КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ

Звіт до лабораторної роботи

з дисципліни
Проблеми багатозначного аналізу

Варіант 6

Виконав
студент II курсу магістратури
групи ОМ-2
Пишко Андрій

Київ – 2023

Зміст

1 Постановка задачі

Задана функція

$$f(x_1, x_2) = |x_1 + x_2 - 4| + |x_1 - 2x_2 + 1|, \quad x = (x_1, x_2) \in \mathbb{R}^2.$$

Знайти субдиференціал функції $f(x)$, проаналізувати його властивості як багатозначного відображення. Побудувати графік субдиференціалу функції $f(x)$ або відобразити його характеристики. Розв'язати задачу

$$f(x) \to \min,$$

застосовуючи один з числових методів негладкої оптимізації.

2 Пошук субдиференціалу

2.1 Перевірка на наявність субдиференціалу

Перед тим як шукати субдиференціал, треба переконатись, що він існує.

Обидва доданки $f(x)$ є опуклими функціями. Звідси випливає опуклість самої функції $f(x)$ (згідно з лемою 14.4 [1]).

Існування субдиференціалу випливає з того, що він є множиною всіх субградієнтів функції. А, як відомо, для опуклої функції, визначеної на відкритій опуклій множині, завжди існує хоча б один субградієнт у будь-якій точці множини.

2.2 Обчислення субдиференціалу

Для обчислення субдиференціалу функції $f(x)$ представимо його як суму диференціалів функцій $U(x)$ і $V(x)$:

$$\partial f\left(x_1, x_2\right) = \partial U\left(x_1, x_2\right) + \partial V\left(x_1, x_2\right),$$

де

$$U(x_1, x_2) = |x_1 + x_2 - 4|,$$
$$V(x_1, x_2) = |x_1 - 2x_2 + 1|.$$

Знайдемо субдиференціал функцій $U(x)$ та $V(x)$ як субдиференціал максимуму опуклих функцій. Для цього спочатку розкриємо модулі функцій $U(x)$ та $V(x)$ і запишемо систему:

$$U(x_1, x_2) = \begin{cases} -x_1 - x_2 + 4, & x_2 < -x_1 + 4, \\ 0, & x_2 = -x_1 + 4, \\ x_1 + x_2 - 4, & x_2 > -x_1 + 4, \end{cases}$$

$$V(x_1, x_2) = \begin{cases} x_1 - 2x_2 + 1, & x_2 < 0.5x_1 + 0.5, \\ 0, & x_2 = 0.5x_1 + 0.5, \\ -x_1 + 2x_2 - 1, & x_2 > 0.5x_1 + 0.5. \end{cases}$$

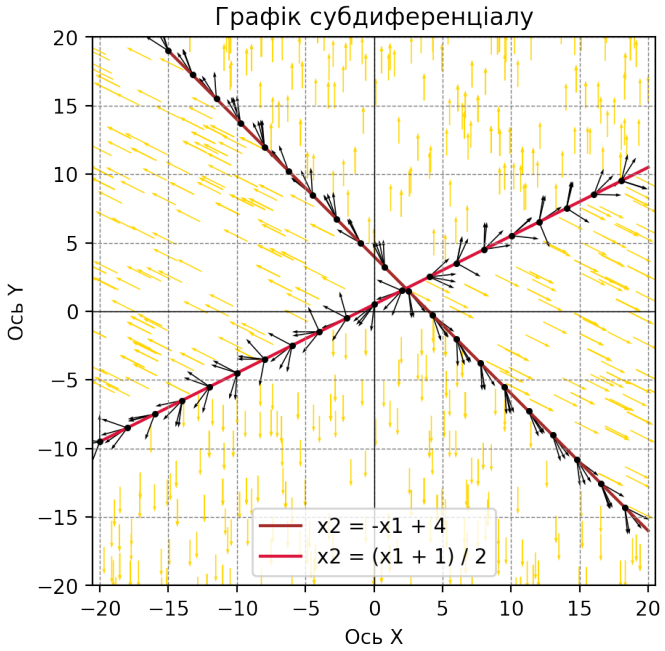
Тоді субдиференціал $U(x)$ та $V(x)$ матиме вигляд:

$$\partial U(x_1, x_2) = \begin{cases} \{(-1,-1)\}, & x_2 < -x_1 + 4, \\ \{(\alpha,\alpha)\}, & x_2 = -x_1 + 4, \quad -1 \le \alpha \le 1, \\ \{(1,1)\}, & x_2 > -x_1 + 4, \end{cases}$$

$$\partial V(x_1, x_2) = \begin{cases} \{(1,-2)\}, & x_2 < 0.5x_1 + 0.5, \\ \{(\beta,-2\beta)\}, & x_2 = 0.5x_1 + 0.5, \quad -1 \le \beta \le 1, \\ \{(-1,2)\}, & x_2 > 0.5x_1 + 0.5. \end{cases}$$

Оскільки $\partial f\,(x_1, x_2) = \partial U\,(x_1, x_2) + \partial V\,(x_1, x_2)$, маємо:

$$\partial f\,(x_1, x_2) = \begin{cases} \{(0,-3)\}, & x_2 < -x_1 + 4, \quad x_2 < 0.5x_1 + 0.5, \\ \{(-1+\beta,-1-2\beta)\}, & x_2 < -x_1 + 4, \quad x_2 = 0.5x_1 + 0.5, \quad -1 \le \beta \le 1, \\ \{(-2,1)\}, & x_2 < -x_1 + 4, \quad x_2 > 0.5x_1 + 0.5, \\ \{(1+\alpha,-2+\alpha)\}, & x_2 = -x_1 + 4, \quad x_2 < 0.5x_1 + 0.5, \quad -1 \le \alpha \le 1, \\ \{(\alpha+\beta,\alpha-2\beta)\}, & x_2 = -x_1 + 4, \quad x_2 = 0.5x_1 + 0.5, \quad -1 \le \alpha \le 1, \quad -1 \le \beta \le 1, \\ \{(-1+\alpha,2+\alpha)\}, & x_2 = -x_1 + 4, \quad x_2 > 0.5x_1 + 0.5, \quad -1 \le \alpha \le 1, \\ \{(2,-1)\}, & x_2 > -x_1 + 4, \quad x_2 < 0.5x_1 + 0.5, \\ \{(1+\beta,1-2\beta)\}, & x_2 > -x_1 + 4, \quad x_2 = 0.5x_1 + 0.5, \quad -1 \le \beta \le 1, \\ \{(0,3)\}, & x_2 > -x_1 + 4, \quad x_2 > 0.5x_1 + 0.5. \end{cases}$$



3 Розв'язання задачі мінімізації

Для того, щоб розв'язати задачу мінімізації функції $f(x_1, x_2)$, скористаємося генетичним алгоритмом.

3.1 Опис алгоритму

Генетичний алгоритм - це еволюційний алгоритм, надхненний природним відбором та генетичними процесами, які відбуваються в живих організмах.

Процес генетичного алгоритму полягає у повторенні ітерацій, в яких створюються нові покоління рішень на основі придатності та генетичних операцій (схрещень та мутацій), поки не досягнута задовільна якість рішення або не вичерпані обмежені ресурси (в нашому випадку – обмежена кількість поколінь).

Будова генетичного алгоритму може бути узагальнена наступним чином:

1. **Ініціалізація**. Випадковим чином генерується початковий набір потенційних рішень (індивідів).
2. **Оцінка придатності**. Кожен індивід оцінюється за допомогою функції придатності (фітнес-функції), яка визначає, наскільки добре відповідає рішення вирішуваній задачі.
3. **Відбір індивідів для схрещування**. Вибираються найбільш придатні індивіди для участі у схрещуванні. У нашому випадку це будуть ті точки, в яких функція прийматиме найменше значення.
4. **Генетичні операції**.
   (a) **Схрещування**. Пари батьків обмінюються генетичною інформацією для створення нащадка.
   (b) **Мутація**. Зміни вносяться в генетичну інформацію індивідів, щоб не «застрягти» у локальному екстремумі.
5. **Формування нового покоління**. Нове покоління формується на основі результатів схрещування та мутації.
6. **Оцінка придатності нового покоління**.
7. **Завершення критерію**. Перевірка, чи виконано критерій завершення (наприклад, досягнення певного рівня придатності чи обмеження кількості ітерацій).
8. **Вивід результатів або повторення**.

Щоб відповісти на питання «Чому саме генетичний алгоритм?», розглянемо деякі його переваги над традиційними методами оптимізації.

1. **Робота з нелінійністю та неперервністю**. Генетичні алгоритми добре пристосовані для оптимізації у випадках, коли функція придатності не має властивостей диференційовності або коли вона має багато локальних мінімумів.

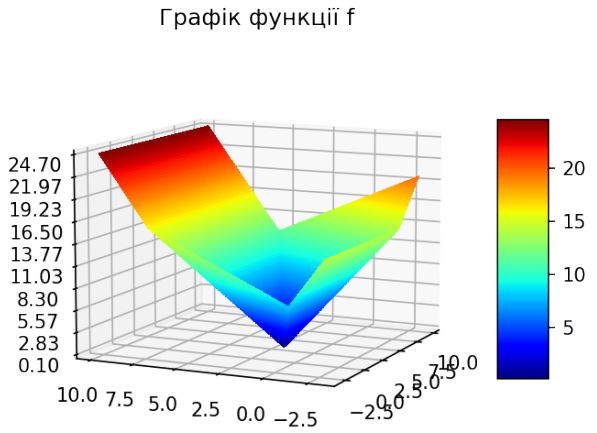
2. **Глобальна оптимізація**. Генетичні алгоритми добре працюють для глобальної оптимізації, оскільки вони здатні виявляти рішення в різних частинах простору пошуку.

3. **Пошук оптимальних рішень в великому просторі**. Генетичні алгоритми дозволяють вирішувати задачі великої розмірності, де, наприклад, субградієнтні методи можуть стикатися з проблемою величезного обчислювального обсягу.

4. **Паралелізм**. Генетичні алгоритми можуть ефективно використовувати паралельні обчислення, що робить їх придатними для використання в обчислювально інтенсивних задачах.

3.2 Застосування алгоритму

Розглянемо трьохмірний графік функції $f(x_1, x_2)$.



З графіка видно, що мінімум $f(x_1, x_2)$ лежить у проміжку $x_1, x_2 \in (0, 10)$. Проте не будемо полегшувати собі задачу та застосуємо генетичний алгоритм для обширного простору значень x_1 та x_2.

Отже, для вхідних параметрів

Діапазон x_1	Діапазон x_2	Розмір популяції	Імовірність мутації
$(-10^{30}, 10^{30})$	$(-10^{30}, 10^{30})$	1000	0,2

з умовою зупинки $f(x_{1,i-1}, x_{2,i-1}) - f(x_{1,i}, x_{2,i}) < 10^{-6}$, отримали наступні результати (табл. 1):

Номер популяції	**Найменше значення функції f**
0	$6.455303395604928 \times 10^{28}$
1	$6.921743117417395 \times 10^{27}$
2	$6.151955859011057 \times 10^{26}$
3	$2.9581850056610194 \times 10^{26}$
4	$6.782960394158161 \times 10^{25}$
5	$4.953978663614238 \times 10^{24}$
6	$1.2527977034656404 \times 10^{24}$
7	$4.953912175531149 \times 10^{23}$
...	...
43	0.0011180133176273976
44	0.0003024255728623082
45	$2.4168908893340557 \times 10^{-05}$
46	$1.8554651727509253 \times 10^{-05}$
47	$1.4117626867538036 \times 10^{-06}$
48	$2.8936710272375876 \times 10^{-07}$
49	$1.6262509028308614 \times 10^{-07}$
50	$1.2548909467113845 \times 10^{-08}$

Таблиця 1

Найменшим значенням функції виявилося $1.2548909467113845 \times 10^{-08}$, при якому $x_1 = 2.3333$ та $x_2 = 1.6666$.

Варто кілька слів сказати про умову завершення алгоритму. Мною було реалізовано два випадки, коли програма зупиняє роботу:

1) $f(x_{1,i-1}, x_{2,i-1}) - f(x_{1,i}, x_{2,i}) < 10^{-6}$,

2) обмеження кількості поколінь.

Проте на практиці виявилося, що перший випадок краще підходить для великого пошукового діапазону, а другий – для малого. Це відбувається через те, що зміна «генів» поколінь відбувається випадковим чином, тому і різниця між поколіннями може бути як малою так і великою в залежності від функції рандому. Через це на малих діапазонах можлива ситуація, коли різниця між поколіннями настільки несуттєва, що виконується умова зупинки, а завдання мінімізації виконано гірше, ніж воно могло бути виконано за другої умови зупинки.

3.3 Результати

Проаналізуємо результати роботи програми для різних вхідних параметрів та умов зупинки.

Для початку застосуємо алгоритм до обширного діапазону $x_1, x_2 \in (-10^{30}, 10^{30})$. При цьому ймовірність мутації залишимо незмінною і рівною 20%, а умовою зупинки оберемо:

$$f(x_{1,i-1}, x_{2,i-1}) - f(x_{1,i}, x_{2,i}) < 10^{-6}.$$

Номер експерименту	**1**	**2**	**3**
Розмір популяції	10^3	10^4	10^5
К-ть створених поколінь	51	50	49
Час виконання, сек	2.4678	27.5919	313.0692
x_{1best}	2.333...	2.333...	2.333...
x_{2best}	1.666...	1.666...	1.666...
Найкращий результат	1.2549×10^{-08}	1.2563×10^{-08}	2.0857×10^{-08}

Таблиця 2

З таблиці 2 бачимо, що мінімальне значення $f(x_1, x_2)$ досягається при $x_1 \approx 2.333$ та $x_2 \approx 1.666$.

Зауважимо, що збільшення розміру популяції суттєвого ефекту не дало та навіть трохи погіршило результат (це не є закономірністю). При цьому є помітним лінійне збільшення складності обчислень (рис. 1).

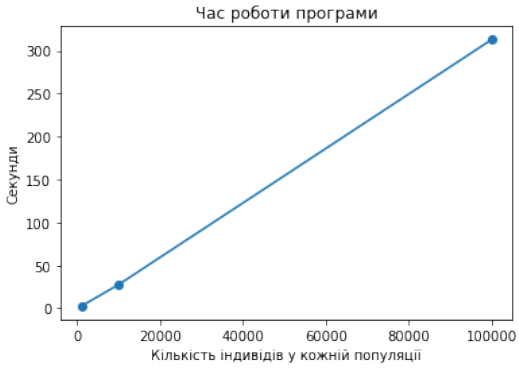


Рисунок 1

Тепер, враховуючи інформацію з попередніх обчислень про місцезнаходження шуканих x_1 та x_2, звузимо простір обчислень до $(-10, 10)^2$. При

цьому ймовірність мутації залишимо незмінною і рівною 20%, а умовою зупинки оберемо обмеження на кількість поколінь у 100 одиниць. Результати роботи алгоритму для даних вхідних параметрів представлені у табл. 3.

Номер експерименту	**1**	**2**	**3**
Розмір популяції	10^3	10^4	10^5
Час виконання, сек	4.9336	56.5565	56.8277
x_{1best}	2.333...	2.333...	2.333...
x_{2best}	1.666...	1.666...	1.666...
Найкращий результат	0.0	0.0	0.0

Таблиця 3

З кожною новою популяцією, шукані x_1 та x_2 все більше й більше наближаються до 2.33(3) та 1.66(6) відповідно. При цьому, наприклад, для першого експерименту вже на 23-му поколінні мінімум функції $f(x_1, x_2)$ набуває настільки малого значення, що комп'ютер перетворює його на нуль (табл. 4).

Номер покоління	**Найменше значення** $f(x_1, x_2)$
0	1.1047633971913076
1	0.2424391407110993
2	0.04295463169720071
3	0.010919297718240628
4	0.0017000493885102053
...	...
20	$3.5083047578154947 \times 10^{-14}$
21	$1.2434497875801753 \times 10^{-14}$
22	$1.7763568394002505 \times 10^{-15}$
23	0.0
24	0.0
...	...
100	0.0

Табл. 4 : Результати 1-го експерименту

Для другого та третього експерименту мінімальне $f(x_1, x_2)$ набувало

настільки ж малого значення на 21-му та 20-му поколіннях відповідно.

Лінійне збільшення складності при збільшені величини поколінь зберігається (рис. 2).

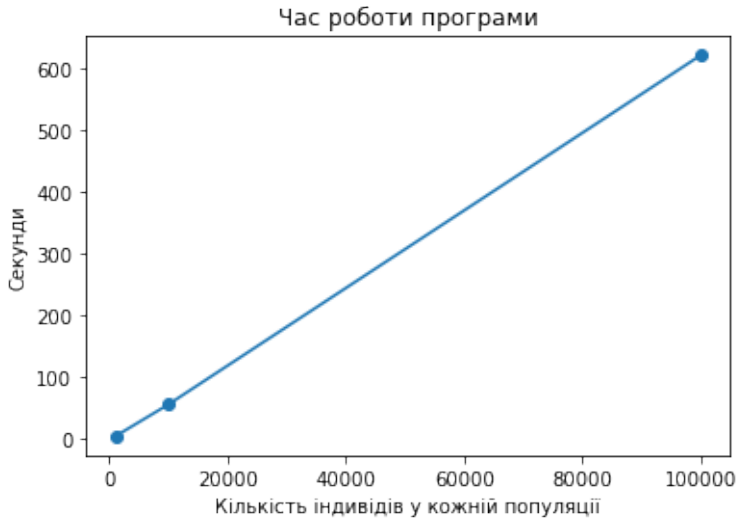


Рисунок 2

Під кінець розглянемо роботу програми при різних значеннях імовірності мутації. При цьому $x_1, x_2 \in (-10, 10)$, кількість індивідів у поколінні обмежимо 100, а за умову зупинки візьмемо обмеження кількості поколінь у 100 одиниць (табл. 5).

Номер експерименту	**1**	**2**	**3**
Імовірність мутації	20%	40%	60%
Час виконання, сек	0.4917	0.5223	0.5167
x_{1best}	2.33336...	2.33339...	2.33334...
x_{2best}	1.66664...	1.66661...	1.66665...
Найкращий результат	8.1923×10^{-5}	1.5739×10^{-4}	2.9659×10^{-5}

Таблиця 5

Зміна величини ймовірності мутації на швидкість роботи програми майже не впливає (рис. 3).

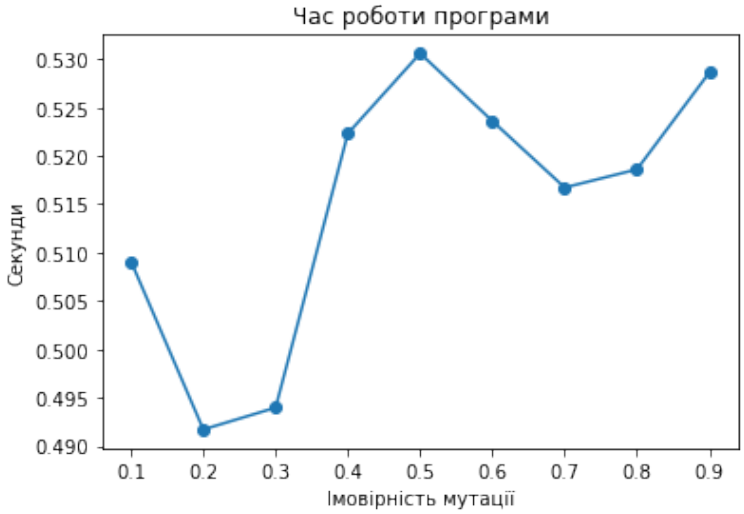


Рисунок 3

4 Висновки

У даній лабораторній роботі було розглянуто поняття субдиференціалу та обраховано його значення для функції

$$f(x_1, x_2) = |x_1 + x_2 - 4| + |x_1 - 2x_2 + 1|, \quad x = (x_1, x_2) \in \mathbb{R}^2.$$

з використанням апарату математичних інстументів. Було також побудовано графік субдиференцалу даної функції з використанням мови програмування Python, зокрема бібліотеки *matplotlib*.

Також було реалізовано генетичний алгоритм для знаходження мінімального значення функції $f(x_1, x_2)$ та розглянуто його роботу при різних вхідних параметрах. Генетичний алгоритм виявився доволі зручним і швидким інструментом, що було, зокрема, продемонстровано на графіках залежності часу роботи програми від зміни певних вхідних параметрів алгоритму.

Список використаних джерел

[1] ЖАЛДАК, М. І.; ТРИУС, Ю. В. Основи теорії і методів оптимізації: навчальний посібник. Черкаси: Брама-Україна, 2005, 608.

[2] MIRJALILI, Seyedali; MIRJALILI, Seyedali. Genetic algorithm. Evolutionary Algorithms and Neural Networks: Theory and Applications, 2019, 43-55.