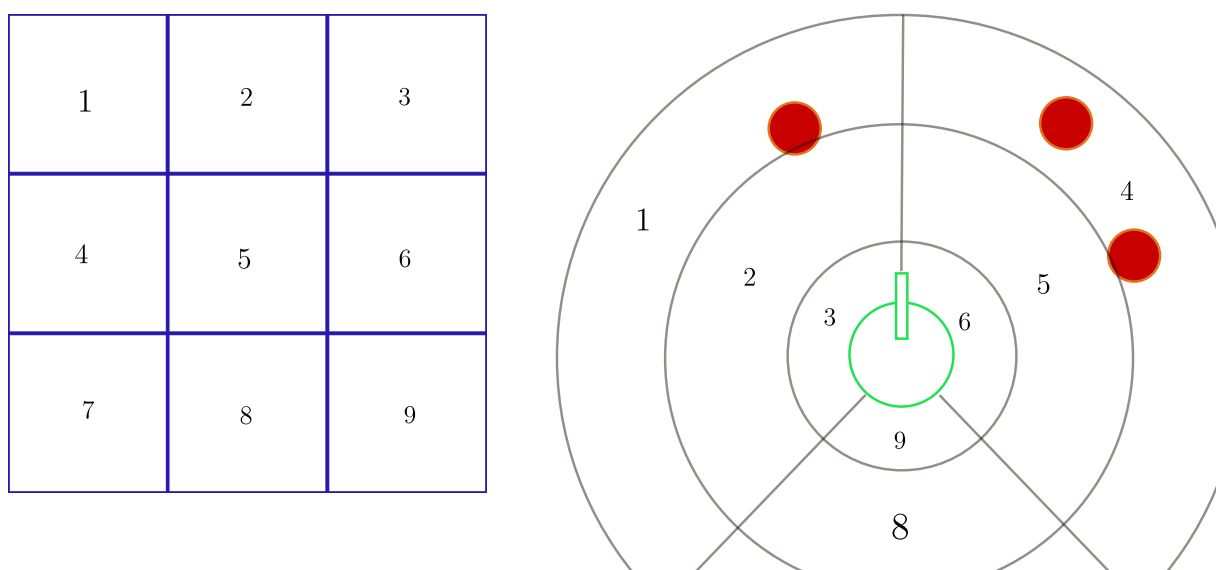


Рисунок 3.2 – Наповнення матриці для механізму зору

цілочисельне кодування, кодування дійсними числами, кодування перестановками та кодування значеннями.

Двійкове кодування представляє рішення за допомогою двійкових цифр (0 і 1). Якщо простір задачі вимагає дискретних величин, краще використовувати цілочисельне кодування, оскільки воно використовує цілі числа. Кодування перестановок підходить для проблем впорядкування або маршрутизації, оскільки воно передбачає розміщення речей або значень у певному порядку. У складних проблемних просторах, де певні стани або атрибути повинні бути представлені явно, кодування значень, також відоме як пряме кодування, передбачає безпосередній запис значень рішення.

У цій реалізації генетичного алгоритму для представлення геному організму можна конфігурувати різні можливі кодування. Це можливо завдяки правильній структурі коду, що дає можливість швидко змінювати параметри моделі.

Але найкраще використати кодування дійсними числами. Перевага кодування в дійсних числах полягає в тому, що воно безпосередньо представляє неперервні змінні, що робить його особливо придатним для

питань, пов'язаних з оптимізацією неперервного простору, таких як оптимізація ваг нейронних мереж.

Ваги та упередженність нейронної мережі безпосередньо кодуються в геномі організму за допомогою цього кодування в дійсних числах. Клас `RealValued` дає можливість кодувати та декодувати ваги організму в одновимірний масив геному. Для цього ваги та зсуви вирівнюються та об'єднуються. Для декодування сплющені масиви повертаються до початкових розмірів.

Завдяки такому кодуванню можна ефективно шукати у неперервному просторі оптимальні параметри нейронної мережі організму. Через те, що ваги та упередженність за своєю природою є дійсними числами, це забезпечує простий та ефективний метод оптимізації нейронних мереж. Імітуючи еволюцію біологічних організмів і використовуючи цей процес для вдосконалення моделей машинного навчання, ця методика втілює суть біологічно натхненного навчання.

3.4 Генетичні оператори

Генетичні оператори грають величезну роль у зміні поведінки організмів та продуктивності генетичного алгоритму. Вони дозволяють вивчати можливі рішення проблеми у фазовому просторі рішень або ж перевикористати поточні варіанти для отримання більш кращого. Вибір і конфігурація цих операторів є важливою частиною проектування генетичного алгоритму.

Мутація вносить невеликі випадкові зміни в геноми індивідів, щоб підтримувати її різноманітність. Було запрограмовано різні типи мутації.

`NonUniformMutation` клас реалізує нерівномірну мутацію, яка змінює гени на основі функції, що зменшується з часом, сприяючи експлуатації, а не дослідженню в подальшому процесі роботи алгоритму. Це дозволяє на кінцевих ітераціях сфокусуватися на отриманні більш точних рішень, а ніж дослідженню простору.

`GaussianMutation` та `UniformMutation` є Гауссовою мутацією та рівномірною мутацією, які вносять зміни з нормального розподілу або з рівномірного розподілу відповідно.

Селекція — це процес вибору особин, або батьків, які дадуть потомство у наступному поколінні. Клас `TruncationSelection` реалізує усічений відбір, який обирає найкращих n особин на основі їхніх значень пристосованості. Цей підхід є простим і ефективним, гарантуючи, що для розмноження будуть обрані найбільш пристосовані особини.

Кросинговер — ще один важливий генетичний оператор. Він дозволяє обмінюватися генетичним матеріалом між батьківськими особинами, створюючи таким чином потомство.

Клас `SBXCrossover` реалізує імітований двійковий кросинговер (SBX), який імітує поведінку одноточкового кросинговеру в генетичних алгоритмах з двійковим кодуванням в контексті кодування з дійсними значеннями. Він генерує нащадків ближче до батьків [3, 2].

Клас `BLXCrossover` реалізує змішаний кросовер (BLX), який створює нащадків у діапазоні, визначеному генами батьків і пропорцією, що дозволяє проводити більш значні дослідження [3].

Клас `ArithmeticCrossover` генерує нащадків, використовуючи лінійну комбінацію генів батьків [4].

Клас `UniformCrossover` реалізує простий рівномірний кросинговер, де кожен ген нащадка випадковим чином вибирається від одного з батьків [4].

3.5 Опис інструментів та збір даних під час моделювання

Механізм еволюції реалізовано з використанням об'єктно-орієнтованого підходу. По суті, це конвеєр, який перетворює популяцію організмів на наступне покоління. Цей процес відбувається в основному за допомогою чотирьох ключових кроків: відбору, проходження процесу елітизму, кросинговеру та мутації.

Також важливим параметром є ввімкнення процесу помирання організмів при досягненні критичної області енергії. Таким чином ми можемо отримати стаціонарний генетичний алгоритм, який не потребує переходів між популяціями, змінюючи час на отримання наступної популяції та кількість осіб у відборі. Його основна перевага це постійний процес еволюції, помирання індивідів та оптимізації по використанню пам'яті.

Було імплементовано комплексну бібліотеку для зручної роботи з дослідженням адаптації організмів до середовища. Підхід при її розробці мав на меті легку параметрицію для впровадження швидких експериментів по моделюванню життя організмів.

Також варто зазначити, що будь-яка частина програми може бути вільно замінена на іншу реалізацію. Тобто більш складні середовища можуть бути легко спровадженні у цей продукт через його модульність. Для дослідження цього проекту було вирішено обмежитися одним середовищем та одним типом організмів. Це дозволило швидко провести аналіз даних, отриманих за допомогою цієї моделі.

Сам же аналіз повинен здійснюватися по певних даних, які мають бути записані під час процесу моделювання. Такий запис повинен бути швидким по часу та займати мало постійної пам'яті, тому на противагу запису в **CSV** файли було обрано **Parquet** тип. Він має стовпчастий формат файлів для зберігання даних, націлений на обробку великих масивів. Також забезпечуються ефективні алгоритми стиснення та кодування.

Для аналізу застосовувалися Jupyter Notebooks як середовище виконання коду. У них відбувалося зчитування даних та візуалізація за допомогою бібліотеки для візуалізації даних **plotly**. Для більш адаптивного аналізу результатів багатьох запусків моделі із різними параметрами використаний фреймворк **dash** для побудови веб-додатків для візуалізації даних.

3.6 Метрики

Для аналізу поколінь організмів цілком розумно було б використати певний набір метрик, які б показали як поведуть себе організми у середовищі, якою є їхня різноманітність та їх рівень розвитку. Використання цих метрик дасть можливість проводити більш обґрунтований та деталізований аналіз.

Однією з таких метрик є пристосованість, що у даній реалізації є не що інше як рівень енергії у організмі. Він надає інформацію про те, наскільки ефективно організм використовує доступні йому ресурси та його здатність адаптуватися до змін у навколишньому середовищі.

Також ми можемо задати метрику максимального та мінімального рівня енергії, що дасть нам більш повну інформацію про різноманітність популяції у використанні ресурсів.

Іншою є метрика співвідношення спожитої їжі до кількості рухів організму. Ця метрика допоможе нам зрозуміти енергоефективність організму, показуючи, наскільки ефективно він використовує отримані від їжі ресурси для своєї активності. Ця метрика обчислюється як відношення кількості рухів до кількості спожитих частинок їжі.

Metrics formulas

3.7 Аналіз впливу параметрів по метрикам

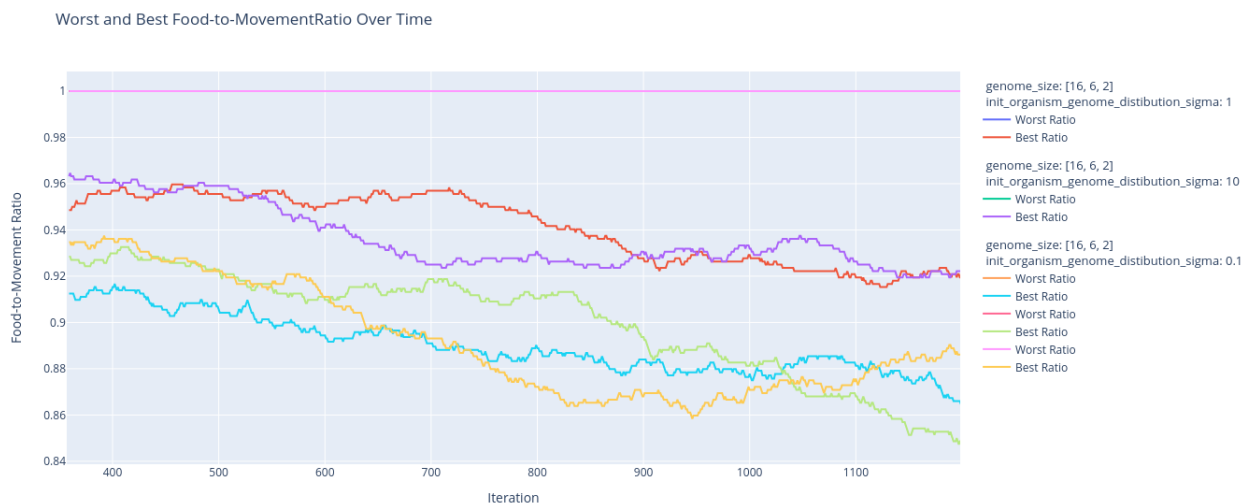
Дана модель може бути параметризована по:

- 1) Розмірам середовища
- 2) Тривалості одного покоління
- 3) Типу зору організма
- 4) Кількості енергії у частинках їжі та параметрам розподілу для генерації цих частинок
- 5) Швидкості втрати організмами енергії при русі

- 6) Методу відбору, мутації та рекомбінації
- 7) Числу осіб, що переходять у наступну популяцію (елітизм)
- 8) Розміру генома, що фактично означає кількості прихованих шарів та нейронів у них
- 9) Кількості їжі, на старті
- 10) Чи помиратимуть організми від нестачі енергії
- 11) Параметрам нормального розподілу для генерації початкових ваг нейронної мережі

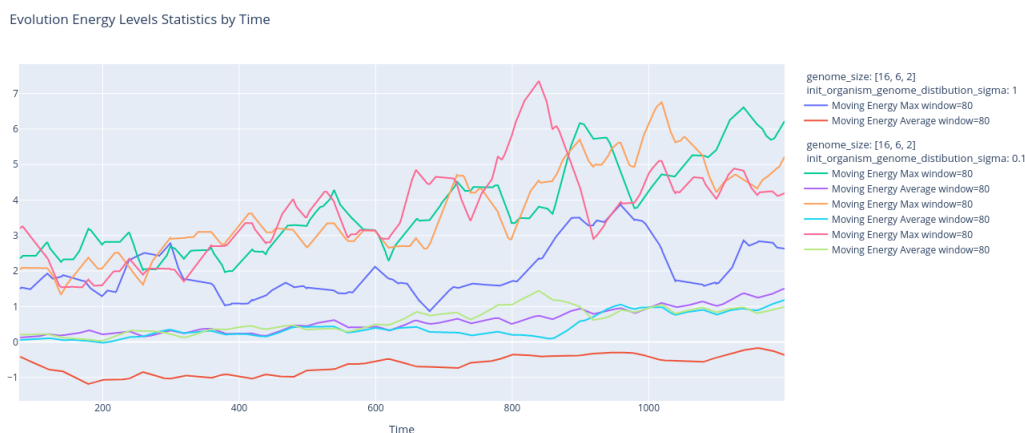
Спочатку спробуємо відшукати кращі параметри для розподілу Гауса, за допомогою якого генеруються початкові ваги нейронних мереж організмів. Спочатку взято три варіанти для стандартного відхилення: 10, 1 та 0.1. Математичне сподівання ж залишимо на рівні 0.

Рисунок 3.3 – Отримані відношення спожитої їжі до руху при пошуку кращого стандартного відхилення для генерації ваг



З отриманого графіку 3.3 відношення спожитої їжі до руху бачимо помітну різницю між організмами, що використовували 0.1 як стандартне відхилення, та іншими. Також середній рівень енергії на графіку 3.4 у осіб з 0.1 стандартним відхиленням значно відрізняється від осіб з іншим параметром: середня енергія значно вища, більша за нуль та зростає.

Рисунок 3.4 – Рівні енергії при пошуку кращого стандартного відхилення для генерації ваг



Отже область простору, де ми отримуємо значення геномів зі стандартним відхиленням у 0.1 надає кращий старт для життя організмів.

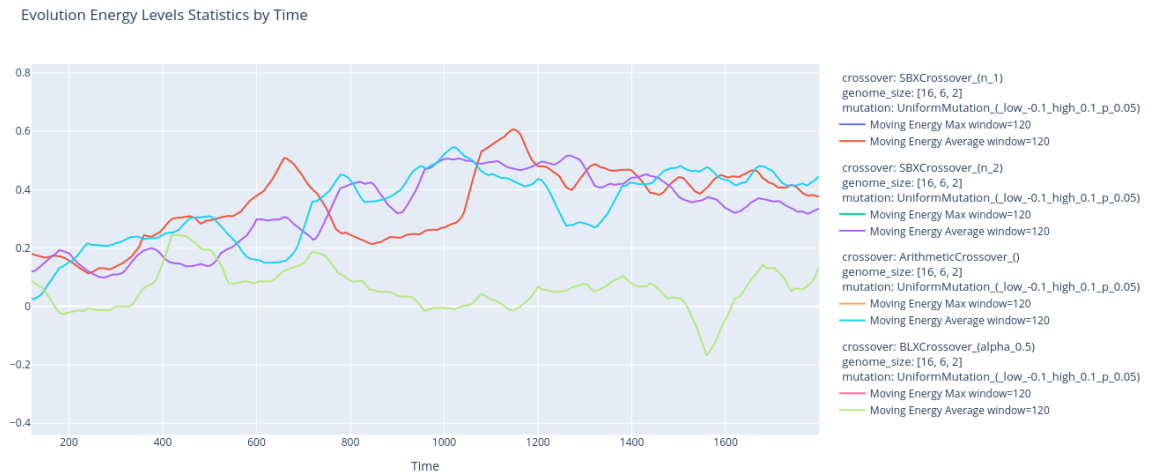
Спробуємо підібрати гарну комбінацію методу рекомбінації та мутації. Оскільки у генетичних алгоритмах рекомбінація є центральним елементом еволюції, то почнемо саме з неї. Візьмемо декілька методів: BLX рекомбінація, Арифметична та SBX із двома різними параметрами — 1 та 2. SBX рекомбінація при параметрі $n = 1$ буде мати експлуатаційний характер, що дозволить мати пошук у локальній області. Проаналізуємо графік середнього рівня енергії по всіх організмах впродовж часу.

На графіку 3.5 більше всіх виділяється саме BLX рекомбінація із параметром 0.5. Він подає гірший результат у порівнянні з двома іншими. А отже для наших цілей краще надалі використовувати SBX або ж арифметичну рекомбінацію.

Вплив мутації не є настільки суттєвим та великим. Під час попереднього моделювання не було помічено випадку, коли певний метод мутації проявив себе краще, за інші. Тому було обрано зупинитися на нерівномірній мутації через її ідею на поліпшенні неявних рішень на завершальних ітераціях алгоритму.

Яка кількість елітів та нащадків дають кращий результат?

Рисунок 3.5 – Середній рівень енергії при порівнянні методів рекомбінації



чи надає ускладнення форми організму йому переваги у життєдіяльності?

Візьмемо такий набір кількості прихованих шарів з кількістю нейронів нейронної мережі:

- (6)
- (9)
- (9, 9)
- (18, 9)

З використанням 4 секцій дистанції та 4 секцій кута для зору отримуємо 16 вхідних нейронів у кожній мережі. Також на вихід маємо видавати два значення. Як результат маємо такий набір нейронних мереж:

- (16, 6, 2)
- (16, 9, 2)
- (16, 9, 9, 2)
- (16, 18, 9, 2)

При тестуванні роботи таких мереж отримали наступний графік рівнів енергії та графік співвідношення спожитої їжі до руху:

З графіку 3.8 суттєвої різниці між діями організмів не видно. А отже навчаються вони досить однаково.

Рисунок 3.6 – Порівняння різних структур нейронних мереж організмів по максимальній енергії

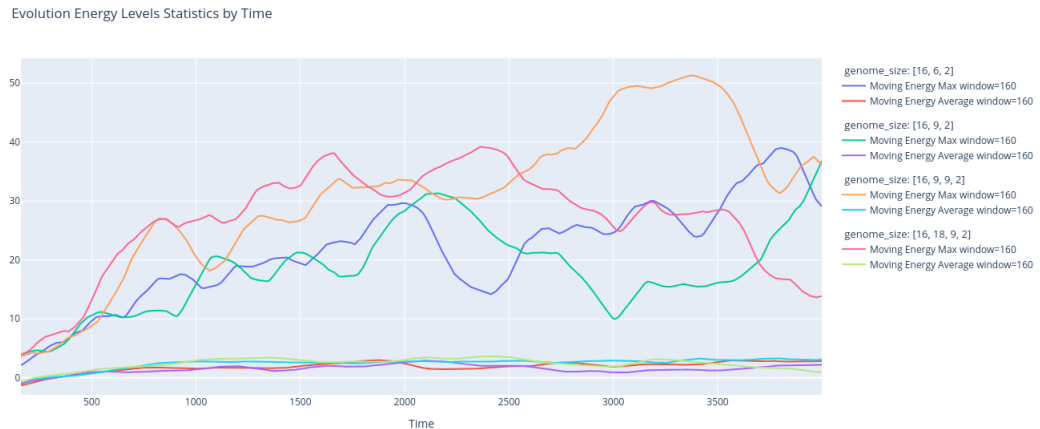
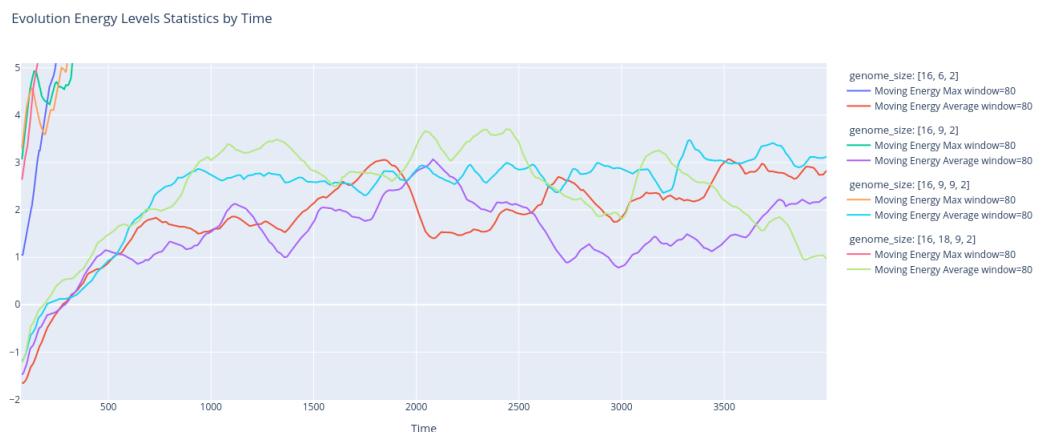


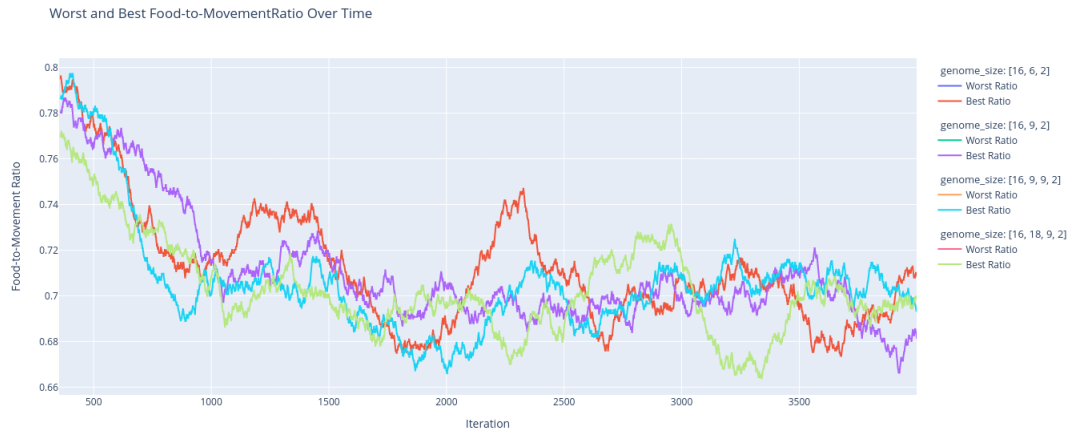
Рисунок 3.7 – Порівняння різних структур нейронних мереж організмів по середньому значенню енергії



По значенню максимальної кількості енергії 3.6 схоже, що у індивідів популяції з більш складнішою структурою існують індивіди, що здатні спожити більшу кількість їжі. Це може означати, що вони розвинули більш досконалі або ефективні методи використання енергії їжі. Така поведінка пояснюється збільшенням кількості нейронів, що могло призвести до покращення здатності вирішувати проблеми та більш ефективної адаптації до навколишнього середовища.

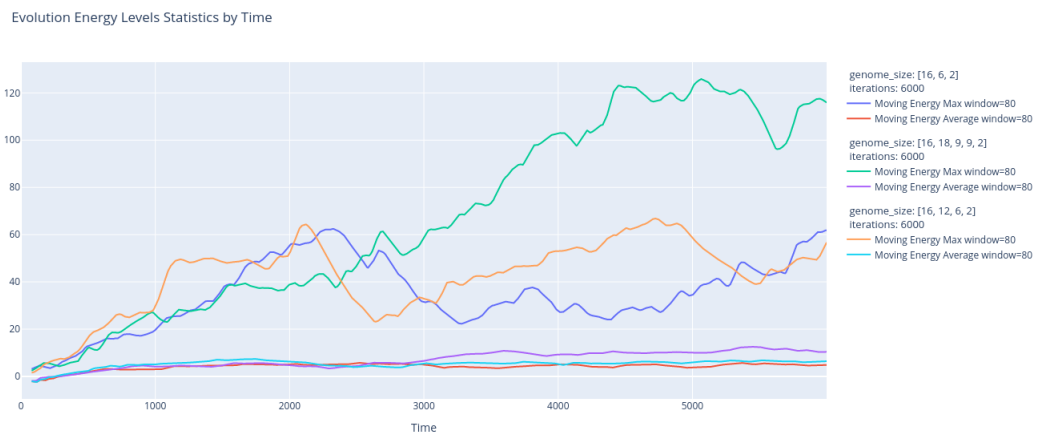
Однозначно стверджувати, який варіант організмів ми поки не можемо. Кількість нейронів для простих організмів може бути не на

Рисунок 3.8 – Порівняння різних структур нейронних мереж організмів по відношенню спожитої їжі до руху



достатньому рівні, щоб швидко отримати потрібні знання для утилізації частинок їжі. Спробуємо порівняти із більш складним організмом на рівні (16, 18, 9, 9, 2) шарів нейронної мережі.

Рисунок 3.9 – Порівняння простої та складної структури нейронної мережі організмів по енергії

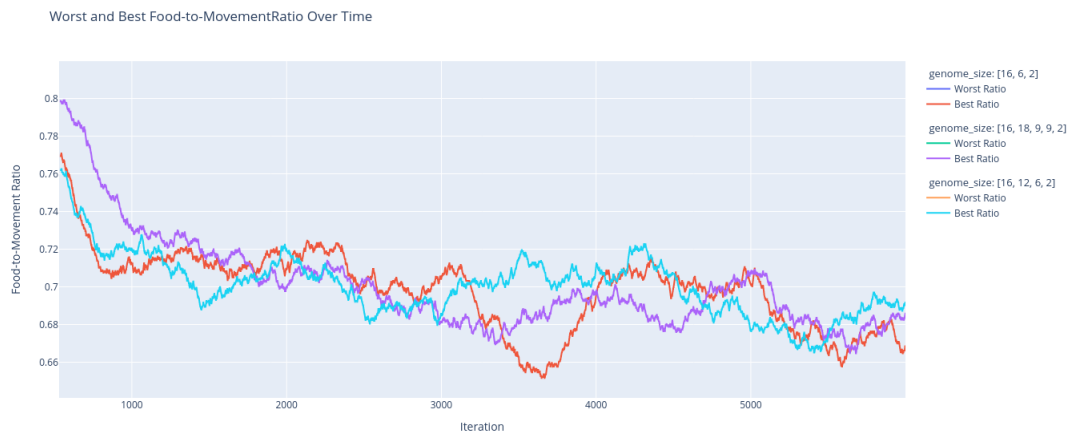


По графіку 3.9 можемо проаналізувати, що простіші структури виявилися кращими на початкових етапах розвитку організмів. Вони змогли вдосталь пристосуватися для отримання певної кількості енергії у короткостроковій перспективі. Але надалі їх розвиток сповільнився. У той

же час складна структура (16, 18, 9, 9, 2) отримувала дані про навколишній простір та зробила сильний ривок за рівнем максимальної кількості енергії, випереджаючи свої простіші аналоги. По рівню середньої енергії по популяції ця структура має також вищі показники.

На перший погляд, простіші структури отримують більше їжі пропорційно до кількості руху та тримають достатній рівень енергії для виживання на певний період. Однак з часом результати роботи як простих, так і складних систем змінюються, що свідчить про важливість навчання та адаптації в обох типах структур.

Рисунок 3.10 – Порівняння простої та складної структури нейронної мережі організмів по відношенню спожитої їжі до руху



По графіку відношення спожитої їжі до руху 3.10 чітко видно, як на початкових стадіях розвитку організмів простіша структура отримує більше їжі, по відношенню до кількості рухів, але надалі обидва варіанти структур показують перемінний результат.

Такі результати можна пояснити через структурну складність конфігурації з більшою кількістю шарів нейронної мережі. Через збільшення шарів у мережі організм містить довший геном, а отже збіжність популяції з такою структурою може бути повільнішою, порівнюючи її з популяцією з меншим геномом. Оскільки існує менше змінних і менший простір для пошуку оптимальних генетичних

комбінацій, коротші геноми з простішою архітектурою можуть забезпечити швидшу адаптацію. Як наслідок, менші структури можуть знайти ефективний спосіб споживання енергії швидше, ніж їхні складніші аналоги.

Коротший геном дозволив популяції знайти оптимальні рішення до отримання все більших запасів енергії за короткий проміжок часу. Але популяція з довшим геномом також навчається та може вмістити у геном більше інформації для реагування на різні події у їхньому житті, оскільки довший геном забезпечує більшу різноманітність генетичних комбінацій і, як наслідок, більшу можливість для довготривалої адаптації. Організмам зі складною структурою може знадобитися більше часу для визначення оптимального набору генетичних конфігурацій, оскільки їх область пошуку є значно більшою. Але після цього вони мають потенціал для адаптації до ширшого діапазону умов та оптимізації споживання енергії в більшій мірі, ніж простіші структури. При зміні правил середовища ці знання можуть зіграти свою роль.

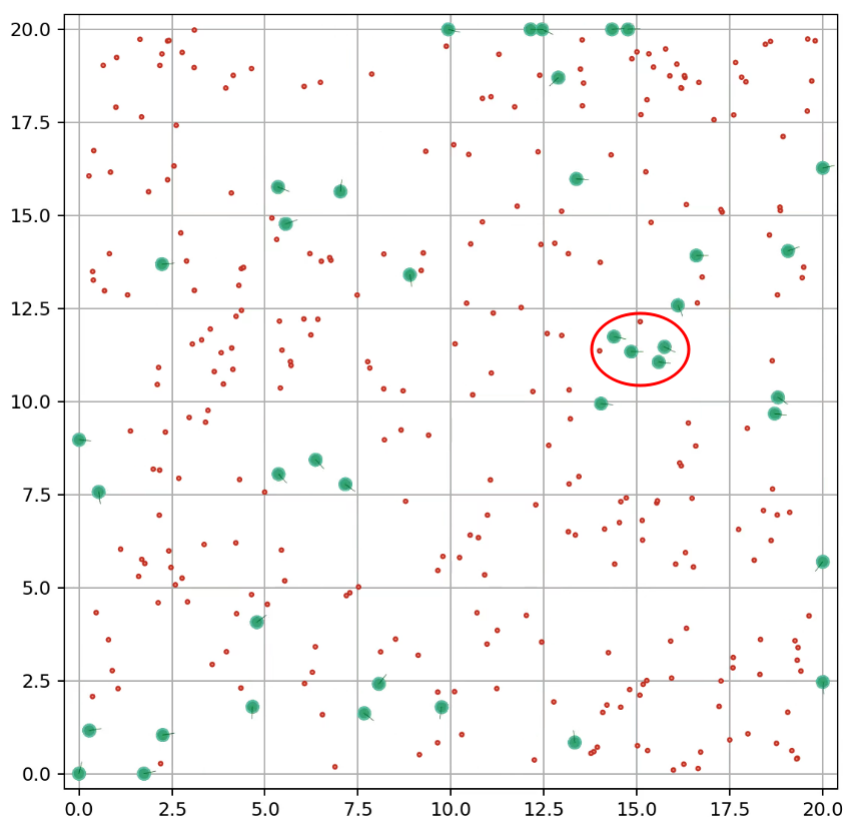
Організми можуть скупчуватися у групи за певних умов. З мисленого експерименту можна стверджувати, що подібне скупчення у групи може відбуватися лише при виконанні як мінімум однієї з двох умов:

- 1) механізм зору в організмів дозволяє їм бачити собі подібних;
- 2) під час еволюції організми можуть досягти стадії розвитку, коли їх поведінка на зовнішні подразники (частинки їжі або інші організми) буде мати схожий характер у межах групи.

Хоча й методи розпізнавання та схожа поведінка можуть відігравати важливу роль, та у формуванні групи можуть діяти й інші процеси. Наприклад, процес комунікації, спільні способи захисту або добування їжі — все може сприяти формуванню групи. Але такі процеси відносяться до більш складних моделей та реальній моделі розвитку організмів.

Звичайно, таку складну ідеалізовану модель важко реалізувати, що беззаперечно дає визнання складності реальних умов. Та все ж під час

Рисунок 3.11 – Утворення групи організмів



спостереження за розвитком організмів під час експерименту було помічено малі групи індивідів, що рухали за схожим шаблоном. Часто саме присутність аналогічного організму поряд давала поштовх змінити курс та слідувати за ним. Це надає переконливі емпіричні докази групової поведінки, навіть якщо ці угруповання не дають переваги у такому конкурентному середовищі. Часто саме такі групи помирали, оскільки пошук їжі у такому бідному середовищі вимагає змагання із усіма, навіть із учасниками групи.

Висновки до розділу 3

Модель репрезентує просту абстракцію реального механізму розвитку організмів, але не є повним еквівалентом, хоча і поділяє кілька основних концепцій. Дана модель наслідує механізми виживання та

кількості енергії: реальні біологічні організми споживають певні ресурси, що надають їм можливість виживати. Цей принцип є фундаментальним у такій моделі. Подібно до справжньої біологічної еволюції, модель використовує генетичні алгоритми для імітації концепцій природного відбору, мутацій та успадкування. Нейронна мережа є абстрактною картиною того, як реальні істоти використовують свій генетичний код для створення складних дій та функцій. Те, як нейронна мережа використовується для відтворення процесу «мислення» організму, схоже на те, як реальні тварини використовують свій мозок для взаємодії з навколишнім середовищем. Однак важливо зазначити, що спрощення та абстракції, необхідні для комп'ютерного моделювання означають, що дана модель не є ідеальним відображенням реального світу через обмеження у: сенсорах, спрощеному середовищі, відсутності росту, складності геному.

Дослідження виявило динамічний зв'язок між складністю геному, швидкістю еволюції та довгостроковою адаптацією. Підібрані гарні параметри для старту моделювання та механізми еволюції. Було показано, що організми з простішою генетичною архітектурою адаптуються швидше через менший простір рішень. Однак, незважаючи на більш повільну початкову еволюцію, індивіди зі складнішою архітектурою геному демонстрували кращий потенціал для довгострокової адаптації. Цікаво, що, незважаючи на простоту моделі, було виявлено зачатки розвитку груп у популяції.

висновки про поставленні завдання

ПЕРЕЛІК ПОСИЛАНЬ

1Robert Bridson. Fast Poisson disk sampling in arbitrary dimensions. In *ACM SIGGRAPH 2007 Sketches*, page 22, San Diego California, August 2007. ACM. Kalyan Deb, Karthik Sindhya, and Tatsuya Okabe. Self-adaptive simulated binary crossover for real-parameter optimization. pages 1187–1194, July 2007. Kalyanmoy Deb and Ram Bhusan Agrawal. SIMULATED BINARY CROSSOVER FOR CONTINUOUS SEARCH SPACE. Padmavathi Kora and Priyanka Yadlapalli. Crossover Operators in Genetic Algorithms: A Review. *International Journal of Computer Applications*, 162:34–36, March 2017. Nathan Rooy. Evolving Simple Organisms using a Genetic Algorithm and Deep Learning from Scratch with Python. <https://nathanrooy.github.io/posts/2017-11-30/evolving-simple-organisms-using-a-genetic-algorithm-and-deep-learning/>.