https://doi.org/10.31891/2219-9365-2024-78-4
УДК 005.21:005.8:004.8.

ЛИСЕНКО Сергій
https://orcid.org/0000-0001-7243-8747
Хмельницький національний університет
МАНДРИК Андрій
Хмельницький національний університет

МЕТОД ОПТИМІЗАЦІЇ ПЛАНУВАННЯ ПРОЕКТІВ ТА ФОРМУВАННЯ КОМАНД З ВИКОРИСТАННЯМ ГЕНЕТИЧНОГО АЛГОРИТМУ

У цій роботі розглянуто методологічні та практичні аспекти оптимізації планування проектів та формування команд з використанням генетичних алгоритмів. Особливий акцент зроблено на важливість правильного технічного оснащення команди, раціональності планування та бюджетування проекту. Основна мета полягає в підвищенні ефективності управління проектами, зниженні ризиків невдалих програмних проектів та підвищенні ефективності використаних ресурсів. Існуючі статистичні дані показують, що лише мала частина проектів виконується успішно, тому виникає необхідність застосування новітніх методів для підвищення їх ефективності.

Ключові слова: оптимізація планування, управління проектами, генетичний алгоритм, формування команд, сфера IT, моделювання, ресурсне планування, технічне оснащення, бюджетування проекту, раціональність планування.

LYSENKO Sergii, MANDRYK Andrii
Khmelnytskyi National University

A METHOD FOR OPTIMIZING PROJECT PLANNING AND TEAM FORMATION USING A GENETIC ALGORITHM

This work discusses the methodological and practical aspects of optimizing project planning and team formation using genetic algorithms. Particular emphasis is placed on the importance of proper technical equipment of the team, rationality of project planning and budgeting.

The main goal is to improve the efficiency of project management, reduce the risks of unsuccessful software projects, and increase the effectiveness of the resources used. Existing statistics show that only a small part of projects are successfully completed, so there is a need to apply the latest methods to improve their efficiency.

In addition to exploring the possibilities of using genetic algorithms in project planning, the article also considers the methodology of their application for personnel planning. In particular, attention is focused on the accurate modeling of the mathematical features of project planning and team formation. This allows us to choose the best strategies for genetic operations such as selection, crossover, and mutation aimed at maximizing fitness over generations. These techniques allows dynamically respond to changes in project conditions, optimizing resources and task execution time. Thanks to genetic algorithms, teams can reduce project implementation time while improving the quality of tasks. Key applications include assigning tasks to employees based on their qualifications and experience. Using hybrid coding, we managed to ensure effective task planning and distribution of responsibilities among the team. Based on the tests, it was found that stochastic selection of the initial population and genetic operations play a key role in the convergence of the algorithm. In general, the studies point to numerous advantages of using genetic algorithms in the field of project planning and team building, and question the need for further research in this area.

Keywords: optimization of planning, project management, genetic algorithm, team formation, IT sphere, modeling, resource planning, technical equipment, project budgeting, planning rationality.

ПОСТАНОВКА ПРОБЛЕМИ У ЗАГАЛЬНОМУ ВИГЛЯДІ ТА ЇЇ ЗВ'ЯЗОК ІЗ ВАЖЛИВИМИ НАУКОВИМИ ЧИ ПРАКТИЧНИМИ ЗАВДАННЯМИ

У сучасному світі ефективне управління проектами має вирішальне значення для того, щоб організації залишалися конкурентоспроможними. Оскільки проекти стають дедалі складнішими, а команди - більш різноманітними та географічно розподіленими, традиційних підходів до управління проектами вже недостатньо. Саме тут стають у нагоді інформаційні технології, зокрема, інтелектуальне управління проектами. Використовуючи можливості штучного інтелекту (ШІ) і передових технологій планування, організації можуть здійснити революцію в плануванні роботи команди і оптимізувати розподіл робіт, що призведе до поліпшення результатів проекту.

Інформаційні технології мають значний вплив на різні аспекти управління проектами. Від інструментів комунікації та співпраці до програмного забезпечення для планування та виконання проектів - технології трансформували способи спільної роботи команд та управління проектами. Однак традиційні підходи до управління проектами часто не справляються з викликами, що виникають у складних проектах і динамічному командному середовищі. Саме тут на допомогу приходить інтелектуальне управління проектами.

Інтелектуальне управління проектами використовує штучний інтелект, машинне навчання та вдосконалені алгоритми планування для оптимізації планування роботи команди, розподілу робочого навантаження та управління ресурсами. Автоматизуючи повторювані завдання та надаючи інформацію на основі даних, інтелектуальні системи управління проєктами дозволяють організаціям приймати обґрунтовані рішення, оптимізувати процеси та підвищувати загальну ефективність проєктів.

Одним з основних обов'язків керівника програмного проекту є визначення того, яка робота буде виконана, як і коли вона буде виконана. Цей обов'язок полягає у визначенні різних продуктів, які мають бути доставлені, оцінці зусиль для виконання кожного завдання, а також у побудові розкладу проекту. Зважаючи на важливість цієї діяльності, вона має бути пріоритетною цієї діяльності, вона повинна мати пріоритет над усіма іншими, і, крім того, графік проекту повинен регулярно оновлюватися, щоб відповідати поточному стану проекту.

Однією з основних проблем планування проекту є проблема представлення [2]. Зокрема, представлення, які надають більшість інструментів планування проектів, "не можуть моделювати еволюційну та паралельну природу розробки програмного забезпечення". Однією з найбільш практичних проблем, з якими стикаються менеджери проектів при побудові розкладів проектів, є той факт, що задачі планування проектів є NP-повними. Не існує алгоритмів, які можуть забезпечити оптимальне рішення за поліноміальний час, а це означає, що методи грубої сили в основному не можуть застосовуватись для вирішення цих задач [3, 4]. Ця проблема загострюється через те, що програмні проекти є нематеріальними за своєю природою та трудомісткими, а отже, пов'язані з ще вищим рівнем складності та невизначеності. Як наслідок унікальності програмного забезпечення, менеджери проектів не можуть повністю покладатися на використання досвіду попередніх проектів, а також не можуть використовувати інформацію про минулі проекти, окрім як в якості орієнтовних вказівок.

Стратегією, яка може бути використана для поборення цієї проблеми, є генетичні алгоритми для оптимізації процесу планування. Генетичні алгоритми - це методи оптимізації та пошуку рішень, які наслідують принципи природного відбору та генетики. Вони можуть бути використані для знаходження оптимальних або підходящих рішень у просторах великої розмірності або там, де традиційні методи оптимізації є неефективними.

У контексті планування програмних проектів генетичні алгоритми можуть бути використані для знаходження оптимального розкладу завдань та ресурсів, при цьому враховуючи різні обмеження та пріоритети проекту. Наприклад, вони можуть бути налаштовані так, щоб мінімізувати загальний час виконання проекту, максимізувати використання ресурсів або забезпечити виконання ключових завдань у встановлені терміни.

2 Відомі методи планування з використанням математичних алгоритмів та штучного інтелекту

Типовий графік роботи в складається з тижневого циклу або повторюваного циклу протягом декількох тижнів. Роботу зазвичай поділяють на різні зміни, а саме: ранкова зміна (Р), вечірня зміна (В) та нічна зміна (Н) [1]. Такий розподіл роботи часто зустрічається у сфері послуг, наприклад, у лікарнях та готельно-ресторанному бізнесі. В одному дослідженні планували позмінну роботу медсестер у лікарні. Розподіл медсестер за часовими інтервалами був організований таким чином, що графік роботи медсестер був складений таким чином, щоб задовольнити всі вимоги до планування. Було розроблено графік з чотиритижневою тривалістю часового інтервалу. Обмеження, яких можна було уникнути в цьому графіку роботи медсестер, - це жорсткі та м'які обмеження. Жорсткі обмеження включали наявність певної кількості медсестер у певних змінах, а також те, що медсестра повинна працювати кожні треті вихідні тощо. М'які обмеження включали уникнення понаднормових годин, уникнення трьох днів роботи поспіль тощо. Рішення розкладу в генетичному алгоритмі було представлено хромосомою [2-3]. У цьому дослідженні було використано два типи хромосом, перший - традиційна біт-рядкова структура для представлення кожного розкладу.

Другим рішенням було використання двовимірних масивів хромосомних структур для представлення кожного розкладу. Результати попереднього дослідження показали, що планування за допомогою двовимірних хромосом дає кращі результати, ніж планування за допомогою бітової хромосоми [4]. Попередні дослідження на цю тему здебільшого розглядають питання планування змінної роботи медсестер з точки зору різних цільових функцій. У цьому дослідженні цілі планування були спрямовані на створення розкладу, який одночасно мінімізує витрати на житло персоналу лікарні та вирівнює загальну кількість понаднормових годин, окрім підтримки високого рівня обслуговування. Для отримання оптимального розкладу роботи медсестер з урахуванням кількості виходів на лікування з імітаційної моделі було використано генетичний алгоритм (ГА) з двоточковим кросинговером і випадковою мутацією. Після запуску алгоритму було порівняно витрати та загальну кількість медсестер між існуючим графіком роботи медсестер та графіком, отриманим за допомогою ГА. У січні 2013 року графік роботи медсестер, отриманий за допомогою ГА, дозволив заощадити 12% витрат на оплату праці на місяць і 13% на кількості медсестер

порівняно з існуючим графіком, при цьому зберігаючи рівний розподіл кількості понаднормових працівників [5, 6].

Системи вдосконаленого планування та складання графіків (APS) застосовують алгоритми оптимізації для подолання обмежень фіксованої логіки, сприяючи створенню більш реалістичного плану [7, 8], який збалансовує як ресурси, так і матеріали, таким чином покращуючи виробниче планування у виробничих умовах. Ці інтелектуальні підходи до оптимізації спрямовані на мінімізацію виробничого графіка за допомогою математичної моделі, що призводить до ефективного планування та оцінювання складних виробничих процесів [9].

Еволюційні обчислення імітують механізм еволюції організмів для вирішення оптимізаційних задач. Разом з метаевристичними алгоритмами вони стали каталізатором численних досліджень, присвячених проблемам оптимізації в реальному світі. Ці методи особливо корисні для генерації корисних структур і покриття масивів для комбінаторного тестування, що сприяє пошуку ефективних рішень [10-12]. Гібридні методи оптимізації, що поєднують інтелект рою та еволюційні алгоритми, привертають все більшу увагу завдяки своєму потенціалу в глобальній оптимізації. Методи, натхненні біологією, такі як оптимізація рою частинок та оптимізація мурашиних колоній, забезпечують надійні рішення для складних проблем [13, 14]. Ці методи були застосовані в різних сферах, включаючи водні ресурси та інженерні проблеми, демонструючи багатообіцяючі результати у вирішенні реальних проблем оптимізації [15].

Аналогічним чином можна скласти графік позмінної роботи лікарів у педіатричному відділенні Військово-медичного центру імені принца Султана в Ер-Ріяді, Об'єднані Арабські Емірати [16]. Підхід генетичного алгоритму з використанням матриці біт-вартість, де кожна клітинка показувала наявність порушення обмежень. Результат випробування показав, що запропонований метод дозволив скласти розклад лікарів швидше і з меншою кількістю порушених обмежень порівняно з традиційним ручним методом [17, 18]. В освітній сфері потрібне планування за допомогою ГА для планування лекцій або викладання. Ця форма планування може бути одноцільовою або багатоцільовою. Одиничне планування представлено дослідженням для планування роботи викладача компанії [19]. Тим часом, багатоцільове планування представлено дослідженням планування лекцій, де це планування має кілька змінних, таких як курси, лекція, студенти, лекційна аудиторія та часовий інтервал [20, 21]. Як правило, при складанні розкладу за допомогою генетичного алгоритму, ресурси розкладу є статичними даними. Як правило, при плануванні за допомогою генетичного алгоритму ресурси планування є статичними даними. Однак в реальності ресурси планування найчастіше є динамічними даними, що підтверджується попередніми дослідженнями використання генетичного алгоритму для динамічного планування [22].

3 метод оптимізації планування проектів та формування команд з використанням генетичного алгоритму

Для початку потрібно визначити набір даних з яким у майбутньому буде працювати алгоритм. Кожен проект має набір завдань, кожне з яких має свої унікальні вимоги. Так само у кожного проєкту є команда, кожен член якої має свої навички та можливості. Ця інформація формує основу вхідних даних нашого алгоритму, таблиця 1.

Таблиця 1.

Вхідні дані

ID завдання	Опис завдання	Необхідні навички	Тривалість (дні)
T1	Дизайн UI/UX	Дизайн, творчість	7
T2	Backend API	Python, бази даних	10
T3	Розробка інтерфейсу	JavaScript, HTML, CSS	15
T4	Налаштування бази даних	SQL, бази даних	5
T5	Розробка мобільних додатків	Swift, Android	20
T6	Тестування та QA	Тестування, орієнтований на деталі	10

Первинне представлення рішення - це хромосома, послідовність, де кожен ген представляє завдання і призначеного члена команди. Можна уявити, що це список, в якому кожна позиція - це завдання і член команди, відповідальний за нього. Після цього генерується випадковий набір цих хромосом, формуючи початкову популяцію. Ця популяція містить безліч різних завдань для членів команди, деякі з яких потенційно кращі за інші.

У генетичних алгоритмах представлення рішення має вирішальне значення. Конкретний метод використовує хромосому для представлення потенційного рішення проблеми планування проекту та формування команди. Кожен ген у цій хромосомі відповідає завданню та призначеному члену команди для виконання цього завдання.

$$C = [g_1, g_2, \dots, g_n] \quad (1)$$

Коли початкова популяція сформована, алгоритм вступає в основний ітеративний цикл, імітуючи процес еволюції.

Для оцінки придатності перш ніж вирішити, які рішення повинні "вижити" і "розмножитися", потрібно дізнатись, наскільки вони хороші. Саме тут у гру вступає фітнес-функція. Вона оцінює кожну хромосому на основі того, наскільки добре члени команди, наведені у таблиці 2 відповідають завданням, і карає конфігурації з будь-якими конфліктами або невідповідностями ресурсів.

Таблиця 2.

Члени команди

ID учасника	Ім'я учасника	Навички	Доступність (дні/місяць)
M1	Аліса	Дизайн, HTML, CSS	20
M2	Боб	Python, бази даних	18
M3	Чарлі	JavaScript, HTML, CSS	22
M4	Дана	SQL, бази даних	19
M5	Емілі	Swift, дизайн	20
M6	Френк	Тестування, Java, Android	21

Маючи результати фітнесу, можна обирати, які хромосоми стануть батьками для наступного покоління. Ті, що мають вищий фітнес, мають більше шансів, гарантуючи, що хороші рішення з більшою ймовірністю передадуть свої "гени". Після чого батьків об'єднують у пари і проводять кросинговер. Це генетичне змішування має на меті об'єднати сильні сторони рішень двох батьків у їхніх нащадків. Наприклад, якщо один з батьків відмінно виконує першу половину завдань, а інший - другу, їхні нащадки можуть успадкувати найкраще від обох.

У пригоді так стають мутації, так як природа не ідеальна, як і алгоритм. Щоб внести деяку випадковість і уникнути застрявання в неоптимальних рішеннях, деякі гени у нащадків випадково змінюються. Ця мутація забезпечує різноманітність рішень і іноді може призвести до несподівано ефективних командних конфігурацій.

Через те що не бажано втратити наші найкращі рішення під час цього процесу. Елітарність гарантує, що найефективніші рішення від поточного покоління безпосередньо передаються наступному, зберігаючи їхню "генетичну" інформацію. Після відбору, кросинговеру, мутацій та елітизму формується нове покоління рішень. Це покоління замінює старе, і алгоритм переходить до наступної ітерації.

Алгоритм не працює нескінченно. Він зупиняється на основі певних критеріїв, таких як досягнення заданої кількості поколінь, досягнення рішення із задовільною придатністю або помітивши, що поліпшення зупинилися на певному рівні протягом декількох ітерацій.

Після завершення алгоритм представляє найкраще рішення, яке він знайшов. Цей результат - детальний розподіл членів команди за завданнями, оптимізований для забезпечення найбільш ефективної відповідності навичок вимогам завдання. Це рішення супроводжується набором метрик, які підсумовують оцінку придатності, загальну прогнозовану тривалість та інші відповідні деталі, набір параметрів алгоритму наведено у таблиці 3.

Таблиця 3.

Параметри ГА

Назва параметра	Опис	Типове значення
Чисельність населення	Кількість рішень в одному поколінні	100
Частота кросинговеру	Ймовірність, з якою відбувається кросинговер між рішеннями	0.8 (80%)
Частота мутацій μ	Ймовірність, з якою ген мутує у розв'язку	0.05 (5%)
Фракція елітарності α	Частка найкращих рішень, які безпосередньо передаються до наступного покоління	0.1 (10%)
Максимум поколінь	Максимальна кількість поколінь до припинення	500

Фінальний алгоритм - це не просто базовий ГА. Він має кілька розширених функцій:

1. Динамічна фітнес-функція: Коли вимоги проекту змінюються, наша фітнес-функція адаптується, гарантуючи, що алгоритм залишається актуальним навіть у мінливих сценаріях.
2. Адаптивна швидкість мутації: Замість статичної швидкості мутацій наш алгоритм визначає, коли різноманітність зменшується, і збільшує мутацію, щоб відновити варіативність.
3. Багатоцільова оптимізація: Проекти часто жонглюють кількома цілями, наприклад, мінімізацією витрат або максимізацією якості. Наш алгоритм може збалансувати та оптимізувати ці цілі.
4. Гібридні підходи: Ми досліджували можливості поєднання нашого ГА з іншими методами оптимізації, щоб підвищити його ефективність і якість рішень.

По суті, покращений генетичний алгоритм забезпечує динамічний, адаптивний і надійний підхід до складного завдання планування проекту та формування команди. Він ітеративно вдосконалює рішення,

забезпечуючи формування команд і призначення завдань таким чином, щоб оптимізувати продуктивність, ефективність і використання ресурсів.

Ефективність роботи ГА значною мірою визначається її фітнес-функцією. Наша функція оцінює як сумісність навичок членів команди з вимогами завдань, так і будь-які потенційні конфлікти в розкладі.

$$f(C) = \sum_{i=1}^{n} s(g_i) - p(g_i) \tag{2}$$

Де:

- $s(g_i)$ Оцінка сумісності навичок члена команди, призначеного на завдання
- $p(g_i)$ Штраф за будь-які конфлікти в задачі

Конкретний метод використовує імовірнісний підхід для відбору хромосом на основі їхніх фітнес-оцінок. Це гарантує, що більш придатні хромосоми мають більше шансів бути відібраними, але також дає можливість відібрати менш придатні хромосоми, підтримуючи різноманітність.

$$P(C) = \frac{f(C)}{\sum_{i=1}^{m} f(C_j)} \tag{3}$$

Де m - загальна кількість хромосом у популяції.

Кросинговер поєднує генетичну інформацію двох батьків для отримання потомства. Наш метод використовує одноточковий кросинговер. Для двох батьків C_1 і C_2 обирається випадкова точка кросинговеру. Гени до цієї точки беруться від C_1, а гени після - від C_2 для формування першого нащадка, і навпаки для другого нащадка. Для батьків C_1 та C_2, якщо точка перетину k, тоді:

$$\text{Нащадок 1} = \left[g_1^{C_1}, g_2^{C_1}, \ldots, g_k^{C_1}, g_{k+1}^{C_2}, \ldots, g_n^{C_2}\right] \tag{4}$$
$$\text{Нащадок 2} = \left[g_1^{C_2}, g_2^{C_2}, \ldots, g_k^{C_2}, g_{k+1}^{C_1}, \ldots, g_n^{C_1}\right]$$

Щоб зберегти різноманітність і уникнути передчасної конвергенції, наш метод вводить мутації. Для кожного гена в хромосомі, з невеликою ймовірністю μ, завдання змінюється на іншого члена команди.

Алгоритм завершує роботу на основі попередньо визначених критеріїв, таких як: досягнення максимальної кількості поколінь або знайдення рішення із задовільним рівнем пристосованості. Проте також якщо не спостерігається значного покращення пристосованості протягом заданої кількості поколінь.

На вхід до алгоритму подається

1. Початкова популяція: Випадково згенеровані завдання для команд.
2. Функція пристосованості: Визначена вище.
3. Параметри: Частота кросинговеру, частота мутацій μ, максимальна кількість поколінь.
4. Інформація про проект: Деталі завдання, необхідні навички, члени команди, їхні навички.

У результаті отримуємо хромосому з найвищим показником фітнесу. Та оцінку найкращого рішення.

Хоча початкове представлення рішення передбачало пряме зіставлення завдань членам команди, також можливо розглянути інші представлення, щоб задовольнити більш складні сценарії, таблиця 4 представляє результат оптимізації алгоритму, а таблиця 5 роботи алгоритму.

Таблиця 4.

Результат оптимізації

ID завдання	ID призначеного учасника	Ім'я призначеного учасника	Фітнес-внесок
T1	M5	Емілі	0.92
T2	M2	Боб	0.85
T3	M3	Чарлі	0.89
T4	M4	Дана	0.87
T5	M1	Аліса	0.9
T6	M6	Френк	0.88

Таблиця 5.

Результат роботи алгоритму

Метрика	Значення
Загальна придатність найкращого рішення	8.65
Загальна тривалість проекту (дні)	45
Кількість виконаних поколінь	350

Замість однієї хромосоми можна використовувати підхід з двома хромосомами, де одна хромосома представляє завдання, а інша - послідовність, в якій завдання повинні бути виконані.

$$C_{\text{завдання}} = [g_1, g_2, \dots, g_n] \tag{5}$$

$$C_{\text{послідовність}} = [t_1, t_2, \dots, t_n]$$

Функція статичної придатності може бути додатково розширена, щоб пристосуватися до динамічних вимог проекту або зміни доступності членів команди.

Статичну функцію придатності можна вдосконалити, щоб врахувати динамічні вимоги проєкту або зміну доступності членів команди. Таким чином можна запровадити фактор розпаду часу, щоб визначити пріоритетність завдань, які є більш чутливими до часу.

$$f(C) = \sum_{i=1}^{n} s(g_i) \times e^{-\lambda t_i} - p(g_i) \tag{6}$$

Де λ - константа розпаду, а t_i позначає час, що минув з моменту введення завдання.

Щоб гарантувати, що ГА не втратить найкращі знайдені рішення з покоління в покоління, використовується елітарність. Частина найкращих рішень з поточної популяції безпосередньо передається наступному поколінню.

$$n_{\text{elite}} = \alpha \times m \tag{7}$$

Де α - частка елітних розв'язків, а m - розмір популяції.

Ще одним покращення може стати заміна фіксованої швидкості мутацій μ на адаптивну швидкість мутації, яка змінюється залежно від різноманітності популяції. Якщо популяція передчасно конвергує, збільште швидкість мутації.

$$\mu = \mu_{\text{base}} + \delta \times \text{Фактор різноманітності} \tag{8}$$

Де μ_{base} - швидкість базової мутації, а δ - коефіцієнт масштабування.

Проекти часто мають кілька цілей, таких як мінімізація витрат, мінімізація часу та максимізація якості. ГА можна розширити, щоб впоратися з багатоцільовою оптимізацією, використовуючи такі методи, як оптимальність за Парето. Нижче наведена саме така модифікована формула де w_i - ваги, а $f_i(C)$ є індивідуальними цільовими функціями.

$$f(C) = w_1 \times f_1(C) + w_2 \times f_2(C) + \dots + w_k \times f_k(C) \tag{9}$$

Щоб ще більше підвищити ефективність методу, ми можемо комбінувати ГА з іншими методами оптимізації, такими як імітаційний відпал або оптимізація рою частинок. Це може забезпечити швидшу збіжність і ширший пошук простору розв'язків, але ці методики не були розглянуті в цій роботі.

4. Експериментальні дослідження методу та програмна реалізація

У цій роботи було використано метод на основі генетичного алгоритму, реалізований за допомогою бібліотеки DEAP та мови програмування Python [23].

Реалізація використовує DEAP (Distributed Evolutionary Algorithms in Python), універсальну бібліотеку, призначену для еволюційних алгоритмів. Модульний дизайн DEAP забезпечує адаптивність, дозволяючи нам формувати генетичний алгоритм відповідно до нюансів нашої проблеми. Бібліотека також полегшує відстеження еволюційної статистики, надаючи уявлення про розвиток алгоритму протягом поколінь.

Початковий експеримент полягав у виконанні шести різних завдань, кожне з яких вимагало певних навичок. Було вирішено об’єнати ці завдання з командою, що складалася з шести членів, кожен з яких мав

унікальну комбінацію навичок. Щоб забезпечити всебічне дослідження простору рішень, генетичний алгоритм був налаштований на ітерації протягом 40 поколінь, кожне з яких містило популяцію з 50 потенційних командних утворень. З ймовірностями кросинговеру та мутації, зафіксованими на рівні 0,7 та 0,2 відповідно, алгоритм мав широкі можливості як комбінувати успішні рішення, так і вводити випадкові варіації.

У міру розгортання ітерацій алгоритм демонстрував чітку траєкторію вдосконалення. До кульмінації 40-го покоління він визначив оптимальний склад команди, який отримав оцінку придатності 6.0. Цей показник свідчив про ідеальну узгодженість, коли навички кожного члена команди резонували з вимогами поставленого перед ним завдання. Наприклад, оптимальне рішення отримала учасниця команди, наділена дизайнерськими навичками, Аліса, яка отримала завдання "Дизайн UI/UX", що було влучно позначено як "Дизайн UI/UX", результат роботи програми у консолі показано на рисунку 1.

```
me $
     gen     nevals
     0   50
     1   31
     2   42
     ... [intermediate generations and evaluations]
     39  36
     40  29

     Best individual:  ['Alice', 'Bob', 'Charlie', 'Dana', 'Emily', 'Frank']
me $ Fitness:  6.0
```

Рис. 1. Результат роботи алгоритму

Помітною перевагою алгоритму була його ефективність. Навіть коли складність проблеми зростала, при десятикратному збільшенні кількості завдань і членів команди, алгоритм незмінно знаходив якісні рішення за відносно короткий проміжок часу, рисунок 2. У порівнянні з традиційними, ручними методами формування команд, перевага нашого алгоритмічного підходу була очевидною. Традиційні стратегії, часто залежні від людського судження, незмінно були більш схильні до помилок, що призводило до неоптимальних конфігурацій команд і, як наслідок, до подовження термінів виконання проектів.

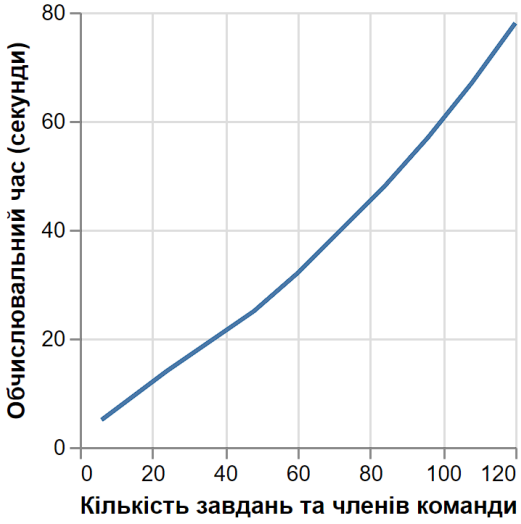

Рис. 2. Обчислювальний час зі збільшенням складності

Ефективність можна представити у вигляді кількості поколінь, необхідних для збіжності алгоритму до задовільного розв'язку. Зі збільшенням складності (тобто збільшенням кількості завдань і членів команди) ми очікуємо, що кількість поколінь, необхідних для збіжності, також зростатиме, але не лінійно, що пов'язано з ефективністю алгоритму, рисунок 3.

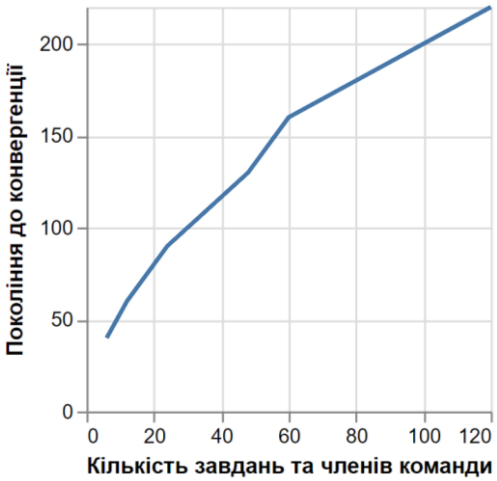

Рис. 3. Відношення кількісті завдань та членів команди для конвергенції (зближення)

Набір даних показує на рисунку 4 демонструє, як покращується показник придатності найкращого рішення в міру того, як алгоритм розвивається протягом декількох поколінь. Вищий показник придатності вказує на краще узгодження членів команди із завданнями.

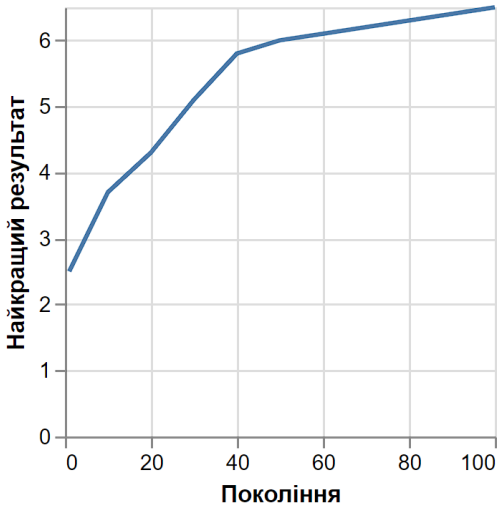

Рис. 4. Покращення фітнес-оцінки з поколіннями

Хоча поточна версія нашого програмного забезпечення вправно справляється з найпростішими проектними сценаріями, у неї є широкі можливості для вдосконалення. Серед запланованих удосконалень - інтеграція методів багатоцільової оптимізації, адаптація до проектів з мінливими вимогами, а також поєднання інших парадигм оптимізації для подальшого підвищення якості рішень. Крім того, щоб зробити інструмент більш зручним для користувачів, особливо для тих, хто не має досвіду роботи з обчислювальною технікою, ми розглядаємо можливість впровадження графічного інтерфейсу користувача, що спростить процес введення даних і візуалізації результатів.

ВИСНОВКИ З ДАНОГО ДОСЛІДЖЕННЯ І ПЕРСПЕКТИВИ ПОДАЛЬШИХ РОЗВІДОК У ДАНОМУ НАПРЯМІ

У роботі представлено результати вичерпного дослідження, присвяченого методу оптимізації планування проектів та формування команд за допомогою генетичних алгоритмів (ГА).

Завдяки ретельному дослідженню стало очевидним, що використання підходу на основі ГА для формування команд і розподілу завдань є потужною стратегією для забезпечення оптимальних результатів проекту та ефективного розподілу ресурсів.

Цей метод підкреслює значне покращення командної гармонізації, гарантуючи, що завдання розподіляються відповідно до індивідуальних навичок та здібностей. Наслідком цього є не лише прискорене завершення проекту, але й помітне підвищення якості виконання завдань та задоволеності команди.

В основі успішного застосування цього методу лежать принципи природного відбору та генетики. Ці принципи, перекладені алгоритмічно, забезпечують динамічну та адаптивну структуру, яка може вирішувати безліч проблем, пов'язаних з різноманітними проектами та різними складами команд.

Далі в роботі було окреслено практичні аспекти реалізації цього методу, висвітлено програмні міркування та алгоритмічні нюанси, що лежать в його основі. Завдяки використанню можливостей бібліотеки DEAP, метод втілюється в життя, пропонуючи відчутне рішення для менеджерів проектів і керівників команд. Цей програмний інструмент, заснований на принципах еволюційних алгоритмів, уможливлює безперебійну реалізацію методу GA, гарантуючи досягнення цілей проекту з точністю та ефективністю. По суті, об'єднання структури генетичних алгоритмів з тонкощами проектного планування пропонує новаторський підхід, що встановлює новий стандарт в методології управління проектами.

References

1. Granfeldt C. Using a Genetic Algorithm to Solve the Rotating Workforce Scheduling Problem Linkoping Univ. 2018.
2. Omara F. A., Arafa M. M. Genetic algorithms for task scheduling problem J. Parallel Distrib. Comput. 2018. 13–22 pp.
3. Dean J. Staff Scheduling by a Genetic Algorithm with a Two-Dimensional Chromosome Structure Proc. 7th Int. Conf. 2018. 1–15 pp.
4. Planning and Scheduling Optimization. URL: https://www.researchgate.net/publication/354875938_Planning_and_Scheduling_Optimization#:~:text=Planning%20and%20Scheduling%20Optimization,a%20production%20schedule%20minimizing
5. How can AI improve production planning in manufacturing. URL: https://www.techtarget.com/searcherp/answer/How-can-AI-improve-production-planning-in-manufacturing#:~:text=AI%20to%20improve%20production%20planning,both%20resources%20and%20material(дата звернення: 18.08.2023)
6. Intelligent Optimization-Based Production Planning and Simulation Analysis for Steelmaking and Continuous Casting Process. URL: https://www.sciencedirect.com/science/article/abs/pii/S1006706X10601367#:~:text=It%20is%20integrated%20with%20the,Development%20Plan%20of%20China (дата звернення: 11.09.2023)
7. Evolutionary Computation and Meta-heuristics. URL: https://link.springer.com/chapter/10.1007/978-981-15-3685-4_1#:~:text=Evolutionary%20computation%20is%20an%20engineering,and%20generate%20a%20beneficial%20structure (дата звернення: 16.10.2023)
8. Evolutionary Computation and Metaheuristics. URL: https://link.springer.com/chapter/10.1007/978-3-662-43429-1_5#:~:text=Chapter%20First%20Online%3A%2001%20January,covering%20arrays%20for%20combinatorial%20testing (дата звернення: 21.08.2023)
9. A Tutorial On the design, experimentation and application of metaheuristic algorithms to real-World optimization problems. URL:https://www.sciencedirect.com/science/article/abs/pii/S2210650221000493#:~:text=In%20the%20last%20few%20years%2C,design%20uprightness%2C%20and%20performance (дата звернення: 13.09.2023)
10. Evolutionary algorithms, swarm intelligence methods, and their applications in water resources engineering: a state-of-the-art review. URL: https://iwaponline.com/h2open/article/3/1/135/74697/Evolutionary-algorithms-swarm-intelligence-methods (дата звернення: 10.08.2023)
11. Large-scale global optimization based on hybrid swarm intelligence algorithm. URL: https://dl.acm.org/doi/10.3233/JIFS-192162#:~:text=Over%20the%20last%20two%20decades,leading%20swarm%20intelligence%20based%20approaches (дата звернення: 26.09.2023)
12. Swarm-based hybrid optimization algorithms: an exhaustive analysis and its applications to electricity load and price forecasting. URL: https://link.springer.com/article/10.1007/s00500-023-07928-0 (дата звернення: 20.10.2023)
13. Implementation of fuzzy logic systems and neural networks in industry. URL: https://www.sciencedirect.com/science/article/abs/pii/S0166361596000747#:~:text=This%20paper%20presents%20details%20of,networks%20and%20fuzzy%20logic%20systems (дата звернення: 17.08.2023)
14. Application of neural networks and neuro-fuzzy models in construction scheduling. URL: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC10200797/#:~:text=Planning%20and%20scheduling%20of%20construction,linear%20relationships%20between%20t (дата звернення: 30.09.2023)
15. A state of the art review of intelligent scheduling. URL: https://link.springer.com/article/10.1007/s10462-018-9667-6#:~:text=In%20this%20paper%2C%20we%20provide,1%20Introduction (дата звернення: 04.10.2023)
16. Leksakul K., and Phetsawat S. Nurse scheduling using genetic algorithm Math. Probl. Eng. 2019 0–17 pp.
17. Alharbi A., Alqahtani K. A. Genetic Algorithm Solution for the Doctor Scheduling Problem The Tenth International Conference on Advanced Engineering Computing and Applications in Sciences. 2022. 91 p.
18. Kristiadi D., and Hartanto R. Genetic Algorithm for lecturing schedule optimization. 2019. 13-83 pp.
19. Parera S., Sukmana H. T., Wardhani L. K. Application of genetic algorithm for class scheduling. 2019. 3–7pp.
20. Mardiyono A. Intelligent System for Course Scheduling in Higher Educations Int. J. Inf. Technol. Bus. Manag. 2022. 29–34 pp.
21. Eletrotécnica E. Dynamic Scheduling for Maintenance Tasks Allocation supported by Genetic Algorithms. 2018. pp.
22. Kaleeswaran A., Ramasamy V., Vivekanandan P. Dynamic Scheduling Of Data Using Genetic Algorithm In Cloud Computing Int. J. Adv. Eng. 2018.
23. Swarm-based hybrid optimization algorithms: an exhaustive analysis and its applications to electricity load and price forecasting. URL: https://link.springer.com/article/10.1007/s00500-023-07928-0 (дата звернення: 20.10.2023)
24. DEAP is a novel evolutionary computation framework for rapid prototyping and testing of ideas. URL: https://deap.readthedocs.io/en/master/ (дата звернення: 10.08.2023)