

НАЦІОНАЛЬНА АКАДЕМІЯ НАУК УКРАЇНИ
ІНСТИТУТ ПРОГРАМНИХ СИСТЕМ

Кваліфікаційна наукова
праця на правах рукопису

ОМЕЛЬЯНЕНКО ЯРОСЛАВ ВІКТОРОВИЧ

УДК 004.8:004.032.26:004.415

ДИСЕРТАЦІЯ

**ОПТИМІЗАЦІЯ ТОПОЛОГІЇ НЕЙРОННОЇ МЕРЕЖІ НА ОСНОВІ
НЕЙРОЕВОЛЮЦІЙНИХ АЛГОРИТМІВ З ВИКОРИСТАННЯМ
РОЗГАЛУЖЕНИХ ОБЧИСЛЕНЬ**

122 Комп'ютерні науки

12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей,
результатів і текстів інших авторів мають посилання на відповідне джерело
_____ Ярослав ОМЕЛЬЯНЕНКО

Науковий керівник: Сініцин Ігор Петрович, д.т.н., професор, член-кор. НАНУ

Київ – 2026

АНОТАЦІЯ

Омельяненко Я.В. Оптимізація топології нейронної мережі на основі нейроеволюційних алгоритмів з використанням розгалужених обчислень. – Кваліфікаційна наукова праця на правах рукопису.

Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 122 – «Комп'ютерні науки». Робота виконана в Інституті Програмних Систем Національної академії наук України.

Дисертаційну роботу присвячено розв'язанню актуальної науково-практичної задачі підвищення продуктивності побудови топології штучних нейронних мереж нейроеволюційним алгоритмом NEAT шляхом оптимізації етапу оцінки придатності організмів із використанням паралельних обчислень.

Алгоритми нейроеволюції, як і інші генетичні алгоритми, базуються на концепції еволюційного розвитку популяції організмів, що кодують топологію та/або вагові коефіцієнти нейронної мережі. При цьому для організації еволюційного процесу кожен організм у популяції має бути оцінений обраною функцією придатності у кожній епосі еволюції. Зазвичай функція придатності базується на параметрах, отриманих шляхом симуляції практичної задачі, яка має бути вирішена кожним організмом у популяції. Класичні реалізації програмних засобів для керування еволюційним процесом в алгоритмах нейроеволюції реалізують зазначену симуляцію послідовно, для кожного організму у популяції, перед генерацією наступного покоління організмів. Подібний підхід має значний вплив на продуктивність алгоритму нейроеволюції. Це пов'язане з тим, що симуляція практичної задачі переважно потребує значних обчислювальних ресурсів та багато часу на виконання. Таким чином, якщо помножити час на пошук рішення окремим організмом в рамках симуляції на загальну кількість організмів у популяції, маємо значну втрату продуктивності алгоритму. Наслідком цього є обмеження сфери застосування алгоритму нейроеволюції для вирішення практичних задач, що потребують складної симуляції. Це є прикрим з огляду на те, що топологія штучної нейронної мережі,

отримана методом нейроеволюції, зазвичай є оптимальною та енергоефективною, що дозволяє використовувати її у ресурсно-обмежених пристроях, таких як автономні роботизовані платформи, безпілотні літальні апарати і т. ін.

Описаної проблеми з продуктивністю можна уникнути, якщо застосувати сучасні програмні засоби для запуску симуляцій практичних задач паралельно у розгалуженому середовищі. У даній дисертаційній роботі пропонується реалізація програмних засобів для використання розгалужених паралельних обчислень на етапі симуляції практичної задачі, яка використовується для оцінювання придатності організмів у популяції при використанні алгоритму нейроеволюції.

Об'єктом дослідження є процес побудови оптимальної топології ШНМ із застосуванням алгоритму NEAT.

Предметом дослідження є методи, моделі та програмні засоби паралельної реалізації етапу оцінки придатності організмів у NEAT, а також методи кількісної оцінки ефективності такого паралельного виконання.

Метою роботи є покращення ефективності алгоритму NEAT для підвищення швидкості навчання ШНМ в умовах ресурсних обмежень. Для досягнення мети поставлено такі задачі:

1. Провести формальний аналіз обчислювальної складності етапу оцінки придатності у NEAT.
2. Визначити необхідні властивості моделі розпаралелювання, які забезпечують коректність еволюційного процесу.
3. Обґрунтувати вибір архітектури паралельних обчислень, придатної для динамічного розподілу навантаження.
4. Розробити реалізацію моделі Master–Worker з використанням технологій мови програмування GO (goroutines + channels) як базового паралельного механізму.
5. Розробити комплексну метрику оцінки ефективності реалізації NEAT у паралельному режимі.

6. Розробити СПЗ для візуалізації отриманих результатів.

7. Розробити метод оптимізації алгоритму NEAT для підвищення швидкості побудови топології ШНМ та покращення ефективності знайдених рішень в умовах обмежених обчислювальних ресурсів

8. Експериментально підтвердити ефективність запропонованого методу на тестових задачах еволюційного конструювання ШНМ.

Дисертаційна робота складається з трьох розділів.

У **першому розділі** проведено аналіз сучасного стану досліджень у галузі нейроеволюційних алгоритмів, зокрема NEAT та його розширень. Розглянуто біологічні та обчислювальні передумови еволюційного формування топології нейронних мереж, проаналізовано існуючі підходи до масштабування та паралелізації нейроеволюції. Виявлено основні обмеження класичних реалізацій NEAT, пов'язані з високою обчислювальною вартістю етапу оцінки придатності.

У **другому розділі** виконано формальний аналіз обчислювальної складності алгоритму NEAT та обґрунтовано доцільність паралельної оптимізації саме етапу оцінки придатності. Запропоновано архітектуру паралельних обчислень на основі моделі Master–Worker та реалізовано її з використанням вбудованих конкурентних механізмів мови програмування Go (goroutines і channels). Розроблено модель оцінки часових витрат та комплексну інтегральну метрику ефективності, що поєднує часові, структурні та еволюційні характеристики процесу нейроеволюції. Розроблено метод оптимізації алгоритму NEAT для підвищення швидкості побудови топології ШНМ та покращення ефективності знайдених рішень в умовах обмежених обчислювальних ресурсів.

У **третьому розділі** наведено опис експериментальних досліджень та проведено ґрунтовний аналіз отриманих результатів. Експериментально підтверджено ефективність запропонованого методу оптимізації алгоритму NEAT за рахунок паралельної оцінки придатності на тестових задачах еволюційного конструювання штучних нейронних мереж, зокрема на задачі балансування подвійного оберненого маятника. Проаналізовано залежності між

розміром популяції, кількістю воркерів, часом виконання, прискоренням та ефективністю використання ресурсів, а також досліджено Парето-оптимальні області параметрів. Крім того, надано експериментальне підтвердження коректності запропонованих моделей оцінки часових витрат та оцінки ефективності знайдених рішень. Надано детальний опис знаходження ключових параметрів цих моделей емпіричним шляхом.

У дисертаційній роботі отримано такі **нові наукові результати**:

1. **Вперше** запропоновано модель розподілених обчислень Master–Worker для алгоритму NEAT, що базується виключно на вбудованих механізмах конкурентного виконання мови програмування GO (goroutines та channels) та забезпечує ефективне масштабування на багатоядерних системах з мінімальними комунікаційними накладними витратами.

2. **Удосконалено та вперше розроблено єдиний формалізований аналітичний підхід до оцінювання та оптимізації продуктивності алгоритму NEAT, що ґрунтується на поєднанні двох взаємодоповнювальних моделей.** У межах цього підходу, по-перше, *удосконалено модель оцінки ефективності побудови топології штучних нейронних мереж* алгоритмом NEAT на основі декомпозиції часових витрат еволюційного циклу, в якій вперше формалізовано аналітичний зв'язок між структурною складністю нейромереж (кількістю вузлів і зв'язків), розміром популяції, кількістю паралельних виконавців (воркерів) та характеристиками продуктивності апаратної платформи, що забезпечує можливість аналітичного прогнозування часу виконання та визначення оптимальної кількості воркерів для конкретної експериментальної задачі. По-друге, *вперше запропоновано комплексну інтегральну метрику оцінки продуктивності алгоритму NEAT*, яка формалізовано поєднує часові характеристики обчислень, швидкість збіжності еволюційного процесу та якість фінального розв'язку, що дозволяє здійснювати об'єктивне порівняння різних реалізацій, архітектур паралельного виконання та параметричних конфігурацій нейроеволюційних алгоритмів, а також слугує критерієм вибору найбільш

ефективних обчислювальних режимів в умовах обмежених обчислювальних ресурсів.

3. **Вперше** запропоновано метод оптимізації алгоритму NEAT для підвищення швидкості побудови топології штучних нейронних мереж, що базується на інтеграції паралельної моделі Master–Worker для паралельної оцінки придатності, моделі часових витрат та інтегральної метрики оцінки ефективності отриманих рішень та продуктивності алгоритму NEAT.

Практичне значення даної роботи:

- Практичне значення дисертаційної роботи полягає у розробленні та експериментальному обґрунтуванні методу прискорення нейроеволюційного алгоритму NEAT за рахунок паралельної оцінки придатності організмів.

- Запропонована модель паралельних обчислень Master–Worker, реалізована з використанням вбудованих механізмів конкурентного виконання мови програмування GO (goroutines, channels), забезпечує ефективне масштабування нейроеволюційного процесу на багатоядерних обчислювальних системах та може бути безпосередньо використана у прикладних програмних реалізаціях.

- Розроблена модель оцінки часових витрат дозволяє аналітично визначати оптимальну кількість воркерів для заданих параметрів популяції та експериментальної задачі, що знижує потребу у трудомісткому емпіричному підборі параметрів і підвищує ефективність використання обчислювальних ресурсів.

- Запропонована інтегральна метрика ефективності нейроеволюційного процесу забезпечує комплексну оцінку продуктивності алгоритму NEAT з урахуванням часу виконання, швидкості збіжності та якості отриманих рішень і може застосовуватись для порівняльного аналізу різних реалізацій і конфігурацій нейроеволюційних алгоритмів.

Ключові слова: нейронні мережі, нейроеволюційні алгоритми, NEAT, генетичні алгоритми, розподілені обчислення, моделювання, симуляція, формальна модель, оптимальна стратегія

ABSTRACT

Omelianenko Ia.V. Optimization of Neural Network Topology Based on Neuroevolutionary Algorithms Using Distributed Parallel Computations. – Qualification scientific work, manuscript rights.

The dissertation is submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy (PhD) in the specialty 122 – “Computer Sciences”. The research was carried out at the Institute of Software Systems of the National Academy of Sciences of Ukraine.

The dissertation is devoted to solving a relevant scientific and applied problem of improving the performance (speed) of artificial neural network (ANN) topology construction using the NEAT neuroevolutionary algorithm by optimizing the fitness evaluation stage through the application of parallel computing techniques.

Neuroevolutionary algorithms, like other genetic algorithms, are based on the concept of evolutionary development of a population of organisms encoding the topology and/or weight coefficients of a neural network. To maintain the evolutionary process, each organism in the population must be evaluated by a selected fitness function at every evolutionary epoch. Typically, the fitness function relies on parameters obtained by simulating a practical task that each organism in the population must solve. Classical software models for managing the evolutionary process in neuroevolutionary algorithms perform such simulations sequentially for each organism before generating the next population. This approach has a significant negative impact on algorithm performance, since practical task simulation usually requires substantial computational resources and execution time. As a result, the total computation time grows proportionally to the population size, leading to a considerable loss of efficiency and limiting the applicability of neuroevolutionary algorithms to real-world problems involving complex simulations of physical environments. This limitation is particularly critical given that neural network topologies obtained via neuroevolution are typically optimal and energy-efficient, making them suitable for deployment on resource-constrained devices such as autonomous robotic platforms and unmanned aerial vehicles.

The described performance issue can be mitigated by employing modern software tools that enable parallel execution of task simulations in a distributed computational environment. This dissertation proposes a software-based approach for utilizing distributed parallel computations at the stage of practical task simulation used for fitness evaluation in neuroevolutionary algorithms.

The **object of research** is the process of constructing an optimal ANN topology using the NEAT algorithm.

The **subject of research** includes methods, models, and software tools for parallel implementation of the fitness evaluation stage in NEAT, as well as methods for quantitative assessment of the efficiency of such approach.

The **aim of the research** is to improve the efficiency of the NEAT algorithm in order to increase the training speed of artificial neural networks under resource-constrained conditions. To achieve this goal, the following tasks were formulated:

- to conduct a formal analysis of the computational complexity of the fitness evaluation stage in NEAT.
- to determine the required properties of a parallelization model that ensure the correctness of the evolutionary process.
- to justify the choice of a parallel computing architecture suitable for dynamic load balancing.
- to develop an implementation of the Master–Worker model using inherent technologies of the GO programming language (goroutines and channels) as the core parallel mechanism.
- to develop a comprehensive metric for evaluating the efficiency of parallel NEAT implementations.
- to develop software tools for visualization of the obtained results.
- to develop a method for optimizing the NEAT algorithm to increase ANN topology construction speed and improve solution efficiency under limited computational resources.
- to experimentally validate the effectiveness of the proposed method on benchmark neuroevolutionary tasks.

The results of research are presented in **three chapters** of the thesis.

The first chapter analyzes the current state of research in the field of neuroevolutionary algorithms, in particular NEAT and its extensions. Biological and computational prerequisites for the evolutionary formation of neural network topologies are examined, and existing approaches to scalability and parallelization of neuroevolution are reviewed. The main limitations of classical NEAT implementations associated with the high computational cost of fitness evaluation are identified.

The second chapter presents a formal analysis of the computational complexity of the NEAT algorithm and substantiates the feasibility of parallel optimization of the fitness evaluation stage. A parallel computing architecture based on the Master–Worker model is proposed and implemented using the built-in concurrency mechanisms of the Go programming language (goroutines and channels). A model for estimating time costs and a comprehensive integral efficiency metric combining temporal, structural, and evolutionary characteristics of the neuroevolutionary process are developed. A method for optimizing the NEAT algorithm to improve ANN topology construction speed and solution efficiency under limited computational resources is proposed.

The third chapter describes the experimental studies and provides a detailed analysis of the obtained results. The effectiveness of the proposed NEAT optimization method based on parallel fitness evaluation is experimentally confirmed using benchmark neuroevolutionary tasks, including the double inverted pendulum balancing problem. Dependencies between population size, number of workers, execution time, speedup, and resource utilization efficiency are analyzed, and Pareto-optimal parameter regions are investigated. Additionally, experimental validation of the proposed time cost models and solution efficiency assessment models is presented, along with a detailed description of empirical parameter estimation procedures.

The following **scientific results** were obtained in the dissertation:

1. **For the first time**, a distributed Master–Worker computing model for the NEAT algorithm is proposed, based exclusively on the built-in concurrency mechanisms of the Go programming language, ensuring efficient scalability on multicore systems with minimal communication overhead.

2. An integrated and formalized analytical approach to evaluating and optimizing the performance of the NEAT algorithm **has been improved and developed for the first time**, based on the combination of two complementary models. Within this approach, first, *a time-cost-based model for evaluating the efficiency of constructing artificial neural network topologies using the NEAT algorithm has been improved*. This model relies on the decomposition of the evolutionary cycle’s time expenditures and, for the first time, formally establishes an analytical relationship between the structural complexity of neural networks (number of nodes and connections), population size, number of parallel executors (workers), and the performance characteristics of the hardware platform. This enables analytical prediction of execution time and determination of the optimal number of workers for a specific experimental task. Second, *a comprehensive integral metric for assessing the performance of the NEAT algorithm has been proposed for the first time*. This metric formally combines computational time characteristics, the convergence speed of the evolutionary process, and the quality of the final solution, allowing for objective comparison of different implementations, parallel execution architectures, and parameter configurations of neuroevolutionary algorithms, and serving as a criterion for selecting the most efficient computational regimes under constrained computational resources.

3. A method for optimizing the NEAT algorithm to accelerate ANN topology construction is **proposed for the first time**, based on integration of the parallel Master–Worker model for fitness evaluation, time cost modeling, and the integral efficiency metric for assessing solution quality and algorithm performance.

Practical significance of this work:

- The practical significance of the dissertation lies in the development and experimental validation of a method for accelerating the NEAT neuroevolution algorithm by means of parallel fitness evaluation of organisms.
- The proposed Master–Worker parallel computing model, implemented using the built-in concurrency mechanisms of the GO programming language (goroutines, channels), provides efficient scaling of the neuroevolution process on multi-core computing systems and can be directly applied in practical software implementations.
- The developed time-cost evaluation model enables analytical determination of the optimal number of workers for given population parameters and a specific experimental task, reducing the need for labor-intensive empirical parameter tuning and improving the efficiency of computational resource usage.
- The proposed integral metric for neuroevolution efficiency provides a comprehensive assessment of NEAT performance by accounting for execution time, convergence speed, and the quality of the obtained solutions, and it can be used for comparative analysis of different implementations and configurations of neuroevolutionary algorithms.

Keywords: neural networks, neuroevolutionary algorithms, NEAT, genetic algorithms, distributed computing, modeling, simulation, formal model, optimal strategy

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА / LIST OF PUBLICATIONS OF THE APPLICANT:

Статті у фахових виданнях, що входять до переліку, затвердженого ДАК України (категорія Б)

1. Сініцин І.П., Дорошенко А.Ю., Омеляненко Я.В., Черняєв А.В., Яценко О.А. Порівняння різних версій YOLO в залежності від обсягу набору даних та якості зображення. Вісник КрНУ імені Михайла Остроградського. Кременчук: КрНУ, 2025. Вип. 5, № 154. с. 145-154. DOI: <https://doi.org/10.32782/1995-0519.2025.5.17> (0.5 бали)
2. Сініцин І.П., Дорошенко А.Ю., Омеляненко Я.В., Кирилов І.І., Яценко О.А. Адаптивні програмні рішення для автоматизованого виявлення, класифікації, захоплення та ураження цілей ударними безпілотними літальними апаратами. Міжнародний науково-технічний журнал "Проблеми керування та інформатики", 2025. Том 70, № 5. с. 5–18. DOI: <https://doi.org/10.34229/1028-0979-2025-5-1> (0.5 бали)
3. Сініцин І.П., Дорошенко А.Ю., Омеляненко Я.В., Симоненко О.В., Яценко О.А. Планування автоматизованого експерименту системи виявлення цілей з використанням технологій штучного інтелекту. Таврійський науковий вісник. Серія: Технічні науки, 2025. Том 1, № 5. с. 300-311. DOI: <https://doi.org/10.32782/tnv-tech.2025.5.1.32> (0.5 бали)
4. Омеляненко Я. В., Сініцин І. П. Штучний ройовий інтелект. Міжнародний науково-технічний журнал "Проблеми керування та інформатики", 2024. Том 69, № 3. с. 91-103. DOI: <https://doi.org/10.34229/1028-0979-2024-3-7> (1 бал)
5. Омеляненко Я. В., Дорошенко А. Ю., Родін Є.С. Застосування стратегії коеволюції для вирішення задачі автономного проходження лабіринту. Проблеми програмування, 2024. №2-3, с. 263–270. DOI: <https://doi.org/10.15407/pp2024.02-03.263> (0.5 бали)

6. Омеляненко, Я.В. Моделювання автономного проходження лабіринту з використанням алгоритму NEAT. Проблеми програмування, 2023. №4, с. 76–89. DOI: <https://doi.org/10.15407/pp2023.04.076> (1 бал)

Статті у закордонних виданнях

1. Omelianenko, I. Creation of Autonomous Artificial Intelligent Agents using Novelty Search method of fitness function optimization. Hal.science, 2018. P. 1–7. URL: <https://hal.science/hal-01868756> (retrieval date: 01.02.2026).
2. Omelianenko, I. Applying Advanced Machine Learning Models to Classify Electro-Physiological Activity of Human Brain for Use in Biometric Identification. arXiv [Cs.LG], ISSN: 23318422, 2017. P. 1–8. DOI: <https://doi.org/10.48550/arxiv.1708.01167>. **(Scopus)**
3. Omelianenko, I. Applying Deep Machine Learning for Psycho-Demographic Profiling of Internet Users Using O.C.E.A.N. Model of Personality. arXiv [Cs.LG], ISSN: 23318422, 2017. P. 1–24. DOI: <https://doi.org/10.48550/arxiv.1703.06914>. **(Scopus)**

Англомовні монографії

1. Omelianenko, I. Autonomous Artificial Intelligent Agents. *Machine Learning and the City*. Hoboken, NJ, USA: John Wiley & Sons, 2022. P. 263–285. ISBN:9781119815075. DOI: <https://doi.org/10.1002/9781119815075.ch21> **(Scopus)**
2. Omelianenko, I. Hands-On Neuroevolution with Python: Build high-performing artificial neural network architectures using neuroevolution-based algorithms. Birmingham, UK: Packt Publishing, 2019. ISBN: 9781838824914, 368 p.

Англомовні матеріали і тези доповідей на конференціях Scopus, DBLP, що мають ISSN

1. Omelianenko, I. Parallel Fitness Scores Evaluation to Improve Training Speed of the NEAT Algorithm Using GO Routines. In: Yang, X.S., Sherratt, R.S., Dey, N., Joshi, A. (eds) Proceedings of Tenth International Congress on Information and Communication Technology. ICICT 2025. Lecture Notes in Networks and Systems. Singapore, Springer, 2025. Vol. 1441, pp. 207-216. ISSN 2367-3370. DOI: https://doi.org/10.1007/978-981-96-6938-7_17 (**Scopus**)
2. Omelianenko, I. Design of Cluster-Computing Architecture to Improve Training Speed of the Neuroevolution Algorithm. In: Yang, X.S., Sherratt, S., Dey, N., Joshi, A. (eds) Proceedings of Ninth International Congress on Information and Communication Technology. ICICT 2024. Lecture Notes in Networks and Systems. Singapore, Springer, ISSN: 2367-3370, 2024. Vol 1012. pp. 397-408. DOI: https://doi.org/10.1007/978-981-97-3556-3_32 (**Scopus**)
3. Omelianenko, I., Doroshenko, A., Rodin, Y. Autonomous Navigation Through the Maze Using Coevolution Strategy. 14th International Scientific and Practical Programming Conference, UkrPROG 2024, Kyiv, Ukraine, May 14-15, 2024. CEUR Workshop Proceedings, ISSN 1613-0073, 2024, Vol-3806, pp. 301-311. URL: https://ceur-ws.org/Vol-3806/S_29_Omelianenko_Doroshenko_Rodin.pdf (retrieval date: 01.02.2026). (**Scopus, DBLP**)

Матеріали і тези доповідей на конференціях

1. Omelianenko, I., Sinitsyn, I. Applying neuroevolutionary algorithms to build models of aerial threats to identify effective countermeasures. UkrProgAsp-2025-5. 5-th Conference of young scientists. (04.11.2025, Kyiv). Proceedings, ISS NASU, pp. 8–10. (**Англomовна**)
2. Омеляненко Я.В. Покращення продуктивності побудови оптимальної топології штучних нейронних мереж з використанням нейроеволюційних алгоритмів за рахунок застосування методу паралельної оцінки придатності. Теоретичні та прикладні аспекти побудови програмних систем: праці 15 міжнародної науково-практичної конференції, Київ, 23-24 грудня 2024р. /за

заг. ред.: М.М.Глибовця, Т.В.Панченка та ін./ Київ: Видавництво НаУКМА, 2024, 126 с., с. 52-53

3. Омеляненко Я.В. Покращення продуктивності алгоритму NEAT за рахунок застосування методу паралельної оцінки придатності. УкрПрогАсп-2024-4. 4-а Конференція молодих вчених з програмування (06.11.2024, Київ). Зб.матер., ІПС НАНУ, 77с. с.31-35.
4. Омеляненко Я.В., Рогозін Д.В. Покращення швидкості навчання алгоритму NEAT за рахунок використання платформи кластерних обчислень Ray framework. УкрПрогАсп-2024-3. 3-я Конференція молодих вчених з програмування (28.08.2024, Київ). Зб.матер., ІПС НАНУ, 70с. с.43-47.
5. Omelianenko, I. Applying Deep Machine Learning for Psycho-Demographic Profiling of Internet Users Using O.C.E.A.N. Model of Personality. In Proceedings of the 2017 Future Technologies Conference (FTC). 29-30 November 2017. Vancouver, BC, Canada. pp. 375-384. URL: https://saiconference.com/Downloads/FTC2017/Proceedings/52_Paper_195-Applying_Deep_Machine_Learning.pdf (retrieval date: 01.02.2026).
(Англomовна)

Опубліковані праці, які додатково відображають наукові результати дисертації

1. Omelianenko, I. The GoLang implementation of NeuroEvolution of Augmenting Topologies (NEAT) algorithm. [Research Software]. v4.2.3, Zenodo, 2 Jan. 2026. DOI: <https://doi.org/10.5281/zenodo.18131733>
2. Omelianenko, I. NEAT efficiency scoring metrics and visualization tools. [Research software]. v0.2.0, Zenodo, 2 Jan. 2026. DOI: <https://doi.org/10.5281/zenodo.18131847>.
3. Omelianenko, I. The GOLang implementation of NeuroEvolution of Augmenting Topologies (NEAT) with Novelty Search optimization. [Research Software]. v4.0.2, Zenodo, 9 Apr. 2024. DOI: <https://doi.org/10.5281/zenodo.10951521>

4. Комп'ютерна програма "Платформа DRMS" / І.П. Сініцин, Я.В. Омеляненко, Є.М. Твердохліб, М.Ю. Степанюк, О.Б. Житкевич, В.М. Вервейко. Свідोцтво про реєстрацію авторського права на твір № 122378 від 26.12.2023. URL: https://iss.nas.gov.ua/storage/editor/files/private/Sinitsin/03-svidotstvo_DRMS.pdf (дата звернення: 01.02.2026).
5. Omelianenko, I. Artificial Swarm Intelligence and Cooperative Robotic Systems. Preprints, 2019. P. 1–8. DOI: <https://doi.org/10.20944/preprints201901.0282.v1>
6. Omelianenko, I. Performance Metric for Comparing the NEAT Libraries. ResearchGate, 2019. P. 1-3. URL: https://www.researchgate.net/publication/335062286_The_efficiency_comparison_of_NEAT_libraries_while_doing_search_for_XOR_problem_solvers (retrieval date: 01.02.2026).
7. Omelianenko, I. The efficiency comparison of NEAT libraries when looking for Double Pole-Balancing controller. ResearchGate, 2019. P. 1-8. DOI: <http://dx.doi.org/10.13140/RG.2.2.23941.77283>
8. Omelianenko, I. The efficiency comparison of NEAT libraries when looking for Single Pole-Balancing controller. ResearchGate, 2019. P. 1-8. DOI: <http://dx.doi.org/10.13140/RG.2.2.13875.44329>
9. Omelianenko, I. The efficiency comparison of NEAT libraries while doing search for XOR problem solvers. ResearchGate, 2019. P. 1-5. DOI: <http://dx.doi.org/10.13140/RG.2.2.20586.32965>
10. Патент на корисну модель. Спосіб обміну даними: пат. 31149 Україна. № u200713820; заявл. 10.12.2007; опубл. 25.03.2008, Бюл. № 6. URL: <https://sis.nipo.gov.ua/en/search/detail/313727/> (дата звернення: 01.02.2026).

Всього за тематикою дисертації:

Основні результати дисертаційної роботи опубліковано у 29 (6+3+2+3+5+10) наукових працях, серед яких: 6 статей у фахових виданнях України (категорії «Б»), 6 публікацій у міжнародних виданнях, індексованих у наукометричних базах Scopus та/або WoS (Springer LNNS, CEUR Workshop

Proceedings, Wiley, arXiv), 1 статтю у закордонному виданні (HAL Science), а також 5 препринтів і технічних звітів.

Опубліковано 1 **англомовну** одноосібну монографію (Packt Publishing, Birmingham, UK, 2019) та 1 розділ у колективній **англомовній** монографії (John Wiley & Sons, Hoboken, NJ, USA, 2022 – **Scopus**).

Результати дослідження апробовано у матеріалах 6 міжнародних науково-технічних конференцій (**3 Scopus, DBLP**), та 2 науково-технічних конференцій в Україні.

Крім того, отримано 1 свідоцтво про реєстрацію авторського права на програмний продукт та 1 патент на корисну модель, а також опубліковано у відкритому доступі 3 програмні бібліотеки (Zenodo, 2024–2026), що є у вільному доступі на GitHub.

Апробація результатів дисертації. Основні результати дисертаційного дослідження доповідалися та обговорювалися на міжнародних та національних науково-практичних конференціях:

1. UkrProgAsp-2025-5. 5-th International Conference of young scientists. 04.11.2025, Kyiv, Ukraine. (**англомовна**)
2. ICICT 2025. Tenth International Congress on Information and Communication Technology. February 18-21, 2025, London, United Kingdom. (**англомовна, Scopus**)
3. TAAPSD 2024. Теоретичні та прикладні аспекти побудови програмних систем. 15-та міжнародна науково-практична конференція. 23-24 Грудня, 2024, Київ, Україна.
4. УкрПрогАсп-2024-4. 4-а Конференція молодих вчених з програмування. 06.11.2024, Київ, Україна.
5. УкрПрогАсп-2024-3. 3-я Конференція молодих вчених з програмування. 25.04.2024, Київ, Україна.

6. UkrPROG-2024. 14-th International Scientific and Practical Programming Conference. May 14-15, 2024, Kyiv, Ukraine. (**англомовна, Scopus, DBLP**)
7. ICICT 2024. Ninth International Congress on Information and Communication Technology. February 19-22, 2024, London, United Kingdom. (**англомовна, Scopus**)
8. FTC 2017. Future Technologies Conference. November 29-30, 2017, Vancouver, BC, Canada. (**англомовна**)

ЗМІСТ

ДИСЕРТАЦІЯ	1
АНОТАЦІЯ	2
ЗМІСТ	19
ВСТУП	22
РОЗДІЛ 1. НЕЙРОЕВОЛЮЦІЙНІ АЛГОРИТМИ, ЇХ ОСОБЛИВОСТІ ТА СФЕРИ ЗАСТОСУВАННЯ	29
1.1. <i>Нейроеволюція – побудова нейронних мереж методами природної еволюції</i>	29
1.2. <i>Нейроеволюція та глибоке машинне навчання</i>	34
1.3. <i>Різноманітність поведінки та пошук новизни</i>	39
1.4. <i>Основні поняття нейроеволюційних алгоритмів</i>	43
1.5. <i>Особливості алгоритму NEAT</i>	50
1.6. <i>NEAT на основі гіперкубу (HyperNEAT)</i>	59
1.7. <i>HyperNEAT із субстратом, що розвивається (ES-HyperNEAT)</i>	65
1.8. <i>Постановка задач дослідження</i>	73
1.9. <i>Висновки до першого розділу</i>	76
РОЗДІЛ 2. РОЗРОБКА МЕТОДУ ОПТИМІЗАЦІЇ АЛГОРИТМУ NEAT ДЛЯ ПІДВИЩЕННЯ ШВИДКОСТІ ПОБУДОВИ ТОПОЛОГІЇ ШНМ	79
2.1. <i>Формальна постановка задачі дослідження</i>	79
2.2. <i>Формальний аналіз обчислювальної складності етапу оцінки придатності у алгоритмі NEAT</i>	81
2.3. <i>Комплексна метрика оцінки ефективності експериментів у нейроеволюційних алгоритмах</i>	87
2.4. <i>Аналіз існуючих підходів до паралельної еволюції нейронних мереж</i>	90

2.5.	<i>Розподілений підхід до оцінки придатності в алгоритмі NEAT та обґрунтування вибору моделі Master-Worker.....</i>	96
2.6.	<i>Обґрунтування використання вбудованих механізмів мови GO у модифікованій моделі Master-Worker</i>	103
2.7.	<i>Опис нової моделі Master-Worker на основі засобів конкурентного програмування мови GO</i>	107
2.8.	<i>Метод оптимізації алгоритму NEAT для підвищення швидкості побудови топології ШНМ та покращення ефективності знайдених рішень в умовах обмежених обчислювальних ресурсів</i>	113
2.9.	<i>Вибір методів оцінки якості та метрик ефективності.....</i>	117
2.10.	<i>Нові наукові результати за результатами дослідження</i>	121
2.11.	<i>Висновки до другого розділу</i>	123
РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ МЕТОДУ		
ОПТИМІЗАЦІЇ АЛГОРИТМУ NEAT ДЛЯ ПІДВИЩЕННЯ ШВИДКОСТІ		
ПОБУДОВИ ТОПОЛОГІЇ ШНМ ТА ПОКРАЩЕННЯ ЕФЕКТИВНОСТІ		
ЗНАЙДЕНИХ РІШЕНЬ		
3.1.	<i>Мета, задачі та загальна методика експериментальних досліджень</i>	126
3.1.1.	<i>Методика проведення досліджень</i>	127
3.1.2.	<i>Висновки до розділу.....</i>	129
3.2.	<i>Експериментальний стенд та програмне забезпечення.....</i>	131
3.3.	<i>Опис експерименту.....</i>	132
3.3.1.	<i>Опис тестового середовища.....</i>	132
3.3.2.	<i>Опис цільової функції (функції придатності)</i>	137
3.3.3.	<i>Висновки до розділу.....</i>	138

3.4. Опис алгоритму паралельної оцінки придатності за модифікованою моделлю Master-Worker	140
3.4.1. Алгоритм – формальний опис.....	141
3.4.2. Алгоритм – структурний опис.....	143
3.4.3. Алгоритм - реалізація мовою програмування GO	145
3.4.4. Висновки до розділу.....	150
3.5. Опис та аналіз результатів експериментальних досліджень.....	152
3.5.1. Параметри моделі часових витрат.....	152
3.5.2. Залежність часу еволюції від кількості воркерів.....	153
3.5.3. Знаходження області раціональної кількості воркерів.....	155
3.5.4. Прогноз оптимальної кількості воркерів для збільшеного розміру популяції.....	165
3.5.5. Аналіз похибок прогнозової моделі часових витрат.....	167
3.5.6. Аналіз прискорення нейроеволюційного процесу при застосуванні запропонованого методу покращення ефективності алгоритму NEAT.....	170
3.6. Аналіз моделі оцінки ефективності знайдених рішень за метрикою EfficiencyScore	173
3.7. Висновки до третього розділу.....	176
ВИСНОВКИ	179
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	182
ДОДАТКИ.....	195
ДОДАТОК А – ЕКСПЕРИМЕНТАЛЬНІ ДАННІ.....	195
ДОДАТОК Б – ДОКУМЕНТИ, ЩО ПІДТВЕРДЖУЮТЬ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОГО ДОСЛІДЖЕННЯ.....	200

ВСТУП

Актуальність теми. На тлі успіху методів глибинного машинного навчання, заснованих на методі зворотного поширення градієнту похибки, у багатьох провідних дослідників виникає сумнів стосовно подальших перспектив цього методу [1]. Цей метод навчання ідеально підходить для розв'язання задач, пов'язаних з пошуком шаблонів у вхідних даних, таких як зображення, аудіосигнали або текстові дані. У той же час він досить часто втрачає свою ефективність у задачах, що потребують розуміння контексту або навчання з одного зразка (one shot learning).

Іншим значним недоліком існуючих методів глибинного навчання є низька енергетична ефективність обчислень. Топологія штучної нейронної мережі (ШНМ) задається наперед, і навчання зводиться до пошуку вагових коефіцієнтів для зв'язків між нейронами. Втім дана топологія в багатьох випадках далека від оптимальної. Це призводить до того, що значна кількість штучних нейронів перебувають в недонасиченому (undersaturated) або перенасиченому (oversaturated) стані. Як результат, вони майже не беруть участі у процесі навчання або формуванні висновків (inference), але на обчислення їхнього стану витрачаються значні обчислювальні потужності. Крім того, на жаль, серед дослідників та практиків зараз є прогресуюча тенденція використовувати готові шаблони архітектури нейронних мереж без аналізу їхньої ефективності для конкретної задачі. Такий підхід призводить до подальшого нарощування обчислювальних потужностей, задіяних під навчання машинних моделей. Це має значний негативний вплив як на навколишнє середовище, так і на здатність інших дослідників з меншими наявними ресурсами відтворити результати досліджень.

Поряд із зазначеними методами глибинного навчання, протягом багатьох років розробляються альтернативні підходи до навчання машинних моделей, які мають гарні перспективи для вирішення вказаних проблем. Йдеться про еволюційні методи навчання, які базуються на принципах теорії еволюції

Дарвіна [2] та використовують абстракції теорії еволюції для створення штучних нейронних мереж.

NEAT [3, 4] – це один із найвпливовіших еволюційних алгоритмів, існуючих на даний час, що довів свою ефективність у багатьох задачах, пов'язаних з навчанням ШНМ для контролерів роботів [5, 6], з розробкою дизайну нових роботизованих контролерів і систем керування [7, 8], розробкою синтетичних біохімічних систем [9], у задачах оброблення результатів у геології [10], а також для оцінки вимірювання топ-кварка з колайдера частинок Теватрон [11], генерування торгових сигналів для фінансових ринків [12], аналізу відкритих фундаментальних питань у галузі вивчення природної еволюції [13]. Крім цього, ведуться дослідницькі проєкти з метою побудови кооперуючих роботизованих систем на базі ройового інтелекту з використанням алгоритму NEAT для коеволюції ройових агентів та вирішення задач із симуляції проходження складних лабіринтів [14, 15, 16, 17, 18].

Але, незважаючи на потужність і енергетичну ефективність топології ШНМ, продукуваних за допомогою алгоритму NEAT, залишається один суттєвий недолік, що гальмує його широке застосування. Цим недоліком є повільна швидкість навчання еволюційних моделей ШНМ на основі як алгоритму NEAT, так і інших нейроеволюційних алгоритмів.

Дана робота покликана дослідити та запровадити механізми розгалужених паралельних обчислень для прискорення навчання машинних моделей з використанням алгоритму NEAT.

Об'єктом дослідження є процес побудови оптимальної топології ШНМ із застосуванням алгоритму NEAT.

Предметом дослідження є методи, моделі та програмні засоби паралельної реалізації етапу оцінки придатності організмів у NEAT, а також методи кількісної оцінки ефективності такого паралельного виконання. Методологія роботи базується на теорії нейронних мереж, еволюційних алгоритмах, теорії програмування та експериментальному моделюванні.

Методи дослідження: теорія програмування, теорія алгоритмів, теорія нейронних мереж, теорія еволюційних алгоритмів, машинне навчання та експериментальному моделюванні.

Мета і задачі дослідження. Метою дисертаційної роботи є покращення ефективності алгоритму NEAT для підвищення швидкості навчання ШНМ в умовах ресурсних обмежень. Для досягнення мети поставлено такі задачі:

1. Провести формальний аналіз обчислювальної складності етапу оцінки придатності у NEAT.
2. Визначити необхідні властивості моделі розпаралелювання, які забезпечують коректність еволюційного процесу.
3. Обґрунтувати вибір архітектури паралельних обчислень, придатної для динамічного розподілу навантаження.
4. Розробити реалізацію моделі Master–Worker з використанням технологій Go (goroutines + channels) як базового паралельного механізму.
5. Розробити комплексну метрику оцінки ефективності реалізації NEAT у паралельному режимі.
6. Розробити СПЗ для візуалізації отриманих результатів.
7. Розробити метод оптимізації алгоритму NEAT для підвищення швидкості побудови топології ШНМ та покращення ефективності знайдених рішень в умовах обмежених обчислювальних ресурсів
8. Експериментально підтвердити ефективність запропонованого методу на тестових задачах еволюційного конструювання ШНМ.

Наукова новизна отриманих результатів.

У дисертаційній роботі отримано такі **нові наукові результати**:

1. **Вперше** запропоновано модель розподілених обчислень Master–Worker для алгоритму NEAT, що базується виключно на вбудованих механізмах конкурентного виконання мови програмування GO (goroutines та channels) та забезпечує ефективне масштабування на багатоядерних системах з мінімальними комунікаційними накладними витратами.

2. Удосконалено та вперше запропоновано єдиний формалізований аналітичний підхід до оцінювання та оптимізації продуктивності алгоритму NEAT, що ґрунтується на поєднанні двох взаємодоповнювальних моделей.

У межах цього підходу, по-перше, *удосконалено модель оцінки ефективності побудови топології штучних нейронних мереж* алгоритмом NEAT на основі декомпозиції часових витрат еволюційного циклу, в якій вперше формалізовано аналітичний зв'язок між структурною складністю нейромереж (кількістю вузлів і зв'язків), розміром популяції, кількістю паралельних виконавців (воркерів) та характеристиками продуктивності апаратної платформи, що забезпечує можливість аналітичного прогнозування часу виконання та визначення оптимальної кількості воркерів для конкретної експериментальної задачі. По-друге, *вперше запропоновано комплексну інтегральну метрику оцінки продуктивності алгоритму NEAT*, яка формалізовано поєднує часові характеристики обчислень, швидкість збіжності еволюційного процесу та якість фінального розв'язку, що дозволяє здійснювати об'єктивне порівняння різних реалізацій, архітектур паралельного виконання та параметричних конфігурацій нейроеволюційних алгоритмів, а також слугує критерієм вибору найбільш ефективних обчислювальних режимів в умовах обмежених обчислювальних ресурсів.

3. Вперше запропоновано метод оптимізації алгоритму NEAT для підвищення швидкості побудови топології штучних нейронних мереж, що базується на інтеграції паралельної моделі Master–Worker для паралельної оцінки придатності, моделі часових витрат та інтегральної метрики оцінки ефективності отриманих рішень та продуктивності алгоритму NEAT.

Практичне значення отриманих результатів.

Практичне значення результатів дисертаційної роботи полягає у можливості їх безпосереднього застосування під час розроблення, оптимізації та експлуатації нейроеволюційних алгоритмів, зокрема алгоритму NEAT, у середовищах з обмеженими або гетерогенними обчислювальними ресурсами.

1. Розроблений метод паралельної оцінки придатності організмів за моделлю Master–Worker може бути безпосередньо використаний для прискорення нейроеволюційних обчислень у багатоядерних та кластерних обчислювальних системах. Запропонована реалізація на основі вбудованих механізмів конкурентного виконання мови програмування GO (goroutines, channels) забезпечує простоту інтеграції, масштабованість та мінімальні комунікаційні накладні витрати, що робить її придатною для використання у прикладних програмних системах та дослідницьких платформах.

2. Запропонована модель оцінки часових витрат нейроеволюційного процесу дозволяє здійснювати аналітичний вибір оптимальної кількості воркерів для конкретної задачі, розміру популяції та характеристик симуляційного середовища. Це забезпечує можливість попереднього налаштування параметрів паралельного виконання без проведення трудомістких емпіричних експериментів, що є особливо важливим для практичних застосувань із жорсткими часовими або енергетичними обмеженнями.

3. Розроблена інтегральна метрика ефективності виконання алгоритму NEAT може бути використана як універсальний інструмент для порівняльного аналізу різних реалізацій нейроеволюційних алгоритмів, різних схем архітектури паралельних обчислень та параметричних конфігурацій. Метрика поєднує часові показники, швидкість збіжності та якість отриманих рішень, що робить її придатною для прийняття інженерних рішень під час оптимізації реальних систем штучного інтелекту.

4. Реалізоване програмне забезпечення та інструменти візуалізації результатів можуть бути використані у навчальному процесі, наукових дослідженнях та прикладних проєктах для аналізу поведінки нейроеволюційних алгоритмів, дослідження ефектів паралелізації та оцінки ефективності різних стратегій оптимізації. Запропоновані засоби дозволяють наочно інтерпретувати результати експериментів та спрощують відтворюваність досліджень.

5. Отримані результати можуть бути застосовані під час розроблення інтелектуальних систем керування, автономних агентів та симуляційних

моделей, де використовується нейроеволюційний підхід до синтезу структури штучних нейронних мереж. Запропоновані методи та моделі забезпечують скорочення часу навчання, підвищення ефективності використання обчислювальних ресурсів та зниження експериментальних витрат у практичних задачах.

Особистий внесок здобувача. Усі основні результати дисертаційного дослідження, що представлені до захисту, одержані автором особисто. Науковому керівнику належить постановка задачі та скеровування виконаних досліджень.

Апробація результатів дисертації. Основні результати дисертаційного дослідження доповідалися та обговорювалися на міжнародних та національних науково-практичних конференціях:

1. UkrProgAsp-2025-5. 5-th International Conference of young scientists. 04.11.2025, Kyiv, Ukraine. **(англомовна)**
2. ICICT 2025. Tenth International Congress on Information and Communication Technology. February 18-21, 2025, London, United Kingdom. **(англомовна, Scopus)**
3. TAAPSD 2024. Теоретичні та прикладні аспекти побудови програмних систем. 15-та міжнародна науково-практична конференція. 23-24 Грудня, 2024, Київ, Україна.
4. УкрПрогАсп-2024-4. 4-а Конференція молодих вчених з програмування. 06.11.2024, Київ, Україна.
5. УкрПрогАсп-2024-3. 3-я Конференція молодих вчених з програмування. 25.04.2024, Київ, Україна.
6. UkrPROG-2024. 14-th International Scientific and Practical Programming Conference. May 14-15, 2024, Kyiv, Ukraine. **(англомовна, Scopus, DBLP)**

7. ICICT 2024. Ninth International Congress on Information and Communication Technology. February 19-22, 2024, London, United Kingdom. (**англомовна, Scopus**)

8. FTC 2017. Future Technologies Conference. November 29-30, 2017, Vancouver, BC, Canada. (**англомовна**)

Публікації. Основні результати дисертаційної роботи опубліковано у 29 (6+3+2+3+5+10) наукових працях, серед яких: 6 статей у фахових виданнях України (категорії «Б»), 6 публікацій у міжнародних виданнях, індексованих у наукометричних базах Scopus та/або WoS (Springer LNNS, CEUR Workshop Proceedings, Wiley, arXiv), 1 статтю у закордонному виданні (HAL Science), а також 5 препринтів і технічних звітів.

Опубліковано 1 одноосібну **англомовну** монографію (Packt Publishing, Birmingham, UK, 2019) та 1 розділ у колективній **англомовній** монографії (John Wiley & Sons, Hoboken, NJ, USA, 2022 – **Scopus**).

Результати дослідження апробовано у матеріалах 6 міжнародних науково-технічних конференцій (**3 Scopus, DBLP**), та 2 науково-технічних конференцій в Україні.

Крім того, отримано 1 свідоцтво про реєстрацію авторського права на програмний продукт та 1 патент на корисну модель, а також опубліковано у відкритому доступі 3 програмні бібліотеки (Zenodo, 2024–2026), що є у вільному доступі на GitHub.

Структура роботи. Дисертаційна робота складається зі вступу, трьох розділів, висновків, 2 додатки та списку використаних джерел, що включає 111 найменувань. Загальний обсяг роботи становить 202 сторінки, основний текст роботи викладено на 160 сторінках. Робота містить 27 рисунків та 13 таблиць.

РОЗДІЛ 1. НЕЙРОЕВОЛЮЦІЙНІ АЛГОРИТМИ, ЇХ ОСОБЛИВОСТІ ТА СФЕРИ ЗАСТОСУВАННЯ

1.1. Нейроеволюція – побудова нейронних мереж методами природної еволюції

Вивчаючи людський мозок, з його 100 трильйонами з'єднань між 100 мільярдами нейронів, дивуєшся, що він є продуктом еволюції – тобто, природним процесом без розумного контролю чи передбачуваності. Незважаючи на те, що штучні нейронні мережі зазнали бурхливого розвитку в останні кілька років, вони все ще лишаються тінню того шедевр, що створила природна еволюція.

Більшість дослідницьких зусиль у теперішній час зосереджені на вивченні методів та алгоритмів так званого глибокого навчання (deep learning) [19, 20] і глибокого навчання з підкріпленням (deep reinforcement learning) [21, 22], де основним засобом навчання ШНМ є використання алгоритму зворотного поширення (backpropagation) [23]. Це дуже ефективний алгоритм для обчислення градієнту функції втрат, який у поєднанні з методом стохастичного градієнтного спуску (stochastic gradient descent) може ефективно змінювати вагові коефіцієнти штучної нейронної мережі для мінімізації похибки (втрат). Використання алгоритму зворотного поширення похибки спричинило справжній прорив у сфері глибокого машинного навчання та дало змогу майже наблизитись, а в деяких галузях і перевищити людські здібності з розпізнавання шаблонів у різних наборах даних (візуальна та аудіоінформація, текстові дані). Завдяки цьому алгоритму стало можливим побудувати модернові великі мовні моделі (large language model) [24], що так зачаровують своєю спроможністю щодо синтезу складних текстів.

Але існує альтернативний підхід, що черпає натхнення з природної еволюції, з того самого процесу, що породив наш мозок. Цей підхід полягає у використанні еволюційних алгоритмів для навчання ШНМ і отримав назву – нейроеволюція. Нейроеволюція прагне алгоритмічно абстрагувати процес

природної еволюції, виділяючи його креативні аспекти для використання в інших сферах, окрім природи. Основна абстракція, запозичена з природної еволюції, полягає в концепції біологічної придатності організму та її ролі у природному доборі. Природний добір визначає, що найбільш біологічно придатні організми, які в середньому дають більше потомства, мають тенденцію витіснити менш придатні. У природній еволюції придатність зазвичай визначається репродуктивною здатністю організму. У той же час, в алгоритмічній абстракції еволюції можна визначити функцію придатності, що орієнтована на успіх у вирішенні конкретного завдання. Наприклад, якщо є завдання — розробити контролер для крокуючого робота, то функцією придатності може бути відстань, яку робот зміг подолати перед падінням. Отже, можна узагальнити, що більші значення функції придатності визначають кращу пристосованість і продуктивність. Таким чином, нейроеволюційний процес діє як оптимізатор функції придатності.

У порівнянні з методами глибокого навчання, які також можна вважати методами оптимізації функції придатності, нейроеволюційні методи мають кілька переваг, котрі зазвичай недоступні градієнтним методам. До таких переваг належать:

- автоматичне визначення окремих блоків ШНМ у процесі навчання (наприклад, функції активації для різних типів нейронів);
- автоматичне визначення архітектури ШНМ у процесі навчання (наприклад, кількість нейронів на шар, скільки є шарів та шаблон з'єднань між ними);
- автоматичне визначення правил самого навчання.

Окрім цього, є ще низка фундаментальних відмінностей нейроеволюції від традиційних методів. Для прикладу, нейроеволюційний алгоритм працює з популяцією рішень та підтримує її протягом еволюційного процесу, що надає свої унікальні переваги, але має і своєрідні недоліки.

Як зазначалося, натхнення для розробки методів нейроеволюції черпалося з природної еволюції мозку. У 1980 р. концепція штучних нейронних мереж була

вже добре означена і дослідники почали задаватись питаннями, чи можливо ці грубі абстракції біологічного мозку розвинути штучно, через застосування еволюційних принципів, подібних до тих, що існують у природі. Еволюційні алгоритми на той час також вже були добре відомі та використовувалися для знаходження рішень у популяції вирішувачів протягом багатьох поколінь, на кшталт того, як це відбувається у природній еволюції [25]. А саме, використовуючи оператори мутації та кросинговеру, популяція вирішувачів поступово еволюціонує, збільшуючи оцінку придатності (fitness score) еволюційних чемпіонів у кожній наступній епосі.

Поєднання цих двох добре відомих на той час концепцій призвело до народження сімейства алгоритмів нейроеволюції. Першими кроками у цьому напрямку було застосування еволюційних алгоритмів лише до вагових коефіцієнтів у невеликих ШНМ з фіксованою топологією. Згодом увагу дослідників привернули більш амбітні можливості, а саме – шляхом еволюційного процесу вирощувати саму топологію ШНМ [26, 27] або, навіть, розвивати динаміку внутрішнього навчання (тобто еволюція правил для оновлення вагових коефіцієнтів як альтернатива навчанню з підкріпленням на основі зворотного поширення похибки) [28, 29].

Ранні нейроеволюційні алгоритми просто застосовували оператор мутації до вагових коефіцієнтів з'єднань що закодовані у матриці [30]. Але, як зазначалося, фокус дослідників швидко змістився на пошук та реалізацію більш складних систем кодування, для опису та модифікацій графів, що описують топологію ШНМ [31]. Крім того, набули популярності непрямі форми кодування, коли геном – це радше формула для генерування топології ШНМ, ніж прямий опис структури ШНМ на основі значень вагових коефіцієнтів з'єднань між вершинами графу [32, 33]. При цьому можна виділити такі непрямі форми – графічні програми в декартовому генетичному програмуванні [34, 35] або неявне кодування зв'язності в аналоговому генетичному кодуванні (AGE) [36]. Останнє натхнене концепцією генної регуляторної мережі (gene regulatory network) [37].

Збільшення кількості атрибутів, що підлягають змінам у процесі нейроеволюції, підштовхнуло дослідників на пошук більш ефективних алгоритмів, які спроможні обслуговувати зростаючі вимоги. Для прикладу, перехід з еволюції вагових коефіцієнтів у вигляді матриці з'єднань до еволюції топологічних структур ШНМ створив нові проблеми, що мали бути вирішені, а саме: перетин структур з різними топологіями (під час кросинговеру потрібно об'єднати топологічні структури батьків для створення нащадків) та проблема захисту більш складних топологічних структур від вимирання, поки не розкрився їхній повний потенціал.

Одним із найвідоміших алгоритмів, що було винайдено для вирішення зазначених вище проблем, можна вважати нейроеволюцію доповнених топологій (NeuroEvolution of Augmenting Topologies), або коротко – алгоритм NEAT [3, 4, 17]. У цьому алгоритмі проблема перетину топологічних структур вирішується введенням історичних позначок, які дозволяють зіставляти геном батьків без складного топологічного аналізу. Окрім того, для збереження інновацій (новоутворених складних топологічних структур) алгоритмом NEAT реалізована концепція видоутворення, що розділяє популяцію на окремі видові ніші. Це захищає інновації та зберігає різноманітність, уникаючи ситуації, коли найбільш успішні на даному етапі рішення витісняють всі інші в рамках всієї популяції. Як результат, це дає можливість розвиватися більш складним та малоефективним на цю епоху організмам (рішенням), які у довгостроковій перспективі можуть породити найкраще рішення стосовно кінцевого результату еволюції. Таким чином, принципи, закладені в основу алгоритму NEAT, дозволяють більш ефективно знаходити в процесі еволюції все більш складні топології ШНМ, починаючи з найпростіших топологій. Це називається принципом поступового ускладнення топології ШНМ.

Нейроеволюційні алгоритми зазнали широкого застосування в різних сферах науки та практики. Вони досить часто використовуються для навчання ШНМ контролерів у робототехніці – це навіть отримало назву еволюційна робототехніка (evolutionary robotics) [38, 39]. Одним з найвідоміших прикладів із

сфери еволюційної робототехніки можна вважати застосування алгоритму нейроеволюції для створення різних стилів ходьби, що використовуються роботом Sony AIBO [40]. Крім того, нейроеволюційні алгоритми знайшли застосування у відеоіграх, де вони використовуються для створення нового ігрового контенту в режимі реального часу [41] або дозволяють гравцям тренувати штучних персонажів як частина гри [42].

Висновок. Незважаючи на наведені яскраві приклади застосування нейроеволюційних алгоритмів, вони зазвичай стосувалися роботи з ШНМ, які мають досить невеликий розмір за сучасними стандартами – сотні або тисячі з'єднань. У сучасному світі глибокі нейронні мережі мають топологію, яка нараховує мільйони з'єднань між вузлами. А коли мова заходить про великі мовні моделі, то доводиться мати справу вже з мільярдами з'єднань [43]. Тому виникає цілком очевидне питання – чи можна застосувати нейроеволюцію у царині глибокого навчання.

1.2. Нейроеволюція та глибоке машинне навчання

Розквіт епохи глибокого машинного навчання стався через винахід сучасних графічних процесорів (GPU), що дозволяють виконувати тисячі обчислювальних операцій паралельно. Самі ж ідеї, покладені в основу алгоритмів глибокого машинного навчання, такі як зворотне поширення градієнту похибки (backpropagation), у поєднанні з методами оптимізації, що базуються на використанні моменту [44, 45], були відомі вже давно. Крім того, десятиліттями вже були відомі основні архітектурні методи побудови глибоких ШНМ, наприклад, згортка (convolution) [46] та одиниці довгострокової пам'яті (long short-term memory units – LSTM) [47]. Але саме після того, як стало можливим покласти добре відомі ідеї та методи на адекватне «залізо», відбувся вибуховий ріст сфер застосування алгоритмів глибокого машинного навчання, поєднаний з астрономічним потоком інвестицій у цю галузь [19, 20, 24, 43, 48–51].

Аналогічно до прориву у сфері глибокого машинного навчання, можна очікувати прориву і у сфері нейроеволюції з появою сучасних обчислювальних ресурсів. Багато досліджень показали, що нейроеволюція стає гарною альтернативою градієнтним методам для навчання глибоких ШНМ, наприклад, у сфері навчання з підкріпленням.

Навчання з підкріпленням (reinforcement learning) базується на принципі навчання через спроби та похибки в оточенні, що не має прямого нагляду (supervision). Замість нагляду, агент, що навчається, випробовує різні послідовності дій та іноді отримує винагороди. На підставі цього він повинен виробити модель поведінки, яка приведе до найбільшої винагороди у майбутньому. Цей тип навчання більш складний, ніж навчання під наглядом (supervised learning), коли в процесі навчання надаються правильні результати на кожен вхідний сигнал. Навчання з підкріпленням потребує ретельного вивчення середовища для побудови моделі оптимальних дій, навіть за умови, що дії та винагороди за них можуть бути рознесені у часі на значний проміжок.

Алгоритми навчання з підкріпленням були відомі вже давно, але, коли наявні обчислювальні ресурси дозволили об'єднати їх з методами глибокого машинного навчання, знову відбувся якісний стрибок. Якщо до цього моменту навчання з підкріпленням могло бути застосовано тільки до дуже простих проблем з маловимірними просторами вхідних значень (low-dimensional input spaces), то після цього, контролюючи ШНМ отримали змогу відчувати та діяти у багатовимірному просторі станів (high-dimensional state spaces). Це дало змогу застосовувати глибоке навчання з підкріпленням для таких задач, як автоматична гра у відеоігри Atari або симуляція ходи роботів різної морфології [52–54].

У той же час деякі дослідження [55] продемонстрували, що класичний еволюційний алгоритм, під назвою природна еволюційна стратегія (natural evolutionary strategy – NES) [56], показує результати, що є не гіршими за результати, отримані одними з найкращих алгоритмів глибокого навчання з підкріпленням: глибокі Q-мережі (deep Q-networks – DQN) [21] та деякі алгоритми, засновані на методі градієнту політик (policy gradient methods), наприклад, асинхронний актор-критичний агент (asynchronous actor-critic agent – A3C) [57]. При цьому, як показано у дослідженні [55], алгоритм NES напряду винаходить вагові коефіцієнти глибокої нейронної мережі (DNN), що має такий самий розмір, як і ті, що використовувались алгоритмами DQN та A3C для автоматичного проходження ігор Atari. У середньому, глибока ШНМ, яка вивчалась у цьому дослідженні, налічувала більше ніж чотири мільйони параметрів (вагових коефіцієнтів з'єднань), що мають бути вивчені.

Ба більше, через деякий час, простий нейроеволюційний алгоритм перевершив алгоритми DQN та A3C у багатьох іграх Atari [58]. А для певних ігор цей алгоритм навіть перевершив більш просунуті та потужні версії алгоритмів DQN і A3C [59, 60]. Згаданий у [58] нейроеволюційний алгоритм базується на популяції агентів, кожен з яких включає вектор параметрів глибокої нейронної мережі. При цьому в кожній епосі еволюції відбувається незалежна мутація кожного з агентів у популяції та репродукція наступного покоління агентів з використанням вектора параметрів від знайдених агентів-чемпіонів. За наявності

достатніх обчислювальних ресурсів, як показано у дослідженнях [55, 58], глибокі нейроеволюційні алгоритми можуть бути набагато швидшими, ніж їхні традиційні опоненти DQN та A3C. Це пов'язано з тим, що нейроеволюційні алгоритми краще піддаються розпаралелюванню на етапі оцінювання придатності знайденого рішення [61].

В іншому дослідженні [62] авторами було показано, що спрощений нейроеволюційний варіант алгоритму еволюційних стратегій (evolution strategies), що займається навчанням одношарової нейронної мережі, видає state-of-the-art результати на задачах контролю гуманоїдного робота, що вчиться ходити. При цьому результати перевершують те, що було отримано з використанням просунутих методів глибокого навчання з підкріпленням на основі градієнтів політик, таких як глибокі детерміновані градієнти політики (deep deterministic policy gradients) [52], оптимізація політики довірчого регіону (trust region policy optimization) [53] та оптимізація проксимальної політики (proximal policy optimization) [54]. Але, як зазначалося, нейроеволюційні алгоритми дуже добре показали себе у сфері еволюційної робототехніки, тому отримані результати не дуже і дивують [38–40, 63, 64].

Заслужовує уваги також запропонований у [65] метод безпечних мутацій, що дозволяє використовувати гібрид градієнтних методів глибокого навчання та нейроеволюційні алгоритми. У даному методі нейроеволюція використовується для застосування безпечних рандомних мутацій до політики (policy) алгоритму навчання з підкріпленням, реалізованої із застосуванням глибоких нейронних мереж (DNN). Безпечні мутації – це мутації, що дозволяють розширити простір пошуку рішення, але не змінюють поведінку нейронної мережі докорінно та не вимагають додаткової взаємодії із середовищем (симулятор фізики або ігровий симулятор). При цьому глибокі нейронні мережі використовуються для відображення вхідних даних у відповідні дії, та нейроеволюція – для еволюції вагових коефіцієнтів зв'язків між вузлами нейронної мережі. Сила мутації вагових коефіцієнтів зв'язків залежить від чутливості виходів нейронної мережі до конкретного вагового коефіцієнту, що вимагає обчислення градієнта виходів

щодо вагових коефіцієнтів замість градієнта помилки, як у звичайному глибокому навчанні. Цей підхід значно збільшує здатність методу нейроеволюції на основі простого генетичного алгоритму знаходити рішення у високовимірних доменах, які потребують глибоких та/або рекурентних нейронних мереж, включаючи домени, які потребують обробки необроблених пікселів (raw pixels). Він значно покращив продуктивність нейроеволюційного алгоритму та дозволив оперувати нейронними мережами з більш ніж сотнею шарів.

Інший метод гібридизації, запропонований у [66, 67], поєднує еволюційні алгоритми, що здійснюють пошук безпосередньо в просторі параметрів нейронної мережі, з алгоритмами градієнту політик (policy gradient) та алгоритмами Q-навчання, які здійснюють пошук у просторі дій і потім змінюють параметри нейронної мережі через зворотне поширення похибки, щоб зробити вигідні дії більш імовірними. Цей підхід створює випадкові збурення параметрів для керування послідовним дослідженням (як еволюційні алгоритми), але потім відображає успішні дії у вагових параметрах з'єднань через зворотне поширення (backpropagation) похибки.

Вражає те, що досягнуті успіхи, описані вище, були реалізовані за допомогою простих нейроеволюційних алгоритмів. Однак спільнота дослідників нейроеволюції розробила багато складних методів, які можуть значно підвищити ефективність цих простих алгоритмів. Багато з цих методів ґрунтуються на спостереженні, що еволюція, як у природному, так і в обчислювальному середовищі, є потужним двигуном інновацій [68]. Ці сучасні методи намагаються алгоритмічно відтворити цю природну креативність для пошуку більш ефективних нейронних мереж.

Висновок. Підсумовуючі, можна виділити такі основні відмінності між нейроеволюційними алгоритмами та методами глибокого машинного навчання.

Підхід до оптимізації нейронних мереж:

- **Глибоке навчання:** Використовує методи оптимізації на основі градієнта похибки, такі як зворотне поширення похибки зі стохастичним градієнтним спуском (SGD), для налаштування вагових коефіцієнтів

нейронної мережі та мінімізації помилок. Цей підхід особливо ефективний у навчанні під наглядом і навчанні з підкріпленням.

- **Нейроеволюція:** Застосовує еволюційні алгоритми, натхненні природною еволюцією, для оптимізації нейронних мереж. Замість коригування вагових коефіцієнтів на основі градієнтів, нейроеволюція досліджує різні нейронні архітектури, гіперпараметри та правила навчання через еволюційний процес.

Різноманітність і паралелізація:

- **Глибоке навчання:** Зосереджується на налаштуванні вагових коефіцієнтів, що обмежує дослідження простору пошуку рішень.
- **Нейроеволюція:** Підтримує широку популяцію рішень, що дозволяє розгалужене дослідження простору пошуку рішень та ефективно розпаралелювання цього процесу.

Одночасно, як зазначено вище, існує багато способів створення гібридних систем, які поєднують нейроеволюційні методи з методами глибокого машинного навчання, щоб скористатися перевагами обох підходів для вирішення практичних завдань.

1.3. Різноманітність поведінки та пошук новизни

Однією з визначних характеристик природної еволюції є вражаюче різноманіття складних і функціональних організмів, які вона породила, — від складних механізмів одноклітинних організмів до величезних спільнот клітин, що формують тіла тварин різних видів, включаючи людей. Це масштабне паралельне дослідження способів життя та поведінки не лише саме по собі є захоплюючим, але і, ймовірно, відіграло ключову роль в еволюції людського інтелекту, оскільки саме різноманітність робить можливими інновації [69, 70]. Виходячи з цього, прагнення до різноманітності є також важливим для нейроеволюції, якщо розглядати її як можливий шлях до штучного інтелекту (ШІ) людського рівня. Саме тому нейроеволюція та еволюційні обчислення загалом тривалий час акцентували увагу на різноманітності (diversity) [71, 72]. І дійсно, еволюційні алгоритми, адаптуючи набір рішень, природньо підходять для одночасного дослідження різноманітних варіантів.

Прості алгоритми нейроеволюції копіюють рішення з найвищим коефіцієнтом пристосованості, відкидаючи менш пристосовані варіанти. Це призводить до зближення популяції рішень і знаходження остаточного рішення. Однак це працює лише за умови, що популяція рішень має схожі значення пристосованості. Водночас прості алгоритми втрачають знайдені рішення та підрозв'язки через такі три ефекти [72]:

- негативний тиск добору;
- шум добору;
- знищення через роботу генетичного оператора.

Негативний тиск добору виникає через нормальну роботу процесу селекції, коли менш придатні рішення зникають з обмеженої популяції. Шум добору є наслідком варіативності процесу селекції, тобто в обмеженій популяції через випадковий добір серед рівних рішень деякі гарні рішення зникають, обмежуючи простір пошуку. Крім того, при застосуванні генетичних операторів, таких як кросинговер та мутація, деякі гарні рішення можуть бути знищені безпосередньо.

Традиційно, більшість зусиль для подолання вищезгаданих недоліків нейроеволюційного процесу були спрямовані на заохочення різноманітності в генетичному просторі, тобто *просторі параметрів*, щоб уникнути локальних оптимумів. Основна ідея полягає в тому, що якщо пошук рішення наблизився до локального оптимуму, то стимулювання досліджень (*exploration*), які віддаляють від цього оптимуму, може бути достатнім для відкриття нового перспективного градієнта покращення. До таких підходів можна віднести скупчення (*crowding*) [73], коли новий індивід замінює іншого, найбільш генетично схожого до нього, та групування за значеннями придатності (*explicit fitness-sharing*) [71], відповідно до якого індивіди групуються за генетичною віддаленістю і обмежуються кількістю осіб у їхньому кластері.

Хоча різноманітність у *просторі параметрів* іноді є ефективною, вона часто не забезпечує широкого спектру різних типів поведінки [74]. Це відбувається тому, що існує безліч способів налаштувати вагові коефіцієнти нейронної мережі, які призводять до однакової поведінки. Це можливо завдяки зміні масштабу вагових коефіцієнтів із збереженням функції [75], перестановці вузлів [76] або надлишковим відображенням (наприклад, багато різних налаштувань ваги можуть спричинити однакове падіння робота). Іншими словами, хоча легко створити різноманітні (але схожі за поведінкою) вектори параметрів, уникнення локальних оптимумів часто вимагає дослідження різноманітної поведінки, як це робить біологічна еволюція, що є важливим при вирішенні проблем твариною [77] або людиною [78].

Через зазначене обмеження генетичного різноманіття (простір параметрів), сучасні підходи безпосередньо винагороджують різноманітність поведінки [74, 79]. Крім того, подальші дослідження привели до пов'язаних ідей, таких як пряма еволюція бажаних якостей, таких як цікавість [80], здатність до розвитку [81] або генерування подиву [82]. Наприклад, один з підходів включає багатоцільовий еволюційний алгоритм (*multiobjective evolutionary algorithm*) [83, 84], який винагороджує індивідів як за підвищення рівня їхньої пристосованості (*fitness score*), так і за відмінності від інших індивідів у визначених

експериментатором характеристиках поведінки. Таким чином, пошук може органічно спрямовувати різних індивідів у популяції до різних компромісів між дослідженням відповідних поведінкових аспектів (exploration) і оптимізацією ефективності (exploitation).

Одним із відомих варіантів алгоритмів, котрі ґрунтуються на припущенні, що різноманітність є основним продуктом природної еволюції, є «пошук новизни» (novelty search – NS) [74]. Цей алгоритм винагороджує нейронні мережі лише за нову поведінку порівняно з раніше створеними індивідами. Тобто, алгоритм не чинить тиску на покращення відповідно до традиційних мір пристосованості (fitness) або ефективності (performance). Головна ідея полягає в тому, що популяція буде поширюватися протягом багатьох поколінь еволюції, охоплюючи широкий спектр поведінки. І, хоча градієнт дивергенції в генетичному просторі може бути неінформативним – оскільки багато різних геномів можуть виробляти однакову нецікаву поведінку, градієнт поведінкової новизни часто містить найбільш корисну інформацію про домен дослідження. Іншими словами, щоб зробити щось нове, часто потрібні навички, які враховують обмеження домену. Наприклад, щоб навчитися виконувати новий трюк на гірських лижах, потрібні рівновага та координація, які можна отримати, просто катаючись. Дійсно, у деяких областях навчання з підкріпленням пошук саме новизни поведінки перевершує цілеспрямований пошук (goal-oriented search) [74, 85]. Спочатку цей підхід застосовувався до невеликих нейронних мереж. Але нещодавно було показано, що пошук новизни добре масштабується до застосування у багатовимірних задачах навчання з підкріпленням [58, 85], покращуючи продуктивність і демонструючи, як нейроеволюція може отримати користь від сучасних обчислювальних потужностей.

Відповідно до визначених властивостей алгоритму пошуку новизни та загальної концепції еволюції як такої, що керована різноманітністю, виникла нова галузь досліджень нейроеволюції, котра вивчає алгоритми «якісного різноманіття» (quality diversity – QD) [86]. Цей алгоритм розроблено, щоб висвітлити різноманітність можливих високоякісних рішень проблеми. Подібно

до того, як природна еволюція виявила добре пристосовані організми в численних екологічних нішах. Наприклад, якщо метою є здатність істоти швидко літати, то виявлення соколу не завадить виявленню мухи – обидва можуть швидко літати порівняно з істотами схожої морфології.

Прикладами ранніх розробок у напрямку алгоритмів якісного різноманіття є пошук новизни з локальною конкуренцією (novelty search with local competition – NSLC) [87] та багатовимірний архів фенотипових еліт (multi-dimensional archive of phenotypic elites – MAP-Elites) [88]. Кожен з цих алгоритмів використовує різні підходи до інтеграції еволюційного тиску в процес диверсифікаційного пошуку рішень. NSLC є модифікацією багатоцільового еволюційного алгоритму, що дозволяє оптимізувати популяцію як для різноманітних, так і для локально оптимальних індивідів. MAP-Elites – це простий, але потужний алгоритм, який розділяє простір можливих варіантів поведінки на окремі ніші, кожна з яких містить одного чемпіона – найефективнішого індивіда цього типу з усіх знайдених на даний момент. При цьому, конкуренція відбувається лише локально, але мутація в межах однієї ніші може породити нового чемпіона в іншій, створюючи ефекти, схожі на екзаптацію. Тобто, досягнення високої ефективності в одній ніші може стати кроком до успіху в іншій. На виході цих алгоритмів зазвичай отримуємо набір різноманітних, але ефективних рішень, на відміну від єдиного оптимального рішення, як в інших алгоритмах. Такий набір називається репертуаром. Як приклад можна навести публікацію [64], де описано застосування алгоритму MAP-Elites для створення різноманітного репертуару вискоелективних способів ходьби. Завдяки цьому робот, навіть зазнавши пошкодження, міг швидко відновлюватися, обираючи найкращі способи ходьби з наявного репертуару, які залишалися ефективними попри отримані пошкодження.

Галузь вивчення алгоритмів якісного різноманіття активно розвивається і має ще багато прикладів успішного застосування [89–93].

1.4. Основні поняття нейроеволюційних алгоритмів

Насамперед важливо розглянути основні поняття, що використовуються для опису та побудови нейроеволюційних алгоритмів.

Генетичні оператори знаходяться в самому серці кожного еволюційного алгоритму. Існує два основні генетичні оператори: мутація і кросинговер (рекомбінація).

Оператор мутації виконує важливу роль збереження генетичного різноманіття популяції в процесі еволюції і запобігає зупинці в локальних мінімумах, коли хромосоми організмів у популяції стають надто схожими. Оператор мутації змінює один або кілька генів у хромосомі відповідно до певної ймовірності. На рис. 1.1 відображено найбільш поширені типи операторів мутації.

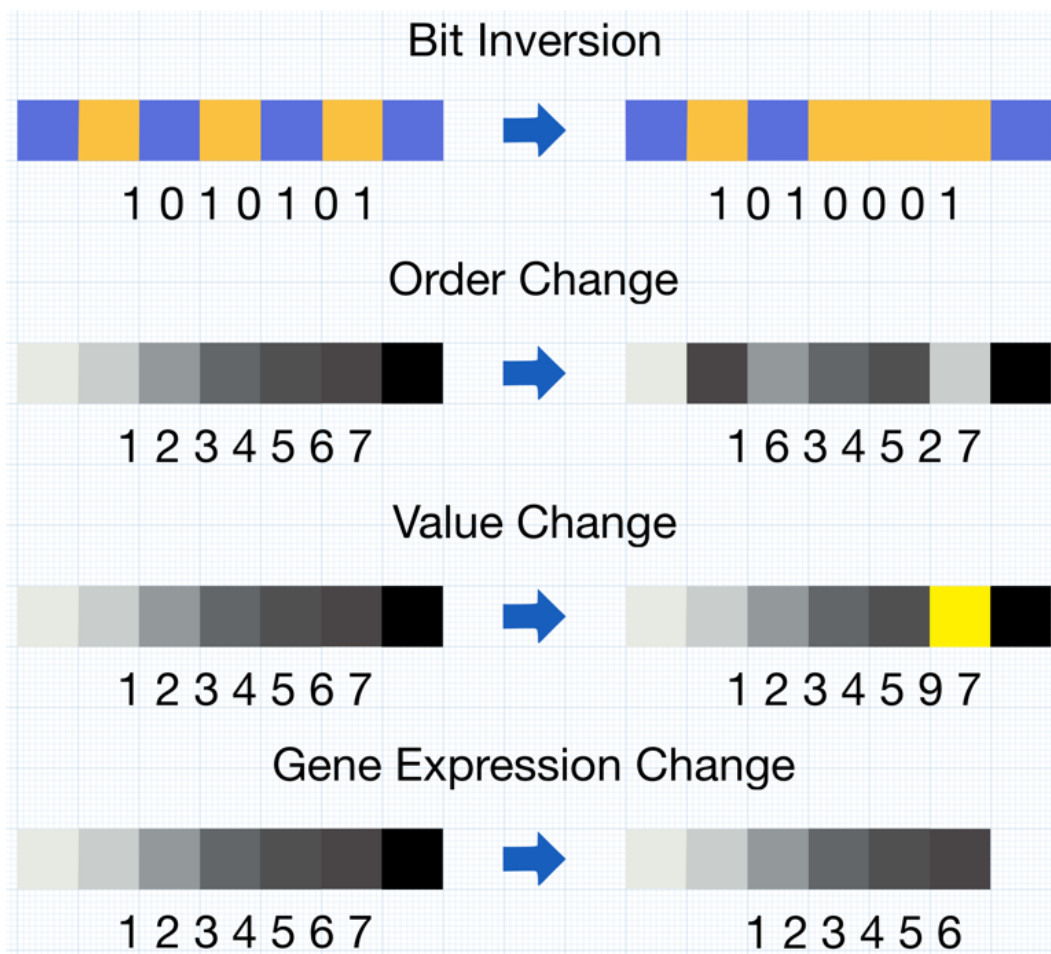


Рис. 1.1. Типи операторів мутації (за матеріалами [4])

Використовуваний тип оператора мутації залежить від типу генетичного кодування, застосованого в конкретному генетичному алгоритмі. Серед основних варіантів оператора мутації можна виділити такі:

- **бітова інверсія (bit inversion):** випадково обраний біт інвертується (бінарне кодування);
- **зміна порядку (order change):** два випадково обрані гени змінюють своє розташування в геномі (кодування пермутації);
- **зміна значення (value change):** до гена у випадковій позиції додається невелике значення (кодування значення);
- **зміна експресії гена (gene expression change):** випадково обраний ген додається до генотипу або вилучається з нього (структурне кодування).

Генотипи можуть бути закодовані за допомогою схем генетичного кодування з фіксованою або змінною довжиною хромосоми. Перші три типи мутацій можуть застосовуватися до обох схем кодування. Останній тип мутації можливий лише в генотипах зі змінною довжиною.

Оператор кросинговера (рекомбінації) дозволяє стохастично створювати нові покоління (рішення) з існуючих популяцій, комбінуючи генетичну інформацію від двох батьків для утворення нащадків. Це дозволяє об'єднувати частини хороших рішень від батьківських організмів, що потенційно може привести до кращого потомства. Зазвичай після операції кросинговера отримане потомство піддається мутації перед додаванням до популяції наступного покоління. На рис. 1.2 показано різні варіанти реалізації оператора кросинговера.

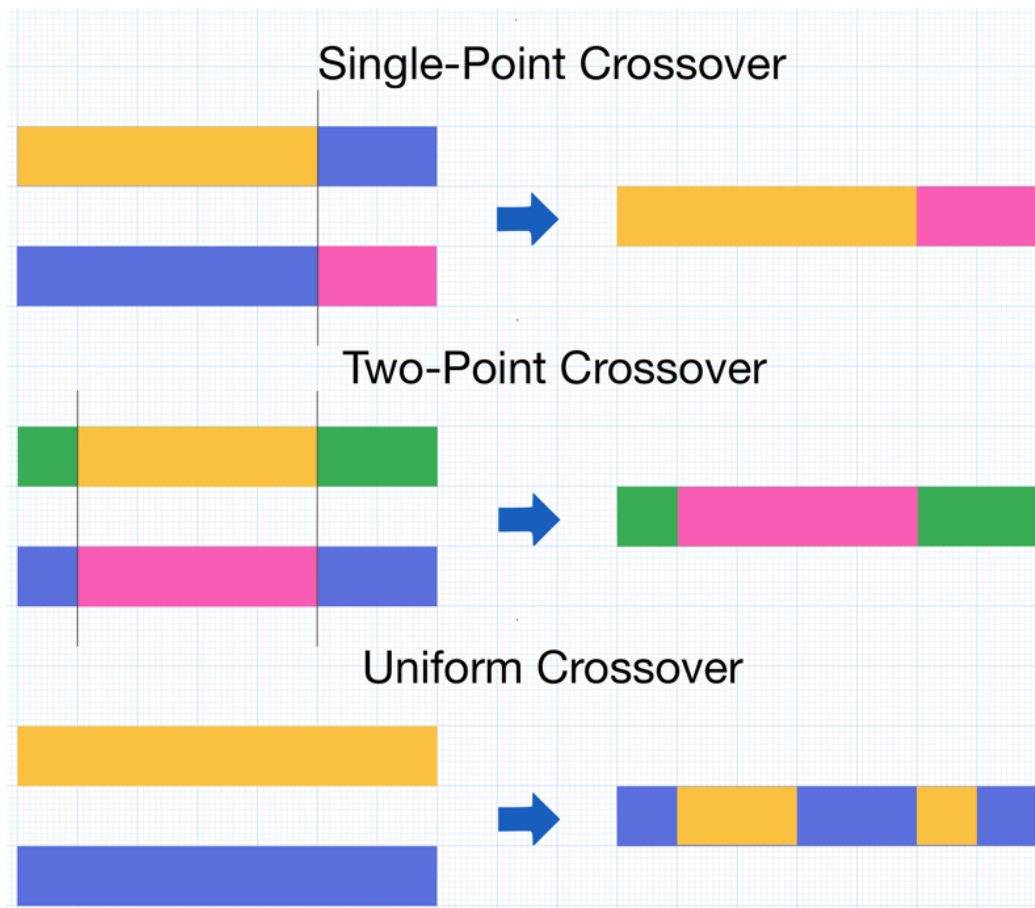


Рис. 1.2. Типи операторів кросинговера (за матеріалами [4])

Різні типи операторів кросинговера також залежать від схеми кодування геному, що використовується конкретними алгоритмами. Найбільш поширеними є:

- **Одноточковий кросинговер (single-point crossover):** вибирається випадкова точка перетину, частина геному від початку до цієї точки копіюється в потомство від одного з батьків, а залишок — від іншого.
- **Двоточковий кросинговер (two-point crossover):** випадковим чином вибираються дві точки перетину. Частина геному від початку до першої точки копіюється від першого батька, частина між першою і другою точками — від другого батька, а залишок знову копіюється від першого батька.
- **Уніфікований кросинговер (uniform crossover):** гени випадковим чином копіюються або від першого, або від другого батька.

Схема кодування геному є одним з найважливіших аспектів при розробці нейроеволюційного алгоритму. Вона визначає, як структура штучної нейронної мережі може бути компактно закодована у вигляді геному, до якого можуть бути застосовані вищезазначені генетичні оператори. Існують дві основні схеми: пряма та непряма. Далі розглянемо кожен більш детально.

Пряма схема кодування геному спершу використовувалася в еволюційних алгоритмах для кодування ШНМ з фіксованою топологією, тобто з топологією, визначеною дослідником заздалегідь. Згідно з цим підходом генетичне кодування реалізується як вектор дійсних чисел, що визначають силу (вагові коефіцієнти) з'єднань між вузлами мережі. При цьому генетичні оператори застосовуються для зміни значень вагових коефіцієнтів у векторі (мутація) та комбінування векторів батьківських геномів для створення геномів нащадків (кросинговер). Такий підхід дозволяє легко застосовувати генетичні оператори, але має значні недоліки. Один з головних недоліків полягає в тому, що топологія ШНМ задається заздалегідь і залишається незмінною для усіх поколінь нащадків протягом усіх епох еволюції. Це суперечить основним принципам природної еволюції, згідно з якими змінюються не тільки особливі властивості організмів, але й їхня фізична структура, що дозволяє максимально розширювати простір пошуку оптимальних рішень у процесі еволюції. На рис. 1.3 наведено приклад еволюційного процесу з використанням прямої схеми кодування.

Щоб усунути недоліки прямого кодування, Кеннет О. Стенлі запропонував інший підхід до еволюційного процесу, який реалізовано в алгоритмі NEAT [3]. У цьому алгоритмі генетичні оператори змінюють не тільки вагові коефіцієнти з'єднань між вузлами, але й саму топологію цих з'єднань. Таким чином, під час еволюції досліджуються різні топології з різними наборами вагових коефіцієнтів. Це відкрило нові можливості для застосування нейроеволюційних алгоритмів у ширшому колі практичних задач. Однак алгоритм NEAT належить до сімейства алгоритмів з прямим кодуванням, що обмежує його використання лише для побудови відносно невеликих топологій ШНМ, оскільки зі

збільшенням кількості вузлів у ШНМ кількість необхідних обчислень зростає нелінійно.

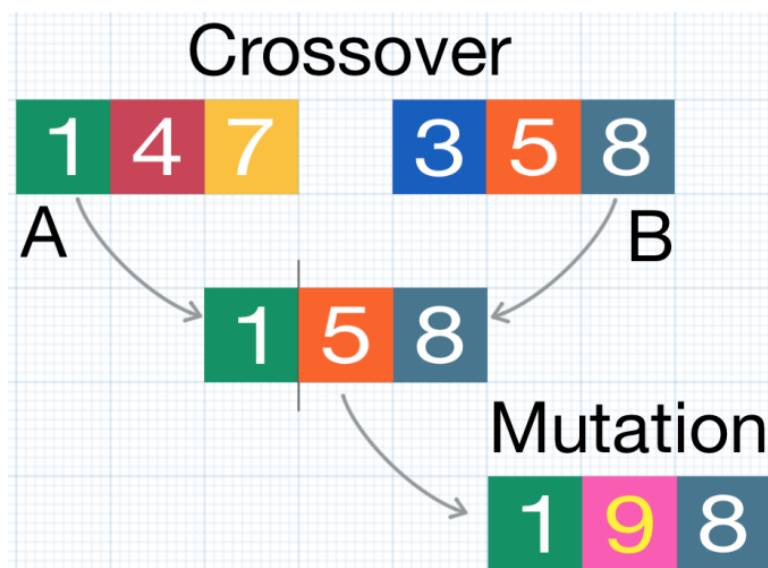


Рис. 1.3. Еволюційний процес з використанням прямої схеми кодування геному (за матеріалами [4])

Непряма схема кодування геному була також запропонована Кеннетом О. Стенлі [63] для подолання обмежень на розмір штучних нейронних мереж, які можна створити за допомогою нейроеволюційних алгоритмів. Це кодування, натхненне тим, як генотипи кодуються в ДНК, базується на тому, що структура фізичної реальності побудована навколо геометрії та просторових закономірностей, де природна симетрія є всюдишущою. Таким чином, розмір кодування будь-якого фізичного процесу можна значно зменшити, повторно використовуючи специфічний набір повторюваних шаблонів і формуючи кінцевий результат. Запропонований алгоритм, відомий як Hypercube-based NeuroEvolution of Augmenting Topologies (HyperNEAT) [94], призначений для створення великомасштабних ШНМ відповідно до виявлених геометричних закономірностей у вирішуваній задачі. Основним компонентом HyperNEAT є специфічна ШНМ під назвою Compositional Pattern Producing Network (CPPN) [95], яка створюється за допомогою алгоритму NEAT під час нейроеволюційного процесу. CPPN є функцією декартового простору, що кодує з'єднання між вузлами більшої ШНМ у вигляді певних точок у багатовимірному гіперпросторі.

За допомогою цього методу менша ШНМ може бути використана для представлення структури значно більшої ШНМ з кількістю вузлів, що відповідає кількості вузлів у ШНМ, які застосовуються в методах глибокого навчання. Це створює нові можливості для використання нейроеволюційних алгоритмів у задачах розпізнавання образів, які зазвичай вирішуються за допомогою глибоких ШНМ.

Коеволюція. У природі різні види часто еволюціонують разом через взаємодію один з одним. Така міжвидова взаємодія називається коеволюцією і є потужною рушійною силою природної еволюції [96]. Можна виділити такі типи коеволюції, що зустрічаються в природі:

- **Мутуалізм** – коли два або більше видів мирно співіснують і отримують взаємну вигоду один від одного.
- **Конкуренція**, яка може бути двох типів:
 - **Хижацтво** – коли один вид вбиває представників іншого виду і споживає їхні ресурси.
 - **Паразитизм** – коли представники одного виду використовують ресурси іншого виду, але не вбивають їх.
- **Коменсалізм** – коли представники одного виду отримують переваги без шкоди або використання ресурсів інших видів [96].

Більшість з перелічених коеволюційних стратегій були використані дослідниками для створення еволюційних алгоритмів. Особливий інтерес становить коменсалізм, де еволюціонують дві популяції: популяція організмів-вирішувачів та популяція кандидатів у цільові функції. У представленій роботі цей підхід буде розглянуто для вирішення задачі навігації у складному середовищі за допомогою алгоритму NEAT.

Модулярність та ієрархічні структури. Дослідження структури мозку людини та інших вищих істот показали, що мозок не є монолітним, а утворює складну ієрархію модульних структур. Крім того, обмеження на швидкість передавання сигналу через нервові шляхи в біологічних тканинах призводять до природної реалізації принципу локальності, коли пов'язані завдання

обробляються геометрично суміжними структурами в мозку. Ці структурні особливості мозку привернули увагу багатьох дослідників нейроеволюційних алгоритмів і знайшли застосування, зокрема, в алгоритмі Evolvable-Substrate Hypercube-based NeuroEvolution of Augmenting Topologies (ES-HyperNEAT) [97].

Висновок. У цьому розділі проведено систематичний аналіз фундаментальних концепцій нейроеволюційних алгоритмів, що становлять теоретичну основу подальших досліджень і розробок у сфері еволюційної побудови топології штучних нейронних мереж. Розглянуто ключові аспекти класичних та сучасних підходів, включаючи концепцію еволюційного представлення нейронної мережі, принципи формування геномів і генотип–фенотипне відображення, механізми видоутворення та інноваційних номерів, а також особливості еволюції вагових коефіцієнтів і структурних модифікацій топології мереж.

1.5. Особливості алгоритму NEAT

Одним з відомих та широко застосованих нейроеволюційних алгоритмів є нейроеволюція доповнених топологій (NeuroEvolution of Augmenting Topologies - NEAT) [3]. Своєї популярності цей алгоритм набув завдяки принципам та ідеям, що покладені в його основу.

Найважливішою особливістю алгоритму NEAT, яка визначає його потенціал, є здатність поступово розвивати топологію штучних нейронних мереж у процесі навчання. Важливість поступового розширення топології мережі стає очевидною, якщо розглянути, як це реалізовано за допомогою NEAT. Досліджуючи простір пошуку рішення, алгоритм NEAT починає з найпростішої базової топології мережі, яка включає лише вхідні та вихідні вузли ШНМ-вирішувачів. Після цього, з кожною епохою еволюції топологія вирішувачів поступово ускладнюється, та найскладніші багатовимірні топології оцінюються лише на завершальних етапах еволюційного процесу. Крім того, найбільш підхожі топологічні структури, знайдені під час еволюції, зберігаються алгоритмом NEAT через покоління, це його особливість.

Подібний підхід поступового ускладнення топології ШНМ-вирішувачів значно скорочує простір пошуку рішення під час навчання. Широкий простір пошуку рішення є одним із значних недоліків інших нейроеволюційних алгоритмів, який алгоритм NEAT ефективно вирішує завдяки своїм ключовим конструктивним особливостям, які будуть розглянуті далі.

Головною метою алгоритму NEAT є зменшення складності знайденої топології мережі в процесі еволюції. І це стосується не лише кінцевого продукту еволюції, але й усіх проміжних поколінь ШНМ-вирішувачів. Для досягнення цього в алгоритмі NEAT реалізовано такі ключові функції:

- **Розширення топології шляхом поступового ускладнення.** Навчання починається з простої базової топології, яка поступово ускладнюється на основі аналізу простору пошуку рішень. При цьому алгоритм NEAT здатний зберігати найменш складну топологію, яка може генерувати остаточне рішення. Як результат, остаточне рішення, ймовірно, матиме

найбільш оптимальну топологію ШНМ. При цьому процес навчання буде максимально наближеним до оптимального за часом завдяки зменшенню обчислювального навантаження, оскільки найскладніші топології будуть обчислюватися лише на завершальних етапах еволюції.

- **Видоутворення.** Це ще одна важлива особливість алгоритму NEAT, яка розподіляє популяцію на еволюційні ніші, використовуючи генетичну відстань між організмами (схожість геномів). Головна мета видоутворення — зменшити негативний адаптаційний тиск, що виникає через збільшення складності нового організму, коли до геному потомства додаються нові вузли або з'єднання. Нова, більш складна топологія може ввести новий напрямок у процес дослідження простору пошуку рішень, що може забезпечити успіх в майбутньому. Однак у поточний період еволюції це досягнення може бути знищене існуючими організмами-чемпіонами через тимчасове зниження придатності знайденого рішення. Тому, щоб підтримувати тиск на виживання на прийнятному рівні, ми дозволяємо еволюційну конкуренцію лише між організмами в одній ніші. Це допомагає зберігати цікаві інновації в популяції, навіть якщо вони не є найефективнішими в поточний період еволюції.
- **Мутація.** Оператор мутації відіграє ключову роль у підтримці генетичного різноманіття популяції під час еволюції та запобігає зупинці цільової функції на локальних мінімумах, коли хромосоми організмів стають надто схожими. Цей оператор змінює один або кілька генів у хромосомах відповідно до ймовірності мутації, визначеної експериментатором. Випадкові зміни в хромосомах ШНМ-вирішувачів дозволяють еволюційному процесу досліджувати нові області в просторі пошуку рішень і знаходити кращі рішення. Оператор мутації є загальною характеристикою майже всіх еволюційних алгоритмів. Однак його специфічна реалізація алгоритмом NEAT змінює не тільки ваги зв'язків між нейронами, але й саму структуру топології ШНМ-вирішувачів.

- **Інноваційні числа або історичні позначки.** Це одна з найважливіших особливостей алгоритму NEAT, яка вирішує проблему перекриття частин геномів ШНМ-вирішувачів. Ці частини створюють подібні топології, що оцінюють одну й ту саму функцію, але були інтегровані в різні структури геному під час репродуктивного кросинговеру в різних епохах еволюції. За рахунок історичних позначок стає можливим порівняння топологій ШНМ-вирішувачів без застосування методів складного топологічного аналізу.
- **Кросинговер або рекомбінація геномів.** Це процес, який дозволяє двом або більше організмам-чемпіонам у межах популяційної ніші створювати потомство з важливими характеристиками обох батьків. Під час цього процесу інноваційні числа допомагають ідентифікувати гени, що перекриваються (збігаються), а також надлишкові або роз'єднані гени. Гени, що збігаються, мають однакове значення інноваційного числа і кодують одну і ту саму топологічну структуру, хоча можуть мати різні значення інших параметрів (наприклад, ваги з'єднань). Під час репродукції гени, що збігаються, випадково вибираються від одного з батьків і успадковуються нащадками. Водночас гени, які присутні лише в геномі одного з батьків, можуть бути роз'єднаними (в межах чисел інновацій іншого батька) або надлишковими (поза числами інновацій іншого батька). Роз'єднані або надлишкові гени успадковуються нащадками від найбільш придатних батьків або від обох батьків випадковим чином, залежно від типу оператора кросинговеру (багатоточкове, багатоточкове середнє, одноточкове або рівномірне).

Далі детально розглянемо реалізацію кожної ключової функції алгоритму NEAT та його варіантів.

Схема кодування геному NEAT. Алгоритм NEAT базується на спеціально розробленій схемі кодування геному, яка забезпечує легке зіставлення відповідних генів хромосом батьків під час операції кросинговеру. Крім того, це кодування є лінійним представленням шаблону з'єднань між вузлами закодованої штучної нейронної мережі, як показано на рис. 1.4.

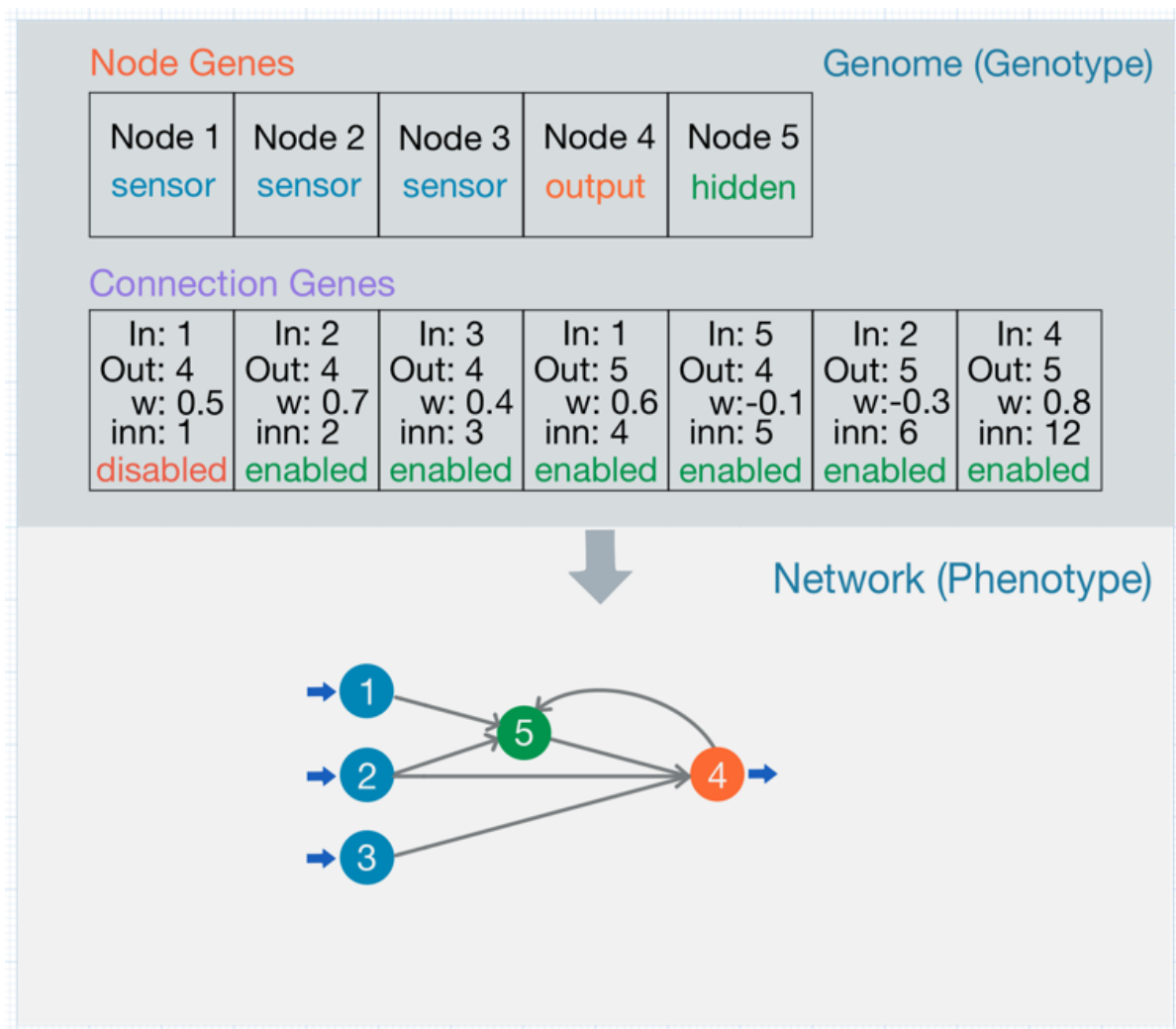


Рис. 1.4. Схема кодування геному NEAT (адаптовано за [15])

Кожен геном у схемі представлений як список генів з'єднань, які кодують інформацію про зв'язки між вузлами штучної нейронної мережі. Крім того, схема кодування включає інформацію про вузли ШНМ у вигляді вузлових генів, що містять такі дані: ідентифікатор вузла, тип вузла та тип функції активації, що застосовується до цього вузла.

Кожен ген з'єднання відіграє важливу роль у нейроеволюційному процесі поступового розширення топології закодованої ШНМ і містить такі параметри, що характеризують зв'язок між двома вузлами ШНМ:

- ідентифікатор вхідного вузла ШНМ;
- ідентифікатор вихідного вузла ШНМ;
- сила або ваговий коефіцієнт з'єднання;
- прапорець, що вказує, чи є це з'єднання активним (вираженим) чи ні;

- історична позначка (інноваційне число), що дозволяє легко зіставляти окремі гени.

Відповідно нижня частина діаграми на рис. 1.4 відображає фенотип або топологію ШНМ, закодованої геномом, представленим у верхній частині.

Структурні мутації. Мутації геномів є важливим рушієм еволюційного процесу як у природі, так і в методі нейроеволюції, оскільки вони дозволяють розширювати простір пошуку рішень і сприяють адаптації до змін у середовищі. У випадку алгоритму NEAT, оператори мутації змінюють не лише вагові коефіцієнти, але й саму структуру фенотипу штучної нейронної мережі. Серед структурних операторів мутації можна виділити два основних:

- додавання нового з'єднання між вузлами ШНМ;
- додавання нового вузла ШНМ.

При застосуванні оператора структурної мутації, кожен новий ген (ген з'єднання або вузловий ген) отримує монотонно зростаючий номер інновації (історичну позначку), як показано на рис. 1.5.

Крім операторів додавання існують також оператори видалення генів, що мають зворотний ефект.

Кросинговер з використанням історичних позначок. У процесі еволюції кожне нове покоління формується шляхом рекомбінації геномів найкращих індивідів попереднього покоління, що дозволяє передавати накопичений досвід через генерації поколінь. Для порівняння геномів батьків необхідно мати можливість зіставляти окремі гени в їхніх хромосомах, щоб визначити, які гени є спільними, а які відрізняються. Сила алгоритму NEAT полягає в тому, що він дозволяє легко здійснювати це зіставлення без складного топологічного аналізу. Це стало можливим завдяки історичним позначкам, які дозволяють чітко відстежувати хронологію структурних мутацій і однозначно ідентифікувати гени незалежно від їхнього розташування в геномі або значення вагових коефіцієнтів з'єднань, які вони кодують. Використання оператора кросинговеру з історичними позначками показано на рис. 1.6.

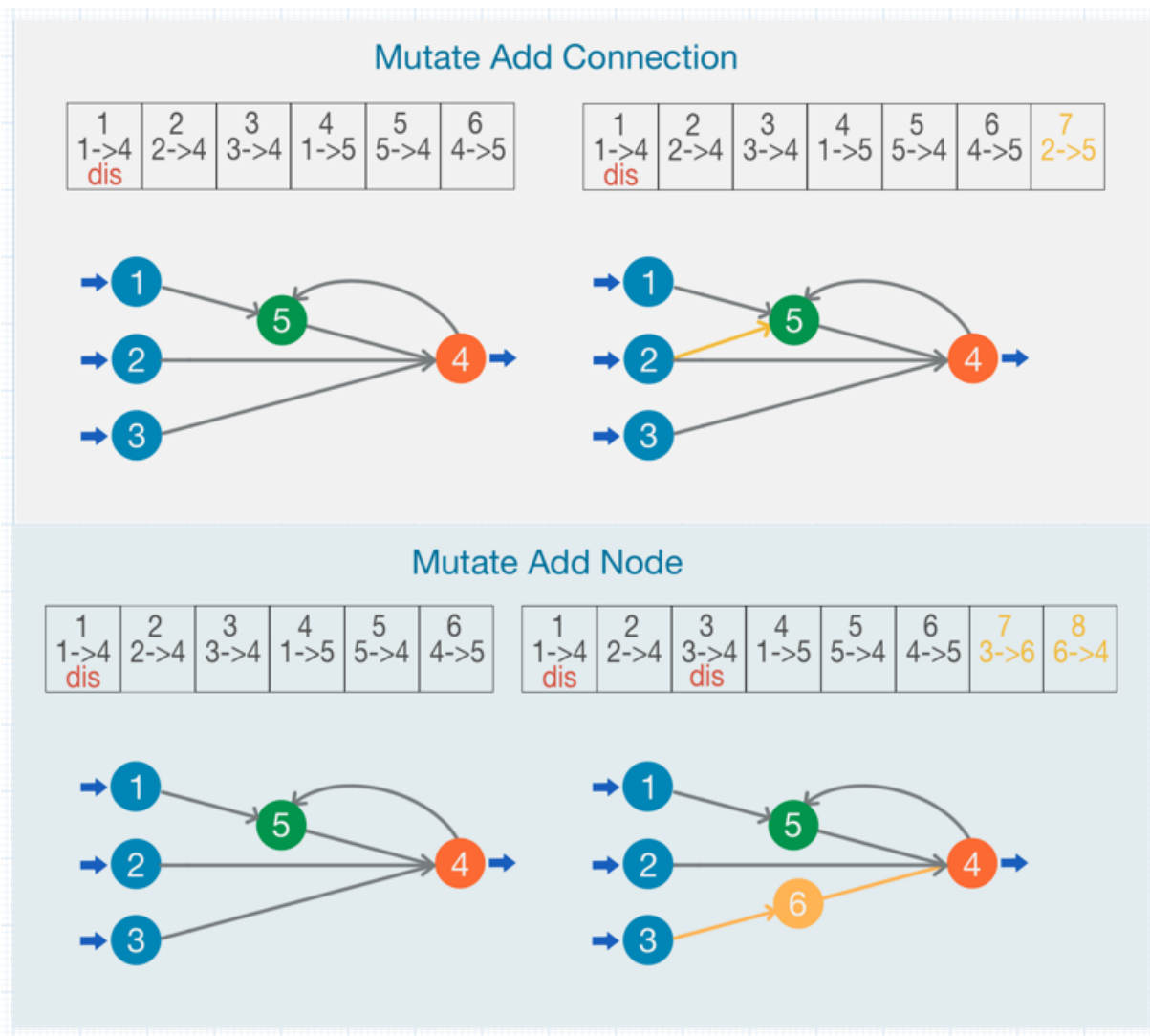


Рис. 1.5. Структурні мутації алгоритму NEAT (адаптовано за [15])

Під час кросингверу кожен нащадок наслідує історичні позначки генів від геномів батьків. Таким чином, значення інноваційних чисел для окремих генів залишаються постійними протягом усього процесу еволюції. Якщо історичні позначки в геномах батьків збігаються, то гени вважаються однаковими (кодують ту саму топологічну структуру фенотипу ШНМ) і передаються нащадкам випадковим чином від одного з батьків. Якщо ж історичні позначки різні, то гени вважаються роз'єднаними (в межах чисел інновацій іншого батька) або надлишковими (поза числами інновацій іншого батька). Роз'єднані або надлишкові гени успадковуються нащадками від найбільш придатних батьків або від обох батьків випадковим чином, залежно від типу оператора

кросинговеру (багатоточковий, багатоточковий середній, одноточковий або рівномірний).

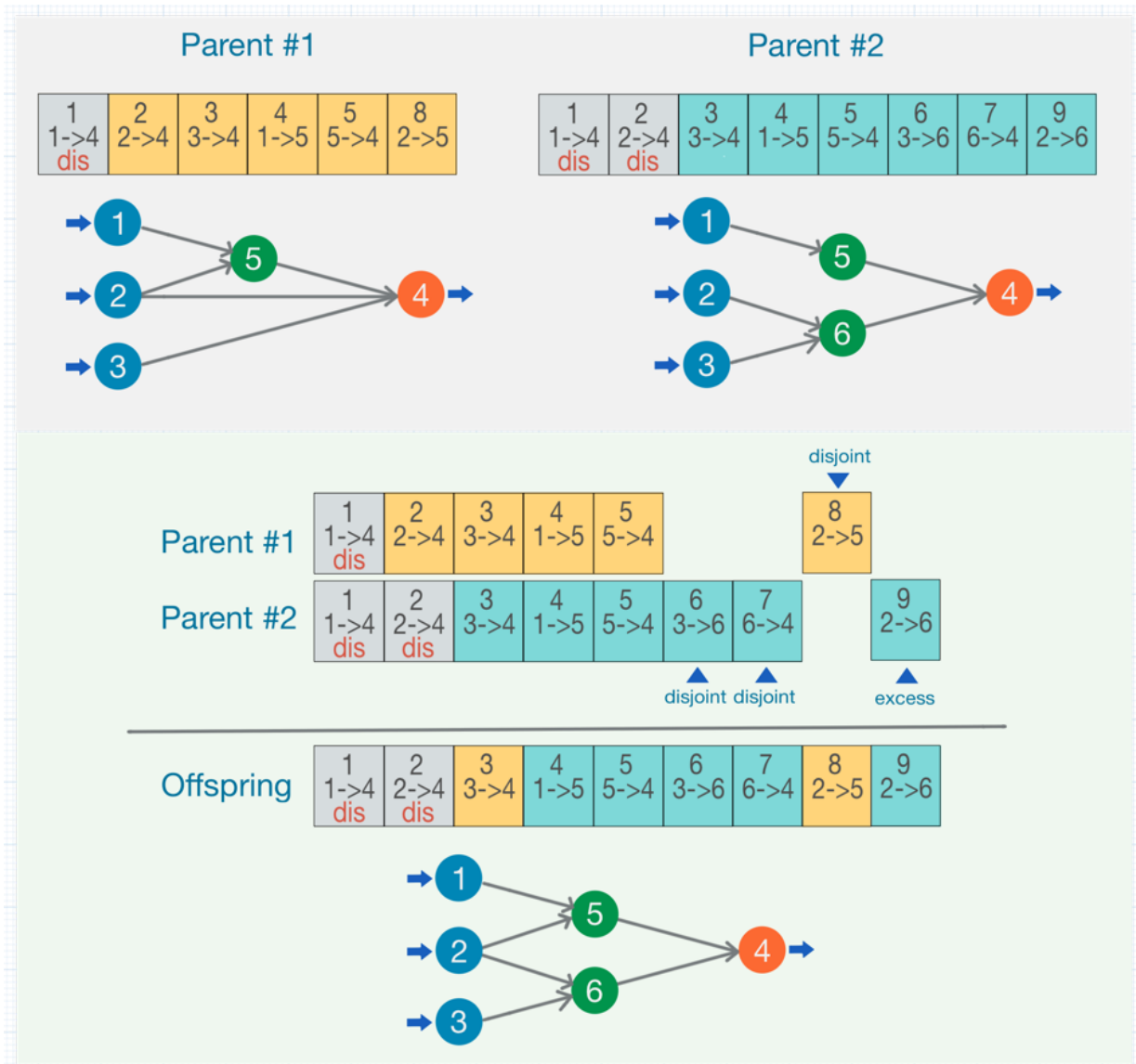


Рис. 1.6. Кросинговер з використанням історичних позначок (адаптовано за [15])

Видоутворення. У процесі еволюційного розвитку організми здатні з покоління в покоління формувати нові топології, а також ініціювати та підтримувати власні топологічні модифікації. Водночас слід зауважити, що мережеві структури невеликого розміру оптимізуються значно швидше порівняно з великими. Це зумовлює зниження імовірності виживання геному-нащадка після введення до його структури нового вузла чи зв'язку. Як наслідок, нещодавно доповнені топології зазнають *негативного еволюційного тиску*, що

пов'язано з тимчасовим зменшенням рівня пристосованості організмів у популяції.

Разом з тим, саме у нових топологічних конфігураціях можуть з'являтися інноваційні елементи, які у подальшому здатні забезпечити суттєве підвищення ефективності еволюційного пошуку. Втрата таких інноваційних рішень, зумовлена дією негативного відбору, була б невиправданою.

З метою подолання зазначеної проблеми у межах алгоритму NEAT було запроваджено концепцію видоутворення [3]. Її основна ідея полягає в обмеженні кола організмів, здатних до кросинговеру (схрещування), шляхом утворення спеціалізованих еволюційних ніш. За такого підходу організми, що належать до однієї ніші, конкурують між собою під час кросинговеру, тоді як конкуренція з усією популяцією виключається. Це дозволяє зменшити селекційний тиск на новостворені топології, забезпечуючи їм час для адаптації та виявлення потенційних переваг.

Реалізація видоутворення здійснюється шляхом кластеризації геномів: організми зі схожою топологічною будовою відносяться до одного виду. Таким чином, популяція розподіляється на підмножини, що відображають різноманітність топологічних рішень. Узагальнена схема роботи алгоритму видоутворення представлена на рис. 1.7.

Algorithm 1: Clustering Genomes into Species

Input: A *Population* of organisms and known *Species*

Result: Organisms will be clustered among *Species*. New *Species* will be created as appropriate.

```
foreach genome  $\in$  Population do
  foreach S  $\in$  Species do
    if genome.IsCompatible(S) then
      // Add compatible Genome to the current species
      S.AddGenome(genome);
    else if S is the last known species then
      // Create new species for a given genome
       $S_{new} \leftarrow \text{create\_new\_species}(\text{genome})$ ;
      // Add new species to the list of known Species
       $\text{Species} \leftarrow \text{Species} \cup S_{new}$ ;
```

Рис. 1.7. Узагальнена форма алгоритму видоутворення (за матеріалами [4])

Висновок. Алгоритм NEAT забезпечує формування складних штучних нейронних мереж, здатних розв'язувати широкий спектр завдань оптимізації керування, а також інші задачі навчання без учителя. Завдяки механізму еволюційного ускладнення топології та процесу видоутворення, побудовані мережі зазвичай характеризуються високою ефективністю як у процесі навчання, так і під час логічного виведення. Також встановлено, що його унікальність полягає у здатності одночасно еволюціонувати як ваги зв'язків, так і структуру мережі, що забезпечує природний механізм адаптивного зростання складності моделі та збереження інновацій у процесі селекції.

При цьому, сформована в результаті еволюції топологія розвивається виключно відповідно до специфіки розв'язуваної задачі, уникаючи появи надлишкових прихованих шарів, які зазвичай додаються під час традиційного проектування нейронних мереж, що ґрунтується на методі зворотного поширення помилки.

1.6. NEAT на основі гіперкубу (HyperNEAT)

Інтелект є результатом діяльності мозку, а сам мозок сформувався як продукт тривалої природної еволюції. Протягом мільйонів років він розвивався під дією жорстких умов довкілля та в умовах постійної конкуренції з іншими видами [2]. У підсумку виникла надзвичайно складна система, що складається з багатьох рівнів і модулів та включає трильйони нейронних зв'язків [61]. Саме ця структура людського мозку слугує орієнтиром у процесі створення систем штучного інтелекту. Водночас постає питання: яким чином можливо досягнути всю його складність за допомогою наявних, відносно обмежених, інструментів?

Нейробіологічні дослідження показали, що просторова організація мозку відіграє ключову роль у процесах сприйняття та пізнання — від обробки сенсорних сигналів до абстрактного мислення. Виявлено низку складних геометричних структур, зокрема клітини кори, які беруть участь в інерціальній навігації, та кортикальні стовпці, що взаємодіють із сітківкою ока для аналізу зорових образів [98]. Доведено, що завдяки своїй структурі мозок здатний ефективно розпізнавати та реагувати на сигнали, які надходять із сенсоріума, за допомогою спеціалізованих нейронних груп, що активуються у відповідь на певні входні патерни. Такий підхід забезпечує надзвичайно результативне подання та обробку різноманітних даних із навколишнього середовища.

Виявлено, що функціонування мозку ґрунтується на повторному використанні нейронних модулів, що суттєво знижує потребу у створенні нових структур. Це стає можливим завдяки складній ієрархічно-модульній організації та просторовій інтеграції різних ділянок мозку. Ці принципи лежать в основі концепції нейронного повторного використання (neural reuse), яка передбачає, що мозок повторно використовує існуючі нейронні схеми для виконання різних функцій, замість того, щоб еволюціонувати нові структури для кожного завдання. Ця теорія доповнює ідею масивної модульності, демонструючи, як одна й та сама ділянка мозку може бути переосмислена для виконання різних когнітивних функцій. Наприклад, дослідження показали, що мозок використовує попередньо існуючі нейронні ресурси замість розвитку нових структур,

підтримуючи кілька когнітивних можливостей через спільні схеми. Це свідчить про гнучку, інтегровану природу когнітивних функцій мозку [99].

Інакше кажучи, біологічний мозок реалізує багаторівневі ієрархічні процедури просторово-орієнтованої обробки даних. Ця властивість стала джерелом натхнення для дослідників у галузі нейроеволюції, які прагнуть впровадити аналогічні принципи у штучні нейронні мережі.

Під час проектування таких систем необхідно враховувати низку проблем, зокрема:

- велику кількість *вхідних ознак і параметрів*, що потребує створення масштабних нейронних архітектур;
- необхідність ефективного подання *природних геометричних закономірностей* та симетрій фізичного світу;
- забезпечення *локальності обробки*, коли просторово й семантично близькі дані обробляються групами взаємопов'язаних нейронних блоків, розташованих у компактних ділянках мережі.

Принципи повторного використання нейронних модулів та ієрархічно-модульна організація мозку, лягли в основу концепції HyperNEAT, яка розширює класичний алгоритм NEAT, дозволяючи ефективно кодувати складні топології нейронних мереж.

HyperNEAT частково базується на ідеї просторової організації обчислювальних структур та повторного використання модулів: замість того, щоб еволюціонувати кожен нейрон окремо, алгоритм формує компактне представлення зв'язків між нейронами на основі координат їх розташування, що дозволяє створювати мережі зі структурою, що повторно використовує функціональні модулі та відображає природні закономірності просторової організації [94, 95].

Таким чином, HyperNEAT поєднує еволюційний пошук оптимальної топології з принципами ефективної модульності та просторової інтеграції, аналогічними до тих, що спостерігаються у біологічному мозку, що робить його потужним інструментом для навчання складних нейронних мереж.

Алгоритм HyperNEAT є розвитком базового алгоритму NEAT і вводить новий тип кодування непрямого геному, відомий як CPPN (Compositional Pattern Producing Network) [95]. Це кодування дозволяє описувати патерни зв'язків нейронної мережі фенотипу як функцію його геометричного розташування.

У HyperNEAT патерн зв'язків мережі зберігається у вигляді чотиривимірного гіперкуба, де кожна точка відповідає конкретному зв'язку між двома нейронами — вихідним та цільовим. CPPN використовується для генерації цих патернів, обчислюючи чотиривимірну функцію:

$$w = CPPN(x_1, y_1, x_2, y_2) \quad (1.1)$$

де (x_1, y_1) — координати вихідного нейрона, а (x_2, y_2) — координати цільового. Результат функції визначає вагу зв'язку між цими нейронами. Якщо обчислена вага менша за мінімальний пороговий рівень w_{min} , зв'язок не враховується в геномі і не формується у фенотипі, тобто не експресується.

Завдяки такому підходу CPPN здатна генерувати довільну топологію мережі, яка може повторно використовувати набір генів для кодування патернів зв'язків у великомасштабних нейронних мережах. Патерн з'єднань, створений CPPN, зазвичай називають субстратом (substrate) і він відображає закономірності, виявлені в навчальних даних, що забезпечує ефективну організацію топології мережі. На рис. 1.8 подано графічну інтерпретацію патерну геометричної зв'язності, представленого у вигляді гіперкубу.

На відміну від традиційних архітектур штучних нейронних мереж, CPPN дозволяє використовувати різні функції активації у прихованих вузлах, що забезпечує відтворення складних геометричних закономірностей. Зокрема, для моделювання повторюваних структур застосовується тригонометрична функція синуса, а для підтримки локальної симетрії у певних областях мережі — гаусівська крива. Такий підхід дозволяє CPPN компактно кодувати патерни з різними просторовими властивостями, включаючи симетрію, повторюваність та закономірні структури, що значно розширює можливості побудови великих нейромереж з урахуванням природних геометричних закономірностей у даних.

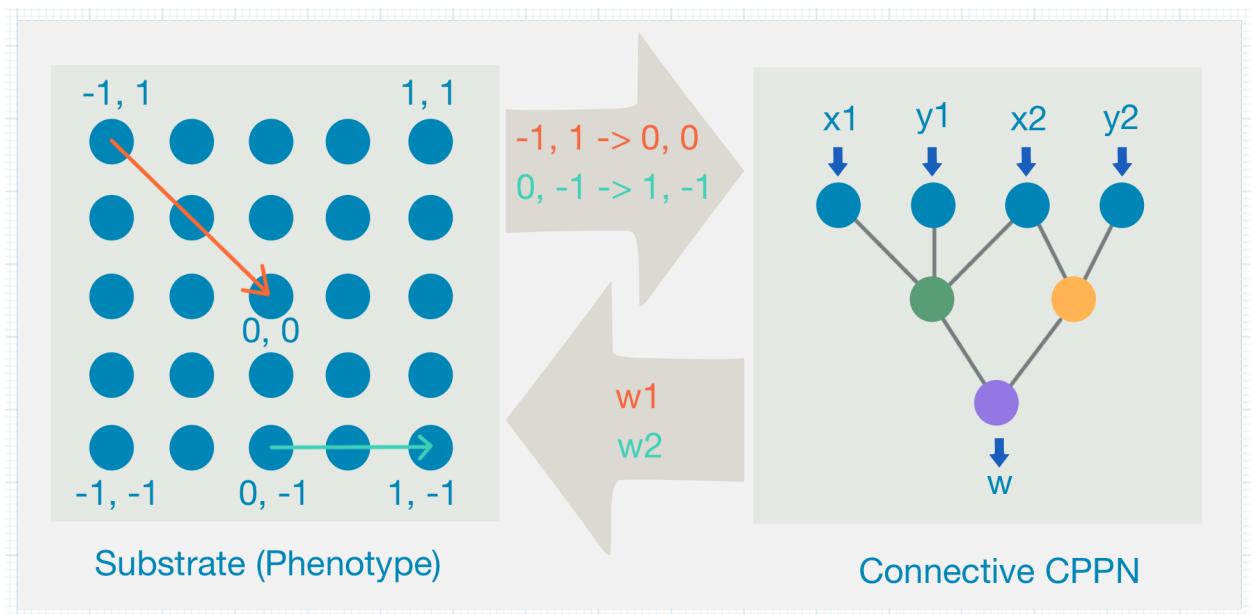


Рис. 1.8. Інтерпретація патерну геометричної зв'язності у вигляді гіперкубу (за матеріалами [4])

Конфігурація субстрату. Розташування вузлів мережі у субстраті, до якого підключається CPPN, може варіюватися залежно від особливостей розв'язуваної задачі. Вибір оптимальної компоновки здійснює дослідник для досягнення максимальної продуктивності алгоритму. Наприклад, для керування радіальною системою, такою як механізм із шістьма кінцівками, доцільно розміщувати вихідні вузли у круговій конфігурації, що дозволяє ефективно представляти патерн зв'язків у полярних координатах.

Найбільш поширені типи конфігурацій субстрату при використанні HyperNEAT наведені на рис. 1.9:

- **двовимірна сітка (2-D Grid):** вузли мережі розташовані у регулярній двовимірній декартовій сітці з центром у точці (0, 0);
- **тривимірна сітка (3-D Grid):** вузли мережі утворюють регулярну тривимірну декартову сітку з центром у точці (0, 0, 0);
- **сендвіч (Sandwich):** дві двовимірні плоскі сітки, що представляють входи та виходи, де один шар формує зв'язки до іншого;
- **кругова (Circular):** радіально впорядкована структура, яка зручна для відображення закономірностей у полярних координатах.

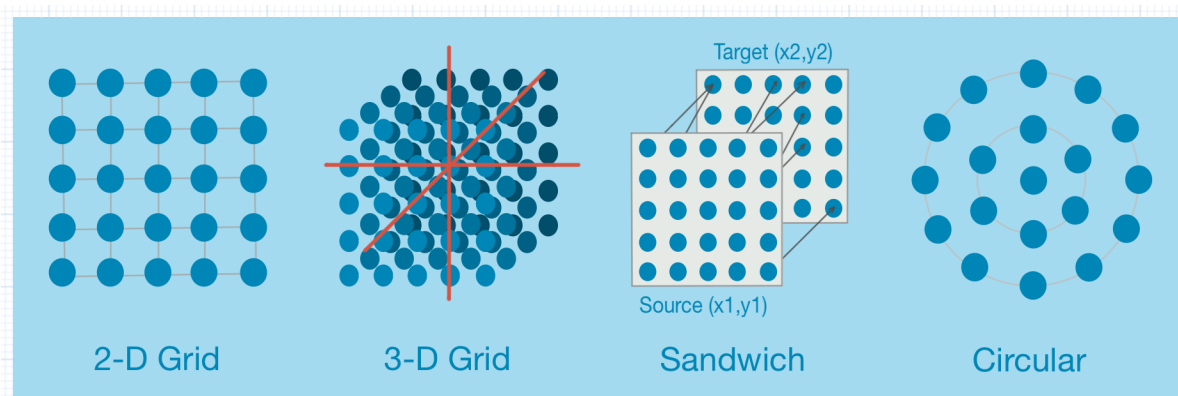


Рис. 1.9. Приклади конфігурації шарів субстрату (за матеріалами [4])

Метод отримав назву HyperNEAT, оскільки він базується на модифікованому алгоритмі NEAT, який використовується для еволюції CPPN, здатних відображати об'ємні структури у гіперпросторі. Кожна експресована (доступна для генетичних операцій) точка патерну в гіперкубі (hypercube) відповідає зв'язку між двома вузлами графа нижчого порядку — субстрату. Внаслідок цього розмірність гіперпростору завжди вдвічі перевищує розмірність нижчого графа найменшого порядку.

Узагальнену схему алгоритму HyperNEAT ілюструє рис. 1.10.

Algorithm 2: The general form of the HyperNEAT algorithm

```

begin
  1. Choose desired Substrate configuration (nodes layout and input/output assignments);
  2. Initialize Population of minimal CPPNs with random weights;
  repeat
    foreach organism ∈ Population do
      3. Query the CPPN of the organism for the weight of each possible connection
         in the Substrate representing its phenotype. If the absolute value of the output
         exceeds a threshold magnitude, create the connection with a weight scaled
         proportionally to the output value.;
      4. Run the Substrate as a phenotype ANN in the task domain to evaluate the
         fitness of found solutions.;
      5. Reproduce CPPNs of the organisms in the Population with NEAT;
  until solution is found;
```

Рис. 1.10. Узагальнена схема алгоритму HyperNEAT (за матеріалами [4])

Будь-який новий **ген зв'язку** або **ген вузла**, доданий до CPPN під час її еволюції, відкриває нові глобальні можливості варіацій патернів зв'язків у

субстраті фенотипу, створюючи нові ознаки. Кожна зміна геному CPPN визначає новий спосіб модифікації всього патерну зв'язків. Крім того, раніше еволюційно створені CPPN можна використовувати як основу для генерації патернів зв'язків на субстратах із вищим дозволом (higher resolution), ніж той, що застосовувався під час навчання. Це дає змогу переносити робоче рішення на нові рівні деталізації без необхідності обмеження верхньої межі.

Висновки. Завдяки використанню непрямого генетичного кодування та можливості формувати складні просторові закономірності зв'язків, алгоритм HyperNEAT є потужним інструментом побудови великомасштабних нейронних мереж, здатних відтворювати властиві біологічним системам регулярність, симетрію та локальність. Однак залежність від заздалегідь фіксованого субстрату, який задається дослідником, обмежує гнучкість цього підходу й може знижувати ефективність еволюції у задачах зі складною або невідомою геометричною структурою. Це підкреслює необхідність методів, що дозволяють автоматично визначати розташування вузлів та адаптувати субстрат у процесі еволюції. Потенційні механізми подолання цих обмежень — зокрема адаптивні методи побудови субстрату, такі як ES-HyperNEAT — будуть детально розглянуті в наступному підрозділі.

1.7. HyperNEAT із субстратом, що розвивається (ES-HyperNEAT)

Метод HyperNEAT базується на концепції, що геометричні структури біологічного мозку можуть бути відтворені штучними нейронними мережами з вузлами, розташованими у певних просторових координатах. Завдяки цьому нейроеволюція отримує значні переваги і дозволяє навчати великомасштабні нейромережі для вирішення складних задач, що було складно або неможливо реалізувати за допомогою стандартного алгоритму NEAT.

Проте, хоча підхід HyperNEAT і надихається структурою природного мозку, він все ще не досягає пластичності, притаманної природній еволюції. Метод обмежує свободу еволюційного процесу: експериментатор повинен заздалегідь визначити розташування вузлів мережі, і будь-яке неточне припущення знижує ефективність навчання.

Розташування вузлів на фіксованому субстраті накладає додаткові обмеження на патерн ваг зв'язків, що створюється CPPN. Це обмеження ускладнює процес кодування природних геометричних закономірностей у топологію нейромережі (фенотипу), оскільки можливі зв'язки існують лише між вузлами, розташованими відповідно до субстрату. Внаслідок цього можуть виникати зайві помилки апроксимації, що погіршують якість отриманих рішень. Дозволивши CPPN формувати патерни зв'язків для вузлів у альтернативних просторових позиціях, можна досягти більш гнучкого й ефективного кодування геометричних закономірностей.

Щільність інформації у гіперкубі. Постає питання доцільності накладання жорстких обмежень на початкове розташування вузлів субстрату. Більш ефективним підходом може бути використання неявних підказок, отриманих із шаблону зв'язків, для визначення оптимальних позицій розміщення нових вузлів, що дозволяє точніше відтворювати природні закономірності фізичного світу.

Аналіз топології мережі показує, що області з однаковими вагами зв'язків несуть обмежену кількість інформації та мають невелике функціональне значення. Натомість ділянки з великими градієнтами ваг є інформаційно

насиченими та потребують додаткових вузлів для точного кодування складних природних процесів. Кожен зв'язок між вузлами субстрату може бути представленим як точка у чотиривимірному гіперкубі.

Основна ідея запропонованого алгоритму ES-HyperNEAT [97] полягає у динамічному розміщенні вузлів: у регіонах гіперкуба з високими варіаціями ваг зв'язків створюється більша кількість гіперточок, а у зонах з низькими варіаціями — відповідно менша. Розташування вузлів та формування зв'язків визначається змінами ваг, які генерує еволюціонуюча CPPN для даної ділянки субстрату. Тобто для прийняття рішення про розміщення нового вузла додаткової інформації не потрібно, оскільки вона вже закодована у патернах зв'язків, що створюються CPPN.

Отже, ключовим принципом формування топографії субстрату стає **щільність інформації** [97], а позиції вузлів у фенотипі нейромережі відображають інформаційний зміст, закодований у патернах зв'язків CPPN, що забезпечує більш ефективне відтворення закономірностей природних процесів.

Квадродерево як потужний інструмент виділення інформації.

Для відображення гіперточок, що кодують ваги зв'язків усередині гіперкуба, в алгоритмі ES-HyperNEAT використовується квадродерево.

Квадродерево являє собою ієрархічну структуру даних, у якій кожен внутрішній вузол має рівно чотири дочірні елементи. Такий тип структури обрано завдяки його здатності ефективно представляти двовимірні області з різними рівнями деталізації.

Застосування квадродерева забезпечує оптимальну організацію пошуку та обробки даних у двовимірному просторі: будь-яка досліджувана область може бути рекурсивно поділена на чотири підобласті, де кожна з них стає листом дерева, а кореневий вузол відповідає за початкову (декомпозовану) область простору.

Застосовуючи метод виділення інформації на основі квадродерева, алгоритм ES-HyperNEAT ітеративно виявляє нові зв'язки між вузлами в двовимірному просторі субстрату нейронної мережі, починаючи з вузлів

введення та виведення, заздалегідь визначених експериментатором. Такий підхід забезпечує значно вищу обчислювальну ефективність порівняно з безпосереднім пошуком у чотиривимірному просторі гіперкуба.

На рис. 1.11 представлено приклад процесу вилучення інформації за допомогою квадродревної структури.

Алгоритм пошуку, що базується на структурі квадродревка, виконується у два основні етапи:

а.) Поділ і ініціалізація. На початковому етапі формується квадродревно шляхом рекурсивного поділу вихідного простору субстрату, який охоплює область від $(-1, -1)$ до $(1, 1)$. Процес поділу триває доти, доки не буде досягнуто заданої глибини дерева, що фактично визначає кількість підпросторів у початковій області (тобто роздільну здатність ініціалізації).

Після цього для кожного вузла дерева з координатами центру (a, b) виконується запит до CPPN із аргументами (a, b, x_i, y_i) з метою обчислення ваги зв'язків. Коли значення ваг для кінцевих вузлів певного вузла p квадродревка отримано, дисперсія цього вузла може бути розрахована за наступною формулою:

$$\sigma^2 = \sum_{i=1}^k (\bar{\omega} - \omega_i)^2 \quad (1.2)$$

У цьому виразі $\bar{\omega}$ позначає середнє значення ваги зв'язків між k кінцевими вузлами, тоді як ω_i — це вага зв'язку з конкретним кінцевим вузлом. Обчислене значення дисперсії виступає евристичним показником наявності інформації в певній області субстрату. Якщо це значення перевищує заданий пороговий рівень, який визначає бажану щільність інформації, для відповідного квадрата виконується повторний поділ. Таким чином, алгоритм динамічно адаптує рівень деталізації, забезпечуючи необхідну щільність інформаційного представлення.

У верхній частині рис. 1.11 наведено приклад процесу поділу та ініціалізації простору за допомогою квадродревної структури.

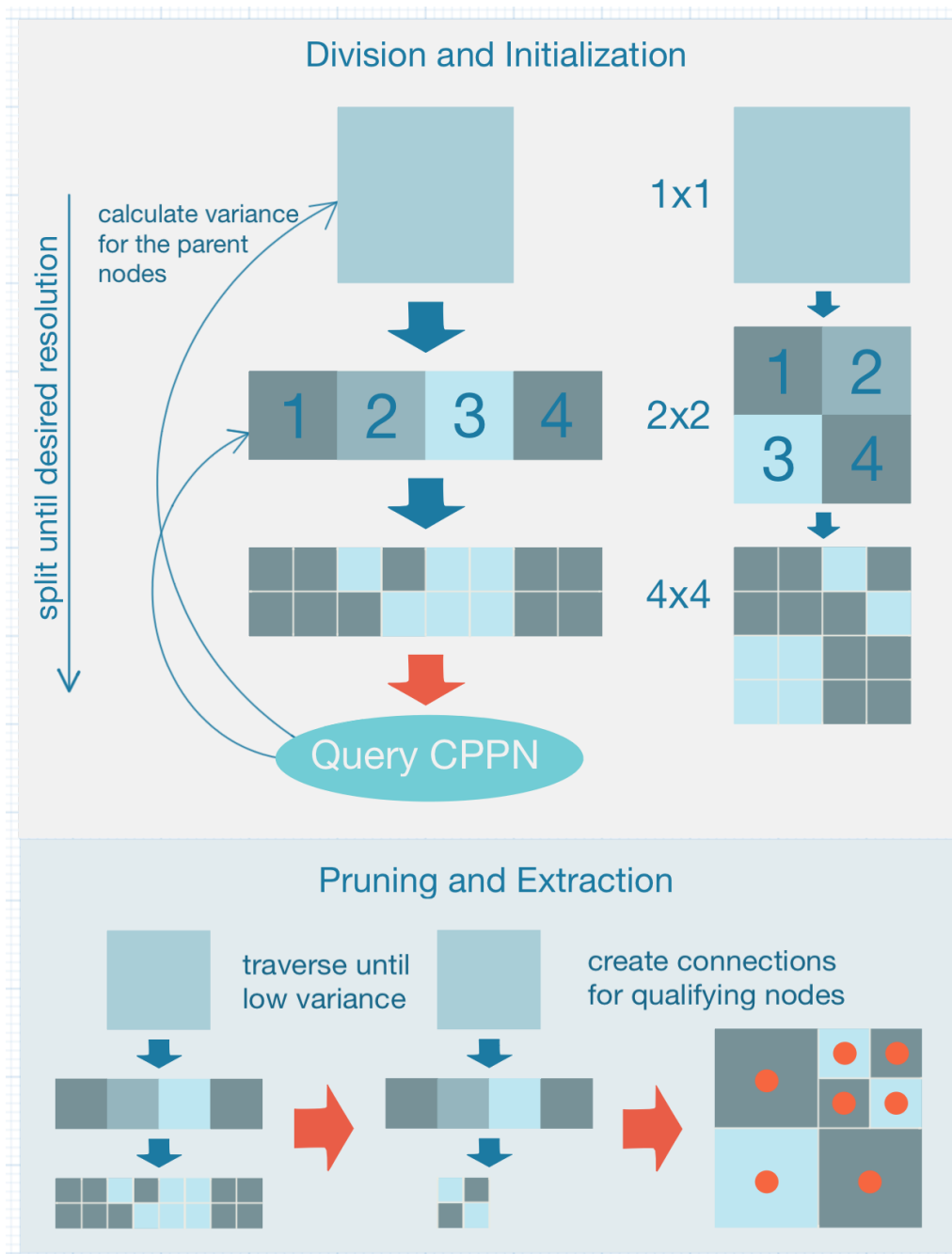


Рис. 1.11. Приклад виділення інформації з використанням квадродерева (за матеріалами [4])

б.) Обрізання та вилучення. Щоб забезпечити експресію більшої кількості зв'язків (а також вузлів субстрату) у ділянках із високою щільністю інформації (тобто з великою дисперсією ваг), виконується процедура обрізання та вилучення над квадродеревом, сформованим на попередньому етапі.

Квадродерево обходиться у глибину доти, доки дисперсія поточного вузла не стане меншою за встановлений поріг дисперсії σ_t^2 або поки вузол не виявиться без дочірніх елементів (нульова дисперсія). Для кожного вузла, який відповідає цим умовам, створюється експресований зв'язок між його центром (x, y) і кожним батьківським вузлом, який уже визначено — або експериментатором, або знайдено під час попереднього виконання цих двох етапів (тобто серед прихованих вузлів, що вже були створені методом ES-HyperNEAT).

Для наочного уявлення роботи етапу обрізання та вилучення зверніться до нижньої частини рис. 1.11.

Алгоритм ES-HyperNEAT починає роботу з вузлів введення (вхідних), визначених користувачем, і докладно досліджує можливі зв'язки, що виходять від них, спрямовуючи їх до нових експресованих прихованих вузлів. Експресія шаблонів вихідних зв'язків і розміщення прихованих вузлів у просторі субстрату здійснюється за допомогою методу вилучення інформації на основі квадродерева, описаного раніше. Процес вилучення інформації виконується ітеративно доти, доки не буде досягнуто бажаної щільності експресії інформації або поки в гіперкубі не залишиться нової інформації для виявлення. Після цього сформована мережа з'єднується з вихідними вузлами, визначеними користувачем, шляхом експресії шаблонів вхідних зв'язків до виходів — для цього також використовується метод вилучення інформації на основі квадродерева. У фінальній мережі зберігаються лише ті приховані вузли, які мають шляхи як до вхідних, так і до вихідних вузлів. На цьому етапі в субстраті фенотипової штучної нейронної мережі вже визначено велику кількість вузлів і зв'язків. Для оптимізації структури мережі може бути корисним видалити частину вузлів шляхом введення додаткової стадії **смугового обрізання** (*band pruning*). Під час цієї процедури зберігаються лише ті точки, що належать до певної смуги, тоді як точки, розташовані на її межах, видаляються. Регулюючи ширину цієї смуги — розширюючи або звужуючи її — **CPPN** може контролювати щільність закодованої інформації в межах субстрату. Узагальнена форма алгоритму ES-HyperNEAT представлена на рис. 1.12.

Алгоритм ES-HyperNEAT поєднує всі переваги методів NEAT і HyperNEAT, водночас запроваджуючи низку потужних нових можливостей, зокрема:

- **Автоматичне розміщення прихованих вузлів** у субстраті таким чином, щоб точно відповідати патернам зв'язності, які задаються еволюціонованими CPPN.
- **Підвищену модульність фенотипових ШНМ**, оскільки архітектура ES-HyperNEAT дозволяє природним чином розпочинати еволюційний пошук із зміщенням у бік локальності — завдяки специфічній побудові початкових CPPN.
- **Можливість поступового розширення структури фенотипової ШНМ** у процесі еволюції шляхом збільшення кількості вузлів і зв'язків у субстраті. Це відрізняє ES-HyperNEAT від HyperNEAT, де кількість вузлів субстрату визначається наперед.

Висновок. Таким чином, алгоритм ES-HyperNEAT надає можливість використовувати базову архітектуру HyperNEAT без необхідності внесення змін до генетичної структури частини NEAT, водночас усуваючи ключове обмеження, притаманне класичному HyperNEAT, пов'язане з потребою ручного визначення конфігурації субстрату. Завдяки механізмам автоматичного аналізу «щільності інформації» в гіперкубі та ієрархічної адаптивної декомпозиції простору за допомогою квадродерева, ES-HyperNEAT здатний самостійно визначати, у яких регіонах субстрату слід розміщувати додаткові вузли й підключення. Це дає змогу формувати топологію фенотипної нейронної мережі, що точніше відповідає закономірностям, які кодує CPPN, і водночас мінімізує надмірність структури.

Запропонований підхід забезпечує автоматичне масштабування та уточнення топології мережі в тих зонах субстрату, де CPPN демонструє високі градієнти або складну просторову структуру зв'язків. У результаті ES-HyperNEAT створює топології, здатні відображати складні геометричні

закономірності без необхідності попереднього знання про розподіл важливих областей вхідного простору. Це робить алгоритм особливо ефективним для задач, де структура фенотипної нейромережі повинна віддзеркалювати складні просторові, сенсорні або моторні залежності, а ручне визначення субстрату є

Algorithm 3: ES-HyperNEAT Algorithm

Parameters: initialDepth, maxDepth, varianceThreshold, bandThreshold, iterationLevel, divisionThreshold

Input : CPPN, InputPositions, OutputPositions

Output : Connections, HiddenNodes

begin

```

// Input-to-hidden node connections
foreach input ∈ InputPositions do
    // Analyze outgoing connectivity pattern from input
    root ← DivisionAndInitialization(input.x, input.y, true);
    // Traverse quadtree and store qualified connections
    PruningAndExtraction(input.x, input.y, inputConnections, root, true);
    foreach c ∈ inputConnections do
        node ← Node(c.x2, c.y2);
        if node ∉ HiddenNodes then
            HiddenNodes ← HiddenNodes ∪ node;

// Hidden-to-hidden node connections
UnexploredHiddenNodes ← HiddenNodes;
for i ← 1 to iterationLevel do
    foreach hidden ∈ UnexploredHiddenNodes do
        root ← DivisionAndInitialization(hidden.x, hidden.y, true);
        PruningAndExtraction(hidden.x, hidden.y, hidnConnections, root, true);
        foreach c ∈ hidnConnections do
            node ← Node(c.x2, c.y2);
            if node ∉ HiddenNodes then
                HiddenNodes ← HiddenNodes ∪ node;

        // Remove explored nodes
        UnexploredHiddenNodes ← HiddenNodes − UnexploredHiddenNodes;

// Hidden-to-output node connections
foreach ouput ∈ OutputPositions do
    // Analyze incoming connectivity to the output
    root ← DivisionAndInitialization(output.x, output.y, false);
    PruningAndExtraction(output.x, output.y, outputConnections, root, false);
    /* No new nodes created here because all hidden nodes that connected
       to input/output already expressed */
connections ← inputConnections ∪ hidnConnections ∪ outputConnections;
Remove nodes and their connections that do not have a path to input and output nodes.

```

нетривіальним або взагалі неможливим.

Рис. 1.12. Узагальнена форма алгоритму ES-HyperNEAT (за матеріалами [4])

Запропонований підхід забезпечує автоматичне масштабування та уточнення топології мережі в тих зонах субстрату, де CPPN демонструє високі градієнти або складну просторову структуру зв'язків. У результаті ES-HyperNEAT створює топології, здатні відображати складні геометричні закономірності без необхідності попереднього знання про розподіл важливих областей вхідного простору. Це робить алгоритм особливо ефективним для задач, де структура фенотипної нейромережі повинна віддзеркалювати складні просторові, сенсорні або моторні залежності, а ручне визначення субстрату є нетривіальним або взагалі неможливим.

Отже, ES-HyperNEAT суттєво підвищує гнучкість, адаптивність та масштабованість еволюції великих штучних нейронних мереж, усуваючи головне вузьке місце HyperNEAT — необхідність статичного задання структури субстрату. Завдяки цьому ES-HyperNEAT відкриває можливість ефективного застосування непрямих генетичних кодувань у широкому спектрі складних задач, включаючи моделювання біологічних систем, розроблення роботизованих контролерів та обробку просторово структурованих даних.

1.8. Постановка задач дослідження

Аналіз класичного алгоритму NEAT та його існуючих імплементацій показав наявність фундаментального протиріччя, що обмежує практичну масштабованість методу. На кожному поколінні еволюції необхідно виконувати оцінку придатності для всіх організмів популяції, яка включає симуляцію поведінки їхніх штучних нейронних мереж у середовищі задачі. При цьому, час на виконання цього етапу, зростає лінійно або надлінійно, за умови збільшення наступних чинників:

- розміру популяції,
- складності моделі середовища,
- тривалості симуляції,
- розміру або глибини розвинутих ШНМ.

Це призводить до різкого збільшення загального часу еволюційного процесу.

У однопотоківих реалізаціях NEAT, які є домінуючими у сучасних програмних бібліотеках, час оцінки придатності стає головним обчислювальним вузьким місцем, що:

- знижує можливість масштабування на більш складні задачі,
- обмежує розмір популяції, який може бути ефективно оброблений,
- подовжує еволюційний цикл та ускладнює проведення експериментальних досліджень.

Водночас огляд сучасних джерел свідчить, що не існує загальноприйнятих, стандартизованих або достатньо ефективних методів розпаралелювання саме етапу оцінки придатності в алгоритмі NEAT. Існуючі спроби масштабування або:

- орієнтовані виключно на НРС-середовища з високими накладними витратами (MPI),
- не забезпечують динамічного балансування навантаження,
- не адаптовані до еволюційної специфіки алгоритму NEAT (інноваційні номери, структура геному),
- або відсутні у загальнодоступних реалізаціях.

Таким чином, виникає наукове протиріччя:

Час на виконання оцінки придатності в однопотокових реалізаціях NEAT невпинно зростає із ускладненням задачі, тоді як ефективні, узагальнені та доступні методи паралелізації цього етапу еволюції відсутні.

Це протиріччя створює наукову проблему, вирішення якої є необхідним для підвищення продуктивності та масштабованості еволюційних алгоритмів побудови топології нейронних мереж.

Наукова проблема, що потребує вирішення. З огляду на сформульоване протиріччя, наукова проблема полягає в тому, що:

- оцінка придатності в алгоритмі NEAT має природну можливість бути розпаралеленою на рівні особин,
- але для практичного застосування потрібна спеціалізована модель паралельних обчислень,
- яка узгоджується з архітектурою NEAT, дотримується властивостей еволюції (видоутворення, інноваційні номери, топологічне зростання),
- і одночасно забезпечує ефективність, стійкість та мінімальні накладні витрати.

Недостатність існуючих методів не дозволяє використовувати масштабування в повному обсязі, особливо у задачах, що вимагають великої кількості поколінь або великої популяції.

Постановка наукової задачі дослідження. Виходячи з виявленого протиріччя, науковою задачею даного дослідження є:

Розробити теоретично обґрунтований та практично ефективний метод організації розподілених обчислень для алгоритму NEAT, який дозволить мінімізувати часові витрати на етапі оцінки придатності організмів у популяції за рахунок оптимального використання паралелізму. Це дозволить підвищити продуктивність алгоритму NEAT та одночасно оптимізувати витрати обчислювальних ресурсів.

Для досягнення цієї мети необхідно вирішити **наступні часткові завдання:**

1. Провести формальний аналіз обчислювальної складності етапу оцінки придатності у NEAT.
2. Визначити необхідні властивості моделі розпаралелювання, які забезпечують коректність еволюційного процесу.
3. Обґрунтувати вибір архітектури паралельних обчислень, придатної для динамічного розподілу навантаження.
4. Розробити реалізацію моделі Master–Worker з використанням технологій Go (goroutines + channels) як базового паралельного механізму.
5. Розробити комплексну метрику оцінки ефективності реалізації NEAT у паралельному режимі.
6. Розробити СПЗ для візуалізації отриманих результатів.
7. Розробити метод оптимізації алгоритму NEAT для підвищення швидкості побудови топології ШНМ та покращення ефективності знайдених рішень в умовах обмежених обчислювальних ресурсів
8. Експериментально підтвердити ефективність запропонованого методу на тестових задачах еволюційного конструювання ШНМ.

1.9. Висновки до першого розділу

У першому розділі дисертаційної роботи виконано комплексний теоретико-аналітичний огляд сучасного стану досліджень у галузі нейроеволюційних алгоритмів, із особливим акцентом на методи еволюційної побудови та оптимізації топології штучних нейронних мереж. Проведений аналіз дозволив систематизувати існуючі підходи, виявити їхні переваги та обмеження, а також сформулювати наукові передумови для постановки задачі подальшого дослідження.

Показано, що нейроеволюційні алгоритми становлять альтернативний клас методів навчання штучних нейронних мереж, який принципово відрізняється від градієнтних підходів глибокого навчання. На відміну від алгоритмів, заснованих на зворотному поширенні похибки, нейроеволюція дозволяє одночасно оптимізувати вагові коефіцієнти, топологію мережі, правила навчання та інші структурні характеристики моделі. Це забезпечує більшу гнучкість у дослідженні простору рішень, особливо в умовах відсутності явної функції втрат або за наявності складних, стохастичних і нелінійних середовищ.

У розділі обґрунтовано ключову роль різноманітності та механізмів її підтримки в еволюційних алгоритмах. Проаналізовано класичні підходи до збереження генетичної різноманітності, а також сучасні концепції, орієнтовані на різноманітність поведінки, зокрема алгоритми пошуку новизни та якісного різноманіття. Показано, що такі методи істотно розширюють можливості нейроеволюції, дозволяючи ефективно уникати передчасної конвергенції та відкривати нові області простору рішень, які є недосяжними для суто цілюорієнтованих стратегій оптимізації.

Значну увагу приділено аналізу алгоритму NEAT як одного з найбільш впливових та концептуально завершених методів нейроеволюції. Детально розглянуто його ключові механізми, зокрема інноваційні номери, видоутворення та принцип поступового ускладнення топології. Показано, що саме ці механізми забезпечують ефективне зростання складності нейронних мереж у процесі

еволюції та збереження перспективних структурних інновацій. Водночас встановлено, що класичний NEAT, як алгоритм із прямим генетичним кодуванням, має обмежену масштабованість і практично не пристосований до побудови надвеликих нейронних мереж, характерних для сучасних задач глибокого навчання.

У розділі також проаналізовано розвиток ідей NEAT у вигляді алгоритмів HyperNEAT та ES-HyperNEAT, які використовують непряме генетичне кодування для формування великомасштабних нейронних архітектур із урахуванням просторових закономірностей. Показано, що застосування CPPN та принципів просторової організації дозволяє суттєво зменшити складність геному та ефективно кодувати регулярні, симетричні та модульні структури. Разом із тим виявлено, що залежність HyperNEAT від наперед заданого субстрату обмежує його адаптивність, тоді як ES-HyperNEAT частково усуває це обмеження за рахунок автоматичного визначення розміщення вузлів на основі щільності інформації.

На основі проведеного огляду встановлено, що, попри значні теоретичні та практичні досягнення нейроеволюційних методів, їхнє широке застосування обмежується високою обчислювальною складністю, зокрема на етапі оцінки придатності організмів у популяції. Показано, що зі зростанням розміру популяції, складності середовища, тривалості симуляцій та глибини нейронних мереж саме цей етап стає домінуючим за часовими витратами та визначає загальну ефективність алгоритму.

Виявлене фундаментальне протиріччя між природною можливістю паралелізації оцінки придатності в алгоритмі NEAT і відсутністю універсальних, ефективних та доступних моделей її практичної реалізації в існуючих програмних імплементаціях. Це протиріччя формує актуальну наукову проблему, розв'язання якої є необхідним для підвищення масштабованості, продуктивності та прикладної цінності нейроеволюційних алгоритмів.

Таким чином, результати першого розділу створюють теоретичну та методологічну основу для подальших досліджень, спрямованих на розробку

спеціалізованих моделей паралельних і розподілених обчислень для алгоритму NEAT. Обґрунтовано доцільність зосередження подальшої роботи на оптимізації етапу оцінки придатності шляхом використання розгалужених обчислень, що дозволить суттєво зменшити часові витрати еволюційного процесу та розширити клас задач, доступних для нейроеволюційних методів.

РОЗДІЛ 2. РОЗРОБКА МЕТОДУ ОПТИМІЗАЦІЇ АЛГОРИТМУ NEAT ДЛЯ ПІДВИЩЕННЯ ШВИДКОСТІ ПОБУДОВИ ТОПОЛОГІЇ ШНМ

2.1. Формальна постановка задачі дослідження

У межах даного дослідження розглядається задача підвищення продуктивності побудови оптимальної топології штучної нейронної мережі при використанні нейроеволюційного алгоритму NEAT за рахунок оптимізації етапу оцінювання придатності організмів у популяції. Зазначений етап є обчислювально найбільш затратною складовою еволюційного процесу, оскільки передбачає багаторазове виконання складних симуляційних моделей для кожного організму в кожному поколінні.

Вхідні дані задачі

Вхідними даними задачі є сукупність параметрів нейроеволюційного процесу та параметрів симуляційного середовища, зокрема:

- початкова популяція організмів, кожен з яких кодує топологію та вагові коефіцієнти штучної нейронної мережі;
- параметри алгоритму NEAT (розмір популяції, імовірності мутацій, правила схрещування, параметри видоутворення тощо);
- опис симуляційного середовища та його початковий стан;
- критерії припинення еволюційного процесу (максимальна кількість поколінь, досягнення заданого рівня придатності);
- обчислювальні ресурси, доступні для виконання експерименту (кількість процесорних ядер, рівень паралелізму, характеристики комунікаційного середовища).

Цільова функція

Цільовою функцією дослідження є мінімізація сукупних часових витрат на побудову оптимальної топології штучної нейронної мережі при збереженні або покращенні якості знайдених рішень. Формально це означає оптимізацію функції часу виконання еволюційного процесу за умови досягнення заданого

рівня функціональної придатності контролюючої ШНМ з топологією знайденою в процесі еволюції.

Таким чином, у межах дослідження оптимізується не лише функція придатності окремих організмів, а й ефективність реалізації самого алгоритму NEAT як обчислювального процесу. Це зумовлює необхідність використання комплексних моделей оцінки, які поєднують якісні та часові показники.

Обмеження системи

Задача оптимізації формулюється з урахуванням низки обмежень, до яких належать:

- обмежені обчислювальні ресурси, зокрема кількість доступних процесорних ядер та обсяг оперативної пам'яті;
- наявність накладних витрат на міжпроцесну комунікацію та синхронізацію при паралельному виконанні обчислень;
- стохастичний характер нейроеволюційного процесу, що ускладнює точне прогнозування часу досягнення оптимального рішення;
- необхідність збереження коректності та відтворюваності результатів еволюції незалежно від рівня паралелізму.

Формулювання наукової задачі

З урахуванням наведених вхідних даних, цільової функції та обмежень, наукова задача даного дослідження полягає у розробці методу оптимізації алгоритму NEAT за рахунок організації паралельної оцінки придатності організмів у нейроеволюційному алгоритмі, який забезпечує зменшення часових витрат еволюційного процесу без погіршення якості знайдених нейронних мереж та з урахуванням обмежених обчислювальних ресурсів.

2.2. Формальний аналіз обчислювальної складності етапу оцінки придатності у алгоритмі NEAT

Алгоритм NEAT (NeuroEvolution of Augmenting Topologies) є одним із найефективніших методів еволюційного проектування штучних нейронних мереж, який поєднує оптимізацію вагових коефіцієнтів та структури мережі. Незважаючи на високу якість отримуваних результатів, класичний NEAT характеризується значними обчислювальними витратами, що суттєво зростають із розміром популяції, складністю фенотипів та розмірністю середовища навчання.

Основна **проблема обчислювальної складності** полягає у високій вартості процедури **оцінки придатності (fitness evaluation)** кожної особини в популяції. Для кожного кандидата-організму необхідно:

- побудувати топологію нейронної мережі згідно з геномом;
- виконати симуляцію її роботи на навчальному середовищі або задачі;
- обчислити метрику якості (помилку, винагороду, точність тощо).

Якщо кількість особин у популяції позначити як N , а середній час евалюації однієї особини як T_{eval} , то загальний час еволюційного кроку можна оцінити як:

$$T_{generation} = NT_{eval} + T_{reproduction} + T_{speciation}, \quad (2.1)$$

де $T_{reproduction}$ — час, витрачений на мутацію та кросовер, $T_{speciation}$ — час, необхідний для кластеризації особин за видовими ознаками (speciation).

На практиці T_{eval} домінує в загальних обчисленнях, особливо у випадках, коли оцінювання включає складні симуляційні середовища, такі як фізичні двигуни, моделі керування роботами або динамічні середовища для навчання з підкріпленням (Reinforcement Learning).

Крім того, із зростанням складності мереж топологічна експресія геному та розрахунок активаційних сигналів стають дедалі дорожчими. Внаслідок цього обчислювальна складність класичного NEAT зростає щонайменше лінійно від

кількості особин і кількості поколінь, а в реальних умовах — суперлінійно через нелінійний вплив топологічного розширення.

Таким чином, вузьким місцем алгоритму NEAT є етап оцінки придатності особин, що становить до 90 % від загального часу виконання еволюційного циклу. Це обмежує масштабованість методу, робить неможливим використання великих популяцій або складних середовищ симуляції на одному обчислювальному вузлі та істотно сповільнює процес еволюції.

Зазначені недоліки створюють передумови для розроблення розподілених методів оцінки придатності, що дозволяють здійснювати паралельну симуляцію особин у популяції, підвищуючи ефективність використання обчислювальних ресурсів і прискорюючи конвергенцію еволюційного процесу.

Далі ми зробимо формальний аналіз часової складності алгоритму NEAT, який дозволяє виявити ключові обмеження алгоритму та окреслити напрями його оптимізації за рахунок розподілених обчислень.

Для цього буде побудована **математична модель обчислювальних витрат алгоритму NEAT**, а також показано їх зменшення за рахунок використання розподілених обчислень.

Параметри моделі:

- N — кількість особин у популяції;
- G — кількість поколінь;
- $\bar{V}$ — середня кількість вузлів (нейронів) у фенотипі однієї особини;
- $\bar{W}$ — середня кількість зв'язків (синапсів);
- L — кількість кроків симуляції або обсяг даних, що використовується під час оцінки однієї особини;
- p — кількість обчислювальних вузлів (робітників) у розподіленій системі;
- C_{comm} — середні комунікаційні накладні витрати на передачу даних на одну особину при розподілених обчисленнях (включає латентність + переданий об'єм);
- d — вартість обчислення відстані між двома геномами під час процедури кластеризації (видоутворення).

Припущення: для feedforward-мереж вартість прогона через мережу за один такт пропорційна $\Theta(\bar{V})$. Якщо використовуються рекурентні мережі або складна симуляція, фактори множаться відповідно.

Обчислювальні етапи NEAT. Одна ітерація (покоління) алгоритму NEAT складається з таких основних етапів:

1. **Оцінка придатності (fitness evaluation) однієї особини** — етап, що домінує за обчислювальними витратами. Для однієї особини вартість обчислення придатності можна подати як:

$$T_{\text{eval}} = \Theta(L\bar{V}), \quad (2.2)$$

де L — кількість ітерацій симуляції, а $\bar{V}$ — середня кількість вузлів (нейронів) у фенотипі однієї особини, які проходяться під час прямого поширення сигналу.

2. **Видоутворення (кластеризація популяції)** — процес групування геномів за ступенем подібності. У найпростішій реалізації вимагає порівняння кожної особини з усіма іншими:

$$T_{\text{spec}} = O(N^2 d), \quad (2.3)$$

де d — складність обчислення метрики відстані між геномами. Використання прискорених структур (наприклад, дерев k-d або геш-функцій чутливих до локальності) дозволяє знизити складність до $O(N \log N)$ або $O(N)$.

3. **Відтворення (reproduction)** — створення нових геномів шляхом кросовера і мутацій:

$$T_{\text{repr}} = O(N(\bar{W} + \bar{V})). \quad (2.4)$$

4. **Допоміжні процедури** — сортування, нормалізація та оновлення структур інновацій. Вони мають порядок складності $O(N \log N)$ або нижчий $O(N)$.

Підсумкова асимптотика для одного покоління (наївна оцінка):

$$T_{\text{gen}} = \Theta(NL\bar{V}) + O(N^2 d) + O(N(\bar{W} + \bar{V})) + O(N \log N).. \quad (2.5)$$

Відповідно, загальна складність для всієї еволюції протягом G поколінь становить:

$$T_{\text{total}} = GT_{\text{gen}}. \quad (2.6)$$

У типових реалістичних сценаріях домінує перший член $\Theta(NL\bar{V})$, що відповідає *процедурі оцінки придатності*, а видоутворення може стати помітним лише при дуже великому N або неефективній метриці для обчислення відстані між геномами. Таким чином, часову складність алгоритму NEAT можна апроксимувати як:

$$T_{\text{total}} = O(GNL\bar{V}). \quad (2.7)$$

Зростання складності під час еволюції. Оскільки NEAT поступово збільшує топологічну складність мереж, середня кількість зв'язків $\bar{W}$ і вузлів $\bar{V}$ зростає з поколіннями. Це призводить до *суперлінійного* збільшення обчислювальних витрат, особливо у задачах, де L є великим (наприклад, при симуляції динамічних середовищ).

Модель розподіленої реалізації. У розподіленій системі оцінка придатності може бути паралелізована. Для p обчислювальних вузлів час одного покоління зменшується до:

$$T_{\text{gen}}^{(p)} \approx \frac{NT_{\text{eval}}}{p} + NC_{\text{comm}} + T_{\text{overhead}}, \quad (2.8)$$

де T_{overhead} — витрати на агрегацію результатів, балансування і керування розподіленими процесами, NC_{comm} — витрати на комунікацію (передача генома/вхідних даних/результату), синхронізація, очікування.

Максимально досяжне теоретичне прискорення описується аналогом закону Амдаля [100]:

$$S(p) \leq \frac{1}{(1-P) + \frac{P}{p} + \text{CommOverhead}}, \quad (2.9)$$

де P — частка часу одного покоління, що підлягає паралелізації, тобто, оцінка придатності (у NEAT зазвичай $P \approx 0.95 - 0.99$ для складних симуляцій), $(1 - P)$ — залишок часу на репродукцію, видоутворення та логіку, CommOverhead — додатковий відносний час на комунікацію. На практиці

досягне прискорення є сублінійним через накладні витрати синхронізації та обмеження пропускну здатності мережі.

Для практичного проектування і оцінки ресурсів можна використовувати наближений вираз для обчислення загального часу навчання з використанням розподіленої оцінки придатності:

$$T_{total} \approx G \left(\frac{N\gamma L\bar{V}}{p} + NC_{comm} + \alpha N \log N + \beta N(\bar{W} + \bar{V}) + \delta \frac{p}{p+p_{sat}} \right), \quad (2.10)$$

- де α, β, γ — константи, що відображають витрати на обчислення видоутворення, репродукції та активації/ваги одного зв'язку; δ — коефіцієнт накладних часових витрат масштабування, що відображає вартість керування процесами, синхронізації, конкуренції за кеші CPU, деградації ефективності при надмірному паралелізмі; p_{sat} — параметр сатурації паралелізму; безрозмірна величина, що визначає характерну кількість воркерів, після якої подальше збільшення p майже не зменшує час виконання, він залежить від наявних апаратних ресурсів та конкретної реалізації конкурентних примітивів мовою програмування.

Параметри δ і p_{sat} визначаються емпірично шляхом калібрувальних прогонів та залежать від:

- архітектури апаратної платформи,
- реалізації конкурентних примітивів мови GO,
- структури обміну даними між компонентами Master–Worker,
- гранулярності задач оцінювання придатності.

Доданок $\delta \frac{p}{p+p_{sat}}$ моделює гладкий перехід до плато продуктивності:

- при $p \ll p_{sat}$ накладні витрати зростають майже лінійно з p ;
- при $p \gg p_{sat}$ цей член прямує до δ , що відображає досягнення межі масштабованості, обумовленої апаратними та системними обмеженнями.

Висновки. Аналіз показує, що основним джерелом обчислювальної складності NEAT є етап оцінки придатності, який має лінійну залежність від розміру популяції та кількості зв'язків у мережі, але зростає суперлінійно в

процесі еволюції через збільшення топологічної складності. Це створює передумови для впровадження розподілених методів оцінки придатності, які дозволяють суттєво зменшити час еволюції при збереженні якості пошуку та генетичної різноманітності популяції.

2.3. Комплексна метрика оцінки ефективності експериментів у нейроеволюційних алгоритмах

Оцінювання продуктивності нейроеволюційних алгоритмів є критичним етапом для об'єктивного порівняння різних конфігурацій, моделей або реалізацій. Зокрема, при аналізі ефективності алгоритму NEAT та його модифікацій доцільно використовувати узагальнені метрики, які відображають не лише успішність розв'язання задачі, але й ресурсну ефективність процесу еволюції. У цьому дослідженні для порівняльного аналізу застосовується метрика EfficiencyScore, реалізована в бібліотеці goNEAT [109], яка дозволяє інтегрально оцінити якість еволюції за декількома ключовими показниками.

Формальне визначення метрики EfficiencyScore. Метрика базується на поєднанні трьох груп показників:

1. успішності алгоритму,
2. якості знайдених рішень,
3. обчислювальної та структурної ефективності еволюції.

Узагальнена формула метрики має вигляд:

$$\text{EfficiencyScore} = \frac{SF}{\ln(TCG) + \epsilon}, \quad (2.11)$$

де

- S — частка успішних експериментів від загальної кількості запусків;
- F — нормалізоване значення середнього fitness-вимірювання;
- T — середній час однієї епохи;
- C — середня топологічна складність мереж-переможців (вузли + зв'язки);
- G — середня кількість поколінь, необхідних для досягнення рішення;
- $\ln(\cdot)$ — логарифмічна функція, що нормалізує вплив добутку часових та структурних характеристик;
- ϵ — константа для забезпечення числової стабільності формули. $\epsilon = 1$

Змістове трактування компонентів метрики

Успішність алгоритму (S). Компонент success rate визначає ймовірність того, що алгоритм зможе знайти рішення у дозволений кількості поколінь. Цей

показник є критичним у випадках, коли оцінка fitness має різко виражену «крутість» ландшафту або коли виникає проблема плато під час еволюції.

Якість знайдених рішень (F). Нормалізований fitness характеризує відповідність результату оптимальній або бажаній поведінці агента. Нормалізація дозволяє порівнювати різні задачі або різні конфігурації алгоритму на уніфікованій шкалі.

Обчислювальна та структурна ефективність (T, C, G). Ці три компоненти відображають ресурсоємність процесу еволюції:

- **T (час)** — триваліші епохи свідчать про високу обчислювальну складність, що негативно впливає на загальну ефективність.
- **C (структурна складність)** — мережі з надлишковою кількістю вузлів та зв'язків погіршують узагальнення, збільшують час обчислень і можуть свідчити про деградацію мінімальної структури.
- **G (швидкість збіжності)** — високе значення означає повільний пошук рішення, що зменшує продуктивність алгоритму.

Добуток $T \cdot C \cdot G$ нормалізується логарифмом для запобігання непропорційного зростання знаменника при великих значеннях будь-якої зі складових.

Переваги застосування метрики EfficiencyScore

Застосування даної метрики має низку суттєвих переваг у контексті дисертаційного дослідження:

- **Комплексність:** об'єднує якісні, кількісні та ресурсні характеристики еволюції в одному числовому показнику.
- **Справедлива нормалізація:** завдяки логарифмічному згладжуванню забезпечується збалансований вплив часових та структурних метрик.
- **Порівнюваність:** дозволяє об'єктивно порівнювати різні реалізації NEAT, у тому числі класичний, модифікований або розподілений варіанти.
- **Відповідність науковим вимогам:** дозволяє формалізувати ефективність запропонованого розподіленого методу оцінки придатності та підтвердити переваги експериментально.

Значення метрики у контексті дослідження

У рамках дисертаційної роботи метрика EfficiencyScore використовується як *основний інтегральний показник ефективності*, що дозволяє:

- оцінити продуктивність розподіленого методу оцінки придатності;
- порівняти роботу алгоритму NEAT до і після модифікацій;
- проаналізувати вплив паралельності на швидкість збіжності та топологічний розмір мереж;
- кількісно підтвердити наукову новизну та практичну цінність запропонованого підходу.

Висновки. Таким чином, застосування метрики EfficiencyScore забезпечує формальний, об'єктивний та науково обґрунтований підхід до оцінки ефективності алгоритмів нейроеволюції, що відповідає вимогам сучасних досліджень у галузі комп'ютерних наук.

2.4. Аналіз існуючих підходів до паралельної еволюції нейронних мереж

Паралельна еволюція штучних нейронних мереж охоплює широкий спектр методів, спрямованих на прискорення еволюційного пошуку або підвищення його ефективності шляхом розподілення обчислень між багатьма вузлами. Нижче наведено огляд основних парадигм, що застосовуються в сучасних підходах.

Master–Worker (Task-farming / Evaluation-level parallelism)

У моделі Master–Worker центральний вузол зберігає популяцію та виконує генетичні операції, тоді як обчислення функції придатності делегуються численним робочим вузлам [101].

Переваги:

- Проста реалізація та відтворюваність.
- Хороша масштабованість при важких обчислювальних задачах на симуляцію середовища та оцінку придатності (evaluation-dominated).
- Підтримує асинхронні та синхронні режими.

Обмеження:

- Master може стати вузьким місцем при дуже великому числі workers (агрегація результатів, видоутворення).
- Висока комунікаційна активність при частих синхронізаціях.
- Потребує механізмів балансування навантаження (pull/push, adaptive chunking).

Підходить для NEAT коли: оцінка домінує у часі, геноми компактні, середовище симульоване локально на worker.

Island Model (coarse-grained parallelism)

Популяція розбивається на кілька незалежних підпопуляцій (островів), які еволюціонують автономно; періодично між ними відбувається міграція [102].

Переваги:

- Підтримує більшу генетичну різноманітність; зменшує ризик локальної збіжності.

- Зменшує частоту глобальної синхронізації (міграції відбуваються рідше).
- Добра масштабованість — кожний острів може бути master-worker по відношенню до своїх workers.

Обмеження:

- Параметри міграції (частота, кількість, політика) сильно впливають на якість і швидкість.
- Складніше управління параметрами та аналіз поведінки системи.
- Потребує додаткових механізмів при об'єднанні статистик і результатів.

Підходить для NEAT коли: прагнете зберегти різноманіття і/або мати гетерогенні середовища/задачі; ефективно працює у хмарних/кластерних умовах.

Fine-grained parallelism (Cellular / diffusion models)

Кожна особина або геном розміщується на вузлах великої сітки, де виконуються локальні операції відбору/кросовера із сусідами (демонструє «локальну» еволюцію) [103].

Переваги:

- Натуральна моделювання локальної взаємодії, повільна дифузія генетичної інформації сприяє збереженню локальних оптимумів.
- Підвищена стійкість до локальної загубленості різноманіття.

Обмеження:

- Важче досягти швидкого глобального пошуку.
- Складніше реалізувати в гетерогенних середовищах.
- Менш ефективне використання великої кількості вузлів при evaluation-dominated задачах.

Підходить для NEAT коли: ціниться локальна еволюція й популяційна топологія сама по собі є об'єктом дослідження.

Parallelization of genetic operators (Genotype-level parallelism)

Паралельне виконання операцій кросовера і мутацій над численними парами/геномами — підхід орієнтований на внутрішню паралельність відтворення [104].

Переваги:

- Висока внутрішня паралельність і простота реалізації на мультикорних та SIMD-архітектурах.
- Використовується разом з іншими парадигмами (наприклад, master–worker).

Обмеження:

- Не вирішує проблеми домінування оцінки за часом.
- Менша вигода якщо evaluation є основним витратним етапом.

Підходить для NEAT коли: мутації/кросовери є важкими (наприклад, складні генетичні трансформації), або в гібридних реалізаціях.

Distributed Speciation / Parallel Speciation

Розпаралелити процес кластеризації (видоутворення) або використовувати евристики, що дозволяють уникнути квадратичної вартості популяції [105].

Переваги:

- Знижує часові витрати на видоутворення у великих популяціях.
- Полегшує масштабування master–worker архітектури.

Обмеження:

- Виклик синхронізації для підтримки консистентності «видів».
- Може вимагати додаткових індексуючих структур або наближених методів порівняння.

Підходить для NEAT коли: N дуже велике або видоутворення існує як вузьке місце.

Asynchronous evolutionary models

Workers оцінюють та повертають результати незалежно; master оновлює популяцію без очікування завершення усіх завдань (steady-state або island/asynchronous GA) [101].

Переваги:

- Зменшує простоту від повільних вузлів; краще використовувати гетерогенні ресурси.

- Може підвищити пропускну здатність і скоротити wall-clock час до досягнення цілі.

Обмеження:

- Утруднює аналіз і теоретичну гарантію еквівалентності з класичним NEAT.
- Може змінити динаміку селекції та видоутворення, що потребує ретельного налаштування.

Підходить для NEAT коли: інфраструктура гетерогенна або бажано максимізувати швидкість експериментів.

GPU / SIMD та батчеве оцінювання

Виконується пакетна оцінка багатьох мереж одночасно на GPU, особливо ефективна для простих feedforward мереж або при великих вхідних батчах [106].

Переваги:

- Великий приріст швидкості для однакових/подібних архітектур; висока пропускну здатність.
- Підходить для великого числа коротких forward-проходів.

Обмеження:

- Не підходить для динамічних або сильно різних топологій – важко батчувати різноманітні мережі.
- Потребує додаткового кодування для динамічних графів (наприклад, padding, grouped execution).

Підходить для NEAT коли: використовуються обмежені сімейства архітектур або застосовується попередня класифікація мереж за розміром для батчингу.

Surrogate-assisted / Fitness approximation

Застосувати дешевші апроксимаційні моделі (surrogates) для частини оцінок, зменшуючи кількість повних симуляцій [107].

Переваги:

- Значне скорочення обчислень при збереженні орієнтиру для пошуку.
- Може добре працювати у поєднанні з розподіленими обчисленнями.

Обмеження:

- Вимагає навчання/оновлення surrogates; ризик помилок апроксимації і втрати якості пошуку.
- Потребує політики вибору, коли викликати повну оцінку.

Підходить для NEAT коли: повні симуляції дуже дорогі, і можливе успішне навчання surrogates.

Гібридні моделі (Island + Master–Worker + GPU + Surrogates)

Комбінувати підходи для досягнення балансу між якістю, швидкістю та гнучкістю. Наприклад, island model для генетичної диверсифікації + master–worker для експлуатації evaluation-level паралелізму + surrogates для економії ресурсів.

Переваги:

- Гнучкість для різних робочих навантажень і інфраструктури.
- Кращі реальні показники при складних задачах.

Обмеження

- Висока складність реалізації та налаштування.
- Необхідність продуманих механізмів контролю і перевірки.

Практичні критерії вибору підходу для NEAT

- Домен задачі
 - Складні симуляції (робототехніка) → master–worker або island + surrogates.
 - Швидкі forward-прогони великої кількості однотипних мереж → GPU-батчинг.
- Розмір і різноманітність популяції
 - Дуже велика N → island model або оптимізоване видоутворення.
- Інфраструктура
 - Гетерогенна (хмара + локальні вузли) → асинхронні та адаптивні стратегії балансування.
 - Кластери з низькою латентністю → синхронні моделі з ефективною агрегацією.

- Потреба у відтворюваності
 - Синхронна модель простіша для точного відтворення результатів.
- Баланс між швидкістю і якістю
 - Island + migration покращує якість за рахунок невеликого збільшення затримок.

Висновок. Сучасні підходи до паралельної еволюції нейронних мереж утворюють спектр від простого master–worker до складних гібридних архітектур. Для задач, де домінує оцінка придатності (типово для NEAT у складних симуляціях), найбільш ефективними є моделі, що паралелізують саме оцінювальний етап (master–worker, island, асинхронні стратегії), за умови ретельної організації балансування навантаження і мінімізації комунікаційних накладних витрат.

2.5. Розподілений підхід до оцінки придатності в алгоритмі NEAT та обґрунтування вибору моделі Master-Worker

Як було показано у підпункті 2.1., етап оцінки придатності генерує найбільше обчислювальне навантаження в класичному алгоритмі NEAT. Оскільки придатність кожної особини визначається незалежно від інших, цей етап *є природним кандидатом для паралелізації*. Розподілений підхід до оцінки придатності забезпечує можливість масштабування еволюційного процесу на кластери, хмарні обчислювальні платформи або гетерогенні обчислювальні системи, що дозволяє суттєво скоротити час навчання.

Мотивація використання розподіленої оцінки придатності. Основні причини переходу до розподіленої моделі такі:

- **Високе навантаження на CPU/GPU** при симуляції середовища або пропусканні великих датасетів через фенотипні нейронні мережі;
- **Зростання складності мереж у процесі еволюції:** збільшення кількості вузлів і зв'язків призводить до лінійного або суперлінійного росту часу оцінки;
- **Природна паралельність задачі:** оцінка придатності не потребує прямої взаємодії особин, що дає можливість виконувати її незалежно на різних обчислювальних вузлах;
- **Необхідність прискорення великих експериментів:** підвищення продуктивності дозволяє виконувати більше експериментальних запусків і зменшити загальний час дослідження.

Обґрунтування вибору моделі Master-Worker для розподіленого NEAT. Ефективна паралелізація алгоритму NEAT потребує обрання такої архітектури обчислень, яка мінімізує синхронізаційні витрати, забезпечує стабільний розподіл навантаження та не порушує логіку еволюційного процесу. Серед різних моделей паралельних еволюційних обчислень — ізольованих островів, демів, асинхронних популяцій та моделі Master-Worker — саме остання найбільш природно відповідає структурі операцій NEAT. Це обумовлено кількома ключовими властивостями алгоритму.

По-перше, основною обчислювальною складовою NEAT є багаторазове виконання процедур оцінки придатності (fitness evaluation) для кожного фенотипу в популяції. Ці операції є незалежними одна від одної, що дозволяє легко винести їх у віддалені обчислювальні вузли без необхідності копіювання або розділення генетичної інформації між підпопуляціями. У моделі Master–Worker майстер-процес централізовано керує еволюційним циклом, а робітники займаються виключно обчисленням придатності для окремих особин. Така декомпозиція відповідає природному поділу задач у NEAT, де більшість часу витрачається на симуляцію фенотипів, а не на генетичні операції.

По-друге, модель Master–Worker спрощує підтримання глобальної когнітивної та структурної консистентності популяції. NEAT використовує складний механізм інноваційних номерів та видів (speciation), який вимагає центрального відстежування історії мутацій, збереження таблиць інновацій та коректної класифікації особин. У розподілених острівних моделях підтримання цієї узгодженості потребує синхронізованих обмінів між островами або складних протоколів узгодження. У моделі Master–Worker необхідні дані залишаються в одному місці — у майстра, що повністю усуває ці труднощі та знижує ризик розходження генетичних ліній.

По-третє, така архітектура забезпечує гнучке масштабування під навантаження. Кількість робітників можна збільшувати відповідно до складності функції придатності або розміру популяції. У задачах, де симуляція середовища є дороговартісною (наприклад, фізичні моделі, складні агенти чи довгі епізоди підкріплення), це дає лінійне або майже лінійне прискорення. Тоді як в острівній моделі швидкість приросту продуктивності часто обмежується потребою періодичного міграційного обміну.

По-четверте, Master–Worker є найбільш передбачуваною в плані балансування навантаження. Майстер може динамічно розподіляти задачі на робітників залежно від їхньої продуктивності, забезпечуючи високий рівень використання всіх доступних ресурсів. Це особливо важливо для NEAT, де різні

особини можуть мати істотно відмінну складність симуляції через різну топологію нейромереж.

Однак оптимальність моделі Master–Worker залежить від певних умов. По-перше, обсяг даних, що передаються між майстром і робітниками, має бути невеликим: геноми у NEAT компактні, що робить передачу швидкою навіть у масштабованих системах. По-друге, латентність мережі має бути відносно низькою, або ж симуляція оточення повинна виконуватись достатньо довго, щоб перекрити витрати на комунікацію. По-третє, функція придатності повинна бути значно дорожчою за керівні операції еволюційного циклу — саме тоді модель Master–Worker дає максимальний виграш продуктивності. Нарешті, виконання генетичних операцій повинно залишатись централізованим, що відповідає класичній архітектурі NEAT.

Таким чином, модель Master–Worker є найбільш придатною для розподілення алгоритму NEAT у випадках, коли (1) оцінка придатності є домінантною частиною обчислень, (2) топологічна складність нейромереж зростає протягом еволюції, (3) важливо зберегти глобальну консистентність інновацій та видів, а також (4) необхідно забезпечити масштабування при використанні гетерогенних або віддалених ресурсів. За таких умов Master–Worker дозволяє істотно зменшити час еволюційної оптимізації, не змінюючи формальної структури NEAT і не порушуючи механізмів його роботи.

За умов складної або нестаціонарної функції фітнесу, Master–Worker забезпечує **максимальне прискорення NEAT** без порушення його еволюційних властивостей.

Традиційна архітектурна модель розподіленої системи Master–Worker. Розподілений підхід ґрунтується на модифікованій моделі «керівник–виконавець». Традиційна архітектура представлена на рис. 2.1 – вона складається з трьох логічних рівнів:

- **Керівник (master).** Забезпечує глобальну координацію еволюційного процесу:
 - формує популяцію та підтримує її генетичну цілісність;

- виконує операції схрещування, мутації та видоутворення (speciation);
- збирає та агрегує значення придатності;
- переходить до наступного покоління після обробки всіх особин.
- **Виконавці (workers).** Окремі обчислювальні вузли, що завершують такі задачі:
 - збір та декодування геному, отриманого від master через брокер завдань;
 - побудова фенотипної нейронної мережі;
 - проведення симуляції або тестування мережі у середовищі;
 - повернення числової оцінки придатності.
- **Брокер завдань (task scheduler).** Забезпечує ефективний обмін даними між master та workers:
 - виконує балансування навантаження;
 - відстежує стан кожного виконавця;
 - забезпечує канали комунікацій.

Передача геному має невеликі накладні витрати, оскільки геном NEAT є компактним набором структурованих даних (список вузлів і зв'язків). Основний час припадає на симуляцію й обчислення вихідного сигналу.

Балансування навантаження. Для ефективності важливо забезпечити рівномірне завантаження всіх виконавців. Для цього можуть застосовуватися такі стратегії:

- **Статичне балансування:** популяція рівними частинами розбивається між виконавцями. Просте, але погано працює при різній складності особин.
- **Динамічне балансування:** кожен виконавець запитує наступну особину після завершення поточної (pull-модель). Забезпечує оптимальне завантаження.
- **Адаптивне балансування:** виконавці з різною продуктивністю отримують різну кількість задач залежно від свого середнього часу обробки.

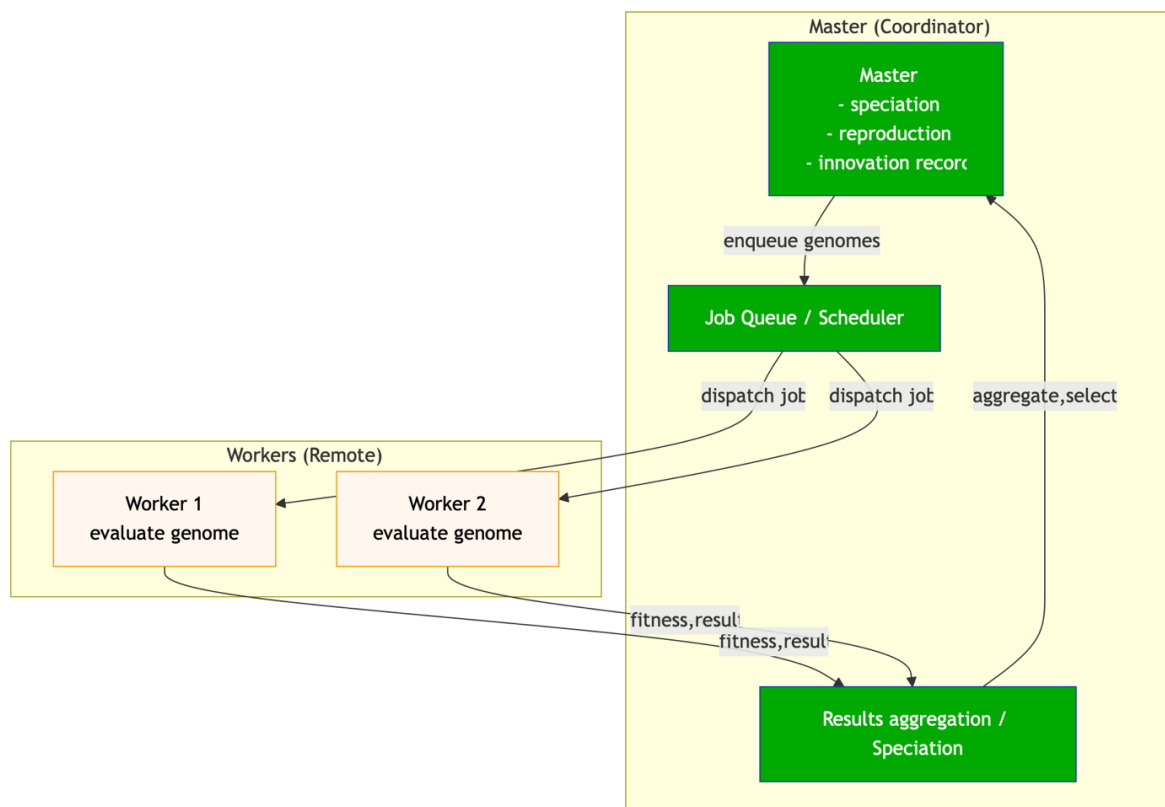


Рис.2.1. Традиційна архітектурна модель Master-Worker для розподілених обчислень придатності організмів алгоритмом NEAT (авторська розробка)

Оцінка придатності є незалежною, тому вузьких місць (bottlenecks) можна уникнути за допомогою асинхронного обміну даними.

Модель комунікацій та накладні витрати. Основні комунікаційні витрати включають:

- передачу геному до виконавця,
- повернення значення придатності,
- синхронізацію між поколіннями.

Як продемонстровано у [108], накладні витрати стають суттєвими лише у випадках:

- невеликих популяцій з простими мережами (де час симуляції малий),
- високої латентності мережі.

У задачах з великою складністю симуляцій (ігрові середовища, робототехніка, системи контролю) накладні витрати становлять <1% загальних витрат. При цьому, оцінка придатності становить до 95–99% усіх обчислень NEAT для подібних симуляцій [101, 108]. Тому, розподіл цього етапу дозволяє досягти майже лінійного прискорення.

Максимальне прискорення може бути вираховано як:

$$S_p \approx \frac{T_{seq}}{T_{gen}^{(p)}}, \quad (2.12)$$

Де T_{seq} – час одного покоління у послідовній реалізації алгоритму, $T_{gen}^{(p)}$ – час одного покоління в розподіленій моделі, що може бути вираховано за формулою (8).

У практичних реалізаціях може досягати 20–50× на кластерах з 32–64 вузлів [106, 108].

Особливості реалізації розподіленої оцінки придатності. До ключових аспектів, які необхідно враховувати при реалізації системи, належать:

- **Синхронна vs асинхронна модель**
 - Синхронна гарантує чіткі покоління, але може чекати повільних виконавців.
 - Асинхронна прискорює еволюцію і дозволяє працювати без чітких бар'єрів поколінь, проте модифікує динаміку відбору.
- **Стабільність видоутворення (speciation).** Необхідно забезпечити однакові умови для кластеризації популяції при розподіленій моделі, щоб уникнути «дрейфу» видів.
- **Повторне використання робітників.** Постійне створення й завершення процесів додає накладні витрати; краще використовувати пул виконавців.
- **Комунікаційні протоколи.** Можуть бути застосовані:
 - gRPC,
 - WebSocket,
 - Message Passing Interface (MPI),
 - ZeroMQ,

- REST (для повільних симуляцій).
- **Толерантність до збоїв.** Виконавці можуть аварійно завершувати роботу, тому необхідна система повторного призначення задач.

Висновки. Розподілений підхід до оцінки придатності дозволяє значно зменшити загальну часову складність алгоритму NEAT, зберігаючи при цьому його еволюційні властивості. Такий підхід *особливо ефективний для складних симуляційних середовищ*, де обчислення придатності домінують над усіма іншими етапами. Застосування розподіленої моделі Master–Worker для оцінки придатності відкриває можливість масштабування NEAT до великих популяцій, збільшення різноманітності та прискорення збіжності еволюції.

2.6. Обґрунтування використання вбудованих механізмів мови GO у модифікованій моделі Master–Worker

Паралелізація процесу оцінки придатності особин у популяції алгоритму NEAT є ключовим інструментом підвищення обчислювальної ефективності систем еволюційного конструювання нейронних мереж. Вибір засобів паралельних обчислень визначає можливість масштабування, рівень накладних витрат, простоту реалізації та стійкість до синхронізаційних конфліктів. У цьому підрозділі здійснюється порівняльний аналіз сучасних підходів: Go (goroutines + channels), класичних потоків (threads), MPI та Actor-based систем (Erlang/Akka). Аналіз орієнтовано на застосування до моделі Master–Worker у задачах розподіленої оцінки придатності в NEAT. Результати наведено у таблиці 2.1.

Критерії порівняння. Для оцінювання підходів використано такі критерії:

1. Вартість створення та планування паралельних одиниць виконання
2. Модель обміну даними та синхронізації
3. Простота розробки та відповідність моделям розподілених обчислень
4. Масштабованість у межах одного вузла та в розподіленому середовищі
5. Надійність, стійкість до помилок та детермінізм виконання
6. Придатність до динамічного завантаження Master–Worker

Таблиця 2.1

Порівняння сучасних засобів конкурентного програмування для реалізації модифікованої моделі Master-Worker для оцінки придатності у алгоритмі NEAT.

Критерій порівняння	Go (goroutines + channels)	Threads	MPI	Actor-based Systems
Вартість створення	Дуже низька (кілька KB на goroutine)	Висока (OS thread)	Низька, але з високими накладними витратами на мережевий стек	Середня — Actors легші за threads, але важчі за goroutines

Модель синхронізації	Канали з вбудованою гарантією безпеки	Locks, mutexes, condition variables	Message passing (ручне)	Message passing автоматизований
Складність розробки	Низька; природна модель CSP	Висока через ручну синхронізацію	Висока; вимагає досвіду HPC	Низька–середня; залежить від фреймворку
Масштабування (1 вузол)	Висока, до сотень тисяч goroutines	Низька, обмежено кількістю OS threads	Висока	Висока
Масштабування (кластер)	Потребує додаткових засобів (gRPC, NATS)	Не призначено для цього	Оптимальний варіант, залежить від мережевої інфраструктури	Вбудовано
Стійкість до помилок	Середня; recovery вручну	Низька	Залежить від MPI middleware	Дуже висока (self-healing)
Придатність для Master–Worker у NEAT	Відмінна для вузла; проста інтеграція	Обмежена через синхронізаційні конфлікти	Добра для великих кластерів	Добра, але складність інтеграції
Загальна ефективність	Найкраща для одновузлової паралелізації	Нижча через важку синхронізацію	Висока для HPC	Висока, але вимагає інсталяції platform runtime

Далі ми розглянемо кожний підхід більш детально.

Go (goroutines + channels)

Модель паралелізації Go побудована на принципах CSP (Communicating Sequential Processes), де обмін даними виконується через канали, а не через спільну пам'ять. Це значно зменшує ризики race conditions і deadlocks.

Goroutine є надзвичайно легкою одиницею виконання, що дозволяє запускати їх у величезній кількості, забезпечуючи масивний паралелізм при мінімальних накладних витратах.

Переваги у контексті NEAT:

- природне відображення Master–Worker як pipeline з каналами задач;
- динамічний розподіл навантаження;
- низькі затрати контекстного перемикавання;
- простота коду та відсутність складних механізмів синхронізації.

Недоліки:

- обмежена масштабованість за межами одного вузла;
- наявність GC створює нерівномірні паузи (не критично для NEAT);
- планувальник Go не дозволяє точного low-level control.

Модель потоків (threads)

Потоки забезпечують повний контроль над виконанням, але вимагають ручної обробки синхронізації через mutex/locks, що значно ускладнює реалізацію.

Для NEAT це створює такі проблеми:

- складність розподілу задач без блокувань,
- висока вартість перемикавання потоків,
- збільшення накладних витрат зі зростанням популяції.

Застосування потоків має сенс лише у низькорівневих оптимізаціях або там, де потрібна жорстка прив'язка до CPU-ресурсів.

MPI (Message Passing Interface)

MPI є стандартом для високопродуктивних обчислень і забезпечує чудову масштабованість у кластерах, але програє Go у простоті реалізації та швидкості розробки.

MPI найбільш ефективний коли:

- обсяг задач великий,
- існує кластер з виділеною мережею,
- потрібна детермінована комунікація.

Для NEAT MPI актуальний лише для дуже великих експериментів, проте невиправдано складний для середніх проєктів.

Actor-based Systems

Підходи типу Erlang/Akka забезпечують високу стійкість до помилок і мають природну модель message-passing.

Вони чудово працюють у розподілених системах, але зазвичай накладні витрати на кожного актора вищі, ніж у goroutines. Це робить їх менш придатними для масової паралелізації на рівні тисяч об'єктів оцінки придатності.

Висновки. Аналіз показує, що для задачі локальної паралельної оцінки придатності в алгоритмі NEAT оптимальним підходом є Go (goroutines + channels), оскільки він:

- забезпечує найнижчу вартість паралельності;
- природно реалізує модель Master–Worker;
- мінімізує ризики синхронізаційних помилок;
- дозволяє динамічно балансувати навантаження;
- значно спрощує реалізацію порівняно з threads і MPI.

MPI може бути кращим лише у випадку багатовузлових НРС-кластерів, тоді як Actor-системи корисні для розподілених fault-tolerant застосувань.

У контексті задач дисертації підхід Go є найбільш збалансованим та впроваджуваним, що підтверджує його вибір як основи розробленої моделі паралелізації Master–Worker для алгоритму NEAT.

2.7. Опис нової моделі Master–Worker на основі засобів конкурентного програмування мови GO

Процес оцінки придатності організмів у алгоритмі NEAT є одним із найбільш ресурсоємних етапів еволюційного циклу, оскільки передбачає незалежне моделювання поведінки кожного індивіда. Ця операція належить до класу *embarrassingly parallel* задач, що створює природні передумови для її ефективного розпаралелювання. Одним з оптимальних підходів до реалізації локальної паралельності є використання моделей конкурентного програмування, зокрема архітектури Master–Worker. У межах цієї роботи така модель реалізована на основі механізмів *goroutines* та *channels* мови програмування Go, що забезпечують низку суттєвих переваг порівняно з традиційними багатопоточними підходами.

Запропонована архітектура модифікованої моделі Master–Worker представлена на рис. 2.2.

Ця діаграма чітко показує функціонування моделі на різних логічних рівнях:

Master

- виконує видоутворення, селекцію, кросовер/мутації, оновлює глобальний реєстр інновацій
- створює *goroutines*-воркерів
- надсилає геноми вирішувачів через *jobs channel*
- отримує результати оцінки придатності через *results channel*

Workers

- нескінченно читають *jobs channel*
- обчислюють *fitness*
- відправляють назад *results channel*

Channels

- двоспрямований неблокуючий механізм комунікації
- забезпечують синхронізацію між *goroutines*

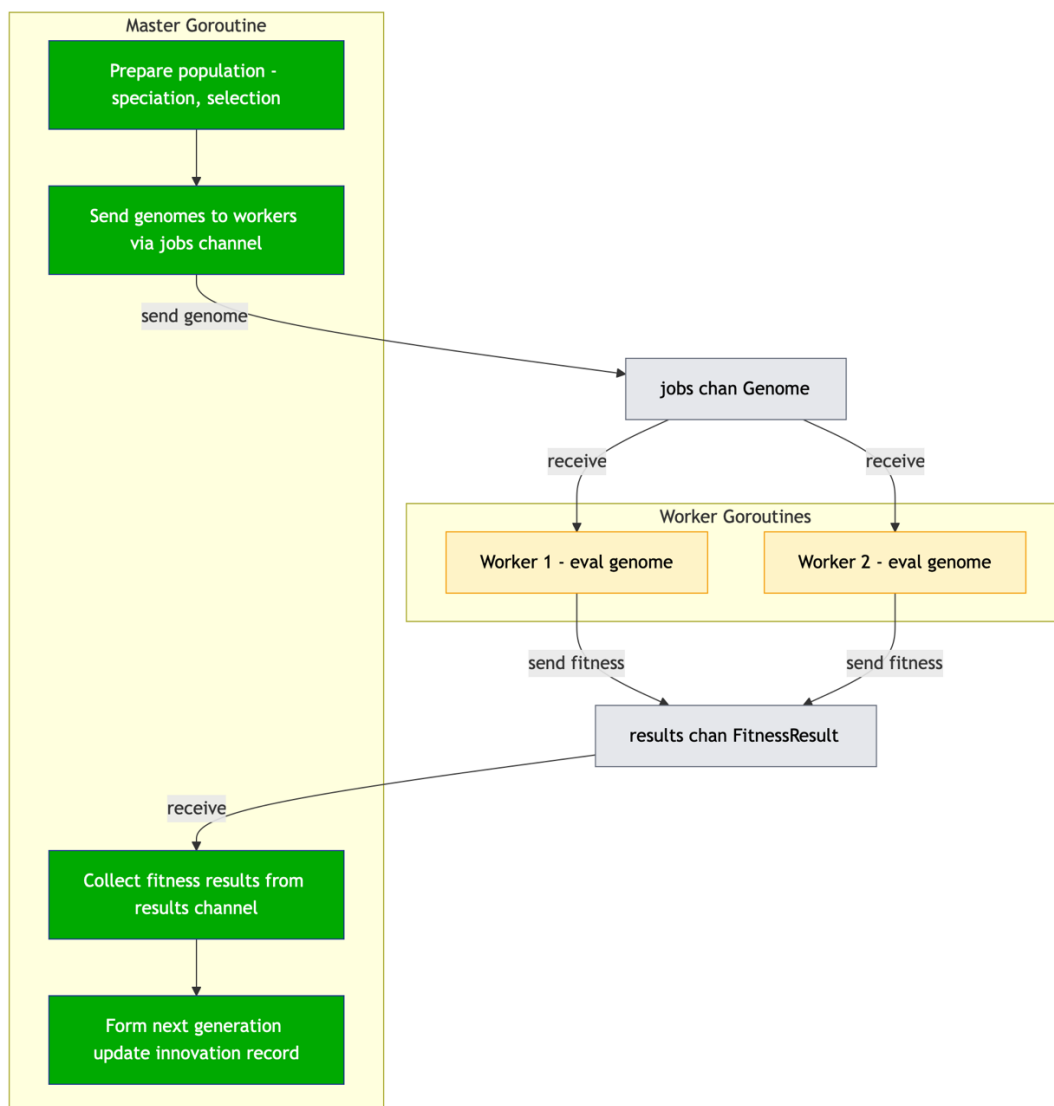


Рис. 2.2. Модифікована архітектурна модель Master-Worker на основі вбудованих механізмів мови GO (авторська розробка)

Для реалізації запропонованої архітектури було розроблено алгоритм, що використовує вбудовані механізми конкурентного програмування мови GO, такі як goroutines і channels для реалізації модифікованої моделі Master-Worker, який представлено на рис. 2.3 в узагальненому вигляді. Більш детально, цей алгоритм буде розглянуто у експериментальній частині цієї роботи. В ньому функція **evaluate(organism)** виконує запуск складного симуляційного середовища, необхідного для обчислення цільової функції та визначення значення придатності організму, переданого їй як параметр.

Механізми міжпроцесної взаємодії та синхронізації обчислювальних процесів реалізовано із використанням стандартних примітивів і структур стандартної бібліотеки мови програмування GO, зокрема `channels` та `sync.WaitGroup`, що забезпечує коректну паралельну координацію виконання.

Algorithm 1: NEAT training with parallel fitness estimation

Input: A number of *Epochs* of evolution to execute.

Result: List of found successful solvers or empty list.

begin

```

1. Initialize randomized Population of simplest organisms;
foreach epoch  $\in$  Epochs do
    Results  $\leftarrow$  List() ;
    // evaluate fitness of each organism in parallel process
    foreach organism  $\in$  Population do
        // start Go routine to evaluate objective function in
        // complex simulated environment
        result  $\leftarrow$  go evaluate(organism) ;
        Results  $\leftarrow$  Results  $\cup$  future ;

    FoundWinners  $\leftarrow$  List() ;
    // wait for all started processes to be completed and
    // process results
    Wait(Results) ;

    foreach result  $\in$  Results do
        if results.HasBeenSolved() then
            FoundWinners  $\leftarrow$  FoundWinners  $\cup$  results.Organisms() ;

    if FoundWinners.length() > 0 then
        return FoundWinners
    else
        2. Run NEAT algorithm over evaluated Population to produce
        new Population for the next epoch of evolution

```

Рис. 2.3. Узагальнений алгоритм модифікованої моделі Master-Worker
(авторська розробка)

Запропонована модифікована архітектура Master-Worker має цілий ряд переваг та недоліків які ми розглянемо далі.

Переваги використання goroutines як обчислювальних воркерів:

- **Легковаговість та ефективне управління ресурсами.** На відміну від потоків операційної системи, goroutines мають мінімальний початковий

розмір стека (приблизно 2–4 KB), який динамічно розширюється залежно від потреби, на відміну від потоків ОС, де розмір стека фіксований (часто 1–8 MB). Це дозволяє створювати тисячі або десятки тисяч паралельних воркерів, що є критично важливим у випадку великих популяцій NEAT. Відсутність суттєвих накладних витрат на створення та переключення контексту забезпечує високий рівень масштабованості експериментальної системи.

- **Низькі накладні витрати на переключення контексту.**

Перемикання між goroutines здійснюється планувальником Go на рівні користувацького простору, що забезпечує значно меншу вартість контекстного перемикання порівняно з потоками ОС. Такий підхід дозволяє ефективно виконувати задачі симуляції малої та середньої тривалості, характерні для оцінки придатності організмів у NEAT.

- **Автоматичне використання багатоядерності.** Планувальник Go реалізує M:N модель відображення, що спрощує використання багатоядерних систем. Розподіл goroutines по доступних ядрах відбувається автоматично, без необхідності ручного керування потоками чи балансування навантаження. Це підвищує продуктивність без значного ускладнення програмної логіки.

Переваги використання channels для синхронізації у моделі Master–Worker:

- **Безпечний обмін даними без блокувальних примітивів.**

Комунікаційні канали (channels) гарантують послідовну та безпечну передачу повідомлень між goroutines без використання mutex, lock або інших блокувальних примітивів. Це дозволяє мінімізувати ризики гонок даних (data races), що є важливим для коректної еволюції геномів, значення придатності яких повертається воркерами.

- **Природне відображення архітектури Master–Worker.**

Використання двох каналів — черги задач та черги результатів — відтворює класичну модель Master–Worker:

- канал задач (jobs) передає геноми для оцінки,
- канал результатів (results) повертає розраховану придатність.

Таким чином, програмна структура відображає концептуальну модель, значно спрощуючи як розробку, так і подальше обслуговування.

- **Блокувальна семантика для автоматичної координації.**

Блокувальна поведінка операцій над каналами (<- та chan <-) забезпечує природну координацію між master та worker-процесами, запобігаючи переповненню черг або втраті даних. Цей механізм особливо корисний у разі нерівномірного навантаження між воркерами.

Узгодженість моделі Master–Worker з природою еволюційних обчислень NEAT

Алгоритм NEAT передбачає незалежну оцінку кожного організму в популяції, що повністю відповідає парадигмі «розподілити → обчислити → зібрати». Використання goroutines та channels дозволяє максимально ефективно реалізувати цей цикл, забезпечуючи:

- мінімальні витрати на комунікацію між підпроцесами,
- відсутність синхронізаційних конфліктів,
- можливість масштабування до великої кількості незалежних воркерів.

Таким чином, застосована архітектура є природною з точки зору структури обчислювального процесу та вимог алгоритму NEAT.

Обмеження та недоліки обраного підходу

Попри численні переваги, використання goroutines та channels має низку обмежень:

- **Відсутність нативної підтримки розподілених обчислень.** Go забезпечує ефективну локальну паралельність, але не містить вбудованих механізмів для мультивузлових обчислень (на відміну від MPI або Spark), що потребує додаткових зовнішніх інструментів. Це не є критичним у рамках цієї роботи так як тут розглядається підтримка ресурсно-обмежених середовищ, які зазвичай одновузлові.

- **Імовірність виникнення deadlock.** Неправильне закриття каналів або недбалий дизайн потоків може спричинити взаємне блокування. Це створює

підвищені вимоги до коректності реалізації. Вирішується за рахунок використання відповідних шаблонів програмування, що зменшують ризик подібних явищ.

- **Обмежений вплив на планувальник.** Механізм планування goroutines оптимізований під загальні випадки, але користувач не може змінювати пріоритети виконання або політики планування на рівні ОС.

Умови, за яких модель Master–Worker на goroutines є оптимальною

Запропонована модель забезпечує максимальну продуктивність за таких умов:

- система є одновузловою або має спільну пам'ять,
- популяція організмів є великою ($N \geq 1000$),
- час оцінки одного організму є незначним або середнім,
- обсяг міжпроцесної комунікації мінімальний,
- кількість доступних ядер процесора ≥ 4 .

У таких випадках goroutines + channels забезпечують найкраще співвідношення між продуктивністю, простотою реалізації та ефективністю використання ресурсів.

Висновки. Використання механізмів goroutines та channels у моделі Master–Worker для реалізації паралельної оцінки придатності в алгоритмі NEAT є науково обґрунтованим та технологічно доцільним. Застосування цих механізмів дозволяє суттєво знизити обчислювальні витрати, підвищити продуктивність еволюційних обчислень та забезпечити можливість масштабування експериментів на сучасних багатоядерних обчислювальних системах. Попри існуючі обмеження, переваги підходу роблять його оптимальним рішенням для реалізації високопродуктивної локальної паралельності в рамках дослідження.

2.8. Метод оптимізації алгоритму NEAT для підвищення швидкості побудови топології ШНМ та покращення ефективності знайдених рішень в умовах обмежених обчислювальних ресурсів

Одним із головних обчислювальних обмежень класичного алгоритму NEAT є значні часові витрати на етапі оцінки придатності кожного організму в популяції. Цей етап передбачає симуляцію поведінки розвиненої нейронної мережі у середовищі задачі й зазвичай є найдорожчим за часом компонентом еволюційного процесу. Зі збільшенням розміру популяції, складності середовища або тривалості симуляції час виконання NEAT може зростати надлінійно.

Для усунення цього обмеження у даній роботі запропоновано метод паралельної оцінки придатності, який базується на модифікованій моделі Master–Worker з використанням механізмів конкурентності мови програмування Go (goroutines та channels). Метод дозволяє реалізувати повністю паралельний та динамічно збалансований розподіл задач оцінювання придатності між незалежними робітниками, значно зменшуючи загальний час еволюції.

До складу розробленого методу включені три моделі, що були описані в цьому розділі:

- Модифікована модель Master-Worker розподілених обчислень для паралельної оцінки придатності рішень у алгоритмі NEAT побудована з використанням вбудованих у мову GO механізмів конкурентного програмування – goroutines і channels
- Удосконалена модель оцінки загальних часових витрат на весь процес еволюції під керуванням алгоритму NEAT
- Комплексна інтегральна модель оцінки ефективності знайдених рішень (фенотипів ШНМ)

Зазначені моделі взаємодіють між собою та алгоритмом NEAT як показано на рис. 2.4.

Запропонований метод складається з наступних етапів:

1. Визначення оптимальної кількості воркерів (p) для мінімізації загального часу з використанням моделі часових витрат згідно визначених параметрів NEAT
2. Формування популяції та генерування списку організмів.
3. Запуск пулу воркерів (goroutines) у кількості p .
4. Передавання організмів у канали задач.
5. Паралельна симуляція середовища кожним воркером.
6. Збирання результатів через канал вихідних значень.
7. Оновлення структури популяції та видів.
8. Оцінка інтегральної ефективності знайденого рішення.

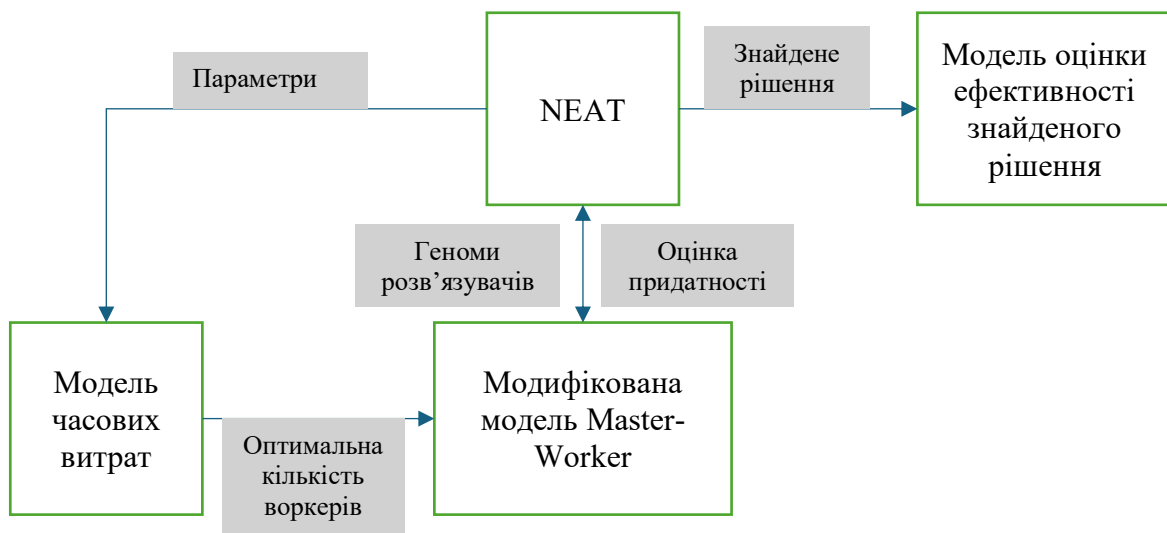


Рис. 2.4. Взаємодія між моделями та алгоритмом NEAT (авторська розробка)

Таким чином метод забезпечує:

- максимальне використання паралелізму,
- динамічне балансування навантаження,
- незалежність воркерів,
- гарантію коректності еволюційного процесу,
- мінімізацію накладних витрат на синхронізацію.

На рис. 2.5 наведено повний цикл роботи паралельного NEAT згідно розробленого методу.

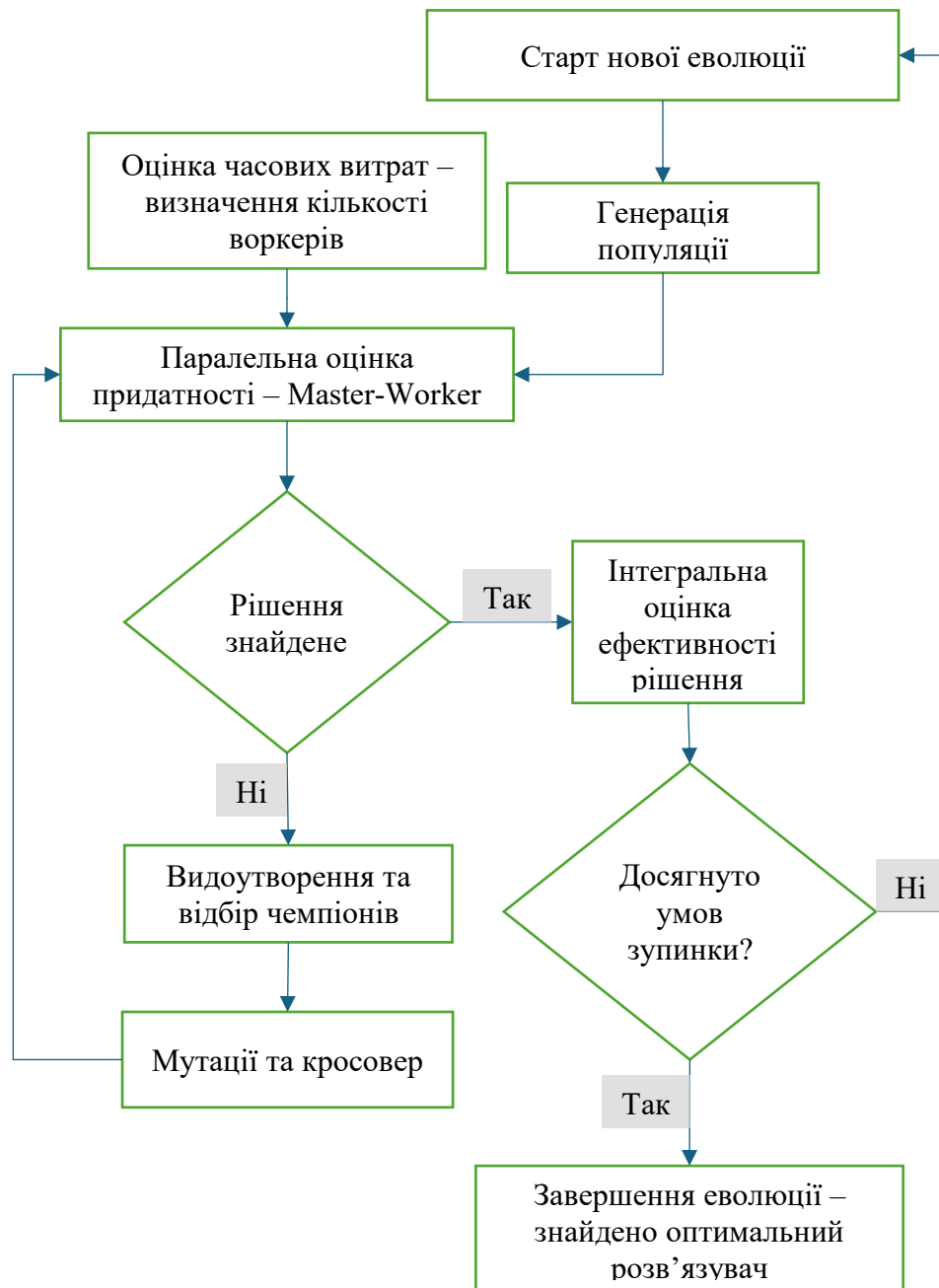


Рис. 2.5. Повний цикл роботи паралельного NEAT згідно запропонованого методу (авторська розробка)

Висновки. У другому розділі було розроблено та обґрунтовано метод оптимізації алгоритму NEAT, який забезпечує істотне прискорення процесу побудови топології штучних нейронних мереж та підвищення ефективності

отримуваних рішень, у середовищах з обмеженими обчислювальними ресурсами.

У сукупності результати розділу демонструють, що розроблений метод оптимізації NEAT є універсальним, масштабованим та орієнтованим на практичне застосування у ресурсно-обмежених обчислювальних середовищах, включно з робототехнікою, краєвими (edge) обчисленнями та автономними системами керування. Запропонований підхід усуває основні обмеження класичного NEAT щодо продуктивності та дозволяє суттєво підвищити ефективність процесу побудови нейронних топологій у складних задачах.

Такий метод становить важливий внесок у розвиток нейроеволюційних алгоритмів і відкриває нові можливості їхнього практичного застосування.

2.9. Вибір методів оцінки якості та метрик ефективності

Оцінювання ефективності запропонованого методу оптимізації алгоритму NEAT здійснюється на основі системи кількісних критеріїв, які дозволяють комплексно проаналізувати як якість знайдених рішень, так і обчислювальну ефективність нейроеволюційного процесу в умовах обмежених ресурсів. Вибір критеріїв зумовлений необхідністю забезпечити об'єктивне порівняння класичної послідовної реалізації алгоритму NEAT із запропонованою паралельною реалізацією, що базується на модифікованій моделі Master–Worker.

Функціональна придатність рішень.

Основним критерієм якості роботи нейроеволюційного алгоритму є значення функції придатності (*fitness score*), що характеризує здатність продукованої штучної нейронної мережі виконувати поставлену керуючу або оптимізаційну задачу. У рамках проведених експериментів функція придатності визначається на основі тривалості підтримання системи у допустимому робочому стані або ступеня досягнення цільових обмежень.

Для забезпечення стабільності процесу навчання та кращої диференціації невдалих і частково успішних рішень застосовується логарифмічне масштабування значень цільової функції. Це дозволяє уникнути надмірної концентрації значень придатності в області малих часових інтервалів, що є характерним для складних динамічних задач керування.

Часові характеристики еволюційного процесу.

Другим ключовим критерієм оцінювання є сукупні часові витрати на виконання нейроеволюційного процесу. Вони включають:

- час оцінки придатності одного організму;
- час обробки одного покоління;
- загальний час еволюції до досягнення заданого рівня якості рішення.

Для формалізації цього аспекту використовується запропонована у роботі модель оцінки часових витрат (формула 2.10), яка враховує кількість організмів у популяції, складність симуляційного середовища, кількість паралельних виконавців та накладні витрати на міжпроцесну комунікацію. Такий підхід

дозволяє не лише емпірично вимірювати продуктивність, але й аналітично прогнозувати масштабованість запропонованого методу.

Інтегральна ефективність реалізації.

З метою узагальненої оцінки результатів експериментів вводиться інтегральний показник ефективності, який поєднує у собі якість знайдених рішень і часові витрати на їх отримання (формула 2.11). Даний показник дозволяє коректно порівнювати різні конфігурації алгоритму NEAT, зокрема за різної кількості паралельних виконавців, та визначати області раціонального використання обчислювальних ресурсів.

Інтегральний критерій є особливо важливим у контексті практичного застосування нейроеволюційних алгоритмів, де досягнення максимальної точності часто має бути збалансоване з обмеженнями на час навчання та доступні апаратні ресурси.

Стабільність та відтворюваність результатів.

Додатково оцінюється стабільність еволюційного процесу, яка визначається варіативністю результатів між незалежними запусками алгоритму за однакових початкових умов. Для цього аналізуються середні значення, дисперсії та довірчі інтервали показників придатності і часу виконання. Такий аналіз дозволяє оцінити чутливість алгоритму до стохастичних факторів і підтвердити відтворюваність отриманих результатів.

Критерії масштабованості.

Окрему групу критеріїв становлять показники масштабованості, які відображають залежність продуктивності алгоритму від кількості паралельних виконавців. Аналізуються як абсолютне прискорення, так і відносна ефективність паралелізації. Це дозволяє визначити межі доцільності подальшого збільшення рівня паралелізму та виявити вплив накладних витрат на комунікацію між процесами.

Таким чином, можна виділити наступні показники, що складають критерії оцінювання ефективності запропонованого методу:

1. Фактичний час еволюції (експерименту)

Тобто час за який було знайдене рішення або час досягнення граничних вимог (кількості епох симуляції) якщо рішення не було знайдене.

$$T_{\text{exp}}$$

2. Прискорення системи

Може бути вираховано згідно формули 12 наступним чином:

$$A(p) = \frac{T(1)}{T(p)}, \quad (2.13)$$

де $T(1)$ – час виконання у послідовному варіанті оцінки придатності, $T(p)$ – час виконання з паралельною оцінкою придатності з використанням p воркерів.

3. Ефективність використання воркерів

Є похідної отриманого прискорення від загальної кількості воркерів, що були задіяні.

$$E(p) = \frac{A(p)}{p}. \quad (2.14)$$

4. Інтегральна оцінка ефективності (EfficiencyScore)

Розроблена модель, що враховує декілька параметрів: час, стабільність, топологічні характеристики мережі та обчислювальну складність. Значення інтегральної ефективності може бути вираховано згідно формули 2.11 наступним чином:

$$\text{EfficiencyScore} = \frac{S \cdot F}{\ln(T \cdot C \cdot G) + \varepsilon},$$

де, S — частка успішних експериментів від загальної кількості запусків, F — нормалізоване значення середнього fitness-вимірювання, T — середній час однієї епохи, C — середня топологічна складність мереж-переможців (вузли + зв'язки), G — середня кількість поколінь, необхідних для досягнення рішення, $\ln(\cdot)$ — логарифмічна функція, що нормалізує вплив добутку часових та структурних характеристик, ε — константа для забезпечення числової стабільності формули. ($\varepsilon = 1$)

Запропонована система критеріїв оцінювання забезпечує багатовимірний аналіз ефективності нейроеволюційного алгоритму NEAT та дозволяє

обґрунтовано оцінити переваги запропонованого методу паралельної оцінки придатності. Поєднання функціональних, часових та інтегральних показників створює надійну основу для порівняльного аналізу та підтвердження наукової і практичної цінності отриманих результатів.

2.10. Нові наукові результати за результатами дослідження

У ході виконання дисертаційної роботи було отримано низку нових наукових результатів, які поглиблюють теоретичні основи та розширюють практичні можливості нейроеволюційних підходів до побудови топології штучних нейронних мереж. Основні висновки сформульовано наступним чином.

1. **Вперше** запропоновано модель розподілених обчислень Master–Worker для алгоритму NEAT, що базується виключно на вбудованих механізмах конкурентного виконання мови програмування GO (goroutines та channels) та забезпечує ефективне масштабування на багатоядерних системах з мінімальними комунікаційними накладними витратами.

2. **Удосконалено та вперше розроблено єдиний формалізований аналітичний підхід до оцінювання та оптимізації продуктивності алгоритму NEAT, що ґрунтується на поєднанні двох взаємодоповнювальних моделей.** У межах цього підходу, по-перше, *удосконалено модель оцінки ефективності побудови топології штучних нейронних мереж* алгоритмом NEAT на основі декомпозиції часових витрат еволюційного циклу, в якій вперше формалізовано аналітичний зв'язок між структурною складністю нейромереж (кількістю вузлів і зв'язків), розміром популяції, кількістю паралельних виконавців (воркерів) та характеристиками продуктивності апаратної платформи, що забезпечує можливість аналітичного прогнозування часу виконання та визначення оптимальної кількості воркерів для конкретної експериментальної задачі. По-друге, *вперше запропоновано комплексну інтегральну метрику оцінки продуктивності алгоритму NEAT*, яка формалізовано поєднує часові характеристики обчислень, швидкість збіжності еволюційного процесу та якість фінального розв'язку, що дозволяє здійснювати об'єктивне порівняння різних реалізацій, архітектур паралельного виконання та параметричних конфігурацій нейроеволюційних алгоритмів, а також слугує критерієм вибору найбільш ефективних обчислювальних режимів в умовах обмежених обчислювальних ресурсів.

3. **Вперше** запропоновано метод оптимізації алгоритму NEAT для підвищення швидкості побудови топології штучних нейронних мереж, що базується на інтеграції паралельної моделі Master–Worker для паралельної оцінки придатності, моделі часових витрат та інтегральної метрики оцінки ефективності отриманих рішень та продуктивності алгоритму NEAT.

Метод ґрунтується на інтеграції трьох моделей, розроблених у межах даної дисертації, а саме:

- моделі оцінки часових витрат – для багатоцільової оптимізації часу виконання проти кількості задіяних воркерів;
- моделі інтегральної оцінки ефективності – для оцінки ефективності знайдених рішень з метою вибору найбільш ефективного (менш енергозатратного) рішення;
- паралельної моделі Master–Worker, адаптованої до можливостей мови програмування GO через використання goroutines і каналів – для прискорення нейроеволюційного процесу за рахунок паралельного обчислення оцінок придатності рішень.

2.11. Висновки до другого розділу

У другому розділі дисертаційної роботи розроблено та теоретично обґрунтовано метод оптимізації алгоритму нейроеволюції доповнених топологій (NEAT), спрямований на підвищення швидкості побудови топології штучних нейронних мереж та зростання загальної ефективності еволюційного процесу в умовах обмежених обчислювальних ресурсів. Основну увагу зосереджено на оптимізації етапу оцінки придатності, який було ідентифіковано як домінантне джерело обчислювальних витрат у класичних реалізаціях NEAT.

У розділі виконано формальну постановку задачі дослідження, в межах якої оптимізація розглядається не лише з точки зору покращення значень функції придатності окремих організмів, а як задача підвищення ефективності алгоритму NEAT у цілому як обчислювального процесу. Такий підхід дозволив перейти від суто евристичного прискорення до системного аналізу часових витрат і ресурсної ефективності еволюції.

Проведений формальний аналіз обчислювальної складності показав, що етап оцінки придатності має лінійну залежність від розміру популяції та складності фенотипних нейронних мереж, але у процесі еволюції призводить до суперлінійного зростання часу виконання через поступове ускладнення топології. Аналітична модель часових витрат дала змогу виокремити ключові параметри, що визначають продуктивність NEAT, та обґрунтувати доцільність розпаралелювання саме цього етапу як найбільш ефективного напрямку оптимізації.

У межах розділу здійснено систематичний аналіз існуючих підходів до паралельної еволюції нейронних мереж, включно з моделями master–worker, island model, fine-grained паралелізації, асинхронних еволюційних схем, GPU-орієнтованих методів та surrogate-assisted підходів. На основі цього аналізу показано, що для алгоритму NEAT, у якому оцінка придатності є обчислювально домінантною, найбільш природною та ефективною є модель паралелізації на рівні оцінювання особин популяції.

Обґрунтовано вибір архітектурної моделі Master–Worker як базової для розподіленої оцінки придатності в NEAT. Показано, що дана модель забезпечує збереження еволюційної логіки алгоритму, централізоване управління інноваційними номерами та видоутворенням, а також дозволяє гнучко масштабувати обчислення за рахунок динамічного балансування навантаження між виконавцями. Особливо підкреслено, що така архітектура мінімізує ризики порушення коректності еволюційного процесу, які можуть виникати у більш децентралізованих моделях.

Вперше у межах даного дослідження *вперше запропоновано* реалізацію модифікованої моделі Master–Worker для алгоритму NEAT на основі вбудованих механізмів конкурентного програмування мови GO — goroutines та channels. Показано, що використання легковагових одиниць виконання та безпечної каналної комунікації дозволяє досягти високого рівня паралелізму з мінімальними накладними витратами, спростити реалізацію та забезпечити ефективне використання багатоядерних обчислювальних систем без залучення важких зовнішніх паралельних фреймворків.

У розділі *вперше запропоновано* удосконалену аналітичну модель оцінки часових витрат еволюційного процесу, яка враховує взаємозв'язок між розміром популяції, складністю топології мереж, кількістю паралельних виконавців і накладними витратами масштабування. Запропонована модель дозволяє не лише пояснювати експериментально спостережувані ефекти прискорення та насичення продуктивності, але й аналітично прогнозувати оптимальну кількість воркерів для конкретної конфігурації задачі та апаратної платформи.

Для забезпечення об'єктивного та комплексного оцінювання ефективності розробленого методу *вперше запропоновано* інтегральну метрику EfficiencyScore, яка поєднує показники успішності еволюції, якості знайдених рішень, часових витрат, топологічної складності мереж і швидкості збіжності. Використання цієї метрики дозволяє здійснювати коректне порівняння різних реалізацій та режимів виконання NEAT, а також формалізувати компроміс між якістю рішень і витратами обчислювальних ресурсів.

У підсумку в другому розділі *вперше запропоновано* цілісний метод оптимізації алгоритму NEAT, який базується на інтеграції паралельної моделі Master–Worker, аналітичної моделі часових витрат і комплексної метрики оцінки ефективності. Запропонований метод забезпечує істотне скорочення часу еволюційного процесу без погіршення якості знайдених нейронних мереж, зберігає стохастичну природу та еволюційні властивості алгоритму і є придатним для використання в умовах обмежених обчислювальних ресурсів.

Отримані результати створюють науково обґрунтовану основу для експериментальної перевірки ефективності розробленого підходу та демонструють його практичну цінність для застосування нейроеволюційних алгоритмів у задачах робототехніки, автономних систем керування, моделювання складних динамічних середовищ та крайових (edge) обчислень.

РОЗДІЛ 3. ЕКСПЕРИМЕНТАЛЬНІ ДОСЛІДЖЕННЯ МЕТОДУ ОПТИМІЗАЦІЇ АЛГОРИТМУ NEAT ДЛЯ ПІДВИЩЕННЯ ШВИДКОСТІ ПОБУДОВИ ТОПОЛОГІЇ ШНМ ТА ПОКРАЩЕННЯ ЕФЕКТИВНОСТІ ЗНАЙДЕНИХ РІШЕНЬ

3.1. Мета, задачі та загальна методика експериментальних досліджень

Метою експериментального розділу є кількісне оцінювання ефективності запропонованого методу розподіленої оцінки придатності організмів у популяції алгоритму NEAT та визначення оптимальних параметрів паралелізації для прискорення еволюційного процесу без втрати його результативності.

Для досягнення поставленої мети було сформульовано такі основні задачі:

- виконати порівняльний аналіз часу еволюції між класичною та паралельною реалізаціями алгоритму NEAT;
- кількісно оцінити вплив кількості воркерів (p) моделі Master–Worker на повний час еволюції та ефективність системи;
- перевірити відповідність теоретичної моделі часових витрат реальним експериментальним даним;
- побудувати Pareto-фронт оптимальних параметрів еволюції для низки практичних задач (повний час еволюції проти задіяна кількість воркерів);
- визначити діапазон значень p , який забезпечує найкраще співвідношення між швидкістю та ефективністю використання ресурсів;
- виконати аналіз стабільності експериментальних результатів за допомогою статистичних методів.

Усі експерименти проводилися з використанням авторської реалізації алгоритму NEAT мовою програмування GO [109] та модифікованої моделі Master–Worker, що базується на вбудованих у мову інструментах конкурентного програмування goroutines та channels.

3.1.1. Методика проведення досліджень

Методика проведення досліджень у даній роботі базується на поетапному поєднанні теоретичного аналізу, експериментального моделювання та кількісної оцінки отриманих результатів. Такий підхід дозволив обґрунтовано дослідити вплив запропонованого методу паралельної оцінки придатності на продуктивність нейроеволюційного алгоритму NEAT та підтвердити його ефективність у практичних умовах. На рисунку 3.1 наведено блок-схему методики досліджень.

Етап теоретичної оцінки

На першому етапі досліджень було виконано теоретичну оцінку оптимальної кількості паралельних виконавців (воркерів), що залучаються до процесу оцінки придатності організмів у популяції. Для цього застосовано розроблену в межах роботи модель оцінки часових витрат еволюційного процесу, яка враховує параметри популяції, складність симуляційного середовища, кількість поколінь, а також накладні витрати на комунікацію та синхронізацію між воркерами.

Оцінка виконувалася з урахуванням конкретної експериментальної задачі — балансування подвійного оберненого маятника, що характеризується високою обчислювальною складністю симуляції та є репрезентативною для аналізу продуктивності нейроеволюційних алгоритмів. На основі отриманих теоретичних залежностей було визначено область раціональних значень кількості воркерів, у якій очікується максимальне прискорення при мінімальних накладних витратах.

Етап експериментального моделювання

На другому етапі було проведено серію експериментальних запусків алгоритму NEAT у двох конфігураціях:

- з класичною однопотоковою схемою оцінки придатності організмів;

- з паралельною схемою оцінки придатності, реалізованою на основі модифікованої моделі Master–Worker з використанням goroutines та channels мови програмування GO.

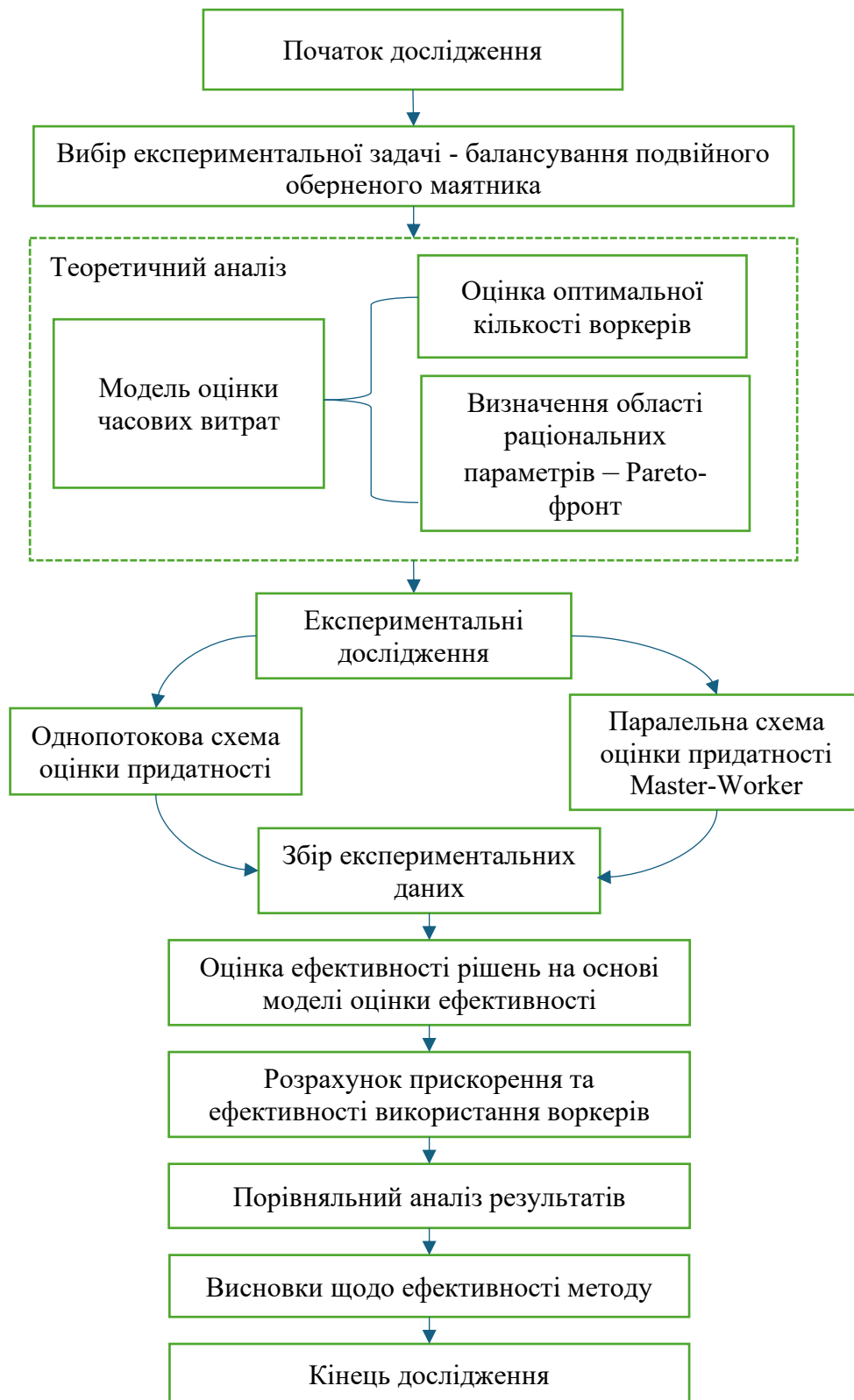


Рис. 3.1. Блок-схема методики досліджень (авторська розробка)

Для забезпечення коректності порівняння всі експерименти виконувалися за однакових початкових умов, параметрів алгоритму NEAT та конфігурації середовища симуляції. Кожна конфігурація запускала багаторазово з метою усереднення результатів та зменшення впливу стохастичних факторів нейроеволюційного процесу.

Етап оцінювання ефективності

На третьому етапі здійснювалася оцінка ефективності знайдених рішень із використанням запропонованої в роботі моделі оцінки ефективності. Аналіз охоплював:

- значення функції придатності керуючих ШНМ, що були побудовані в процесі нейроеволюції;
- часові витрати на побудову рішень;
- інтегральний показник ефективності реалізації алгоритму NEAT.

На основі отриманих експериментальних даних обчислювалося досягнуте прискорення паралельної реалізації відносно однопотокової схеми, а також ефективність використання воркерів, що характеризує ступінь наближення до ідеального лінійного прискорення.

Підсумковий аналіз

Завершальним етапом методики стало узагальнення результатів теоретичної оцінки та експериментальних досліджень з метою виявлення відповідності між прогнозованими та фактичними показниками продуктивності. Отримані результати дозволили підтвердити доцільність використання запропонованого методу паралельної оцінки придатності та визначити умови його найбільш ефективного застосування для складних нейроеволюційних задач.

3.1.2. Висновки до розділу

У підрозділі 3.1 сформульовано мету експериментальних досліджень, визначено сукупність задач, необхідних для її досягнення, а також описано загальну методику проведення експериментів, спрямованих на перевірку ефективності запропонованого методу підвищення ефективності алгоритму

NEAT за рахунок паралельної оцінки придатності. Мета експериментів чітко орієнтована на кількісне та якісне підтвердження теоретичних положень, викладених у попередніх розділах, зокрема запропонованих вперше моделей часових витрат та інтегральної оцінки ефективності, а також модифікованої моделі паралельних обчислень Master-Worker.

Запропонована методика досліджень базується на поєднанні аналітичного обґрунтування параметрів експерименту з практичними обчислювальними випробуваннями у контрольованому симуляційному середовищі. Такий підхід забезпечує відтворюваність результатів, коректність порівняння однопотокової та паралельної реалізацій алгоритму NEAT, а також дозволяє дослідити вплив кількості воркерів на часові характеристики та якість знайдених рішень.

Визначені задачі експериментальних досліджень охоплюють повний цикл перевірки запропонованого методу — від теоретичного визначення оптимальної конфігурації обчислювальних ресурсів до аналізу досягнутого прискорення, ефективності використання паралелізму та оцінки стабільності еволюційного процесу. Це створює цілісну експериментальну рамку, достатню для обґрунтованих висновків щодо практичної доцільності та масштабованості розробленого підходу.

Таким чином, результати підрозділу 3.1 формують методологічну основу експериментальної частини дисертації та забезпечують логічний перехід до детального опису тестового середовища, параметрів експериментів і аналізу отриманих результатів у наступних підрозділах третього розділу.

3.2. Експериментальний стенд та програмне забезпечення

Апаратна конфігурація

Експерименти виконувалися на сучасній багатоядерній обчислювальній платформі зі стандартною конфігурацією:

- багатоядерний процесор (Apple M3 Max, 16 CPU cores),
- оперативна пам'ять 36 ГБ,
- операційна система на базі macOS Tahoe 26.2,
- локальне виконання моделі Master–Worker без мережеских накладних витрат або з їх моделюванням за допомогою параметра `Ccomm`.

Програмно-алгоритмічне забезпечення

Для проведення досліджень використовувалися такі інструменти:

- GO 1.24 — основне середовище розробки;
- авторська реалізація NEAT (goNEAT), опублікована на Zenodo [109];
- авторська реалізація моделі оцінки ефективності мовою програмування GO, опублікована на Zenodo [109];
- авторська реалізація модифікованої моделі Master–Worker з використанням goroutines та channels, опублікована на Zenodo [109];
- авторська реалізація Python-бібліотеки для реалізації запропонованої моделі часових витрат, опублікована на Zenodo [110];
- авторська реалізація Python-бібліотеки для побудови графіків та 3D-моделей, опублікована на Zenodo [110].

Реалізація всіх компонентів є відтворюваною та доступною у відкритому доступі.

3.3. Опис експерименту

Експериментальні дослідження у даній роботі спрямовані на перевірку ефективності запропонованого методу оптимізації алгоритму NEAT для підвищення швидкості побудови топології ШНМ та на підтвердження коректності теоретичних моделей оцінки часових витрат і інтегральної ефективності. Для цього було обрано складну динамічну задачу керування, яка є репрезентативною з точки зору обчислювальної складності та широко використовується для оцінювання нейроеволюційних методів.

3.3.1. Опис тестового середовища

Для оцінювання обрано класичне тестове завдання нейроеволюції: Double Pole Balancing (DPB) — балансування подвійного оберненого маятника.

Завдання DPB є загальноприйнятим бенчмарком, що дозволяє оцінити:

- швидкодію еволюційних алгоритмів,
- чутливість до вибору параметрів,
- ефективність паралелізації обчислень.

Система з двома оберненими маятниками (double-pole balancer) є нестійкою динамічною системою, у якій центр мас розташований вище точки опори. Її можна стабілізувати шляхом прикладання зовнішніх сил за допомогою спеціалізованої системи, що відстежує кут відхилення маятників та коригує положення точки опори в горизонтальному напрямку, утримуючи центр мас у стані рівноваги в момент початку падіння. Це класичне завдання динаміки та теорії керування, яке широко використовується як бенчмарк для перевірки різних стратегій керування, зокрема тих, що базуються на методах навчання з підкріпленням.

У межах даного дослідження увагу зосереджено на реалізації специфічного алгоритму керування, який використовує нейроеволюційні методи для стабілізації обернених маятників протягом заданого проміжку часу. Оцінювання придатності популяції розв'язувачів передбачає симуляцію системи з оберненими маятниками, реалізованої у вигляді візка, що може переміщуватися

горизонтально, з точкою опори, закріпленою у верхній частині конструкції, як показано на рис. 3.2.

Керувальний сигнал у цій системі представляє собою силу, що прикладається до центра мас візка на кожному кроці симуляції, причому сила має однакову амплітуду, але може бути спрямована в протилежні боки. Такий тип системи керування відомий як **bang-bang controller** (релейний керувач), оскільки він оперує лише двома дискретними значеннями керувального впливу.

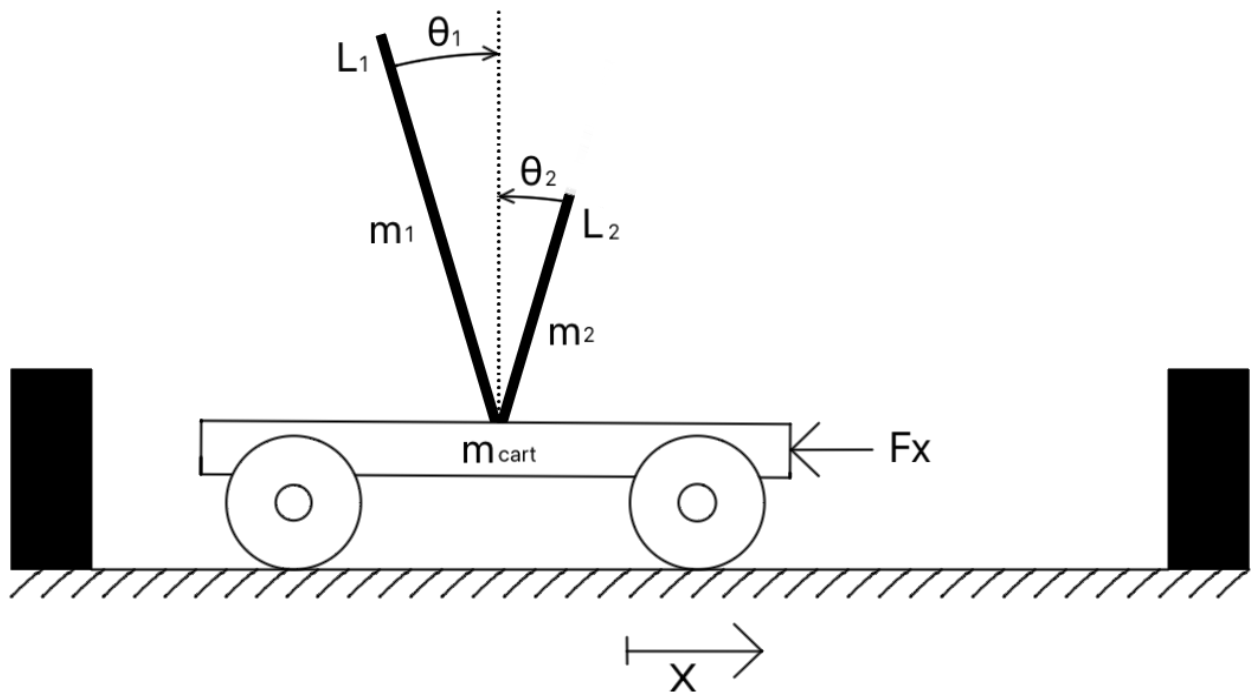


Рис. 3.2. Схема завдання на балансування оберненого маятника з двома щоглами (за матеріалами [108])

Метою керувальної ШНМ у цьому експерименті є генерування імпульсів сили, що прикладаються до центра мас візка, таким чином, щоб обидва маятники залишалися у збалансованому стані якомога довше, а сам візок при цьому не виходив за встановлені горизонтальні межі.

На входи керувальної нейронної мережі подається поточне положення візка на рейці (x), його лінійна швидкість ($\dot{x}$), кут відхилення першого маятника

(Θ_1) та його кутова швидкість $(\dot{\Theta}_1)$, а також кут відхилення другого маятника (Θ_2) та його кутова швидкість $(\dot{\Theta}_2)$.

Рівняння руху для двох нез'єднаних обрєнених маятників, збалансованих на одному візку, за умови нехтування тертям між колесами візка та поверхнею рейки, мають наступний вигляд:

$$\ddot{x} = \frac{F_x - \sum_{i=1}^2 \tilde{F}_i}{M + \sum_{i=1}^2 \tilde{m}_i}, \quad (3.1)$$

$$\ddot{\theta}_i = -\frac{3}{4L_i} \left(\ddot{x} \cos \theta_i + g \sin \theta_i + \frac{\mu_{pi} \dot{\theta}_i}{m_i L_i} \right), \quad (3.2)$$

де $\tilde{F}_i$ – це сила реакції i -ї щогли на візок згідно наступного рівняння:

$$\tilde{F}_i = m_i L_i \dot{\theta}_i^2 \sin \theta_i + \frac{3}{4} m_i \cos \theta_i \left(g \sin \theta_i + \frac{\mu_{pi} \dot{\theta}_i}{m_i L_i} \right), \quad (3.3)$$

де m_i – це ефективна маса i -ї щогли згідно рівняння:

$$\tilde{m}_i = m_i \left(1 - \frac{3}{4} \cos^2 \theta_i \right), \quad (3.4)$$

У таблиці 3.1 наведені параметри середовища симуляції, що були використанні при проведенні експерименту.

Таблиця 3.1

Основні параметри середовища симуляції балансування оберненого маятника з двома щоглами.

Символ	Опис	Значення
x	Позиція візка на траку	$[-2.4, 2.4] \text{ m}$
θ	Кут відхилення щогли від вертикалі	$[-0.63, 0.63] \text{ rad}$
F_x	Контролююча сила прикладена до візка	$\pm 10 \text{ N}$
L_i	Відстань від центра мас маятника до точки його кріплення	$L_1 = 0.5 \text{ m}$ $L_2 = 0.05 \text{ m}$
M	Вага візка без щогл	1.0 kg
m_i	Вага i -ї щогли	$m_1 = 0.1 \text{ kg}$ $m_2 = 0.01 \text{ kg}$
μ_{pi}	Коефіцієнт тертя у точці кріплення i -ї щогли	0.000002
g	Прискорення вільного падіння	-9.8 m/s^2

Сигнал підкріплення (reinforcement signal). Середовище симуляції повинно надавати мінімально необхідну інформацію про стан системи після

виконання керувальних дій у вигляді сигналу підкріплення. Цей сигнал вказує, чи порушує система з балансування подвійного маятника встановлені обмежувальні умови після прикладення керувального впливу. Він може бути визначений наступним чином:

$$r_t = \begin{cases} 0 & \text{if } -0.63 \text{ rad} < \theta_i^t < 0.63 \text{ rad and } -2.4 \text{ m} < x_t < 2.4 \text{ m} \\ 1 & \text{otherwise} \end{cases}$$

Вектор стану системи. Стан системи «візок–подвійний маятник» у момент часу t описується шести компонентним вектором:

$$\mathbf{s}(t) = \begin{bmatrix} x(t) \\ \dot{x}(t) \\ \theta_1(t) \\ \dot{\theta}_1(t) \\ \theta_2(t) \\ \dot{\theta}_2(t) \end{bmatrix}, \quad (3.5)$$

де

- $x(t)$ — горизонтальна координата візка;
- $\dot{x}(t)$ — горизонтальна швидкість візка;
- $\theta_1(t)$ — кут відхилення першого (довшого) маятника від вертикалі;
- $\dot{\theta}_1(t)$ — кутова швидкість першого маятника;
- $\theta_2(t)$ — кут відхилення другого (коротшого) маятника;
- $\dot{\theta}_2(t)$ — його кутова швидкість.

Таким чином, динаміка системи у векторній формі може бути подана як:

$$\frac{ds}{dt} = f(\mathbf{s}(t), u(t)), \quad (3.6)$$

де $u(t)$ — керувальний вплив (імпульс сили), що генерується еволюційним контролером.

Початкові умови та оновлення стану. Система стартує з нульовими швидкостями візка та обох маятників. Початкове відхилення довгого маятника становить один градус від вертикалі, тоді як короткий маятник розташований

точно у вертикальному положенні. Тобто, маємо наступні початкові значення параметрів:

$$\begin{aligned}x &= 0 \text{ m}, \\ \dot{x} &= 0 \text{ m/s}, \\ \theta_1 &= \frac{\pi}{180} \text{ rad}, \\ \dot{\theta}_1 &= 0 \text{ rad/s}, \\ \theta_2 &= 0 \text{ rad}, \\ \dot{\theta}_2 &= 0 \text{ rad}.\end{aligned}$$

Стан системи «візок–маятники» оновлюється на кожному кроці симуляції шляхом чисельного наближення рівнянь руху із використанням методу Рунге–Кутти четвертого порядку [111] при розмірі кроку 0,01 секунди. Застосування цього методу забезпечує обчислення реакції системи на основі значень змінних стану поточного кроку інтегрування.

Нові керувальні впливи генеруються кожної 0,02 секунди, що відповідає частоті формування керувальних сигналів 50 Гц, тоді як оновлення стану системи відбувається з частотою 100 Гц.

Метод Рунге–Кутти четвертого порядку для оновлення стану системи.

Нехай динаміка системи описується диференціальним рівнянням:

$$\frac{ds}{dt} = f(s, t), \quad (3.7)$$

де $\mathbf{s}$ — вектор стану системи, а $f(\mathbf{s}, t)$ — функція, що визначає його зміну.

Тоді метод Рунге–Кутти четвертого порядку визначає стан системи x_{n+1} у наступний момент часу $t_{n+1} = t_n + h$ за формулами:

$$k_1 = f(s_n, t_n), \quad (3.8)$$

$$k_2 = f\left(s_n + \frac{h}{2}k_1, t_n + \frac{h}{2}\right), \quad (3.9)$$

$$k_3 = f\left(s_n + \frac{h}{2}k_2, t_n + \frac{h}{2}\right), \quad (3.10)$$

$$k_4 = f(s_n + h k_3, t_n + h), \quad (3.11)$$

$$s_{n+1} = s_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4). \quad (3.12)$$

де h — крок інтегрування (у даному дослідженні $h = 0.01$ секунди).

Керуючі дії. Система керування для експерименту з балансуванням подвійного маятника формує лише два можливі керуючі сигнали: прикладання сили ліворуч або праворуч із сталою амплітудою. Таким чином, керуюча сила у момент часу t може бути визначена наступним чином:

$$F_t = \begin{cases} 10N, & \text{if } a[t] = 1 \\ -10N, & \text{if } a[t] = 0 \end{cases} \quad (3.13)$$

де, $a[t]$ — це сигнал дії отриманий від керуючої ШНМ у момент часу t .

Взаємодія між керуючою ШНМ та середовищем симуляції. Перед подачею на входи керуючої штучної нейронної мережі змінні стану системи масштабуються з метою приведення їхніх значень до діапазону $[0,1]$. Відповідно, рівняння попередньої обробки (нормалізації) вхідних змінних стану мають такий вигляд:

$$s_0[t] = \frac{1}{4.8}(x[t] + 2.4), \quad (3.14)$$

$$s_1[t] = \frac{1}{3}(\dot{x}[t] + 1.5), \quad (3.15)$$

$$s_2[t] = \frac{1}{1.256}(\theta_1[t] + 0.628), \quad (3.16)$$

$$s_3[t] = \frac{1}{4}(\dot{\theta}_1[t] + 2), \quad (3.17)$$

$$s_4[t] = \frac{1}{1.256}(\theta_2[t] + 0.628), \quad (3.18)$$

$$s_5[t] = \frac{1}{4}(\dot{\theta}_2[t] + 2). \quad (3.19)$$

Де значення $[s_0 \dots s_5]$ відповідають вхідним параметрам керуючої ШНМ, а саме: горизонтальному положенню візка, його горизонтальній швидкості, куту відхилення першого маятника від вертикалі та його кутовій швидкості, а також куту відхилення другого маятника і його кутовій швидкості.

3.3.2. Опис цільової функції (функції придатності)

Цільова функція або функція придатності для цієї задачі задається такими рівняннями:

$$L = \frac{\log t_{max} - \log t_{eval}}{\log t_{max}}, \quad (3.20)$$

$$F = 1.0 - L. \quad (3.21)$$

У наведених рівняннях t_{max} — це очікувана кількість часових кроків, задана в конфігурації експерименту (100 000), а t_{eval} — фактична кількість часових кроків, протягом яких керувальний алгоритм був здатний утримувати систему балансування маятника у стійкому стані в межах заданих обмежень.

Логарифмічне масштабування використовується з огляду на те, що більшість випробувань завершуються невдачею вже протягом перших кількох сотень кроків, тоді як максимальна тривалість експерименту становить 100 000 кроків. Застосування логарифмічної шкали забезпечує більш рівномірний розподіл значень придатності, зокрема й для випробувань, що завершуються на ранніх етапах.

Перше з наведених рівнянь (3.20) визначає функцію втрат (loss), значення якої належить інтервалу $[0,1]$, тоді як друге рівняння (3.21) задає показник придатності (fitness), що є доповненням до значення втрат. Таким чином, значення функції придатності також лежать у межах $[0,1]$, причому більші значення відповідають кращим результатам роботи керуючої ШНМ.

3.3.3. Висновки до розділу

У даному розділі було детально описано експериментальне середовище та методику проведення досліджень, спрямованих на оцінювання ефективності запропонованого методу паралельної оцінки придатності в алгоритмі NEAT. Як тестову задачу обрано задачу балансування подвійного оберненого маятника, яка характеризується високою динамічною складністю та значними обчислювальними витратами на симуляцію, що робить її репрезентативною для перевірки масштабованості нейроеволюційних алгоритмів.

Визначено цільову функцію та критерії оцінювання якості знайдених рішень, що дозволяють об'єктивно порівнювати результати, отримані при використанні однопотокової та паралельної схем оцінки придатності.

Таким чином, експериментальна постановка, описана у даному розділі, створює надійну основу для подальшого кількісного аналізу результатів, порівняльної оцінки продуктивності та формування узагальнених висновків щодо доцільності застосування запропонованого методу в практичних задачах нейроеволюції, що розглядається у наступному розділі дисертації.

3.4. Опис алгоритму паралельної оцінки придатності за модифікованою моделлю Master-Worker

Досліджуваний алгоритм реалізує нейроеволюційний пошук керуючої штучної нейронної мережі (контролера) для розв’язання задачі балансування подвійного оберненого маятника (double-pole balancing) шляхом еволюційного розвитку популяції геномів відповідно до алгоритму NEAT. У межах кожного покоління здійснюється відбір, репродукція та структурна модифікація геномів, які кодують топологію та вагові коефіцієнти нейронних мереж, що керують динамікою фізичної системи.

Ключовою особливістю даної задачі є необхідність багаторазового запуску симуляційного середовища для оцінювання якості кожного окремого організму. Саме етап оцінювання придатності, що передбачає чисельну інтеграцію рівнянь руху подвійного маятника на великому часовому інтервалі, є обчислювально найбільш ресурсомісткою частиною алгоритму і визначає загальну тривалість нейроеволюційного процесу.

З метою зменшення часових витрат на цьому етапі в роботі застосовано паралельну схему оцінювання придатності, побудовану на основі модифікованої моделі Master-Worker. У межах цієї моделі центральний керуючий компонент (Master) відповідає за формування завдань оцінювання, їх розподіл між пулом воркерів та агрегацію результатів, тоді як воркери (Worker) виконують незалежні симуляції фізичного процесу для окремих організмів популяції.

Реалізація паралельної моделі базується на використанні конкурентних примітивів мови програмування GO, зокрема легковагових потоків виконання (goroutines), каналів обміну даними (channels) та механізмів синхронізації (WaitGroup). Такий підхід дозволяє ефективно задіяти наявні обчислювальні ресурси багатоядерних систем, мінімізувати накладні витрати на керування потоками виконання та забезпечити масштабованість алгоритму при збільшенні розміру популяції або складності симуляційного середовища.

Вхідні дані

1. *Конфігурація NEAT*: параметри еволюції (розмір популяції, коефіцієнти мутацій/кросинговеру, критерії зупинки тощо).
2. *Початковий геном* (seed genome) або стартова популяція.
3. *Опис експерименту* (double-pole balancing): параметри симуляції, обмеження, горизонт планування, правило підрахунку значення придатності (fitness score).
4. *Параметри паралелізму*: кількість воркерів p . Визначається аналітично за допомогою побудови Парето-оптимального фронту на основі моделі часових витрат.

Вихідні дані

1. Найкращий знайдений геном / фенотип (ШНМ-контролер).
2. Логи/метрики еволюції: покоління, max/avg fitness, швидкодія, час виконання, (за потреби) збережені артефакти експерименту.

3.4.1. Алгоритм – формальний опис

Нехай:

- G — кількість поколінь до зупинки,
- N — розмір популяції,
- p — кількість паралельних воркерів (goroutines),
- $Evaluate(o)$ — функція оцінки придатності організму o через симуляцію фізичного процесу або явища.

Тоді алгоритм має наступний вигляд:

1. **Ініціалізація нейроеволюційного процесу**
 - 1.1. Зчитати конфігурацію NEAT та експерименту.
 - 1.2. Ініціалізувати початкову популяцію $P_0 = o_i$ розміру N .
2. **Цикл еволюції для покоління $t \in [0, G - 1]$**
 - 2.1. **Ініціалізація паралельної оцінки придатності:**
 - Створити канали jobs, results.
 - Запустити p воркерів $W_1 \dots W_p$, кожен виконує цикл:

for $o \in \text{jobs}$: $f \leftarrow \text{Evaluate}(o)$; $\text{results} \leftarrow (o.\text{id}, f)$

2.2. Паралельна оцінка придатності:

- Master надсилає у jobs усі організми P_t .
- Master отримує з каналу results рівно N результатів та присвоює $\text{fitness}(o_i)$ кожному організму у популяції.
- За потреби виконується агрегація статистики покоління (max/avg fitness , час тощо).
- Закрити канал jobs , дочекатися завершення воркерів, закрити канал results .

2.3. Селекція та відтворення (класичний NEAT-пайплайн):

- формування видів (*speciation*), розрахунок розподілу нащадків між видами;
- елітизм (збереження кращих);
- кросинговер та мутації (ваги/вузли/зв'язки);
- формування наступної популяції P_{t+1} розміру N .

2.4. Перевірка критерію зупинки:

- досягнення порогу fitness (рішення знайдено) або максимальної кількості поколінь.

3. Завершення експерименту

- Повернути найкраще знайдене рішення та артефакти експерименту.

Формальна специфікація функції оцінювання

Функція $\text{Evaluate}(o)$ реалізує:

1. побудову фенотипу (ШНМ) з геному o ;
2. прогін симуляції *double-pole balancing* на горизонті T кроків або до порушення обмежень;
3. обчислення скалярного $\text{fitness}(o) \in [0,1]$.

3.4.2. Алгоритм – структурний опис

Алгоритм паралельної оцінки придатності організмів у процесі нейроеволюції реалізовано у вигляді керованої архітектури *Master–Worker*, представлено на рисунку 3.3.

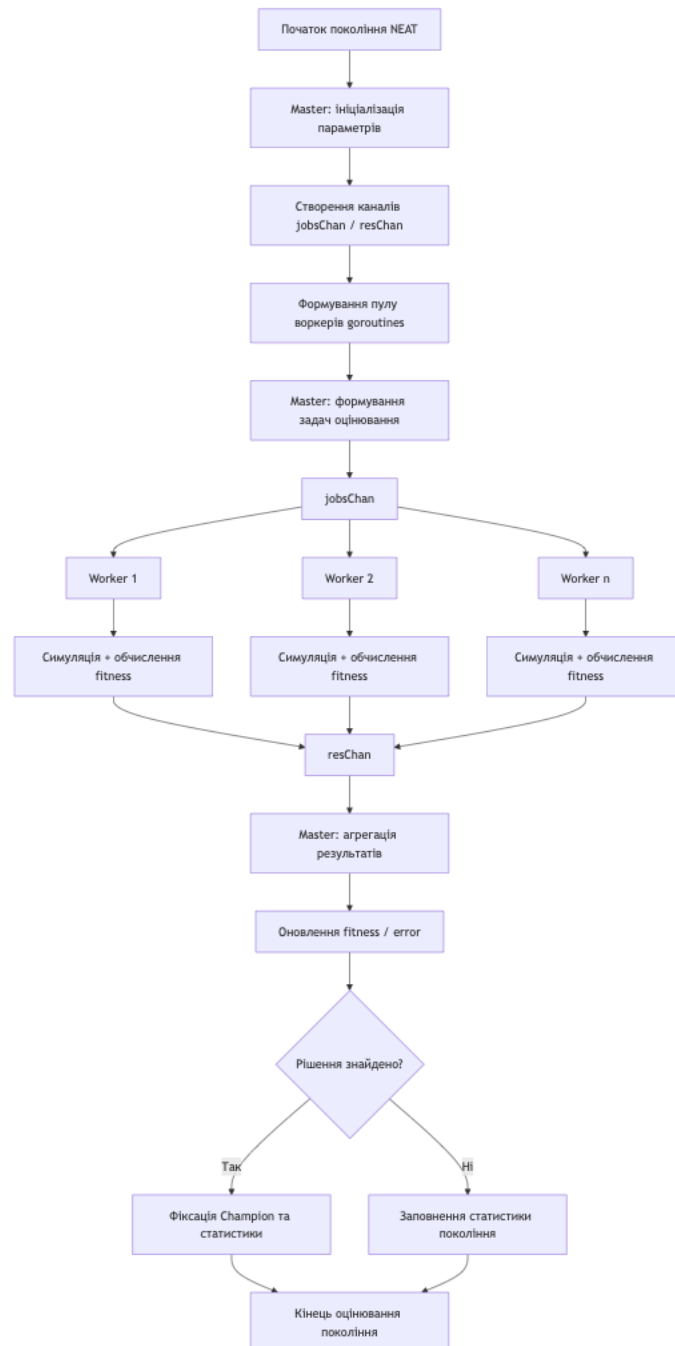


Рис. 3.3. Структурна схема алгоритму паралельної оцінки придатності організмів NEAT за моделлю *Master–Worker* (авторська розробка)

Алгоритм складається з таких функціональних блоків:

1. Ініціалізація керуючого процесу (Master)

- зчитування параметрів NEAT та конфігурації експерименту;
- визначення розміру популяції та кількості воркерів;
- ініціалізація каналів передачі задач (*jobsChan*) і результатів (*resChan*);
- створення структури відповідності ідентифікаторів геномів організмам популяції.

2. Формування пулу воркерів

- запуск фіксованої кількості горутин (воркерів);
- ініціалізація локального середовища симуляції в кожному воркері;
- переведення воркерів у стан очікування задач.

3. Декомпозиція задач і розподіл навантаження

- кожному організму популяції відповідає окрема задача оцінювання;
- задачі асинхронно передаються у канал *jobsChan*;
- вільні воркери динамічно отримують задачі, забезпечуючи баланс навантаження.

4. Паралельна оцінка придатності (Workers)

- виконання симуляції фізичного процесу;
- обчислення значень *fitness* та *error*;
- перевірка досягнення умов розв'язання задачі;
- передача результатів у канал *resChan*.

5. Синхронізація завершення обчислень

- очікування завершення всіх воркерів за допомогою *WaitGroup*;
- завершення передачі результатів.

6. Агрегація результатів (Master)

- обробка можливих помилок;
- оновлення значень придатності організмів;
- визначення найкращого організму покоління (*champion*);
- оновлення статистики покоління.

7. Завершення етапу оцінювання покоління

- фіксація факту розв'язання задачі (за наявності);
- збереження результатів та топології ШНМ;
- перехід до наступного еволюційного етапу.

3.4.3. Алгоритм - реалізація мовою програмування GO

Авторська реалізація алгоритму паралельної оцінки придатності у нейроеволюційному процесі, розроблена з використанням мови програмування GO, у вигляді псевдокоду наведена на рисунку 3.4.

На цьому рисунку наведено першу, керуючу частину алгоритму, що реалізує модель *Master–Worker* для паралельної оцінки придатності організмів у межах нейроеволюційного процесу. Даний етап відповідає за організацію та ініціалізацію паралельних обчислень і виступає центральним компонентом, який координує взаємодію між основним керуючим процесом (*master*) та множиною обчислювальних виконавців (*workers*).

На цьому етапі здійснюється декомпозиція загального обсягу обчислювальних робіт, пов'язаних з оцінюванням придатності всіх організмів поточного покоління, на окремі незалежні задачі. Кожна така задача відповідає процесу симуляції та оцінювання одного організму і може виконуватися автономно, без необхідності доступу до спільного змінного стану. *Завдяки цій властивості забезпечується можливість ефективного розпаралелювання обчислень і мінімізації взаємного блокування між потоками виконання.*

Алгоритм передбачає створення пулу воркерів фіксованого розміру, що визначається параметрами конфігурації експерименту та відповідає доступним обчислювальним ресурсам. Кожен воркер запускається у окремому легковажному потоці (*goroutine*) та перебуває у стані очікування надходження задач на оцінювання через спеціалізований канал обміну даними. Такий підхід дозволяє реалізувати динамічний та збалансований розподіл задач між воркерами, оскільки кожна задача обробляється першим вільним воркером, що *забезпечує рівномірне завантаження обчислювальних ресурсів.*

Algorithm 1: Паралельна оцінка придатності покоління (Master–Worker) для задачі балансування подвійного маятника

Input: *ctx* — контекст виконання з параметрами NEAT;
pop — популяція організмів поточного покоління;
epoch — структура статистики покоління;
markov — режим середовища;
actionType — тип керувальної дії;
p — кількість воркерів (goroutines)
Output: Оновлені *Fitness/Error* усіх організмів у *pop*;
Оновлені поля *epoch* (статистика, *Champion*, ознака розв’язання)

```
// (Master)
1 options ← FromContext(ctx)
2 if options не знайдено then
3   └ return ErrNEATOptionsNotFound
4 N ← |pop.Organisms|
5 resChan ← makeChannel(evaluationJobResult, buffer = N)
6 jobsChan ← makeChannel(evaluationJob, buffer = N)
7 WG ← WaitGroup()
8 mapping ← ∅ // map: genomeId → *Organism

// Створення пулу воркерів
9 for i ← 1 to p do
10  └ WG.Add(1)
11  └ spawn Worker(markov, actionType, jobsChan, resChan, WG)

// Публікація задач оцінювання для кожного організму
12 foreach org ∈ pop.Organisms do
13   └ if mapping вже містить ключ org.Genotype.Id then
14     └ return error("duplicate organism id")
15   └ mapping[org.Genotype.Id] ← org
16   └ jobsChan ← evaluationJob{organism : org}
17 close(jobsChan)
18 WG.Wait()
19 close(resChan)
```

Рис. 3.4. Алгоритм реалізації Master-Worker моделі для паралельної оцінки придатності мовою GO, частина 1 (авторська розробка)

Механізми обміну даними та конкурентної синхронізації реалізовано із застосуванням вбудованих примітивів мови програмування *GO*, зокрема каналів (*channels*) для асинхронної передачі задач та результатів обчислень, а також груп очікування (*WaitGroup*) для узгодженого завершення роботи пулу воркерів.

Канали забезпечують потокобезпечну передачу даних без необхідності явного використання механізмів блокування, тоді як *WaitGroup* гарантує коректну синхронізацію завершення всіх паралельних обчислень перед переходом алгоритму до наступного етапу — агрегації результатів та оновлення стану еволюційного процесу.

Застосування описаної архітектури дозволяє досягти високого рівня масштабованості та адаптивності алгоритму до різних обчислювальних середовищ, забезпечуючи ефективне використання паралелізму на етапі оцінювання придатності — найбільш обчислювально затратному етапі алгоритму NEAT.

На рисунку 3.5 продемонстровано другу частину алгоритму, що займається агрегацією отриманих результатів.

На етапі агрегації результатів оцінювання придатності, отриманих від пулу воркерів, здійснюється централізована обробка всіх проміжних результатів паралельних обчислень. Передусім виконується перевірка наявності помилок, що могли виникнути під час виконання симуляцій у воркерах; у разі виявлення критичних помилок процес оцінювання поточного покоління переривається з відповідним повідомленням, що забезпечує коректність подальшого перебігу еволюційного алгоритму.

Для кожного організму популяції на основі отриманих результатів оновлюються значення функції придатності (*fitness*) та функції втрат (*error*), які безпосередньо використовуються на наступних етапах еволюційного циклу, зокрема під час селекції, репродукції та мутації генотипів. Одночасно виконується ідентифікація організмів, що досягли умов розв'язання поставленої задачі моделювання, та визначається поточний найкращий індивід покоління (*Champion*) за критерієм максимальної придатності.

Окрім цього, на даному етапі формується та зберігається розширена статистика, що характеризує структурні властивості знайдених рішень, зокрема кількість вузлів, кількість генів зв'язків, глибину активації мережі та інші топологічні показники штучних нейронних мереж. Отримані статистичні дані

використовуються як для подальшого аналізу динаміки нейроеволюційного процесу, так і для порівняльної оцінки ефективності різних конфігурацій алгоритму та обчислювальних схем його реалізації.

```

// Агрегація результатів оцінювання
20 foreach  $r \in resChan$  do
21   if  $r.err \neq \emptyset$  then
22     return  $r.err$ 
23   if mapping не містить ключ  $r.genomeId$  then
24     return error("organism not found in mapping")
25    $org \leftarrow mapping[r.genomeId]$ 
26    $org.Fitness \leftarrow r.fitness$ 
27    $org.Error \leftarrow r.error$ 
28   if  $r.winner = true$  and
      ( $epoch.Champion = \emptyset \vee org.Fitness > epoch.Champion.Fitness$ ) then
29      $epoch.Solved \leftarrow true$ 
30      $epoch.WinnerNodes \leftarrow |org.Genotype.Nodes|$ 
31      $epoch.WinnerGenes \leftarrow org.Genotype.Extrons()$ 
32      $epoch.WinnerEvals \leftarrow options.PopSize \cdot epoch.Id + org.Genotype.Id$ 
33      $epoch.Champion \leftarrow org$ 
34      $org.IsWinner \leftarrow true$ 

```

Рис. 3.5. Алгоритм реалізації Master-Worker моделі для паралельної оцінки придатності мовою GO, частина 2 (авторська розробка)

На рисунку 3.6 наведено структурно-алгоритмічну схему функціонування індивідуального воркера в межах запропонованої моделі паралельної оцінки придатності. Воркери виконують роль незалежних обчислювальних одиниць, кожна з яких асинхронно отримує задачі на оцінювання придатності окремих організмів популяції через канал міжпроцесної взаємодії *jobsChan*. Отримана задача містить посилання на відповідний генотип організму, для якого необхідно здійснити повний цикл симуляції та обчислення значення цільової функції (fitness score).

Кожен воркер ініціалізує власне ізольоване середовище моделювання фізичного процесу балансування подвійного оберненого маятника, що забезпечує відсутність побічних ефектів між паралельними обчисленнями та виключає необхідність синхронізації станів симулятора між різними потоками

виконання. Після початкової ініціалізації дане середовище симуляції повторно використовується для послідовної обробки множини задач, що надходять до воркера, без повторного створення об'єктів середовища, що *суттєво знижує накладні витрати на керування пам'яттю та ініціалізацію ресурсів*.

Algorithm 2: Worker: обробка задач оцінювання придатності організмів

Input: *markov* — режим середовища;
actionType — тип керувальної дії;
jobsChan — канал задач;
resChan — канал результатів;
WG — лічильник завершення
Output: Передача в *resChan* результатів (genomeId, fitness, error, winner) або помилки

```

1 defer WG.Done()
  // Ініціалізація локального симулятора (ізоляція стану на воркер)
2 env ← NewCartPole(markov)
3 env.nonMarkovLong ← false
4 env.generalizationTest ← false
5 while job ← recv(jobsChan) поки канал не закритий do
6   (winner, err) ← OrganismEvaluate(job.organism, env, actionType)
7   if err ≠ ∅ then
8     send(resChan, evaluationJobResult{err : err})
9     return
10  r ← evaluationJobResult{
11    genomeId : job.organism.Genotype.Id,
12    fitness : job.organism.Fitness,
13    error : job.organism.Error,
14    winner : winner
15  }
16  send(resChan, r)

```

Рис. 3.6. Структурно-алгоритмічна схема функціонування індивідуального воркера для паралельної оцінки придатності організмів у модифікованій моделі Master-Worker (авторська розробка)

У процесі обробки кожної задачі воркер виконує повну симуляцію динаміки системи відповідно до поточного генотипу організму, визначає значення функції придатності (fitness) та функції втрат (loss), а також фіксує факт досягнення умов розв'язання задачі, якщо такі виконуються. Отримані результати інкапсулюються у відповідну структуру даних і асинхронно

передаються через канал *resChan* до головного керуючого процесу (master), який відповідає за агрегацію результатів та оновлення стану еволюційного алгоритму.

Застосування описаного підходу дозволяє досягти високого рівня завантаження обчислювальних ресурсів, мінімізувати прості воркерів та зменшити сумарний час виконання етапу оцінювання придатності, що є найбільш ресурсоємною складовою алгоритму NEAT. Це, у свою чергу, *забезпечує масштабованість запропонованої реалізації та її ефективність у середовищах з обмеженими або нерівномірно доступними обчислювальними ресурсами.*

3.4.4. Висновки до розділу

У даному розділі було детально розглянуто запропонований алгоритм паралельної оцінки придатності організмів у процесі нейроеволюції, побудований на основі модифікованої моделі Master–Worker та реалізований з використанням механізмів конкурентного програмування мови GO. Алгоритм орієнтований на усунення одного з ключових обмежень класичного NEAT — високих часових витрат на етапі оцінки придатності, що зростають пропорційно розміру популяції та складності симуляційного середовища.

Показано, що розділення процесу еволюції на централізовану координаційну частину (Master) та незалежні обчислювальні компоненти (Worker) дозволяє ефективно розпаралелювати найбільш ресурсоємну частину алгоритму без внесення змін у генетичні механізми NEAT. Застосування каналів обміну даними (channels) та груп очікування (WaitGroup) забезпечує коректну синхронізацію паралельних процесів, детерміновану агрегацію результатів та стійкість алгоритму до помилок виконання окремих задач.

Особливістю запропонованого підходу є *багаторазове використання середовища симуляції в межах кожного воркера, що зменшує накладні витрати на ініціалізацію та сприяє більш ефективному використанню обчислювальних ресурсів.* Алгоритм забезпечує масштабованість за кількістю воркерів та

адаптивність до апаратних обмежень, що є критично важливим для практичного застосування нейроеволюційних методів у складних задачах керування.

Таким чином, запропонований алгоритм створює формальну та практичну основу для подальшого аналізу ефективності паралельної оцінки придатності, експериментальної перевірки досягнутого прискорення та обґрунтування доцільності його використання в умовах обмежених обчислювальних ресурсів, що розглядається у наступних розділах роботи.

3.5. Опис та аналіз результатів експериментальних досліджень

У даному розділі наведено результати експериментальних досліджень, спрямованих на емпіричну перевірку ефективності запропонованого методу підвищення ефективності нейроеволюційного алгоритму NEAT за рахунок паралельної оцінки придатності організмів. Експерименти проводилися з метою підтвердження теоретичних положень, сформульованих у попередніх підрозділах, зокрема аналітичної моделі часових витрат та моделі оцінки ефективності реалізації алгоритму.

Основна увага у цьому розділі приділяється аналізу залежності часу еволюції від кількості паралельних воркерів, оцінці досягнутого прискорення та ефективності використання обчислювальних ресурсів, а також порівнянню однопоточної та паралельної схем оцінки придатності. Як тестову задачу було обрано задачу балансування подвійного оберненого маятника, що характеризується високою обчислювальною складністю симуляційного середовища та є репрезентативною для класу задач керування з безперервною динамікою.

Отримані експериментальні дані аналізуються з позицій відповідності теоретичним прогнозам, виявлення практично оптимальної кількості воркерів та оцінки впливу накладних витрат на масштабованість запропонованого підходу. Результати подано у вигляді таблиць, графіків та узагальнених показників, що дозволяє зробити обґрунтовані висновки щодо доцільності використання запропонованої моделі Master–Worker та розроблених методів оцінювання ефективності в умовах обмежених обчислювальних ресурсів.

3.5.1. Параметри моделі часових витрат

У дослідженні використовувалися такі сталі параметри моделі часових витрат які наведено у таблиці 3.2. Ця модель використовується для оцінювання теоретичних меж продуктивності методу та для побудови Pareto-фронту залежності між кількістю воркерів та часом виконання.

Таблиця 3.2

Задані параметри моделі часових витрат для задачі балансування візка з двома оберненими маятниками.

Позначення	Параметр	Значення	Одиниці	Опис
G	Кількість поколінь	100	безрозмірна	Загальна кількість еволюційних циклів, що виконуються в експерименті.
$\bar{W}$	Середня кількість вагових операцій у мережі	43.4	безрозмірна	Середня кількість обчислень ваги зв'язків, необхідних для оцінки одного організму.
N	Розмір популяції	1 000	безрозмірна	Кількість особин (організмів), які оцінюються в кожному поколінні.
$\bar{V}$	Середня кількість вузлів	20.5	безрозмірна	Середня кількість нейронних вузлів, що беруть участь в обчисленнях розв'язувача.
L	Кількість кроків симуляції	100 000	безрозмірна	Тривалість симуляції задачі балансування маятника у тиках середовища.

Інші параметри теоретичної моделі часових витрат визначаються емпіричним шляхом за рахунок серії калібрувальних прогонів, що буде описано далі.

3.5.2. Залежність часу еволюції від кількості воркерів

Одним із ключових чинників, що визначають ефективність паралельної реалізації алгоритму NEAT, є кількість воркерів p , задіяних на етапі оцінки придатності організмів популяції. Оскільки саме цей етап є найбільш ресурсомістким з точки зору обчислювальних витрат, вибір раціонального

значення p безпосередньо впливає на загальний час виконання еволюційного процесу.

Для теоретичного аналізу цієї залежності в роботі використано запропоновану модель часових витрат, згідно з якою загальний час виконання еволюції може бути апроксимований наступним виразом (див. формулу 2.10):

$$T_{total} \approx G \left(\frac{N\gamma L\bar{V}}{p} + NC_{comm} + \alpha N \log N + \beta N(\bar{W} + \bar{V}) + \delta \frac{p}{p+p_{sat}} \right),$$

де p – кількість паралельно працюючих воркерів, p_{sat} – параметр сатурації паралелізму; безрозмірна величина, що визначає характерну кількість воркерів, після якої подальше збільшення p майже не зменшує час виконання.

З наведеного виразу видно, що лише перший доданок є обернено пропорційним до p . Він відповідає безпосередньо за паралелізовану частину алгоритму — виконання симуляції середовища балансування подвійного оберненого маятника для кожного організму популяції. Останній доданок, що також залежить від p , моделює гладкий перехід до плато продуктивності при досягненні межі масштабованості.

Таким чином, основне обчислювальне навантаження масштабується як $O(NL\bar{V})$ і добре піддається розпаралелюванню. Інші складові моделі не залежать від p або залежать від нього опосередковано:

- NC_{comm} – описує витрати на передачу завдань і результатів між Master та Worker і є лінійною функцією від розміру популяції;
- $\alpha N \log N$ – враховує логарифмічні накладні витрати, пов'язані з операціями сортування, кластеризації за видами та нормалізації;
- $\beta N(\bar{W} + \bar{V})$ – описує лінійні накладні витрати, зумовлені обробкою топологічної структури мереж для створення наступних поколінь, тобто операції кросинговеру та мутації.

Зі зростанням кількості воркерів p час виконання паралелізованої частини алгоритму зменшується пропорційно $1/p$. Проте при досягненні певного значення p подальше збільшення кількості воркерів перестав давати суттєвий виграш у часі виконання, оскільки домінуючими стають непаралелізовані

складові моделі, що виконуються в одному потоці керування нейроеволюційним процесом. Окрім того, подальше збільшення кількості воркерів може призвести до погіршення часу виконання, за рахунок збільшення накладних витрат пов'язаних з масштабуванням та синхронізацією у конкурентному середовищі.

Це означає, що існує область раціональних значень p , для яких:

$$\frac{N\gamma L\bar{V}}{p} \approx NC_{\text{comm}} + \alpha N \log N + \beta N(\bar{W} + \bar{V}) + \delta \frac{p}{p+p_{\text{sat}}}, \quad (3.22)$$

Тобто час виконання паралельної частини стає порівняним з накладними витратами.

3.5.3. Знаходження області раціональної кількості воркерів

Для вибору оптимальної кількості воркерів позначимо складові частини рівняння (35) як:

$$A = N\gamma L\bar{V}, \quad B = NC_{\text{comm}} + \alpha N \log N + \beta N(\bar{W} + \bar{V}), \quad (3.23)$$

Тоді, загальний час еволюції як функція від кількості воркерів може бути виражений наступним чином:

$$T_{\text{total}}(p) = G \left(\frac{A}{p} + B + \delta \frac{p}{p+p_{\text{sat}}} \right). \quad (3.24)$$

Члени цього рівняння мають наступну дію на модель:

- член $\frac{A}{p}$ – сприяє зменшенню часу при зростанні p ;
- δ, p_{sat} – описують межу ефективного паралелізму (насичення).

Оптимальна кількість воркерів p^* визначається балансом між цими двома ефектами та аналітично визначається як мінімум функції $T_{\text{total}}(p)$:

$$p^* = \arg \min_p T_{\text{total}}(p). \quad (3.25)$$

Знаходження цього мінімуму здійснюється шляхом диференціювання за змінною p :

$$\frac{dT_{\text{total}}}{dp} = G \frac{d}{dp} \left(\frac{A}{p} + B + \delta \frac{p}{p+p_{\text{sat}}} \right) = G \left(-\frac{A}{p^2} + \delta \frac{p_{\text{sat}}}{(p+p_{\text{sat}})^2} \right). \quad (3.26)$$

Таким чином, мінімум досягається, коли:

$$-\frac{A}{p^2} + \delta \frac{p_{\text{sat}}}{(p+p_{\text{sat}})^2} = 0, \quad (3.27)$$

Звідки отримуємо аналітичний вираз для оптимальної кількості воркерів:

$$\delta p_{\text{sat}} p^2 = A(p + p_{\text{sat}})^2. \quad (3.28)$$

$$p^* = \frac{p_{\text{sat}}}{\sqrt{\frac{\delta p_{\text{sat}}}{A} - 1}}. \quad (3.29)$$

Отримане значення p^* відповідає балансу між виграшем від паралелізації обчислень та зростанням накладних витрат, пов'язаних із конкурентною синхронізацією та керуванням виконанням паралельних процесів.

У цьому виразі (3.29) параметри δ та p_{sat} є емпіричними та залежать від:

- архітектури апаратної платформи,
- реалізації конкурентних примітивів мови GO,
- структури обміну даними між компонентами Master–Worker,
- гранулярності задач оцінювання придатності.

Калібрувальні прогони. Дана робота *передбачала виконання короткої серії калібрувальних експериментів*, за результатами яких здійснюється регресійна оцінка параметрів A , B , δ та p_{sat} . Отримані значення використовуються для аналітичного визначення p^* та області раціональних значень p , що можуть бути використанні для подальшого планування повномасштабних експериментальних досліджень.

У межах експериментальних досліджень було виконано серію запусків нейроеволюційного алгоритму NEAT для задачі балансування подвійного оберненого маятника з використанням паралельної моделі оцінки придатності за схемою *Master–Worker*.

Для аналізу, у даній роботі, було розглянуто чотири варіанти розміру популяції $N \in \{500, 1000, 2000, 3000\}$ для калібрувальних прогонів. Для кожного значення N експериментально вимірювався загальний час виконання одного покоління еволюції при різній кількості воркерів. Усі часові вимірювання

були приведені до секунд та використані для побудови емпіричних залежностей $T_{\text{exp}}(p)$.

Отримані експериментальні данні (Додаток А) дозволили створити регресивну модель на основі формули (3.24) та емпірично визначити параметри часової моделі, що не було задано: A , B , δ та p_{sat} .

Тобто, для кожного фіксованого значення N було:

1. виконано регресійне наближення параметрів моделі;
2. знайдено експериментальне оптимальне значення p_{exp}^* ;
3. аналітично визначено оптимум моделі p_{model}^* шляхом мінімізації $T_{\text{total}}(p)$.

Результати зіставлення експериментальних даних та аналітичної моделі подано на рисунку 3.7, де для кожного значення N наведено:

- експериментальні залежності $T_{\text{exp}}(p)$;
- відповідні регресійні криві моделі;
- експериментально спостережувані оптимальні значення p_{exp}^* ;
- аналітично обчислені значення p_{model}^* .

Експериментальне оптимальне значення кількості воркерів p_{exp}^* визначалось не як мінімум серед дискретно виміряних точок, а шляхом локальної екстраполяції. Для цього у околі найменшого спостережуваного значення часу $T(p)$ формувалось локальне вікно з k сусідніх точок, на яких методом найменших квадратів будувалась квадратична апроксимація $T(p) = ap^2 + bp + c$. Значення p_{exp}^* знаходилось як вершина параболи $p_{\text{exp}}^* = -\frac{b}{2a}$ з додатковим обмеженням на належність до інтервалу локального вікна, що зменшує залежність результату від кроку дискретизації експерименту. Перевагою цього підходу є його висока чутливість до локальної кривизни функції та здатність точно виявляти точку мінімальної обчислювальної вартості. Водночас результат істотно залежить від вибору локального інтервалу апроксимації та може бути чутливим до шуму експериментальних вимірювань.

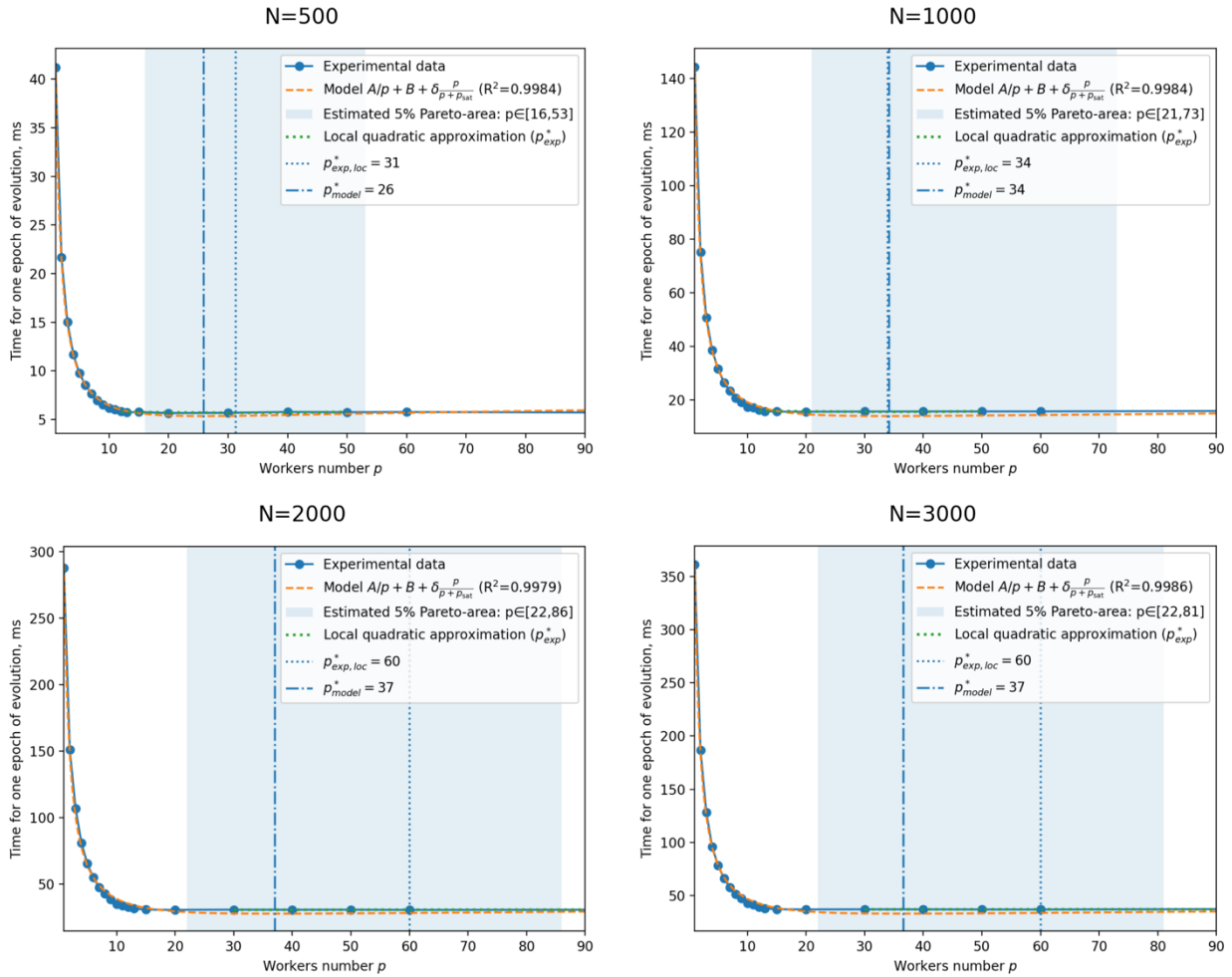


Рис. 3.7. Залежність часу від кількості воркерів при різних розмірах популяції. Визначення аналітичного та експериментального оптимуму кількості воркерів, а також області раціональних значень (5% Pareto) (створено автором)

Розбіжність між теоретичними оцінками оптимальної кількості воркерів p_{model}^* та експериментальними p_{exp}^* слугує індикатором наявності інтервалу *практично оптимальних значень кількості воркерів*. Це підтверджує доцільність використання області Парето-оптимальних значень параметра p замість єдиного точкового оптимуму.

Дані отримані в результаті експериментальних вимірювань наведені у таблиці 3.3.

За результатами параметричної ідентифікації узагальненої моделі часових витрат паралельної оцінки придатності організмів встановлено, що для більшості досліджених конфігурацій розміру популяції параметр B , який відповідає

постійній складовій часу виконання одного покоління, набуває надзвичайно малих значень або прямує до нуля. Це свідчить про відсутність істотних серіалізованих накладних витрат, незалежних від кількості воркерів, у досліджуваній реалізації алгоритму NEAT.

Таблиця 3.3

Результати ідентифікації параметрів моделі часових витрат та оцінки оптимальної кількості воркерів.

N	p_{exp}^*	p_{model}^*	R^2	A	B	δ	p_{sat}
500	31	26	0.9983	40.529	0.595	6.251	24.927
1000	34	34	0.9984	144.035	$6.09 \cdot 10^{-18}$	17.161	26.371
2000	60	37	0.9979	288.216	1.338	32.043	26.394
3000	60	37	0.9986	360.789	$1.44 \cdot 10^{-14}$	40.330	27.011

Значні варіації значень параметра B , отриманих у результаті регресійного аналізу для різних розмірів популяції, є очікуваними та не свідчать про некоректність моделі. Параметр B у запропонованій аналітичній моделі виконує роль ефективного адитивного зсуву та не відповідає окремому фізичному етапу обчислювального процесу. Через сильну кореляцію з іншими параметрами моделі (δ , p_{sat}) та домінування паралелізованих і насичувальних складових, значення B є слабо ідентифікованим і може суттєво змінюватися без впливу на точність апроксимації або положення оптимуму p^* . Таким чином, інтерпретація параметра B має розглядатися виключно в контексті чисельної підгонки, а не як окрема часова характеристика алгоритму.

З урахуванням графіків на рисунку 3.7 та наведених числових параметрів можна зробити такі узагальнення:

1. За малих значень p (<15) домінує обчислювальне навантаження симуляції, і збільшення p призводить до майже експоненціального прискорення.
2. За середніх значень p (15-80) відбувається поступове насичення ефекту паралелізму, оскільки частка накладних витрат зростає.

3. За великих значень p (>100) приріст продуктивності стає незначним або зникає повністю та система виходить на плато, що робить подальше масштабування нераціональним.

Таким чином, *оптимальна кількість воркерів визначається балансом між обсягом паралельної роботи та сумарними накладними витратами, які не можуть бути усунуті за рахунок збільшення ступеня паралелізму.*

Отримані результати підтверджують фізичну узгодженість запропонованої моделі часових витрат та її здатність адекватно відображати зміну режимів продуктивності паралельної реалізації алгоритму NEAT залежно від масштабу задачі. Це створює підґрунтя для використання даної моделі як інструменту аналітичного вибору раціональної кількості воркерів перед проведенням експериментальних досліджень, що дозволяє зменшити кількість емпіричних налаштувань та підвищити ефективність використання обчислювальних ресурсів.

Парето-оптимальна область. Для вирішення практичних задач має сенс ще більше опимізувати кількість воркерів на основі розрахунку Парето-оптимальної області в межах 5% від теоретичного мінімуму функції $T(p)$. Це можна розрахувати наступним чином.

Нехай:

$$T_{min} = \min_{1 \leq p} T(p). \quad (3.30)$$

Тоді 5%-ва Парето-оптимальна область визначається як множина точок:

$$\mathcal{P}_{5\%} = \{p | T(p) \leq 1.05 \cdot T_{min}\}. \quad (3.31)$$

Отримані розрахункові результати також можуть бути підтверджені наочно, побудовою графіку функції $T(p)$ для різних значень кількості воркерів, при незмінності інших параметрів моделі часових витрат наведених у таблицях 3.1 і 3.2. Графік зображено на рисунку 3.8.

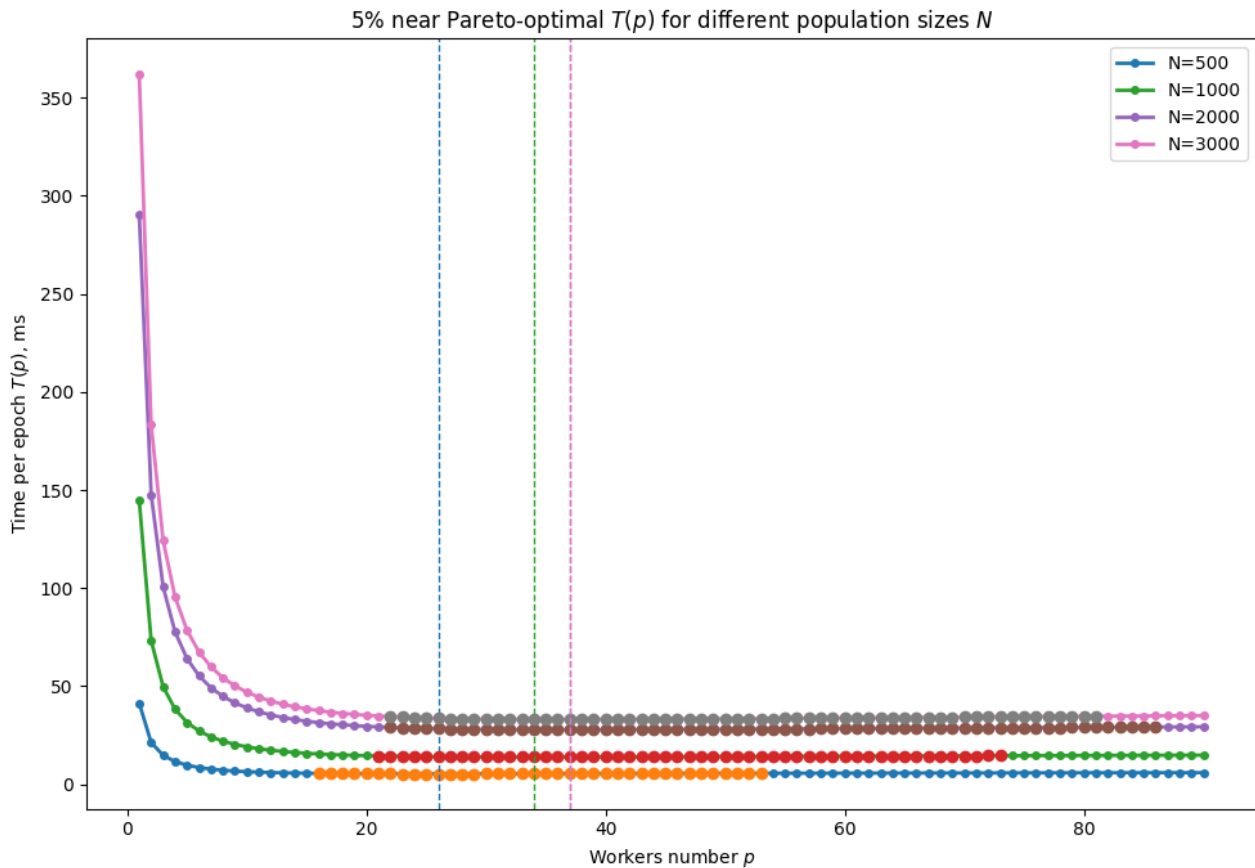


Рис. 3.8. Залежність часу виконання однієї епохи $T(p)$ від кількості воркерів p для різних розмірів популяції N , побудована за параметризованою моделлю. Вертикальні пунктирні лінії позначають модельні оптимальні значення p^* , а маркерами виділено 5%-ву Парето-оптимальну область відносно мінімального значення часу для кожного N (створено автором)

Вплив розміру популяції. В межах експериментальних досліджень важливим також є аналіз впливу розміру популяції на вибір оптимальної області значень кількості воркерів з позицій Парето-оптимальності. З огляду на те, що розмір популяції безпосередньо визначає кількість незалежних задач оцінювання придатності в кожному поколінні, його зміна істотно впливає на характер масштабування паралельної реалізації алгоритму NEAT та ефективність використання обчислювальних ресурсів.

З цією метою було побудовано тривимірну поверхню залежності загального часу еволюції від трьох параметрів: розміру популяції N , кількості

паралельних воркерів p та часу виконання нейроеволюційного процесу T_{total} . На цій поверхні додатково виділено область Парето-оптимальних рішень, що відповідає значенням часу, які не перевищують 5% від строгого теоретичного мінімуму функції $T_{total}(N, p)$ для заданого розміру популяції, що розраховується як значення функції наступного вигляду:

$$\mathcal{P}_{5\%}(N) = \min\{p: T(N, p) \leq 1.05 \cdot T_{min}(N)\}, \quad (3.32)$$

де $T_{min}(N) = \min_{1 \leq p \leq N} T(N, p)$ – це мінімальне значення функції часу для заданого розміру популяції.

Такий підхід дозволяє ідентифікувати не окрему оптимальну точку, а цілу зону раціональних параметрів, у межах якої забезпечується майже мінімальний час виконання.

Аналіз графіка, поданого на рисунку 3.9, демонструє, що у випадку малих значень кількості воркерів навіть незначне збільшення розміру популяції призводить до різкого зростання загального часу еволюції. Це пояснюється домінуванням послідовної складової оцінювання придатності, коли обчислювальні ресурси не встигають компенсувати зростаючий обсяг розрахунків симуляції.

Водночас у області значень p , близьких до теоретично оптимальної кількості воркерів, визначеної на основі моделі часових витрат, спостерігається суттєво інша поведінка системи. Збільшення розміру популяції в цій області призводить лише до незначного зростання загального часу еволюції або майже не впливає на нього. Це свідчить про те, *що при достатньому рівні паралелізму основна обчислювальна складність ефективно розподіляється між воркерами, а накладні витрати стають домінуючим, але відносно стабільним фактором.*

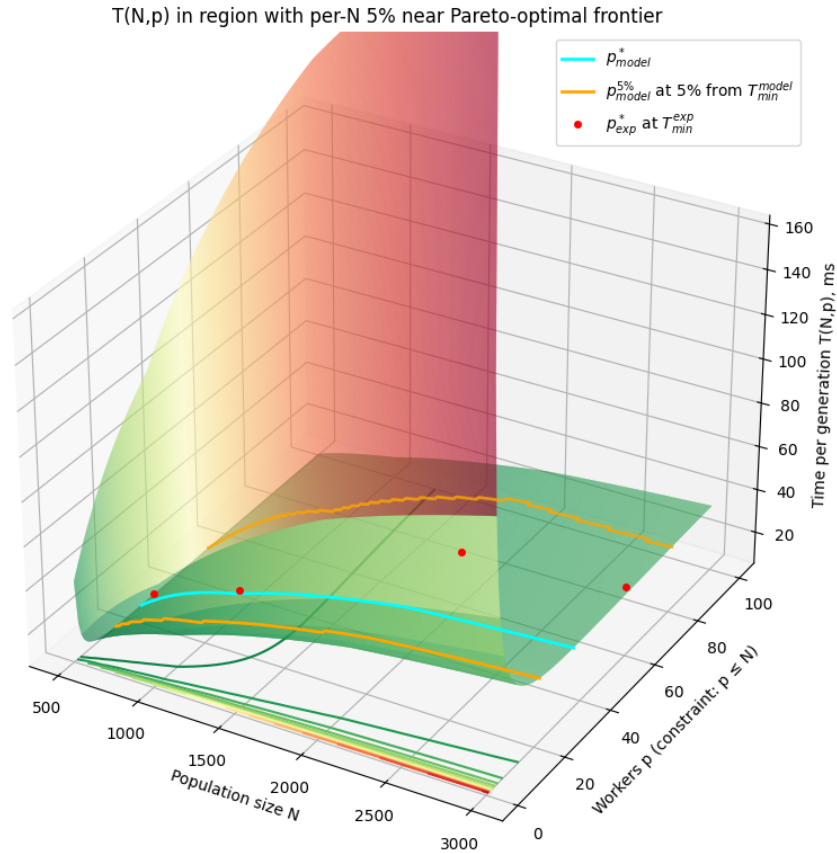


Рис. 3.9. Тривимірна залежність загального часу еволюції T_{total} від розміру популяції N та кількості воркерів p з виділеною областю Парето-оптимальних значень $p_{\text{model}}^{5\%}$ (при 5% від мінімуму часу на розмір популяції $T_{\text{min}}^{\text{model}}(N)$) (створено автором)

У таблиці 3.4 наведено 5%-ві Парето-оптимальні області для різних розмірів популяції N , тобто діапазони кількості воркерів p , за яких час виконання однієї епохи $T(N, p)$ не перевищує $1.05 \cdot T_{\text{min}}^{\text{model}}(N)$. Для кожного N визначено модельний мінімум $T_{\text{min}}^{\text{model}}(N)$, відповідне оптимальне значення p_{model}^* та інтервал $[p_{\text{min}}^{5\%}, p_{\text{max}}^{5\%}]$, що характеризує зону стабільних, близьких до оптимуму конфігурацій.

Таблиця 3.4

Розрахункові параметри моделі та 5%-ва Парето-оптимальна область у термінах цілочисельних значень p , які задовольняють умовам формул (43, 44).

N	p_{model}^*	$T_{min}^{model}(N)$	Поріг 1.05 $\cdot T_{min}^{model}(N)$	5% область p $\in [p_{min}^{5\%}, p_{max}^{5\%}]$	К-сть значень p
500	26	5.36	5.63	[16;53]	38
1000	34	13.93	14.63	[21;73]	53
2000	37	27.85	29.24	[22;86]	65
3000	37	33.09	37.74	[22;81]	60

Висновки. Отримані результати свідчать, що розрахункова та експериментальна оптимальна кількість воркерів зростає повільно зі збільшенням розміру популяції та лежить у відносно вузькому діапазоні. Водночас 5%-ва Парето-оптимальна область є достатньо широкою, що підтверджує стабільність продуктивності: вибір p у межах цієї області практично не впливає на загальний час виконання. Це дозволяє гнучко налаштовувати кількість воркерів з урахуванням доступних обчислювальних ресурсів без суттєвої втрати ефективності.

Проведений аналіз експериментальних результатів у поєднанні з аналітичною моделлю часових витрат доводить практичну придатність запропонованого підходу до оптимізації паралельної оцінки придатності в алгоритмі NEAT та обґрунтовує його використання в умовах обмежених обчислювальних ресурсів. Запропонована модель повинна застосовуватися як *інструмент попередньої оцінки та оптимізації кількості воркерів у зоні досягнення мінімуму часових витрат*, але не як універсальна модель для опису асимптотичної поведінки системи при надмірному паралелізмі.

3.5.4. Прогноз оптимальної кількості воркерів для збільшеного розміру популяції

На основі ідентифікованих параметрів узагальненої моделі часових витрат паралельної оцінки придатності організмів (таблиця 3.2) була розглянута можливість прогнозування оптимальної кількості воркерів p^* для нових конфігурацій експерименту, в тому числі тих, які не входили до множини безпосередньо вимірюваних значень. Зокрема, проаналізовано випадок збільшення розміру популяції до $N = 5000$ особин.

На рисунку 3.10 наведено зіставлення експериментально вимірюваної залежності часу виконання однієї епохи еволюції $T(p)$ від кількості воркерів p для популяції розміру $N \in [3000, 5000]$ з прогнозною кривою, побудованою на основі параметрів моделі, відкаліброваних за результатами перших трьох експериментів для $N = \{500, 1000, 2000\}$. Заштрихована область відповідає 5%-вій Pareto-оптимальній зоні, визначеній за прогнозною моделлю. Результати демонструють узгодженість експериментальних даних з теоретичним прогнозом у межах допустимої похибки та підтверджують придатність моделі для попереднього визначення раціональної кількості воркерів.

Прогнозна крива, побудована шляхом лінійної екстраполяції параметрів моделі часових витрат, що наведені у таблиці 3.2. Отримана прогнозна крива коректно відтворює:

- загальний спадний тренд;
- порядок величин часу епохи;
- наявність зони насичення при зростанні p .

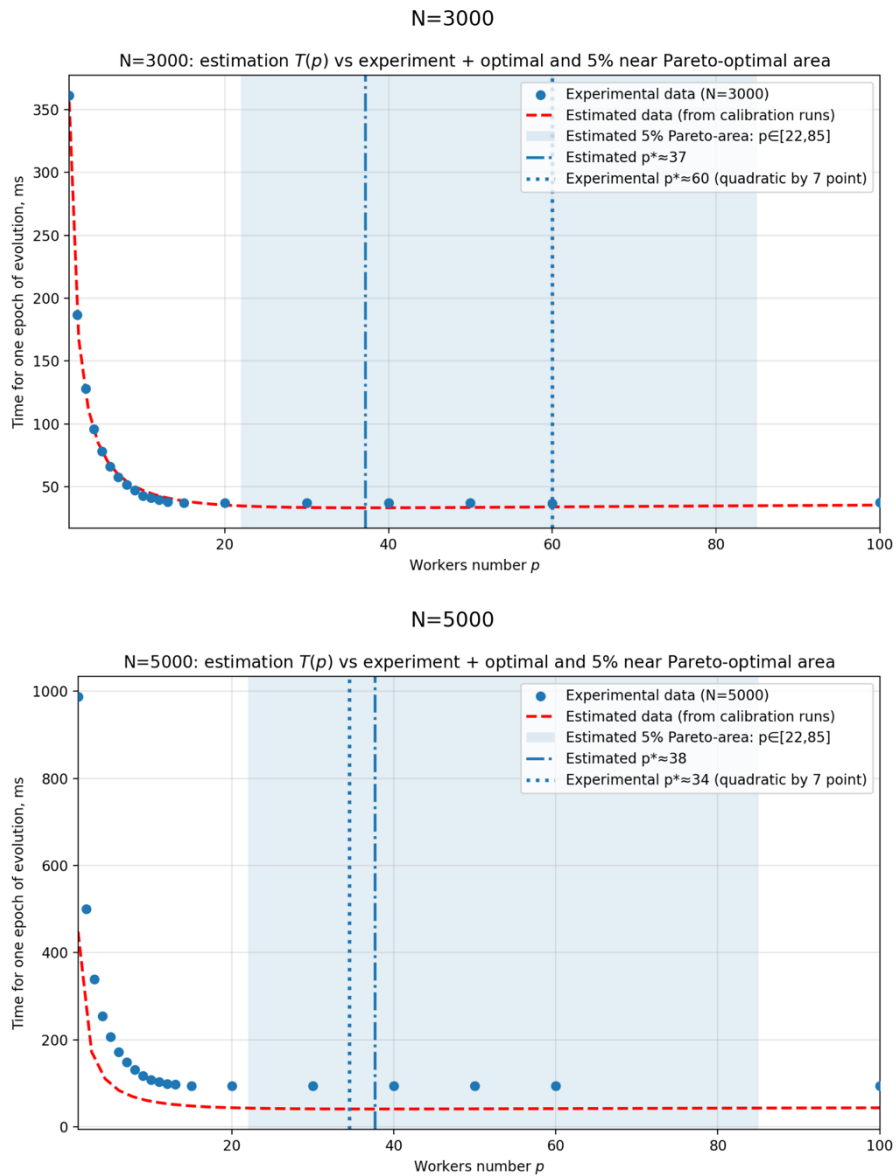


Рис. 3.10. Порівняння прогностної моделі та експериментальних результатів залежності часу виконання одного покоління нейроеволюційного алгоритму NEAT від кількості воркерів p для популяцій $N \in [3000, 5000]$ (створено автором)

Таким чином, можна зробити висновок, що запропонована модель забезпечує коректну оцінку області оптимальних значень кількості воркерів. Незважаючи на відхилення точкового прогнозу оптимального значення p^* , експериментально знайдений оптимум стабільно належить до 5%-вої Pareto-

оптимальної області, що підтверджує придатність моделі для попереднього аналітичного налаштування параметрів паралельної нейроеволюції.

3.5.5. Аналіз похибок прогнозової моделі часових витрат

У даному підрозділі здійснено кількісний та якісний аналіз похибок прогнозування при використанні моделі оцінювання часу виконання одного покоління нейроеволюційного алгоритму NEAT залежно від кількості воркерів p та розміру популяції N . Метою аналізу є перевірка адекватності запропонованої моделі для практичного використання під час попереднього налаштування параметрів паралельної нейроеволюції.

Для оцінювання точності прогнозу використано такі показники:

- **Абсолютна похибка оптимальної кількості воркерів**

$$\Delta p^* = |p_{\text{exp}}^* - p_{\text{model}}^*|, \quad (3.33)$$

де p_{exp}^* — експериментально визначене оптимальне значення,

p_{model}^* — оптимум, отриманий з прогнозової моделі.

- **Відносна похибка оптимальної кількості воркерів**

$$\varepsilon_{p^*} = \frac{|p_{\text{exp}}^* - p_{\text{model}}^*|}{p_{\text{exp}}^*} \cdot 100\%. \quad (3.34)$$

- **Відносна похибка прогнозу мінімального часу виконання**

$$\varepsilon_T = \frac{|T_{\min}^{\text{exp}} - T_{\min}^{\text{model}}|}{T_{\min}^{\text{exp}}} \cdot 100\%, \quad (3.35)$$

де $T_{\min}^{\text{exp}}$ та $T_{\min}^{\text{model}}$ — відповідно експериментально виміряне та прогнозоване мінімальні значення часу виконання.

У таблиці 3.5 наведено порівняльні результати між експериментально визначеною оптимальною кількістю воркерів p_{exp}^* та теоретично прогнозованою кількістю p_{model}^* , отриманою на основі запропонованої моделі часових витрат нейроеволюційного процесу. Аналіз виконано для різних значень розміру популяції N , що дозволяє оцінити *масштабованість моделі* та її *прогностичну здатність*.

Аналіз результатів прогнозування оптимальної кількості воркерів для популяцій розміром $N=3000$ та $N=5000$ показав, що запропонована аналітична модель демонструє різний рівень точності залежно від режиму масштабування системи. Для популяції $N=3000$ спостерігається суттєве відхилення між експериментальним та модельним значенням оптимальної кількості воркерів ($\epsilon_p^* = 38.1\%$), тоді як для $N=5000$ відповідна похибка зменшується до 9.3% . Це свідчить про немонотонний характер зміни оптимуму та вплив апаратних обмежень при переході до більших масштабів паралелізму.

Таблиця 3.5

Похибки прогнозування оптимальної кількості воркерів та мінімального часу виконання для $N \in [3000, 5000]$.

N	p_{exp}^*	p_{model}^*	Δp^*	ϵ_p^* , %	T_{min}^{exp} , мс	T_{min}^{model} , мс	ΔT , мс	ϵ_T , %	Потрапляння в 5% Pareto
3000	60	37	23	38.1	36.96	33.28	3.68	10.0	так
5000	34	38	4	9.3	93.24	41.27	51.97	55.7	так

Водночас для обох випадків модель систематично занижує мінімальний час виконання еволюційного процесу, що особливо помітно для популяції $N=5000$, де відносна похибка за часом перевищує 50% . Така поведінка вказує на наявність додаткових накладних витрат, пов'язаних із сатурацією обчислювальних ресурсів, які не повністю враховуються базовою аналітичною моделлю.

Виявлені похибки мають системний характер і пояснюються такими чинниками:

- Ефекти апаратного насичення.**

За великих значень p продуктивність обмежується кількістю фізичних ядер, пропускною здатністю кеш-пам'яті та планувальником операційної системи, що не враховується в аналітичній моделі явно.

- Нелінійні накладні витрати керування паралелізмом.**

Витрати на синхронізацію, контекстні перемикання та роботу з каналами зростають нерівномірно зі збільшенням кількості воркерів, що призводить до зсуву експериментального оптимуму.

3. Плоский характер мінімуму $T(p)$.

У зоні оптимальних значень функція часу має пологі форму, внаслідок чого навіть значні відхилення p від точкового оптимуму призводять до незначних змін часу виконання.

Важливим результатом є те, що в усіх експериментах експериментально знайдені оптимальні значення p_{exp}^* належать до 5%-вої Pareto-оптимальної області, визначеної за прогнозною моделлю (формула 3.31):

$$\mathcal{P}_{5\%} = \{p | T(p) \leq 1.05 \cdot T_{\min}\}.$$

Це свідчить про те, що навіть за наявності похибки у визначенні точкового оптимуму, модель коректно локалізує область ефективних значень кількості воркерів. Проведений аналіз показав, що прогнозна модель часових витрат є достатньо точною для практичного використання на етапі попереднього планування експериментів. Хоча точковий прогноз оптимальної кількості воркерів має обмежену точність, *модель надійно визначає Pareto-оптимальну область*, у межах якої забезпечується майже мінімальний час виконання. Це підтверджує доцільність використання запропонованого підходу для раціонального вибору параметрів паралельної нейроеволюції в умовах обмежених обчислювальних ресурсів.

Отримані результати підтверджують, що запропонована модель:

- є *достатньо точною* для аналітичного вибору кількості воркерів до запуску експерименту;
- забезпечує *гарантоване потрапляння в область квазі-оптимальних рішень* (Pareto-optimal область раціональних рішень);
- дозволяє *істотно зменшити простір перебору параметрів*, що є критично важливим в умовах обмежених обчислювальних ресурсів.

Таким чином, модель часових затрат досягає своєї основної мети — *підтримки інженерного та наукового прийняття рішень* щодо ефективної

паралелізації нейроеволюційного алгоритму NEAT, навіть за наявності неминучої системної нелінійності у великомасштабних експериментах.

3.5.6. Аналіз прискорення нейроеволюційного процесу при застосуванні запропонованого методу покращення ефективності алгоритму NEAT

У даному розділі наведено аналіз досягнутого прискорення нейроеволюційного процесу алгоритму NEAT за рахунок застосування паралельної оцінки придатності організмів із використанням оптимальної кількості воркерів. Аналіз виконується шляхом порівняння часу виконання одного покоління в однопоточному режимі (один воркер) та у паралельному режимі.

Для кількісної оцінки виграшу від паралелізації використовуються дві метрики паралельних обчислень:

1. Прискорення

$$A(p) = \frac{T(1)}{T(p)}, \quad (3.36)$$

де $T(1)$ — час виконання одного покоління нейроеволюції при використанні одного воркера (послідовна схема), $T(p)$ — час виконання при використанні p воркерів.

2. Ефективність використання воркерів

$$E(p) = \frac{A(p)}{p}. \quad (3.37)$$

Ця величина характеризує, наскільки повно задіяні обчислювальні ресурси, та дозволяє оцінити втрати ефективності зі зростанням рівня паралелізму.

Аналіз проведено для всіх доступних експериментальних наборів даних з розмірами популяції $N \in [500, 1000, 2000, 3000, 5000]$ та наведено у таблиці

3.6. Для кожного N було визначено:

- експериментальний оптимум кількості воркерів p_{exp}^* ,
- відповідне мінімальне значення часу виконання $T(p_{exp}^*)$,

- прискорення $A(p_{exp}^*)$,
- ефективність $E(p_{exp}^*)$.

Таблиця 3.6

Розрахункові значення прискорення $A(p_{exp}^*)$ та ефективності використання воркерів $E(p_{exp}^*)$ на основі отриманих експериментальних даних.

N	p_{exp}^*	$T(1)$, ms	$T(p_{exp}^*)$, ms	$A(p_{exp}^*)$	$E(p_{exp}^*)$
500	31	41.17	5.69	7.24	0.23
1000	34	144.28	15.52	9.29	0.27
2000	60	287.55	30.64	9.39	0.16
3000	60	361.07	36.96	9.77	0.16
5000	34	986.94	93.25	10.58	0.31

Отримані експериментальні результати демонструють істотне прискорення нейроеволюційного процесу при переході від однопоточної схеми виконання до паралельної оцінки придатності з оптимально підбраною кількістю воркерів p_{exp}^* . Для всіх досліджуваних розмірів популяції спостерігається зростання коефіцієнта прискорення $A(p_{exp}^*)$ який набуває значень у діапазоні від 7.24 до 10.58.

Аналіз показує, що зі збільшенням розміру популяції N абсолютне прискорення загалом зростає або стабілізується на рівні $A(p_{exp}^*) \approx 9 - 10$. Зокрема, для $N = 500$ прискорення становить $A = 7.24$, тоді як для більших популяцій ($N \geq 1000$) воно зростає до значень $9.29 - 10.58$. *Це свідчить про те, що паралельна модель Master–Worker є найбільш ефективною саме для обчислювально інтенсивних режимів, у яких час симуляції одного організму домінує над накладними витратами.*

Водночас коефіцієнт ефективності використання воркерів $E(p_{exp}^*)$ залишається істотно меншим за одиницю для всіх експериментів і знаходиться в межах $0.16 - 0.31$. Така поведінка є типовою для паралельних систем і

пояснюється впливом синхронізації, комунікаційних накладних витрат, а також ефектами сатурації апаратних ресурсів. Найменші значення ефективності ($E \approx 0.16$) спостерігаються для популяцій $N = 2000$ та $N = 3000$, де оптимальна кількість воркерів є великою ($p_{\text{exp}}^* = 60$), що призводить до зменшення граничної користі від додаткового паралелізму.

Варто відзначити, що для популяції $N = 5000$ зафіксовано найбільше абсолютне прискорення ($A = 10.58$) і водночас відносно високу ефективність ($E = 0.31$) при меншій оптимальній кількості воркерів ($p_{\text{exp}}^* = 34$). Це вказує на більш збалансований режим паралельного виконання, у якому співвідношення між обсягом обчислень та накладними витратами є найбільш сприятливим.

Загалом результати підтверджують, що запропонований метод паралельної оцінки придатності дозволяє досягти *практично значущого прискорення (до ~ 10 разів)* навіть за умови неповного завантаження воркерів, а оптимальний вибір їх кількості є ключовим чинником підвищення продуктивності нейроеволюційного алгоритму в умовах обмежених обчислювальних ресурсів.

3.6. Аналіз моделі оцінки ефективності знайдених рішень за метрикою EfficiencyScore

Для узагальненої оцінки ефективності паралельної нейроеволюції в роботі використано інтегральну метрику EfficiencyScore, яка поєднує показники якості знайдених рішень, стабільності еволюційного процесу, часових витрат та структурної складності побудованих штучних нейронних мереж. На відміну від класичних метрик прискорення, дана метрика прямо враховує ймовірність успішного розв'язання задачі, що є принципово важливим для коректного аналізу результатів нейроеволюційних експериментів.

Згідно формули (2.11) метрика має наступний вигляд:

$$\text{EfficiencyScore} = \frac{SF}{\ln(TCG) + \varepsilon},$$

та враховує наступні параметри нейроеволюційного процесу:

- **стабільність розв'язання задачі** (частку успішних запусків S);
- **якість знайдених рішень** (нормалізований середній fitness score F);
- **обчислювальні витрати** (середній час однієї епохи T);
- **структурну складність** отриманих нейромереж (комплексність C);
- **швидкість еволюційної конвергенції** (середню кількість поколінь G).

Аналіз отриманих результатів для фіксованої кількості воркерів $p=30$ (таблиця 3.7) показує, що **не всі серії експериментів були повністю успішними**. Зокрема, для частини конфігурацій лише 90 % запусків завершилися знаходженням коректного розв'язку. Таким чином, значення коефіцієнта успішності $S < 1$ безпосередньо впливає на підсумкове значення **EfficiencyScore**, зменшуючи його навіть за умови високого значення оцінки придатності (fitness) знайдених рішень.

Попри це, для всіх досліджених розмірів популяції середній fitness успішних прогонів залишається близьким до оптимального значення, що свідчить про здатність алгоритму знаходити якісні керуючі нейромережі. Водночас часткова неуспішність окремих запусків вказує на зростання стохастичності еволюційного процесу при масштабуванні задачі або при

фіксованій кількості воркерів, що може бути пов'язано з підвищенням різноманітності популяції та ускладненням ландшафту пошуку.

Таблиця 3.7

Результати оцінювання метрики *EfficiencyScore* при $p = 30$

N	Всього запусків	Успішних запусків	Частка успіху S	Час епохи T, мс	Складність мереж C	Сер. к-сть поколінь G	Eff. Score
500	10	10	1.00	5.81	71.2	53.0	10.01
1000	10	10	1.00	15.81	63.8	46.9	9.29
2000	10	9	0.90	31.58	43.2	35.3	8.22
3000	10	9	0.90	37.29	36.7	20.6	8.51
5000	10	10	1.00	96.10	36.8	20.4	8.94

Отримані значення метрики *EfficiencyScore* демонструють узгоджену поведінку при масштабуванні розміру популяції за фіксованої кількості воркерів. Для малих і середніх популяцій ($N = 500, 1000$) досягаються найвищі значення метрики, що пояснюється поєднанням повної успішності експериментів ($S = 1$), відносно малих часових витрат та помірної топологічної складності рішень.

При збільшенні розміру популяції до $N \geq 2000$ спостерігається зменшення значення *EfficiencyScore*, що зумовлено одночасним зростанням середнього часу епохи та зниженням частки успішних запусків ($S = 0.9$). Водночас зменшення складності мереж-переможців і кількості поколінь частково компенсує негативний вплив часових витрат, що особливо помітно для $N = 3000$ та $N = 5000$.

Таким чином, отримані результати демонструють, що метрика *EfficiencyScore* адекватно реагує не лише на зміну обчислювальних витрат і складності рішень, але й на зниження надійності еволюційного процесу, відображене через коефіцієнт успішності S . Це робить її придатною для комплексної оцінки ефективності паралельних реалізацій алгоритму NEAT, особливо в умовах стохастичних експериментів з обмеженою кількістю запусків.

Узагальнюючи, можна зробити висновок, що навіть при частковій неуспішності окремих прогонів паралельна реалізація алгоритму NEAT з

фіксованою кількістю воркерів демонструє збалансований компроміс між швидкістю, якістю та стабільністю нейроеволюційного пошуку, що підтверджується значеннями інтегральної метрики *EfficiencyScore*.

3.7. Висновки до третього розділу

У третьому розділі дисертаційної роботи виконано комплексну експериментальну перевірку *вперше запропонованого* методу оптимізації алгоритму NEAT шляхом паралельної оцінки придатності організмів у популяції. Основною метою експериментальних досліджень було кількісне підтвердження ефективності *вперше запропонованої* моделі Master–Worker, а також валідація *вперше запропонованих* аналітичних моделей часових витрат і інтегральної оцінки ефективності, сформульованих у другому розділі.

Показано, що задача балансування подвійного оберненого маятника є репрезентативним тестовим середовищем для оцінювання масштабованості нейроеволюційних алгоритмів, оскільки вона поєднує високу динамічну складність, значні обчислювальні витрати на симуляцію та чутливість до якості керуючих стратегій. Це забезпечує коректну перевірку як швидкодії, так і результативності еволюційного пошуку.

Експериментально підтверджено, що етап оцінки придатності домінує у загальній структурі часових витрат алгоритму NEAT і визначає тривалість одного покоління еволюції. Реалізована паралельна схема оцінювання придатності за модифікованою моделлю Master–Worker дозволила ефективно розпаралелити саме цей етап без внесення змін у генетичні механізми алгоритму, зберігаючи коректність еволюційного процесу, механізми видоутворення та узгодженість інноваційних номерів.

Отримані експериментальні залежності часу виконання одного покоління від кількості паралельних воркерів продемонстрували чітко виражений нелінійний характер масштабування. Показано, що зі збільшенням кількості воркерів на початкових етапах спостерігається майже лінійне прискорення, однак після досягнення певного порогового значення виникає ефект насичення паралелізму. Подальше збільшення кількості воркерів не приводить до істотного скорочення часу виконання, а в окремих випадках може навіть *погіршувати продуктивність* через зростання накладних витрат синхронізації та апаратні обмеження.

Запропонована аналітична модель часових витрат адекватно відтворює експериментально спостережувані режими роботи системи, зокрема спадний тренд часу виконання, наявність плато продуктивності та залежність оптимальної кількості воркерів від розміру популяції. Показано, що хоча точковий прогноз оптимального значення кількості воркерів може мати обмежену точність, модель стабільно локалізує 5%-ву Парето-оптимальну область параметрів, у межах якої забезпечується практично мінімальний час виконання еволюційного процесу.

Аналіз Парето-оптимальних областей для різних розмірів популяції показав, що зона раціональних значень кількості воркерів є відносно широкою. Це свідчить про високу стійкість продуктивності паралельної реалізації алгоритму NEAT до похибок у виборі параметрів паралелізації та дозволяє гнучко адаптувати конфігурацію обчислювальних ресурсів під конкретні апаратні обмеження без суттєвої втрати ефективності.

Експериментально встановлено, що використання оптимально або квазіоптимально підібраної кількості воркерів **забезпечує прискорення нейроеволюційного процесу до 10–11 разів порівняно з однопотоковою реалізацією алгоритму NEAT.** При цьому коефіцієнт ефективності використання воркерів закономірно зменшується зі зростанням ступеня паралелізму, що відповідає класичним закономірностям паралельних обчислень і підтверджує адекватність отриманих результатів.

Застосування інтегральної метрики EfficiencyScore дозволило здійснити комплексну оцінку ефективності паралельної нейроеволюції з урахуванням не лише швидкодії, а й якості знайдених рішень, стабільності еволюційного процесу, структурної складності побудованих нейронних мереж і швидкості збіжності. Показано, що паралельна реалізація алгоритму NEAT забезпечує збалансований компроміс між цими характеристиками та не призводить до деградації якості рішень навіть за значного скорочення часу еволюції.

Проведений аналіз похибок прогнозової моделі часових витрат показав, що основні відхилення зумовлені ефектами апаратного насичення, нелінійними

накладними витратами керування конкурентними процесами та пологим характером мінімуму функції часу. Водночас навіть за наявності таких похибок модель залишається придатною для практичного використання як інструмент попереднього аналітичного налаштування параметрів паралельної нейроеволюції.

Узагальнюючи результати третього розділу, можна стверджувати, що експериментальні дослідження повністю підтвердили теоретичні положення, викладені у попередніх розділах дисертації. Вперше запропонований метод паралельної оцінки придатності на основі моделі Master–Worker є ефективним, масштабованим і практично доцільним засобом підвищення продуктивності алгоритму NEAT в умовах обмежених обчислювальних ресурсів. Отримані результати створюють надійну експериментальну основу для узагальнених висновків дисертації та обґрунтовують можливість застосування розробленого підходу у реальних задачах нейроеволюції та керування складними динамічними системами.

ВИСНОВКИ

У дисертаційній роботі розв'язано актуальну науково-практичну задачу підвищення продуктивності алгоритму NEAT шляхом оптимізації етапу оцінки придатності організмів на основі паралельних обчислень, що має важливе значення для застосування нейроеволюційних методів у задачах з високою обчислювальною складністю та обмеженими ресурсами.

У ході дослідження послідовно виконано всі поставлені часткові завдання та отримано такі узагальнені результати.

1. **Проведено формальний аналіз обчислювальної складності етапу оцінки придатності у NEAT**, який показав, що саме цей етап є домінуючим у загальній структурі часових витрат еволюційного циклу. Встановлено залежність часу виконання від розміру популяції, складності топології мереж, тривалості симуляції та параметрів обчислювальної платформи, що створило теоретичне підґрунтя для подальшої оптимізації.
2. **Визначено ключові властивості моделі розпаралелювання**, необхідні для збереження коректності еволюційного процесу NEAT, зокрема незалежність оцінки придатності організмів, відсутність побічних ефектів між паралельними симуляціями, коректну синхронізацію результатів та детермінованість процедур агрегації.
3. **Обґрунтовано вибір архітектури паралельних обчислень Master–Worker** як найбільш придатної для динамічного розподілу навантаження на етапі оцінки придатності. Показано, що дана архітектура забезпечує природне балансування навантаження, масштабованість та простоту інтеграції у нейроеволюційний цикл.
4. **Розроблено та реалізовано модель Master–Worker з використанням конкурентних механізмів мови програмування Go** (goroutines та channels). Запропонована реалізація дозволяє ефективно використовувати багатоядерні процесори з мінімальними комунікаційними накладними

витратами та не потребує зовнішніх бібліотек розподілених обчислень, що істотно спрощує практичне застосування.

5. **Удосконалено модель оцінки ефективності побудови топології штучних нейронних мереж алгоритмом NEAT на основі часових витрат**, у межах якої вперше формалізовано зв'язок між структурною складністю мережі, масштабом популяції та продуктивністю апаратної платформи. Модель дозволяє аналітично визначати оптимальну кількість воркерів для конкретної задачі та виконувати багатоцільову оптимізацію.
6. **Вперше запропоновано комплексну інтегральну метрику оцінки продуктивності алгоритму NEAT**, яка поєднує часові характеристики, швидкість збіжності еволюційного процесу та якість фінального рішення. Запропонована метрика дозволяє об'єктивно порівнювати різні реалізації NEAT та різні конфігурації паралельного виконання з урахуванням не лише швидкості, а й ефективності використання ресурсів.
7. **Розроблено спеціалізоване програмне забезпечення для візуалізації результатів експериментів**, що забезпечує наочний аналіз залежностей між часом виконання, кількістю воркерів, розміром популяції та ефективністю еволюції, а також підтримує аналіз Парето-оптимальних областей.
8. **Вперше запропоновано метод оптимізації алгоритму NEAT для підвищення швидкості побудови топології ШНМ та покращення ефективності знайдених рішень в умовах обмежених обчислювальних ресурсів**. До складу розробленого методу включені три моделі:
 - Модифікована модель Master-Worker розподілених обчислень для паралельної оцінки придатності рішень у алгоритмі NEAT побудована з використанням вбудованих у мову GO механізмів конкурентного програмування – goroutines і channels
 - Удосконалена модель оцінки загальних часових витрат на весь процес еволюції під керуванням алгоритму NEAT

- Комплексна інтегральна модель оцінки ефективності знайдених рішень (фенотипів ШНМ)

9. **Експериментально підтверджено ефективність запропонованих моделей та методу** на задачах еволюційного конструювання штучних нейронних мереж, зокрема на класичній задачі балансування подвійного оберненого маятника. Результати показали суттєве зменшення загального часу еволюції та зростання прискорення нейроеволюційного процесу при оптимальному виборі кількості воркерів. Було досягнуто прискорення нейроеволюційного процесу **до 10–11 разів** (9.29 – 10.58) порівняно з класичним однопоточним варіантом для різних розмірів популяції вирішувачів.

У сукупності отримані результати **підтверджують наукову новизну та практичну значущість дисертаційної роботи**. Запропоновані моделі, методи та програмні рішення створюють цілісну методологію оптимізації нейроеволюційних алгоритмів у паралельному середовищі та можуть бути використані як у подальших наукових дослідженнях, так і у прикладних системах штучного інтелекту, що працюють в умовах обмежених обчислювальних ресурсів.

Узагальнюючи, можна зробити висновок, що дисертаційна робота містить **науково обґрунтоване та експериментально підтверджене рішення проблеми масштабованості алгоритму NEAT**, яке поєднує теоретичні моделі, алгоритмічні удосконалення та практичну реалізацію.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. LeCun, Yann and Courant, A. Path Towards Autonomous Machine Intelligence Version 0.9.2, 2022-06-27. (2022). URL: <https://openreview.net/pdf?id=BZ5a1r-kVsf> (retrieval date: 01.02.2026).
2. Darwin, Charles Robert. On the origin of species. (1859).
3. Stanley, K.O., Miikkulainen, R. Evolving neural networks through augmenting topologies. *Evolutionary Computations*, 2002. Vol. 10, N 2. P. 99–127. DOI: <https://doi.org/10.1162/106365602320169811>
4. Omelianenko, I. Hands-On Neuroevolution with Python: Build high-performing artificial neural network architectures using neuroevolution-based algorithms. Birmingham, UK : Packt Publishing, 2019. ISBN: 9781838824914, 368 p.
5. Siebel, N.T., Sommer, G. Evolutionary reinforcement learning of artificial neural networks. *International Journal of Hybrid Intelligent Systems*, 2007. Vol. 4, N 3. P. 171–183. DOI: <https://dl.acm.org/doi/10.5555/1367012.1367016>
6. Wen R., Guo Z., Zhao T., Ma X., Wang Q., Wu Z. Neuroevolution of augmenting topologies based muscular-skeletal arm neurocontroller. In: *2017 IEEE international instrumentation and measurement technology conference (I2MTC)*, 2017. P. 1–6. DOI: <https://doi.org/10.1109/I2MTC.2017.7969727>
7. Buchanan E., Le Goff L.K., Li W., Hart E., Eiben A.E., De Carlo M., Winfield A.F., Hale M.F., Woolley R., Angus M., Timmis J., Tyrrell A.M. Bootstrapping artificial evolution to design robots for autonomous fabrication. *Robotics*, 2020. Vol. 9, N 4. P. 1–24. DOI: <https://doi.org/10.3390/robotics9040106>
8. Вітюк А.С., Дорошенко А.Ю. Використання нейроеволюції при пошуку політик в формі нейромереж для управління робочою кінцівкою. Перша Науково-теоретична конференція “Пріоритети і виклики реалізації Стратегії розвитку штучного інтелекту в Україні”. *Artificial Intelligence*, 2023 (2). ISSN 2710 – 1673
9. Dinh, H., Aubert, N., Noman, N., Fujii, T., Rondelez, Y., Iba, H. An effective method for evolving reaction networks in synthetic biochemical systems. *IEEE*

- Transactions on Evolutionary Computation*, 2015. Vol. 19, N 3. P. 374–86. DOI: <https://doi.org/10.1109/TEVC.2014.2326863>
10. Wang, G., Cheng, G., Carr, T. The application of improved neuroevolution of augmenting topologies neural network in Marcellus Shale lithofacies prediction. *Computers & Geosciences*, 2013. Vol. 54. P. 50–65. DOI: <https://doi.org/10.1016/j.cageo.2013.01.022>
 11. Aaltonen, T., Adelman, J., Akimoto, T., Albrow, M., et al. Measurement of the top-quark mass with dilepton events selected using neuroevolution at CDF. *Physical Review Letters*, 2009. Vol. 102. N 15. P. 1–7. DOI: <https://doi.org/10.1103/PhysRevLett.102.152001>
 12. Nadkarni, J., Ferreira Neves, R. Combining neuroevolution and principal component analysis to trade in the financial markets. *Expert Systems with Applications*, 2018. Vol. 103. P. 184–95. DOI: <https://doi.org/10.1016/j.eswa.2018.03.012>
 13. Clune, J., Mouret, J.B., Lipson, H. The evolutionary origins of modularity. *Proceedings of the Royal Society B*, 2013. Vol. 280, N 1775. P. 1–9. DOI: <https://doi.org/10.1098/rspb.2012.2863>
 14. Omelianenko, I. Artificial Swarm Intelligence and Cooperative Robotic Systems. *Preprints.org*, 2019, P. 1–8. DOI: <https://doi.org/10.20944/preprints201901.0282.v1>
 15. Omelianenko, I. Autonomous Artificial Intelligent Agents. *Machine Learning and the City*. Hoboken, NJ, USA : John Wiley & Sons, 2022. P. 263–285. DOI: <https://doi.org/10.1002/9781119815075.ch21>
 16. Omelianenko, I. and Sinitsyn, I. Artificial swarm intelligence. *International Scientific Technical Journal "Problems of Control and Informatics"*, 2024, Vol. 69. N 3. P. 91–103. DOI: <https://doi.org/10.34229/1028-0979-2024-3-7>
 17. Sinitsyn, I., Doroshenko, A., Mamedov, T., Yatsenko, O. A method of automated design of neuroevolution algorithms based on Glushkov algebra of algorithms. *International Scientific Technical Journal "Problems of Control and Informatics"*, 2023. Vol. 68, N 3. P. 74–85. DOI: <https://doi.org/10.34229/1028-0979-2023-3-8>

18. Omelianenko, I. Simulation of the autonomous maze navigation using the NEAT algorithm. *Problems in programming*, 2023. Vol. 4, P. 76-89. DOI: <https://doi.org/10.15407/pp2023.04.076>
19. Goodfellow, I., Bengio, Y. and Courville, A. Deep Learning. Cambridge, MA, USA : MIT Press, 2016. ISBN: 0262035618, 800 p.
20. LeCun, Y., Bengio, Y. and Hinton, G. Deep learning. *Nature*, 2015. Vol 521. P. 436–444. DOI: <https://doi.org/10.1038/nature14539>
21. Mnih, V. et al. Human-level control through deep reinforcement learning. *Nature*, 2015. Vol. 518. P. 529–533. DOI: <https://doi.org/10.1038/nature14236>
22. Sutton, R. S. and Barto, A. G. Reinforcement Learning: An Introduction. Cambridge, MA : MIT Press, 2018. ISBN: 9780262257053, 342 p.
23. Rumelhart, D. E., Hinton, G. E. and Williams, R. J. Learning representations by back-propagating errors. *Nature*, 1986. Vol 323. P. 533–536. DOI: <https://doi.org/10.1038/323533a0>
24. Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser Ł., and Polosukhin I. Attention is all you need. *In Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS'17)*. Red Hook, NY, USA : Curran Associates Inc., 2017. ISBN: 9781510860964, P. 6000–6010.
25. De Jong, K. A. Evolutionary Computation: A Unified Perspective. Cambridge, MA, USA : MIT Press, 2002. ISBN: 9780262255981, 250 p.
26. Gruau, F. Automatic definition of modular neural networks. *Adaptive Behavior*, 1994. Vol. 3, N 2. P. 151–183. DOI: <https://doi.org/10.1177/10597123940030020>
27. Yao, X. A review of evolutionary artificial neural networks. *International Journal of Intelligent Systems*, 1993. Vol. 8. P. 539–567. DOI: <https://doi.org/10.1002/int.4550080406>
28. Floreano, D., Dürri, P. and Mattiussi, C. Neuroevolution: from architectures to learning. *Evolutionary Intelligence*, 2008. Vol. 1. P. 47–62. DOI: <https://doi.org/10.1007/s12065-007-0002-4>

29. Soltoggio, A., Stanley, K. O., Risi, S. Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. *Neural Networks*, 2018. Vol. 108. P. 48–67. DOI: <https://doi.org/10.1016/j.neunet.2018.07.013>
30. Dasgupta, D., McGregor, D. Designing application-specific neural networks using the structured genetic algorithm. *In Proceedings of COGANN-92: International Workshop on Combinations of Genetic Algorithms and Neural Networks* 87–96. Baltimore, MD, USA : IEEE, 1992. P. 87-96. DOI: <https://doi.org/10.1109/COGANN.1992.273946>
31. Pujol, J. C. F., Poli, R. Evolving the topology and the weights of neural networks using a dual representation. *Applied Intelligence Journal*, 1998. Vol. 8. P. 73–84. DOI: <https://doi.org/10.1023/A:1008272615525>
32. Bongard, J. C., Pfeifer, R. Evolving Complete Agents using Artificial Ontogeny. *In Proceedings of Morpho-functional Machines: The New Species* (eds F. Hara and R. Pfeifer). Tokyo, JP : Springer, 2003. P. 237–258.
33. Gruau, F. Genetic synthesis of modular neural networks. *In Proceedings of 5th International Conference on Genetic Algorithms* (ed. S. Forrest). San Francisco, CA, USA : Morgan Kaufmann, 1993. P. 318–325.
34. Khan, M. M., Ahmad, A. M., Khan, G. M. and Miller, J. F. Fast learning neural networks using cartesian genetic programming. *Neurocomputing*, 2013. Vol. 121. P. 274–289. DOI: <https://doi.org/10.1016/j.neucom.2013.04.005>
35. Turner, A. J., Miller, J. F. Neuroevolution: evolving heterogeneous artificial neural networks. *Evolving Intelligence*, 2014. Vol. 7. P. 135–154. DOI: <https://doi.org/10.1007/s12065-014-0115-5>
36. Mattiussi, C., Floreano, D. Analog genetic encoding for the evolution of circuits and networks. *IEEE Transactions on Evolutionary Computation*, 2006. Vol. 11. P. 596–607. DOI: <https://doi.org/10.1109/TEVC.2006.886801>
37. Kim, D., Tran, A., Kim, H.J. et al. Gene regulatory network reconstruction: harnessing the power of single-cell multi-omic data. *npj Systems Biology and Applications*, 2023, Vol. 9, N 51. P. 1–13 DOI: <https://doi.org/10.1038/s41540-023-00312-6>

38. Moriarty, D. E., Miikkulainen, R. Evolving obstacle avoidance behavior in a robot arm. In *From Animals to Animats 4: Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior* (eds P. Maes et al.). Cambridge, MA, USA : MIT Press, 1996. P. 468–475. DOI: <https://doi.org/10.7551/mitpress/3118.001.0001>
39. Nolfi, S., Floreano, D. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. Cambridge, MA, USA : MIT Press, Cambridge, 2000. 332p. DOI: <https://doi.org/10.7551/mitpress/2889.001.0001>
40. Hornby, G et al. Evolving robust gaits with AIBO. Proceedings 2000 ICRA. Millennium Conference. *IEEE International Conference on Robotics and Automation*. San Francisco, CA, USA : IEEE, 2000. Vol. 3. P. 3040-3045. DOI: <https://doi.org/10.1109/ROBOT.2000.846489>
41. Togelius, J., Yannakakis, G. N., Stanley, K. O. and Browne, C. Search-based procedural content generation: a taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, 2011. Vol. 3, N 3. P. 172–186. DOI: <https://doi.org/10.1109/TCIAIG.2011.2148116>
42. Stanley, K. O., Bryant, B. D. and Miikkulainen, R. Real-time neuroevolution in the NERO video game. *IEEE Transactions on Evolutionary Computation*, 2005. Vol. 9. P. 653–668. DOI: <https://doi.org/10.1109/TEVC.2005.856210>
43. Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., et al. (2024). A Comprehensive Overview of Large Language Models. *arXiv [Cs.CL]*, 2024. P. 1-46. DOI: <https://doi.org/10.48550/arXiv.2307.06435>
44. Polyak, B. T. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 1964. Vol. 4. N 5. P. 1–17. DOI: [https://doi.org/10.1016/0041-5553\(64\)90137-5](https://doi.org/10.1016/0041-5553(64)90137-5)
45. Qian, N. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 1999. Vol. 12. N 1. P. 145–151. DOI: [https://doi.org/10.1016/S0893-6080\(98\)00116-6](https://doi.org/10.1016/S0893-6080(98)00116-6)

46. LeCun, Y., Bottou, L., Bengio, Y. and Haffner, P. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 1998. Vol. 86. N 11. P. 2278–2324. DOI: <https://doi.org/10.1109/5.726791>
47. Hochreiter, S., Schmidhuber, J. Long short-term memory. *Neural Computation*, 1997. Vol. 9. N 8. P. 1735–1780. DOI: <https://doi.org/10.1162/neco.1997.9.8.1735>
48. Dahl, G., Yu, D., Deng, L. and Acero, A. Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 2012. Vol. 20, N 1. P. 30–42. DOI: <https://doi.org/10.1109/TASL.2011.2134090>
49. Hinton, G. et al. Deep neural networks for acoustic modeling in speech recognition: the shared views of four research groups. *IEEE Signal Processing Magazine*, 2012. Vol. 29, N 6. P. 82–97. DOI: <https://doi.org/10.1109/MSP.2012.2205597>
50. Krizhevsky, A., Sutskever, I. and Hinton, G. E. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 2017. Vol. 60, N 6. P. 84-90. DOI: <https://doi.org/10.1145/3065386>
51. Turner, R. E. An Introduction to Transformers. *arXiv [Cs.LG]*, 2024. P. 1-10. DOI: <https://doi.org/10.48550/arXiv.2304.10557>
52. Lillicrap, T.P., Hunt, J.J., Pritzel, A., Heess, N.M., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv [Cs.LG]*, 2016. P. 1-14. DOI: <https://doi.org/10.48550/arXiv.1509.02971>
53. Schulman, J., Levine, S., Abbeel, P., Jordan, M. and Moritz, P. Trust region policy optimization. *In Proceedings of the 32nd International Conference on International Conference on Machine Learning*. Lille, France, 2015. P. 1889–1897. DOI: <https://doi.org/10.48550/arXiv.1502.05477>
54. Schulman, J., Wolski, F., Dhariwal, P., Radford, A. and Klimov, O. Proximal policy optimization algorithms. *arXiv [Cs.LG]*, 2017. P. 1-12. DOI: <https://doi.org/10.48550/arXiv.1707.06347>
55. Salimans, T., Ho, J., Chen, X. and Sutskever, I. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv, [stat.ML]*, 2017. P. 1-13. DOI: <https://doi.org/10.48550/arXiv.1703.03864>

56. Wierstra, D. et al. Natural evolution strategies. *Journal of Machine Learning Research*, 2014. Vol. 15, N 1. P. 949–980. URL: <https://jmlr.org/papers/volume15/wierstra14a/wierstra14a.pdf> (retrieval date: 01.02.2026).
57. Mnih, V. et al. Asynchronous methods for deep reinforcement learning. *In Proceedings of the 33-rd International Conference on Machine Learning*. New York, NY, USA : PMLR, 2016. Vol. 48. P. 1928–1937. URL: <https://proceedings.mlr.press/v48/mniha16.html> (retrieval date: 01.02.2026).
58. Such, F. P. et al. Deep neuroevolution: genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv, [cs.NE]*, 2017. P. 1–16. DOI: <https://doi.org/10.48550/arXiv.1712.06567>
59. Hessel, M. et al. Rainbow: combining improvements in deep reinforcement learning. *In proceedings of the 32nd AAAI Conference on Artificial Intelligence*. New Orleans, Louisiana, USA : AAAI press, 2018. Vol. 32, N 1. P. 3215–3222. DOI: <https://dx.doi.org/10.1609/aaai.v32i1.11796>
60. Horgan, D. et al. Distributed prioritized experience replay. *In Proceedings of the 2018 International Conference on Learning Representations*. OpenReview, 2018. P. 1–19. URL: <https://openreview.net/forum?id=H1Dy---0Z> (retrieval date: 01.02.2026).
61. Herculano-Houzel, S. The human brain in numbers: a linearly scaled-up primate brain. *Frontiers in Human Neuroscience*, 2009. Vol. 3, N 31. P. 1–11. DOI: <https://doi.org/10.3389/neuro.09.031.2009>
62. Mania, H., Guy, A. and Recht, B. Simple random search provides a competitive approach to reinforcement learning. *arXiv, [cs.LG]*, 2018. P. 1–22. DOI: <https://doi.org/10.48550/arXiv.1803.07055>
63. Clune, J., Stanley, K. O., Pennock, R. T. and Ofria, C. On the performance of indirect encoding across the continuum of regularity. *IEEE Transactions on Evolutionary Computation*, 2011. Vol. 15, N 3. P. 346–367. DOI: <https://doi.org/10.1109/TEVC.2010.2104157>

64. Cully, A., Clune, J., Tarapore, D. and Mouret, J.-B. Robots that can adapt like animals. *Nature*, 2015. Vol. 521. P. 503–507. DOI: <https://doi.org/10.1038/nature14422>
65. Lehman, J., Chen, J., Clune, J. and Stanley, K. O. Safe mutations for deep and recurrent neural networks through output gradients. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2018. P. 117–124. DOI: <https://doi.org/10.1145/3205455.3205473>
66. Fortunato, M. et al. Noisy networks for exploration. *In Proceedings of the 2018 International Conference on Learning Representations*. OpenReview, 2018. P. 1-21. URL: <https://openreview.net/forum?id=rywHCPkAW> (retrieval date: 01.02.2026).
67. Plappert, M. et al. Parameter space noise for exploration. *In Proceedings of the 2018 International Conference on Learning Representations*. OpenReview, 2018. P. 1-18. URL: <https://openreview.net/forum?id=ByBA12eAZ> (retrieval date: 01.02.2026).
68. Lehman, J. et al. The surprising creativity of digital evolution: A collection of anecdotes from the evolutionary computation and artificial life research communities. *arXiv [cs.NE]*, 2018. P. 1-34. DOI: <https://doi.org/10.48550/arXiv.1803.03453>
69. Dawkins, R. *The Extended Phenotype: The Gene as the Unit of Selection*. Oxford University Press, 1982. 307 pp.
70. Gould, S. J. *Full House: the spread of excellence from Plato to Darwin*. New York, NY, USA : Three Rivers Press, 1997. 244 pp.
71. Goldberg, D. E., Richardson, J. Genetic algorithms with sharing for multimodal function optimization. *In Proceedings of the 2nd International Conference on Genetic Algorithms*. Cambridge, MA, USA : L. Erlbaum Associates, 1987. P 41-49
72. Mahfoud, S. W. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana-Champaign, 1995. 264 pp. URL: http://www.leg.ufpr.br/~leonardo/artigos/tese_mahfoud.pdf (retrieval date: 01.02.2026).

73. Jong, De, K. A. An Analysis of the Behaviour of a Class of Genetic Adaptive Systems. PhD thesis, University of Michigan, 1975. 266 pp. URL: <https://www.proquest.com/docview/302788400/previewPDF/3635B1EA386A4F82PQ/1> (retrieval date: 01.02.2026).
74. Lehman, J., Stanley, K. O. Abandoning objectives: evolution through the search for novelty alone. *Evolutionary Computations*. Cambridge, MA, USA : MIT Press, 2011. Vol. 19, N 2. P. 189–223. DOI: https://doi.org/10.1162/EVCO_a_00025
75. Neyshabur, B., Salakhutdinov, R. R. and Srebro, N. Path-SGD: path-normalized optimization in deep neural networks. *In Advances in Neural Information Processing Systems 28 (NIPS 2015)*. Cambridge, MA, USA : MIT, 2015. P. 2422–2430. URL: <https://papers.nips.cc/paper/5797-path-sgd-path-normalized-optimization-in-deep-neural-networks.pdf> (retrieval date: 01.02.2026).
76. Radcliffe, N. J. Genetic set recombination and its application to neural network topology optimisation. *Neural Computing and Applications*, 1993. Vol. 1. P. 67–90. DOI: <https://doi.org/10.1007/BF01411376>
77. Benson-Amram, S., Holekamp, K. E. Innovative problem solving by wild spotted hyenas. *Proceedings of the Royal Society B: Biological Sciences*, 2012. Vol. 279, N 1744. P. 4087–4095. DOI: <https://doi.org/10.1098/rspb.2012.1450>
78. Kanter, R. M. The Change Masters: Innovation and Entrepreneurship in the American Corporation. New York, NY, USA : Simon & Schuster, 1984. 436 pp. ISBN: 9780671528003
79. Mouret, J.B., Doncieux, S. Encouraging behavioral diversity in evolutionary robotics: an empirical study. *Evolutionary Computation*. Cambridge, MA, USA : MIT Press, 2012. Vol 20, N 1. P. 91–133. DOI: https://doi.org/10.1162/EVCO_a_00048
80. Stanton, C., Clune, J. Deep curiosity search: Intra-life exploration improves performance on challenging deep reinforcement learning problems. *arXiv, [cs.AI]*, 2018. P. 1-15. DOI: <https://doi.org/10.48550/arXiv.1806.00553>
81. Mengistu, H., Lehman, J. and Clune, J. Evolvability search: directly selecting for evolvability in order to study and produce it. *In Proceedings of the Genetic and*

- Evolutionary Computation Conference (GECCO)*. ACM, 2016. P. 141–148. DOI: <https://doi.org/10.1145/2908812.2908838>
82. Gravina, D., Liapis, A. and Yannakakis, G. Surprise search: beyond objectives and novelty. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2016. P. 677–684. DOI: <https://doi.org/10.1145/2908812.2908817>
83. Deb, K., Pratap, A., Agarwal, S. and Meyarivan, T. A. M. T. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computations*. IEEE, 2002. Vol. 6, N 2. P. 182–197. DOI: <https://doi.org/10.1109/4235.996017>
84. Zitzler, E., Thiele, L. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE Transactions on Evolutionary Computations*. IEEE, 1999. Vol. 3, N 4. P. 257–271. DOI: <https://doi.org/10.1109/4235.797969>
85. Conti, E. et al. Improving exploration in evolutionary strategies for deep reinforcement learning via a population of novelty-seeking agents. In *proceedings of the 32nd International Conference on Neural Information Processing Systems (NIPS'18), Montreal, Canada*. Red Hook, NY, USA : Curran Associates, 2018. P. 5032–5043. URL: <https://dl.acm.org/doi/pdf/10.5555/3327345.3327410> (retrieval date: 01.02.2026).
86. Pugh, J. K., Soros, L. B. and Stanley, K. O. Quality diversity: a new frontier for evolutionary computation. *Frontiers in Robotics and AI*, 2016. Vol. 3, N 40. 17 pp. DOI: <https://doi.org/10.3389/frobt.2016.00040>
87. Lehman, J., Stanley, K. O. Evolving a diversity of virtual creatures through novelty search and local competition. In *the Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO)*. ACM, 2011. P. 211–218. DOI: <https://doi.org/10.1145/2001576.2001606>
88. Mouret, J. B., Clune, J. Illuminating search spaces by mapping elites. *arXiv, [cs.AI]*, 2015. P. 1-15. DOI: <https://doi.org/10.48550/arXiv.1504.04909>

89. Brant, J. C., Stanley, K. O. Minimal criterion coevolution: a new approach to open-ended search. *In Proceedings of the Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2017. P. 67–74: DOI: <https://doi.org/10.1145/3071178.3071186>
90. Hodjat, B., Shahrzad, H. and Miikkulainen, R. Distributed age-layered novelty search. In *Proceedings of the 15th International Conference on the Synthesis and Simulation of Living Systems (Alife XV)* Cambridge, MA, USA : MIT Press, 2016. P. 131–138. DOI: <https://doi.org/10.1162/978-0-262-33936-0-ch027>
91. Huizinga, J., Mouret, J.-B. and Clune, J. Does aligning phenotypic and genotypic modularity improve the evolution of neural networks? *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2016. P. 125–132. DOI: <https://doi.org/10.1145/2908812.2908836>
92. Meyerson, E., Miikkulainen, R. Discovering evolutionary stepping stones through behavior domination. *In Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*. ACM, 2017. P. 39–146. DOI: <https://doi.org/10.1145/3071178.3071315>
93. Nguyen, A., Yosinski, J. and Clune, J. Understanding innovation engines: automated creativity and improved stochastic optimization via deep learning. *Evolutionary Computation*, 2016. Vol. 24, N 3. P. 545–572. DOI: https://doi.org/10.1162/EVCO_a_00189
94. Stanley, K. O., D'Ambrosio, D. B. and Gauci, J. A Hypercube-Based Encoding for Evolving Large-Scale Neural Networks. *Artificial Life*, 2009. Vol. 15, N. 2, P. 185–212. DOI: <https://doi.org/10.1162/artl.2009.15.2.15202>
95. Stanley, O. Kenneth. Compositional pattern producing networks: A novel abstraction of development. *Genetic Programming and Evolvable Machines*. Springer, 2007. Vol. 8. P. 131–162. DOI: <https://doi.org/10.1007/s10710-007-9028-8>
96. Omelianenko, Ia., Doroshenko, A., Rodin, Ye. Autonomous Navigation Through the Maze Using Coevolution Strategy // UkrPROG-2024, 14-th International Scientific and Practical Programming Conference, Kyiv, Ukraine, May 14-15,

- 2024, CEUR Workshop Proceedings, 2024, Vol-3806, pp. 301-311. ISSN 1613-0073. URL: https://ceur-ws.org/Vol-3806/S_29_Omelianenko_Doroshenko_Rodin.pdf (retrieval date: 01.02.2026).
97. Risi, S., Stanley, O. K. An Enhanced Hypercube-Based Encoding for Evolving the Placement, Density, and Connectivity of Neurons. *Artificial Life*, 2012. Vol. 18, N. 4. P. 331-363, DOI: https://doi.org/10.1162/ARTL_a_00071
 98. Hawkins J, Lewis M, Klukas M, Purdy S, Ahmad S. A Framework for Intelligence and Cortical Function Based on Grid Cells in the Neocortex. *Frontiers in Neural Circuits*, 2019. Vol. 12, N. 121. DOI: <https://doi.org/10.3389/fncir.2018.00121>
 99. Benso, F., Chiorri, C., Ardu, E., Venuti, P., & Pasqualotto, A. Beyond modular and non-modular states: theoretical considerations, exemplifications, and practical implications. *Frontiers in psychology*, 2025. Vol. 16, N. 1456587. DOI: <https://doi.org/10.3389/fpsyg.2025.1456587>
 100. McCool, M., Robison, A. D., and Reinders. J. Structured Parallel Programming. Chapter 2. Morgan Kaufmann, Boston, 2012. Pp 39-75, DOI: <https://doi.org/10.1016/B978-0-12-415993-8.00002-5>, ISBN: 978-0-12-415993-8
 101. Alba, E., Tomassini, M. Parallelism and evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 2002. Vol. 6, N. 5, pp. 443-462. DOI: <https://doi.org/10.1109/TEVC.2002.800880>.
 102. Gong, Y., Fukunaga, A. Distributed island-model genetic algorithms using heterogeneous parameter settings. In *2011 IEEE congress of evolutionary computation (CECO)*. IEEE, 2011. pp. 820-827. URL: <https://www.metahack.org/gong-fukunaga-island-model-cec2011.pdf> (retrieval date: 01.02.2026).
 103. Alba, E., Dorronsoro, B. Cellular Genetic Algorithms. Springer New York, NY, 2008. 248 p. DOI: <https://doi.org/10.1007/978-0-387-77610-1>. ISBN: 978-0-387-77609-5
 104. Cantu-Paz, E. A Survey of Parallel Genetic Algorithms. *Calculateurs Paralleles*, 1998. Vol. 10, N. 2, pp. 141- 171. URL: <https://scispace.com/pdf/a-survey-of-parallel-genetic-algorithms-rh37ea0cwr.pdf> (retrieval date: 01.02.2026).

105. Li, Z., Liu, J., and Wu, K. A Multiobjective Evolutionary Algorithm Based on Structural and Attribute Similarities for Community Detection in Attributed Networks. *IEEE Transactions on Cybernetics*. IEEE, 2018. Vol. 48, N. 7, pp. 1963-1976. DOI: <https://doi.org/10.1109/TCYB.2017.2720180>.
106. Miikkulainen, R., Liang, J., Meyerson, E., Rawal, A., Fink, D., Francon, O., Raju, B., Shahrzad, H., Navruzian, A., Duffy, N. Evolving deep neural networks. In *Artificial Intelligence in the Age of Neural Networks and Brain Computing (Second Edition)*. Academic Press. Elsevier Amsterdam, Netherlands, 2024. pp. 269-287. DOI: <https://doi.org/10.1016/B978-0-323-96104-2.00002-6>.
107. Liang, J., Lou, Y., Yu, M. et al. A survey of surrogate-assisted evolutionary algorithms for expensive optimization. *Journal of Membrane Computing*. Springer, 2025. Vol. 7. pp. 108–127. DOI: <https://doi.org/10.1007/s41965-024-00165-w>
108. Omelianenko, I. Parallel Fitness Scores Evaluation to Improve Training Speed of the NEAT Algorithm Using GO Routines. In: Yang, X.S., Sherratt, R.S., Dey, N., Joshi, A. (eds) Proceedings of Tenth International Congress on Information and Communication Technology. ICICT 2025. *Lecture Notes in Networks and Systems*. Singapore, Springer, 2025. Vol. 1441, pp. 207-216. ISSN 2367-3370. DOI: https://doi.org/10.1007/978-981-96-6938-7_17
109. Omelianenko, I. The GoLang implementation of NeuroEvolution of Augmenting Topologies (NEAT) algorithm. [Research Software]. v4.2.3, Zenodo, 2 Jan. 2026. DOI: <https://doi.org/10.5281/zenodo.18131733>
110. Omelianenko, I. NEAT efficiency scoring metrics and visualization tools. [Research software]. v0.2.0, Zenodo, 2 Jan. 2026. DOI: <https://doi.org/10.5281/zenodo.18131847>.
111. Hairer, E., Nørsett, S. P., Wanner, G. Solving Ordinary Differential Equations I: Nonstiff Problems. Springer Series in Computational Mathematics, Springer, 1993. Vol. 8. 528p. ISBN: 9783540788621. DOI: <https://doi.org/10.1007/978-3-540-78862-1>

ДОДАТКИ

ДОДАТОК А – ЕКСПЕРИМЕНТАЛЬНІ ДАННІ

Таблиця 1

Експериментальні данні отримані для задачі балансування подвійного
оберненого маятника з розміром популяції 500 ($N = 500$)

Кількість воркерів	Час однієї епохи, мс	Розрахований показник (EfficiencyScore)
1	41.173942	8.365687
2	21.678326	8.840093
3	15.005901	9.137236
4	11.689053	9.350655
5	9.769996	9.510132
6	8.542701	9.633111
7	7.668004	9.734405
8	6.982998	9.823894
9	6.532027	9.88875
10	6.160648	9.946323
11	6.012069	9.970533
12	5.851492	9.997519
13	5.738246	10.020473
15	5.767527	10.011986
20	5.656775	10.03146
30	5.683619	10.026698
40	5.77491	10.010704
50	5.748878	10.015234
60	5.771086	10.017999
100	5.702111	10.023434
300	6.008497	9.964172
500	6.116721	9.949243
700	6.383736	9.904387
1000	6.695565	9.857823
1500	7.059238	9.813425
2000	7.626916	9.730228

Експериментальні данні отримані для задачі балансування подвійного
оберненого маятника з розміром популяції 1000 ($N = 1000$)

Кількість воркерів	Час однієї епохи, мс	Розрахований показник (EfficiencyScore)
1	144.276536	7.70683
2	75.068294	8.115454
3	50.584984	8.384039
4	38.478843	8.580834
5	31.470805	8.732662
6	26.451437	8.867203
7	23.266494	8.967978
8	20.702005	9.064184
9	18.969023	9.136584
10	17.296786	9.214277
11	17.062522	9.22587
12	16.162297	9.270889
13	15.725275	9.295865
15	15.570932	9.303038
20	15.535276	9.306381
30	15.611465	9.28979
40	15.641802	9.300466
50	15.6414	9.300597
60	15.655747	9.300248
100	15.83812	9.292405
300	15.593716	9.30313
500	15.813212	9.293764
700	16.188686	9.270835
1000	16.288148	9.265574
1500	16.81193	9.237142
2000	17.100283	9.226665

Експериментальні данні отримані для задачі балансування подвійного
оберненого маятника з розміром популяції 2000 ($N = 2000$)

Кількість воркерів	Час однієї епохи, мс	Розрахований показник (EfficiencyScore)
1	287.55262	6.838564
2	151.133212	7.189984
3	106.772777	7.395262
4	80.947856	7.567441
5	65.594256	7.703673
6	55.012554	7.821454
7	47.667895	7.920089
8	42.849208	7.995069
9	38.602173	8.069896
10	34.93581	8.14276
11	33.783133	8.167553
12	32.680992	8.192211
13	31.652684	8.216121
15	30.868966	8.234969
20	30.69161	8.239313
30	30.828122	8.235967
40	30.75774	8.23769
50	30.810563	8.236397
60	30.637711	8.240639
100	30.830138	8.235918
300	31.17827	8.227464
500	31.357623	8.223152
700	31.652879	8.216117
1000	32.296355	8.201049
1500	32.241891	8.202311
2000	33.221185	8.180005
2500	32.976582	8.185503

Експериментальні данні отримані для задачі балансування подвійного
оберненого маятника з розміром популяції 3000 ($N = 3000$)

Кількість воркерів	Час однієї епохи, мс	Розрахований показник (EfficiencyScore)
1	361.070872	7.008884
2	186.778208	7.388135
3	128.114827	7.624083
4	95.750234	7.8169
5	78.277505	7.95613
6	66.2066	8.075695
7	57.51457	8.178985
8	51.592895	8.260552
9	47.345881	8.326201
10	42.98408	8.40132
11	41.344327	8.431934
12	39.442038	8.469309
13	38.10792	8.496823
15	37.187085	8.51649
20	37.036837	8.519754
30	37.132592	8.517672
40	37.11789	8.517991
50	37.005417	8.520439
60	36.95755	8.521483
100	37.421948	8.511419
300	37.372744	8.512479
500	37.58701	8.507878
700	38.056061	8.497916
1000	38.476437	8.48911
1500	38.58492	8.486856
2000	39.610245	8.465919
2500	39.867172	8.45597
3000	39.963953	8.458845
3500	40.817274	8.442082

Експериментальні данні отримані для задачі балансування подвійного
оберненого маятника з розміром популяції 5000 ($N = 5000$)

Кількість воркерів	Час однієї епохи, мс	Розрахований показник (EfficiencyScore)
1	986.938639	7.398835
2	500.160863	7.790612
3	338.788686	8.034446
4	254.592656	8.223212
5	206.286003	8.367993
6	171.713188	8.498445
7	149.135408	8.601494
8	131.333751	8.696579
9	117.645083	8.78063
10	107.807133	8.84848
11	103.492456	8.880576
12	99.267378	8.91357
13	97.305678	8.929456
15	94.126411	8.956022
20	93.875	8.958168
30	93.983077	8.957245
40	94.425832	8.953475
50	94.607273	8.951937
60	94.8109	8.950214
100	94.275667	8.954751
300	94.687262	8.951259
500	95.069143	8.948036
700	94.944338	8.949088
1000	95.204058	8.9469
1500	96.195654	8.938614
2000	96.428772	8.93668
2500	97.218259	8.930173
3000	98.102922	8.922955
4000	99.32552	8.913105
5000	100.049738	8.907337

ДОДАТОК Б – ДОКУМЕНТИ, ЩО ПІДТВЕРДЖУЮТЬ ВПРОВАДЖЕННЯ РЕЗУЛЬТАТІВ ДИСЕРТАЦІЙНОГО ДОСЛІДЖЕННЯ

ЗАТВЕРДЖУЮ
Директор ТОВ Н'ЮГРАУНД
Є.С. РОДІН

29 січня 2026 року

АКТ
про впровадження результатів дисертаційного дослідження
ОМЕЛЬЯНЕНКО Ярослава Вікторовича

на здобуття наукового ступеня доктора філософії
за спеціальністю 122 – комп'ютерні науки

Комісія у складі: голови комісії – директора ТОВ Н'ЮГРАУНД – Родіна Є.С.
членів комісії – директора з розвитку перспективних напрямлень Білоконь В.Г.,
головного аналітика відділу інформатизації Немоловського О.В.
встановила та цим актом засвідчує, що нижчеперелічені наукові положення, а
саме:

1. Вперше запропоновано реалізацію моделі розподілених обчислень Master-Worker для алгоритму NEAT, що базується виключно на вбудованих механізмах конкурентного виконання мови програмування GO та забезпечує ефективне масштабування на багатоядерних системах з мінімальними комунікаційними накладними витратами.
 2. Удосконалено модель оцінки ефективності побудови топології штучних нейронних мереж алгоритмом NEAT на основі аналізу часових витрат, у межах якої вперше формалізовано зв'язок між структурною складністю мережі, розміром популяції, кількістю воркерів та продуктивністю апаратної платформи.
 3. Вперше запропоновано комплексну інтегральну метрику оцінки продуктивності алгоритму NEAT, яка поєднує часові характеристики, швидкість збіжності еволюції та якість фінального рішення, що дозволяє об'єктивно порівнювати різні реалізації та конфігурації нейроеволюційних алгоритмів.
 4. Вперше запропоновано метод оптимізації алгоритму NEAT для підвищення швидкості побудови топології ШНМ, що базується на інтеграції паралельної моделі Master-Worker для паралельної оцінки придатності, моделі часових витрат та інтегральної метрики ефективності отриманих рішень.
- розроблені особисто Омеляненко Ярославом Вікторовичем та застосовані при розробках систем моделювання потоків громадського транспорту з метою оптимізації маршрутів. Роботи виконуються для замовників з США.

Даний акт не є підставою для фінансових зобов'язань.

Голова комісії

Члени комісії



Є.С. РОДІН

В.Г. БІЛОКОНЬ

О.В. НЕМОЛОВСЬКИЙ



НКЕК
ДЕРЖАВНЕ ПІДПРИЄМСТВО
«УКРАЇНСЬКИЙ ДЕРЖАВНИЙ ЦЕНТР РАДІОЧАСТОТ»

проспект Берестейський, 151, м. Київ, 03180
тел.: (044) 422-81-01, e-mail: centre@ucrf.gov.ua, http://www.ucrf.gov.ua,
код ЄДРПОУ 01181765

26.01.2026 № 80/14-16

На № _____ від _____

Директору Інституту програмних
систем член-кореспонденту
НАН України
Ігорю СІНІЦИНУ

На вих. від 21.01.2026 №239/1-23

Шановний пане Ігоре!

В УДЦР було опрацьовано лист ІПС НАНУ від 21.01.2026 № 239/1-23. За результатом опрацювання повідомляємо наступне.

Під час виконання ДКР, шифр «ПОКРОВА», у якості співвиконавця, було залучено Інститут програмних систем Національної академії наук України для реалізації складової частини ДКР (СЧДКР).

В ході виконання СЧДКР фахівцями ІПС НАНУ було спроектоване, розроблене та поставлене спеціальне програмне забезпечення, в якому, в тому числі, реалізовано:

модель розподілених обчислень Master-Worker для алгоритму NEAT, що забезпечує ефективне масштабування на багатоядерних системах з мінімальними комунікаційними накладними витратами;

удосконалену модель оцінки ефективності побудови топології штучних нейронних мереж алгоритмом NEAT на основі аналізу часових витрат, у межах якої вперше формалізовано зв'язок між структурною складністю мережі, розміром популяції, кількістю воркерів та продуктивністю апаратної платформи;

комплексну інтегральну метрику оцінки продуктивності алгоритму NEAT, яка поєднує часові характеристики, швидкість збіжної еволюції та якість фінального рішення, що дозволяє об'єктивно порівнювати різні реалізації та конфігурації неревольюційних алгоритмів;

метод оптимізації алгоритму NEAT для підвищення швидкості побудови топології ШНМ, що базується на інтеграції паралельної моделі Master-Worker для паралельної оцінки придатності, моделі часових витрат та інтегральної метрики ефективності отриманих рішень.

На підтвердження реалізації зазначених моделей та методів складено відповідний акт, що додається.

Додаток: Акт про впровадження результатів дисертаційного дослідження ОМЕЛЬЯНЕНКА Ярослава Вікторовича – єдиний примірник, на 1 арк., тільки на адресу.

З повагою
Генеральний директор

Володимир КОРСУН

АКТ
про впровадження результатів дисертаційного дослідження
ОМЕЛЬЯНЕНКО Ярослава Вікторовича

на здобуття наукового ступеня доктора філософії
за спеціальністю 122 – комп'ютерні науки

Державне підприємство "Український державний центр радіочастот" цим актом засвідчує, що під час виконання СЧДКР, шифр «ПОКРОВА», нижчеперелічені наукові положення, що розроблені особисто Омеляненко Ярославом Вікторовичем, а саме:

- **реалізацію моделі розподілених обчислень Master-Worker для алгоритму NEAT**, що базується виключно на вбудованих механізмах конкурентного виконання мови програмування та забезпечує ефективне масштабування на багатоядерних системах з мінімальними комунікаційними накладними витратами.

- **удосконалену модель оцінки ефективності побудови топології штучних нейронних мереж алгоритмом NEAT на основі аналізу часових витрат**, у межах якої вперше формалізовано зв'язок між структурною складністю мережі, розміром популяції, кількістю воркерів та продуктивністю апаратної платформи.

- **комплексну інтегральну метрику оцінки продуктивності алгоритму NEAT**, яка поєднує часові характеристики, швидкість збіжності еволюції та якість фінального рішення, що дозволяє об'єктивно порівнювати різні реалізації та конфігурації нейроеволюційних алгоритмів.

- **метод оптимізації алгоритму NEAT для підвищення швидкості побудови топології ШНМ**, що базується на інтеграції паралельної моделі Master-Worker для паралельної оцінки придатності, моделі часових витрат та інтегральної метрики ефективності отриманих рішень,

були застосовані при розробці та впровадженні систем моделювання поведінки повітряних загроз.

Даний акт не є підставою для фінансових зобов'язань.

Керівник ДКР



Сергій СИРОВЕЦЬ