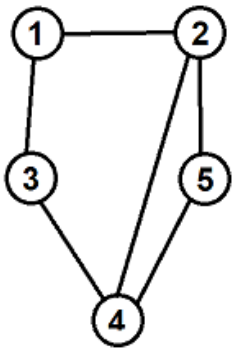
Kolorowanie grafu

Szymon Szymankiewicz 151821
Alicja Lis 151569

Algorytm Genetyczny

1. **Inicjalizacja**

Przykładowy graf nieskierowany



2. **Opis Algorytmu**

Algorytm genetyczny opiera się na procesie **doboru naturalnego**, ma podstawowe właściwości **naturalnej selekcji** i stosuje je do problemu, który próbujemy rozwiązać. Algorytm usuwa słabe jednostki z populacji i tworzy nową, silniejszą. Celem uniknięcia lokalnego optimum stosuje się **mutację**. Aby tworzyć coraz lepszą populację algorytm używa:

a. **Inicjalizacja** - stworzenie początkowej populacji. W naszym przypadku jest ona generowana losowo. Rozmiar populacji może być dowolny, podany jako parametr przy tworzeniu.
b. **Ocena (fitness)** - jest to miara jakości dla każdego osobnika w populacji. Osobniki z wyższą oceną (w naszym programie z większą wartością fitness) są bardziej prawdopodobne do wybrania do kolejnych kroków w algorytmie, takich jak krzyżowanie, reprodukcja i mutacja, a tym samym mają większe prawdopodobieństwo do bycia rozwiązaniem optymalnym.
c. **Krzyżowanie (crossover)** - łączenie dwóch rozwiązań w celu stworzenia nowych, które potencjalnie mogą okazać się lepsze. Dla naszego algorytmu krzyżowanie polega na wymieszaniu informacji o kolorach wierzchołków pomiędzy dwoma chromosomami.

Dzięki krzyżowaniu, nowe rozwiązania mogą powstać ze złożenia optymalnych połączeń różnych rozwiązań.

d. **Mutacja (mutation)** - Mutacja jest procesem losowych zmian chromosomu w algorytmie. Celem mutacji jest zwiększenie różnorodności populacji i zwiększenie szansy na znalezienie lepszego rozwiązania. Dzięki temu również, nowe rozwiązania mogą powstać z różnych kombinacji kolorów, co zwiększa szansę na znalezienie lepszego rozwiązania.

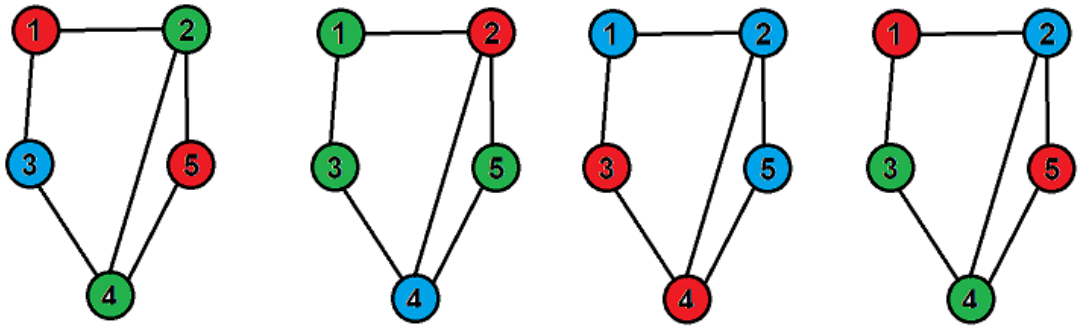
Algorytm wykonuje się tak długo, aż nie zostanie osiągnięty określony warunek stopu. Warunek stopu jaki zaimplementowaliśmy to przekroczenie liczby iteracji (generacji), która jest ustalona przed uruchomieniem programu. Im większa ilość iteracji, tym jest większy potencjał na znalezienie lepszego rozwiązania, co wiąże się z dłuższym czasem wykonania się programu.

3. **Pseudokod**

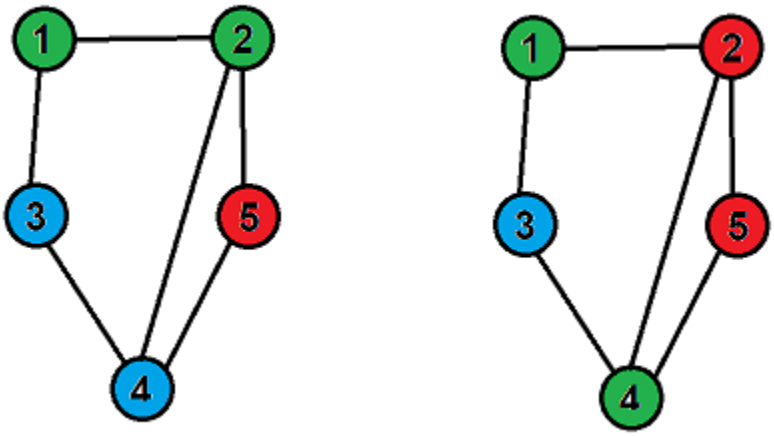
```
funkcja rozwiązanie_algorytmu_genetycznego(rozwiązanie, liczba_iteracji):
        populacja = generuj_populację(); //losowo

        oceń_populację(); // tworzy listę zaczynają od osobniką z najlepszą
oceną fitness

        dopóki liczba_iteracji:
                najlepsze_rozwiązania = znajdź_najlepsze_rozwiązania()
//wybierany według fitness (% najlepszych wyników jest ustalony)
                rodzice = wybierz_rodziców() // część populacji która
będzie produkowała potomstwo
                potomstwo = stwórz_potomstwo(rodzice)
                populacja = najlepsze_rozwiązania + potomstwo
                oceń_populację()
                jeżeli najlepszy_wynik == 0 zwróć prawda
                liczba_iteracji--
        zwróć fałsz

funkcja wybierz_rodziców():
        rodzice = rozmiar_populacji(%rodzice)
        dopóki rodzice--:
                dodaj_rodziców(losuj(0,n4)) // od 0 do n4
        zwróć rodzice

funkcja stwórz_potomstwo():
        potomstwo
        dopóki rozmiar_populacji(%z_mutacji)--:
                dodaj_podomstwo(mutuj(losuj(rodzic))

        dopóki rozmiar_populacji*(%z_crossover)--:
                a = 2_losowi_rodzice;
                b = 2_losowi_rodzice;
                dodaj_potmstwo(crossover(lepszy(a),lepszy(b))
                zwróć potomstwo

funkcja main:
        rozwiązanie = zachłanny(graf)-1
        liczba_iteracji //podana
        dopóki rozwiązanie_algorytmu_genetycznego(rozwiązanie--,liczba_iteracji)
==prawda
```

4. **Przykład obrazujący działanie**

Inicjalizacja przykładowej populacji i nadanie każdemu osobnikowi oceny (wartości fitness)



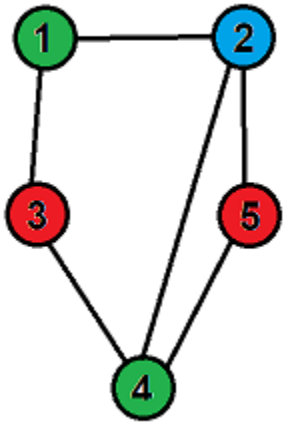
Mutacja kolejnego pokolenia z dwóch najbardziej odpowiednich kandydatów (1 i 2 - największa wartość fitness)



Po tym następuje powtórzenie kroku i wybranie najlepszych kandydatów na rodziców kolejnego pokolenia

5. **Finalizacja**

Ostateczny wynik podany przez algorytm genetyczny:



Wykresy

1. Porównanie algorytmu gentycznego z algorytmem zachłannym

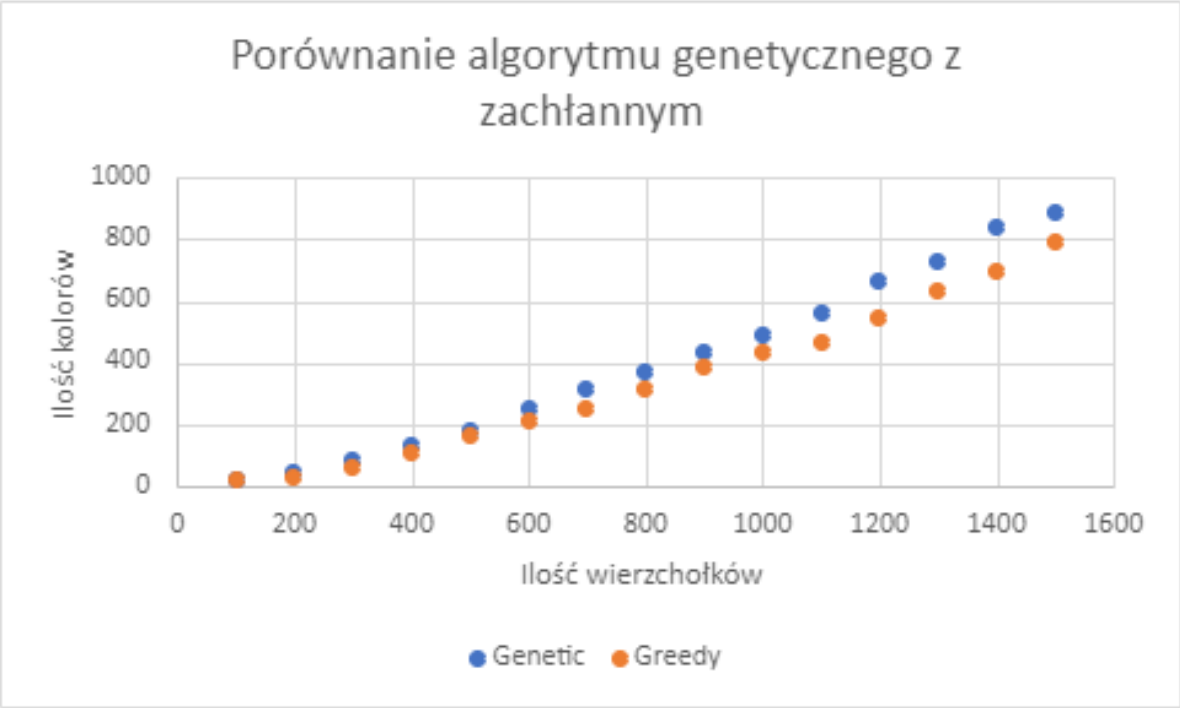
W tym celu użyte zostało 15 instancji grafu nieskierowanego wygenerowanych losowo ze stałym nasyceniem - 50%. Ilość wierzchołków w grafie rośnie co 100 zaczynając od 100 wierzchołków. Wyniki zostały umieszczone w tabelach, a następnie porównane na wykresie.

Algorytm genetyczny:

Ilość wierzchołków	Ilość kolorów
100	20
200	45
300	85
400	134
500	184
600	252
700	318
800	374
900	435
1000	515
1100	563
1200	663
1300	727
1400	836
1500	883

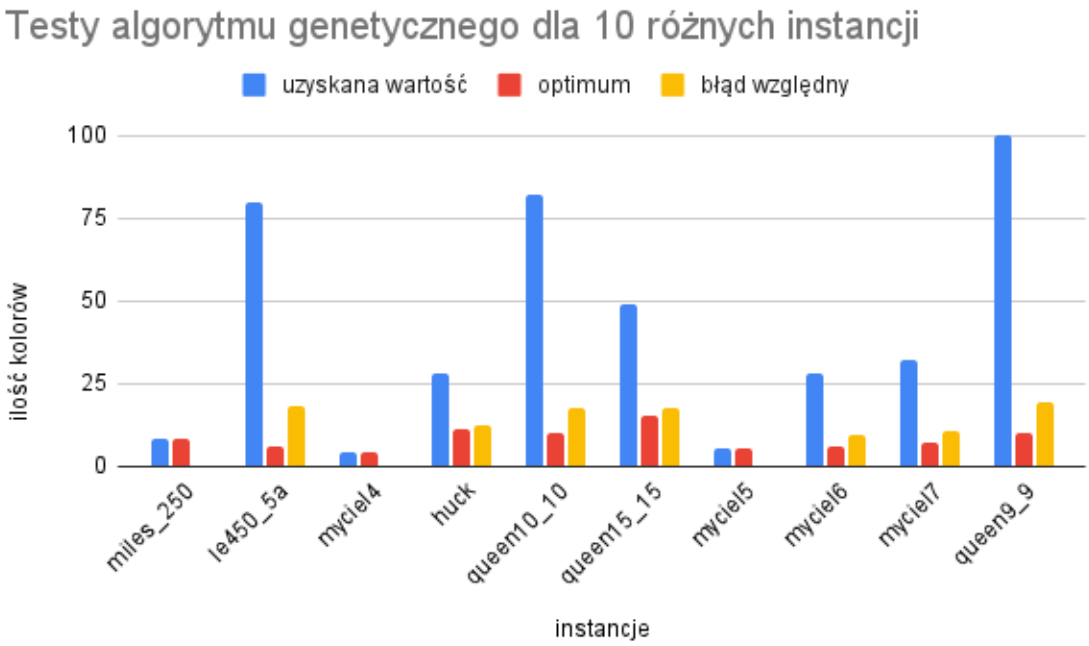
Algorytm zachłanny:

Ilość wierzchołków	**Ilość kolorów**
100	21
200	33
300	65
400	109
500	165
600	212
700	256
800	312
900	389
1000	433
1100	469
1200	546
1300	634
1400	695
1500	789



Jak można zauważyć algorytm genetyczny i algorytm zachłanny w przypadku instancji o dużym nasyceniu wykazały porównywalną ilość kolorów, która jak można zauważyć rośnie kwadratowo z każdym zwiększeniem ilości wierzchołków. Algorytm zachłanny radzi sobie lepiej niż algorytm genetyczny co można zauważyć przy większej ilości wierzchołków. Algorytm zachłanny jest zwykle uważany za lepszy niż algorytm genetyczny w przypadku kolorowania grafu, ponieważ jest on bardziej skupiony na lokalnym optymalizowaniu rozwiązania niż na globalnym. Jest to ważne, ponieważ kolorowanie grafu jest problemem NP-trudnym i trudno jest znaleźć globalnie optymalne rozwiązanie.

2. **Wartość błędu względnego algorytmu genetycznego w stosunku do wartości optymalnej**



3. **Ranking instacji**

Instancja 1 **queen6**	**Wynik rankingowy** **7**	**Wynik uzyskany** **7** **(1500 osobników, 1000 iteracji)**
Instancja 2 **miles250**	**Wynik rankingowy** **8**	**Wynik uzyskany** **8** **(150 osobników, 1000 iteracji)**
Instancja 3 **gc500**	**Wynik rankingowy** **127**	**Wynik uzyskany** **215** **(1000 osobników, 1000 iteracji)**
Instancja 4 **le450_5a**	**Wynik rankingowy** **6**	**Wynik uzyskany** **30** **(2000 osobników, 1000 iteracji)**