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 Орєхов 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 _____

Київ — 2023

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»

fill

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 Орєхов 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 попунктно)*

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

fill

Студент _____ Житкевич І.О.

Керівник _____ Орєхов О.А.

РЕФЕРАТ

Кваліфікаційна робота містить: ??? стор., ??? рисунки, ??? таблиць, ??? джерел.

У рефераті роботи ви повинні коротко (два-три абзаци) викласти, що саме було зроблено у цій роботі. Перші три речення реферату (після статистичних даних) повинні окреслити мету роботи, об'єкт та предмет дослідження. Після цього викладаються основні результати, одержані в ході дослідження.

Наприкінці анотації великими літерами зазначаються ключові слова. Ось так:

КЛЮЧОВІ СЛОВА, СИМЕТРИЧНА КРИПТОГРАФІЯ, ФІЗТЕХ НАЙКРАЩІЙ

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

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 проставляється 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 інтуітивно 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 (EA) — це алгоритм, що належить до колекції технік еволюційних обчислень, які є натхненні біологічною еволюцією.

Більшість EA 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 [1].

Об'єктом цих алгоритмів є популяції індивідів, яких ми інтерпретуємо як потенційні рішення задачі. 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 степенем 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 [1].

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 алгоримах 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 досліду 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 експлуататорським (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 [1].

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 відрізняти 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 і-ої особи, а N — загальна кількість осіб у популяції, то ймовірність відбору $P(i)$ для і-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 мешне. 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 (crossingover). 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 відбору. 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 ліпшого. Кожен з них показує себе краще у певних своїх задачах. Для створення кращого алгоритму та параметрів під нього потрібно експерментувати та досліджувати простір рішень у задачі. Саме сукупнусть процесу відбору, рекомбінації та мутації в залежності від обраних варіантів може давати різні результати. Важливо знайти тей, 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 певном 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 [1].

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 [2, 1]. 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 предтавляють 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 [3] 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 [3].

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

Щось на мові вступу.

2.1 Нейронні мережі

2.2 Навчання нейронної мережі за допомогою генетичних алгоритмів

2.3 Процес моделювання розвитку простих організмів

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.

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 вибраніч точці середовища.

3.2 Організми

Об'єкт організму репрезентує модель біологічного індивіда, що має певний набір характеристик: геном, розмір, швидкіст 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. Але кожна команда впирається у фізичні обмеження цього організму. Ми обмежуємо його пересування у середовищі задаючи максимальну можливу швидкість, прискорення та зміну напряму. 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 напряму 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 якось

змінюватися енергія. Так ось і буде вона змінюватися за простим законом. Вводимо коефіцієнт зміни енергії та будемо задавати зміну енергії так:

$$E^{(t+1)} = E^{(t)} - \overline{o}$$

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

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, 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 можливіть 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 сфокосуватися на отриманні більш точних рішень, а ніж дослідженню простору.

GaussianMutation та UniformMutation є Гауссовую 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, 4].

Клас BLXCrossover реалізує змішаний кросовер (BLX), який створює нащадків у діапазоні, визначеному генами батьків і пропорцією, що дозволяє проводити більш значні дослідження [5, 6].

Клас ArithmeticCrossover генерує нащадків, використовуючи лінійну комбінацію генів батьків.

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

3.5 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. Його основна перевага це постійний процес еволюції, помирання індивідів та оптимізації по використанню пам'яті [7].

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.

3.6 Метрики

Для аналізу поколінь організмів цілком розумно було б використати певний набів метрик, які б показали як поводять 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.

3.7 Аналіз впливу параметрів по метрикам

Розглянемо роботу метрик та порівняємо два різних кросинговера та дві конфігурації нейронної мережі. Як кросинговери візьмемо BLXCrossover та арифметичний кросинговер та візьмемо конфігурації [16, 6, 2] і [16, 12, 6, 2], тобто спробуємо порівняти більш просту та більш складну мережі.

Висновки до розділу 3

Висновки до останнього розділу є, фактично, підсумковими під усім дослідженням; однак вони повинні стостуватись саме того, що розглядалось у розділі.

Загальні висновки до роботи повинні підсумовувати усі ваші досягнення у даному напрямку досліджень.

За кожним пунктом завдань, поставлених у вступі, у висновках повинен міститись звіт про виконання: виконано, не виконано, виконано частково (І чому саме так). Наприклад, якщо першим поставленим завданням у вас іде «огляд літератури за тематикою досліджень», то на початку висновків ви повинні зазначити, що «у ході даної роботи був проведений аналіз опублікованих джерел за тематикою (...), який показав, що (...)». Окрім простої констатації про виконання ви повинні навести, які саме результати ви одержали та проінтерпретувати їх з точки зору поставленої задачі, мети та загальної проблематики.

В ідеалі загальні висновки повинні збиратись з висновків до кожного розділу, але ідеал недосяжний. :) Однак висновки не повинні містити формул, таблиць та рисунків. Дозволяється (та навіть вітається)

використовувати числа (на кшталт «розроблена методика дозволяє підвищити ефективність пустопорожньої балаканини на 2.71%»).

Наприкінці висновків необхідно зазначити напрямки подальших досліджень: куди саме, як вам вважається, необхідно прямувати наступним дослідникам у даній тематиці.

ПЕРЕЛІК ПОСИЛАНЬ

1 Luke Sean. Essentials of metaheuristics: a set of undergraduate lecture notes; Online Version 2.0. — 2. ed ed. — S.l. : Lulu, 2013. — ISBN: 978-1-300-54962-8. Slowik Adam, Kwasnicka Halina. Evolutionary algorithms and their applications to engineering problems // Neural Computing and Applications. — 2020. — Aug. — Vol. 32, no. 16. — P. 12363–12379. — online; accessed: https://doi.org/10.1007/s00521-020-04832-8 (online; accessed: 2023-06-03). Rooy Nathan. Evolving Simple Organisms using a Genetic Algorithm and Deep Learning from Scratch with Python. — online; accessed: https://nathanrooy.github.io/posts/2017-11-30/evolving-simple-organisms-using-a-genetic-algorithm-and-deep-learning/ (online; accessed: 2023-05-31).

ДОДАТОК А ТЕКСТИ ПРОГРАМ

Тексти інструментальних програм для проведення експериментальних досліджень необхідно виносити у додатки.

А.1 Програма 1

Зауважте, як змінилась нумерація.

ДОДАТОК Б ВЕЛИКІ РИСУНКИ ТА ТАБЛИЦІ

Якщо результати вашої роботи описуються величезними рисунками і таблицями (один аркуш та більше) у незліченній кількості, їх також необхідно виносити у додатки.