УДК 004.4'22 https://doi.org/10.15407/pp2021.01.016

А.Ю. Дорошенко, І.З. Ашур

ЗАСТОСУВАННЯ ЗАСОБІВ НЕЙРОЕВОЛЮЦІЇ В ТЕХНІЧНИХ СИСТЕМАХ АВТОМАТИЗАЦІЇ КЕРУВАННЯ

В роботі запропоновано нове застосування техніки нейроеволюції наростаючих топологій для розв'язування задач автоматизації керування на одному з прикладів добре відомих стандартних та популярних задач керування технічними системами, де використовується навчання з підкріпленням. Використовується відкритий інструментарій для розробки та порівняння алгоритмів навчання з підкріпленням OpenAI Gym, повноцінна реалізація з відкритим програмним кодом генетичного алгоритму нейроеволюції NEAT під назвою SharpNEAT, та проміжне програмне забезпечення для оркестрації зазначених компонентів.

Ключові слова: штучні нейронні мережі, навчання з підкріпленням, генетичні алгоритми, автоматизація керування в технічних системах.

Вступ

В сучасній інженерній науці все більш важливими стають методи боротьби з невизначеністю, серед яких значне місце займають штучні нейронні мережі та методи навчання. Навчання з підкріпленням – галузь машинного навчання, що базується на тому, як програмним агентам слід виконувати дії у середовищі з метою максимізації поняття кумулятивної винагороди. Навчання з підкріпленням – це одна з трьох основних парадигм машинного навчання разом з контрольованим навчанням та навчанням без вчителя. Навчання з підкріпленням відрізняється від контрольованого навчання відсутністю необхідності розмічених вхідних та вихідних даних та відсутністю необхідності явного коригування субоптимальних дій. Натомість навчання з підкріпленням фокусується на пошуку балансу між дослідженням «невідомої» території та використанням набутих знань. Середовище для навчання з підкріпленням зазвичай описується у формі марківського процесу рішень, адже багато алгоритмів навчання з підкріпленням використовують техніки динамічного програмування. Головна різниця між класичними методами динамічного програмування та алгоритмами навчання з підкріпленням полягає у тому, що останні не оперують знаннями про чітку математичну модель марківського процесу рішень та орієнтуються на великі марківські процеси, в яких застосування точних методів стає неможливим. Завдяки рівню абстракції навчання з підкріпленням, воно використовується у ряді галузей знань, як теорія ігор, теорія керування, дослідження операцій, теорія інформації, оптимізація на основі симуляції, багатоагентні системи, колективний інтелект, статистика тощо. У літературі, присвяченій дослідженню операцій та теорії керування, навчання з підкріпленням зазвичай називається наближеним динамічним програмуванням або *нейродинамічним програмуванням* [1, 2].

У цій роботі використовується метод нейроеволюції наростаючих топологій (NEAT) [3], що являє собою генетичний алгоритм для генерації еволюціонуючих штучних нейронних мереж, заснований на трьох ключових принципах: спостереження (трекінг) генів з історичними відмітками для схрещування (кросинговеру) між топологіями, використання поняття еволюції видів для підтримання інновацій та інкрементальний розвиток топологій, починаючи з простих початкових структур. Алгоритм NEAT дозволяє використовувати генетичні алгоритми для визначення кращої і мінімально необхідної топології нейронної мережі.

У роботі використано також OpenAI Gym [4] – набір інструментів для розробки та порівняння алгоритмів навчання з підкріпленням. Gym являє собою колекцію тестових задач-середовищ, що можуть бути використані у користувацьких алгоритмах навчання з підкріпленням. Ці середо-

вища мають спільний інтерфейс, що дає змогу користувачам використовувати загальні, спільні підходи до розробки власних алгоритмів.

Суть даної роботи полягає у розробці й дослідженні можливості та ефективності застосування техніки нейроеволюції наростаючих топологій на прикладі однієї з стандартних та популярних задач навчання з підкріпленням з використанням OpenAI Gym, повноцінної імплементації з відкритим програмним кодом алгоритму NEAT під назвою SharpNEAT [5], та проміжного програмного забезпечення для оркестрації зазначених компонентів.

Навчання з підкріпленням

Навчання з підкріпленням – це галузь машинного навчання, де програмним агентам слід виконувати дії у середовищі з метою максимізації поняття кумулятивної винагороди. Навчання з підкріпленням є однією з трьох основних парадигм машинного навчання разом з контрольованим навчанням та навчанням без вчителя.

Навчання з підкріпленням відрізняється від контрольованого навчання у відсутності необхідності розмічених вхідних та вихідних даних та відсутності необхідності явного коригування субоптимальних дій. Натомість навчання з підкріпленням фокусується на пошуку балансу між дослідженням «невідомої» території та використанням набутих знань.

Середовище для навчання з підкріпленням зазвичай описується у формі марківського процесу рішень, адже багато алгоритмів навчання з підкріпленням використовують техніки динамічного програмування.

Типовий алгоритм навчання з підкріпленням моделюється марківським процесом рішень [6], де ми маємо:

- набір станів середовища та агента *S*;
- набір дій агента *A*;
- вірогідність переходу (у момент *t*) з одного стану до іншого при виконанні дії $a \in A$;
- негайну винагороду після переходу з одного стану до іншого при виконанні дії $a \in A$.

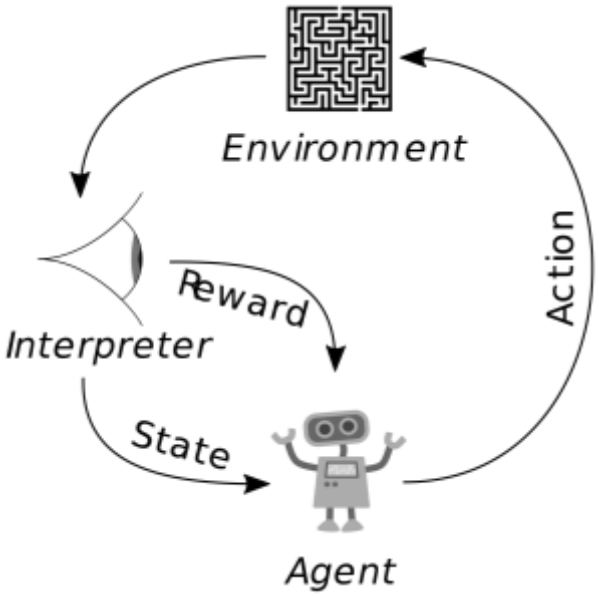

Рис. 1. Схема взаємодії компонентів в алгоритмі навчання з підкріпленням

Агент алгоритму навчання з підкріпленням взаємодіє з середовищем у дискретні проміжки часу. Для кожного проміжку часу агент отримує поточний стан та винагороду. Після цього, агент обирає дію з набору доступних дій (для дискретного простору дій) або значення (для неперервного простору дій), що, в свою чергу, надсилається до середовища. Середовище переходить у новий стан та визначається винагорода для переходу. Задача агента навчання з підкріпленням полягає у вивченні політики, що максимізує очікувану кумулятивну винагороду.

Застосування алгоритму NEAT

Нейроеволюція наростаючих топологій (NEAT) це – генетичний алгоритм для генерації еволюціонуючих штучних нейронних мереж, що встановлює параметри зв'язків між нейронами та структури мереж у спробах знаходження балансу між придатністю еволюціонуючих рішень та їх різноманіттям.

Традиційно, топологія нейромережі обирається людиною, що проводить експеримент, а вдалі значення вагів зв'яз-

ків між нейронами отримуються в процесі навчання. Це призводить до ситуації, де необхідно застосування підходу проб та помилок для знаходження вдалої топології для даної задачі. NEAT – приклад алгоритму, що еволюціонує і топологію, і значення вагів зв'язків штучної нейронної мережі одночасно.

Для кодування нейронної мережі у фенотип для генетичного алгоритму, NEAT використовує пряму схему кодування, в такій схемі явно представлений кожен зв'язок та нейрон.

Підхід NEAT починається з мережі, подібної до перцептрону, з лише вхідним та вихідним прошарком нейронів. З плином дискретних кроків еволюції, складність топології нейронної мережі може зростати за рахунок створення нових нейронів у шляхах зв'язків або за рахунок створення нових зв'язків між попередньо роз’єднаними нейронами.

Проблема договору про конкуренцію з'являється, коли існує декілька способів представлення інформації у фенотипі. Наприклад, якщо геном містить нейрони A, B та C, що представлені саме у цій послідовності, та схрещується з ідентичним за функціональністю, проте різним за представленням геномом [C B A], кросинговер створить виплід, якому бракує інформації, як [A B A] або [C B C]. NEAT вирішує цю проблему за рахунок трекінгу історії генів з використанням глобального числа інновацій, що збільшується з утворенням нових генів. З додаванням нових генів, глобальне число інновацій інкрементується та присвоюється цьому гену. Таким чином, новіші гени мають більші значення маркеру глобального числа інновацій. Для окремо взятого покоління, при виникненні ідентичної мутації у декількох геномах, всім таким геномам присвоюється одне і те ж значення маркеру глобального числа інновацій. Ці маркери числа інновацій дозволяють алгоритму обирати гени для кросинговеру [7].

SharpNEAT представляє собою імплементацію еволюційного алгоритму, а саме алгоритму еволюції нейронних мереж. Еволюційний алгоритм використовує еволюційні механізми мутації, рекомбінації та відбору для пошуку нейронних мереж, функціонал та поведінка котрих задовільнила б попередньо визначену задачу.

Фреймворк SharpNEAT надає набір вбудованих задач-прикладів для демонстрації роботи алгоритму еволюції. Імплементація SharpNEAT є модульною, що дозволяє легко вносити зміни, розширення та повторно використовувати її частини. SharpNEAT створений, зокрема, для задоволення інтересів питань біологічної еволюції та границь нейроеволюції у зрізі рівня складності проблем, рішення яких можуть запропонувати алгоритми нейроеволюції.

Існує дві основні концепції навчання з підкріпленням: середовище (навколишній світ) та агент (алгоритм). Агент надсилає команди до середовища, а середовище відповідає спостереженнями та винагородами. Головний інтерфейс gym стандартизований та універсальний, інтерфейси для агентів відсутні.

Основні методи середовища OpenAI Gym:

- *reset(self):*

скидання стану середовища. Повертає спостереження.

- *step(self, action):*

симулювати середовище на один часовий крок вперед. Повертає спостереження, винагороду та флаг закінчення епізоду.

- *render(self, mode='human'):*

повертає один кадр середовища.

OpenAI Gym не оперує будь-якими припущеннями щодо структури агенту.

У даній роботі розглядається середовище BipedalWalker-v3 [8], просте середовище з роботом, що крокує (див. рис. 2) Робот має чотири шарніри для його кінцівок.

Середовище існує у двох реалізаціях:

- “Normal”, з легкими нерівностями поверхні;

- “Hardcore”, зі сходами, пеньками та ямами.

Винагорода видається за рух вперед. Найвіддаленіші точки середовища відповідають 300 і більше балам винагороди. При падінні робот отримує штраф у 100 балів. Застосування скрутного моменту до шарнірів також має відносно невелику вартість у балах винагороди, а отже ефективніші агенти отримують більші винагороди.

Спостереження за середовищем складається з кутової швидкості корпусу, кутових швидкостей, горизонтальних швидкостей, вертикальних швидкостей, положень шарнірів, контакту ніг з землею, та десяти значень вимірювань лідару. Абсолютні координати у спостереженні за середовища відсутні.

Для успішного вирішення середовища необхідно набрати 300 балів винагороди не довше, ніж за 1600 часових ітерацій [9].

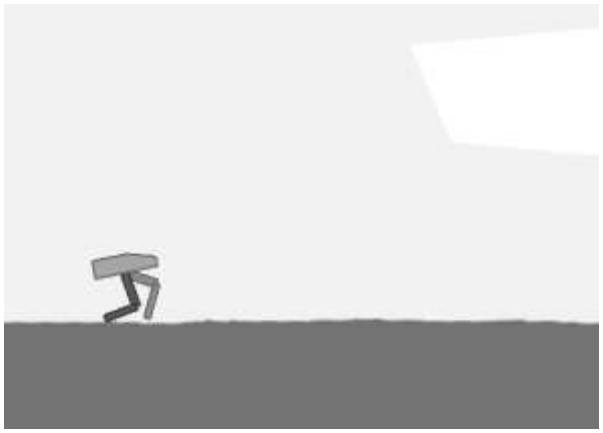

Рис. 2. Кадр з візуалізації середовища BipedalWalker-v3

Для поєднання середовища OpenAI Gym з реалізацією алгоритму Neat SharpNEAT, були застосовані засоби міжпроцесової взаємодії типу іменований канал.

Іменований канал – це один з механізмів обміну даними між процесами Unix та подібних операційних систем. На відміну від неіменованих каналів, доступ до яких мають лише процеси, породжені від спільного батьківського, доступ до іменованих каналів мають всі процеси, що знають ім'я цього каналу та мають відповідні права читання та/або запис в канал. Як і неіменований канал, іменований канал забезпечує обмін даними через оперативну пам'ять. В Unix-системах ім'я іменованого каналу зберігається у файловій системі як ім'я файлу каналу і доступне після перезавантаження системи. У MS Windows файл каналу створюється у пам'яті і знищується між завантаженнями [10].

Як реалізація іменованих каналів на стороні SharpNEAT було обрано сімейство класів .NET System.IO.Pipes: NamedPipeClientStream та NamedPipeServerStream [11].

Як реалізація іменованих каналів на стороні Python-обгортки середовища OpenAI Gym було обрано сімейство pywin32: win32file та win32pipe [12].

Експеримент SharpNEAT був сконфігурований наступним чином:

функція активації нейронів: Leaky rectified linear unit, Leaky ReLU, нещільний лінійний випрямляч, монотонна [13].

Випрямляч у контексті штучних нейромереж є передавальною функцією. Вона є аналогом напівперіодичного випрямляча у схемотехніці. Ця передавальна функція запроваджена для динамічних мереж Ганлозером (англ. Hahnloser) та іншими у 2000 році з біологічним підґрунтям та математичним обґрунтуванням. В 2011 році вперше продемонстровано, як забезпечити краще навчання глибинних мереж, на відміну від передавальних функцій, які широко використовувались до цього, а саме, логістичною функцією (яка була запозичена з теорії ймовірностей), і виявились більш практичними ніж гіперболічний тангенс. ReLU є, станом на 2018 рік, найбільш популярною передавальною функцією для глибинних нейронних мереж [14].

Нещільна ReLU (рис. 3) використовує невеликий додатній градієнт, коли передавач не активний, з рівнянням

$$f(x) = \begin{cases} 0,001x & x < 0 \\ x & x \geq 0 \end{cases}$$

і областю значень: $(-\infty, \infty)$.

До переваг ReLU відносяться.

1. *Біологічна правдоподібність* – одностороння, на відміну від центрально симетричного гіперболічного тангенса.

2. *Розріджена активація* – наприклад, у випадково ініціалізованій мережі, тільки близько 50 % прихованих елементів активуються (мають не нульове значення).

3. *Краще градієнтне поширення*: рідше виникає проблема зникнення градієнта у порівнянні з сигмоїдальною передавальною функцією, яка може виникнути в обох напрямках.

4. *Швидкість обчислення*: тільки порівняння, додавання та множення.

5. *Інваріантність відносно масштабування*: $\max(0, ax) = a\max(0, x)$ для $a \geq 0$.

ReLU використано для відокремлення специфічного збудження та неспецифічної заборони (інгібування) у піраміді з нейронною абстракцією (Neural Abstraction Pyramid), яка була навчена з учителем, щоб вирішувати декілька завдань комп'ютерного зору. У 2011 році, ReLU використовували як елемент нелінійності з метою показати, можливість глибокого навчання нейронної мережі без попереднього навчання без учителя. ReLU, на відміну від сигмоїда та подібних передавальних функцій, дозволяє швидше та ефективніше навчання глибоких нейронних мереж на великих та складних наборах даних.

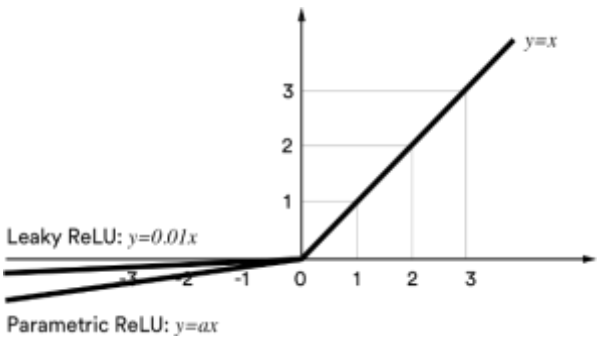

Рис. 3. Графік функції Leaky ReLU

Кількість видів: 10.

Відсоток елітарності: 20 %. Геноми різних видів відсортовані за успішністю, кращі з них, елітарні, зберігаються, інші ж вибраковуються щоб надати дорогу виплоду.

Розмір популяції: 2000. Кількість геномів у популяції.

Початкова частка зв'язків: 5 %.

Відсоток відбору: 20 %. Геноми різних видів сортуються за успішністю та обираються батьківські геноми для створення виплоду. Відбір відбувається до застосування правил елітарності.

Відсоток безстатевого виплоду: 50 %. Відсоток виплоду, шо створиться безстатевим шляхом, за рахунок мутацій.

Відсоток статевого виплоду: 49 %. Відсоток виплоду, що утворюється статевим шляхом.

Відсоток міжвидового розмноження: 1 %. Відсоток статевого розмноження, що задіє геноми різних видів.

Ймовірність мутації ваги зв'язків: 94 %.

Ймовірність мутації створення нового нейрону: 1 %.

Ймовірність мутації створення зв'язку: 2,5 %.

Ймовірність мутації видалення зв'язку: 2,5 %.

Результати експериментів

В результаті застосування алгоритму NEAT/SharpNAET з попередньо визначеною конфігурацією, середовищем та проміжними прошарками поєднуючого програмного забезпечення, поставлену задачу вдалося вирішити (рис. 6) за 10972000 епізодів симуляції. Під успішним вирішенням розуміється набір необхідних 300 балів винагороди найкращою еволюційною нейромережею у кожній із 100 послідовних тестових симуляцій.

На рис. 4 та рис. 5 зображені візуалізації топологій нейромереж геномів-чемпіонів у різні випадкові етапи еволюції.

На рис. 7–10 зображені графіки основних показників історичних даних експерименту: складності та придатності створених геномів.

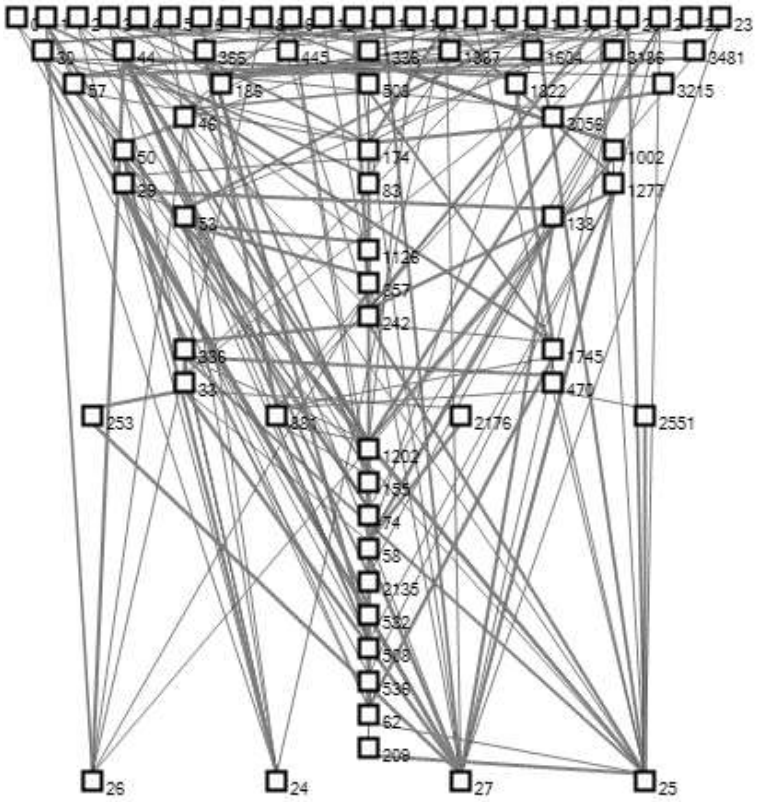

Рис. 4. Візуалізація випадкової топології нейромережі проміжних поколінь

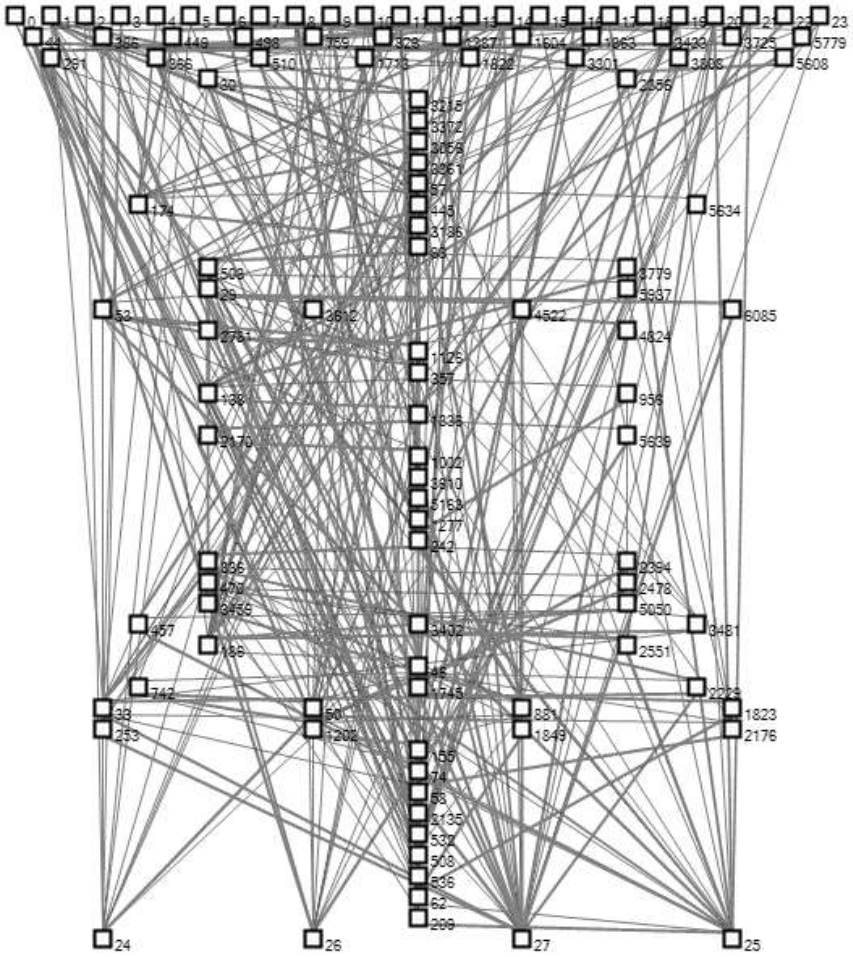

Рис. 5. Візуалізація однієї з найкращих топологій нейромережі

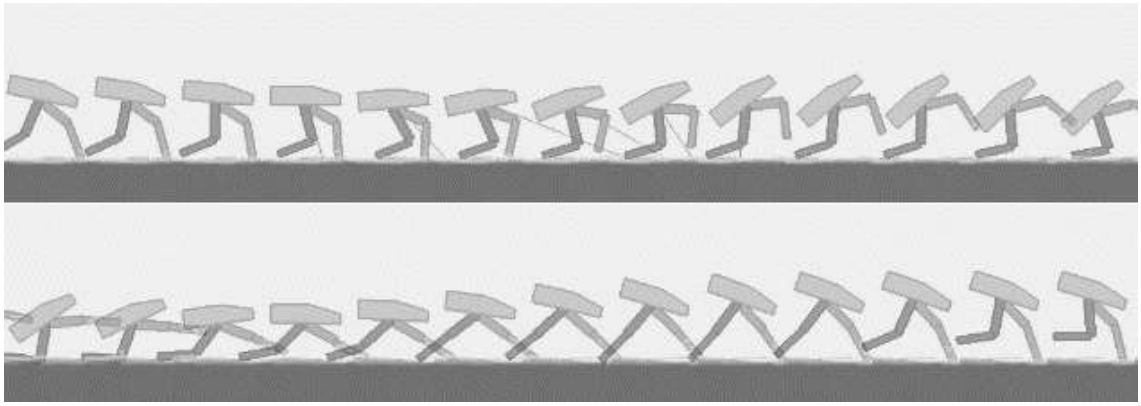

Рис. 6. Комбінована візуалізація кадрів симуляції середовища з використанням агента NEAT

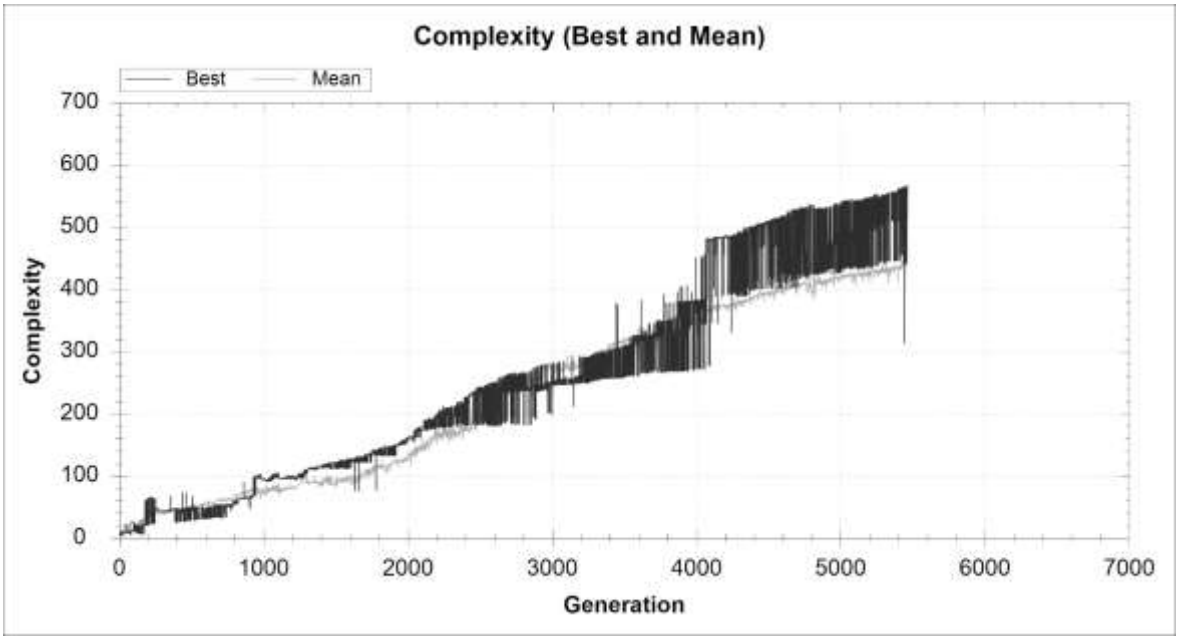

Рис. 7. Графік залежності складності кращої та середньої топології від покоління

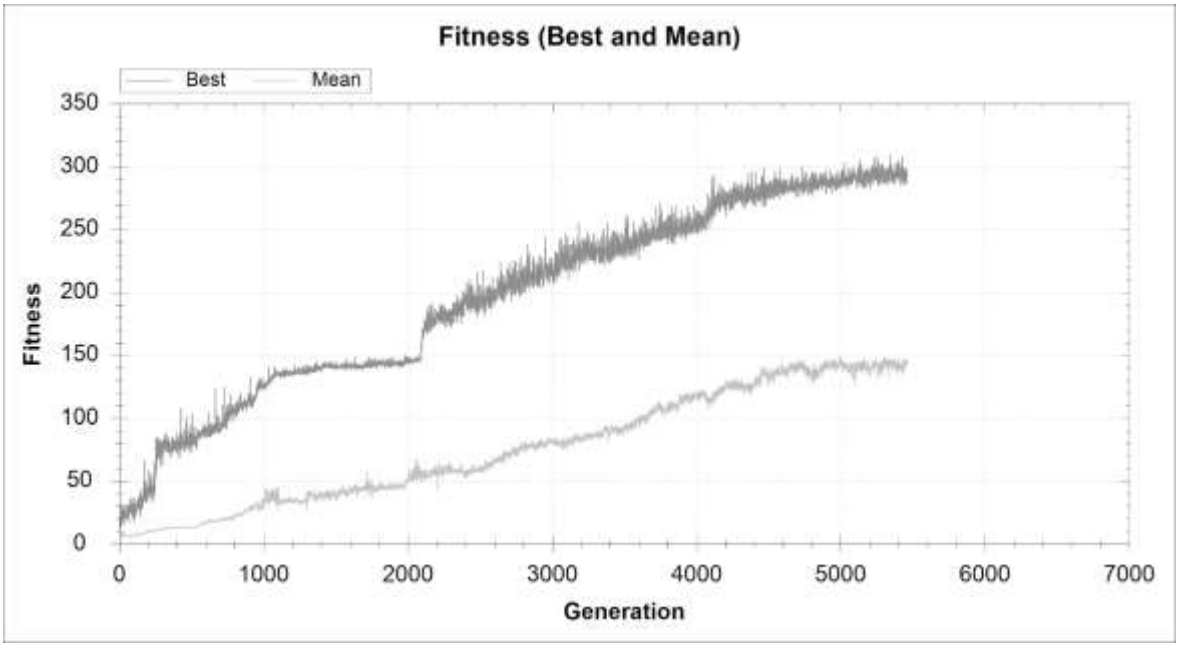

Рис. 8. Графік залежності найкращої та середньої винагороди агентів від покоління

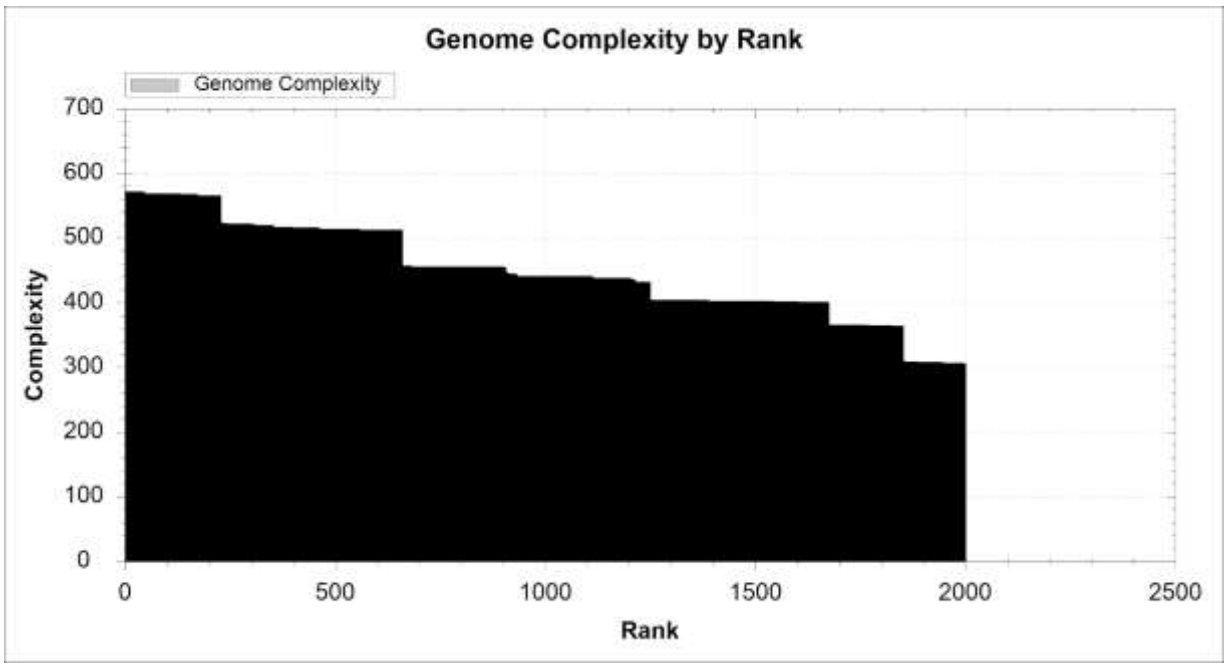

Рис. 9. Графік розподілу складності геному за його рангом

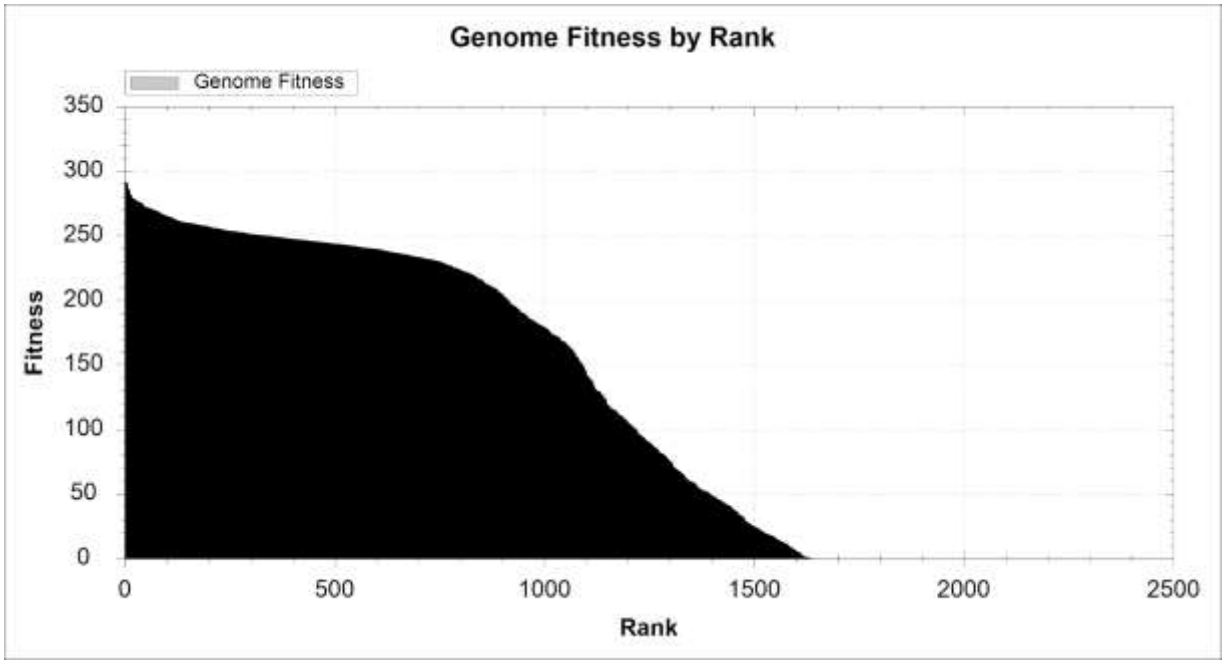

Рис. 10. Графік розподілу придатності геному за його рангом

Висновки

В роботі запропоновано нове застосування техніки нейроеволюції наростаючих топологій для розв'язування задач автоматизації керування на одному з прикладів моделювання задач керування технічними системами, де використовується навчання з підкріпленням. Використовується набір інструментів для розробки та порівняння алгоритмів навчання з підкріпленням OpenAI Gym, повноцінна реалізація з відкритим програмним кодом генетичного алгоритму нейроеволюції NEAT під назвою SharpNEAT та проміжне програмне забезпечення для оркестрації зазначених компонентів. Алгоритм нейроеволюції наростаючих топологій демонструє знаходження ефективних нейронних мереж на прикладі вирішення простих стандартних галузевих задач з

системами з неперервним керуванням OpenAI Gym. Наступними кроками може бути дослідження можливості та ефективності застосування розглянутої техніки та супровідних компонентів середовища на прикладі масштабних задач, задач інших напрямків, дослідження можливості та ефективності застосування паралельних, розподілених та апаратно-прискорених обчислень у NEAT.

Література

1. Хайкин С. Нейронные сети: полный курс, 2-е изд., Испр.: Пер. с англ. М.: ООО «И.Д. Вильямс », 2006. 1104 с.
2. Wilson, Halsey (2018). Artificial intelligence. Grey House Publishing. 184 p.
3. Kenneth O. Stanley. Ph.D. Dissertation: EFFICIENT EVOLUTION OF NEURAL NETWORKS THROUGH COMPLEXIFICATION [Електронний ресурс] / Kenneth O. Stanley // Department of Computer Sciences, The University of Texas at Austin. 2004. Режим доступу до ресурсу: http://nn.cs.utexas.edu/keyword?stanley:phd04.
4. https://gym.openai.com/
5. https://sharpneat.sourceforge.io/
6. Russell Stuart J.; Norvig Peter (2010). Artificial intelligence: a modern approach (Third ed.). Upper Saddle River, New Jersey. P. 830, 831.
7. Kenneth O. Stanley and Risto Miikkulainen (2002). "Evolving Neural Networks Through Augmenting Topologies". Evolutionary Computation 10 (2). P. 99–127.
8. https://gym.openai.com/envs/BipedalWalker-v2/
9. https://github.com/openai/gym/blob/master/gym/envs/box2d/bipedal_walker.py
10. https://tldp.org/LDP/lpg/node15.html
11. https://docs.microsoft.com/en-us/windows/win32/ipc/named-pipes?redirectedfrom=MSDN
12. https://github.com/mhammond/pywin32
13. https://medium.com/@himanshuxd/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e
14. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

References

1. Haykin, Simon S. (1999). Neural networks: a comprehensive foundation. Prentice Hall.
2. Wilson, Halsey (2018). Artificial intelligence. Grey House Publishing. 184 pages.
3. Kenneth O. Stanley. Ph.D. Dissertation: EFFICIENT EVOLUTION OF NEURAL NETWORKS THROUGH COMPLEXIFICATION / Kenneth O. Stanley // Department of Computer Sciences, The University of Texas at Austin. – 2004. – http://nn.cs.utexas.edu/keyword?stanley:phd04.
4. https://gym.openai.com/
5. https://sharpneat.sourceforge.io/
6. Russell, Stuart J.; Norvig, Peter (2010). Artificial intelligence: a modern approach (Third ed.). Upper Saddle River, New Jersey. P. 830, 831.
7. Kenneth O. Stanley and Risto Miikkulainen (2002). "Evolving Neural Networks Through Augmenting Topologies". Evolutionary Computation 10 (2): 99-127
8. https://gym.openai.com/envs/BipedalWalker-v2/
9. https://github.com/openai/gym/blob/master/gym/envs/box2d/bipedal_walker.py
10. https://tldp.org/LDP/lpg/node15.html
11. https://docs.microsoft.com/en-us/windows/win32/ipc/named-pipes?redirectedfrom=MSDN
12. https://github.com/mhammond/pywin32
13. https://medium.com/@himanshuxd/activation-functions-sigmoid-relu-leaky-relu-and-softmax-basics-for-neural-networks-and-deep-8d9c70eed91e
14. https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6

Одержано 21.02.2021

***Про авторів*:**

Дорошенко Анатолій Юхимович,
доктор фізико-математичних наук,
професор, завідувач відділу теорії
комп'ютерних обчислень,
професор кафедри автоматики та
управління в технічних системах
НТУУ "КПІ імені Ігоря Сікорського".

Кількість наукових публікацій в
українських виданнях – понад 190.
Кількість наукових публікацій в
зарубіжних виданнях – понад 80.
Індекс Гірша – 6.
http://orcid.org/0000-0002-8435-1451,

Ашур Ілля Зін-Еддінович,
аспірант 3 курсу в
НТУУ “КПІ імені Ігоря Сікорського”.

Місце роботи авторів:

Національний технічний університет
України "КПІ імені Ігоря Сікорського",
проспект Перемоги 37.

Інститут програмних систем
НАН України,
03187, м. Київ-187,
проспект Академіка Глушкова, 40.

Тел.: (044) 526 3559.

E-mail: doroshenkoanatoliy2@gmail.com,
ilyaachour@gmail.com