

Algorytm Genetyczny: Maksymalizacja Funkcji Matematycznej

Sara Fijołek i Miron Kwiatkowski

Celem projektu jest stworzenie intuicyjnej aplikacji internetowej, która umożliwia użytkownikowi:

- Wprowadzenie granic przedziału dla zmiennej x ,
- Określenie liczby populacji, liczby pokoleń oraz wskaźnika mutacji,
- Uruchomienie algorytmu genetycznego,
- Otrzymanie rezultatu w postaci optymalnej wartości x oraz wartości maksymalnej funkcji $f(x)$.

1 Algorytm Genetyczny

Algorytm genetyczny wykorzystywany w projekcie opiera się na podstawowych operacjach genetycznych, takich jak:

- **Funkcja celu:** $f(x) = x \cdot \sin(x)$, która jest funkcja matematyczna podlegająca maksymalizacji.
- **Stworzenie populacji:** Początkowa populacja liczy określona liczbę jednostek losowo z przedziału $[a, b]$.
- **Selekcja rodziców:** Do rodzicielskiego doboru używana jest metoda selekcji turniejowej, gdzie najlepsze jednostki zostają wybrane do rozmnażania.
- **Krzyżowanie (crossover):** Krzyżowanie jest wykonywane losowo, z zachowaniem średniej wartości pomiędzy dwoma wybranymi rodzicami.
- **Mutacja:** Każdy nowo stworzony osobnik ma szansę na mutację w zakresie podanego wskaźnika mutacji.

1.1 Szczegóły Algorytmu Genetycznego

- **Funkcja celu:** $f(x) = x \cdot \sin(x)$ jest wykorzystywana do obliczania wartości dla danej zmiennej x .
- **Selekcja turniejowa:** Przeprowadzana jest selekcja najlepszych jednostek z grupy losowych osobników w celu dalszego rozmnażania.

- **Krzyżowanie:** Dwóch wybranych rodziców wymienia między sobą geny, tworząc potomków.
- **Mutacja:** Nowo utworzeni osobniki mają szansę na mutację, co pozwala na generowanie różnych możliwych rozwiązań.

2 Struktura Aplikacji

Aplikacja webowa składa się z dwóch głównych komponentów:

- **Serwer Flask:** Obsługuje żądania HTTP, renderuje odpowiednie strony HTML oraz wprowadza logikę backendową.
- **Interfejs Użytkownika:** Strona internetowa z formularzem pozwala użytkownikowi na wprowadzenie danych wejściowych.

2.1 Schemat Działania

1. Użytkownik wchodzi na stronę główną aplikacji.
2. Wprowadza parametry: dolną i górną granicę, liczbę populacji, liczbę pokoleń oraz wskaźnik mutacji.
3. Po przesłaniu danych formularza, serwer Flask uruchamia algorytm genetyczny z wprowadzonymi parametrami.
4. Wynik jest obliczany przez algorytm genetyczny, który szuka optymalnej wartości x maksymalizującej $f(x) = x \cdot \sin(x)$.
5. Wynik, czyli najlepsza wartość x oraz odpowiadająca jej wartość funkcji $f(x)$, zostaje wyświetlony na stronie jako rezultat.

3 Kod aplikacji

3.1 Kod Python: Algorytm Genetyczny i Serwer Flask

Listing 1: Kod Python

```

1 from flask import Flask, render_template, request
2 import numpy as np
3 import random
4
5 app = Flask(__name__)
6
7 # Funkcja do maksymalizacji
8 def target_function(x):
9     return x * np.sin(x)
10
11 # Algorytm Genetyczny
12 def genetic_algorithm(lower_bound, upper_bound, population_size,
13                       generations, mutation_rate):
14     def fitness(individual):

```

```

14         return target_function(individual)
15
16     def create_population(size):
17         return np.random.uniform(lower_bound, upper_bound, size)
18
19     def select_parents(population, fitnesses):
20         tournament_size = 3
21         selected = []
22         for _ in range(2):
23             tournament = random.sample(range(len(population)),
24                                         tournament_size)
25             best = max(tournament, key=lambda idx: fitnesses[idx])
26             selected.append(population[best])
27         return selected
28
29     def crossover(parent1, parent2):
30         alpha = random.random()
31         child1 = alpha * parent1 + (1 - alpha) * parent2
32         child2 = alpha * parent2 + (1 - alpha) * parent1
33         return child1, child2
34
35     def mutate(individual):
36         if random.random() < mutation_rate:
37             mutation = np.random.uniform(-1, 1)
38             individual = np.clip(individual + mutation, lower_bound,
39                                 upper_bound)
40         return individual
41
42     population = create_population(population_size)
43     for generation in range(generations):
44         fitnesses = np.array([fitness(ind) for ind in population])
45         new_population = []
46         while len(new_population) < population_size:
47             parent1, parent2 = select_parents(population, fitnesses)
48             child1, child2 = crossover(parent1, parent2)
49             new_population.extend([mutate(child1), mutate(child2)])
50         population = np.array(new_population[:population_size])
51
52     best_idx = np.argmax([fitness(ind) for ind in population])
53     return population[best_idx], fitness(population[best_idx])
54
55 # Strona g wna
56 @app.route("/", methods=["GET", "POST"])
57 def index():
58     result = None
59     if request.method == "POST":
60         try:
61             lower_bound = float(request.form["lower_bound"])
62             upper_bound = float(request.form["upper_bound"])
63             population_size = int(request.form["population_size"])
64             generations = int(request.form["generations"])
65             mutation_rate = float(request.form["mutation_rate"])
66
67             # Uruchomienie algorytmu genetycznego
68             best_x, best_fitness = genetic_algorithm(
69                 lower_bound, upper_bound, population_size, generations,
70                 mutation_rate
71             )

```

```

69         result = {
70             "best_x": round(best_x, 4),
71             "best_fitness": round(best_fitness, 4)
72         }
73     except ValueError:
74         result = {"error": "Podano nieprawid owe dane. Spr buj
75                     ponownie."}
76
77     return render_template("index.html", result=result)
78
79 if __name__ == "__main__":
80     app.run(debug=True)

```

3.2 Kod HTML: Interfejs Użytkownika

Listing 2: Kod HTML

```

1 <!DOCTYPE html>
2 <html lang="pl">
3 <head>
4     <meta charset="UTF-8">
5     <meta name="viewport" content="width=device-width, initial-scale
6         =1.0">
7     <title>Algorytm Genetyczny</title>
8     <link rel="stylesheet" href="{{ url_for('static', filename='style.
9         css') }}">
10 </head>
11 <body>
12     <h1>Algorytm Genetyczny: Maksymalizacja Funkcji Matematycznej</h1>
13     <form method="POST">
14         <label for="lower_bound">Dolna granica:</label>
15         <input type="number" step="0.01" id="lower_bound" name="
16             lower_bound" required>
17         <label for="upper_bound">G rna granica:</label>
18         <input type="number" step="0.01" id="upper_bound" name="
19             upper_bound" required>
20         <label for="population_size">Rozmiar populacji:</label>
21         <input type="number" id="population_size" name="population_size
22             " required>
23         <label for="generations">Liczba pokole :</label>
24         <input type="number" id="generations" name="generations"
25             required>
26         <label for="mutation_rate">Wska nik mutacji (0-1):</label>
27         <input type="number" step="0.01" id="mutation_rate" name="
28             mutation_rate" required>
29         <button type="submit">Uruchom algorytm</button>
30     </form>
31
32     {% if result %}
33         {% if result.error %}
34             <p style="color: red;">{{ result.error }}</p>
35         {% else %}
36             <h2>Wyniki:</h2>
37             <p><strong>Najlepsze x:</strong> {{ result.best_x }}</p>
38             <p><strong>Maksymalna warto   funkcji:</strong> {{ result
39                 .best_fitness }}</p>
40         {% endif %}

```

```
33     {% endif %}  
34 </body>  
35 </html>
```

3.3 Kod CSS: Stylizacja Strony

Listing 3: Kod CSS

```
1 body {  
2     font-family: Arial, sans-serif;  
3     background-color: #12dac9;  
4     color: #333;  
5     text-align: center;  
6 }  
7  
8 h1 {  
9     color: #e6242e;  
10    font-size: 24px;  
11    margin-bottom: 10px;  
12 }  
13  
14 form {  
15     margin-top: 20px;  
16     display: inline-block;  
17     text-align: left;  
18 }  
19  
20 form label {  
21     display: block;  
22     margin: 10px 0 5px;  
23 }  
24  
25 form input, form button {  
26     padding: 10px;  
27     margin: 5px 0;  
28     font-size: 16px;  
29 }  
30  
31 form button {  
32     background-color: #bb0c15;  
33     color: white;  
34     border: none;  
35     cursor: pointer;  
36 }  
37  
38 form button:hover {  
39     background-color: #3a8b47;  
40 }
```

4 Zrzuty ekranu z aplikacji

Algorytm Genetyczny: Maksymalizacja Funkcji Matematycznej

Maksimum funkcji:

$f(x) = x \cdot \sin(x)$, gdzie $x \in [0, 10]$

Dolna granica:

Górna granica:

Rozmiar populacji:

Liczba pokoleń:

Wskaźnik mutacji (0-1):

Algorytm Genetyczny: Maksymalizacja Funkcji Matematycznej

Maksimum funkcji:

$f(x) = x \cdot \sin(x)$, gdzie $x \in [0, 10]$

Dolna granica:

Górna granica:

Rozmiar populacji:

Liczba pokoleń:

Wskaźnik mutacji (0-1):

Podano nieprawidłowe dane. Spróbuj ponownie.

Algorytm Genetyczny: Maksymalizacja Funkcji Matematycznej

Maksimum funkcji:

$f(x) = x \cdot \sin(x)$, gdzie $x \in [0, 10]$

Dolna granica:

Górna granica:

Rozmiar populacji:

Liczba pokoleń:

Wskaźnik mutacji (0-1):

Uruchom algorytm

Wyniki:

Najlepsze x: 0.8

Maksymalna wartość funkcji: 0.5739