Bartosz Jurczewski 234067 234067@edu.p.lodz.pl
Karol Podlewski 234106 234106@edu.p.lodz.pl

Zadanie 1: Optymalizacja funkcji wielu zmiennych

1. Cel

Celem zadania była optymalizacja funkcji wielu zmiennych na zadanym przedziale. Dla potrzeby opracowanego rozwiązania przyjęliśmy liczbę zmiennych jako 2.

2. Opis implementacji

Zadanie zostało zrealizowane przy użyciu frameworka *.NET Core* w wersji 3.1, języka *C#* , z wykorzystaniem bibliotek: *GeneticSharp*[1].

3. Wprowadzenie

3.1. Algorytm

W przygotowanym rozwiązaniu użytkownik może wybrać jeden z dwóch algorytmów. **Reguła 1/5 sukcesu** dokłada do **SGA** zmienny krok mutacji.

1. **Simple Genetic Algorithm, SGA** - bazowy algorytm genetyczny
2. **Reguła 1/5 sukcesu** - Jeśli przez k kolejnych generacji więcej niż 1/5 mutacji kończy się sukcesem, należy zwiększyć krok mutacji. W przeciwnym wypadku krok mutacji należy zmniejszyć.

3.2. Krzyżowanie

W przygotowanym rozwiązaniu użytkownik może wybrać jedną z dwóch metod krzyżowania:

1. **Uniform Crossover** - każdy bit jest losowo wybierany od jednego z dwóch rodziców z równym prawdopodobieństwem,
2. **Three Parent Crossover** - selekcja, która polega na wybraniu trzech losowych rodziców. Po kolei porównywany jest każdy bit pierwszego oraz drugiego rodzica - jeżeli bit jest identyczny, przechodzi on do potomstwa, jeżeli się różni - brany jest bit od trzeciego rodzica.

3.3. Mutacja

W przygotowanym rozwiązaniu użytkownik może wybrać jedną z dwóch metod mutacji:

1. **Flip Bit Mutation** - odwrócenie wartości wybranego genu (zamiana 0 na 1, 1 na 0),
2. **Reverse Sequence Mutation** - bierzemy sekwencję S ograniczoną przez dwie losowo wybrane pozycje i i j, takie że $i < j$, którą odwracamy.

3.4. Selekcja

W przygotowanym rozwiązaniu użytkownik może wybrać jedną z dwóch metod selekcji:

1. **Elite Selection** - wybiera najlepsze chromosomy pod względem dopasowania,
2. **Roulette Wheel Selection** - metoda koła ruletki, która polega na przypisaniu każdemu osobnikowi prawdopodobieństwa selekcji (im większe przystosowanie, tym większe prawdopodobieństwo), a następnie wylosowaniu chromosomów z całej puli.

3.5. Funkcje celu

W przygotowanym rozwiązaniu użytkownik może wybrać jedną z dwóch funkcji celu:

1. $y = sin(x_1)cos(x_2)$,
2. $y = sin(x_1)cos(x_2)x_1 - sin(x_2)cos(x_1)$

4. Wyniki

Dla każdego wariantu zostały wykonana seria przynajmniej 10 pomiarów. Z uzyskanych wyników usuwano te, które wpadły w wyjątkowo niekorzystne ekstrema lokalne. Wartość dopasowania jest średnią z tych pomiarów.

4.1. Wpływ mutacji i sposobu krzyżowania

Argumenty stałe dla tej sekcji badań:
— **Warunek stopu:** 1000 epok,
— **Funkcja celu:** $y = sin(x_1)cos(x_2)$,
— **Wartość minimalna:** -100,
— **Wartość maksymalna:** 100,
— **Wielkość populacji:** 50.

	Krzyżowanie	**Mutacja**	**Selekcja**	**Dopasowanie**
1	Uniform	Flip Bit	Elite	0,831
2	Uniform	Flip Bit	Roulette Wheel	0,862
3	Uniform	Reverse Sequence	Elite	0,963
4	Uniform	Reverse Sequence	Roulette Wheel	0,872
5	Three Parent	Flip Bit	Elite	0,981
6	Three Parent	Flip Bit	Roulette Wheel	0,813
7	Three Parent	Reverse Sequence	Elite	0,971
8	Three Parent	Reverse Sequence	Roulette Wheel	0,825

Tabela 1. Porównanie wpływu krzyżowania, mutacji oraz selekcji dla algorytmu SGA

	Krzyżowanie	Mutacja	Selekcja	Dopasowanie
9	Uniform	Flip Bit	Elite	0,844
10	Uniform	Flip Bit	Roulette Wheel	0,827
11	Uniform	Reverse Sequence	Elite	0,968
12	Uniform	Reverse Sequence	Roulette Wheel	0,951
13	Three Parent	Flip Bit	Elite	0,983
14	Three Parent	Flip Bit	Roulette Wheel	0,884
15	Three Parent	Reverse Sequence	Elite	0,991
16	Three Parent	Reverse Sequence	Roulette Wheel	0,899

Tabela 2. Porównanie wpływu krzyżowania, mutacji oraz selekcji dla algorytmu One-fifth-rule

Dyskusja

Użycie krzyżowania *Uniform* negatywnie wpłynęło na dopasowanie - niezależnie od algorytmu czy mutacji. Podobnie miała się sprawa z selekcją ruletki, która poza słabszymi wynikami, miała wyjątkową tendencję do wpadania w niekorzystne ekstrema lokalne. Wariantem o najwyższym (czyli najlepszym) wyniku dopasowania dla algorytmu *One-fifth-tule* okazał się wariant z krzyżowaniem *Three Parent*, mutacją *Reverse Sequence*, oraz selekcją *Elite*. Dla algorytmu *SGA* najlepszym wariantem okazał się ten z krzyżowaniem *Three Parent*, mutacją *Flip Bit* oraz selekcją *Elite*. To właśnie te parametry wykorzystamy w dalszej części badań.

4.2. Porównanie działania dla różnych funkcji celu

Argumenty stałe dla tej sekcji badań:
— **Krzyżowanie:** Three Parent Crossover,
— **Selekcja:** Elite Selection,
— **Warunek stopu:** 1000 epok,
— **Wartość minimalna:** -100,
— **Wartość maksymalna:** 100,
— **Wielkość populacji:** 50.

	Algorytm	Mutacja	Dopasowanie
17	SGA	Flip Bit	0,981
18	One-fifth-rule	Reverse Sequence	0,991

Tabela 3. Porównanie wyników dla pierwszej funkcji celu

	Algorytm	Mutacja	Dopasowanie
19	SGA	Flip Bit	82,946
20	One-fifth-rule	Reverse Sequence	84,344

Tabela 4. Porównanie wyników dla drugiej funkcji celu

Dyskusja

Dla obu porównywanych przez nas funkcji celu algorytm **One-fifth-rule** wykazał lepsze średnie dopasowanie od SGA. To właśnie ten algorytm wykorzystamy w ostatniej części badań.

4.3. Porównanie działania w zależności od jego parametrów

Argumenty stałe dla tej sekcji badań:
— **Algorytm:** One-fifth-rule,
— **Krzyżowanie:** Three Parent Crossover,
— **Mutacja:** Reverse Sequence Mutation,
— **Selekcja:** Elite Selection,
— **Funkcja celu:** $y = sin(x_1)cos(x_2)$,
— **Wartość minimalna:** -100,
— **Wartość maksymalna:** 100.

	Populacja	**Liczba epok**	**Dopasowanie**
21	25	500	0,925
22	25	1000	0,927
23	25	2000	0,98
23	50	500	0,978
25	50	1000	0,991
26	50	2000	0,985
27	100	500	0,982
28	100	1000	0,99
29	100	2000	0,992

Tabela 5. Porównanie dopasowania w zależności od populacji oraz liczby epok

Dyskusja

Zmiana populacji oraz liczby epok w znaczącym stopniu nie wpływa na wynik dopasowania - jedynie zmniejszenie populacji wpływa na zauważalne pogorszenie wyników. Zwiększenie liczby epok nie koniecznie musi oznaczać lepsze dopasowanie.

5. Wnioski

— Bardzo ważne jest dopranie odpowiedniego sposobu krzyżowania, mutacji oraz selekcji, przez co możemy poprawić wyniki algorytmu.
— Dodatkowe modyfikacje algorytmu w czasie jego działania, takie jak reguła 1/5 sukcesu, pozytywnie wpływają na algorytm optymalizacyjny.
— Nie ma potrzeby dopierania nieskończenie dużej wielkości populacji czy liczby epok potrzebnych do ukończenia procesu optymalizacyjnego, gdyż nie wpłynie to znacząco na wyniki, a może wydłużyć czas działania algorytmu.

Literatura

[1] *GeneticSharp*, https://github.com/giacomelli/GeneticSharp
[2] *Function optimization with GeneticSharp*, http://diegogiacomelli.com.br/function-optimization-with-geneticsharp/