Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
(повна назва)

Кафедра _____Інформаційних управляючих систем_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

_____Дослідження моделей використання засобів штучного інтелекту_____
_____для планування ІТ-проєктів_____
(тема)

Виконав:
студент _2_ курсу, групи _УПГІТм-22-1____
___Краснєнков Ілля Олександрович_____
(прізвище, ім'я, по батькові)
Спеціальність ____122 Комп'ютерні_____
_____науки_____
(код і повна назва спеціальності)
Тип програми ____освітньо-наукова_____
(освітньо-професійна або освітньо-наукова)
Освітня програма _ Управління проектами в галузі інформаційних технологій_____
(повна назва освітньої програми)

Керівник проф. каф. ІУС Максим ЄВЛАНОВ_
(посада, власне ім'я, прізвище)

Допускається до захисту

Зав. кафедри _____ ___Костянтин ПЕТРОВ__
(підпис) (власне ім'я, прізвище)

2024 р.

Харківський національний університет радіоелектроніки

Факультет _____Комп'ютерних наук_____
Кафедра _____Інформаційних управляючих систем_____
Рівень вищої освіти _____другий (магістерський)_____
Спеціальність _____122 Комп'ютерні науки_____
(код і повна назва)
Тип програми __освітньо-наукова_____
(освітньо-професійна або освітньо-наукова)
Освітня програма_Управління проектами в галузі інформаційних технологій
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_01__»__квітня__ 20 24_ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові_____Краснєнкову Іллі Олександровичу_____
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження моделей використання засобів штучного інтелекту для планування ІТ-проєктів
_____
затверджена наказом університету від _01 квітня_2024_ р. № _№ 258 Cm_
2. Термін подання студентом роботи до екзаменаційної комісії _03_ _06_ 2024 р.
3. Вихідні дані до роботи Науково-технічні публікації та інтернет джерела з тематики кваліфікаційної роботи
4. Перелік питань, що потрібно опрацювати в роботі аналіз використання методів штучного інтелекту в плануванні ІТ-проєктів; дослідження модифікації генетичного алгоритму для розв'язання задачі планування ІТ-проєктів з обмеженими ресурсами; реалізація гібридного алгоритму для розв'язання задачі планування ІТ-проєктів з обмеженими ресурсами; апробація результатів кваліфікаційної роботи
_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	Оцінка сучасного стану об’єкта дослідження	01.04.2024 – 05.04.2024	виконано
2	Огляд існуючих методів вирішення проблеми планування ІТ-проєктів з обмеженими ресурсами	06.04.2024 – 07.04.2024	виконано
3	Огляд існуючих варіантів вирішення проблеми планування ІТ-проєктів з обмеженими ресурсами	08.04.2024 – 09.04.2024	виконано
4	Огляд і аналіз існуючих новітніх технологій та їх алгоритмів вирішення проблеми планування ІТ-проєктів з обмеженими ресурсами	10.04.2024 – 15.04.2024	виконано
5	Постановка задачі дослідження	16.04.2024 – 17.04.2024	виконано
6	Аналіз використання методів штучного інтелекту в плануванні ІТ-проєктами	18.04.2024 – 24.04.2024	виконано
7	Дослідження модифікації генетичного алгоритму для розв’язання задачі планування ІТ-проєктів з обмеженими ресурсами	25.04.2024 – 30.04.2024	виконано
8	Реалізація гібридного алгоритму для розв’язання задачі планування проєктів з обмеженими ресурсами	01.05.2024 – 08.05.2024	виконано
9	Апробація результатів кваліфікаційної роботи	09.05.2024 – 15.05.2024	виконано
10	Оформлення пояснювальної записки	16.05.2024 – 20.05.2024	виконано
11	Розробка презентації	21.05.2024	виконано
12	Захист роботи	05.06.2024	виконано

Дата видачі завдання __01__ ___квітня__2024_ р.

Студент _____
(підпис)

Керівник роботи _____ _проф. каф. ІУС Максим ЄВЛАНОВ
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи: 116 сторінок, 15 рисунків, 10 таблиць, 21 формула, 2 додатка, 41 джерело.

АНАЛІЗ, ГЕНЕТИЧНИЙ АЛГОРИТМ, ГІБРИДНИЙ АЛГОРИТМ, ДІАГРАМА АКТИВНОСТЕЙ, ДОСЛІДЖЕННЯ, ПЛАНУВАННЯ ІТ-ПРОЄКТІВ, РЕАЛІЗАЦІЯ, САМООРГАНІЗАЦІЙНІ КАРТИ КОХОНЕНА, PYTHON, UML.

Об’єктом дослідження кваліфікаційної роботи є розробка гібридного алгоритму для оптимізації процесу рішення проблеми планування ІТ-проєктів з обмеженими ресурсами.

Мета роботи – дослідження доцільності використання та ефективності гібридного алгоритму для вирішення проблеми планування.

У кваліфікаційній роботі як методи дослідження було використано спостереження, наукові факти, порівняння, експеримент та аналіз.

У результаті виконання кваліфікаційної роботи було запропоновано використання нового підходу до вирішення проблеми планування ІТ-проєктів з обмеженими ресурсами, а саме гібридний алгоритм на основі комбінації генетичного алгоритму та самоорганізаційних карт Кохонена. Після проведення експериментального дослідження роботи гібридного алгоритму на основі тестових даних, було отримано результати, що демонструють обмежене покращення.

Даний напрямок має перспективи для подальшого дослідження та вимагає більше детального аналізу в рамках оптимізації вхідних параметрів для налагодження ефективної роботи гібридного алгоритму.

ABSTRACT

Explanatory note to the qualification work: 116 pages, 15 pictures, 10 tables, 21 formulas, 2 appendices, 41 sources.

ACTIVITY DIAGRAM, ANALYSIS, IT-PROJECT PLANNING, GENETIC ALGORITHM, HYBRID ALGORITHM, IMPLEMENTATION, KOHONEN SELF ORGANIZING MAPS, PYTHON, RESEARCH, UML.

The research object of the qualification work is the development of a hybrid algorithm for optimizing the process of solving the resource constrained IT project scheduling problem.

The purpose of the work is to study the feasibility of using and the efficiency of the hybrid algorithm for solving the planning problem.

Observation, scientific facts, comparison, experiment and analysis were used as research methods in the qualifying work.

As a result of the qualification work, the use of a new approach to solving resource constrained IT project scheduling problem was proposed, namely a hybrid algorithm based on a combination of the genetic algorithm and Kohonen self-organizing maps. After conducting an experimental study of the performance of the hybrid algorithm based on test data, results were obtained showing limited improvement.

This direction has prospects for further research and requires more detailed analysis within the framework of optimization of input parameters to establish the effective operation of the hybrid algorithm.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

ГА – генетичний алгоритм

ІТ – інформаційні технології

МН – машинне навчання

НМ – нейронні мережі

ОПМ – обробка природної мови

ШІ – штучний інтелект

CCM – critical chain method

CPM – critical path method

LFT – latest finish time

PMBOK – project panagement body of knowledge

RCPSP – resource-constrained project scheduling problem

SOM – self-organizing map

ВСТУП

У сучасному динамічному бізнес-середовищі більшість проєктів характеризуються високим рівнем складності та вимагають залучення значної кількості ресурсів. Ефективна організація проєкту передбачає оптимальний розподіл трудових та матеріальних ресурсів, що становить значний виклик для проєктних менеджерів у процесі формування плану проєкту. Особливо гостро ця проблема постає в галузі інформаційних технологій, де проєкти мають свою специфіку та вимагають застосування спеціалізованих підходів до планування та управління.

Проблема планування проєктів з обмеженими ресурсами є однією з ключових задач в управлінні проєктами. Залежно від пріоритетів та цілей проєкту, задача оптимізації може варіюватися від мінімізації загальної тривалості проєкту до оптимального розподілу ресурсів між роботами. Слід виділити дану проблему у контекстів проєктів у галузі інформаційних проєктів, оскільки вони мають власну специфіку, яка відрізняється від інших галузей. Незважаючи на різноманітність критеріїв оптимізації, складність процесу планування залишається високою, вимагає застосування ефективних методів розв'язання та немає універсального ефективного рішення.

З метою покращення ефективності розв'язання задачі планування проєктів, пропонується дослідити можливість інтеграції самоорганізаційних карт Кохонена з генетичним алгоритмом. Застосування SOM в комбінації з генетичним алгоритмом має потенціал до підвищення ефективності пошуку оптимальних рішень та запобігання передчасній збіжності алгоритму до локальних оптимумів.

Кваліфікаційна робота виконується згідно з державними стандартами [1]-[2].

1 АНАЛІЗ ВИКОРИСТАННЯ МЕТОДІВ ШТУЧНОГО ІНТЕЛЕКТУ В ПЛАНУВАННІ ІТ-ПРОЄКТІВ

1.1 Огляд процесів планування ІТ-проєктів

У сфері управління проєктами інформаційних технологій (ІТ) ефективні процеси планування мають першорядне значення для успіху. Вдалий проєкт починається з правильного планування, яке включає детального аналізу та стратегічного підходу для визначення обсягу робіт, ресурсів, часових рамок та вимог до якості.

Планування проєктів передбачає чималу кількість викликів, серед яких постає проблема встановлення плану робіт з обмеженими ресурсами. Існуючі засоби планування, а саме стандарти Project Management Body Of Knowledge (PMBOK), ISO 15288 та ISO 12207, мають значну кількість тверджень по ефективному менеджменту проєктів, але вони надають лише загальні рекомендації по плануванню проєктів та не вирішують виникаючі перешкоди.

Розвиток штучного інтелекту запроваджує багато засобів та інструментів, які можуть стати в нагоді вирішенню поставлених перепон, щодо ефективного планування проєктів.

Розглянемо як саме поставлено процеси планування ІТ-проєктів відповідно до загальних рекомендацій.

Планування ІТ-проєктів вимагає чіткого визначення мети, цілей, обсягу робіт, і основних зацікавлених сторін на початковому етапі. Стандарти ISO 15288 та ISO 12207 слугують основою для цього процесу, встановлюючи рамки для різноманітних процесів життєвого циклу систем і програмного забезпечення. Процес планування проєкта, згідно з цими стандартами, охоплює вироблення та координацію ефективних планів, визначення проєктних цілей та обмежень, управління ресурсами, встановлення структури розподілу робіт на основі системної архітектури, та планування технічного

управління. Ці процеси включають детальне планування задач, графіків виконання робіт, визначення критеріїв успіху, а також розподіл ролей і відповідальностей у межах проєктної команди.

Важливо також інтегрувати планування ризиків на ранньому етапі, що включає ідентифікацію, аналіз, та розробку стратегій реагування на потенційні ризики. Ефективне управління ризиками дозволяє мінімізувати потенційний негативний вплив на проєкт та його цілі. Регулярне переглядання прогресу проєкту та його відповідності до встановлених цілей допомагає вчасно ідентифікувати та коригувати відхилення, що забезпечує адаптивність та гнучкість управління проєктом. Всі ці елементи разом формують основу для створення стійкого та успішного IT-проєкту.

На рисунках 1.1 та 1.2 наведено процеси планування проєкту та їх декомпозицію, відповідно до поставленої задачі дослідження у даній роботі.

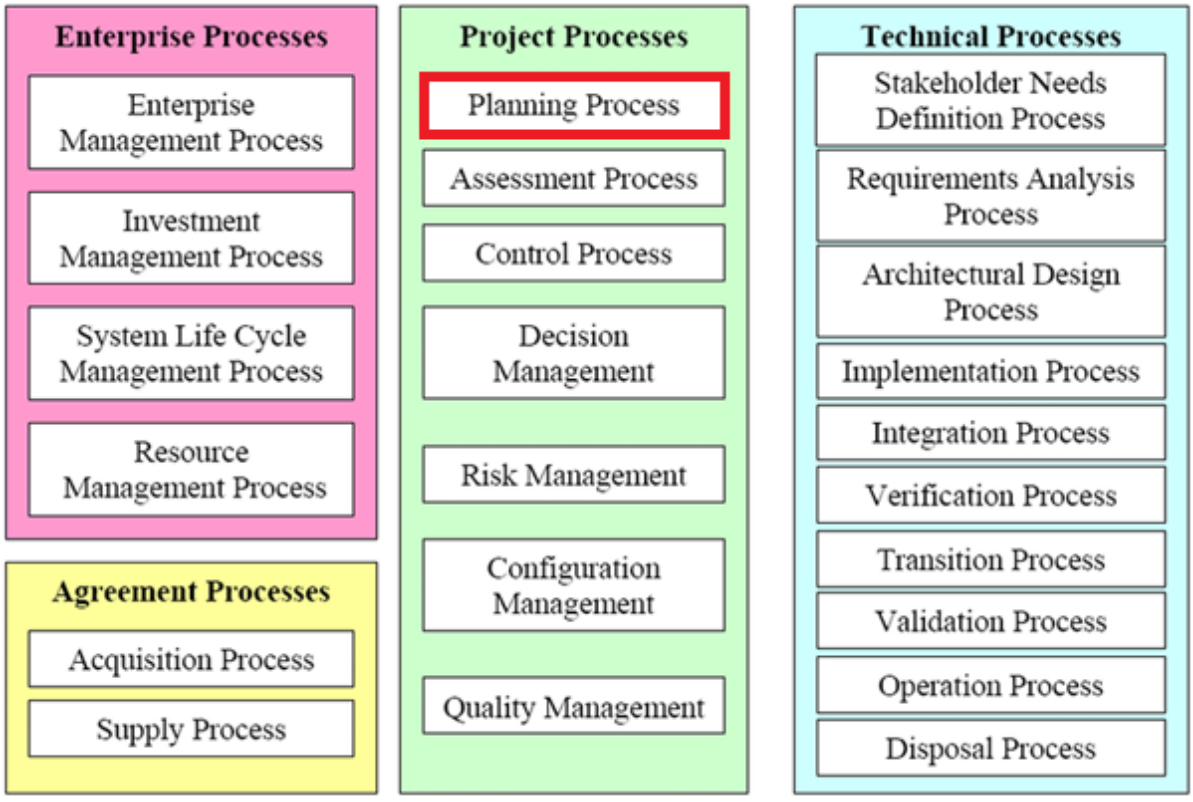

Рисунок 1.1 – Процеси планування проєкту відповідно до стандарту ISO 15288

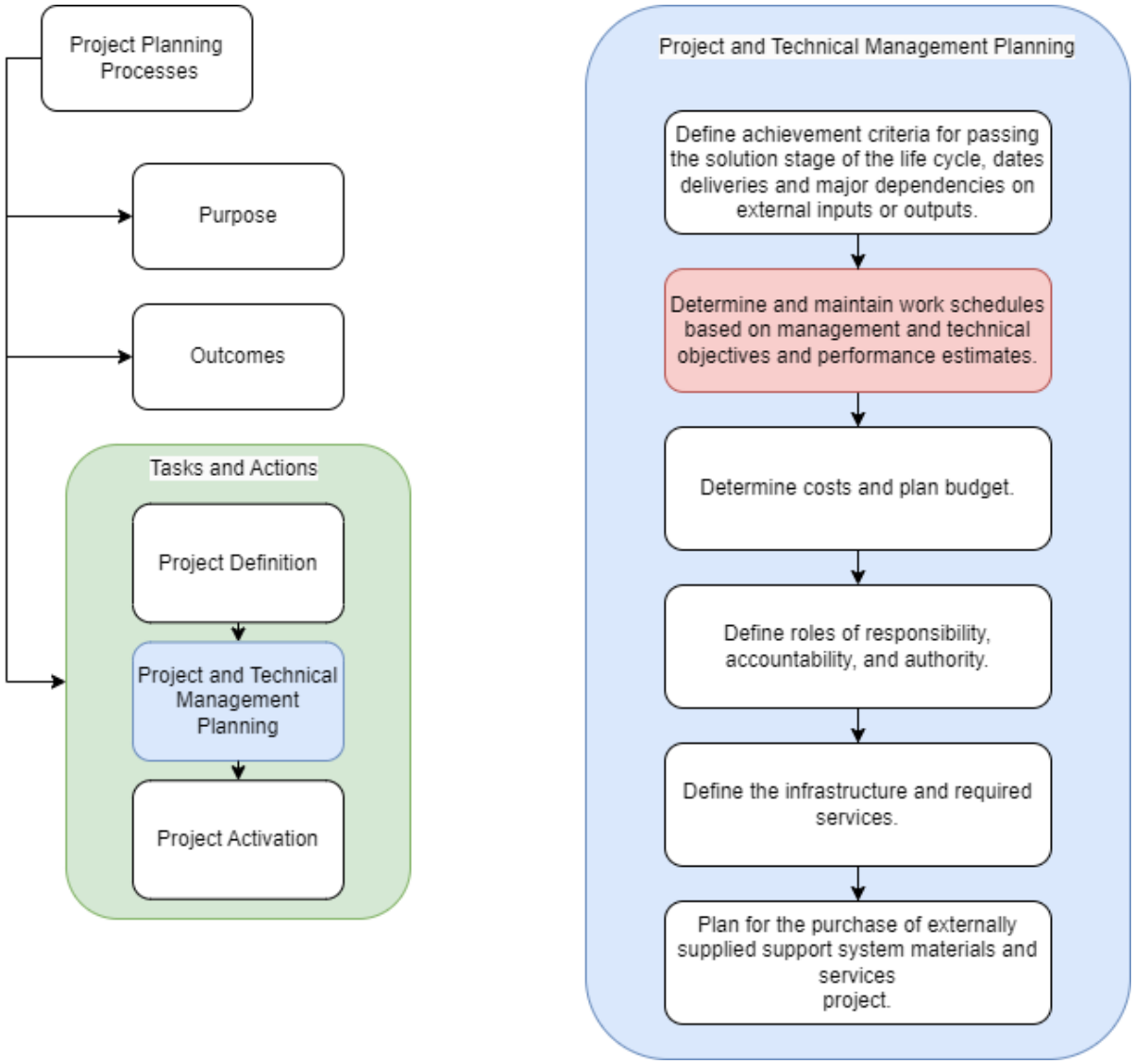

Рисунок 1.2 – Декомпозиція процесів планування проєкту відповідно до стандарту ISO 15288

У сфері планування проєктів інформаційних технологій методології, визначені стандартами ISO 15288 та ISO 12207, є типовою структурою, кожна з яких пропонує комплексний підхід до управління життєвим циклом системи та програмного забезпечення відповідно. ISO 15288 фокусується на проєктуванні систем, представляючи структурований процес, який охоплює всі фази життєвого циклу системи, від концепції до утилізації. Він наголошує на інтеграції різних дисциплін і процесів для забезпечення ефективної та ефективної доставки системи. У цьому стандарті описано набір процесів, розділених на чотири групи: процеси узгодження, процеси, що сприяють

організаційному проєкту, процеси проєкту та технічні процеси. Ці процеси призначені для адаптації до потреб конкретних проєктів, сприяючи гнучкості та налаштуванню.

У свою чергу, ISO 12207 націлений на процеси життєвого циклу програмного забезпечення, надаючи основу для управління розробкою, експлуатацією та обслуговуванням програмних продуктів. Він визначає набір дій і завдань, пов’язаних із розробкою програмного забезпечення, включаючи аналіз вимог, проєктування, впровадження, тестування, розгортання, експлуатацію, технічне обслуговування та, зрештою, утилізацію. ISO 12207 структуровано навколо трьох типів процесів: первинних процесів, допоміжних процесів та організаційних процесів, кожен з яких сприяє комплексному нагляду за проєктами розробки програмного забезпечення.

Обидва стандарти виступають за процесний підхід до управління проєктом, підкреслюючи важливість чітко визначених процесів, ролей і відповідальності протягом життєвого циклу проєкту. Їх об’єднує спільна мета — забезпечення якості, надійності та ефективності реалізації IT-проєктів. У той час як ISO 15288 забезпечує широку структуру, застосовну до будь-якого системного проєкту, ISO 12207 спеціально адаптується до нюансів розробки програмного забезпечення, пропонуючи детальні вказівки щодо управління складнощами, властивими проєктам програмного забезпечення [3-4].

Отже, стандарт ISO 12207 є доповненням до ISO 15288 орієнтуючись на проєкти з розробки інформаційних технологій. Відповідність багатьох пунктів між цими стандартами очевидна та це поширюється на процес формування графіку робіт, який розглядається у рамках цієї роботи.

1.2 Аналіз методів планування проєктів по часу та ресурсам

В управлінні IT-проєктами, методи оптимізації графіків відіграють критичну роль, дозволяючи керівникам проєктів максимально ефективно використовувати доступний час і ресурси. У сучасному швидкоплинному технологічному середовищі, де гнучкість і швидкість є ключовими для успіху, здатність точно планувати і оптимізувати робочі процеси стає вирішальним фактором. Ці методи не тільки допомагають забезпечити виконання проєктів у встановлені терміни та в рамках бюджетних обмежень, але й сприяють підвищенню якості кінцевого продукту за рахунок більш раціонального розподілу ресурсів та ефективного управління ризиками.

1.2.1 Метод критичного шляху

Метод критичного шляху (CPM) є фундаментальним інструментом управління проєктами, який використовується для планування та координації складних проєктів. Цей метод дозволяє визначити необхідний мінімальний час для завершення проєкту та ідентифікувати завдання, які безпосередньо впливають на загальний термін виконання проєкту [8].

При застосуванні CPM, спочатку визначаються всі діяльності, які потрібно виконати у рамках проєкту, після чого оцінюється тривалість кожної діяльності. На основі цієї інформації створюється мережевий графік, що демонструє взаємозалежності між різними діяльностями. Через цей графік можна простежити критичний шлях - послідовність завдань, які безпосередньо впливають на дату закінчення проєкту [9].

На рисунку 1.3 визначено критичний шлях червоним кольором, який пролягає через завдання F, G. Ці завдання формують критичний шлях, оскільки будь-яке затримання в цих завданнях призведе до затримки усього проєкту. Завдання на критичному шляху не мають запасу часу; тобто, їх «slack» або «float» дорівнює нулю, як показано на зображенні. Завдання, які не є частиною критичного шляху, мають певний запас часу, який дозволяє затримку без впливу на загальний графік проєкту.

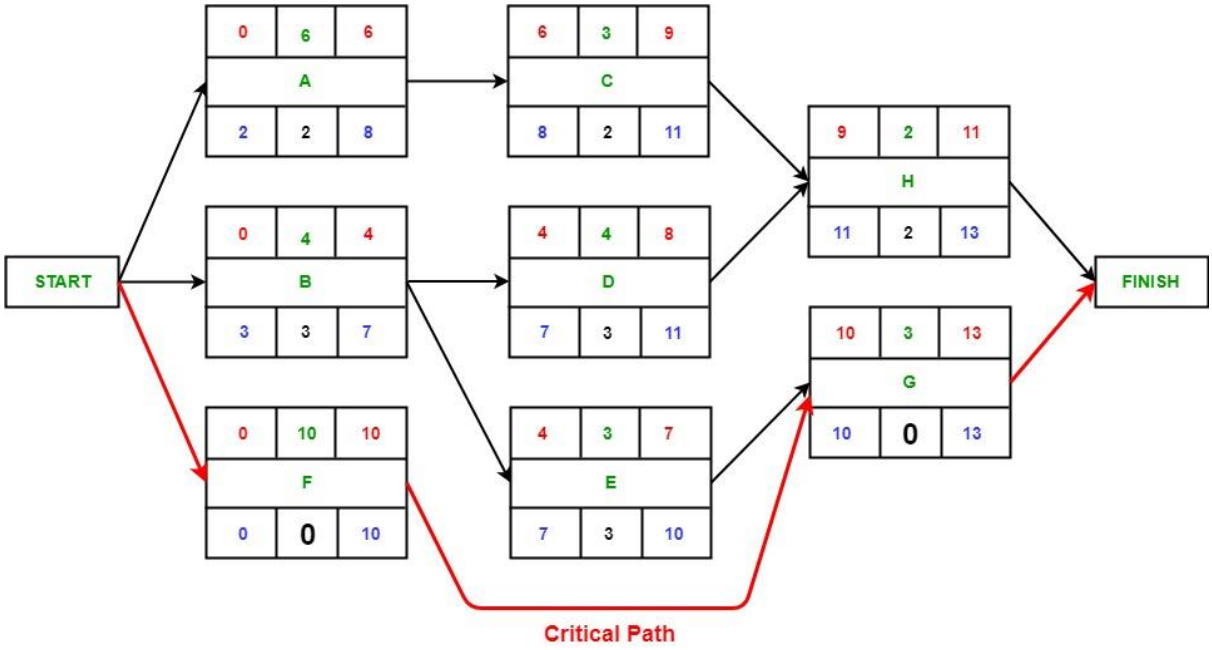

Рисунок 1.3 – Метод критичного шляху

1.2.2 Метод критичних ланцюгів

Метод критичних ланцюгів (CCM) — це підхід до управління проєктами, який фокусується на ресурсах, необхідних для виконання проєктних завдань. Він забезпечує просте використання і допомагає менеджеру зробити завдання легшими, заощаджуючи кінцевий час, а також зберігає споживчі витрати [10].

CCM використовується для ідентифікації критичного ланцюга завдань, що впливають на тривалість проєкту, аналогічно до критичного шляху в CPM. Однак, відмінність полягає в тому, що CCM бере до уваги не тільки послідовність завдань, але й доступність та використання ресурсів, необхідних для їхнього виконання. Такий підхід дозволяє більш реалістично оцінювати тривалість завдань, враховуючи можливі затримки через обмеження ресурсів.

Крім того, CCM вводить поняття буферів часу (часових резервів), які додаються до не критичних завдань, а також до кінця проєкту для забезпечення гнучкості у випадку змін або непередбачених затримок. Це забезпечує «захисний шар» для графіка проєкту, дозволяючи керувати ризиками більш ефективно.

На рисунку 1.4 показано приклад мережевої діаграми з критичним ланцюгом. На ньому зазначено критичний ланцюг, послідовність задач, які йому відповідають та буфери часу. Завдання на критичному ланцюзі є тими, що безпосередньо впливають на загальний графік проєкту, і будь-яка затримка в цих завданнях може призвести до затримки всього проєкту.

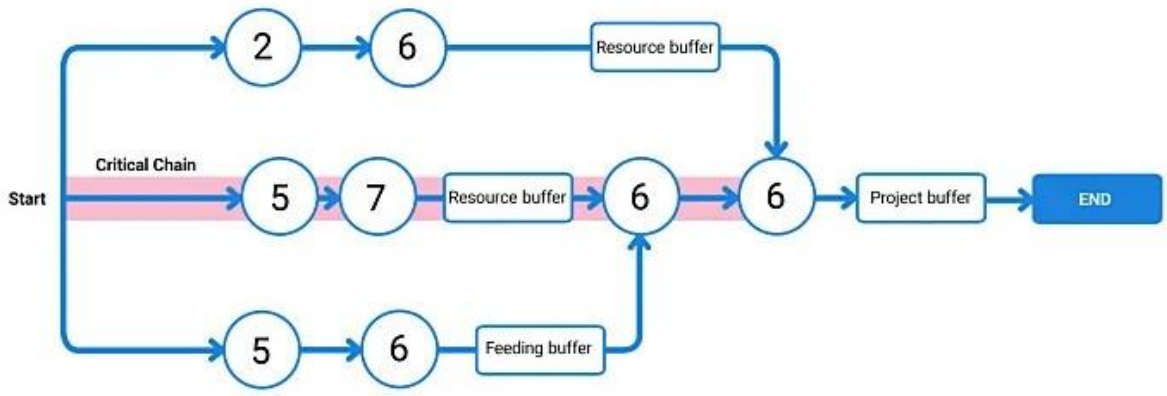

Рисунок 1.4 – Метод критичних ланцюгів

1.2.3 Генетичні алгоритми

Зусилля керівництва щодо планування та контролю графіків часто затьмарюються неадекватністю координації та розподілу ресурсів у плануванні.[27] Задля запобігання таких обставин слід розглянути інші, сучасні методи, які можуть задовільнити потреби проєктного менеджменту у певних умовах. До таких методів відносяться генетичні алгоритми (ГА) — це методи пошуку та оптимізації, які наслідують принципи природного відбору та спадковості. Вони використовуються для вирішення складних завдань шляхом створення «популяції» потенційних рішень, оцінки їх ефективності та використання найкращих «генів» для формування нових рішень.

На рисунку 1.5 представлено умовну схему генетичного алгоритму.

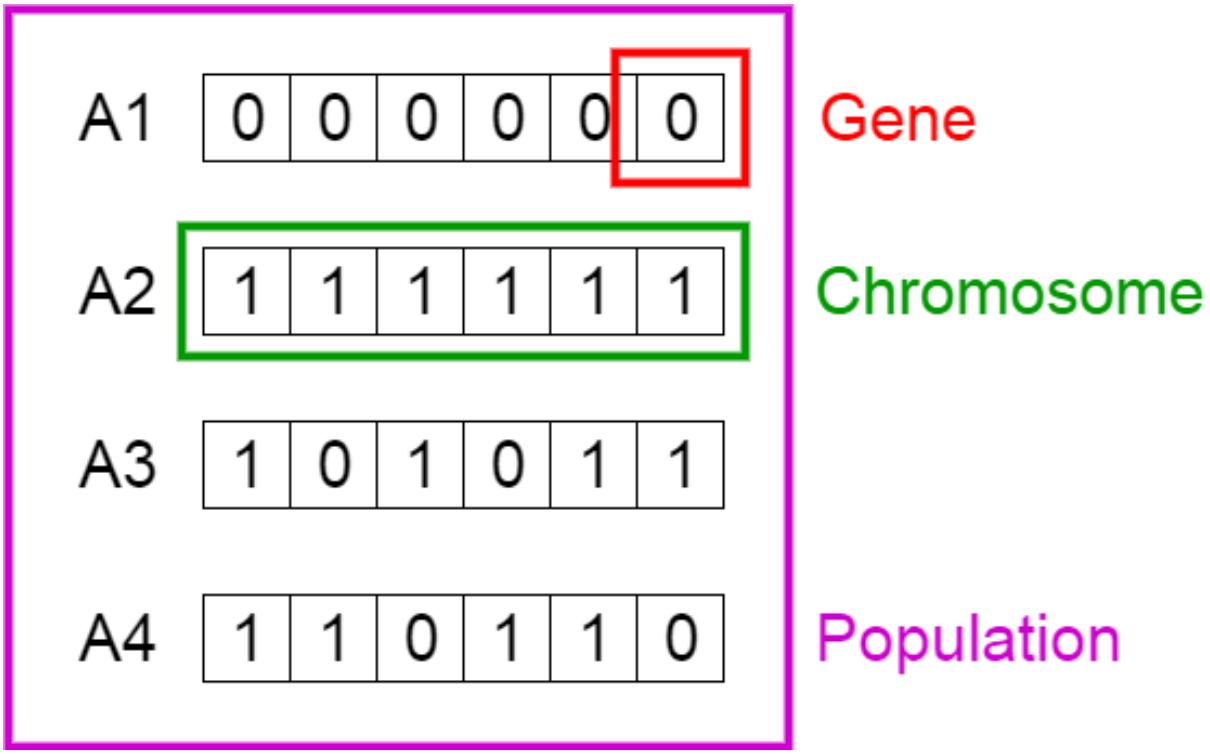

Рисунок 1.5 –Умовна схема генетичного алгоритму

Кожен рядок може представляти «хромосому» або конкретне рішення, де «гени» (елементи хромосоми) визначають параметри рішення. «Популяція» складається з багатьох таких хромосом. В процесі еволюції відбувається відбір найкращих хромосом, їхнє схрещування та мутація для створення нових поколінь рішень, які краще адаптовані до вирішення задачі. Через багато таких поколінь алгоритм збільшує шанси знаходження оптимального або наближеного до оптимального рішення.

У контексті оптимізації графіків проєктів, генетичні алгоритми можуть бути використані для виявлення найбільш ефективного порядку завдань, розподілу ресурсів, або графіка, що забезпечує виконання проєкту в мінімальний час або з мінімальними витратами.

Для ілюстративного порівняння методів оптимізації графіків проєктів - CPM, CCM, та генетичних алгоритмів, створимо порівняльну таблицю.

Таблиця 1.1 – Порівняння методів оптимізації графіків проєктів

Критерій	CPM	CCM	Генетичні алгоритми
1	2	3	4
Базовий принцип	Планування за найтривалішою послідовністю завдань	Планування, враховуючи обмеження ресурсів та введення буферів часу	Імітація природного відбору для знаходження оптимальних рішень
Застосування	Прості та помірно складні проєкти	Складні проєкти зі значними ресурсними обмеженнями	Складні та динамічні проєкти

Кінець таблиці 1.1

1	2	3	4
Складність впровадження	Низька	Середня	Висока
Гнучкість	Низька	Середня	Висока
Оптимізація	Орієнтована на час	Орієнтована на ресурси	Декілька параметрів
Управління ризиками	Обмежено, не враховує невизначеність	Обмежено, фокусується на ресурсах	Висока адаптивність

Аналіз таблиці показує, що CPM найкраще підходить для проєктів, де важливо слідувати фіксованому графіку, і де зміни та ресурси є передбачуваними. CCM краще підійде для проєктів, де існують значні ресурсні обмеження і потреба у більшій гнучкості у плануванні. Генетичні алгоритми ефективні для складних проєктів, де потрібно адаптуватися до частих змін і де потрібен глибокий аналіз для знаходження оптимального рішення. Однак, вони вимагають більших ресурсів для реалізації та аналізу.

1.2.4 Аналіз і критика Project Management Body Of Knowledge

PMBOK визнаний золотим стандартом в управлінні проєктами, пропонуючи кращі практики та рекомендації, які охоплюють усі аспекти проєкту [7]. Втім, складність та бюрократичність PMBOK часто стають предметом критики, особливо в контексті швидкоплинних та гнучких

проєктів. Цей стандарт може виявитися складним для адаптації у малих організаціях або в умовах, що швидко змінюються.

На рисунку 1.6 наведено життєвий цикл проєкту.

Рисунок 1.6 – Стандарт керування проєктами PMBOK

Дивлячись на сучасні умови існує потреба в інтеграції технологій, щоб подолати розрив між тим, що вимагає сучасне середовище управління проєктами (стабільним і процвітаючим на бізнес-ринку), і тим, що здатні традиційні інструменти планування управління проєктами. зробити або забезпечити [11].

Генетичні алгоритми пропонують альтернативний підхід, дозволяючи адаптуватися до змін і оптимізувати процеси управління проєктами. Використовуючи принципи еволюції, такі алгоритми дозволяють об'єднувати різні задачі проєкту в єдиний оптимізаційний процес, вибудовуючи максимально ефективний план реалізації.

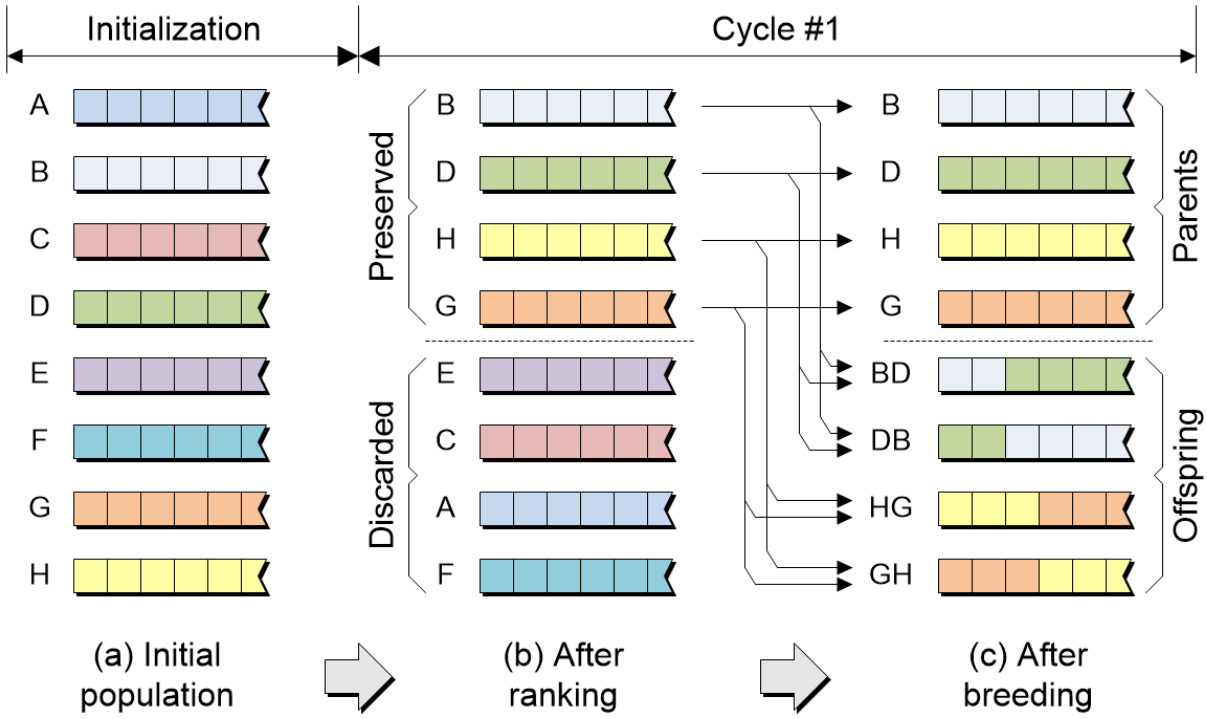

Рисунок 1.7 – Генетичний алгоритм

Недоліки рекомендацій PMBOK виявляються у їх недостатній гнучкості для динамічного управління проєктами. PMBOK наголошує на строгій структурі та послідовності етапів проєкту, що може обмежити спроможність команди швидко реагувати на зміни в умовах проєкту. В той же час, методи, засновані на генетичних алгоритмах, дозволяють інтегрувати гнучкість і адаптивність в процеси планування, виходячи за рамки традиційних підходів, і таким чином забезпечують більшу ефективність управління проєктами.

1.3 Огляд сучасних засобів для планування IT-проєктів

1.3.1 Визначення поняття «штучний інтелект» та його класифікація

Штучний інтелект (ШІ) — це дисципліна комп'ютерних наук, що зосереджена на створенні систем, котрі можуть виконувати завдання, які

зазвичай вимагають людського розуму, такі як вивчення, визначення мови, розпізнавання образів та прийняття рішень. ШІ має за мету не лише імітувати людську поведінку, а й здатний до самонавчання та адаптації за допомогою алгоритмів машинного навчання.

ШІ поділяється на слабкий та сильний типи. Слабкий ШІ (або вузький) автоматизує конкретні завдання, зазвичай перевершуючи людей, але діючи в обмежених рамках. Сильний ШІ (або загальний штучний інтелект) описує моделі, які можуть імітувати людське навчання та мислення [9].

Існує певна кількість технік, за яким працює ШІ:

– машинне навчання (МН) – алгоритми, які дозволяють комп'ютерам навчатися на основі даних;

– обробка природної мови (ОПМ) – технології, спрямовані на взаємодію між комп'ютером та людиною за допомогою природної мови;

– нейронні мережі (НМ) – представляють собою набір алгоритмів, які оброблюють дані імітуючи структуру людського мозку;

– комп’ютерне бачення – машина на вході отримує зображення, відео чи візуальні медіа, зчитує інформацію з них, виокремлюючи лише те, що стосується поставленої задачі.

Отже, спектр задач, які ШІ здатен вирішувати дуже багатий. Такий інструмент може використовуватись у багатьох сферах людської діяльності. Для вирішення задачі планування ІТ-проєктів слабкий ШІ буде якісним засобом, який допоможе покращити результати планування за коротші терміни із врахуванням потенційних ризиків.

1.3.2 Аналіз способів застосування засобів штучного інтелекту для вирішення задачі планування проєктів з обмеженими ресурсами

Проблема планування проектів з обмеженими ресурсами (англ. Resource-Constrained Project Scheduling Problem, далі RCPSP) є класичною проблемою у плануванні проєктів. Хоча вона доволі глибоко досліджена, але все ж таки не має універсального рішення та потребує специфічних рішень відповідно до поставлених вимог. Динамічність проєктів, зміна кадрів, економічні та соціальні чинники впливають на формування графіку проєкту з урахування можливих ризиків.

Широкий спектр засобів ШІ дозволяє вирішувати поставлену задачу різними способами, враховуючи необхідні показники, які допоможуть закінчити проєкт у поставлені терміни з максимальною низкою виконаних задач.

Розглянемо деякі інструменти ШІ, які допомагають вирішити поставлену проблему.

МН – описує набір методів, які зазвичай використовуються для вирішення різноманітних проблем реального світу за допомогою комп’ютерних систем, які можуть навчитися вирішувати проблему замість того, щоб бути явно запрограмованими для цього [12].

Використання даного засобу для вирішення задачі планування може бути ефективним рішенням для великих проєктів, які мають схожу динаміку та загальні дані. Дана методика включає різні типи навчання, такі як навчання під наглядом, під частковим наглядом, без вчителя та навчання з підкріпленням.

Кожен з цих напрямків має власні переваги та перешкоди, наприклад, щоб перевірити правила, можна використовувати методи МН, але вони

потребують високоякісних навчальних даних, щоб адаптувати дані та сформувати точну надійну модель для точного прогнозування [13].

Навчання без вчителя дозволяє програмному забезпеченню власними силами відслідковувати патерни поведінки, які забезпечать прогнозування результатів, однак такий результат скоріш за все буде більш специфічним до якогось проєкту.

Нейронні мережі можуть використовуватись також для значної кількості задач. Одне із досліджень показує, що їх використання може поширюватись на створення інструменту підтримки рішень для планування робочої сили та графіку. Створена модель використовує методи підгонки для прогнозування попиту та виробництва, що є складним етапом для виробничих систем із непередбачуваним та нестабільним попитом [11].

Хоча нейронні мережі є ефективним засобом, при їх навчанні з’являються певні труднощі, які можуть знизити якість результатів та надавати не оптимальні рішення. До таких проблем належать погана якість навчальних даних, гіперпараметричні налаштування, градієнтні проблеми, комплексність проблеми тощо [14].

Евристичні алгоритми відносяться до класу алгоритмічних рішень, які у результаті видають не завжди оптимальний варіант рішення, але наближений до оптимального. До таких алгоритмів відносяться генетичні алгоритми, гармонічний пошук, гравітаційний пошук та багато інших [15].

Найбільша перевага такого типу алгоритмів полягає в їх ефективності у вирішенні поставлених задач. Як правило точні алгоритми вимагають значних обчислювальних ресурсів для пошуку найкращого рішення та іноді можуть бути непрактичними за обсягом свої моделей. Швидкість та гнучкість такого типу алгоритмів забезпечує широке використання у динамічних проєктах зі значними обсягами даних, які треба оптимізувати.

Нажаль, відсутність гарантії знаходження оптимального рішення робить такий тип невикористовуваним для специфічних задач. Ефективність

також залежить від домену, в якому відбувається оптимізація, оскільки поставлені задачі можуть вимагати значних конфігурацій, які зроблять алгоритми менш ефективними.

Тим не менш для вирішення проблеми не гарантованості оптимального рішення було розроблено гібридні алгоритми, які є результатом поєднання різних моделей для пошуку глобального оптимуму, що забезпечить найкращий результат своєї роботи. Існує велика кількість таких видів алгоритмів, які використовуються у багатьох сферах діяльності.

Порівняльну характеристику методів ШІ наведено у таблиці 1.2.

Таблиця 1.2 – Порівняльна характеристика методів ШІ

Підхід	Переваги	Недоліки	Придатність застосування
МН	Точні прогнози тривалості активності; Керований даними	Залежить від якості даних; Непрямий підхід до вирішення	Великі проєкти з історичними даними
НМ	Потужне розпізнавання патернів; Моделювання складних відносин	Вимагає великих навчальних даних; По природі «Чорна скринька»	Складні проєкти, зі специфічною діяльністю
Евристичний	Гнучкий і адаптивний; Ефективний для задовільних рішень	Не завжди оптимальний результат; Потрібне налаштування параметрів	Проєкти, що вимагають швидких практичних рішень

Гібридний	Поєднує в собі сильні сторони різних методів; Покращена якість рішень	Складність інтеграції та налаштування	Комплексні та масштабні проєкти

За результатами проведеного аналізу певного висновку щодо визначення найкращого методу дати не можна, оскільки кожен з них виступає краще серед інших при виконанні певних умов.

Задача планування проєкту з обмеженими ресурсу представляє собою вирішення проблеми з динамічними показниками та має вирішуватись постійно для коригування графіку задач та налаштування під них відповідних ресурсів. МН та НМ можуть бути краще бути використані на великих, довготривалих проєктах, які мають великий обсяг роботи та історичні дані. Гібридні алгоритми хоча й не вимагають історичних даних але є більш складними та можуть бути краще пристосовані для певних проєктів та не завжди будуть виступати у якості найкращого рішення. У свою чергу для стартапів та невеликих проєктів евристичні алгоритми були б найкращим способом вирішення задачі планування, оскільки б не вимагали багатьох конфігурацій та давали б приблизно оптимальні результати.

Далі у роботі буде розглянуто роботу гібридного алгоритму та його застосування для вирішення RCPSP.

1.4 Постановка задачі дослідження

Метою кваліфікаційної роботи є розробка та аналіз методики використання генетичних алгоритмів для оптимізації процесу планування в ІТ-проєктах, зокрема для розв'язання задачі RCPSP. Як було зазначено раніше, загально рекомендовані методики формування плану робіт не завжди є

ефективними для вирішення задач, тому дослідження цієї галузі буде актуальним напрямком для подальшого розвитку.

Основні завдання дослідження включають:

– аналіз сучасного стану планування в ІТ-проєктах – вивчення існуючих методів та інструментів планування, аналіз їх переваг та недоліків у контексті проєктів з обмеженими ресурсами;

– дослідження ефективності застосування генетичних алгоритмів для розв’язання RCPSP – розробка методики застосування генетичних алгоритмів, що включає алгоритмічні моделі, параметри оптимізації, процедуру навчання та оптимізації, а також аналіз результативності генетичних алгоритмів;

– розробка практичного рішення з використанням генетичного алгоритму для рішення RCPSP;

– проведення апробації генетичного алгоритму для рішення RCPSP на основі тестових даних.

Очікувані результати включають підвищення ефективності планування в ІТ-проєктах за рахунок застосування генетичних алгоритмів, мінімізацію загального часу виконання проєктів та оптимізацію використання ресурсів. Також очікується розширення можливостей управління проєктами через адаптацію до змінних умов і вимог, що забезпечить більшу гнучкість і динамічність у плануванні та виконанні проєктних робіт.

Це дослідження має на меті не лише підвищити ефективність управління ІТ-проєктами за допомогою інноваційних технологій штучного інтелекту, але й надати проєктним менеджерам потужний інструментарій для оптимізації робочих процесів. Генетичні алгоритми, завдяки своїй універсальності та адаптивності, відкривають нові перспективи для ефективного розв'язання складних задач планування, що є особливо актуальним у сучасних динамічних умовах ІТ-галузі.

2 ДОСЛІДЖЕННЯ МОДИФІКАЦІЇ ГЕНЕТИЧНОГО АЛГОРИТМУ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ ПЛАНУВАННЯ ІТ-ПРОЄКТІВ З ОБМЕЖЕНИМИ РЕСУРСАМИ

2.1 Опис задачі планування проектів з обмеженими ресурсами та математичне представлення

RCPSP - це класична задача комбінаторної оптимізації, яка має широке практичне застосування в управлінні проектами та виробництвом. Задача комбінаторної оптимізації визначається простором рішень X, який є дискретним або який можна звести до дискретного набору, а також підмножиною можливих рішень $\subseteq$, що відносяться до цільової функції [16].

Проект в контексті RCPSP представляється у вигляді множини робіт, для кожної з яких задані тривалість виконання та потреби в ресурсах. Ресурси, необхідні для виконання робіт, доступні в обмеженій кількості. Крім того, між роботами існують відношення передування, які визначають порядок їх виконання: деякі роботи можуть розпочатися лише після завершення певних інших робіт.

RCPSP належить до класу NP-складних задач, що означає відсутність точних алгоритмів її розв'язання за поліноміальний час для задач великої розмірності. Зі збільшенням кількості робіт та ресурсів обчислювальна складність задачі суттєво зростає.

Метою розв'язання задачі RCPSP є побудова допустимого розкладу виконання робіт проекту, який задовольняє обмеження на порядок виконання робіт і доступність ресурсів та оптимізує певний критерій ефективності, найчастіше - час завершення всього проекту.

В загальному вигляді RCPSP можна сформулювати наступним чином:

– набір активностей = , який включає фіктивні активності для початкової та кінцевої подій;

– набір тривалостей для кожної діяльності: = ;

– набір доступних ресурсів = , де m – номер кожного ресурсу;

– Rik – це кількість ресурсів k необхідних для активності i. Rk – це сумарна кількість доступного ресурсу k;

– – набір послідовностей та зв'язків між активностями. Активність i та активність j, визначають, що перша з них має спершу закінчитись, ніж почнеться наступна.

Для даної роботи буде розглянуто одну цільову функцію, яка ставить метою мінімізацію загального часу завершення проєкту, який є часом завершення останньої діяльності.

$$Cmax \quad , \qquad (2.1)$$

де – тривалість проєкту.

Оскільки врахування обмежень є критично важливим, то такі обмеження необхідно визначити. До таких обмежень відносяться відповідність до порядку виконання та обмеження за ресурсами.

Формула обмеження за порядком виконання записується наступним чином:

$$xj \geq xi + di, \quad \forall i,j \in P \quad , \qquad (2.2)$$

де – набір послідовностей та зв'язків між активностями;

xi – активність;

di – тривалість активності.

Формула обмеження за ресурсами:

$$i \in N rik \cdot uit \leq Rk, \quad \forall k \in \{1,...,m\} \quad , \tag{2.3}$$

де uit – використання активності і в часовий інтервал t;

Rk – це сумарна кількість доступного ресурсу k.

Час початку кожної діяльності має бути не меншим за 0:

$$xi \geq 0, \quad \forall i \in N \quad , \tag{2.4}$$

де – множина натуральних чисел.

У кваліфікаційній роботі розглядається модифікація генетичного алгоритму з поєднанням SOM для забезпечення різноманітності поколінь та способу пошуку глобального оптимуму при дотриманні вищевказаних обмежень на рішення задачі оптимізації.

2.2 Архітектурний опис генетичних алгоритмів

Генетичні алгоритми – це потужний метаевристичний підхід до вирішення складних оптимізаційних задач, який ґрунтується на принципах еволюції та природного відбору. ГА належить до класу еволюційних алгоритмів і має широке застосування в різноманітних галузях, таких як

штучний інтелект, машинне навчання, операційні дослідження та інженерія [17-18].

Основна ідея ГА полягає в моделюванні процесу еволюції шляхом роботи з популяцією потенційних розв'язків, представлених у вигляді хромосом. Кожна хромосома є закодованим представленням розв'язку і складається з послідовності генів, які можуть бути бінарними, дійсними числами або іншими типами даних, залежно від природи задачі.

Функція придатності відіграє ключову роль у ГА, оскільки вона визначає якість кожного розв'язку в популяції. Значення функції придатності використовується для керування процесом еволюції шляхом надання переваги більш придатним розв'язкам під час селекції батьківських особин для створення нащадків.

Оператор селекції в ГА відповідає за вибір батьківських особин з популяції на основі їх значень придатності. Існує кілька поширених методів селекції, таких як рулеткова селекція, турнірна селекція та ранжувальна селекція. Ці методи забезпечують баланс між дослідженням простору пошуку та використанням знайдених перспективних розв'язків.

Схрещування є ключовим оператором в ГА, який забезпечує обмін генетичною інформацією між батьківськими хромосомами для створення нащадків. Існують різні типи операторів схрещування, такі як одноточкове, двоточкове та рівномірне схрещування, кожен з яких має свої особливості та сфери застосування.

Оператор мутації в ГА вносить випадкові зміни в гени хромосом з певною ймовірністю. Мутація допомагає підтримувати різноманітність популяції та запобігає передчасній збіжності до локальних оптимумів. Ймовірність мутації зазвичай встановлюється на низькому рівні, щоб зберегти баланс між дослідженням та використанням знайдених розв'язків.

Популяція – це колекція потенційних рішень задачі, представлених у формі хромосом. Популяція записується як

$$t=\{x1,x2,...,xN\} \quad , \tag{2.5}$$

де – номер покоління;

$x\Lambda$ – індивіди.

Функція придатності використовується для оцінки якості рішення, а саме відповідності індивіда до поставленої умови. У даній роботі функція придатності має на меті мінімізацію тривалості проєкту:

$$\rightarrow \quad , \tag{2.6}$$

де R – множина значень для функції ;

X – область визначення функції .

Вибір особин з популяції на основі їх пристосованості для участі в репродукції (стають батьками). Це може бути представлено як функція відбору *S*, застосована до популяції:

$$Pt=\{y1,y2,...,yM\} \quad , \tag{2.7}$$

де $M \leq N$ – кількість обраних особин;

$y\Lambda$ – відібрані особини.

Кросовер застосовується до пар батьківських особин для створення потомства наступного покоління. Функція кросоверу *C* може бути представлена як:

$$yi,yj=zi,zj \quad , \tag{2.8}$$

де yi,yj – батьківські особини;

zi,zj – результати потомства.

Мутація вносить варіації у потомство та записується як:

$$zi=zi' \quad , \tag{2.9}$$

де zi – особина потомства перед мутацією;

zi – особина потомства після мутації.

Оновлення популяції – формується нове покоління особин, можливо, шляхом поєднання батьківських особин і потомства з подальшим відбором для підтримки розміру популяції:

$$t+1=UPt,\{z1',z2',...,zK'\} \quad , \tag{2.10}$$

де U — функція оновлення, що формує наступне покоління;

K — кількість потомства;

zk – особина потомства після мутації.

Ітераційний процес повторюється протягом T поколінь або до досягнення деякого критерію зупинки з метою еволюції особин до кращих рішень. На рисунку 2.1 зображено загальний алгоритм роботи генетичного алгоритму [19].

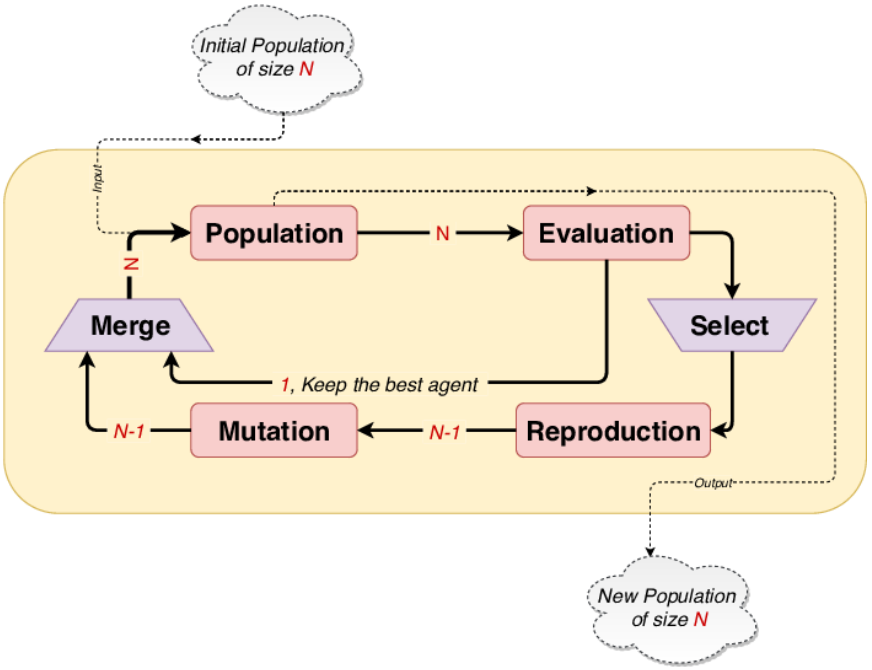

Рисунок 2.1 – Алгоритм роботи генетичного алгоритму

2.3 Опис самоорганізаційних карт Кохонена

Самоорганізаційна карта Кохонена (SOM) – це тип штучної нейронної мережі, яка використовує неконтрольоване навчання для відображення багатовимірних даних на низьковимірну дискретну карту [20]. SOM використовується в задачах кластеризації, візуалізації даних та зменшення розмірності.

Архітектура SOM складається з одношарової нейронної мережі, в якій нейрони організовані у вигляді двовимірної решітки. Кожен нейрон пов'язаний з усіма вхідними елементами через вагові коефіцієнти, які адаптуються у процесі навчання. Ключовою особливістю SOM є збереження

топологічних властивостей вхідних даних, тобто близькі вхідні вектори будуть відображатися на близькі нейрони на карті.

SOM можна застосовувати для вирішення складних проблем оптимізації, таких як планування проєкту, щоб мінімізувати його тривалість. Ці проблеми часто пов’язані з плануванням набору заходів із урахуванням послідовності та обмежень ресурсів. RCPSP є NP-складною проблемою, і евристичні підходи, такі як SOM, можна використовувати для пошуку майже оптимальних рішень.

Щоб застосувати SOM до проблем оптимізації, вхідний рівень представляє компоненти проблеми, а кожен нейрон у карті функцій представляє можливе рішення. Ваги зв’язків між вхідним шаром і картою функцій кодують змінні рішення. Під час навчання SOM вчиться відображати схожі рішення ближче одне до одного на карті функцій, тоді як різні рішення відображаються далі одне від одного.

Використання SOM для вирішення задач оптимізації пропонує кілька переваг. SOM можуть обробляти багатовимірні вхідні дані та фіксувати складні зв’язки між проблемними компонентами та обмеженнями. Властивість топологічного збереження SOM дозволяє досліджувати подібні рішення в просторі рішень, збільшуючи шанси знайти рішення близькі до оптимальних. Крім того, SOM можна виконувати паралельно, що робить їх обчислювально ефективними для великих проєктів.

Важливо зазначити, що SOM, як і інші евристичні підходи, не гарантують знаходження оптимального рішення для поточної проблеми. Якість рішень, отриманих за допомогою SOM, залежить від різних факторів, таких як розмір карти ознак, швидкість навчання, функція сусідства та кількість ітерацій навчання.

Поєднання SOM з ГА, яке пропонується у даній роботі може допомогти у вирішенні задачі зі збільшеною вірогідністю пошуку оптимального розкладу.

2.4 Опис гібридного алгоритму самоорганізуючих карт Кохонена та генетичного алгоритму

RCPSP включає в себе планування набору активностей з обмеженнями ресурсів і зв’язками пріоритету таким чином, щоб загальна тривалість проекту була мінімізована. Складність RCPSP полягає в його комбінаторній природі та наявності обмежень, які значно ускладнюють можливий простір пошуку.

У типовому ГА популяція рішень (розкладів) розвивається протягом поколінь за допомогою таких операторів, як відбір, кросовер і мутація. Мета полягає в тому, щоб ефективно дослідити простір рішень і знайти оптимальне або майже оптимальне рішення. Однак GA можуть стикатися з передчасною конвергенцією через втрату різноманітності, особливо в оманливих розкладах, які можуть зустрічатись у великих проєктах.

У даній роботі пропонується дослідити гібридний алгоритм SOM-GA для вирішення RCPSP, оскільки SOM може кластеризувати вектори вхідних рішень (хромосоми в ГА) на основі їхніх характеристик, які часто визначаються їх значеннями придатності та генетичним складом. Кожен вузол у SOM представляє кластер рішень, які схожі одне на одного, і ця подібність зменшується зі збільшенням відстані між вузлами на карті.

Шляхом відображення популяції в SOM утворюються різні кластери рішень. Кожен кластер представляє нішу в просторі рішень, яка містить генетично подібних індивідуумів. Такий спосіб кластеризації природним чином допомагає підтримувати різноманітність у популяції, оскільки еволюційні операції обмежені кластерами, зменшуючи ймовірність зближення всієї популяції до єдиного локального оптимуму.

Використання гібридного алгоритму надаватиме наступні переваги.

Покращене дослідження пошукового простору. SOM-GA покращує пошук завдяки здатності SOM картографувати популяції в різних нішах (кластерах). Це гарантує, що різні регіони простору рішення досліджуються. Локалізовані операції ГА в кожному кластері дозволяють проводити інтенсивний пошук поблизу перспективних ділянок.

Запобігання передчасній конвергенції. Просторова організація рішень завдяки SOM дозволяє здійснювати контрольований кросинговер і мутацію. Обмежуючи ці операції в межах кластерів та іноді між ними, генетичне різноманіття в популяції зберігається довше, таким чином зменшуючи ризик передчасної конвергенції.

Динамічне формування кластерів. На відміну від методів, де кластери часто визначені заздалегідь, SOM динамічно створює та коригує їх на основі поточної популяції. Цей динамічний аспект має вирішальне значення для адаптації до змін у динаміці популяції протягом поколінь.

Механізм зворотного зв’язку для ГА. SOM забезпечує механізм зворотного зв’язку для ГА, визначаючи недосліджені області в просторі рішень. Хромосоми, що відображаються в малонаселених вузлах на SOM, можуть бути націлені на відтворення (кросовер і мутація), таким чином спрямовуючи пошук ГА на менш досліджені регіони.

Хоча переваги, які надає гібридний алгоритм можуть дійсно покращити спосіб рішення RCPSP але він також має певні недоліки. Невеликим проєктам, які мають малий перелік задач для виконання, використання запропонованого підходу може бути не доцільним через його складність. Оскільки задача RCPSP NP-складна, то застосування такого підходу буде найбільш привабливим для великих проєктів, де кількість потенційних рішень дуже значна та ефективний метод пошуку допоможе у її рішенні.

2.5 Огляд існуючих робіт

Запропонований підхід використовує властивості SOM для кластеризації хромосом, створених генетичним алгоритмом під час процесу оптимізації для вирішення RCPSP. У кожному поколінні ГА виробляє популяцію хромосом, що представляють потенційні рішення. Потім ці хромосоми відображаються на SOM, де схожі за характеристиками індивіди розподіляються по кластерам. Особливість ГА полягає в проблемі пошуку глобального оптимуму. Через відсіювання індивідів, які мають гірші результати відповідно до функції придатності, то відбувається стагнація алгоритму, через що він застрягає у локальному оптимумі.

Даний підхід пропонує вирішити цю проблему та коли ГА починає стагнувати та зближується до локального оптимуму- застосовується метод для збільшення різноманіття хромосом шляхом їх вибору із різних кластерів для створення нових нащадків. Це допомагає ГА уникнути локальних оптимумів і продовжити дослідження простору пошуку, щоб знайти глобальний оптимум.

Інтеграцію SOM та ГА було досліджено для інших типів задач в інших наукових роботах.

У [21] автори розробили гібридний алгоритм GASOM, який використовує SOM для отримання даних з процесу еволюції та керування стратегією пошуку. Вони показали, що GASOM ефективно запобігає передчасній конвергенції та перевершує інші варіанти ГА в оманливих і мультимодальних проблемах.

Подібним чином, у [22] було запропоновано гібридний підхід до навчання з використанням кількох SOM та ГА. SOM групують індивідуумів у кожному поколінні, а ГА виконує операції в межах субпопуляцій, визначених кластерами. Їх результати показують, що цей підхід покращує ефективність

локального пошуку ГА та знаходить кращі рішення швидше, ніж стандартний ГА.

У контексті RCPSP використання SOM може надати цінну інформацію про структуру простору рішень. Об’єднуючи схожі рішення разом, SOM можуть визначити перспективні регіони, які ГА слід використовувати далі. У той же час, зберігаючи різноманітність шляхом відбору хромосом з різних кластерів, цей підхід може досліджувати ширший діапазон простору пошуку та уникнути потрапляння в пастку локальних оптимумів.

Робота [22] щодо кластерної проблеми комівояжера додатково підтверджує ідею використання кластеризації в поєднанні з ГА для задач планування. Хоча ця задача відрізняється від RCPSP, обидві проблеми передбачають пошук оптимальних послідовностей або розкладів із дотриманням певних обмежень. Автори показали, що їхній ГА, який включає в себе знання про конкретну проблему через спеціалізовані оператори, перевершує інші евристичні підходи.

Також ідею використання кластерів у комбінації з ГА для рішення RCPSP було розглянуто у роботах [24] та [25].

На основі наведеного аналізу робіт можна зробити висновок, що використання комбінації SOM та ГА є перспективним направленням у дослідницьких роботах, оскільки такий підхід може використовувати різноманітні техніки та комбінації, в результаті яких формування різноманітних індивідів у популяції ГА може призвести до виходу з локального оптимуму. Звісно таке поєднання може зменшити продуктивність роботи коду та бути менше ефективним порівняно зі спеціалізованими алгоритмами пошуку, але додаткова візуалізація кластерів завдяки SOM може бути корисною для користувачів, адже буде позначати розподіл індивідів. Слід зазначити, що оптимізація конфігурацій при навчанні SOM та ГА значно впливає на результативність підходу, що було показано в наведених наукових роботах.

2.6 Опис схеми реалізації гібридного алгоритму

Для реалізації гібридного алгоритму необхідно сформувати послідовність дій, що будуть виконані в процесі виконання. Візуалізація такої послідовності може бути виконана завдяки діаграмі активностей, що відображатиме взаємодію між різними компонентами системи.

Формування наочного алгоритму роботи алгоритму дозволить встановити як саме працює гібридний алгоритм та виділити нові кроки, що додаються для модифікації звичайного алгоритму.

Для цього було побудовано діаграму активностей, яку наведено на рисунку 2.2.

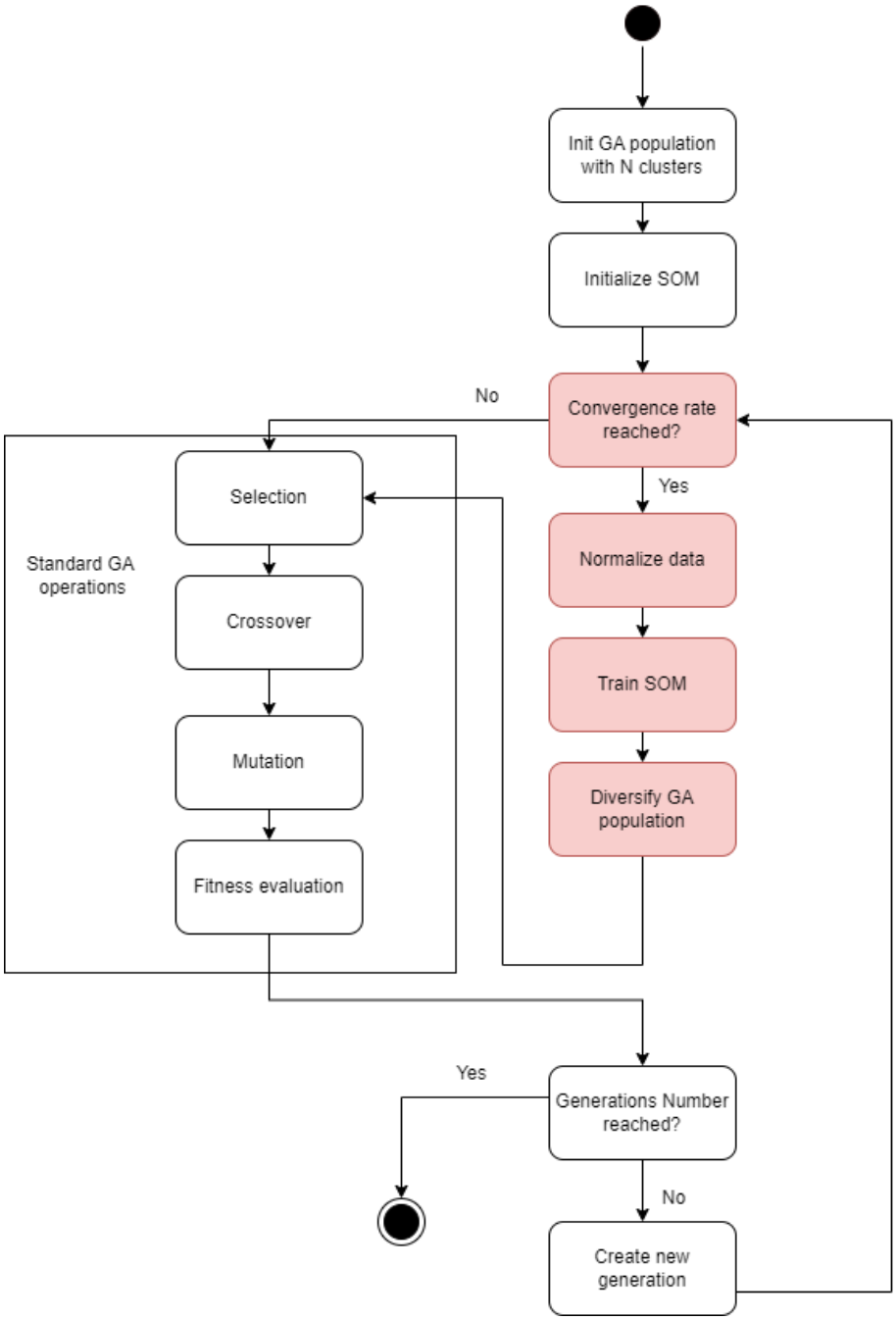

Рисунок 2.2 – Діаграма активностей роботи гібридного алгоритму

З наведеної діаграми можна побачити, що більша частина логіки генетичного алгоритму залишається як і раніше, але у гібридній реалізації додаються кроки для SOM. Оскільки навчання SOM є трудомістким та значно впливає на ефективність роботи програми в залежності від вхідних параметрів, то його навчання слід проводити тільки у випадках, коли це насправді необхідно.

2.7 Вибір методів для реалізації інтеграції генетичного алгоритму та самоорганізуючих карт Кохонена

2.7.1 Ініціалізація генетичного алгоритму

Фаза ініціалізації в генетичному алгоритмі для RCPSP є достатньо важливою, оскільки вона встановлює початкову популяцію, яка еволюціонує для пошуку оптимального графіку проєкту. Зокрема, ініціалізація спрямована на створення популяції хромосом, де кожна хромосома представляє потенційне рішення, тобто повний розклад проекту із зазначенням часу початку для кожної діяльності з урахуванням обмежень ресурсів.

Щоб ініціалізувати популяцію, було обрано механізм вибору колеса рулетки, який визначає пріоритетність дій на основі їх мінімального часу останнього завершення (LFT). LFT діяльності — це найпізніший час, до якого дія має завершитися, щоб забезпечити своєчасне завершення проекту, враховуючи тривалість і залежність наступних дій.

Встановлюючи пріоритети діяльності з нижчими мінімальними LFT, процес ініціалізації спрямований на створення хромосом, які, швидше за все, призведуть до скорочення тривалості проекту.

Процес вибору колеса рулетки на основі мінімальних LFT описується наступним чином.

Розрахунок мінімального LFT для кожної діяльності в мережі проекту, використовуючи техніку планування вперед-назад. Прохід вперед обчислює час найранішого старту і найранішого фінішу, тоді як прохід назад обчислює час останнього старту і останнього фінішу.

Визначення набору можливих дій для вибору на кожному кроці. Діяльність вважається можливою, якщо всі її попередники вже вибрані.

Обчислення максимального LFT серед можливих видів діяльності:

$$Tmax=\max i\in PLFTi \quad , \tag{2.11}$$

де P – множина всіх можливих активностей;

i – це найпізніший час завершення активності i.

Обчислення ваги вибору для кожної можливої діяльності:

$$i=LFTmax-LFTi+1 \quad , \tag{2.12}$$

де $Tmax$ – максимальний LFT серед можливих видів діяльності;

i – найпізніший час завершення активності i.

Обчислення загальної ваги вибору:

$$= \quad , \tag{2.13}$$

де P - множина всіх можливих активностей;

i – вага вибору діяльності i.

Призначення ймовірності вибору для кожної можливої діяльності. Наступна формула гарантує, що дії з нижчими LFT мають вищу ймовірність бути обраними.

$$i=wiW \quad , \tag{2.14}$$

де i – вага вибору діяльності i;

– загальна вага вибору.

Вибір активності завдяки механізму колеса рулетки на основі ймовірностей вибору. Для цього генерується випадкове число r від 0 до 1 та обирається активність i, для якої кумулятивна ймовірність перевищує r.

Обрана активність додається до хромосоми та видаляється з набору можливих активностей.

Перелічені кроки повторюються доки не буде утворено повноцінну хромосому та заповнено популяцію відповідно до вхідних параметрів.

Таким чином початкова популяція міститиме суміш хромосом, які схильні до діяльності з мінімальними LFT. Такий підхід дозволяє включити в початкову популяцію розклади схильні до мінімальної тривалості, які потім, вирогідно, зможуть давати кращих нащадків.

2.7.2 Метод селекції для генетичного алгоритму

Ранговий відбір — це метод відбору, який використовується в ГА для вибору батьківських хромосом для відтворення. Цей метод спрямований на підтримку різноманітності в популяції та запобігання передчасній конвергенції шляхом ранжування хромосом на основі їх значень придатності та призначення ймовірностей відбору відповідно до їх рангів. Ранговий відбір ґрунтується на відносному порядку хромосом.

Для проведення рангового відбору хромосоми в популяції спочатку сортуються в порядку спадання на основі їх значень придатності, від найбільш придатних до найменш придатних. Потім кожній хромосомі присвоюється ранг на основі її позиції в відсортованому списку, причому найбільш придатна хромосома отримує ранг N, а найменш придатна

хромосома отримує ранг 1. Імовірність вибору для кожної хромосоми розраховується за формулою:

$$i=rij=1Nj=2riNN+' \quad (2.15)$$

де N — розмір популяції;

r_i – її ранг.

Завдяки наведеній формулі хромосоми з вищим рангом мають вищу ймовірність бути відібраними, тоді як хромосоми з нижчим рангом все ще мають ненульову ймовірність, що забезпечує баланс між використанням найкращих рішень і дослідженням простору пошуку.

Ранговий вибір має кілька переваг перед іншими методами відбору. Він підтримує постійний тиск відбору протягом усього процесу еволюції, запобігаючи домінуванню здорових особин на ранніх етапах алгоритму. Крім того, він менш чутливий до масштабування значень придатності, оскільки враховує лише відносні рейтинги хромосом, а не їх абсолютні значення придатності. Даючи шанс менш придатним особам бути обраними в якості батьків, ранговий відбір допомагає зберегти різноманітність популяції та зменшує ризик передчасної конвергенції.

Такий метод є популярним вибором у багатьох рішеннях з ГА через його здатність підтримувати різноманітність і запобігати передчасній конвергенції. Він пропонує баланс між використанням найкращих рішень і дослідженням простору пошуку.

2.7.3 Вибір методу кросинговеру для генетичного алгоритма

У кваліфікаційній роботі в якості методу кросинговеру було обрано багатоточковий кросинговер. Це генетичний оператор, який використовується в ГА для створення нащадків шляхом обміну генетичним матеріалом між батьківськими хромосомами в кількох випадково вибраних точках. Ця техніка є розширенням одноточкового кросинговеру, який обмінює генетичний матеріал лише в одній точці. Багатоточкове схрещування дозволяє більш ретельно досліджувати простір пошуку, створюючи більшу різноманітність нащадків і зберігаючи цінні будівельні блоки від обох батьків.

Для вирішення проблеми RCPSP багатоточковий кросовер застосовується до списків активностей, що представляють хромосоми. Процес багатоточкового кросинговеру описується наступним чином.

Обираються дві батьківські хромосоми з популяції за допомогою відбору за рангом.

Випадково обирається k точок перетину, де k — заздалегідь визначений параметр. Точки кросинговеру є індексами по довжині хромосом, і вони поділяють хромосоми на k+1 сегментів.

Створюються дві нащадкові хромосоми, почергово копіюючи сегменти з батьківських хромосом. Перше потомство успадковує непарні сегменти від першого батька і парні сегменти від другого батька. Друге потомство успадковує парні сегменти від першого батька і непарні сегменти від другого батька.

Хромосоми нащадків проходять перевірку на послідовність активностей та обмеження ресурсів для підтвердження придатності для рішення проблеми та, у випадку, коли перевірку не пройдену, то спрацьовує механізм ремонту.

Багатоточковий кросинговер у спрощеному варіанті зображено на рисунку 2.3 [26].

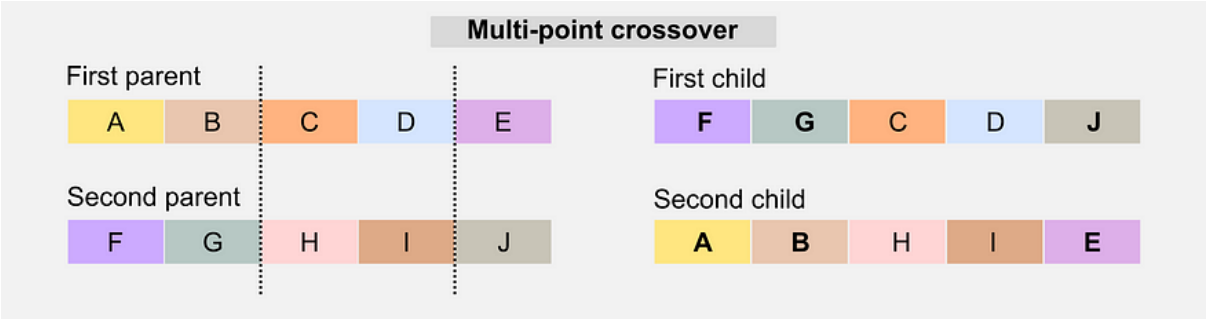

Рисунок 2.3 – Приклад багатоточкового кросинговеру

У якості реалізації механізму ремонту було обрано наступний спосіб: видалення повторюваних дій і вставка відсутніх дій таким чином, щоб зберегти відносний порядок дій у батьківських хромосомах.

Кількість точок кросинговеру, k, є гіперпараметром, який можна налаштувати, щоб контролювати ступінь обміну генетичним матеріалом між батьками. Більше значення k призводить до більш руйнівного кросинговеру, тоді як нижче значення призводить до більш консервативного обміну генетичним матеріалом. Оптимальне значення k залежить від конкретної проблеми та бажаного балансу між розвідкою та експлуатацією в процесі пошуку.

До переваг багатоточкового кросинговеру відносяться дослідження ширшого пошукового простору, легкість реалізації та адаптивність до поставлених задач.

Тим не менш, у випадках, коли необхідно задіяти механізм відновлення, то він може бути досить коштовним для великих проєктів. Також оптимальна кількість точок перетину може змінюватися залежно від проблеми та поточного стану процесу пошуку, що ускладнює визначення найкращого значення для k та потребує додаткових експериментів.

2.7.4 Вибір методу мутації для генетичного алгоритму

У якості алгоритму мутації було обрано рівномірну мутація, яка представляє простий і широко використовуваний оператор для ГА, що вносить випадкові зміни в гени хромосоми з певною ймовірністю. На відміну від мутації Гауса, яка додає безперервний шум у гени, рівномірна мутація вносить дискретні зміни шляхом заміни значення гена новим значенням, отриманим із рівномірного розподілу в дозволеному діапазоні. Цей оператор особливо добре підходить для проблем із дискретними просторами пошуку або категоріальними змінними.

Процес рівномірної мутації описується наступним чином.

Обирається хромосома з популяції для мутації на основі попередньо визначеної ймовірності мутації pm.

Для кожного гена (активності) у вибраній хромосомі з ймовірністю pm замініюється поточна активність іншою випадково вибраною діяльністю з набору придатних, які ще не заплановано, а попередників яких уже заплановано.

Відбувається перевірка придатності мутованої хромосоми на відповідність обмеженням RCPSP, після чого мутація відбувається до найближчої придатної хромосоми.

Імовірність мутації pm визначає ймовірність мутації гена. Більше значення pm призводить до більш масштабних змін у хромосомі, тоді як нижче значення призводить до меншої кількості модифікацій. Зазвичай, значення pm використовується у межах від 0,01 до 0,1 для уникнення руйнування хороших рішень і водночас дозволити достатнє дослідження.

Однією з переваг рівномірної мутації є її простота та легкість реалізації. Завдяки випадковій заміні активності в хромосомі рівномірна мутація може ввести нові варіації та допомогти GA досліджувати різні регіони простору пошуку. Цей оператор особливо ефективний у задачах із

дискретними змінними або коли оптимальне рішення, ймовірно, можна знайти шляхом комбінування будівельних блоків із різних хромосом.

2.7.5 Вибір алгоритму ініціалізації для самоорганізаючих карт Кохонена

Ініціалізація SOM впливає на визначення початкового стану карти та може значно вплинути на процес навчання та кінцевий результат. У якості методу ініціалізації було обрано ініціалізацію з лінійним розпадом, яка використовується для ініціалізації вагових коефіцієнтів SOM у структурований спосіб із поступовим зменшенням. У випадку гібридного алгоритму ініціалізація лінійного розпаду застосовується до ваг SOM, які представляють розклади проекту.

Для виконання ініціалізації виконуються наступні дії:

Визначається розмір і топологію SOM, як правило, це двовимірна сітка нейронів.

Визначається діапазон вхідних характеристик (тривалість діяльності, доступність ресурсів) і бажаний діапазон початкових ваг.

Для кожного нейрона (i, j) у SOM ініціалізується його ваговий вектор wij за допомогою функції лінійного розпаду:

$$wij=wmax-wmax-wmin\cdot dijdma, \quad (2.16)$$

де wma і wmi — максимальне та мінімальне значення бажаного діапазону ваги відповідно;

dij — це відстань між нейроном (i, j) і вибраною опорною точкою, тобто, центром карти;

dma — максимальна відстань між будь-яким нейроном і контрольною точкою.

Наведений перелік дій повторюється для усіх нейронів у SOM, ініціалізуючи їхні ваги за допомогою функції лінійного розпаду.

Вибір розміру карти також є важливим елементом алгоритму ініціалізації. Відповідно до рекомендацій з [27] розмірність карти має залежати від кількості вхідних елементів. Це необхідно для коректного розміщення нейронів між собою та забезпечення логічною відстані між ними, оскільки значна відстань між нейронами не дасть зрозуміти коректність їх кластерного розташування. Далі наведено формулу обчислення розмірності карти:

$$*\quad , \tag{2.17}$$

де N – розмір популяції, на основі якої буде проведено навчання.

Завдяки функції лінійного розпаду ваги нейронів лінійно зменшуються зі збільшенням відстані від контрольної точки. Ця схема ініціалізації створює плавний градієнт ваг по всій карті з вищими значеннями біля центру та меншими значеннями до країв.

Ініціалізація лінійного розпаду забезпечує структурований і детермінований спосіб ініціалізації ваг SOM, зменшуючи випадковість і мінливість у початковому стані карти. Також плавний градієнт вагових коефіцієнтів допомагає зберегти топологічні зв’язки між вхідними характеристиками, полегшуючи процес навчання та формування значущих кластерів.

Слід зазначити, що ефективність ініціалізації залежить від вибору вагового діапазону та контрольної точки. Якщо діапазон надто вузький або контрольна точка не підходить для вхідних даних, початкова карта може не охопити повну мінливість вхідного простору. Лінійна функція розпаду

передбачає симетричний і рівномірний розподіл вхідних характеристик, що може бути не завжди. У деяких задачах нелінійна або асиметрична схема ініціалізації може бути більш прийнятною.

Незважаючи на обмеження, ініціалізація лінійного розпаду є широко використовуваною технікою в SOM через її простоту, детермінізм і можливість створення структурованої початкової карти. У поєднанні з відповідними алгоритмами та параметрами навчання ініціалізація лінійного розпаду може допомогти SOM ефективно охопити структуру вхідного простору та підтримувати процес оптимізації в RCPSP та інші проблеми.

2.7.6 Вибір функції сусідства для самоорганізаючих карт Кохонена

Функція сусідства Гауса є фундаментальним компонентом SOM, який визначає вплив нейрона-переможця на сусідні нейрони під час процесу навчання. У контексті інтеграції SOM з ГА для вирішення проблеми RCPSP функція сусідства Гауса відіграє вирішальну роль у збереженні топологічної структури вхідного простору та керуванні оновленням ваги нейронів.

Функція сусідства Гауса застосовується після ідентифікації переможного нейрона у SOM на основі подібності між вхідним вектором x і ваговими векторами нейронів. Функція обчислює вплив сусідства нейрона-переможця на кожен нейрон і на карті, враховуючи їх відстань dic у просторі карти. Вплив обчислюється за допомогою функції Гауса, яка визначається як:

$$hcit = \exp -d2rc,ri2\sigma t2, \quad (2.18)$$

де $hcit$ – функція околу навколо найкрощої одиниці c для нейрона i;

t – ітерація;

rc,r – позиції найкращих одиниць та нейронів на карті відповідно;

) – радіус сусідства, який зменшується з часом.

Функція Гауса забезпечує плавний і безперервний вплив, який швидко зменшується зі збільшенням відстані від нейрона-переможця. Завдяки локалізованій природі функції Гауса нейрон-переможець має найсильніший вплив на своїх безпосередніх сусідів, тоді як вплив поступово зменшується для нейронів, що знаходяться далі.

Ступінь впливу сусідства контролюється параметром ширини σ функції Гауса. Більше значення призводить до більшого впливу, дозволяючи переможному нейрону впливати на ширше оточення, тоді як менше значення σ фокусує вплив на вужче оточення. Вибір параметру має велике значення для ефективності SOM, оскільки він визначає баланс між захопленням локальної структури вхідного простору та узагальненням для невидимих даних.

Після обчислення впливу сусідства вагові вектори нейронів у SOM оновлюються за допомогою правила навчання, яке включає вплив і швидкість навчання. Правило навчання регулює ваги нейронів на основі різниці між вхідним вектором і їхніми поточними векторами ваг, зважених за впливом сусідства та швидкістю навчання. Правило оновлення ваги визначається як:

$$wit+1=wit+\alpha t\cdot hic\cdot x-wit \quad , \qquad (2.19)$$

де wit – вектор ваги нейрона і на кроці часу t;

t – швидкість навчання на кроці часу t, яка визначає швидкість адаптації;

h_{ic} – це вплив сусідства переможного нейрона c на нейрон i;

x – вхідний вектор.

Цей процес оновлення дозволяє SOM адаптувати свої ваги для кращого представлення вхідного простору, зберігаючи при цьому топологічні зв’язки між нейронами. Швидкість навчання t контролює величину оновлення ваги, при чому вища швидкість навчання призводить до швидшої адаптації, але потенційно меншої стабільності, тоді як нижча швидкість навчання призводить до повільніших, але більш поступових оновлень.

Функція сусідства Гауса має кілька переваг, які роблять її популярним вибором для SOM. Плавний і безперервний вплив допомагає підтримувати топологічну структуру вхідного простору, гарантуючи, що найближчі нейрони на карті представляють схожі вхідні шаблони. Локалізований характер функції дозволяє поступово адаптувати вагові коефіцієнти, запобігаючи різким змінам і сприяючи стабільному процесу навчання. Крім того, регульований параметр ширини σ забезпечує гнучкість в управлінні ступенем впливу сусідства, дозволяючи досягти балансу між захопленням локальної та глобальної структури.

Однак функція сусідства Гауса також має деякі обмеження. Ефективність функції значною мірою залежить від відповідного вибору параметра ширини , для визначення якого може знадобитися метод проб і помилок. Крім того, функція Гауса передбачає симетричний та ізотропний вплив сусідства, який не завжди може бути оптимальним для конкретної задачі. У деяких випадках анізотропні або адаптивні функції сусідства, які враховують специфічні характеристики вхідного простору, можуть бути більш придатними.

Незважаючи на обмеження, функція сусідства Гауса надає плавність і здатність зберігати топологічну структуру вхідного простору, що робить її цінним інструментом для організації базових шаблонів і підтримки процесу

оптимізації в RCPSP. У поєднанні з відповідними темпами навчання, розмірами карти та спеціальними адаптаціями до проблеми функція сусідства Гауса може значно підвищити продуктивність SOM у вирішенні складних задач оптимізації.

2.7.7 Вибір функції функції навчання для самоорганізаючих карт Кохонена

Швидкість навчання є важливим параметром у SOM, який визначає швидкість і величину оновлення ваги під час процесу навчання. Він контролює ступінь коригування ваги нейронів на основі вхідних шаблонів і впливу сусідів.

Лінійне затухання – це часто використовуваний розклад швидкості навчання, який поступово зменшує швидкість навчання з часом. Ідея лінійного розпаду полягає в тому, щоб почати з відносно високої швидкості навчання, щоб забезпечити швидку початкову адаптацію, а потім поступово зменшувати швидкість навчання, щоб точно налаштувати вагові коефіцієнти та стабілізувати SOM. Лінійний графік розпаду визначається як:

$$t=\alpha 01-tT \quad , \qquad (2.20)$$

де t – швидкість навчання на кроці часу t;

αl – початкова швидкість навчання;

t – поточний крок часу;

T – загальна кількість кроків часу або ітерації.

Лінійний розклад гарантує, що швидкість навчання лінійно зменшується від початкового значення α_0 до нуля протягом процесу навчання. На початку навчання, коли t є малим відносно T, швидкість навчання близька до α_0, що дозволяє значно оновлювати вагу та швидко адаптуватися до шаблонів введення. У міру того, як тренування просувається і t наближається до T, швидкість навчання поступово наближається до нуля, зменшуючи величину оновлення ваги та сприяючи конвергенції та стабільності.

Вибір початкової швидкості навчання α_0 є важливим для продуктивності SOM. Висока початкова швидкість навчання дозволяє швидше початкової адаптації, але може призвести до нестабільності та перевищення оптимальної ваги. З іншого боку, низька початкова швидкість навчання може призвести до повільної конвергенції та довшого часу навчання. Відповідне значення α_0 залежить від конкретної задачі, складності вхідного простору та бажаного балансу між швидкістю та стабільністю.

Лінійний графік розпаду має кілька переваг. Він забезпечує простий та інтуїтивно зрозумілий спосіб контролювати швидкість навчання з часом, дозволяючи плавно переходити від швидкої адаптації до тонкого налаштування. Поступове зниження швидкості навчання допомагає SOM наблизитися до стабільного стану, одночасно зменшуючи ризик коливань або розбіжності. Крім того, лінійний розклад розпаду обчислювально ефективний і простий у реалізації, що робить його популярним вибором у різних програмах SOM.

Лінійний розпад передбачає фіксовану загальну кількість ітерацій T, яка не завжди може бути відома заздалегідь або не може бути оптимальною для всіх випадків проблеми. У деяких випадках може бути більш придатним адаптивний або залежний від даних розклад швидкості навчання, наприклад, експоненціальне загасання або обернене загасання за часом. Крім того,

графік лінійного затухання не враховує специфічні характеристики вхідного простору або хід процесу навчання, що може призвести до неоптимальної конвергенції або переобладнання.

Щоб усунути ці обмеження, були запропоновані різні модифікації та розширення графіка лінійного розпаду. Наприклад, швидкість навчання можна регулювати на основі відстані між вхідним шаблоном і виграшним нейроном, що дозволяє проводити більш локалізовані оновлення. Інший підхід полягає у використанні кусково-лінійного розкладу розпаду, де швидкість навчання зменшується поетапно, що дозволяє більше контролювати процес навчання.

Для розв’язання RCPSP лінійний графік розпаду застосовується до швидкості навчання SOM під час фази навчання. SOM використовується для кластеризації рішень, створених ГА, забезпечуючи компактне представлення простору пошуку. Швидкість навчання визначає швидкість і величину оновлень ваг нейронів на основі вхідних рішень і впливу сусідства. Використовуючи лінійний розклад розпаду, SOM може спочатку швидко адаптуватися до рішень, створених ГА, а потім поступово стабілізуватися та зблизитися до репрезентативної карти простору пошуку.

При застосуванні інтеграції SOM і ГА для вирішення RCPSP лінійний розклад розпаду може допомогти SOM ефективно кластеризувати та представляти рішення, створені ГА, підтримуючи процес оптимізації. Однак важливо враховувати специфічні характеристики RCPSP і потенційно адаптувати графік швидкості навчання, щоб краще відповідати проблемній області.

2.7.8 Вибір топології карти для самоорганізаючих карт Кохонена

Топологія карти визначає розташування та зв’язок нейронів у просторі. Вибір топології може суттєво вплинути на процес навчання, представлення вхідного простору та можливість інтерпретації отриманої карти. У контексті інтеграції SOM з ГА топологія гексагональної сітки особливо ефективною.

Топологія гексагональної сітки розташовує нейрони за правильною шестикутною схемою, де кожен нейрон з’єднаний із шістьма безпосередніми сусідами. Це розташування відрізняється від більш поширеної топології прямокутної сітки, де кожен нейрон з’єднаний із чотирма безпосередніми сусідами (вгорі, внизу, ліворуч і праворуч). Гексагональна топологія сітки пропонує кілька переваг перед прямокутною топологією сітки:

Покращений зв’язок: у шестикутній сітці кожен нейрон рівновіддалений від своїх шести сусідів, що призводить до більш рівномірного та симетричного шаблону зв’язку. Ця властивість забезпечує більш плавне та природне представлення вхідного простору, оскільки відстані між нейронами є більш послідовними та ізотропними.

Покращена локальна взаємодія: шестикутне розташування сприяє сильнішій локальній взаємодії між нейронами, оскільки кожен нейрон має більше безпосередніх сусідів порівняно з прямокутною сіткою. Ця розширена зв’язність забезпечує кращий охоплення та збереження локальної структури та топології вхідного простору.

Зменшене спотворення: топологія гексагональної сітки мінімізує спотворення відстаней і зв’язків між нейронами, особливо під час відображення просторів вхідних даних великої розмірності на двовимірну карту. Рівновіддалений характер гексагонального розташування допомагає точніше підтримувати відносні відстані та схожість між шаблонами введення.

Покращена візуальна інтерпретація: шестикутна сітка забезпечує візуально привабливе та інтуїтивно зрозуміле представлення карти, оскільки вона нагадує структуру, схожу на стільники. Цей макет полегшує

ідентифікацію кластерів, шаблонів і зв’язків між точками вхідних даних, підвищуючи інтерпретацію результуючої карти.

Щоб реалізувати топологію гексагональної сітки в SOM, нейрони зазвичай розташовуються у двовимірній решітці, де кожен нейрон ідентифікується своїми координатами i,j . Відстань між двома нейронами в гексагональній сітці можна обчислити за допомогою гексагональної метрики відстані, яка враховує унікальний шаблон зв’язності гексагонального розташування. Розглянемо для цього наступну формулу:

$$dhexi1,j1,i2,j2=maxi1-i2,j1-j2,i1+j1-i2-j2 \quad , \quad (2.21)$$

де $i1$, $j1$ – координати першого нейрона;

$i2$, $j2$ – координати другого нейрона.

Ця формула враховує максимум абсолютних різниць координат з урахуванням діагональної зв’язності гексагональної сітки.

При застосуванні топології гексагональної сітки до RCPSP з використанням SOM і ГА, карта використовується для кластеризації та представлення рішень, створених ГА. Кожен нейрон на карті відповідає певній області в просторі рішень, а ваги нейронів представляють характеристики або особливості рішень у цій області. Гексагональне розташування дозволяє більш природно та плавно відобразити простір рішення, фіксуючи схожість і зв’язки між різними рішеннями.

Під час процесу навчання SOM адаптує свої ваги на основі вхідних рішень і впливу сусідства, визначеного гексагональною відстанню. Гексагональна топологія гарантує, що оновлення вагових коефіцієнтів нейрона враховує рішення в його найближчому сусідстві, зберігаючи локальну структуру простору рішень. Ця властивість особливо корисна в

RCPSP, де подібні рішення можуть мати загальні характеристики або подібні шаблони розподілу ресурсів.

Топологія гексагональної сітки також полегшує візуалізацію та інтерпретацію отриманої карти. Розташувавши нейрони у формі шестикутника, карта може ефективно виділяти кластери схожих рішень, визначати області високоякісних рішень і виявляти закономірності чи зв’язки між рішеннями. Це візуальне представлення може надати цінну інформацію про структуру простору рішень і керувати дослідженням і експлуатацією перспективних регіонів ГА.

Таким чином, топологія гексагональної сітки є ефективним вибором для розташування нейронів у самоорганізуючих картах. Порівняно з прямокутною топологією сітки, вона пропонує покращене підключення, покращену локальну взаємодію, зменшене спотворення та кращу візуальну інтерпретацію. У застосуванні до інтеграції SOM і ГА для вирішення RCPSP топологія гексагональної сітки забезпечує більш природне та гладке представлення простору рішень, полегшуючи кластеризацію та дослідження високоякісних рішень.

2.8 Висновки до другого розділу

У результаті виконання цього розділу було проведено глибокий аналіз структури ГА та SOM, їх архітектурні особливості, позитивний та негативний вплив на вирішення RCPSP.

Завдяки проведеному аналізу було визначено техніки для використання у гібридному алгоритмі. До них відносяться методи селекції, мутації, вибору топології SOM, спосіб навчання та інші параметри. Також було сформовано послідовність кроків виконання гібридного алгоритму.

3 РЕАЛІЗАЦІЯ ГІБРИДНОГО АЛГОРИТМУ ДЛЯ РОЗВ'ЯЗАННЯ ЗАДАЧІ ПЛАНУВАННЯ ІТ-ПРОЄКТІВ З ОБМЕЖЕНИМИ РЕСУРСАМИ

3.1 Опис інструментів для реалізації гібридного алгоритму

Реалізація гібридного алгоритму, який поєднує ГА та SOM вимагає сучасних інструментів для забезпечення різноманіття засобів й задоволення поставлених вимог у якості алгоритму.

У якості мови програмування було обрано Python v.3.9.0.

Вибір мови програмування Python для реалізації гібридного алгоритму SOM-GA для вирішення RCPSP обумовлений рядом факторів та переваг, які надає ця мова для розробки складних алгоритмічних рішень.

Python є високорівневою мовою програмування, яка надає широкий спектр вбудованих функцій та бібліотек, що значно полегшує розробку складних алгоритмів.

Перелік усіх використаних бібліотек:

- math;
- random;
- os;
- timeit;
- pylab;
- minisom;
- numpy;
- matplotlib;
- sklearn.

Для реалізації SOM-GA важливою є бібліотека NumPy, яка містять оптимізовані функції для роботи з багатовимірними масивами та матрицями, а також надає засоби для ефективного виконання математичних операцій [28].

Ця бібліотеки дозволяють швидко та зручно маніпулювати великими обсягами даних, що є критичним при розв'язанні задач комбінаторної оптимізації, таких як RCPSP.

Також Python має потужні засоби для роботи з візуальними об’єктами, що є ключовими структурами даних при розв'язанні RCPSP. Оскільки алгоритм SOM часто використовується саме для візуалізації багатовимірних даних на карті, яка візуально зрозуміла для людини, то в роботі також було виконано візуалізацію популяцій індивідів. Для цього було використано бібліотеку matplotlib та її графічні компоненти, які дозволяють будувати складні об’єкти відповідно до обраної архітектури SOM [29].

Бібліотека Math надає набір математичних функцій та констант, які використовуються для різноманітних обчислень у процесі роботи алгоритму. Зокрема, функції з цієї бібліотеки застосовуються для розрахунку розміру карти SOM, що є важливим параметром для ефективного навчання та кластеризації індивідів [30].

Random – це бібліотека для генерації псевдовипадкових чисел, яка відіграє ключову роль у реалізації стохастичних операцій генетичного алгоритму. Вона використовується для випадкового вибору індивідів під час ініціалізації початкової популяції, а також для внесення випадкових змін у хромосоми під час мутації. Це забезпечує різноманітність популяції та дозволяє дослідити різні області простору пошуку [31].

Бібліотека Os надає функції для взаємодії з операційною системою, зокрема для роботи з файловою системою. Вона використовується для зчитування вхідних даних проекту з файлів, що дозволяє гнучко налаштовувати алгоритм для різних задач RCPSP без необхідності модифікації вихідного коду [32].

Timeit – це бібліотека для вимірювання часу виконання фрагментів коду. Вона застосовується для оцінки швидкодії різних компонентів алгоритму та вимірювання загального часу роботи програми. Це дає змогу

аналізувати ефективність запропонованого підходу та порівнювати його з іншими методами [33].

Scikit-learn – це бібліотека машинного навчання, яка надає різноманітні інструменти для обробки та аналізу даних. У даному випадку використовується клас MinMaxScaler для нормалізації вхідних даних перед навчанням SOM, що покращує якість кластеризації та запобігає зміщенню ваг [34].

Окрім наведених аргументів слід зазначити, що на даній мові програмування існує велика кількість бібліотек та репозиторіїв, написаних спільнотою програмістів, що дозволяє використати код з відкритим доступом та пришвидшити написання коду. Таким чином було обрано бібліотеку minisom, де вже існує стандартний функціонал для SOM, що дозволило використати існуючі методи [35].

За основу реалізації було взято відкритий репозиторій з кодом ГА [36]. Даний репозиторій використовує методи, описані у розділі 2. Логіку алгоритму побудовано на основі [37].

3.2 Опис реалізації гібридного алгоритму

Реалізація гібридного алгоритму вимагає чіткої структури майбутнього коду. Оскільки мова програмування Python підтримує об’єктно-орієнтовану парадигму, то було використано класи для структуризації даних та методів. Таким чином стає зрозуміло якими характеристиками володіють певні об’єкти.

На рисунку 3.1 наведено діаграму класів гібридного алгоритму.

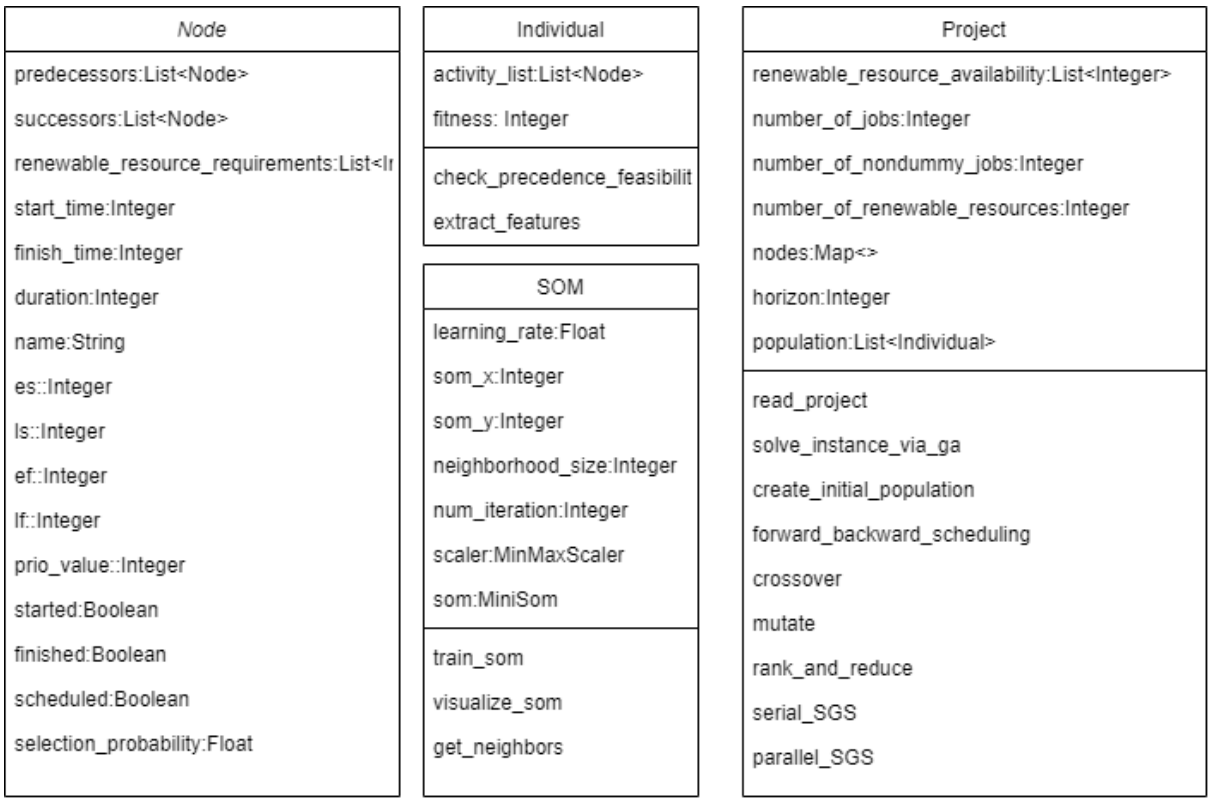

Рисунок 3.1 – Діаграма класів

З наведеного рисунку можна побачити розподіл методів та полів для кожного класу, які логічно віднесено до необхідних структур.

Клас Node відповідає за репрезентацію кожної активності проєкту. Оскільки кожна активність представляє собою складний об’єкт, то вона має доволі велику кількість пов’язаних полів для подальшої обробки.

Клас Individual представляє собою, власне, графік проєкту та характеризується переліком активностей та методами необхідними для перевірки їх послідовності, щоб такий графік відповідав поставленим вимогам та дотримувався послідовності подій. Така логіка є критично важливою, оскільки без неї не було б відповідності до ситуацій з реальних проєктів.

Клас Som реалізує обгортку, над MiniSom з бібліотеки minisom. Це необхідно для пристосування основного класу до особливостей RCPSP та ідеї, яка пропонується в даній роботі, а саме пошук сусідів найбільш

відповідного індивіда для різноманіття популяції та додаткова візуалізація потенційних результатів на 2-вимірній карті.

Останній клас – Project – містить основну логіку ГА та вилучення даних з тестового набору. Наведені методи використовують механізми, які детально описано в минулому розділі. Зчитування тестових даних відбувається з набору, поширеного у відкритому доступі у мережі інтернет, який зазвичай використовується для тестування робіт, що вирішують RCPSP [38].

Розглянемо елементи кожного класу, які наведено в таблицях 3.1-3.4.

Таблиця 3.1 – Опис елементів класу Node

Назва	Призначення	Тип даних	Модифікатор доступу
1	2	3	4
predecessors	Список попередніх активностей	list	public
successors	Список наступних активностей	list	public
renewable_resource_requirements	Список вимог до поновлюваних ресурсів	list	public
start_time	Час початку активності	int	public
finish_time	Час завершення активності	int	public
duration	Тривалість активності	int	public
name	Назва активності	str	public
es	Ранній час початку	int	public

	активності		
ls	Пізній час початку активності	int	public

Кінець таблиці 3.1

1	2	3	4
ef	Ранній час завершення активності	int	public
lf	Пізній час завершення активності	int	public
prio_value	Пріоритетне значення активності	int	public
started	Прапорець, що вказує, чи розпочата активність	bool	public
finished	Прапорець, що вказує, чи завершена активність	bool	public
scheduled	Прапорець, що вказує, чи запланована активність	bool	public
selection_probability	Ймовірність вибору активності	float	public

Таблиця 3.2 - Опис елементів класу Project

Назва	Призначення	Тип даних	Модифікатор доступу
renewable_resource_availability	Список доступності поновлюваних ресурсів	list	public
number_of_jobs	Кількість робіт у проекті	int	public
number_of_nondummy_jobs	Кількість робіт у проекті, виключаючи фіктивні роботи	int	public
number_of_renewable_resources	Кількість типів поновлюваних ресурсів	int	public
nodes	Словник активностей проекту	dict	public
horizon	Горизонт планування (максимальна тривалість проекту)	int	public
population	Список індивідів у популяції	list	public

Таблиця 3.3 - Опис елементів класу Individual

Назва	Призначення	Тип даних	Модифікатор доступу
fitness	Значення фітнес-функції індивіда	float	public
activity_list	Список активностей індивіда	list	public

Таблиця 3.4 – Опис елементів класу Som

Назва	Призначення	Тип даних	Модифікатор доступу
som_x	Розмір карти SOM по осі X	int	public
som_y	Розмір карти SOM по осі Y	int	public
learning_rate	Швидкість навчання SOM	float	public
neighborhood_size	Розмір околу в SOM	int	public
num_iteration	Кількість ітерацій навчання SOM	int	public
scaler	Скейлер для нормалізації	MinMaxScaler	public

	даних		
som	Об'єкт карти SOM	MiniSom	public

Діаграму активностей варіантів реалізації пошуку сусідів для диверсифікації популяції наведено на рисунку 3.2.

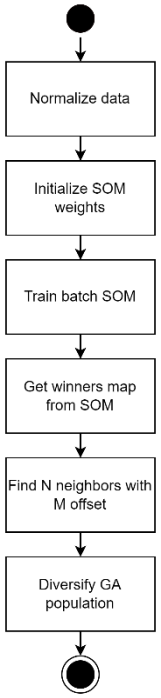

Рисунок 3.2 – Діаграма активностей варіантів реалізації диверсифікації популяції генетичного алгоритму

Як можна побачити на рисунку 3.2 початковим кроком є нормалізація даних за якими слідкує процес ініціалізації важелів та тренування самоорганізуючих карт. В результаті навчання можна отримати карту переможців, яка представляє собою список координат у двовимірному просторі. Оскільки шукати завжди найближчих сусідів може не мати сенсу через значну схожість графіків, то для цього було введено змінну offset, яка визначає скільки найближчих сусідів необхідно пропустити, щоб отримати переможців з інших кластерів. Таким чином різноманіття індивідів у

популяції за результатами навчання SOM можна регулювати завдяки встановленим параметрам. Після отримання сусідів та пошуку відповідних індивідів з ненормалізованими даними – вони заміщують встановлену кількість індивідів в популяції та отримується бажаний результат.

Операція навчання SOM може бути трудомісткою, в залежності від встановлених параметрів. Для запобігання можливих затримок в обрахунках було вирішено проводити навчання лише в залежності від параметру конвергенції, який визначає скільки поколінь ГА найкращий результат придатності не змінюється та при його досягненні відбувається навчання.

3.3 Висновки до третього розділу

У результаті виконання реалізації гібридного алгоритму було використано сучасні засоби та підходи до написання коду, виконано задачу по розробці коду, що поєднує 2 наведених алгоритми та додано проміжну візуалізацію результатів навчання SOM.

Для збільшення розуміння роботи гібридного алгоритму було додано діаграму класів, що використовуються під час реалізації та їх опис, також наведено діаграму активностей, що показує яким чином буде проведено диверсифікацію популяції з метою спроби вийти з локального оптимуму.

Повний код програми наведено у додатку А.

4 АПРОБАЦІЯ РЕЗУЛЬТАТІВ КВАЛІФІКАЦІЙНОЇ РОБОТИ

4.1 Загальний опис апробації

Апробація розробленого гібридного алгоритму SOM-GA для розв'язання задачі RCPSP має на меті дослідити ефективність запропонованого підходу порівняно зі стандартним генетичним алгоритмом. Для цього було використано набір тестових даних, який є широко вживаним у науковій спільноті для оцінки алгоритмів планування проєктів з обмеженими ресурсами.

Процес апробації включає проведення серії запусків як стандартного ГА, так і гібридного SOM-GA з різними параметрами, такими як розмір популяції, ймовірності схрещування та мутації, розмір карти SOM та кількість ітерацій. Результати кожного запуску фіксуються та усереднюються для отримання надійних оцінок продуктивності алгоритмів.

Ключовими показниками ефективності, які розглядаються під час апробації, є час виконання проєкту та швидкість збіжності алгоритмів. Статистичний аналіз результатів дозволяє визначити значущість відмінностей між стандартним ГА та гібридним SOM-GA.

Окрім кількісного аналізу, апробація також включає якісну оцінку роботи гібридного алгоритму, зокрема візуалізацію процесу навчання SOM та кластеризації рішень. Це надає додаткове розуміння динаміки пошуку оптимальних рішень та ролі SOM у підтримці різноманітності популяції та запобіганні передчасній збіжності.

4.2 Опис вхідних даних

Для проведення апробації гібридного алгоритму SOM-GA було обрано набір тестових даних J30, який є одним із стандартних наборів для оцінки алгоритмів розв'язання задачі RCPSP. Цей набір даних містить 480 задач, кожна з яких складається з 30 активностей проєкту [38].

Задачі в наборі J30 мають різні характеристики, що дозволяє оцінити ефективність алгоритмів у різних умовах. Ці характеристики включають:

– кількість ресурсів: задачі містять різну кількість ресурсів, які необхідні для виконання активностей проєкту;

– складність мережі: задачі відрізняються за структурою мережі активностей, що визначає залежності та обмеження між ними;

– коефіцієнт використання ресурсів: задачі мають різний ступінь завантаженості ресурсів, що впливає на складність планування;

– тривалість активностей: задачі містять активності з різною тривалістю виконання.

Приклад вхідних даних з наведеного набору тестових даних наведено у таблиці 4.1.

Таблиця 4.1 – Вхідні дані для тестування гібридного алгоритму

Характеристика	Опис	Дані
1	2	3
projects	Кількість проєктів	1
jobs (incl. supersource/sink)	Кількість робіт, включаючи фіктивні	32
horizon	Максимальна тривалість проєкту	166

Кінець таблиці 4.1

1		2	3
resources	renewable	Кількість поновлюваних ресурсів	4
	nonrenewable	Кількість непоновлюваних ресурсів	0
	doubly constrained	Кількість подвійно обмежених ресурсів	0

Таблиця 4.2 – Вхідні дані з інформацією про проєкт

Характеристика	Опис	Дані
pronr.	Ідентифікатор проєкту	1
#jobs	Кількість робіт, включаючи фіктивні	32
rel. date	Дата початку проєкту	0
duedate	Очікувана дата завершення проєкту	51
tardcost	Вартість запізнення завершення проєкту на одиницю часу	21
MPM-Time	Мінімальна тривалість проєкту за методом критичного шляху	51

Таблиця 4.3 – Вхідні дані з інформацією про послідовність активностей проєкту

Характеристика	Опис	Дані
jobnr.	Ідентифікатор активності	1
#modes	Кількість режимів	1
#successors	Кількість пов’язаних наступних активностей	3
successors	Ідентифікатори активностей пов’язаних наступників	2 3 4

Таблиця 4.4 – Вхідні дані з інформацією про тривалість активностей проєкту та їх запити щодо ресурсів

Характеристика	Опис	Дані
jobnr.	Ідентифікатор активності	1
mode	Режим	1
duration	Тривалість активності	10
R1	Ресурс 1	6
R2	Ресурс 2	9
R3	Ресурс 3	7
R4	Ресурс 4	10

Доступність ресурсів визначається кількістю часу для їх використання на кожну активність відповідно з урахуванням часу скільки знадобиться для її виконання. Кожен ресурс має власну показник доступності та у тестовому наборі це записується як R1 = 17, R2 = 15, R3 = 17, R4 = 17.

Кожна задача в наборі J30 представлена у вигляді файлу з описом активностей, їх тривалості, потреб у ресурсах та відношень передування. Ці файли є вхідними даними для алгоритмів планування проєктів, які мають на

меті знайти оптимальний розклад виконання активностей з урахуванням обмежень на ресурси та відношень передування.

Використання стандартного набору даних J30 для апробації гібридного алгоритму SOM-GA дозволяє отримати порівнювані результати з іншими дослідженнями та оцінити ефективність запропонованого підходу в контексті відомих задач RCPSP.

4.3 Хід апробації гібридного алгоритму

Апробація гібридного алгоритму передбачає процес виконання програми двома способами: з використанням існуючого підходу, тобто стандартного ГА, та з використанням запропонованого гібридного підходу, який поєднує ГА з самоорганізаційними картами Кохонена (SOM-GA). Метою такого порівняння є оцінка ефективності та доцільності застосування гібридного алгоритму для розв'язання задачі RCPSP.

Перш ніж перейти до безпосереднього розгляду відмінностей у результатах роботи обох підходів, доцільно проаналізувати вихідні дані, отримані в процесі роботи гібридного алгоритму SOM-GA. Ці дані можуть надати цінну інформацію про характеристики алгоритму та особливості його роботи.

Одним з ключових аспектів вихідних даних гібридного алгоритму є візуалізація процесу навчання SOM та кластеризації індивідів. SOM дозволяє відобразити багатовимірний простір пошуку на двовимірну сітку нейронів, зберігаючи топологічні властивості даних. Візуалізація карти SOM може показати розподіл індивідів у просторі пошуку, виявити кластери схожих розв'язків та допомогти зрозуміти структуру простору пошуку.

Аналіз візуалізації SOM може надати інформацію про те, наскільки ефективно алгоритм досліджує простір пошуку, чи формуються чіткі кластери розв'язків, та чи відбувається конвергенція до оптимальних або близьких до оптимальних розв'язків. Якщо на карті SOM спостерігаються чіткі кластери з низькою внутрішньокластерною відстанню та високою міжкластерною відстанню, це може свідчити про ефективність алгоритму в знаходженні різноманітних та якісних розв'язків.

Крім того, вихідні дані гібридного алгоритму містять інформацію про найкращі знайдені розв'язки, а саме їх фітнес-значення. Аналіз цих даних дозволяє оцінити якість отриманих розв'язків, перевірити дотримання обмежень на ресурси та пріоритетні відношення між активностями, а також порівняти результати з відомими оптимальними розв'язками або розв'язками, отриманими іншими методами.

Розглянемо візуалізацію SOM на рисунку 4.1.

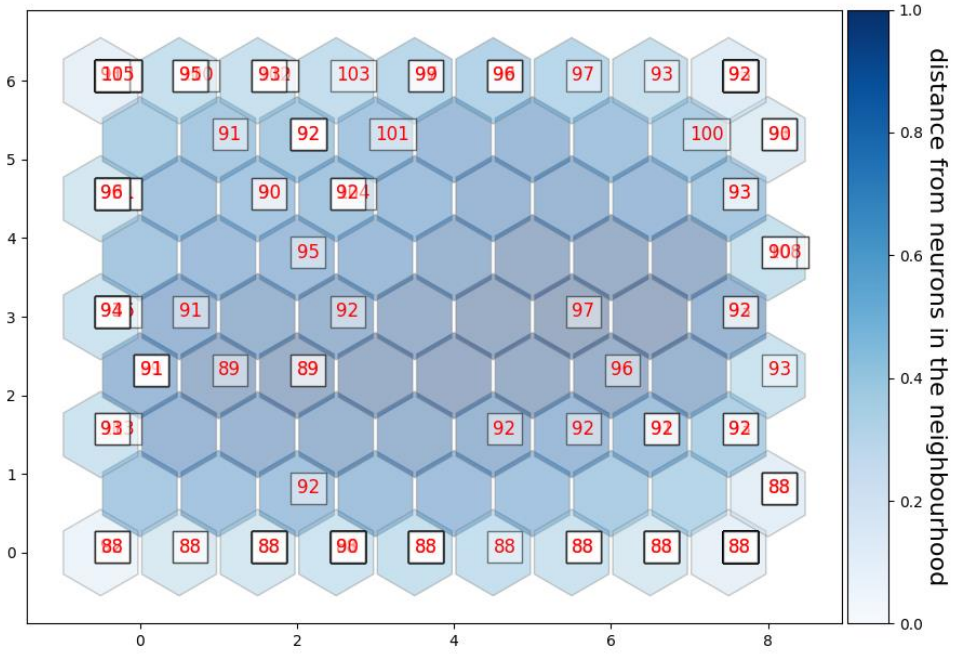

Рисунок 4.1 – Візуалізація кінцевого виклику навчання самоорганізаційних карт Кохонена

На даному рисунку візуально помітно розподіл між різними індивідами відповідно до їх тривалостей проєкту. Вони поділяються на кластери, кожен з яких можна виділити візуальним шляхом. Кластеризація відбувається на основі схожості послідовностей активностей за що відповідає процес навчання у SOM. З кожною ітерацією кількість вхідних даних збільшується, оскільки з періодичністю в 5 кроків додаються нові індивіди для розширення популяції для навчання. Таким чином карта з кожним навчанням буде змінюватись та формувати нові кластери відповідно до наявних даних.

У результаті кожного запуску програму у консоль виводиться найкраща придатність з усієї популяції. Приклад наведено на рисунку 4.2.

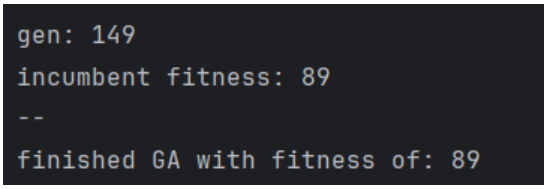

Рисунок 4.2 – Приклад консольного виводу після виконання програми

Іншим аспектом вихідних даних гібридного алгоритму є статистичні показники, такі як похибка квантування та топографічна. Ці показники можуть допомогти оцінити стабільність та ефективність роботи алгоритму, а також зробити висновки про вплив різних параметрів на його продуктивність.

Похибка квантування (quantization error) – це міра відмінності між вхідними векторами та їх найближчими нейронами-переможцями на карті SOM після навчання. Вона показує, наскільки добре карта SOM апроксимує розподіл вхідних даних.

Для кожного вхідного вектора знаходиться нейрон-переможець - нейрон, ваговий вектор якого має найменшу евклідову відстань до вхідного вектора. Похибка квантування обчислюється як середнє значення квадратів відстаней між кожним вхідним вектором та його нейроном-переможцем.

Топографічна похибка (topographic error) – це міра збереження топології вхідних даних на карті SOM після навчання. Вона показує, наскільки добре карта SOM зберігає просторові відношення між вхідними векторами.

Для кожного вхідного вектора знаходяться два нейрони-переможці: перший та другий за величиною активації. Якщо ці два нейрони не є сусідами на карті SOM, то вважається, що для даного вхідного вектора топологія не була збережена. Топографічна похибка обчислюється як частка вхідних векторів, для яких перший та другий нейрони-переможці не є сусідами [39].

На рисунку 4.3 зображено графік з результатами наведених похибок протягом 10000 ітерацій.

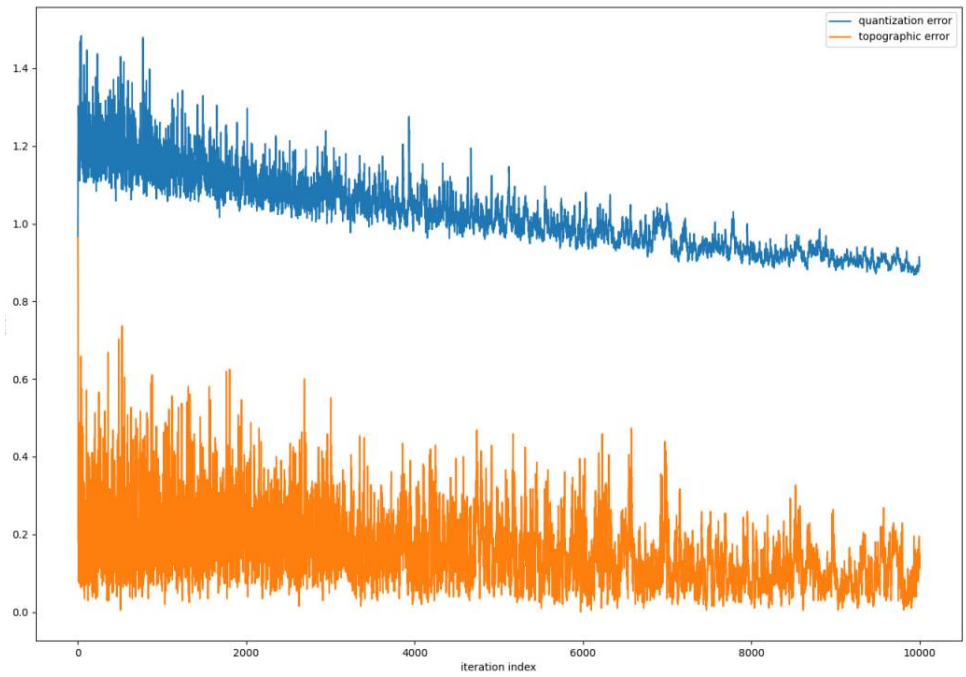

Рисунок 4.3 – Похибка квантизації та топографічна

З наведеного рисунку можна побачити, що в результаті проведення навчання похибка мала значні коливання, але згодом зменшувалась.

Для виконання апробації гібридного алгоритму було взято один з наборів даних описаних раніше. Його найкраща тривалість знайдена евристичним методом становить 85 одиниць часу. Для проведення досліду

було використано 2 методи, а саме ГА без модифікації інтеграції іншого алгоритму та гібридний алгоритм. Щоб виконати дослід та перевірити поведінку було виконано 50 ітерацій для кожного методу.

Було використано наступні параметри:

Кількість індивідів у популяції – 50, кількість поколінь – 150, ймовірність мутації 0.25, рівень досягання конвергенції – 30, рівень навчання для SOM – 0.7, відстань між сусідами – 10, кількість ітерації SOM – 10000, кількість індивідів для заміщення після навчання SOM – 20.

У результаті отримано наступні результати: середня тривалість – 87.56; час виконання в секундах – 280.

Результати виконання гібридного алгоритму наступні: середня тривалість – 87.28; час виконання в секундах – 440.

Відповідно до отриманих результатів апробації гібридного алгоритму SOM-GA, можна зробити висновки, що покращення у роботі алгоритму в порівнянні зі стандартним генетичним алгоритмом відбулось незначне. Хоча інтеграція самоорганізаційних карт Кохонена мала на меті підвищити ефективність пошуку оптимальних розв'язків та запобігти передчасній збіжності, реальний вплив на якість розв'язків виявився обмеженим.

Одним з факторів, які потрібно враховувати, є збільшення часу виконання гібридного алгоритму порівняно зі стандартним ГА. Додаткові обчислення, пов'язані з навчанням та використанням SOM, призводять до зростання обчислювальної складності та, відповідно, до збільшення часу роботи алгоритму. Це може бути особливо відчутним для великих проектів з великою кількістю активностей та обмежень.

Варто зазначити, що наведені показники ефективності відрізняються в залежності від набору тестових даних, на яких проводилась апробація. Це свідчить про необхідність подальшого дослідження та аналізу факторів, які впливають на ефективність SOM-GA в різних умовах.

Одним з напрямків для потенційного покращення роботи гібридного алгоритму є зміна стратегії заміщення індивідів на основі результатів кластеризації SOM. У поточній реалізації використовується підхід, коли індивіди заміщуються найближчими сусідами відносно найкращого індивіда. Однак, можливо, інші стратегії заміщення, такі як вибір представників з різних кластерів або врахування щільності розподілу індивідів, могли б привести до кращих результатів, що потребує додаткового дослідження та експериментів.

Крім того, оптимізація вхідних параметрів алгоритму, таких як розмір популяції, ймовірності мутації та схрещування, розмір карти SOM та кількість ітерацій навчання, також може суттєво вплинути на ефективність розв'язання задачі RCPSP. Однак, пошук оптимальних значень цих параметрів вимагає значних обчислювальних ресурсів та часу, оскільки потребує багаторазового запуску алгоритму на різних наборах даних та з різними комбінаціями параметрів.

4.4 Висновки до четвертого розділу

У результаті проведення апробації було встановлено, що існуюча реалізація гібридного алгоритму має обмежене покращення ефективності, але гібридний алгоритм SOM-GA залишається перспективним напрямком для подальшого дослідження та вдосконалення. Майбутні дослідження можуть зосередитись на пошуку більш ефективних стратегій інтеграції SOM та ГА, оптимізації параметрів алгоритму, а також на аналізі факторів, які впливають на його ефективність в різних умовах.

Завдяки проведеному аналізу можна сказати, що додаткові дослідження в даній області можуть надати більш оптимальні результати та принести

додаткову користь у сфері планування ІТ-проєктів.

ВИСНОВКИ

У ході кваліфікаційної роботи було досліджено методи вирішення проблеми планування ІТ-проєктів з обмеженими ресурсами за допомогою засобів штучного інтелекту.

У першому розділі проведено огляд та аналіз існуючих методів вирішення проблеми планування проєктів, таких як метод критичного шляху, метод критичних ланцюгів та генетичні алгоритми. Визначено перспективність використання генетичних алгоритмів у поєднанні з самоорганізаційними картами Кохонена для розв'язання задачі планування проєктів з обмеженими ресурсами.

У другому розділі досліджено модифікацію генетичного алгоритму для розв'язання RCPSP шляхом інтеграції з самоорганізаційними картами Кохонена. Проаналізовано структуру та архітектурні особливості генетичних алгоритмів і карт Кохонена, визначено техніки для використання в гібридному алгоритмі SOM-GA. Сформовано послідовність кроків виконання гібридного алгоритму.

У третьому розділі здійснено програмну реалізацію гібридного алгоритму SOM-GA з використанням сучасних інструментів та підходів. Розроблено діаграму класів та діаграму активностей, що відображають структуру та логіку роботи алгоритму. Реалізовано механізм диверсифікації популяції генетичного алгоритму за допомогою карт Кохонена.

У четвертому розділі проведено апробацію розробленого гібридного алгоритму SOM-GA на тестових даних. Здійснено порівняльний аналіз ефективності роботи гібридного алгоритму та стандартного генетичного алгоритму. Отримані результати демонструють обмежене покращення

ефективності гібридного підходу, однак вказують на перспективність подальших досліджень у даному напрямку.

За результатами дослідження можна зробити висновок, що використання генетичних алгоритмів у поєднанні з самоорганізаційними картами Кохонена є перспективним підходом для вирішення задачі планування ІТ-проєктів з обмеженими ресурсами. Запропонований гібридний алгоритм SOM-GA демонструє потенціал для покращення ефективності пошуку оптимальних розкладів проєктів, однак потребує подальшого вдосконалення та дослідження.

Напрямками майбутніх досліджень можуть бути оптимізація параметрів алгоритму, пошук ефективніших стратегій інтеграції карт Кохонена з генетичним алгоритмом, а також аналіз факторів, що впливають на ефективність гібридного підходу в різних умовах. Практична цінність розробленого програмного засобу полягає в можливості його використання для оптимізації планування ІТ-проєктів з обмеженими ресурсами.

За тематикою кваліфікаційної роботи було опубліковано тези доповіді на тему «Дослідження моделей використання засобів штучного інтелекту для планування ІТ-проєктів» [40].

Кваліфікаційну роботу виконано відповідно до методичних вказівок [41].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання, Чинний від 22.06.2015. Київ: ДП «УкрНДНЦ», 2016, 26 с.2.

2. ДСТУ 8302:2015 «Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання».

3. Iso 12207. Process Improvement with CMMI® v1.2 and ISO Standards. 2008. С. 321–357. URL: https://doi.org/10.1201/9781420052848.axf (дата звернення: 01.04.2024).

4. Iso 15288. Process Improvement with CMMI® v1.2 and ISO Standards. 2008. С. 295–319. URL: https://doi.org/10.1201/9781420052848.axe (дата звернення: 01.04.2024).

5. Kafile M. Project Planning and Scheduling in the Face of the Fourth Industrial Revolution (4IR). Journal of Business Administration Research. 2021. Т. 4, № 3. URL: https://doi.org/10.30564/jbar.v4i3.3398 (дата звернення: 01.04.2024).

6. AlNasseri H., Aulin R. Assessing Understanding of Planning and Scheduling Theory and Practice on Construction Projects. Engineering Management Journal. 2015. Т. 27, № 2. С. 58–72. URL: https://doi.org/10.1080/10429247.2015.1035963 (дата звернення: 01.04.2024).

7. PMBOK. Encyclopedia of Education and Information Technologies. Cham, 2020. С. 1258. URL: https://doi.org/10.1007/978-3-030-10576-1_300500 (дата звернення: 01.04.2024).

8. Liberzon V., Khalid S., Dyariwe A. What are the advantages and disadvantages of using critical chain method over critical path method?. URL:

https://www.linkedin.com/advice/1/what-advantages-disadvantages-using-critical (дата звернення: 03.04.2024).

9. Schroer A. Artificial Intelligence (AI): What Is AI and How Does It Work?. URL: https://builtin.com/artificial-intelligence (дата звернення: 03.04.2024).

10. Ancveire I., Poļaka I. Application of Genetic Algorithms for Decision-Making in Project Management: A Literature Review. Information Technology and Management Science. 2019. T. 22. C. 22–31. URL: https://doi.org/10.7250/itms-2019-0004 (дата звернення: 10.04.2024).

11. Artificial Intelligence Applications in Project Scheduling: A Systematic Review, Bibliometric Analysis, and Prospects for Future Research / Z. Bahroun та ін. Management Systems in Production Engineering. 2023. T. 31, № 2. C. 144–161. URL: https://doi.org/10.2478/mspe-2023-0017 (дата звернення: 10.04.2024).

12. Artificial intelligence and machine learning / N. Kühl та ін. Electronic Markets. 2022. URL: https://doi.org/10.1007/s12525-022-00598-0 (дата звернення: 11.04.2024).

13. Florez-Perez L., Song Z., Cortissoz J. C. Using machine learning to analyze and predict construction task productivity. Computer-Aided Civil and Infrastructure Engineering. 2022. URL: https://doi.org/10.1111/mice.12806 (дата звернення: 13.04.2024).

14. Shelar V., Saranathan G., Rezaei S. What challenges can you expect when training HM models?. URL: https://www.linkedin.com/advice/3/what-challenges-can-you-expect-when-training-ann-txsre (дата звернення: 15.04.2024).

15. Patnaik M., Melani Adrian A. Cognitive Big Data Intelligence with a Metaheuristic Approach. 2022. 356 c.

16. Artigues C. The Resource-Constrained Project Scheduling Problem. Resource-Constrained Project Scheduling. London, UK. C. 19–35. URL: https://doi.org/10.1002/9780470611227.ch1 (дата звернення: 20.04.2024).

17. Zhang H., Xu H., Peng W. A Genetic Algorithm for Solving RCPSP. 2008 International Symposium on Computer Science and Computational Technology, м. Shanghai, China, 20–22 груд. 2008 р. 2008. URL: https://doi.org/10.1109/iscsct.2008.255 (дата звернення: 20.04.2024).

18. Eiben A. E., Smith J. E. Introduction to Evolutionary Computing. Berlin, Heidelberg : Springer Berlin Heidelberg, 2015. URL: https://doi.org/10.1007/978-3-662-44874-8 (дата звернення: 20.04.2024).

19. Cetin U., Gundogmus Y. E. Feature Selection with Evolving, Fast and Slow Using Two Parallel Genetic Algorithms. 2019 4th International Conference on Computer Science and Engineering (UBMK), м. Samsun, Turkey, 11–15 верес. 2019 р. 2019. URL: https://doi.org/10.1109/ubmk.2019.8907165 (дата звернення: 23.05.2024).

20. Kohonen T. Self-Organizing Maps. Berlin, Heidelberg : Springer Berlin Heidelberg, 2001. URL: https://doi.org/10.1007/978-3-642-56927-2 (дата звернення: 20.04.2024).

21. Amor H. B., Rettinger A. Intelligent exploration for genetic algorithms. the 2005 conference, м. Washington DC, USA, 25–29 черв. 2005 р. New York, New York, USA, 2005. URL: https://doi.org/10.1145/1068009.1068250 (дата звернення: 14.04.2024).

22. Kan S., Fei Z., Kita E. Application of self-organizing maps to genetic algorithms. OPTI 2009, м. Algarve, Portugal, 8–10 черв. 2009 р. Southampton, UK, 2009. URL: https://doi.org/10.2495/op090011 (дата звернення: 14.04.2024).

23. Liu D., Wang X., Du J. A Clustering-Based Evolutionary Algorithm for Traveling Salesman Problem. 2009 International Conference on Computational Intelligence and Security, м. Beijing, China, 11–14 груд. 2009 р. 2009. URL: https://doi.org/10.1109/cis.2009.80 (дата звернення: 14.04.2024).

24. Cheng M.-Y., Huang K.-Y. GENETIC ALGORITHM-BASED CHAOS CLUSTERING APPROACH FOR NONLINEAR OPTIMIZATION. Journal of

Marine Science and Technology. 2010. T. 18, № 3. URL: https://doi.org/10.51400/2709-6998.1891 (дата звернення: 14.04.2024).

25. Cheng M.-Y., Tran D.-H., Wu Y.-W. Using a fuzzy clustering chaotic-based differential evolution with serial method to solve resource-constrained project scheduling problems. Automation in Construction. 2014. T. 37. C. 88–97. URL: https://doi.org/10.1016/j.autcon.2013.10.002 (дата звернення: 14.04.2024).

26. Rajput M. Building generations by Genetic algorithm. URL: https://ai.plainenglish.io/building-generations-by-genetic-algorithm-1c02ebf6fd8b (дата звернення: 22.04.2024).

27. Advances in Computational Intelligence / ред.: I. Rojas, G. Joya, A. Catala. Cham : Springer International Publishing, 2015. URL: https://doi.org/10.1007/978-3-319-19222-2 (дата звернення: 18.05.2024).

28. NumPy documentation – NumPy v1.26 Manual. NumPy. URL: https://numpy.org/doc/stable/ (дата звернення: 02.05.2024).

29. API Reference – Matplotlib 3.9.0 documentation. Matplotlib – Visualization with Python. URL: https://matplotlib.org/stable/api/index.html (дата звернення: 19.05.2024).

30. math - Mathematical functions. Python documentation. URL: https://docs.python.org/3/library/math.html (дата звернення: 19.05.2024).

31. random - Generate pseudo-random numbers. Python documentation. URL: https://docs.python.org/3/library/random.html (дата звернення: 19.05.2024).

32. os - Miscellaneous operating system interfaces. Python documentation. URL: https://docs.python.org/3/library/os.html (дата звернення: 19.05.2024).

33. timeit - Measure execution time of small code snippets. Python documentation. URL: https://docs.python.org/3/library/timeit.html (дата звернення: 19.05.2024).

34. API Reference. scikit-learn. URL: https://scikit-learn.org/stable/modules/classes.html (дата звернення: 19.05.2024).

35. MiniSom is a minimalistic implementation of the Self Organizing Maps. GitHub. URL: https://github.com/JustGlowing/minisom (дата звернення: 19.05.2024).

36. Python implementation of a Genetic Algorithm for the Resource-Constrained Project Scheduling Problem. GitHub. URL: https://github.com/derhendrik/RCPSP_GA/ (дата звернення: 19.05.2024).

37. Hartmann S. A competitive genetic algorithm for resource-constrained project scheduling. Naval Research Logistics. 1998. Т. 45, № 7. С. 733–750. URL: https://doi.org/10.1002/(sici)1520-6750(199810)45:7%3C733::aid-nav5%3E3.0.co;2-c (дата звернення: 15.05.2024).

38. Single Mode Data Sets. URL: https://www.om-db.wi.tum.de/psplib/getdata_sm.html (дата звернення: 19.05.2024).

39. Jevtić M., Mladenović S., Granić A. Source Code Analysis in Programming Education: Evaluating Learning Content with Self-Organizing Maps. Applied Sciences. 2023. Т. 13, № 9. С. 5719. URL: https://doi.org/10.3390/app13095719 (дата звернення: 19.05.2024).

40. Краснєнков І.О. Дослідження моделей використання засобів штучного інтелекту для планування ІТ-проєктів. 27-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». 2023. Т. 6. С. 373–374.

41. Методичні вказівки щодо розробки та оформлення кваліфікаційної роботи другого (магістерського) рівня вищої освіти за освітньо-науковою програмою «Управління проектами в галузі інформаційних технологій» / Упоряд.: Петров К.Е., Левикін В.М., Чалий С.Ф., Євланов М.В., Міхнов Д.К., Міхнова А.В., Чала О.В. – Харків: ХНУРЕ, 2024. – 24 с