ІНФОКОМУНІКАЦІЙНІ ТЕХНОЛОГІЇ ТА ЕЛЕКТРОННА ІНЖЕНЕРІЯ
INFORMATION AND COMMUNICATION TECHNOLOGIES, ELECTRONIC ENGINEERING

№ 3 (2), 2023

https://doi.org/10.23939/ictee2023.02.095

ГЕНЕТИЧНИЙ АЛГОРИТМ ЯК ЗАСІБ РОЗВ'ЯЗАННЯ ОПТИМІЗАЦІЙНИХ ЗАДАЧ

Я. Пиріг [ORCID: 0009-0001-2104-8439], **М. Климаш** [ORCID: 0000-0002-1166-4182], **Ю. Пиріг** [ORCID: 0000-0002-8973-4005], **О. Лаврів** [ORCID: 0000-0002-2847-764X]

Національний університет "Львівська політехніка", вул. С. Бандери, 12, Львів, 79013, Україна

Відповідальний за рукопис: Юлія Пиріг (e-mail: yuliia.v.klymash@lpnu.ua)

(Подано 12 липня 2023)

Розглянуто особливості використання генетичного алгоритму (ГА) для розв'язання оптимізаційних задач. Наведено класифікацію оптимізаційних задач. Детально описано структурні елементи ГА та їх роль для розв'язання задачі комівояжера. Для оцінювання впливу параметрів ГА на ефективність його застосування здійснено дослідження впливу розміру популяції на довжину маршруту комівояжера. На основі отриманих результатів показано, що розмір популяції впливає на довжину маршруту, а оптимальне значення розміру популяції для цієї задачі становить 150. Використання турнірного оператора відбору, оператора впорядкованого схрещування, інверсійного оператора мутації дало змогу сформувати маршрут комівояжера завдовжки 9271,735 км, який, на основі викладених у роботі результатів, є оптимальним для відвідування 29 міст.

Ключові слова: *генетичний алгоритм; задача комівояжера; маршрут; популяція.*
УДК: 519.876

1. Вступ

Генетичні алгоритми є потужним інструментом для розв'язання різноманітних оптимізаційних задач. Вони використовують принципи природного відбору та генетичної еволюції, щоб знайти оптимальні рішення у складних умовах [1].

Важливість застосування генетичних алгоритмів в оптимізації важко переоцінити. Вони можуть насамперед вирішити проблеми, для яких немає точного аналітичного розв'язку або у яких великі розмір та складність. Завдяки природі стохастичного пошуку, генетичні алгоритми можуть знаходити прийнятні розв'язки навіть у просторах з багатьма локальними максимумами або мінімумами. Другим важливим аспектом є здатність генетичних алгоритмів працювати з нелінійними та недиференційованими функціями. Це особливо корисно в областях, де велика кількість параметрів, що взаємодіють між собою.

Крім того, генетичні алгоритми є гнучкими і їх можна легко адаптувати до різноманітних задач та вимог. Вони можуть оптимізувати неперервні, дискретні або змішані проблеми, а також враховувати обмеження та умови, ураховуючи відомі або невідомі функції обмежень.

Однією із ключових переваг генетичних алгоритмів є їх еволюційний підхід. Вони працюють з популяцією рішень, яка еволюціонує із покоління у покоління, покращуючи якість розв'язків з

кожним наступним поколінням. Це дає змогу зберегти різноманітність, виявляти нові й ефективніші рішення та уникати застрягання у локальних оптимумах.

Нещодавні досягнення у генетичних алгоритмах, такі як розширення здатностей до мультимодальної оптимізації, розроблення нових видів мутацій та схрещувань, покращення методів відбору та пошуку, роблять їх ще потужнішими й ефективнішими в розв'язанні складних оптимізаційних задач, до яких належать задача комівояжера, планування розкладу, пакування рюкзака, маршрутизації, наземної транспортної логістики, найкоротшого шляху, розподілу ресурсів тощо [2].

Отже, генетичний алгоритм є важливим засобом, який дає змогу отримувати розв'язок для різних оптимізаційних задач.

2. Класифікація оптимізаційних задач

Для впорядкування та спрощення розуміння різних типів задач використовується їх класифікація. Вона допомагає визначити спільні характеристики та властивості задач, що належать до одного класу, а також розробити для них спеціалізовані методи та алгоритми .

На рис. 1 наведено основні типи оптимізаційних задач.

Рис. 1. Класифікація оптимізаційних задач

Класифікація за розмірністю дає змогу визначити різні підходи до оптимізації задач залежно від кількості змінних або параметрів.

В одновимірних задачах наявна одна змінна або параметр, що описує простір пошуку. Такі задачі є найпростішими формами оптимізаційних задач. У багатовимірних задачах простір пошуку складається із декількох змінних або параметрів. Кожна змінна має певні обмеження.

Відмінність між одновимірними та багатовимірними задачами полягає у кількості змінних і складності простору пошуку. У багатовимірних задачах кількість можливих комбінацій змінних значно більша, що призводить до їх вищої складності. Вибір між одновимірними та багатовимірними класифікаціями залежить від конкретної задачі та її характеристик.

Класифікація за структурою розглядає однокритеріальні та багатокритеріальні задачі.

Однокритеріальні задачі мають одну цільову функцію, яку потрібно мінімізувати або максимізувати. У цього типу задач є лише один оптимальний розв'язок, що задовольняє цільову функцію.

Багатокритеріальні задачі мають більше від однієї цільової функції, які потрібно оптимізувати. Мета таких задач – знайти набір розв'язків, що є компромісом між різними цілями. Кожен розв'язок у багатокритеріальних задачах називають "парето-оптимальним" або "недомінованим", оскільки він не може бути покращений у всіх цільових функціях без погіршення хоча б однієї з них. Багатокритеріальні задачі складніші, оскільки вимагають знаходження оптимального балансу між різними цілями.

Класифікація за типом функцій описує континуальні та дискретні задачі. Континуальні задачі – це клас оптимізаційних задач, де простір пошуку параметрів є неперервним. У таких задачах параметрам можуть присвоюватись будь-які значення ss певного діапазону, який визначений дійсними числами або векторами у *n*-вимірному просторі. Наприклад, у задачах оптимізації функцій на континуальному просторі пошуку параметрів цільова функція залежить від неперервних змінних. Мета таких задач – знайти набір значень параметрів, який мінімізує або максимізує значення цільової функції.

У дискретних задачах змінним присвоюється обмежена кількість дискретних значень, а цільова функція і обмеження можуть бути виражені як дискретні функції.

Класифікація за видом оптимізації поділяється на локальну та глобальну.

Локальна оптимізація спрямована на знаходження оптимального розв'язку в обмеженому просторі навколо поточної точки або локального максимуму/мінімуму. Пошук зосереджується на вузькій області простору параметрів, де здійснюють аналіз і покращення розв'язку. Цей підхід може бути ефективним, коли наявна інформація про початкове наближення або є результат, близький до оптимального розв'язку. Однак локальна оптимізація не гарантує знаходження глобально оптимального розв'язку, оскільки може застрягати в локальних максимумах або мінімумах.

Глобальна оптимізація орієнтована на знаходження глобально оптимального розв'язку в усьому просторі параметрів. Це потребує ширшого обсягу пошуку і розгалуженіших алгоритмів. Глобальна оптимізація може здійснюватись із перебиранням всіх можливих розв'язків або за допомогою розумних стратегій, таких як еволюційні алгоритми або стохастичні методи. Відмінністю цього виду оптимізації є те, що вона може знаходити глобально оптимальні розв'язки, але може потребувати більше обчислювальних ресурсів і часу для пошуку.

Отже, основна відмінність між локальною і глобальною оптимізацією полягає у масштабі пошуку і можливості досягнення глобально оптимального розв'язку. Локальна оптимізація зосереджена на невеликій області, де шукають локальний максимум або мінімум, тоді як глобальна оптимізація ставить за мету знайти глобально оптимальний розв'язок в усьому просторі параметрів.

Вибір між локальною та глобальною оптимізацією залежить від вимог задачі та доступних ресурсів, таких як обчислювальна потужність і час, які можна витратити на пошук розв'язку.

3. Структура та основні компоненти генетичного алгоритму

У класичному ГА кожне рішення кодується у певному вигляді – таке подання називається хромосомою, а саме рішення – індивідом. Множина рішень називається популяцією, ітерації алгоритму – поколіннями, пошукові оператори – схрещуванням і мутацією. Отже, для опису роботи генетичного алгоритму використовують терміни із біології.

ГА основані на колективному навчальному процесі всередині популяції індивідумів, кожен з яких являє собою певну точку в просторі допустимих рішень [3]. Спочатку популяція випадково ініціалізується, потім вона охоплює найкращі регіони пошукового простору за допомогою таких процесів, як схрещування та мутація.

Отже, ГА ґрунтується на таких операціях еволюції, як відбір, схрещування та мутація .

Схрещування важливе для збереження і поліпшення вмбраних рис у популяції, мутація важлива для генетичного розмаїття і запобігає стагнації популяції, природний відбір веде еволюцію до найпристосованіших популяцій, збільшуючи тривалість життя індивідів, що містять найпридатніші групи генів [4].

Як загальна стратегія, кількість членів одного покоління в майбутньому залишається незмінною, отже, ключовим параметром є розмір популяції: у надто малих популяціях починають домінувати окремі індивіди, що призводить до стагнації та мізерної вибірки простору параметрів, тоді як надто великі можуть сповільнити збіжність до оптимального розв'язку. Важливою особливістю ГА є притаманний йому паралелізм, що дає можливість в один і той самий час опрацьовувати різних членів популяції, тобто одночасно досліджуючи різні області простору параметрів.

На рис. 2 наведено блок-схему узагальненого ГА. Розглянемо особливості кожного із етапів.

Початкова популяція формується із потенційних рішень (індивідумів), які вибрані випадково. Кожне згенероване рішення подається як хромосома. Формат хромосом має відповідати прийнятим для розв'язуваної задачі правилам.

Для представлення маршруту всі міста кодуються як унікальні цілі числа та впорядковані в хромосому. Положення в хромосомі вказує на відвідуваний порядок міста. Якщо *i, j*=1, 2,…, n, тобто якщо місто і є *j*-м елементом хромосоми, це означає, що таке місто потрібно відвідати. Початкова популяція впливає як на швидкість збіжності, так і на можливості глобального пошуку ГА.

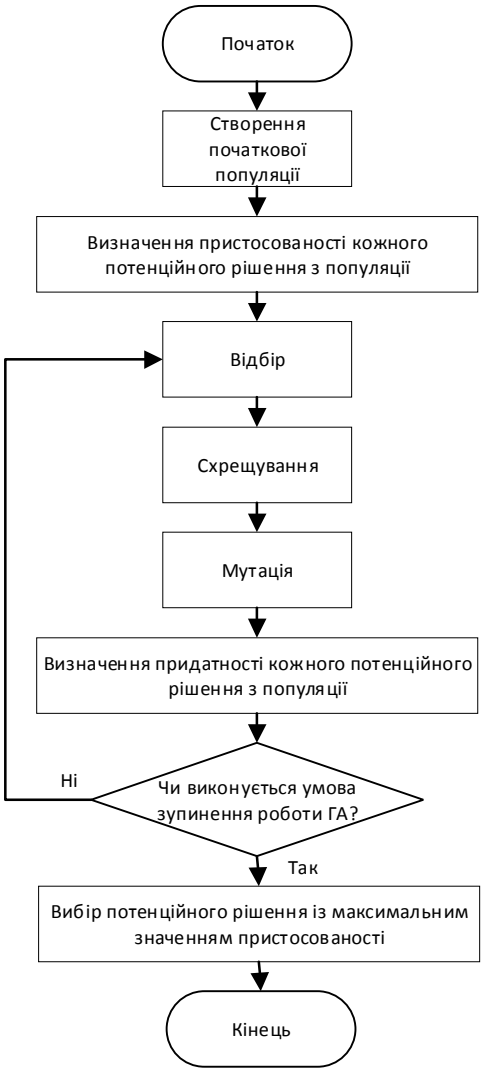

Рис. 2. Блок-схема узагальненого ГА

Хромосома може представлятись на основі таких методів:

– Двійкове кодування: кожна хромосома подається на основі бітів 0 і 1. Такий вид представлення застосовують для задачі пакування рюкзака.

– Перестановлене кодування: кожна хромосома подається у вигляді рядка чисел, що позначають певну позицію у послідовності. Цей метод використовують для задачі комівояжера.

– Кодування значень: хромосоми подаються за допомогою послідовності деяких значень, таких як дійсні числа, символи або об'єкти.

– Кодування дерева: кожна хромосома розглядається як дерево певних елементів, таких як команди чи функції.

Розглянемо найпоширеніші методи формування початкової популяції, які використовують для ГА [5, 6].

– Випадкова ініціалізація. Цей метод є найпростішою і найпоширенішою технікою, яка генерує початкову популяцію для ГА. Для задачі комівояжера відповідно до цього методу довільно вибирають міста. Під час генерації індивіда цей метод генерує випадкове число від 1 до *n*. Якщо поточний індивід вже містить згенероване число, то генерується нове число. В протилежному випадку згенероване число додається до поточного рішення. Операція повторюється доти, доки не буде досягнуто потрібного розміру популяції.

– Найближчий сусід. Цей метод часто використовують для формування початкової популяції. Генерація кожного рішення починається із вибору випадкового міста як початкового міста, а потім найближчого міста, що додається як нове початкове місто. Ітеративно метод додає найближче місто до поточного міста, яке не було додано до індивідума, доки не буде охоплено всі міста. Згенеровані рішення можуть покращити пошук у наступних поколіннях, оскільки вони були створені з міста, яке є найближчим до поточного.

– Вибіркова ініціалізація. Основою цього методу є використання підграфу K-найближчого сусіда (KHC) для побудови графа, що містить усі міста на основі матриці відстані. Вищий пріоритет надається ребрам підграфа KHC, відповідно з початкового міста наступне місто для маршруту буде випадково вибрано із списку KHC. Після того, які усі міста з цього списку вибрано, здійснюється випадковий вибір наступного міста із невідвіданих.

– Заповнення популяції на основі вставляння. Заповнення початкової популяції починається з часткового маршруту, який містить декілька випадково вибраних міст. Далі здійснюється ітеративне вставляння найближчого міста до будь-якого міста на частковому маршруті. Після цього виконується додавання ребра до позиції із найнижчою вартістю на маршруті .

Визначення пристосованості кожного потенційного рішення із популяції. Після створення початкової популяції визначають пристосованість її кожного рішення. Для цього використовують "функцію пристосованості (фітнес-функцію)", яка дає змогу оцінити якість кожного рішення і визначити, наскільки воно відповідає поставленій задачі або критеріям оптимальності. Визначення пристосованості може варіюватись залежно від конкретного завдання, але основні методи передбачають [7, 8]:

– Пряме обчислення. Цей метод визначає пристосованість кожного рішення із обчисленням значення функції пристосованості безпосередньо на основі вхідних даних. Наприклад, у задачах оптимізації це може бути об'єктивна функція, яка оцінює якість рішення за певними критеріями.

– Симуляційні моделі. У деяких випадках реальне обчислення пристосованості може бути складним або дорогим, тоді використовуються симуляційні моделі, що апроксимують функцію пристосованості. Вони дають змогу швидко оцінювати якість рішень без виконання реальних обчислень.

– Експертна оцінка. В деяких випадках пристосованість може бути визначена на основі експертних знань або експертної оцінки. Наприклад, у задачах класифікації, експерти можуть оцінювати якість класифікатора на основі його точності, використовуючи свої знання та досвід.

– Порівняння із ідеальним рішенням. Якщо ідеальне рішення відоме, то пристосованість можна визначити, порівнявши поточне рішення з ідеальним. Наприклад, у задачах оптимізації, де постійно збільшується значення цільової функції, пристосованість можна визначити як різницю між поточним значенням функції та ідеальним значенням.

– Використання випадкової проби. Якщо функцію пристосованості неможливо обчислити аналітично або із використанням симуляційних моделей, можливе використання випадкової проби або здійснення експерименту для оцінки пристосованості.

Визначення пристосованості є важливим етапом в ГА, оскільки на нього впливає процес вибору, розмноження та еволюції рішень в популяції. Якщо функція пристосованості некоректно відображає якість рішень або не враховує потреби задачі, це може призвести до неправильної збіжності або втрати цінних розв'язків.

Наступним етапом роботи ГА є відбір (селекція), на якому відбувається визначення тих хромосом, які будуть використовуватись на етапах схрещування та мутації. Для цього на початку кожної ітерації циклу роботи ГА із поточної популяції вибирають найпристосованіших індивідумів, які надалі стануть “ батьками” індивідумів у інших поколіннях.

Схрещування є одним з основних операторів ГА і відіграє важливу роль у здійсненні еволюційного пошуку. Його основна функція полягає в комбінуванні генетичного матеріалу (хромосом) двох батьківських індивідів з метою створення нащадка з новим набором генів. Схрещування в ГА забезпечує:

– Різноманітність. Схрещування збільшує різноманітність популяції завдяки комбінуванню генетичного матеріалу різних батьківських індивідів. Це дає змогу вносити нові комбінації генів і сприяє розширенню простору пошуку для знаходження оптимального розв’язку.

– Комбінацію корисних властивостей: Схрещування дозволяє поєднувати корисні властивості батьківських індивідів у нащадках. Якщо один батьківський індивід має високу придатність в одній частині простору пошуку, а інший – в іншій частині, то їх сполучення може привести до нащадка з високою придатністю в обох частинах.

– Уникнення передчасної збіжності. За наявності різних комбінацій генів в популяції збільшується ймовірність знаходження оптимального розв’язку й уникнення застрягання в локальних мінімумах.

Оператори схрещування відрізняються за підходом до комбінації генетичного матеріалу та типом результуючих нащадків. До найпоширеніших операторів схрещування належать:

– Одноточковий. Вибір точки перерізу здійснюється випадково в генетичних послідовностях батьків. Ліва частина генетичного матеріалу одного батька об’єднується із правою частиною генетичного матеріалу другого батька, щоб створити нащадка. Цей оператор простий у реалізації, проте може спричиняти невисокий рівень різноманітності у популяції.

– Багатоточковий. Декілька точок перерізу вибирають на випадкових позиціях в генетичних послідовностях батьків. Генетичний матеріал між цими точками перерізу обмінюється між батьками для створення нащадків. Цей оператор порівняно із одноточковим забезпечує вищий рівень різноманітності у популяції та допомагає уникнути передчасної збіжності алгоритму. Недоліком є складність реалізації та обчислень.

– Рівномірний. Кожен ген у генетичних послідовностях батьків має ймовірність бути вибраним для обміну. Батьки обмінюються генетичним матеріалом на основі цих ймовірностей, створюючи нащадків. Цей оператор забезпечує рівномірніший розподіл генетичного матеріалу між нащадками порівняно із попередніми операторами. До його недоліків зараховують здійснення великої кількості обчислень, що потребує відповідних ресурсів, та можливість втрати певних властивостей розглянутого рішення.

– Узагальнений. Здійснює комбінацію генетичного матеріалу батьків, використовуючи ваги, які відображають відстань між ними. Це дає змогу створити нащадків, які ближче до батьківських особин у просторі пошуку. Цей оператор може приводити до перехрещування генетичного матеріалу з різних областей простору рішень. Це допомагає ГА ефективніше рухатись в напрямку глобальних оптимумів, зокрема в складних задачах із багатьма локальними оптимумами.

Недоліками узагальненого оператора є висока обчислювальна складність через залучення додаткових механізмів для генерації нових комбінацій. Це може призвести до збільшення часу виконання генетичного алгоритму. Також цей оператор може призводити до втрати певної структури або залежності між генами. Це може вплинути на якість рішення та здатність алгоритму знаходити оптимальні комбінації.

– Змішаний. Цей оператор комбінує різні види схрещування, щоб забезпечити більшу різноманітність нащадків. Він може поєднувати одноточкове, багатоточкове, рівномірне схрещування або інші оператори, змінюючи їх параметри під час схрещування. А це сприятиме знаходженню оптимальних рішень швидше та ефективніше, оскільки використовуються різні стратегії схрещу-

вання. До недоліків змішаного оператора належать складність вибору операторів та залежність ефективності схрещування від їх вибору. Неправильний вибір операторів може призвести до недостатнього або недоцільного дослідження простору нових рішень, що може погіршити результати оптимізації. Крім цього, потрібно враховувати різні параметри та налаштування для кожного оператора схрещування, що ускладнює розв'язання задачі.

– Впорядкований. Цей оператор використовується для створення нових індивідів (рішень) за допомогою комбінування генетичної інформації батьківських індивідів. Операція впорядкованого схрещування виконується на основі послідовності генів у батьківських індивідах. Здійснюється вибір двох батьківських індивідів, після чого використовують випадкове розрізання. Далі вибирають діапазон генів одного з батьківських індивідів, і цей діапазон копіюється в нащадка. Ті гени з другого батьківського індивіда, які залишились, заповнюються у тій самій послідовності, що і в збереженому другому батьківському індивіді.

Отже, кожен із розглянутих операторів має певні переваги та недоліки, їх вибір залежить від конкретної задачі та типу даних, із якими працює ГА. Оператори схрещування допомагають внести різноманітність у популяцію, поєднуючи корисні характеристики батьківських особин та створюючи нові комбінації генетичного матеріалу, що можуть покращити якість рішення у ГА.

Оператори мутації використовують для внесення випадкових змін до індивідів у популяції. Це дає змогу розширити простір пошуку та зберегти різноманітність у популяції. До основних операторів мутації належать:

– Одноточкова мутація. Вибирається одна випадкова позиція в геномі й замінюється на нове випадкове значення. Наприклад, якщо геном представлений рядком бітів, то одноточкова мутація може змінити один біт з 0 на 1 або з 1 на 0.

– Багатоточкова мутація. За цієї мутації вибирають декілька випадкових позицій в геномі та замінюють на нові значення. Кількість точок мутації та нові значення можуть варіюватися від одного випадку до іншого.

– Інверсія. Здійснюється вибір певного діапазону генів у геномі, і послідовність цих генів обертається. Наприклад, якщо геном представлений послідовністю чисел, то інверсія може змінити послідовність [1, 2, 3, 4] на [4, 3, 2, 1].

– Вставляння. Вибирають певний діапазон генів у геномі, і цей діапазон переміщується на іншу випадкову позицію у геномі. Це може створити нові комбінації генів, які не існували в початковому індивіді.

– Заміна. У разі заміни вибирають одну випадкову позицію в геномі та замінюють на нове випадкове значення із заданого діапазону. Це можна застосовувати до числових генотипів.

– Додавання. Вибирають певний діапазон генів у геномі, і до цього діапазону додають новий ген з випадковим значенням. Це може дозволити інтродукцію нових варіантів генів до популяції.

– Видалення. Вибирають певний діапазон генів у геномі, і цей діапазон видаляють, зменшуючи довжину геному. Це може призвести до збільшення концентрації певних генів у популяції.

– Змішування. Вибирають два індивіди з популяції, які випадково обмінюються частинами їх геномів. Це дає змогу поєднувати корисні комбінації генів з різних індивідів.

Оператори мутації застосовують із певною ймовірністю до індивідів у популяції. Їх використання дає змогу вносити різноманітність і розвивати нові комбінації генів, що сприяє знаходженню оптимальніших рішень у ГА. Вибір певних операторів та їх параметрів залежить від конкретної задачі та може варіюватися для досягнення найкращих результатів.

Визначення придатності кожного потенційного рішення із популяції. Цей етап ГА застосовують для оцінювання якості рішень у контексті задачі, яку алгоритм намагається розв'язати. Внаслідок цього відбувається відбір кращих рішень та стимулювання їх розмноження та еволюції в майбутніх поколіннях популяції. Придатність оцінюють на основі об'єктивної міри, яка відображає, наскільки добре рішення виконує поставлене завдання. Визначення цієї міри залежить від конк-

ретної задачі, його може встановити користувач або встановлюється автоматично. Є різні форми оцінювання придатності, залежно від типу задачі. Є такі основні підходи до визначення придатності розглянутого рішення із популяції:

– Пряма функція від цільової функції. У деяких задачах, де цільова функція визначає якість рішення, придатність може бути простою функцією цільової функції. Це означає, що чим більше значення цільової функції, тим вища придатність рішення.

– Обернена функція від цільової функції. В деяких задачах, де здійснюється мінімізація цільової функції, можливе використання оберненої функції від цільової. Отже, що менше значення цільової функції, то вища придатність рішення.

– Рангова функція. У деяких випадках, коли немає однозначної цільової функції або коли потрібно враховувати декілька критеріїв, здійснюється сортування рішень за їх відносною якістю та присвоєння рангів. Рішення із вищим рангом вважається придатнішим.

– Порогова функція. В окремих випадках можна використовувати порогову функцію для визначення придатності рішень. Наприклад, можна встановити певний поріг, і всі рішення, які перевищують цей поріг, вважають придатними, тоді як рішення, що не досягають цього порога, відкидають.

Під час оцінювання придатності кожного рішення з популяції застосовують вибрані метрики та критерії, які відображають якість рішення. Ці метрики можуть враховувати обмеження задачі, ефективність, точність, швидкодію, економічні показники тощо, залежно від контексту задачі.

Оцінюють придатність кожного рішення, обчислюючи значення відповідної метрики або функції для кожного рішення. Чим вище це значення, тим кращим вважається рішення у контексті задачі. Після визначення придатності всіх рішень у популяції їх можна відсортувати за зниженням або зростанням значень придатності. Це дає змогу виділити найкращі рішення, які будуть використовуватися для формування нового покоління популяції .

Умови зупинки роботи ГА встановлюють залежно від конкретної задачі та потреб користувача. До них належать [9]:

– Досягнення максимальної кількості ітерацій. ГА може бути налаштований на виконання певної кількості ітерацій або поколінь. Після досягнення цієї максимальної кількості алгоритм зупиняється.

– Досягнення заданої придатності. ГА може бути зупинений, якщо одне зі згенерованих рішень досягає певного рівня придатності, який вважається достатньо хорошим для задачі.

– Зміна придатності нижче від заданого порога. Якщо середнє значення придатності популяції падає нижче від певного порогового значення, алгоритм може бути зупинений. Це може свідчити про те, що алгоритму не вдається знайти оптимальне рішення.

– Закінчення часу. ГА може бути зупинений після вичерпання заданого обмеження часу. Це використовується, коли час є критичним фактором, а користувач хоче отримати найкраще можливе рішення протягом обмеженого часу.

Комбінація цих умов може бути використана для зупинки ГА. Використовуючи генетичний алгоритм, важливо збалансувати компроміс між часом виконання та якістю знайденого рішення.

4. Особливості застосування генетичного алгоритму для різних задач оптимізації

Генетичний алгоритм є ефективним і гнучким методом для розв’язання різних оптимізаційних задач. Він може застосовуватись для пошуку оптимальних рішень в проблемах з різною природою і вимогами. Основне призначення ГА в оптимізації полягає в тому, щоб ефективно вирішувати проблеми зі складними просторами пошуку, обмеженнями, багатокритеріальними цілями та невідомими аналітичними функціями.

У табл. 1 наведено особливості застосування ГА залежно від розглянутої задачі оптимізації.

Таблиця 1

Особливості застосування генетичного алгоритму для різних задач оптимізації

Вид оптимізаційної задачі	Особливості застосування ГА
Задача комівояжера	– Подання рішення: може являти собою послідовність міст, які потрібно відвідати. У ГА ця послідовність може бути закодована у вигляді хромосоми, де кожен ген представляє одне місто. – Оцінка придатності: Щоб оцінити якість кожного рішення, потрібно обчислити відстань між містами у послідовності. Це роблять за допомогою функції оцінки, яка обчислює загальну відстань подорожі
Задача планування розкладу	– Представлення розкладу: Розклад може бути поданий у вигляді послідовності генів, де кожен ген відповідає окремій задачі або завданню, яке треба виконати. Наприклад, для розкладу університетських занять ген може вказувати на конкретний предмет і час проведення. – Функція пристосованості: повинна оцінювати, наскільки добре відповідний розклад задовольняє обмеження, такі як наявність вільного часу, мінімізація перекриття, забезпечення пріоритетів тощо
Задача пакування рюкзака	– Кодування рішення: це може бути бінарне кодування, де кожен біт відповідає наявності або відсутності предмета в рюкзаку. – Функція пристосованості: може враховувати два аспекти: сумарну вартість предметів у рюкзаку (яку треба максимізувати) і сумарну масу предметів (яку треба обмежити максимальною масою рюкзака)
Задача максимального потоку	– Представлення рішення: можна використовувати хромосоми, що складаються з послідовності ребер або вершин у мережі. Кожен ген хромосоми може кодувати наявність або вагу ребра або іншу інформацію, яка впливає на потік у мережі. – Функція пристосованості: може вимірювати загальний обсяг потоку у мережі, що проходить через задані ребра або вершини. Ця функція має бути максимізована, оскільки мета задачі – знайти максимальний потік у мережі

Важливо зазначити, що кожна задача оптимізації має певні особливості, і можуть бути розроблені спеціальні модифікації ГА, які враховують ці особливості.

Отже, варто налаштовувати та модифікувати ГА відповідно до конкретної задачі для досягнення найкращих результатів.

5. Дослідження використання генетичного алгоритму для задачі комівояжера

У задачі комівояжера міста можна позначити від 0 до n–1, а можливі рішення – послідовностями таких чисел. Для прикладу розглянемо прокладання оптимального маршруту для комівояжера, якому необхідно відвідати 29 міст Баварії. Для реалізації цього дослідження використано бібліотеку TSPLIB, яка містить приклади задач комівояжера для реальних міст, що є текстовими файлами [10].

У цій роботі використано загальнодоступний файл bayg29.tsp, у якому є дані про 29 міст Баварії (рис. 3). Ці міста утворюють набір точок, між якими задані відстані, і використовуються для прокладання маршруту комівояжера.

Детальніше розглянемо структуру цього файла:

– TYPE: тип задачі – "TSP" (симетрична задача комівояжера).

– COMMENT: коментар або опис задачі – "29 Cities in Bavaria, geographical distances (Groetschel,Juenger,Reinelt)".

– DIMENSION: кількість міст (розмір задачі) – 29.

– EDGE_WEIGHT_TYPE: тип ваги ребер – "EXPLICIT" (явно задані ваги).

– EDGE_WEIGHT_FORMAT: формат задання ваг ребер – "UPPER_ROW" (верхній трикутник матриці ваг).

– DISPLAY_DATA_TYPE: Тип відображення даних – "TWOD_DISPLAY" (двовимірне відображення).

Після цих полів наводять інформацію про ваги ребер та координати міст:

– EDGE_WEIGHT_SECTION: вказує, що наступні рядки містять ваги ребер.

У наступних рядках вказані ваги ребер у верхньому трикутнику матриці, відсортовані за рядками. Наприклад, у першому рядку містяться ваги ребер, що з'єднують перше місто з усіма іншими містами.

– DISPLAY_DATA_SECTION: вказує, що наступні рядки містять координати міст. У наступних рядках наведені номер міста та його координати (x, y).

На рис. 3 для наочності вказано координати тільки для одного міста, координати інших міст не наведено у зв'язку із великим розміром цього файла.

```
NAME: bayg29
TYPE: TSP
COMMENT: 29 Cities in Bavaria, geographical distances (Groetschel,Juenger,Reinelt)
DIMENSION: 29
EDGE_WEIGHT_TYPE: EXPLICIT
EDGE_WEIGHT_FORMAT: UPPER_ROW
DISPLAY_DATA_TYPE: TWOD_DISPLAY
EDGE_WEIGHT_SECTION
 97 205 139  86  60 220  65 111 115 227  95  82 225 168 103 266 205 149 120  58 257 152  52 180 136  82  34 145
129 103  71 105 258 154 112  65 204 150  87 176 137 142 204 148 148  49  41 211 226 116 197  89 153 124  74
219 125 175 386 269 134 184 313 201 215 267 248 271 274 236 272 160 151 300 350 239 322  78 276 220  60
167 182 180 162 208  39 102 227  60  86  34  96 129  69  58  60 120 119 192 114 110 192 136 173 173
 51 296 150  42 131 268  88 131 245 201 175 275 218 202 119  50 281 238 131 244  51 166  95  69
279 114  56 150 278  46 133 266 214 162 302 242 203 146  67 300 205 111 238  98 139  52 120
178 328 206 147 308 172 203 165 121 251 216 122 231 249 209 111 169  72 338 144 237 331
169 151 227 133 104 242 182  84 290 230 146 165 121 270  91  48 158 200  39  64 210
172 309  68 169 286 242 208 315 259 240 160  90 322 260 160 281  57 192 107  90
140 195  51 117  72 104 153  93  88  25  85 152 200 104 139 154 134 149 135
320 146  64  68 143 106  88  81 159 219  63 216 187  88 293 191 258 272
174 311 258 196 347 288 243 192 113 345 222 144 274 124 165  71 153
144  86  57 189 128  71  71  82 176 150  56 114 168  83 115 160
 61 165  51  32 105 127 201  36 254 196 136 260 212 258 234
106 110  56  49  91 153  91 197 136  94 225 151 201 205
215 159  64 126 128 190  98  53  78 218  48 127 214
 61 155 157 235  47 305 243 186 282 261 300 252
105 100 176  66 253 183 146 231 203 239 204
113 152 127 150 106  52 235 112 179 221
 79 163 220 119 164 135 152 153 114
236 201  90 195  90 127  84  91
273 226 148 296 238 291 269
112 130 286  74 155 291
130 178  38  75 180
281 120 205 270
213 145  36
 94 217
162
DISPLAY_DATA_SECTION
   1    1150.0  1760.0
```

Рис. 3. Частина структури файла bayg29.tsp

Загалом, цей код містить інформацію про задачу комівояжера з 29 містами в Баварії, враховуючи ваги ребер та координати міст. Ця інформація використовується для подальшої обробки та розв'язання задачі комівояжера.

На рис. 4 подано основні кроки для дослідження застосування ГА для задачі комівояжера.

Для визначення пристосованості використано функцію tspDistance, яка здійснює обчислення відстані між двома містами. Вона приймає географічні координати (широту і довготу) двох міст і повертає евклідову відстань між ними. У випадку задачі комівояжера географічні координати міст можуть бути використані як вхідні дані для функції tspDistance. Застосування цієї функції між будь-якими двома містами в графі дає змогу обчислити їхню відстань.

Загалом, цю функцію використовують для створення матриці відстаней, яка визначає всі відстані між містами у графі задачі комівояжера.

Як оператор відбору було вибрано турнірний відбір розміру 3, оскільки він дає змогу зберегти різноманітність у популяції та має невелику обчислювальну складність.

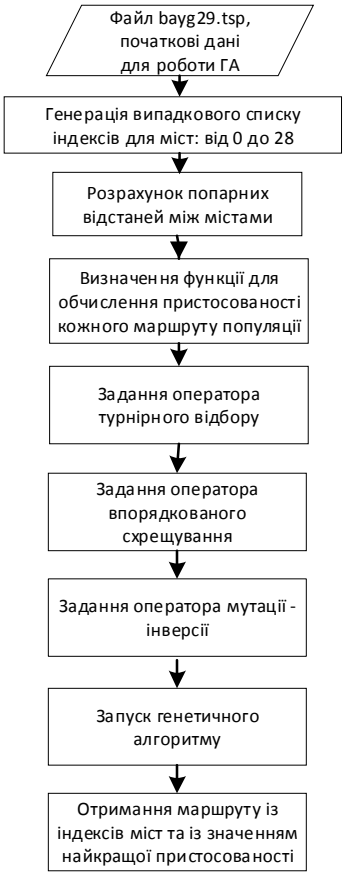

Рис. 4. Основні кроки дослідження застосування ГА для задачі комівояжера

Для схрещування вибрано впорядковане схрещування, оскільки цей оператор гарантує збереження послідовності генів, отриманих від одного з батьків, що робить його особливо корисним для задач, де послідовність генів важлива, зокрема в задачі комівояжера. Для мутації вибрано інверсійний оператор, оскільки він доволі простий та швидкий для виконання. Це робить його ефективним варіантом для використання за великих об'ємів даних або у складних задачах комівояжера. Для проведення досліджень створено програму на мові Python, яка ґрунтується на використанні фреймворку DEAP [11] та бібліотеки TSPLIB.

У табл. 2 наведено результати здійсненого моделювання із використанням оператора турнірного відбору за сталої кількості поколінь 120 та зміни розміру популяції від 100 до 250 з кроком 50 . Містам присвоюють індекси від 0 до 28.

Таблиця 2

Результати у разі зміни розміру популяції

Розмір популяції	Індекси міст для формування найкращого маршруту	Довжина найкращого маршруту, км
100	[24, 6, 15, 22, 27, 5, 11, 8, 4, 28, 2, 25, 0, 23, 7, 26, 12, 20, 1, 19, 9, 3, 14, 17, 13, 16, 21, 10, 18]	10503,1435546875
150	[14, 17, 16, 21, 13, 10, 18, 15, 24, 6, 22, 7, 26, 23, 0, 27, 5, 11, 8, 25, 2, 28, 4, 20, 1, 19, 9, 12, 3]	9271,7353515625
200	[2, 28, 1, 19, 12, 15, 23, 0, 27, 7, 26, 22, 6, 24, 18, 10, 21, 16, 13, 17, 14, 3, 9, 20, 5, 11, 8, 4, 25]	9523,2568359375
250	[7, 23, 12, 3, 10, 21, 16, 13, 17, 14, 18, 24, 6, 22, 26, 15, 9, 19, 1, 20, 4, 28, 2, 25, 8, 11, 5, 27, 0]	9660,9111328125

Рис. 5 та 6 демонструють отримані результати для найкращого маршруту, який встановлений за таких параметрів ГА: розмір популяції – 150, к-сть поколінь – 120.

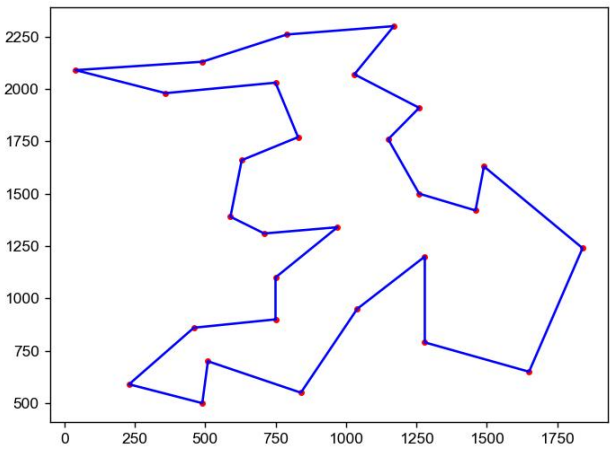

Рис. 5. Побудова оптимального маршруту комівояжера на основі відвідування 29 міст

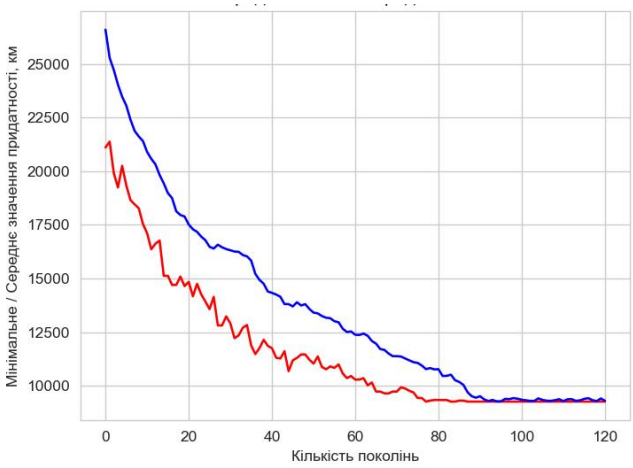

Рис. 6. Залежність значення придатності рішення (його довжини) від кількості поколінь

Найкращий результат, відповідно до табл. 2, отримано за розміру популяції 150. Отже, найкоротший маршрут комівояжера полягає у відвідуванні міст у такій послідовності [14, 17, 16, 21, 13, 10, 18, 15, 24, 6, 22, 7, 26, 23, 0, 27, 5, 11, 8, 25, 2, 28, 4, 20, 1, 19, 9, 12, 3], його довжина становить 9271.7353515625 км.

Висновки

У статті розглянуто класифікацію оптимізаційних задач. Здійснено аналіз основних структурних елементів генетичного алгоритму. Наведено особливості його використання для різних оптимізаційних задач. Продемонстровано застосування генетичного алгоритму для розв'язання задачі комівояжера та здійснено дослідження впливу розміру популяції на довжину маршруту. Показано, що найкоротший маршрут для розглянутої задачі комівояжера формується, якщо розмір популяції – 150.

Отже, у роботі продемонстровано пошук маршруту комівояжера на основі генетичного алгоритму, який ефективно оптимізує маршрути з великою кількістю міст.

Список використаних джерел

[1] *Liu, R. and Wang, Y. (2019), "Research on TSP Solution Based on Genetic Algorithm", IEEE ACIS 18th International Conference on Computer and Information Science (ICIS), Beijing, China, pp. 230–23. DOI: 10.1109/ICIS46139.2019.8940186*

[2] *Pavlenko, O., Tymoshenko, A., Tymoshenko, O., Luntovskyy, A., Pyrih, Y. and Melnyk, I. (2023), Searching Extreme Paths Based on Travelling Salesman's Problem for Wireless Emerging Networking. In: Klymash, M., Luntovskyy, A., Beshley, M., Melnyk, I., Schill, A. (eds) Emerging Networking in the Digital Transformation Age. TCSET 2022. Lecture Notes in Electrical Engineering, Vol. 965. Springer, Cham. DOI: https://doi.org/10.1007/978-3-031-24963-1_16*

[3] *Pyrih, Y., Kaidan, M., Tchaikovskyi, I. and Pleskanka, M. (2019), "Research of Genetic Algorithms for Increasing the Efficiency of Data Routing", 3rd International Conference on Advanced Information and Communications Technologies (AICT), Lviv, Ukraine, pp. 157–160. DOI: 10.1109/AIACT.2019.8847814*

[4] *Swarnakar, S., Kumar, N., Kumar, A. and Banerjee, C. (2020), "Modified Genetic Based Algorithm for Load Balancing in Cloud Computing", IEEE 1st International Conference for Convergence in Engineering (ICCE), Kolkata, India, pp. 255–259. DOI: 10.1109/ICCE50343.2020.9290563*

[5] Zitar, R. (2022), "A Review of the Genetic Algorithm and JAYA Algorithm Applications ", 15th International Congress on Image and Signal Processing, (CISP-BMEI), Beijing, China, pp. 1–7. DOI: 10.1109/CISP-BMEI56279.2022.9980332

[6] Abdallah, W. and Val, T. (2020), "Genetic-Voronoi algorithm for coverage of IoT data collection networks", 30th International Conference on Computer Theory and Applications (ICCTA), Alexandria, Egypt, pp. 16–22. DOI: https://doi.org/10.48550/arXiv.2202.13735

[7] Sharaf, A. and Pillai, M. (2019), "Genetic Algorithm Based Clustering Techniques in Wireless Sensor Networks: A Comprehensive Study", 1st International Conference on Innovations in Information and Communication Technology (ICIICT), Chennai, India, pp. 1–5. DOI: 10.1109/ICIICT1.2019.8741485

[8] Klymash, M., Kaidan, M., Strykhalyuk, B., Pyrih, Y. and Pyrih, Y. (2023), "Method for Estimating the Topological Structure of Self-Organized Networks", 17th Int. Conf.on the Experience of Designing and Application of CAD Systems (CADSM), Jaroslaw, Poland, 2023, pp. 14–17. DOI: 10.1109/CADSM58174.2023.10076498.

[9] Grant, R. (2022), TSPLIB 95 Documentation, Release 0.7.1, 52 p.

[10] Rainville, F., Fortin, F., Gardner, M., Parizeau, M., Gagné, C. (2012), "DEAP: a python framework for evolutionary algorithms", 14th annual conference companion on Genetic and Evolutionary Computation (GECCO '12). Association for Computing Machinery, New York, NY, USA, pp. 85–92. DOI: 10.1145/2330784.2330799

GENETIC ALGORITHM AS A TOOL FOR SOLVING OPTIMISATION PROBLEMS

Yaroslav Pyrih, Mykhailo Klymash, Yuliia Pyrih, Orest Lavriv

Lviv Polytechnic National University, 12, S. Bandery str., Lviv, 79013, Ukraine

The article focuses on the peculiarities of using the genetic algorithm (GA) for solving optimization problems. It provides a classification of optimization problems and offers a detailed description of the structural elements of the GA and their role in solving the traveling salesman problem. To assess the impact of GA parameters on its effectiveness, a study on the influence of population size on the length of the traveling salesman's route is conducted. Based on the obtained results, it is shown that population size affects the route length, and the optimal population size for this problem is found to be 150. Using the tournament selection operator, the ordered crossover operator, and the inverse mutation operator, we obtained a salesman's route of 9271.735 km, which, based on the results presented in this paper, is optimal for visiting 29 cities.

Key words: *genetic algorithm; traveling salesman problem; route; population.*