Chromatyczna Teoria Grafów

Dokumentacja końcowa

Temat: (15). Zaprojektować algorytm kolorujący graf którego wierzchołki są dane na płaszczyźnie, a krawędź jest pomiędzy wierzchołkami w odległości geometrycznej między 1 a 2, wykorzystujący geometryczne własności grafu. Porównać z algorytmem DSatur i GIS.

Opracowali: Jan Serwatka i Jarosław Rolski

serwatkaj@student.mini.pw.edu.pl

j.rolski@student.mini.pw.edu.pl

Spis treści

1 Sposób działania programu

1.1 Obsługa programu

Po uruchomieniu programu otworzy się okno z pustym wykresem, oraz przyciskami:

1. `'New Random Graph'`
   Tworzy graf losowy i przedstawia go na wykresie.
2. `'DSatur'`
   Koloruje stworzony wcześniej graf według algorytmu DSatur.
3. `'GIS'`
   Koloruje stworzony wcześniej graf według algorytmu GIS.
4. `'GISbis'`
   Koloruje stworzony wcześniej graf według algorytmu GISbis.
5. `'TURBOColor3000'`
   Koloruje stworzony wcześniej graf według algorytmu TURBOColor3000.
6. `'Change Colors'`
   Zmienia kolory podstawiane pod liczby przypisane wierzchołkom przez powyższe algorytmy.

   (Kolory są dobierane losowo, więc zdarza się, że różne kolory są na rysunku nieodróżnialne, wtedy należy nacisnąć przycisk 'Change Colors'.)

Użytkownik może również podać ilość wierzchołków grafu oraz wymiary powierzchni, na której mają być one losowane, służądo tego pola:

1. `'No. vertices'` - liczba wierzchołków
2. `'yrange'` - wysokość płaszczyzny
3. `'xrange'` - szerokość płaszczyzny

Domyślna liczba wierzchołków to 100, domyślne wymiary to 10x10.

1.2 Pseudokody algorytmów

1.2.1 DSatur

Algorithm 1 DSatur

Znajdź wierzchołek o największym stopniu.
while istnieją niepokolorowane wierzchołki:
    Znajdź niepokolorowany wierzchołek taki, że liczba jego pokolorowanych sąsiadów jest największa i pokoloruj go na najmniejszy dostępny kolor.

1.2.2 GIS, GISbis, TURBOColor3000 - funkcja główna

Poniższe algorytmy działają według następującego kodu, różnią natomiast się tym jak wygląda funkcja `independent`.

Algorithm 2 Funkcja główna

$ind :=$ `independent`(G)
$i := 0$
while $G \neq (\emptyset, \emptyset)$:
    Pokoloruj ind na i-ty kolor.
$G := G \setminus ind$
$i := i + 1$
$ind :=$ `independent`(G)

(G oznacza kolorowany graf.)

1.2.3 GIS - funkcja `independent`

Algorithm 3 `independent`

if $|V(G)| = 0$:
    return $\emptyset$
Znajdź wierzchołek v o najmniejszym stopniu.
$ind :=$ `independent`$(G \setminus (N(v) \cup \{v\})$
return $ind \cup \{v\}$

1.2.4 GISbis - funkcja `independent`

Algorithm 4 `independent`

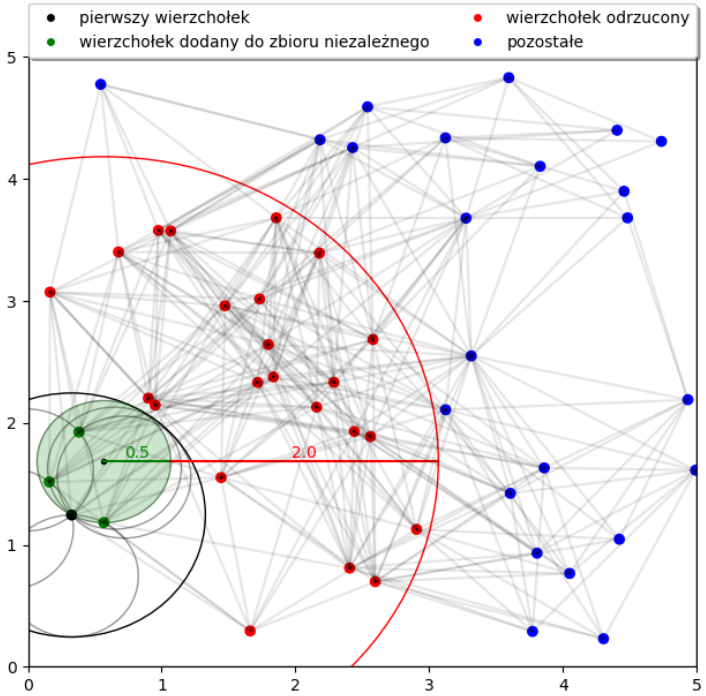
if $|V(G)| = 0$:
    return $\emptyset$
Znajdź wierzchołek v najbliższy punktu $(0, 0)$.
$ind :=$ `independent`$(G \setminus (N(v) \cup \{v\})$
return $ind \cup \{v\}$

1.2.5 TURBOColor3000 - funkcja `independent`

Algorithm 5 `independent`

if $|V(G)| = 0$:
    return $\emptyset$
Znajdź wierzchołek v najbliższy punktu $(0, 0)$.
$Circles := \emptyset$
for $w \in B(v, 1) \cap V(G)$:
    Niech K_1, K_2 będą kołami (domkniętymi) o promieniu 0,5, takimi, że ich brzeg przechodzi przez v i w.
    $Circles := Circles \cup \{K_1, K_2\}$
Niech k będzie takim kołem z $Circles$, które zawiera najwięcej wierzchołków.
Niech K będzie kołem o tym samym środku co k, o promieniu 2,5.
$ind :=$ `independent`$(G \setminus K)$
return $ind \cup (k \cap V(G))$

Następująca grafika przedstawia sposób w jaki szukany jest zbiór niezależny. Na zielono zaznaczone jest koło k z powyższego algorytmu, czerwony okrąg to brzeg koła K, wierzchołki niebieskie tworzą zbiór $G \setminus K$, natomiast wierzchołki czerwone to wierzchołki odrzucone.



2 Plan eksperymentów wraz z otrzymanymi wynikami:

Kolorowania grafów przeprowadziliśmy w dwóch turach. W pierwszej badaliśmy algorytmy `D_Satur, GIS i GISbis` na powierzchni `10x10`, a w drugiej algorytmy `D_Satur, GIS, GISbis i TURBOColor3000` na powierzchni `5x5`. W obu przypadkach przeprowadzaliśmy obliczenia dla zmieniającej się liczby wierzchołków (zagęszczenia wierzchołków), a losowanie położenia wierzchołków wykonaliśmy rozkładem jednostajnym dwuwymiarowym.

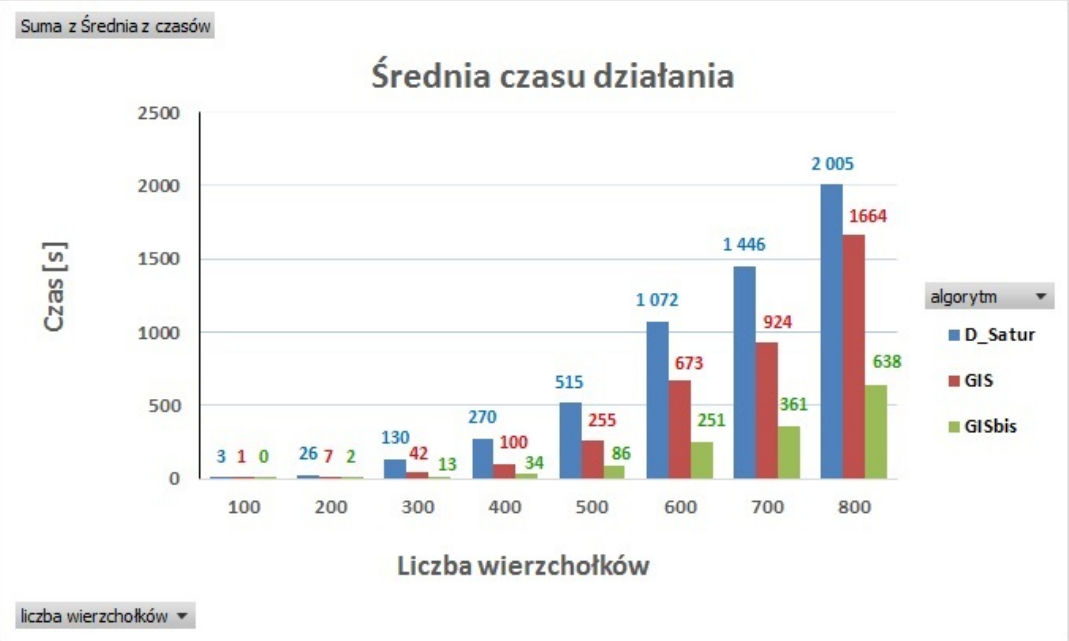
2.1 Pierwsza tura eksperymentów, algorytmy: `D_Satur, GIS i GISbis`.

Eksperymenty przeprowadzaliśmy na powierzchni `10x10` dla liczby wierzchołków od `100` do `800` z krokiem `100` w następujacych etapach:

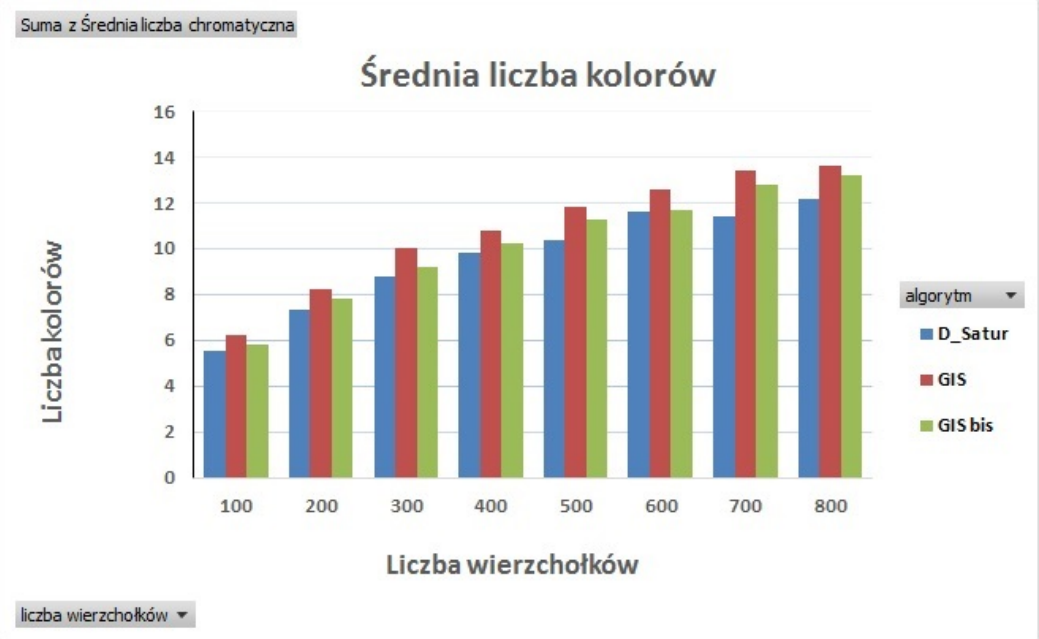
1. dla każdej liczby wierzchołków tworzymy po dziesięć grafów
2. obliczamy stałe chromatyczne wyliczone przy pomocy każdego algorytmu
3. zapisujemy wyliczone stałe i czasy działania algorytmów do bazy danych

Otrzymane dane zprezentowaliśmy w postaci wykresów:

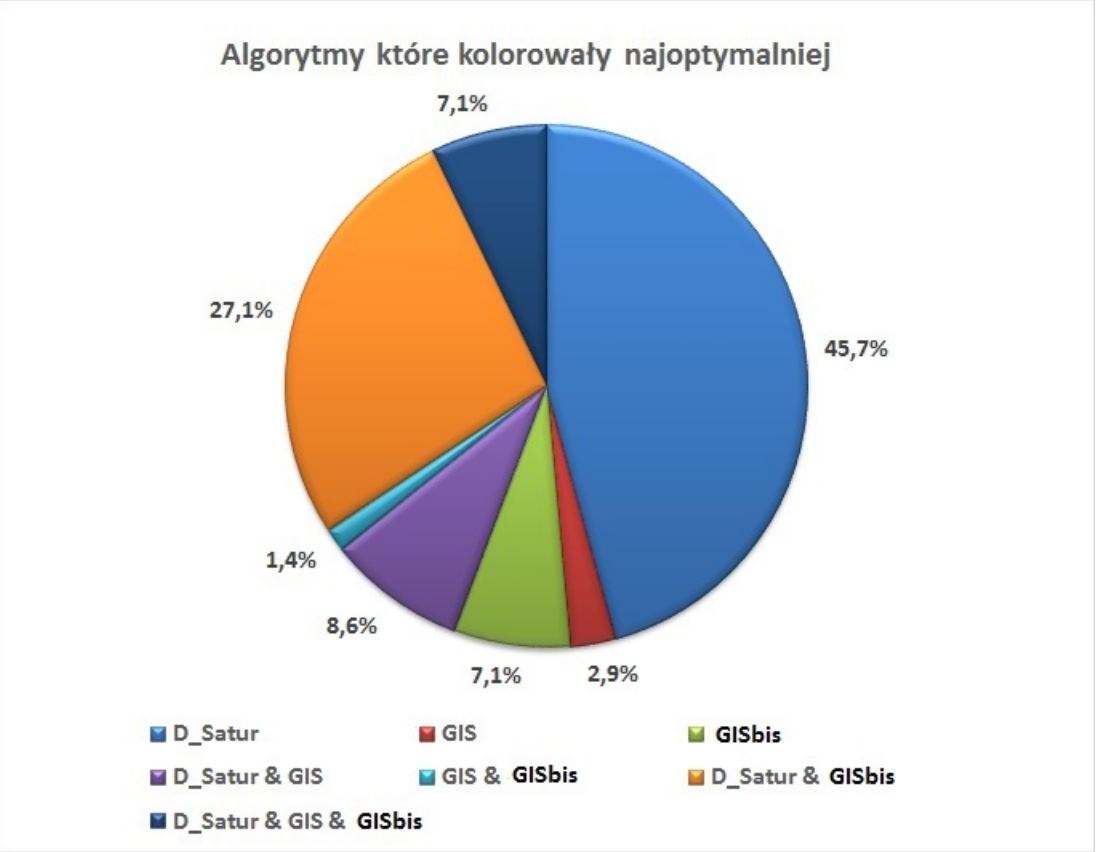
2.1.1 Średnia czasu działania algorytmów dla ustalonej liczby wierzchołków.



2.1.2 Średnia liczba kolorów którymi pomalowały algorytmy dla ustalonej liczby wierzchołków.



2.1.3 Procent optymalnego kolorowania dla algorytmów.



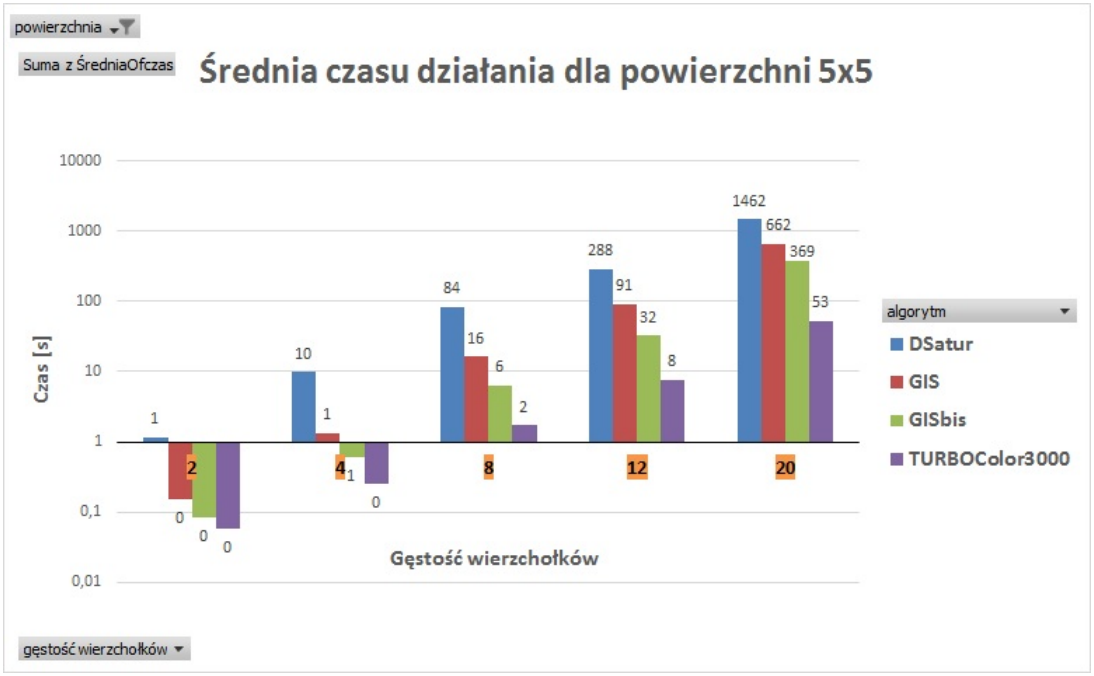
2.2 Druga tura eksperymentów, algorytmy: `D_Satur, GIS, GISbis, TURBOColor3000.`

Eksperymenty przeprowadzaliśmy na powierzchni `5x5` dla gęstości wierzchołków: `2, 4, 8, 12, 20` w następujacych etapach:

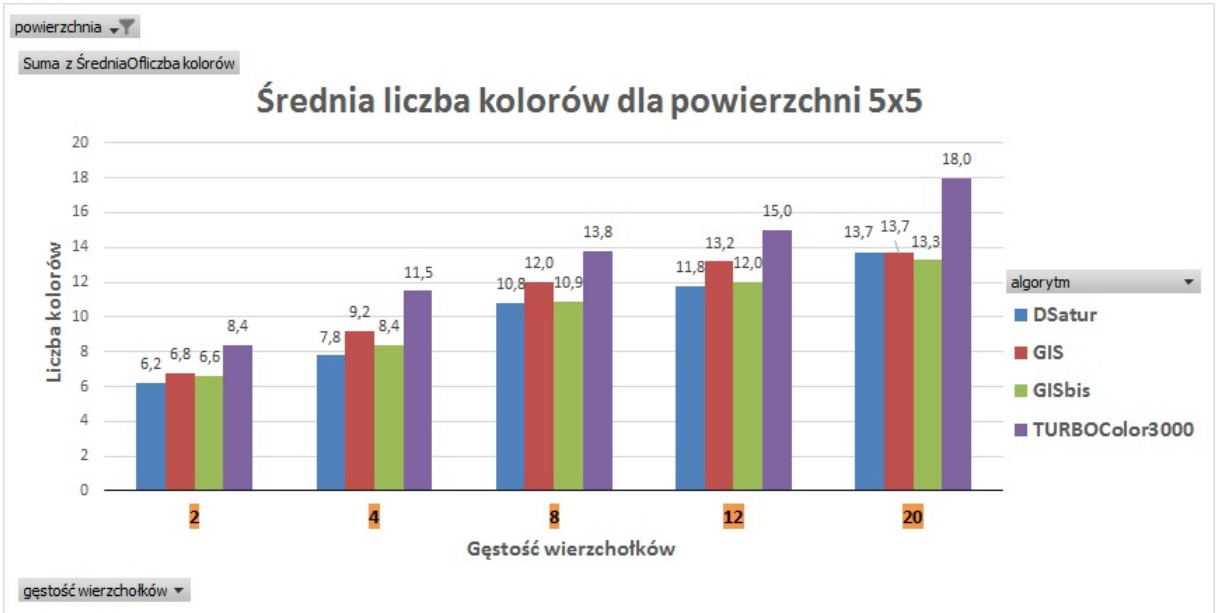
1. dla każdej liczby wierzchołków tworzymy po dziesięć grafów
2. obliczamy stałe chromatyczne wyliczone przy pomocy każdego algorytmu
3. zapisujemy wyliczone stałe i czasy działania algorytmów do bazy danych

Otrzymane dane zprezentowaliśmy w postaci wykresów:

2.2.1 Średnia czasu działania algorytmów dla ustalonej liczby wierzchołków.



2.2.2 Średnia liczba kolorów którymi pomalowały algorytmy dla ustalonej liczby wierzchołków.



3 Analiza wyników i wnioski:

3.1 Analiza wyników:

Z otrzymanych wyników dla pierwszej tury testów możemy zauważyć że średnio biorąc najlepszym algorytmem jest `D_Satur`, który pomalował nie gorzej od innych algorytmów 88,6% badanych przez nas grafów. Niestety wiąże się to z kilkurotnie większym czasem obliczeń niż algorytm `GISbis` który pomalował nie gorzej od innych algorytmów 35,6% badanych grafów. `GIS` najlepiej pomalował tylko 2,9% grafów co jest najgorszym wynikiem z między badanymi algorytmami.
Druga tura eksperymentów potwierdziła zależność czasową między trzema pierwszymi algorytmami, z tym że algorytm `TurboColor3000` okazał się być znacznie szybszy od pozostałych. Algorytm `TurboColor3000` pod względem liczby używanych kolorów okazał się być bardzo nieoptymalny i znacząco odstaje na tle innych algorytmów. Najciekawsze co można zauważyć to że dla dużej gęstości wierzchołków algorytm `GISbis` zaczyna dorównywać średniej liczbie kolorów algorytmowi `D_Satur` a przy gęstości 20 nawet koloruje lepiej od niego.

3.2 Wnioski:

Algorytm `GISbis` który stworzono na podstawie `GIS` gdzie zmodyfikowano wyszukiwanie zbiorów niezależnych na podstawie znanej strutury grafu, pozwoliło na zmniejszenie złożoności obliczeniowej tego algorytmu i zmniejszenie uzyskiwanej liczbie chromatycznej. Algorytm ten jednak nie jest lepszy od `D_Satur` przy niedużej gęstości wierzchołków w grafie, ale dla gęstych grafów jest to najlepiej kolorujący algorytm.
Algorytm `TURBOColor3000` koloruje bardzo szybko, jednak najmniej optymalnie, przy gęstości wierzchołków większej niż (lub równej) 8 liczba kolorów jest większa niż 12. Na 12 kolorów można pokolorować graf używając stałego podziału powierzchni na sześciokąty, a więc jeszcze szybciej. Dla gęstości mniejszych niż 8 różnica w czasie jest na tyle nieznaczna, że bardziej opłaca się użyć jednego z pozostałych algorytmów. Dlatego w testowanych przypadkach `TURBOColor3000` okazał się być bezużyteczny.