18. Dem'ianenko V.V. Modeliuvannia reklamnykh kampanij zasobamy sit'ovoho planuvannia ta upravlinnia / V.V. Dem'ianenko, S.D. Potapenko // Marketynh v Ukraini. — 2003. — № 1. — S. 34–38.
19. Dem'ianenko V.V. Optymizatsiia rozpodilu biudzhetu reklamnoi kampanii dlia poshyrennia reklamnykh povidomlen' / V.V. Dem'ianenko, SD. Potapenko // Marketynh v Ukraini. — 2003. — № 3. — S. 10–12.

Статтю подано до редакції 18.10.2018 р.

УДК 519.615.2

Бондаренко М.В., аспірант
кафедри економіко-математичного моделювання
ДВНЗ «Київський національний економічний
університет ім. Вадима Гетьмана»
Бондаренко В.М., к.ф.-м.н., доцент,
кафедра конструювання електронно-обчислювальної апаратури,
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Bondarenko Maksym, postgraduate
Department of mathematical modeling in economics,
SHEE «Kyiv National Economic University
named after Vadym Hetman»
Bondarenko Victor, Ph.D, Associate Professor,
Department of Design of Electronic Computational Equipment,
National Technical University of Ukraine
«Igor Sikorsky Kyiv Polytechnic Institute»

КЛАСИЧНИЙ ТА БІНАРНИЙ ГЕНЕТИЧНІ АЛГОРИТМИ ДЛЯ ЗНАХОДЖЕННЯ ГЛОБАЛЬНОГО ОПТИМУМУ В ЗАДАЧАХ НЕВИПУКЛОЇ ОПТИМІЗАЦІЇ

CLASSIC AND BINARY GENETIC ALGORITHMS FOR SEARCHING GLOBAL OPTIMUM IN PROBLEMS OF NONCONVEX OPTIMIZATION

Анотація. Розглянуто класичний і бінарний підходи до імплементації генетичного алгоритму оптимізації. Генетичний алгоритм оптимізації дозволяє знаходити глобальний мінімум як випуклих функцій (таких, що мають лише один мінімум), так і функцій, що мають кілька екстремумів. Це дозволяє застосовувати його в погано поставлених задачах, де існує нескінченна множина розв`язків. Окрім того алгоритм не вимагає диференційованості функції в усіх точках і може використовуватись як ефективна альтернатива градієнтному спуску у вирішенні задач, де необхідний розрахунок багатовимірних градієнтів. Проведено експериментальне дослідження з порівняння ефективності класичного та бінарного підходів. Експеримент застосовано до двох фукнцій: квад-

ратичної функції (випукла функція) та функції Растригіна (приклад ні випуклої, ні впуклої функції). До параметрів цих функцій застосовано генетичні оператори селекції (відбору), мутації та схрещення. Після цього для результуючих параметрів розраховано функцію прийняття, яка перевіряє, чи параметр досі задовольняє умовам задачі. Цільова функція визначає, чи оптимальне рішення знайдено. Для оцінки ефективності двох підходів використано дві міри: середня кількість ітерацій до збіжності та час збіжності (у мілісекундах). Окрім цього ми варіювали коефіцієнт мутації аби знайти таке його значення, за якого дві дані міри мінімізуються. Виявлено, що за зменшення парматера мутації збіжність класичного алгоритму покращується, проте після досягнення ним певного рівня є ризик того, що алгоритм почне розходитись. При цьому не було виявлено явної залежності між збіжністю бінарного алгоритму та параметром мутації, що робить його універсальнішим у використанні. Зроблено висновок, що класичний підхід у рамках даної програмної реалізації є ефективнішим за часом виконання та має вищу точність, хоча і є складнішим в реалізації.
Ключові слова: генетичний алгоритм, оптимізація, стохастична оптимізація, мутація, схрещування, відбір.

Anotation. Classic and binary approaches to implementation of genetic algorithm of optimization are proposed. Genetic algorithm allows to find a minima of both convex functions (such that have only one minima) and functions that have multiple minimums. Thus it can be applied to ill-posed problems where there is an infinite number of solutions. Moreover the algorithm does not require differentiability of the optimized function and can be used as efficient alternative to gradient descent in problems that require calculation of multi-dimensional gradients. We conducted experimental research in comparison of efficiency of classic and binary approaches. Experiment is applied to two functions: quadratic function (convex) and Rastrigin function (as an example of neither convex nor concave function). Then genetic operators of selection, mutation and crossover are applied to function's parameters. Resulting parameters are then used to calculate the acceptance function that checks if the parameters still satisfy problem's conditions. Cost function is used to check if the optimal solution is found. To compare the efficiency of both approaches we used two measures: average number of iterations to convergence and time of convergence (in miliseconds). We also varied the mutation coefficient to find such its value that minimizes value of both measures. It turned out that decreasing mutation parameter improves convergence of classic algorithm but after reaching certain level it can cause divergence. No correlation between convergence of binary algorithm and mutation parameter was found so binary algorithm turns out to be more generic in use. Concluded that classic approach within current program implementation is more efficient in terms of execution time and has bigger precision despite of being more complex.
Keywords: genetic algorithm, optimization, stochastic optimization, mutation, crossover, selection.

Сьогодні генетичний алгоритм оптимізації, запропонований Джоном Холандом [1], властивості збіжності якого були доведені Рафаелем Серфом [2] і Дель Морелом [3], є одним з найефективніших стохастичних алгоритмів, застосованих до вирішення задач невипуклої оптимізації. Проте, цей класичний алгоритм є досить складним і на практиці часто використовуються більш «натуральні» імплементації, які зокрема характеризуються інтуї-

тивністю поняття схрещування. Підхід до однієї з таких імплементацій був представлений Франком Дієтерлом [4] та іншими вченими, проте така «бінарна» імплементація поки не була широко досліджена та формалізована.

Метою цієї роботи є експериментальне дослідження ефективності двох підходів до імплементації генетичного алгоритму на відомих випуклих і невипуклих функціях.

Генетичний алгоритм має ряд переваг перед градієнтними методами:

- не вимагає ані випуклості ані диференційованості цільової функції;
- не потрапляє в пастку «локального» мінімуму для функцій з багатьма екстремумами;
- у випадку оптимізації функції багатьох змінних є швидшим за градієнтний спуск через відсутність необхідності обчислень багатовимірних градієнтів.

Класичний генетичний алгоритм — метод стохастичного пошуку, що базується на принципах природного процесу еволюції. Такий пошук відбувається шляхом випадкової еволюції системи частинок (індивідів) та їх адаптації у не обовязково гомогенному середовищі, представленому як набір цільових функцій [3].

Так найпростіший генетичний алгоритм складається з двох стадій генетичних трансформацій, застосованих до набору частинок:

$$\xi_n = (\xi_n^1 \dots \xi_n^N) \rightarrow \text{Відбір} \rightarrow \hat{\xi}_n = (\hat{\xi}_n^1 \dots \hat{\xi}_n^N) \rightarrow \text{Мутація} \rightarrow \xi_{n+1},$$

де N — розмір популяції незалежних випадкових частинок $\xi^i_n \in R^d$ розмірності d (у цій роботі ми розглядаємо випадок при $d = 0$, тобто кожна частинка є дійсним числом, проте в задачах оптимізації функції багатьої змінних d може бути як завгодно великим числом).

Процедура відбору:

У процедурі відбору кожна з частинок $\tilde{\xi}_n = (\tilde{\xi}_n^1 \dots \tilde{\xi}_n^N)$ відбирається випадково та незалежно від попередньої конфігурації відповідно до міри Гіббса:

$$G(\xi_n^i) = e^{-V(\xi_n^i)/T_n}, \quad (1)$$

де $V(\xi^i_n)$ — цільова функція, T_n — параметр температури (зменшується в процесі проведення ітерацій).

Імовірність того, що *i*-ту частинку буде відібрано:

$$P\left(\hat{\xi}_n^i = \xi_n^i \middle| \xi_n\right) = G\left(\xi_n^i\right) \quad . \tag{2}$$

Якщо *i*-ту частинку не відібрано, то на її місце відбирається інша *j*-та частинка з імовірністю:

$$P\left(\hat{\xi}_n^i = \xi_n^j \middle| \xi_n\right) = \frac{G\left(\xi_n^j\right)}{\sum_{i=1}^{N} G\left(\xi_n^i\right)} \quad . \tag{3}$$

Процедура мутації:

Частинки в новій популяції отримуємо таким чином:

$$\xi_{n+1}^i = \begin{cases} \hat{\xi}_n^i + t(2\,rand() - 1), & \xi_{n+1}^i \in C \\ \hat{\xi}_n^i, & \xi_{n+1}^i \in C^c \end{cases} , \tag{4}$$

де t — відносно мале число, що характеризує «амплітуду» мутації частинки, *rand*() — функція, що повертає рівномірно розподілену величину в проміжку [0, 1]. C — множина значень ξ, що задовольняють умовам-обмеженням, C^c — комплемент множини C.

Процедури відбору та мутації відбуваються доки цільова функція не буде мінімізована. При цьому параметр температури T_n зменшуємо після проходження певної кількості ітерацій — це збільшує «тиск відбору» та забезпечує кращу збіжність алгоритма.

Бінарний генетичний алгоритм базується на іншому підході, ідея якого — представити кожну частинку $\xi^i_n \in R^d$ як $\xi_n^i \in R^{d+1}$ у бінарному вигляді (бінарна 32 або 64-бітна «хромосома» є набором «генів» — бітів). Наприклад у випадку $d = 0$ (частинка є дійсним числом), бінарна частинка матиме розмірність $d = 1$, тобто буде вектором (рис. 1). У цьому випадку до кожного булівського значення такого вектора застосовуються бінарні генетичні оператори. Такий оператор визначає процес мутації, а ймовірнісний підхід до процедури відбору заміняється відомою технікою «турнірного відбору». Також застосовується оператор схрещування, що в теорії пришвидшує збіжність. Оператор схрещування існує і в класичній теорії, але він є менш інтуїтивним і поступається мутації за важливістю, у той час як у бінарному підході він посідає центральне місце.

Першим кроком є формування популяції частинок (хромосом) ξ^i_0 у 32 або 64-бітній формі залежно від бажаної точності. На рис. 1 представлено 64-бітну форму числа з рухомою комою (знак, експонента та мантиса) [5]:

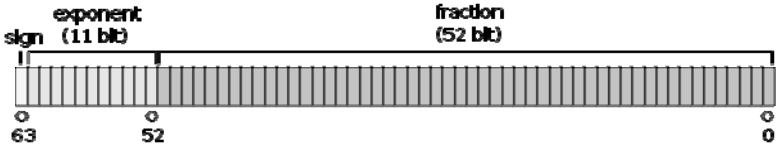

Рис. 1. Частинка, представлена у 64-бітній формі

Процедура відбору:

«Турнірний відбір» передбачає повторення N разів наступної процедури (N розмір популяції): з усіх частинок відбирається 2 групи по m частинок, де $m < N$ — розмір «турнірного пулу» $P_i(1)$ та $P_i(2)$, $i \in [1. N]$. Поміж учасників кожного з двох пулів відбирається найбільш «пристосована» (така, значення цільової функції для якої є найменшим в пулі) частинка $\hat{\xi}^i_n(1) = (\inf_{\xi \in P_i(1)} V(\xi))$ та $\hat{\xi}^i_n(2) = (\inf_{\xi \in P_i(2)} V(\xi))$.

Процедура схрещення:

На всіх ітераціях $i \in [1. N]$ проводимо схрещення шляхом обміну генами між $\hat{\xi}^i_n(1)$ та $\hat{\xi}^i_n(2)$. Формально: для кожного біту, індексованого $b = [1.64]$, $\hat{\xi}^i_n[b] = \hat{\xi}^i_n(1)[b]$ з імовірністю ½ та відповідно $\hat{\xi}^i_n[b] = \hat{\xi}^i_n(2)[b]$ з імовірністю ½. Таким чином, індивід нової популяції набуває кожен ген від одного з відповідних генів батьків. Звісно, якщо в обох батьків i-ий ген однаковий (рівний 0 або 1), то нова частинка матиме це саме значення з імовірністю 1.

Процедура мутації:

Для кожної частинки $\hat{\xi}^i_n$ де $i \in [1. N]$, для кожного біту, індексованого $b = [1.64]$, змінюємо його стан $\hat{\xi}^i_n[b] = 0$ на $\hat{\xi}^i_n[b] = 1$ (та навпаки) з певною імовірністю p, симулюючи дискретний розподіл імовірностей. Відповідно, з імовірністю $1 - p$ мутація

гена не відбувається. Таким чином у подібному до біологічної мутації вигляді ми спостерігаємо перепрограмування хромосоми.

Функція прийняття: Очевидно, що випадкова зміна генів у 64-бітній конструкції може викликати те, що отримане число не буде задовольняти умовам. Так само схрещення чи навіть випадкова генерація частинок у бінарному вигляді може викликати появу чисел несумісних з умовами задачі. Для уникнення цього вводиться функція прийняття: При застосуванні кожного бінарного оператора ця функція перевіряє, чи оператор може бути застосований до частинки (чи належатиме значення частинки до множини умов *C* після застосування такого оператора).

Саме функція прийняття через велику кількість умов перевірки гарантує збіжність бінарного алгоритму, але в той же час робить його повільнішим порівняно з класичним. Проведемо чисельний експеримент експеримент з метою порівняння ефективності обох підходів до імплементації генетичного алгоритму.

Чисельний експеримент проведемо на прикладі типової задачі, що полягає у знаходженні за допомогою двох алгоритмів глобального мінімуму квадратичної (випуклої) функції $f(x) = x^2 - 6x - 3$ та функції Растригіна $f(x) = An + \sum_{i=1}^{n}[x_i^2 - A\cos(2\pi x_i)]$, встановивши $n = 1$.

Конфігурація класичного алгоритму: $T_o = 1000$, $N_{mc} = 2^{14}$ (максимальна кількість ітерацій), $\xi_0 = (-1 + 2rand(30))5$ — генерація першої популяції з 30 частинок, де функція *rand*(30) повертає 30 частинок, що лежать в множині умов $\xi_0^i \in [-C, C]$ ($C = 5$) $\forall\, i \in [1.30]$.

$\epsilon = 0.001$ — бажана точність. Реальний мінімум параболи лежить в точці (3, –12), а функції Растригіна — в точці (0,0) *t* — амплітуда мутації.

Конфігурація бінарного алгоритму: $\xi_0 = (-1 + 2rand(30))5$ — генерація першої популяції з 30 частинок, де функція *rand*(30) повертає 30 частинок, що лежать у множині умов $\xi_0^i \in [-C, C]$ ($C = 5$) $\forall\, i \in [1.30]$.

$\epsilon = 0.001$ — бажана точність. *p* — параметр мутації.

Чисельні результати: Для кожного алгоритму для всіх значень параметрів проведено по 10 експериментів і вирахувано середню кількість ітерацій і середній час, витрачений на ці ітерації. Результати наведено в табл. 1 і 2.

Таблиця 1

РЕЗУЛЬТАТИ ЗБІЖНОСТІ КЛАСИЧНОГО ГЕНЕТИЧНОГО АЛГОРИТМУ

Тестова Функція	Результати збіжності	$p = 0.2$	$p = 0.1$	$p = 0.5$	$p = 0.02$	$p = 0.01$	$p = 0.05$
Квадратична функція	Сер. час збіжності (мс)	6	10	10	90	140	680
	Сер. к-сть ітерацій	7.8	13.7	14.6	203.9	217.2	787.6
Функція Растригіна	Сер. час збіжності (мс)	40	40	50	130	120	140
	Сер. к-сть ітерацій	62.5	84.4	98.7	204.5	165.1	210.1

Таблиця 2

РЕЗУЛЬТАТИ ЗБІЖНОСТІ БІНАРНОГО ГЕНЕТИЧНОГО АЛГОРИТМУ

Тестова Функція	Використані генетичні оператори	Результати збіжності	$p = 0.2$	$p = 0.1$	$p = 0.5$	$p = 0.02$	$p = 0.01$	$p = 0.005$
Квадратична функція	Тільки Мутація	Сер. час збіжності (мс)	6913	2962	7683	6312	8740	9008
		Сер. к-сть ітерацій	4.11	1.66	5.88	4.66	6.77	7.11
	Тільки Схрещення ($p = 0$)	Сер. час збіжності (мс)	5822					
		Сер. к-сть ітерацій	3.33					
	Мутація і схрещення	Сер. час збіжності (мс)	7057	7816	9763	6372	5793	8018
		Сер. к-сть ітерацій	4.44	4.88	6.33	3.88	3.22	5.11
Функція Растригіна	Тільки Мутація	Сер. час збіжності (мс)	130395	258667	144544	146560	49122	123028
		Сер. к-сть ітерацій	115	229.33	114.33	115	43.66	103
	Тільки Схрещення ($p = 0$)	Сер. час збіжності (мс)	68766.78					
		Сер. к-сть ітерацій	59					
	Мутація і схрещення	Сер. час збіжності (мс)	146658	56667	112000	48620	41016	62454
		Сер. к-сть ітерацій	129.3	49	98.33	43.33	38.66	59.66

На рис. 2 і 3 зображено збіжність класичного алгоритму для квадратичної функції та функції Растригіна відповідно. Для бінарного алгоритму такі рисунки будуть ідентичними.

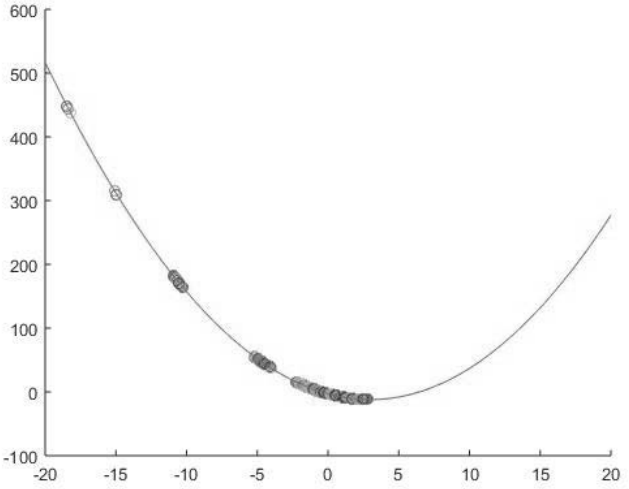

Рис. 2. Збіжність класичного алгоритму (квадратична функція)

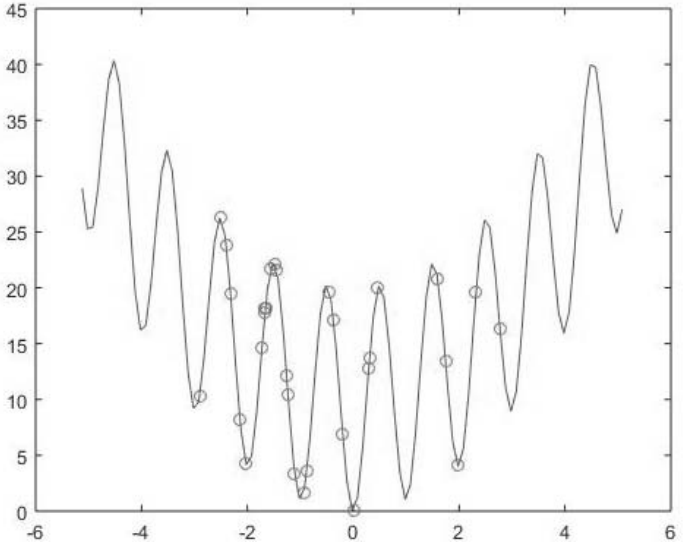

Рис. 3. Збіжність класичного алгоритму (функція Растригіна)

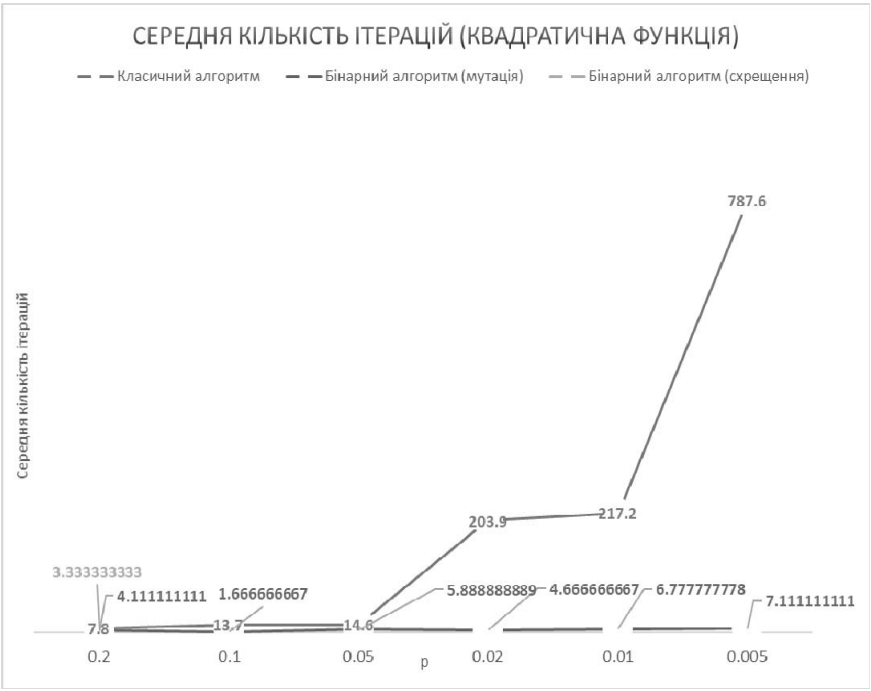

Рис. 4. Середня кількість ітерацій двох алгоритмів
(для квадратичної функції)

На рис. 4 зображено результати експерименту. Представлені результати для класичного алгоритму та двох випадків реалізації бінарного алгоритму: бінарний алгоритм з мутацією та без схрещування, бінарний алгоритм зі схрещуванням і без мутації.

Для ясності рисунку вважаємо параметри p і t еквівалентними. З отриманих даних робимо такий висновок: збіжність класичного алгоритму залежить від параметра мутації t, при зменшенні якого збільшується час збіжності (відповідно, знижується ефективність). Варто зазначити, що для випадків з $t = \{0.02, 0.01, 0.005\}$ збіжність деколи не досягається взагалі. На рис. 5 деталізовано результати збіжності бінарного алгоритму.

Рис. 5. Середня кількість ітерацій бінарного алгоритму
(для квадратичної функції)

На рис. 5 розглядаємо виключно бінарний алгоритм. Експериментальні результати було отримано для трьох випадків: бінарний алгоритм з мутацією та без схрещування, бінарний алгоритм зі схрещуванням і без мутації, бінарний алгоритм з мутацією та схрещуванням. З результатів робимо такий висновок: немає явної залежності між кількістю ітерацій до збіжності бінарного алгоритму та параметром мутації p.

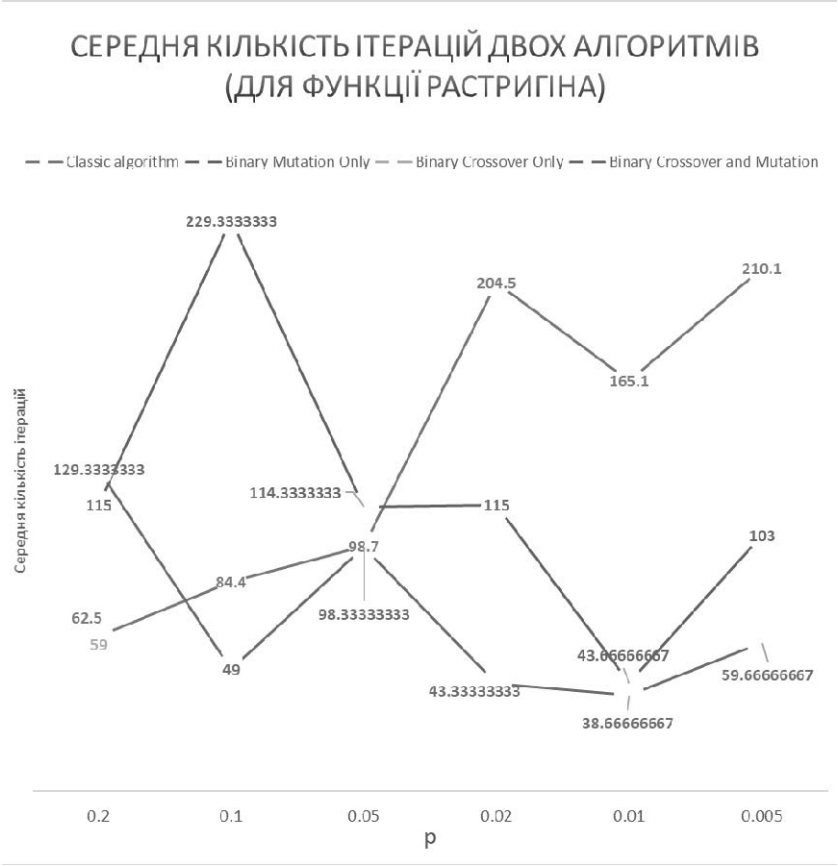

Рис. 6. Середня кількість ітерацій двох алгоритмів
(для функції Растригіна)

На рис. 6 зображено результати експерименту (кількість ітерацій до збіжності класичного та бінарного алгоритмів для квадратичної функції). При цьому представлені результати для трьох випадків реалізації бінарного алгоритму: бінарний алгоритм з мутацією та без схрещування, бінарний алгоритм зі схрещуванням і без мутації, бінарний алгоритм з мутацією та схрещуванням.

Представлені результати підтверджують висновки щодо класичного та бінарного алгоритмів, отримані в результаті оптимізації квадратичної функції (рис. 4).

З отриманих результатів модельних експериментів можемо зробити такі висновки, які є справедливими для обох функцій:

- генетичний алгоритм дозволяє знайти глобальний оптимум функції, не вимагаючи від неї ані випуклості, ані диференційованості в усіх точках, проте є ефективним і для випадку випуклої функції;
- збіжність класичного алгоритму залежить від параметра мутації t, при зменшенні якого збільшується час збіжності (відповідно, знижується ефективність алгоритму);
- для випадків з занадто малим t ($t = \{0.02, 0.01, 0.005\}$) збіжність класичного алгоритму деколи не досягається взагалі;
- немає явної залежності між кількістю ітерацій до збіжності бінарного алгоритму та параметром мутації p;
- час виконання однієї ітерації для бінарного алгоритму (за даної програмної реалізації) перевищує такий для класичного.

Повільність бінарного алгоритму зумовлюється масивністю функції прийняття, що значно підвищує його складність. Якщо отримати таку функцію прийняття, із використанням якої складність буде зростати лінійно відносно розміру популяції чи кількості генів у хромосомах, цей алгоритм може виявитись привабливішим за класичний у випадку, коли структура досліджуваної функції вимагає використання $t < 0.05$. Як було продемонстровано на рис. 4 та рис. 6, у цьому випадку кількість ітерацій до збіжності класичного алгоритму суттєво зростає.

У цій роботі було продемонстровано, що класичний та бінарний алгоритми здатні знаходити оптимум як випуклих, так і багатоекстремальних функцій, не зупиняючись у локальних мінімумах. Іншою перевагою таких стохастичних алгоритмів є відсутність необхідності рохрахунку багатовимірних градієнтів, що дає відчутний виграш у швидкості збіжності за умови, що функція має багато змінних. Залежно від структури досліджуваної функції, кількість ітерацій до збіжності класичного генетичного алгоритму залежить від значення параметра мутації, у той час як збіжність бінарного алгоритму не залежить від параметру мутації.

Генетичні алгоритми є потужним інструментарієм для вирішення широкого класу задач пошуку глобального оптимуму різних цільових функцій. Зокрема, генетичні алгоритми широко застосовуються у задачах калібрування моделей на фінансовому ринку, побудованих за технологіями штучного інтелекту.

Література

1. Holland, J.H. *Adaptation* in *Natural* and *Artificial Systems* // *Ann Arbor* MI: *University* of *Michigan Press*, *Ann Arbor* (2nd Edition, MIT *Press, 1992*) — 1976. [Електронний ресурс] — Режим доступу: www.pdfs.semanticscholar.org/1e21/c514f89375098dec5b947aa5f6bcdd0377c5.pdf

2. Cerf R. Asymptotic convergence of genetic algorithms // *Advances in Applied Probability* — 1997 — V.30, Iss.2 — P. 521–550.

3. Del Moral P., Miclo L. Asymptotic results for genetic algorithms with aplications to non-linear estimation // *Theoretical Aspects of Evolutionary Computing* — 2001 — P. 439–493.

4. Dieterle F. Variable selection by genetic algorithms, 2012. [Електронний ресурс] — Режим доступу: www.frank-dieterle.de/phd/2_8_5.html.

5. Mississippi College. Decimal to Floating-Point Conversions. [Електронний ресурс] — Режим доступу: www.sandbox.mc.edu/~bennet/cs110/flt/dtof.html.

References

1. Holland, J.H. Adaptation in Natural and Artificial Systems // Ann Arbor MI: University of Michigan Press, Ann Arbor (2nd Edition, MIT Press, 1992) — 1976. [Elektronnyj resurs] — Rezhym dostupu: www.pdfs.semanticscholar.org/1e21/c514f89375098dec5b947aa5f6bcdd0377c5.pdf

2. Cerf R. Asymptotic convergence of genetic algorithms // Advances in Applied Probability — 1997 — V.30, Iss.2 — p. 521–550.

3. Del Moral P., Miclo L. Asymptotic results for genetic algorithms with aplications to non-linear estimation // Theoretical Aspects of Evolutionary Computing — 2001 — p. 439–493.

4. Dieterle F. Variable selection by genetic algorithms, 2012. [Elektronnyj resurs] — Rezhym dostupu: www.frank-dieterle.de/phd/2_8_5.html.

5. Mississippi College. Decimal to Floating-Point Conversions. [Elektronnyj resurs] — Rezhym dostupu: www.sandbox.mc.edu/~bennet/cs110/flt/dtof.html.

Статтю подано до редакції 22.10.2018 р.