Metaheurystyki — zadanie 2

Algorytm symulowanego wyżarzania

GRUPA 3 — piątek 10:15

Bartosz Kołaciński
251554

Nikodem Nowak
??????

4 listopada 2025

Użyte technologie	Python 3.13
Użyte biblioteki	random, collections.abc, time, math, numpy, matplotlib.pyplot

Spis treści

1 Opis zasad działania algorytmu

1.1 Opis algorytmu symulowanego wyżarzania

1.2 Opis implementacji rozwiązania

Cała logika algorytmu symulowanego wyżarzania została zaimplementowana w klasie `SimulatedAnnealing`. `self.func` to funkcja której maksimum chcemy znaleźć, `self.domain` to obszar w którym szukamy maksimum, a `self.dimensions` to liczba wymiarów problemu optymalizacyjnego.

```
    class SimulatedAnnealing:
        def __init__(self,
                     func: Callable,
                     domain: tuple[float, float] | list[tuple[float, float]]):
        self.func = func
        if isinstance(domain, tuple) and len(domain) == 2 and isinstance(
                 domain[0], (int, float)):
            self.domain = [domain]
        else:
            self.domain = domain
        self.dimensions = len(self.domain)

```

Kod 1: Inicjalizacja klasy SimulatedAnnealing

Metoda `run_epochs` klasy `SimulatedAnnealing` odpowiada za przeprowadzenie określonej liczby epok algorytmu symulowanego wyżarzania. W każdej epoce algorytm wykonuje określoną liczbę iteracji, w trakcie których poszukuje lepszego rozwiązania.
`epochs` to liczba epok do wykonania, `attempts_per_epoch` to liczba prób w każdej epoce, `init_temp` to początkowa temperatura, `alpha` to funkcja schładzająca temperaturę, a `k` to współczynnik wpływający na prawdopodobieństwo akceptacji gorszych rozwiązań.

```
    def run_epochs(self,
                   epochs: int,
                   attempts_per_epoch: int,
                   init_temp: float,
                   alpha: Callable[[float], float],
                   k: float) -> tuple[list[float], float]:

```

Kod 2: Metoda run_epochs klasy SimulatedAnnealing

Na początku metody `run_epochs` zajmujemy się inicjalizacją dwóch wartości. Początkowy punkt (nieograniczony do jednego wymiaru) jest wybierany losowo z podanego przedziału lub obszaru określonego przez `self.domain`. Wartość funkcji w tym punkcie jest obliczana i przechowywana jako `best_f_value`. Temperatura jest inicjalizowana wartością `init_temp`.

```
    best_point = [uniform(self.domain[i][0], self.domain[i][1])
                      for i in range(self.dimensions)]
    best_f_value = self.func(*best_point)
    temp = init_temp

```

Kod 3: Wybór losowego punktu oraz inicjalizacja temperatury

W dalszej części metody `run_epochs` znajduje się główna pętla algorytmu symulowanego wyżarzania. Główna pętla algorytmu (ta z warunkiem stopu) składa się z dwóch zagnieżdżonych pętli: zewnętrznej, która iteruje przez określoną liczbę epok, oraz wewnętrznej, która wykonuje określoną liczbę prób w każdej epoce. W każdej próbie generowany jest nowy punkt poprzez wybranie go z losowego sąsiedztwa aktualnie najlepszego punktu, z uwzględnieniem bieżącej temperatury. Wartość funkcji w nowym punkcie jest obliczana i porównywana z najlepszą dotychczasową wartością. Jeśli nowa wartość jest lepsza, to staje się nowym najlepszym punktem. Jeśli jest gorsza, to istnieje prawdopodobieństwo akceptacji tego gorszego rozwiązania, które zależy od różnicy wartości funkcji oraz bieżącej temperatury. Na koniec każdej epoki temperatura jest schładzana za pomocą funkcji `alpha`.

```
    for _ in range(epochs):
        for _ in range(attempts_per_epoch):
            new_point = []
            for i in range(self.dimensions):
                left, right = self.domain[i]
                new_coord = uniform(best_point[i] - 2 * temp,
                                    best_point[i] + 2 * temp)
                new_coord = max(left, min(new_coord, right))
                new_point.append(new_coord)

            f_value = self.func(*new_point)

            if f_value > best_f_value:
                best_point, best_f_value = new_point, f_value
            else:
                p = uniform(0, 1)
                if exp((f_value - best_f_value) / (k * temp)) > p:
                    best_point, best_f_value = new_point, f_value

        temp = alpha(temp)

    return best_point, best_f_value

```

Kod 4: Główna pętla algorytmu symulowanego wyżarzania

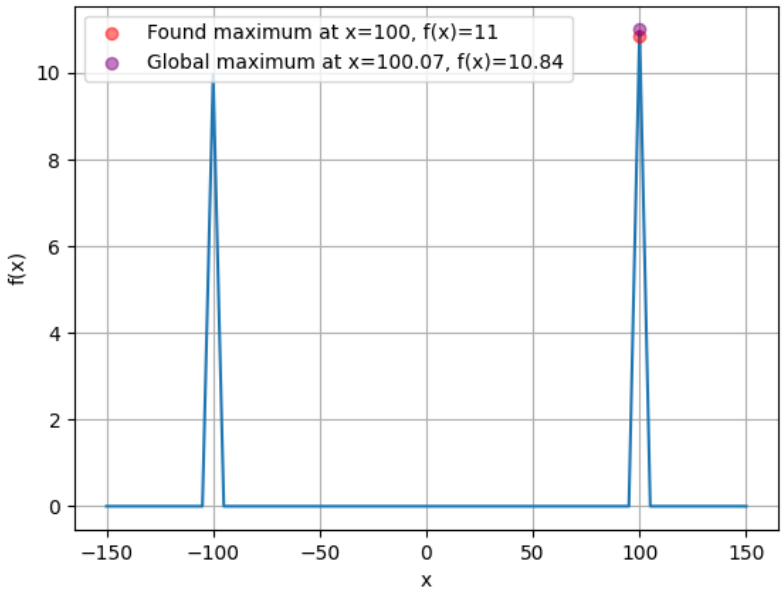
1.3 Opis uruchomienia programu

2 Odtworzenie eksperymentów z artykułu

2.1 Wybrana funkcja z rozdziału 3

Z sekcji 3 wybraliśmy funkcję $f(x)$ z przykładu 1, określoną w przedziale $[-150, 150]$, wyrażoną wzorem:

$$f(x) = \begin{cases} -2|x + 100| + 10 & \text{dla } x \in (-105, -95) \\ -2.2|x - 100| + 11 & \text{dla } x \in (95, 105) \\ 0 & \text{dla } x \notin (-105, -95) \cup (95, 105) \end{cases}$$



Rysunek 1: Wykres funkcji $f(x)$ w przedziale $[-150, 150]$ z oznaczonym znalezionym maksimum

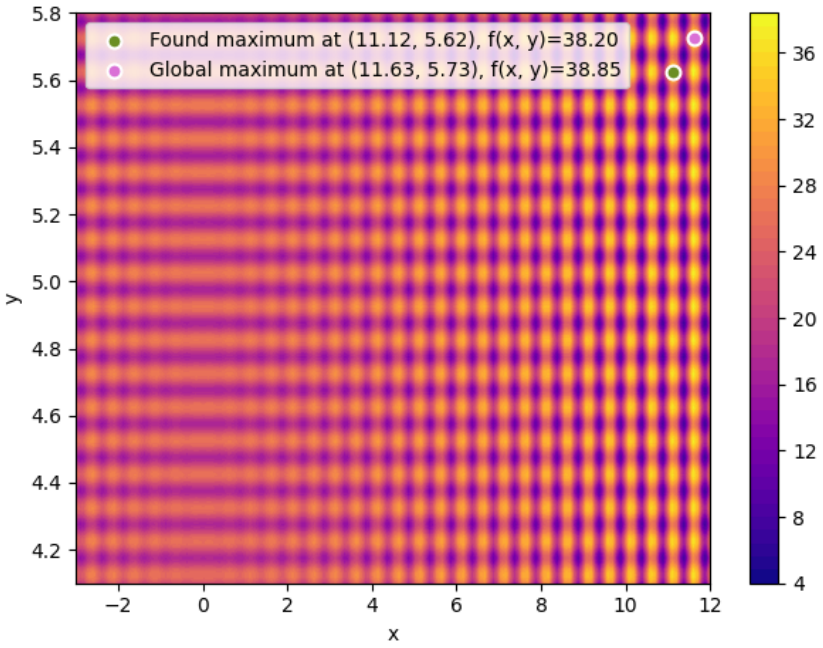
Wybrana funkcja posiada dwa ekstrema lokalne - $f(-100) = 10$ oraz $f(100) = 11$. Funkcja poza okolicami tych ekstremów przyjmuje wartość 0. Jej maksimum globalne to $f(100) = 11$.

Znalezione przez nas maksimum funkcji $f(x)$ to $f(100.07) = 10.84$ (rysunek 1), co różni się od maksimum globalnego o $\delta = |10.84 - 11| = 0.16$.

2.2 Wybrana funkcja z rozdziału 4

Z sekcji 4 wybraliśmy funkcję $f(x)$ z przykładu 5, określoną w obszarze $[-3, 12] \times [4.1, 5.8]$, wyrażoną wzorem:

$$f(x, y) = 21.5 + x \cdot \sin(4 \cdot \pi \cdot x) + y \cdot \sin(20 \cdot \pi \cdot y)$$



Rysunek 2: Wykres funkcji $f(x, y)$ w obszarze $[-3, 12] \times [4.1, 5.8]$ z oznaczonym znalezionym maksimum oraz maksimum globalnym

Wybrana funkcja posiada bardzo dużo ekstremów lokalnych w określonym obszarze. Jej maksimum globalne to $f(12, 5.7) = 38.85$.

Znalezione przez nas maksimum funkcji $f(x, y)$ to $f(11.12, 5.62) = 38.20$ (rysunek 2), co różni się od maksimum globalnego o $\delta = |38.20 - 38.85| = 0.65$.

3 Wnioski