КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інтелектуальних та інформаційних систем

Лабораторна робота
За темою
Диференціальна еволюція

З дисципліни “Еволюційні обчислення в задачах Web-, Text- та Genetic Mining”

Виконав студент ІІІ курсу ФІТ
Групи КН – 31
Гурленко Антон Євгенович

Перевірив Завідувач кафедри
інтелектуальних та
інформаційних систем
Снитюк Віталій Євгенович

Київ 2016

Постановка задачі

Провести оптимізацію цільових функцій із використанням диференціальної еволюції. Отримані результати порівняти з бажаними, зробити висновки.

Короткі відомості

Диференціальна еволюція (англ. *differential evolution*) — метод багатовимірної математичної оптимізації, що відноситься до класу стохастичних алгоритмів оптимізації (тобто працює з використанням випадкових чисел) і використовує деякі ідеї генетичних алгоритмів.

Це прямий метод оптимізації, тобто він вимагає тільки можливості обчислювати значення цільової функцій, але не її похідних. Метод диференціальної еволюції призначений для знаходження глобального мінімуму (або максимуму) недиференційованих, нелінійних, мультимодальних (що мають, можливо, велику кількість локальних екстремумів) функцій від багатьох змінних. Метод простий у реалізації та використання (містить мало керуючих параметрів, що потребують підбору), легко розпаралелюється.

Метод диференціальної еволюції був розроблений Рейнером Сторно і Кеннетом Прайсом, вперше опублікований ними в 1995 році і розроблений в подальшому в їх пізніших роботах.

Алгоритм

Набір *D* оптимізаційних параметрів називається індивідом, він представлений *D*-вимірним параметричним вектором.

Популяція складається з *NP* параметричних векторів x_i^G, i = 1, 2, ..., *NP*.

G позначає одне покоління.

NP - це число членів в популяції. Воно не змінюється в процесі еволюції. Початкова популяція вибирається випадково при рівномірному розподілі.

DE має три оператора: мутації, кросовера і відбору.

Вирішальна ідея DE - схема генерації пробних векторів (trial vectors).

- **Ініціалізація**

Створюємо первісну популяцію

Кожен параметр рішення в кожному векторі вихідної популяції визначається рандомно, відповідно до граничних обмежень:

$$x_{ij}^0 = a_j + rand_j(b_j - a_j) \qquad (2)$$

$rand_j$ - рівномірно розподілене число в діапазоні від [0, 1]

a_j, b_j - нижня і верхня границі для кожного j-го параметра рішення відповідно.

- **Мутація**

Генеруємо нове покоління векторів

Для кожного базового вектора (target vector) x_i^G генерується мутантний (mutant vector) v_i^{G+1}:

$$v_i^{G+1} = x_{r_1}^G + F(x_{r_2}^G - x_{r_3}^G), \qquad r_1 \neq r_2 \neq r_3 \neq i \qquad (3)$$

з рандомно обраними індексами та $r_1, r_2, r_3 \in \{1,2,\dots, NP\}$.

F - дійсне число, що контролює збільшення різниці векторів $(x_{r_2}^G - x_{r_3}^G)$. $F \in [0; 2]$

- **Кросовер**

Базовий вектор схрещується з мутантним вектором для отримання пробного вектора (trial vector) u_i^{G+1} за наступною формулою:

$$u_{ij}^{G+1} = \begin{cases} v_{ij}^G, & rand(j) \le CR \ \ or\ j = randn(i), \\ x_{ij}^G, & rand(j) > CR \ \ and\ j \ne randn(i) \end{cases} \qquad (4)$$

де $j = 1,\ 2,\ \ldots,\ D,\ \ rand(j)\ \epsilon\ [0,1]$ – j-та оцінка рівномірно розподіленого генератора випадкових чисел

$CR\ \epsilon\ [0,1]$ – ймовірнісна константа кросовера

$randn(i)\ \epsilon\ \{1,\ 2,\ \ldots,\ D\}$ – рандомно обраний індекс, який гарантує, що u_i^{G+1} отримає хоча б один елемент від v_i^{G+1}; інакше популяцій не зміниться.

- **Відбір**

DE використовує жадібну стратегію відбору

Тільки якщо пробний вектор u_i^{G+1} дає краще значення функції пристосованості, ніж x_i^G, тоді u_i^{G+1} стає x_i^{G+1} – в новому поколінні базовий вектор замінюється на пробний.

В іншому випадку, старий вектор x_i^G зберігається.

$$x_i^{G+1} = \begin{cases} u_i^{G+1}, & f\left(u_i^{G+1}\right) < f\left(x_i^G\right), \\ x_i^G, & f\left(u_i^{G+1}\right) \ge f\left(x_i^G\right) \end{cases} \qquad (5) \quad \text{(для мінімізації)}$$

- **Умови закінчення**

Вичерпана задана гранична кількість епох еволюції

Вичерпаний заданий граничний фізичний розрахунковий час

Значення критерію оптимізації кращого вектора покоління не змінюється протягом заданої граничної кількості епох еволюції

Досягнуте задовільне значення критерію оптимізації.

Опис програми

- Ініціалізувати всі x з випадковими позиціями в пошуковому просторі.
- Доки не досягнуто заданої кількості ітерацій:
  - Для кожного x в популяції:
    - Беремо три числа a, b і c з популяції у випадковому порядку, вони повинні відрізнятися один від одного, а також від x
    - Беремо випадковий індекс $R \in \{1, \ldots, n\}$ (n - розмірність задачі).
    - Розраховуємо нове значення $y = [y_1, \ldots y_n]$ наступним чином:
      - Для кожного $i \in \{1, \ldots, n\}$, обираємо рівномірно розподілене число $rand(0,1)$
      - Якщо $r_i < CR$ або $i = R$ то $y_i = a_i F(b_i - c_i)$ в іншому випадку $y_i = x_i$
    - Якщо $f(y) < f(x)$ замінюємо в популяції x на y.
- Обираємо розв'язок з популяції, що має найкращу пристосованість і призначаємо його найкращим кандидатом.

Порівняльний аналіз

Порівняльний аналіз

Цільова функція:

Ackley's Function

$$f(x) = 20 + e - 20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}} - e^{\frac{1}{n}\sqrt{\sum_{i=1}^{n}\cos(2\pi x_i)}}$$

$$x \in (-30; 30)$$

$$min \to x_i = 0$$

f – генерується випадково на інтервалі $[0.5; 1.0]$

№	Розмір популяції	Кількість змінних	Ймовірність кросоверу	К-сть ітерацій	ФФ
1	5	2	0.2	10	2.888886142776847
2	5	3	0.5	50	0.04005066011261915
3	5	5	1	100	15.15362028730383
4	20	2	0.2	10	3.5799103673160757
5	20	3	0.5	50	0.08907970439506885
6	20	5	1	100	0.19719878518878176
7	50	2	0.2	10	3.3517847206017377
8	50	3	0.5	50	0.023566506628618367
9	50	5	1	100	0.4835858634527588

Як бачимо, на результат найбільше впливають кількість змінних і ймовірність кросоверу, а вже потім все інше. Тобто, оптимально підібравши параметри, отримуємо:

Розмір популяції	Кількість змінних	Ймовірність кросоверу	К-сть ітерацій	ФФ
50	5	0.7	100	0.004868877912881553
50	5	0.7	1000	-4.440892098500626e-16

Висновок

Диференційна еволюція на мою думку є найефективнішим методом оптимізації. Виходячи з результатів вище, видно, що для точного і швидкого знаходження мінімуму необхідно правильно підібрати основі параметри алгоритму, але після цього алгоритм справився за 100 ітерацій, що не вдавалося жодному з попередніх методів. Алгоритм також має задовільну швидкість в плані комп’ютерного обчислення.