Висновки

Створена інтелектуальна система підтримки прийняття рішень дає змогу спростити та полегшити процес нарахування та виплат соціальних допомог управліннями праці та соціального захисту населення. Крім того, розрахунки прогнозу кількості отримувачів соціальних допомог, проведені за допомогою методу множинної регресії, показують достатньо високу кореляцію з реально отриманими результатами соціального моніторингу, що свідчить про доцільність практичного застосування запропонованого методу. Впровадження системи дозволить більш ефективно розподілити обмежені ресурси державного та місцевих бюджетів, які скеровують на адресну допомогу населенню. Слід зазначити, що лише за умови використання взаємопов'язаних способів отримання інформації про майбутнє можна інтерпретувати та правильно застосувати результати соціального прогнозування.

1. Закон України "Про державну допомогу сім'ям з дітьми" від 21 листопада 1992 року № 2811-XII. 2. Закон України "Про державну допомогу малозабезпеченим сім'ям" від 1 червня 2000 р. за № 1768-III. 3. Горелов С. Математические методы в прогнозировании. – М.: Прогресс, 1993. – 316с. 4. Катренко А. В. Системний аналіз об'єктів та процесів комп'ютеризації: Навчальний посібник. – Львів: „Новий світ-2000", 2003 – 424с. 5. Кремер Н. Ш., Путко Б. А. Эконометрика: Учебник для втузов / Под ред. проф. Н. Ш. Кремера. – М.: ЮНИТИ-ДАНА, 2002–311с.6. Литвак Б.Г. Экспертные оценки и принятие решений. – М.: Патент, 1996.7. Писарева О. М. Методы социально-экономического прогнозирования: Учебник / ГУУ – НФПК. – М., 2003. – 396 с. 8. Шевчук П.І. Соціальна політика – Львів: Світ, 2003. – 400 с.

УДК 683.1

Т.В. Шестакевич, Б.О. Коссак*
Національний університет "Львівська політехніка",
кафедра інформаційних систем та мереж,
*Львівський національний університет імені ІванаФранка,
кафедра прикладної математики

МОДИФІКОВАНИЙ ГЕНЕТИЧНИЙ АЛГОРИТМ ТА ЙОГО ПРОГРАМНА РЕАЛІЗАЦІЯ

Розглянуто основні означення та положення теорії еволюційних алгоритмів. Описано теоретичне підгрунтя модифікованого генетичного алгоритму, розглянуто його програмну реалізацію. Запропоновано тестові приклади для перевірки роботи алгоритму. Наведено застосування алгоритму для модельної задачі –оптимізації вирощування пеніциліну.

The basis of evolutionary algorithms theory is considered in this paper. A theoretical background of modified genetic algorythm is described and its program realization is considered. The test examples are proposed to check the program. The implementation of genetic algorithm is proposed – the task of penicillium growing is considered.

Постановка проблеми в загальному вигляді

Множина класичних методів для розв'язування задач певного типу нерідко виявляється незастосовною для іншої парадигми проблем та завдань. Основною причиною цього є вузька спеціалізація методу – якщо алгоритм працює, наприклад, виключно для лінійних задач. Для

застосування класичного підходу необхідно, щоб задача цілком описувалася визначеною детермінованою моделлю (деяким набором відомих функцій і параметрів). У такому випадку алгоритм дає точну відповідь. Якщо параметри моделі неправильні (для даного прикладу – не лінійні), то і результати будуть далекими від істинних. Над способом вирішення такої задачі достатньо ефективно працюють сучасні методи – аналітичні технології, побудовані на основі штучного інтелекту та наслідуванні природніх процесів. Сьогодні для пошуку оптимальних рішень все ширшого застосування набувають методи, що використовують елемент випадковості. Одним з типів таких "частково" випадкових методів є еволюційні обчислення (еволюційний пошук).

Аналіз останніх досліджень

Широкі можливості застосування аналітичних технологій ініціювали збільшення кількості наукових праць в царині еволюційних досліджень. Великий обсяг публікацій, що пропонують вичерпне теоретичне обгрунтування та опис роботи еволюційних алгоритмів, свідчить про підвищений інтерес до цієї галузі. Значна кількість інформації вимагає її узагальнення та структурування: дані необхідно аналізувати як на теоретичному, так і на практичному рівнях. Разом з тим, коди програм, що відтворюють класичний еволюційний алгоритм, з'являлись вже в кількох виданнях (для програмування, переважно, використовувалась мова Pascal), але рівень практичних розробок у галузі еволюційних обчислень ще доволі низький.

Цілі статті

Сформулювати завдання цієї статті можна так: узагальнити та структурувати знання про класичний та модифікований генетичний алгоритм, ознайомитись із реалізацією модифікованого генетичного алгоритму у вигляді компоненти для Delphi версій 4, 5, 6, 7, 8 та Kylix 2 (Delphi під Linux), що дозволяє створювати програми оптимізації для двох операційних систем: Windows та Linux; протестувати роботу алгоритму.

Варто зазначити, що базові відомості про класичний генетичний алгоритм подано як основу для порівняння з модифікованими операторами. Як класичний, так і модифікований алгоритми було реалізовано програмно, що дозволяє порівнювати результати роботи за різними підходами.

Основний матеріал

Основні означення та положення теорії генетичних алгоритмів

Еволюційні обчислення – термін, що застосовується для загального опису алгоритмів пошуку та оптимізації, заснованих на деяких формалізованих принципах природного еволюційного процесу. Основна перевага еволюційних обчислень в цій галузі полягає в можливості розвязувати багатомодальні задачі (такі, що мають кілька локальних екстремумів) подібно до того, як це відбувається в природному середовищі.

Еволюційна теорія стверджує, що кожен біологічний вид розвивається і змінюється цілеспрямовано для того, щоб найкраще пристосуватися до навколишнього середовища. В процесі еволюції деякі види комах і риб отримали захисне забарвлення, людина – складну нервову систему. Можна сказати, що еволюція – це процес оптимізації всіх живих організмів. Основним правилом при цьому є закон еволюції: "виживає сильніший", тобто більш пристосовані особини мають більше можливостей для виживання та розмноження і, відповідно, приносять більше потомства, ніж гірше пристосовані особини.

Іншим важливим фактором ефективності еволюційного пошуку є моделювання розмноження та наслідування. Варіанти рішень за певним правилом породжують нові рішення, які будуть наслідувати кращі риси своїх "батьків". Отже, нащадки сильних індивідумів також будуть відносно добре пристосовані, а їх частка в загальній масі особин буде зростати. Після зміни декількох

десятків або сотень поколінь середня пристосованість особин даного виду помітно зросте. Такий спосіб оптимізації живої природи в математиці дістав назву генетичний алгоритм.

Основна відмінність генетичних алгоритмів від інших методів оптимізації полягає в зображенні будь-якої альтернативи рішення у вигляді бітового рядка фіксованої довжини, маніпуляції з яким проводяться за відсутності усілякого зв'язку з її смисловою інтерпретацією. Тобто застосовується єдине універсальне зображення довільної задачі.

Генетичні алгоритми являють собою алгоритми пошуку, побудовані на правилах, подібних до принципів природного відбору та генетики. Якщо узагальнити, то вони об'єднують в собі принцип виживання найбільш пристосованих особин-рішень та структурований обмін інформацією, в якому присутній елемент випадковості, що моделює природні процеси наслідування та мутації.

Переваги генетичних алгоритмів стають більш очевидними, якщо розглянути їх основні відмінності від традиційних методів. Виділимо три відмінності:

1. Генетичні алгоритми працюють з кодами, в яких зображено набір параметрів, що напряму залежить від аргументів цільової функції. Причому інтерпретація цих кодів відбувається лише перед початком роботи алгоритму та після її завершення для одержання результату. В процесі роботи маніпуляція з кодами відбувається незалежно від їх інтерпретації, код розглядається просто як бітовий рядок.

2. Для пошуку генетичний алгоритм використовує кілька точок пошукового простору одночасно, а не переходить від точки до точки, як це роблять традиційними методами. Це дозволяє подолати один з недоліків – небезпеку потрапити в локальний екстремум цільової функції, якщо вона не є унімодальною. Використання кількох точок одночасно значно знижує таку можливість.

3. Генетичні алгоритми в процесі роботи не використовують жодної додаткової інформації, що збільшує швидкість роботи. Єдина інформація, що застосовується – це область допустимих значень параметрів цільової функції в точці.

Основи теорії генетичних алгоритмів (класичний генетичний алгоритм зокрема) вже достатньо відомі. Достатньо перелічити базові поняття та терміни, що використовуються в генетичних алгоритмах:

- Генотип та фенотип;
- Гени та хромосоми;
- Особина та пристосування особини;
- Популяція та розмір популяції;
- Батьки та нащадки.

До характеристик генетичного алгоритму належать:

- Розмір популяції;
- Оператор селекції;
- Оператор репродукції;
- Оператор схрещення та імовірність його використання;
- Оператор мутації та імовірність мутації;
- Критерії зупинки.

Аналіз методів та модифікація класичного генетичного алгоритму

Класичний генетичний алгоритм (описаний вище) використовує бінарне зображення особин, селекцію за допомогою кола рулетки, а також одноточкове схрещення з імовірністю $p_c = 1$. З метою підвищення ефективності роботи генетичного алгоритму з'явилось багато його варіацій. Стосуються вони інших методів селекції, модифікації генетичних операторів, особливо оператора схрещення, перетворення функції пристосування (шкалювання), а також іншого способу кодування параметрів.

Методи селекції

Метод селекції, базований на колі рулетки, є основним методом вибору особин в батьківську популяцію для подальших перетворень за допомогою генетичних операторів. Тут батьківські особини хоч і обираються імовірнісно, але пропорційно до свого пристосування. При використанні такого методу імовірність обрання тієї чи іншої хромосоми визначається її пристосуванням. Недоліком такої селекції є можливість застосування її лише до максимізації або лише до мінімізації функції і незастосовність для від'ємних функцій.

Іншою слабкою стороною рулетки є те, що особина з дуже малим пристосуванням занадто рано виводиться з популяції, що може приводити до передчасної збіжності алгоритму. Для запобігання цьому використовується певне перетворення.

Зазначені вище недоліки спонукали до створення нових методів селекції. Одним з них є *турнірна селекція* (англ. – *tournament selection*).

У цьому методі особини популяції діляться на підгрупи визначеної розмірності, а далі в кожній з них обирається особина з найкращим пристосуванням. Підгрупи можуть бути довільної розмірності, найчастіше з популяції виділяються множини з 2-х або 3-х хромосом. Турнірна селекція дозволяє працювати з максимізацією та мінімізацією від'ємних функцій.

Інші модифікації оператора селекції є комбінацією вже описаних. Наприклад, утворювати підгрупи в турнірній селекції можна за допомогою рулетки. В *детерміністичній* селекції кожна хромосома гарантовано передасть в батьківську популяцію стільки кодів, скільки становить ціла частина від такого виразу:

$$e(A_i) = \frac{fit(A_i)}{fit_s}, \qquad (1)$$

де $fit(A_i)$ – пристосування *i*-ї хромосоми; fit_s – середнє пристосування усієї популяції. Незаповнені місця в батьківській популяції займають кращі хромосоми попередньої популяції, впорядковані за залишками $e(A_i)$.

Методи репродукції

Найпростіший підхід, використаний в класичному генетичному алгоритмі – випадковий вибір батьківської пари, коли обидві особини обираються з усієї батьківської популяції, і довільна особина може стати членом кількох пар. Такий підхід є універсальним для розв'язування багатьох класів задач.

Однією з особливих стратегій вважається *елітарна (елітна)*. Класична репродукція не завжди забезпечує перехід найкращої хромосоми в наступну популяцію. Елітарна стратегія дозволяє користувачу визначити відсоток найкращих особин батьківської популяції, які без змін, не потрапляючи під дію генетичних операторів, перейдуть в результуючу множину хромосом. Вакантні місця в результуючій популяції заповнюються особинами, що утворені звичайно з батьквського набору хромосом (включно з "елітними").

Треба зауважити, що в більшості джерел елітарна стратегія репродукції описана так, як відтворено вище. Однак елітарною називається відмінна стратегія. Назвімо її *природною елітарною* репродукцією.

Природна елітарна стратегія полягає у тому, що, застосувавши довільні оператори селекції, репродукції та схрещення, в проміжну популяцію вносяться не лише нащадки, але і батьки. Популяція сортується за зростанням, і в результуючу множину особин вноситься "еліта" розширеної популяції – так, щоб її об'єм дорівнював розмірам вихідної популяції. Такий спосіб є природним, бо в живій природі батьки, давши потомство, переважно не гинуть, а існують водночас з нащадками.

Інший спосіб відбору особин в батьківську пару – *селективний.* Він полягає в тому, що батьками можуть стати лише особини, рівень пристосування яких вищий за середній рівень пристосування особин усієї популяції. Такий підхід забезпечує більш швидку збіжність алгоритму.

Але селективна репродукція незастосовна тоді, коли сформульовано задачу пошуку кількох екстремумів, бо для таких задач алгоритм швидко прямуватиме до одного розв'язку.

Схрещення

Поряд з одноточковим схрещенням існують ще *двоточковий* та *багатоточковий* кроссовер, а також *рівномірне* схрещення.

Батько1	00000000	000~0000~0	?	000~1111~0	00011110 Нащадок1
Батько2	11111111	111~1111~1	?	111~0000~1	11100001 Нащадок2

Очевидно, що багатоточкове схрещення в часткових випадках є двоточковим та одноточковим. Останній спосіб вже описано, тож розглянемо двоточковий кроссовер. При його застосуванні випадково обираються вже дві точки розриву, за якими хромосоми обмінюються внутрішніми частинами, як зображено на схемі.

У процесі багатоточкового схрещення хромосоми діляться на відповідну кількість підланцюжків і обмінюються кожною парною частиною (якщо починати рахувати їх з першої).

Рівномірне схрещення відбувається згідно з певним взірцем, що вказує, які гени наслідуються від першого, а які від другого з батьків. Припустимо, що ймовірнісно обраний взірець буде таким: 0111011, де 1 означає, що нащадок на відповідному місці отримає ген від першого батька, а 0 – від другого батька. Так утворюється перший нащадок. Другий нащадок утворюється за подібним правилом, але 1 означає, що нащадок отримає ген від другого батька, а 0 – від першого. Схематично такий спосіб схрещення відтворено нижче.

Ном.	1	2	3	4	5	6	7
Взірець	0	1	1	1	0	1	1

Батько1	0111010	?	1111010	Нащадок1
Батько2	1011011	?	0011011	Нащадок2

Нововведень в процесі *мутації* небагато, наприклад, можна розрізняти мутацію хромосом до застосування схрещення або вже після нього в популяції нащадків.

Інверсія є одним з підвидів мутації, коли з певною ймовірністю обирається блок генів та інвертується їх порядок. Нижче наведено приклад для блоку розміром у 3 біти і починається він з другого біту.

00011111 ? 0**001**1111 ? 01001111

Методи кодування

У класичному генетичному алгоритмі застосовується бінарне кодування. Цей найпростіший спосіб, однак, має свої недоліки. Один з них полягає в тому, що сусідні числа відрізняються в значеннях кількох бітів, наприклад, числа 7 та 8 у бітовому поданні відрізняються в 4-х позиціях. Це вповільнює роботу генетичного алгоритму та збільшує час збіжності. *Кодування Грея* вирішує цю проблему, тут

кожне наступне число відрізняється від попереднього на одну позицію. З іншого боку, про безумовну доцільність застосування бінарного кодування чи кодів Грея ще точаться суперечки.

Іншою проблемою для бінарного кодування може бути випадок, коли необхідно проводити обчислення з високою точністю, коли обмеження на змінні оптимізації є достатньо широкими або якщо простір пошуку великий і оптимізації підлягає значна кількість змінних. Тоді бінарний ланцюжок буде дуже довгим, робота з ним ускладниться. Для відвернення цієї проблеми варто застосовувати *логарифмічне кодування.*

Застосування такого методу кодування передбачає вичленування першого біту (α) двійкового ланцюжка як знаку степеня степеневої функції, другий біт (β) – як знак цієї степеневої функкції; інші біти ($[bin]$) – степінь згаданої функції:

$$[\alpha\beta \text{bin}] = (-1)^{\beta} e^{(-1)^{\alpha}[\text{bin}]_{10}}, \tag{2}$$

де $[bin]_{10}$ – десяткове значення бінарного ланцю4жка *bin*. Наприклад, знайдемо десяткове значення деякого двійкового коду.

$$[100101] \rightarrow x_1 = (-1)^0 e^{(-1)^1[0101]} = e^{-5} = 0{,}00696917$$

Тобто, ланцюжком у шість бітів можна закодувати значення з проміжку $\left[-e^{15}, e^{15}\right]$, тоді як використовуючи бінарне кодування – значення з проміжку $[0{,}63]$.

Для максимізації функції $f(x_1, x_2, ..., x_n)$ для $x_i \in [a_i, b_i] \subset R; i = 1,2,...,n$, з точністю розв'язку до s знаків після коми для кожної змінної x_i застосовуємо подібний принцип, як і при класичному кодуванні. Найменше натуральне m_i, яке задовольняє нерівність

$$(b_i - a_i) \cdot 10^s \le e^{2^{m_i} - 1} \tag{3}$$

визначає довжину бінарного ланцюжка, необхідного до закодування значення з проміжку $[a_i, b_i]$ з точністю s.

Розкодування значення в такому випадку відбувається за такою формулою:

$$x_i = \frac{b_i - a_i}{2 \cdot e^z} \cdot x_i^{\log} + \frac{b_i + a_i}{2}, \tag{4}$$

де $x_i^{\log}$ – значення, отримане за формулою (11), $z = [bin]_{10}$.

Шкалювання функції пристосування

Для перетворення функції пристосування є дві причини. По-перше, щоб запобігти передчасній збіжності алгоритму. По-друге, в кінцевій фазі роботи алгоритму популяція може стати дуже різнорідною, причому середнє значення несильно відрізнятиметься від максимального. Шкалювання може виправити ситуацію, коли особини найкращої та середньої якості під час селекції отримують майже таку саму кількість копій. Передчасна збіжність алгоритму полягає в тому, що доволі якісні, але не оптимальні хромосоми домінують в популяції (особливо якщо застосовувати метод рулетки). Шкалювання функції пристосування оберігає популяцію від домінування неоптимальної особини, тобто від передчасної збіжності.

Лінійне шкалювання перетворює функцію пристосування F в функцію F' таким лінійним законом:

$$F' = a \cdot F + b, \tag{5}$$

де a та b – сталі, які забезпечують такі вимоги, накладені на новоутворену функцію:

- середнє значення функції пристосування після шкалювання не змінюється;
- максимальне значення функції пристосування після шкалювання стає кратним значенню функції пристосування до перетворення.

Коефіцієнт кратності обирається від 1.2 до 2 та накладаються умови на додатність функції F'.

Відтинання типу sigma перетворює функцію пристосування F в функцію F' таким законом:

$$F' = F + \left(\overline{F} - c \cdot \sigma\right), \tag{6}$$

де $\overline{F}$ – середня вартість функції пристосування популяції; c – ціле число від 1 до 5; σ – стандартне відхилення популяції.

Якщо F' стає від'ємною, вважають , що вона дорівнює нулеві.

Степеневим шкалюванням проводять перетворення так:

$$F' = F^k, \tag{7}$$

k обирають близьким до одиниці, припустимо k=1,004.

Узагальнимо дані про оператори класичного та модифікованого генетичних алгоритмів за допомогою табл. 1, 2.

Таблиця 1

Класичні та модифіковані оператори генетичного алгоритму

Назва оператора	**Класичний ГА**	**Модифікації ГА**		
Кодування	Бінарне	Грея	Логарифмічне	
Селекція	Рулетка	Турнірна	Детерміністична	
Репродукція	З імовірністю=1	Елітарна	Природна елітарна	
Схрещення	Одноточкове	Двоточкове	Багатоточкове	Рівномірне
Мутація	Із заданою імов.	Інверсія	До або після схрещення	
Шкалювання	Відсутнє	Лінійне	Відтинання типу sigma	Степеневе

Наступна таблиця відображає ті види операторів, що були реалізовані програмно.

Таблиця 2

Перелік реалізованих операторів генетичного алгоритму

Назва оператора	**Класичний ГА**	**Модифікації ГА**	
Кодування	Бінарне	Логарифмічне	
Селекція	Рулетка	Турнірна	
Репродукція	З імовірністю=1	Елітарна	
Схрещення	Одноточкове	Двоточкове	Рівномірне
Шкалювання	Відсутнє	Лінійне	

Компонента geneticalgoritm

Після інсталяції створеної компоненти в мовне середовище її можна знайти на закладці ExComponents набору компонент Delphi. Вигляд її є такий (рис. 1).

У компоненті описано три класи: *TgeneticAlgoritm* – Основний клас, що реалізує базові функції генетичного алгоритму, *Tchromosomes* – клас, що працює з набором хромосом, *Tchromosome* – клас, що реалізує роботу з однією хромосомою.

Рис. 1. Вигляд компоненти в палітрі компонент

Властивості класу tgeneticalgoritm

Property CountChromosomes : integer	Значення кількості хромосом, що будуть міститися в популяції.
Property CountOfEpochs : Integer	Значення кількості епох (кроків) генетичного алгоритму.
Property Mutation_P : Double	Імовірність мутації хромосом, є в межах [0,1].
Property TypeCoding : TtypeCoding, де *TTypeCoding = (Normal, Exponential);*	У цій властивості вказується, який метод кодування буде використовуватися при переході з десяткової системи числення в двійкову.
Property TypeCrossover : TtypeCrossover, де *TtypeCrossover = (OnePointCrossover, TwoPointCrossover, LogicCrossover)*	Властивість, що задає тип схрещування для хромосом. Є три можливі типи схрещування: одноточкове схрещення (*OnePointCrossover*), двоточкове схрещення (*TwoPointCrossover*) та рівномірне схрещення (*LogicCrossover*).
property TypeOptimization : TmypeOptimization TTypeOptimization = (Maximum, Minimum);	Властивість, що визначає тип оптимізації. Якщо потрібно максимізувати цільову функцію потрібно задати *Maximum*, якщо мінімізувати – *Minimum.*
property TypeSelection : TtypeSelection TTypeSelection = (Ruletka, Turnir_Selection);	Задає тип селекції при виборі проміжної популяції для схрещення. Реалізовано два види селекції: Рулетка (*Ruletka*), що вибирає проміжну популяцію подібно до розміченого кола рулетки, та турнірний відбір (*Turnir_Selection*), що утворює проміжну популяцію за описаним раніше правилом.
AllParams : Array of Tparams; де *TParams = record a : Double; b : Double; s : Integer; end;*	Для роботи генетичного алгоритму потрібно задати кількість змінних цільової функції та обмеження для кожної змінної. Кількість змінних є розмірністю масиву *AllParams*. Кожний елемент даного масиву містить інформацію про змінну, а саме: проміжок *[a,b]* та точність кодування цієї змінної *s*.

Наступні три події виконуються на початку схрещення, під час роботи схрещення та на завершення схрещення відповідно на кожному кроці генетичного алгоритму.

Type

TOnExecuteBegin = Procedure (Sender : TObject; MaxCount : Integer);
TOnExecuteWork = Procedure (Sender : TObject; NowIndex : Integer; NowChromosomes : TChromosomes);
TOnExecuteEnd = Procedure (Sender : TObject);

Наступні три події виконуються під час роботи генетичного алгоритму. Подія *OnExecuteBegin* виконується на старті генетичного алгоритму та повертає значення кількості епох. *OnExecuteWork* виконуються після завершення виконання кроку генетичного алгоритму, тобто тоді, коли утвориться нова популяція. Ця подія повертає номер кроку генетичного алгоритму та набір хромосом, що були утворені генетичними операторами. *OnExecuteEnd* виконується на завершення генетичного алгоритму.

property OnExecuteBegin : TOnExecuteBegin;
property OnExecuteWork : TOnExecuteWork;
property OnExecuteEnd : TOnExecuteEnd;

Методи класу tgeneticalgoritm

Procedure Init;	Ініціалізує початкові значення генетичного алгоритму. Цю процедуру потрібно виконати після задання кількості параметрів та їх значень.
Procedure GenerateFirstPopulation;	Ця процедура генерує початкову популяцію. Її потрібно виконати на початку роботи генетичного алгоритму.
Function GetFitness(NuberPopulation, NumberChromosome : Integer) : Double;	Повертає значення пристосування (результат цільової функції) для заданої хромосоми з певної популяції.
Function GetFitnessPopulation (NumberPopulation : Integer) : Double;	Повертає пристосування загальної популяції, тобто суму пристосувань усіх хромосом, що знаходяться у даній популяції.
Procedure Execute;	Основна процедура генетичного алгоритму, що розпочинає його роботу.

Тестові приклади

Для демонстрації роботи генетичного алгоритму використаємо створену програму.

Пошук глобального мінімуму для багатоекстремальної функції

Знайти мінімум функції

$$f(x) = x^2 - 10(\cos(2\pi x)) \tag{8}$$

на проміжку [-5.12, 5] при s=3. Глобальний мінімум знаходиться в точці 0, значення функції для нього 0.

Графік функції наведено на рис. 2.

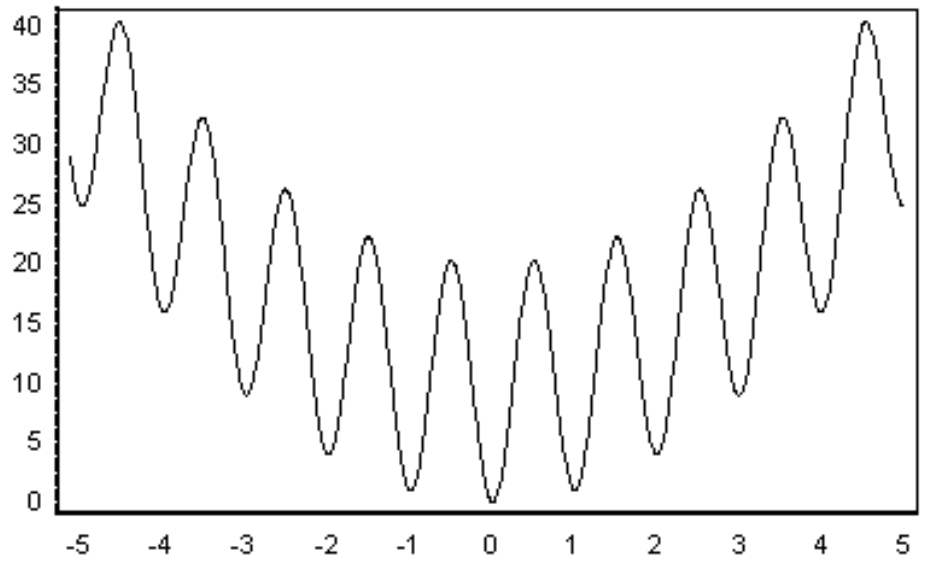

Рис. 2. Графік функції $f(x) = x^2 - 10(\cos(2\pi x))$

Для роботи задаються такі параметри:

Кількість хромосом	20
Кількість епох	20
Імовірність схрещення	0.8
Імовірність мутації	0.1
Селекція	Турнірна
Схрещення	Рівномірне
Кодування	Класичне

Результати роботи подаватимуться у вигляді усереднених даних після 50 запусків генетичного алгоритму.

При заданих параметрах одержимо, що алгоритм в кількох випадках потрапив у локальні мінімуми (рис. 3).

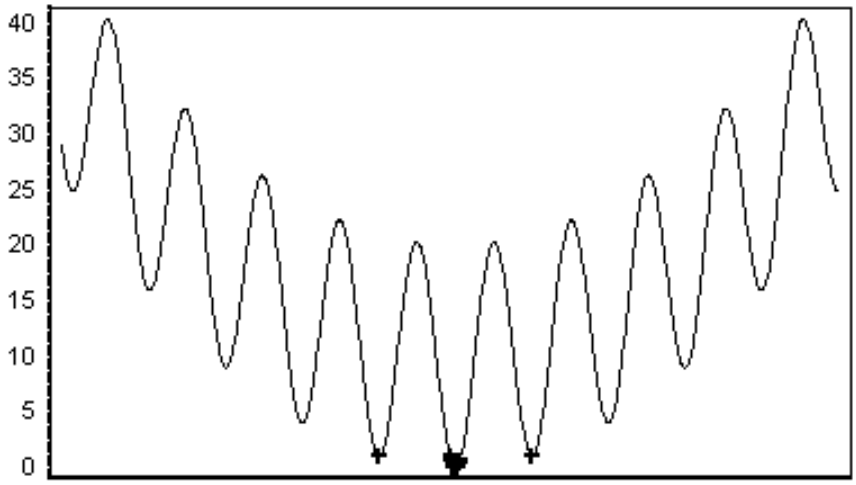

Рис. 3. Локальний мінімум для функції $f(x)=x^2-10(\cos(2\pi x))$

Експерименти показали, що для уникнення цієї проблеми можна використовувати, зокрема, два шляхи: збільшити кількість хромосом в популяції або замість класичного кодування використати логарифмічне. Обидва експерименти описано нижче. Зауважимо, що при використанні логарифмічного кодування для даного прикладу точка мінімуму [0,0] знаходилася дуже швидко (в середньому 3,74 ітерації) і дуже точно – в 98% випадків.

Для збільшеної кількості хромосом (60 хромосом) алгоритм вже не потрапляє в локальні мінімуми, а розв'язки скупчуються біля точки мінімуму, як зображено на рис. 6. Глобальний мінімум було досягнуто в 58% випадків (значення з проміжку [-0.01, 0.01] вважалось глобальним мінімумом).

Цей приклад підтверджує правильність роботи алгоритму. Проблеми, що виникають при певних налаштуваннях класичного алгоритму, можуть бути вирішені застосуванням модифікованих операторів.Розглянемо процес, описаний задачею Коші для системи звичайних диференціальних рівнянь

$$\begin{cases} \dfrac{dx}{dt} = f\left(t,u,x(t,u)\right), \quad t \in \Omega = (t_0, t_l] \\ x(t_0,u) = x_0, \end{cases} \tag{9}$$

де $u=\left[u_1(t),\ ...,\ u_k(t)\right]^T$, $x=\left[x_1(t,u),\ ...,\ x_r(t,u)\right]^T$, $x_0=\left[x_{01},\ ...,\ x_{0r}\right]^T$, $f=\left[f_1,\ ...,\ f_r\right]^T$, t – час; u – функція керування, $\boldsymbol{x_i}\left(\boldsymbol{t},\boldsymbol{u}\right)$ – фазові змінні.

Тоді, враховуючи позначення, запишемо загальний вигляд задачі оптимального керування:

$$\begin{aligned} &F(x,u)\to\max,\\ &x_i\big|_{i=0}=v_i,\ \ i=\overline{1,r}\\ &u_i^-\le u_i\le u_i^+,\ \ v_j^-\le v_j\le v_j^+. \end{aligned} \tag{10}$$

Тут $F(x,u)$ – цільова функція; $x_i\left(t,u\right)$ – фазові змінні; u_i,v_j – керуючі змінні; u_i^-, u_i^+, v_i^-, v_i^+ – обмеження на керуючі змінні.

Як модельний приклад ми розглядали задачу утворення пеніциліну. Цей процес описується задачею Коші з відповідними початковими умовами.

Пеніцилін – продукт життєдіяльності грибка Penicillium. Якщо розглядати обмежену кількість мікроорганізмів, то забезпечити максимальний вихід продукту можна лише тоді, коли забезпечено найбільш сприятливі умови. Проблему оптимізації таких умов можна розв'язувати різними шляхами. Один з них є такий: під час процесу не вимірюють і не регулюють жодного параметра, крім температури. Тоді керування процесом зводиться до знаходження оптимального початкового складу середовища і найкращого температурного режиму.

У багатьох випадках постійність умов і складу середовища не забезпечують максимального виходу продукту. Так, швидкий ріст клітин не сприяє утворенню другорядного продукту, а саме він і є важливим у нашій задачі. Водночас на початковій стадії періодичного процесу низька швидкість росту клітини небажана.

При промисловому виробництві пеніциліну відомо, що підвищення температури (30 °C) сприяє прискореному росту гриба, тоді як більш низька температура (20 °C) сприяє прискореному синтезу пеніциліну. Раніше промислово пеніцилін отримували при температурі, яка лежить між цими двома значеннями (від 24 до 25 °C). Але програмована зміна температури під час процесу дозволяє суттєво підвищити вихід пеніциліну порівняно з процесом, що проходить при будь-якій сталій температурі.

Для опису кінетики росту клітин та продукту метаболізму створено відповідну математичну модель, що являє собою задачу Коші для системи диференціальних рівнянь з початковими умовами. Враховуючи співвідношення (3), для керування температурою записуються такі співвідношення (тут r=2, k=1):

$$\begin{cases} \dfrac{dx_1}{dt}=b_1(\Theta)x_1\left(1-\dfrac{x_1}{b_2(\Theta)}\right),\\ \dfrac{dx_2}{dt}=b_3(\Theta)x_1-b_4(\Theta)x_2+b_1(\Theta)b_5(\Theta)x_1\left(1-\dfrac{x_1}{b_2(\Theta)}\right), \end{cases}$$
$$x_1(0)=0.0294,\quad x_2(0)=0, \tag{11}$$

де x_1, x_2 – безрозмірні концентрації біомаси та пеніциліну відповідно. Коефіцієнти b_1-b_5 залежать від змінної температури Θ. Останній доданок другого рівняння відображає процес деградації пеніциліну.

Отже, задача оптимального керування процесом утворення пеніциліну може бути сформульована так: максимізувати вихід пеніциліну в кінцевий момент часу при обмеженнях, накладених на температуру:

$$F\left(x_1(t,\Theta),\ x_2(t,\Theta),\ \Theta\right)=x_2\left(t_1,\Theta\right)\to\max, \tag{12}$$

$$
\begin{aligned}
&20 \le \Theta \le 30;\\
&b_i(\Theta) = b_{i0}\, g(\Theta)\ i = 1,2;\\
&g(\Theta) = 1{,}143\left[1 - 0{,}005\,(30 - \Theta)^2\right];\\
&b_3(\Theta) = 1{,}143\, b_{30}\left[1 - 0{,}005\,(\Theta - 20)^2\right];\\
&b_4(\Theta) = b_{40}\exp\left[-6145\left(\frac{1}{273{,}1 + \Theta} - \frac{1}{298}\right)\right];
\end{aligned}
\qquad (13)
$$

тут

$$
b_5 = 0,\ b_{10} = 13{,}099,\ b_{20} = 0{,}9426,
$$
$$
b_{30} = 4{,}6598,\ b_{40} = 4{,}4555.
$$

Для максимізації виходу пеніциліну розв'язувалась задача оптимального керування з температурою як функцією керування. Як було сказано вище, при розв'язуванні цієї задачі з керованою температурою вихід пеніциліну підвищується порівняно з результатами, отриманими при сталій температурі. Підтвердимо це за допомогою генетичного алгоритму. Зазначимо, що при ізотермічному процесі за одиничний проміжок часу утворюється 0.9352 одиниці пеніциліну.

Для роботи алгоритму задаються такі параметри:

Кількість хромосом	40
Кількість епох	20
Імовірність схрещення	0.7
Імовірність мутації	0.75
Точність s	7
Селекція	Турнірна
Схрещення	Рівномірне
Кодування	Класичне
Проміжок	[0,1]
Початкові умови	$x_1(0) = 0.0294,$ $x_2(0) = 0$

Обгрунтуємо таке налаштування параметрів.

Зазначена кількість хромосом відповідає середнім розмірам популяції. Двадцять епох дають прийнятний результат, і збільшення їх кількості відчутно результат не покращує, а лише вповільнює роботу програми. Окремо треба сказати про величину імовірності мутації. У більшості робіт для класичного генетичного алгоритму величину цього параметру визначають як достатньо малу, приблизно 1–2.5 % бітів усієї популяції підлягають мутації. У цій задачі ми зіткнулися з тим, що покращити результат можна з допомогою підвищення рівня мутації. У деяких публікаціях для перевірки роботи алгоритму на тестових прикладах вважають, що імовірність мутації дорівнює 0.95; висловлено думку, що підвищувати мутацію варто лише наприкінці роботи алгоритму. Для цієї задачі це теж має сенс: підвищення значення цього параметра дозволяє урізноманітнити популяцію.

Для усіх вищенаведених прикладів селекція за допомогою кола рулетки давала гірший результат порівняно з турнірною селекцією. Остання обрана ще й з міркувань економії машинного часу, який значно зменшився при використанні турнірної селекції.

Оскільки популяція є середнього розміру, не було особливої різниці у застосуванні того чи іншого схрещення, тому обрано було рівномірне схрещення. Класичне кодування давало кращий результат, ніж логарифмічне. Це пояснюється тим, що функція керування лише одна.

Порівняємо розв'язки на кількох етапах роботи, коли проміжок подрібнювався на 2 та 8 відрізків. Температура на кожному з них змінювалась за лінійним законом.

Ріст пеніциліну відбувався так (рис. 4).

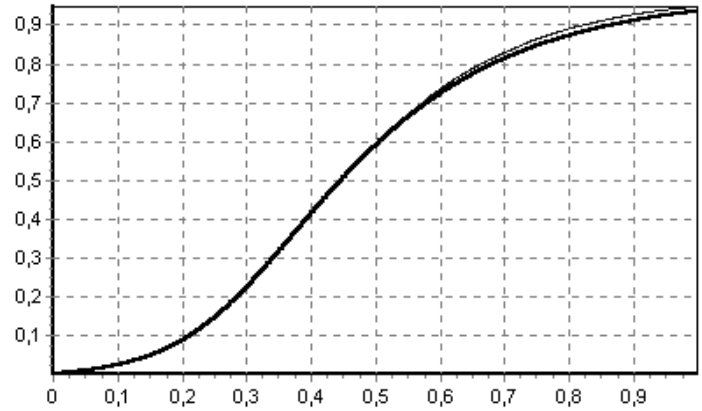

Рис. 4. Графік росту пеніциліну при розбитті проміжку на 2 відрізки

На рис. 4 тонша лінія відображає ріст пеніциліну при керованій температурі, грубша лінія відтворює ріст пеніциліну при незмінній температурі 25 градусів. З рис. 5 видно, що покращання росту відбувається, але воно доволі незначне.

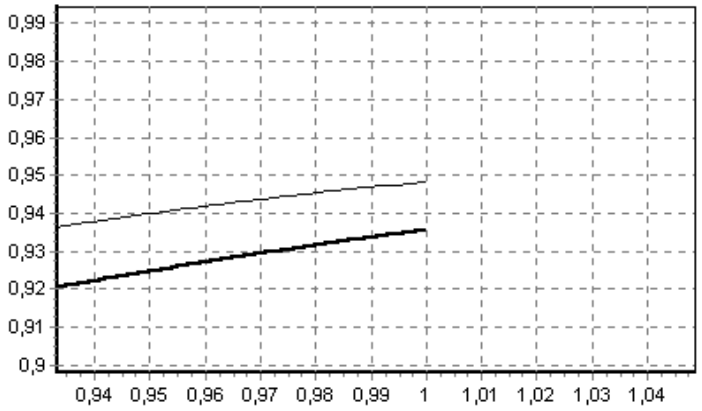

Рис. 5. Вихідний об'єм пеніциліну при розбитті проміжку на 2 відрізки

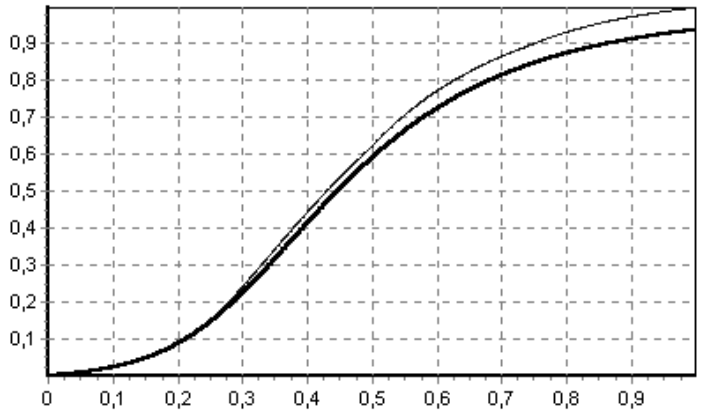

Рис. 6. Графік росту пеніциліну при розбитті проміжку на 8 відрізків

Числові результати такі: максимальне значення розв’язку – 0,94751, середнє – 0,92652. Середнє значення температури для проміжку – 24,316 та 22,771 відповідно.

Розглянемо результати при подрібненні проміжку на вісім відрізків (рис. 6). Знову передбачається зростання виходу пеніциліну в останній момент часу та спадний кусковий графік температури.

Числові результати такі: максимальне значення розв’язку – 1,0017Ї середнє – 1,0016.

№ пром.	Сер. Знач. темп.
1	24,5361
2	23,8662
3	20,0606
4	22,9697
5	21,3591
6	20,8554
7	20,4885
8	20,2649

Числові результати свідчать про додатковий приріст продукту порівняно з результатами, отриманими при попередньому розбитті. Звідси випливає кілька висновків:

По-перше, дійсно, для покращання розв'язку температура зменшується. В даному випадку для зазначених налаштувань параметрів генетичного алгоритму вона не сягає значень, близьких до верхньої межі проміжку.

По-друге, відчутне збільшення розв'язку надалі не передбачається, тому проводити надалі експерименти з подрібненням проміжку не потрібно.

По-третє, генетичний алгоритм виконав своє завдання і збільшив вихід пеніциліну порівняно з некерованим процесом. Логічно, що чим дрібніше задавалося розбиття часового відрізка, тим більший вихід продукту фіксувався.

Висновки

Генетичні алгоритми виникли завдяки спостереженням та спробам наслідувати природні процеси, що відбуваються у світі живих організмів. Хоча основні ідеї еволюції і було структуровано, та між живою природою і комп'ютерним генетичним алгоритмом існує одна характерна різниця, яка стосується часу. Більшість еволюційних процесів відбуваються протягом навіть мільярдів років, а на обчислювальних машинах аналогічний процес відбувається протягом мілісекунд.

На основі теоретичних знань про модифікований генетичний алгоритм було створено програму, що симулює еволюційні процеси. Її ефективність було протестовано на ряді прикладів різної складності, які враховували особливості роботи алгоритму. Наприклад, розглядались від'ємні функції, функції з багатьма локальними оптимумами, багатопараметричні функції. Результати, отримані при розв'язуванні тестових задач, підтвердили правильність реалізації алгоритму та ефективність його роботи.

Перспективою для роботи є, очевидно, реалізація додаткових можливостей генетичного алгоритму. Різноманітність способів відтворення тих чи інших генетичних операторів спонукає до порівняння ефективності їх роботи. Не всі прийоми, описані вище, було реалізовано програмно, адже постійно з'являються все нові та нові способи опрацювання генетичного матеріалу.

Специфіка роботи алгоритму дозволяє охопити значне коло задач: з їх допомогою можна програмувати комп'ютери, навчати штучні нейронні мережі, оптимізувати різноманітні процеси, знаходити екстемуми функцій. Методи еволюційних обчислень виявились ефективними для роз-

в'язування ряду реальних задач інженерного проектування, планування, маршрутизації та розміщення, прогнозування та в багатьох інших галузях. Тобто генетичні алгоритми є однією з важливих парадигм алгоритмів пошуку оптимальних рішень.

1. Goldberg David E. Algorytmy genetyczne i ich zastosowania. – Warsz.:WNT, 1998. 2. Rutkowska D., Pilinski M., Rutkowski L. Sieci neuronove, algorytmy genetyczne i systemy rozmyte. – Łodż: Wyd. Naukowe PWN Warszawa, 1999. 3. Генетичні алгоритми в екстремальних задачах // Вісн. Львів. ун-ту. Серія прикл. матем. та інформат. – 2000. – Вип 2.– С. 197–204. 4.Уоссермен Ф. Нейрокомпьютерная техника: Теория и практика. – Перевод Ю.А. Зуева, В.А. Точенова, 1992. 5.Батищев Д.И., Исаев С.А. Оптимизация многоэкстремальных функций с помощью генетического алгоритма. – http://www.chat.ru/~saisa. 6. Корнеев В.В., Гареев А.Ф., Васютин С.В., Райх В.В. Базы данных. Интеллектуальная обработка информации. – М.: Нолидж, 2000. 7.Савула Я.Г., Коссак Б.О., Шестакевич Т.В. Застосування генетичних алгоритмів для управління розв'язками задачі Коші // Сучасні проблеми прикладної математики та інформатики: Тези доп. Дев'ятої Всеукраїнськ. наук. конф. (24–26 вересня 2002р., м.Львів). – Львів, 2002. – С. 114.