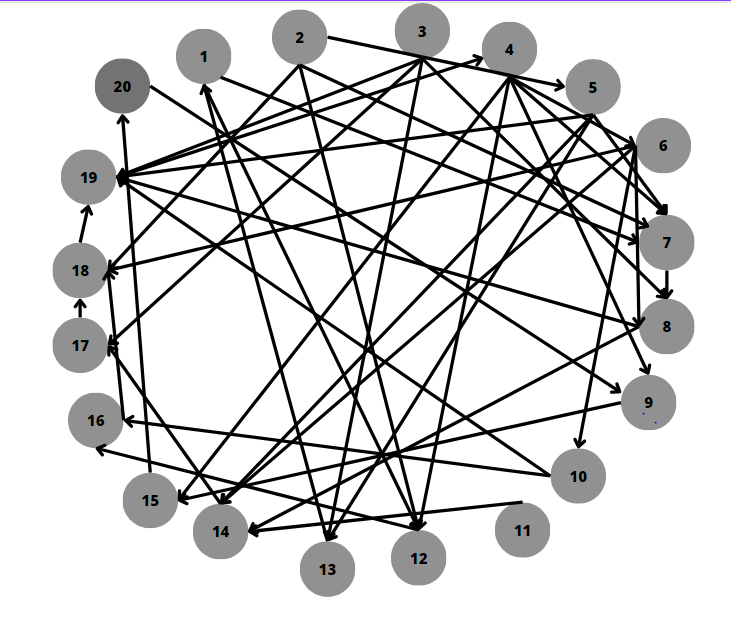
**Kolorowanie grafów**

Jakub Gośliński 154012

Aleksander Karczewski 154373

**Algorytm genetyczny (metaheurystyka)**

1. **Inicjalizacja**

****

2 .**Opis Algorytmu**

Algorytm genetyczny działa na populacji potencjalnych rozwiązań, które reprezentowane są przez osobniki. Każdy osobnik odpowiada jednemu potencjalnemu rozwiązaniu kolorowania grafu. Na początku algorytmu tworzona jest początkowa populacja, na przykład losowo generowana.

Następnie algorytm ewoluuje poprzez iteracyjne zastosowanie czterech podstawowych operatorów genetycznych: selekcji, krzyżowania, mutacji i oceny. Selekcja polega na wyborze najlepszych osobników z populacji, które mają większe szanse na przekazanie swoich genów do następnej generacji. Krzyżowanie polega na wymianie informacji genetycznej między wybranymi osobnikami, co prowadzi do powstania nowych osobników potomnych. Mutacja polega na wprowadzaniu losowych zmian w genotypie pewnych osobników w celu zachowania różnorodności genetycznej populacji. Ocena polega na przypisaniu każdemu osobnikowi oceny jakości kolorowania grafu na podstawie pewnych kryteriów, na przykład liczby konfliktów kolorów.

Algorytm kontynuuje iteracyjne zastosowanie tych operatorów genetycznych przez pewną liczbę generacji lub do momentu osiągnięcia satysfakcjonującego wyniku. Celem jest znalezienie osobnika, który ma najmniejszą liczbę konfliktów kolorów i zapewnia poprawne kolorowanie grafu.

**Pseudokod**

Funkcja geneticAlgorithm:

1. Inicjalizuj populację początkową o rozmiarze populationSize

Dla każdej generacji od 1 do numGenerations wykonuj:

1. Oblicz koszt (funkcję oceny) dla każdego rozwiązania w populacji

2. Posortuj populację według kosztu (od najmniejszego do największego)

3. Wybierz najlepsze rozwiązanie jako bestSolution

4. Stwórz nową populację

Dopóki nowa populacja nie osiągnie rozmiaru populationSize:

1 Wybierz dwa rozwiązania rodzicielskie (np. poprzez turniej dwóch rozwiązań)

2. Wykonaj krzyżowanie rodziców, tworząc nowe rozwiązanie potomne

3. Wykonaj mutację na nowym rozwiązaniu potomnym

4. Dodaj nowe rozwiązanie potomne do nowej populacji

5. Zaktualizuj populację: population = nowa populacja

6. Zwróć najlepsze rozwiązanie: bestSolution

Funkcja oceny:

1. Inicjalizuj koszt na 0

2. Oblicz stopień wierzchołków na podstawie macierzy podobieństwa i posortuj malejąco

3. Przechodząc po kolejnych wierzchołkach koloruj je, ograniczając możliwość wystąpienia konfliktów

4. Zwróć koszt

Funkcja krzyżowania:

1. Wybierz punkt krzyżowania losowo

2. Stwórz nowe rozwiązanie potomne

3. Skopiuj kolory z rodzica pierwszego do punktu krzyżowania

4. Skopiuj kolory z rodzica drugiego od punktu krzyżowania

5. Zwróć nowe rozwiązanie potomne

Funkcja mutacji

1 Wybierz losowo jeden wierzchołek

2 Wybierz nowy kolor dla wierzchołka, który jest różny od aktualnego

3 Zaktualizuj kolor wierzchołka

Wywołanie głównego programu:

1. Stwórz graf na podstawie danych wejściowych

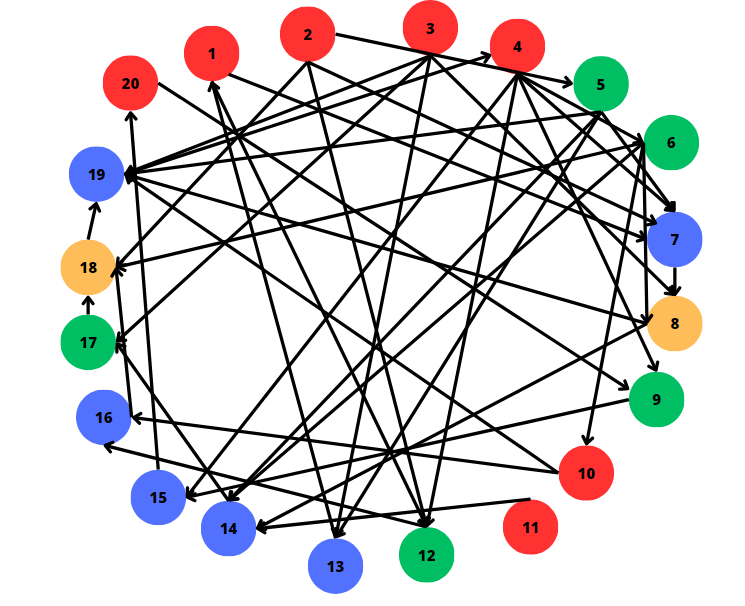
2. Wywołaj genethicAlgorithm

3. Wyświetl najlepsze rozwiązanie (kolorowanie grafu) i jego koszt

**Przykład obrazujący działanie**

Najlepiej 2, 3 rysunki przedstawiające obraz instancji po najważniejszych krokach algorytmu. **UWAGA!** Przykład z tą samą instancją jak w pkt.1

**Finalizacja**



**Wykresy**

1. Porównaj optymalizowaną wartość (wynik) Algorytmu z A. zachłannym.

(wykres pokaże czy Algorytm jest lepszy i o ile od A. zach.  
Instancje wygenerowane losowe, 15 punktów pomiarowych)

Dane 1\_1 oznaczają 100 wierzchołków/ 100 połączeń między nimi itd. Zbliżone wyniki prawdobodnie są wynikiem małego zakresu generowanych danych.

1. Korzystając z bibliotek instancji (benchmarków) pokaż na wykresie wartość błędu względnego Algorytmu w stosunku do wartości optymalnej.
2. Zamieść **swoje** wyniki w Tabelce z **Rankingu**, instancje rankingowe są w katalogu <http://www.cs.put.poznan.pl/mmachowiak/instances/>

| Queen6 | 10 |
| --- | --- |
| Miles250 | 9 |
| Gc500 | 85 |
| Gc1000 | 151 |
| Le450\_5a | 13 |