Politechnika Wrocławska
Wydział Informatyki i Telekomunikacji

Kierunek: **Informatyka Algorytmiczna (INA)**

PRACA DYPLOMOWA
INŻYNIERSKA

Preferencyjne kolorowanie grafów losowych

Karol Janic

Opiekun pracy
dr Dominik Bojko

Słowa kluczowe: model Barabási—Albert, planarność, kolorowanie grafu, grubość grafu

WROCŁAW 2024

STRESZCZENIE

Celem pracy było opracowanie metod kolorowania krawędziowego grafów losowych Barabási—Albert tak, aby minimalizując liczbę niezbędnych kolorów zachować planarność monochromatycznych podgrafów. Rozważano także opcję kolorowania zadaną liczbą kolorów przy minimalizacji liczby przecięć w każdym z podgrafów.

W tym celu przedstawiono znane algorytmy wybierania maksymalnego planarnego podgrafu oraz zbadano planarność grafów Barabási—Albert poprzez estymację oczekiwanej liczby podpodziałów grafów Kuratowskiego w grafie losowym. Na tej podstawie zmodyfikowano dotychczasowe znane rozwiązania poprzez dodanie do nich pewnej metryki oraz zaproponowano nowe podejścia. Wszystkie je przetestowano i porównaniu pod względem jakości generowanych rozwiązań jak i czasu działania. Udało się znacząco poprawić jakość otrzymywanych rozwiązań nie powodując znaczących zmian w złożonościach obliczeniowych danych algorytmów. Dodatkowo opisano jedno z zastosowań takiego kolorowania.

ABSTRACT

The aim of this work was to develop methods for edge coloring of Barabási—Albert random graphs in such a way as to minimize the number of required colors while preserving the planarity of monochromatic subgraphs. An additional goal was to explore coloring with a given colors number that minimize the number of crossing edges in each subgraph. An option of coloring with a given number of colors while minimizing the number of crossings in each subgraph was also considered.

To achieve this, well-known algorithms for selecting the maximum planar subgraph were presented, and the planarity of Barabási–Albert graphs was examined by estimating the expected number of Kuratowski graph subdivisions in the random graph. Based on this, existing solutions were modified by incorporating a specific metric, and new approaches were proposed. All methods were tested and compared in terms of the quality of generated solutions as well as execution time. Significant improvements in the quality of the solutions were achieved without causing substantial changes to the computational complexity of the algorithms. Additionally, one application of such coloring was described.

SPIS TREŚCI

WPROWADZENIE

Grafy losowe odgrywają ważną rolę w modelowaniu i analizie złożonych systemów, takich jak sieci społecznościowe, sieci biologiczne czy sieć WWW. Stanowią one matematyczny model grafów, w których połączenia pomiędzy wierzchołkami są tworzone w sposób losowy. Jedną z istotnych koncepcji w teorii grafów losowych jest model Barabási—Albert [2], który pozwala na generowanie grafów rozrastających się i odzwierciedlających zjawisko „preferencyjnego przyłączania". W tych grafach nowe wierzchołki mają tendencję do łączenia się z tymi, które mają już wiele połączeń, co prowadzi do powstania grafu z wierzchołkami o dużej liczbie sąsiadów („tzw. hubów").

Model ten dobrze opisuje także topologię schematów elektronicznych [7], gdzie kilka komponentów pełni centralną rolę. W ich przypadku ważna jest kwestia planarności, ponieważ przy realizacji układu na płytce drukowanej połączenia między komponentami nie mogą się przecinać. W przypadku skomplikowanych układów może się ona składać z kilku warstw, które trzeba efektywnie wyznaczyć.

CEL PRACY

Celem niniejszej pracy inżynierskiej jest zbadanie grafów modelu Barabási—Abert pod kątem ich planarności oraz opracowanie algorytmów do kolorowania krawędziowego grafów w sposób umożliwiający utrzymanie planarności ich monochromatycznych podgrafów lub minimalizację liczby krawędzi powodujących brak planarności.

1. WSTĘP TEORETYCZNY

W niniejszym rozdziale zdefiniuję terminy używane w dalszej części pracy, opiszę model grafu Barabási—Albert oraz problem planarności i twierdzenia z nim związane. Omówię także problem preferencyjnego kolorowania i sposoby oceny jego rozwiązań.

Definicja 1. *Niech* $[n]$ *oznacza zbiór* $\{1, 2, \ldots, n\}$.

Definicja 2. *Niech* H_n^k *oznacza* n*-tą liczbę harmoniczną* k*-tego rzędu:*

$$H_n^{(k)} = \sum_{i=1}^{n} \left(\frac{1}{i}\right)^k . \tag{1.1}$$

Uwaga: H_n oznacza $H_n^{(1)}$.

Twierdzenie 1. *Liczba harmoniczna* $H_n^{(k)}$ *dla* $k > 1$ *jest skończona i zbiega do wartości funkcji zeta Riemanna* $\zeta(k)$*, gdy* n *dąży do nieskończoności.*

Definicja 3. *Niech* f *i* g *będą funkcjami* $\mathbb{N} \to \mathbb{R}$*, a ich argument* n *dąży do nieskończoności. Zapis* $f = O(g)$ *lub* $f =_O g$ *oznacza, że funkcja* f *jest asymptotycznie niewiększa niż* g*:*

$$(\exists c > 0)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0) \quad |f(n)| < c \cdot |g(n)| . \tag{1.2}$$

Zapis $f = \Theta(g)$ *lub* $f =_\Theta g$ *oznacza, że funkcje* f *i* g *są asymptotycznie podobne:*

$$(\exists c_1, c_2 > 0)(\exists n_0 \in \mathbb{N})(\forall n \geq n_0) \quad c_1 \cdot |f(n)| < |g(n)| < c_2 \cdot |f(n)| . \tag{1.3}$$

Zapis $f(n) \sim g(n)$ *oznacza, że funkcje są asymptotycznie równoważne:*

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = 1. \tag{1.4}$$

Definicja 4. *Grafem prostym* G *nazywamy parę rozłącznych zbiorów* (V, E) *takich, że* $E \subseteq V^{[2]}$*, gdzie* $V^{[2]}$ *to zbiór wszystkich 2-elementowych podzbiorów* V*. Mówimy, że wierzchołki* $v_1, v_2 \in V$ *są połączone krawędzią ze zbioru* E*, gdy* $\{v_1, v_2\} \in E$*.*

Definicja 5. *Grafem skierowanym* G *nazywamy parę rozłącznych zbiorów* (V, E) *takich, że* $E \subseteq V \times V$*, gdzie* $V \times V$ *to zbiór par uporządkowanych. Mówimy, że wierzchołki* $v_1, v_2 \in V$ *są połączone krawędzią ze zbioru* E*, gdy* $(v_1, v_2) \in E$ *lub* $(v_2, v_1) \in E$*.*

W dalszej części pracy, jeśli nie podano wprost, czy graf jest prosty, czy skierowany, to stwierdzenie zachodzi dla obu przypadków.

Definicja 6. *Drogą długości $k+1$ w grafie $G=(V,E)$ nazywamy sekwencję parami różnych wierzchołków $v_1, v_2, \ldots, v_k$, w której każda para kolejnych wierzchołków jest połączona krawędzią ze zbioru E.*

Definicja 7. *Cyklem w grafie $G=(V,E)$ nazywamy drogę, w której pierwszy i ostatni wierzchołek są tożsame.*

Definicja 8. *Graf $H=(V_H,E_H)$ jest podgrafem grafu $G=(V_G,E_G)$ jeżeli $V_H \subseteq V_G$ oraz $E_H \subseteq E_G$.*

Definicja 9. *Graf prosty $G=(V,E)$ jest spójny, jeśli pomiędzy każdymi dwoma jego wierzchołkami istnieje droga.*

Definicja 10. *Komponentą $C=(V_C,E_C)$ grafu prostego $G=(V_G,E_G)$ jest największy (w sensie inkluzji) możliwy spójny podgraf G.*

Definicja 11. *Drzewem $T=(V_T,E_T)$ nazywamy graf prosty, który jest acykliczny i spójny.*

Definicja 12. *Stopniem wierzchołka v w grafie prostym G nazywamy liczbę krawędzi, które są z nim incydentne. Oznaczamy go przez $deg_G(v)$.*

Definicja 13. *Stopniem wejściowym wierzchołka v w grafie skierowanym G nazywamy liczbę krawędzi, które są do niego skierowane. Oznaczamy go przez $\deg_G^{\mathrm{IN}}(v)$.*

Definicja 14. *Stopniem wyjściowym wierzchołka v w grafie skierowanym G nazywamy liczbę krawędzi, które są z niego wychodzące. Oznaczamy go przez $\deg_G^{\mathrm{OUT}}(v)$.*

Definicja 15. *Sąsiedztwem wierzchołka v w grafie prostym $G=(V,E)$ nazywamy zbiór wierzchołków połączonych krawędzią z danym wierzchołkiem v. Oznaczamy je przez $N_G(v)$:*

$$N_G(v) = \{u \in V : \{v,u\} \in E\} . \tag{1.5}$$

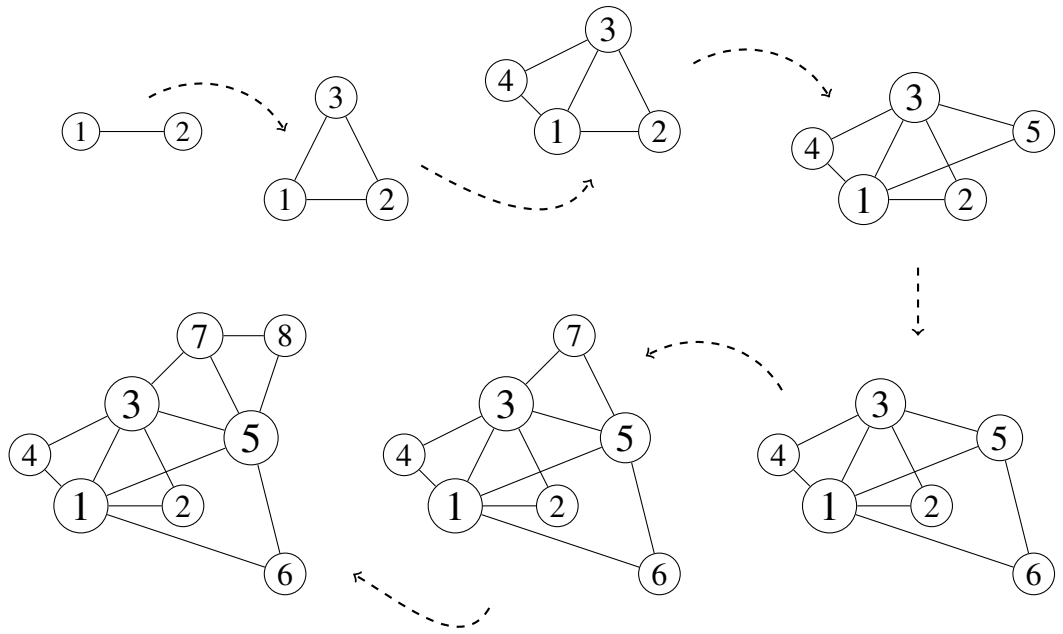
Definicja 16. *Sąsiedztwem wierzchołka v w grafie skierowanym $G=(V,E)$ nazywamy zbiór wierzchołków, do których prowadzą krawędzie od v. Oznaczamy je przez $N_G(v)$:*

$$N_G(v) = \{u \in V : (v,u) \in E\} . \tag{1.6}$$

1.1. MODEL BARABÁSI—ALBERT

Definicja 17. *Modelem Barabási—Albert (BA) nazywamy model opisujący proces powstawania nieskierowanego grafu losowego G_m^n ($n, m \in \mathbb{N}^+$). Definiuje on ciąg grafów $G_m^{n_0}, G_m^{n_0+1} \ldots, G_m^n$ ($n_0, m_0 \in \mathbb{N}^+ \wedge n_0 \leq n \wedge m_0 \geq m$) takich, że $G_m^{n_0}$ jest ustalonym grafem o m_0 krawędziach i n_0 wierzchołkach numerowanych liczbami ze zbioru $[n_0]$ takich, że każdy z nich ma co najmniej jedną krawędź incydentną. Graf G_m^{k+1} powstaje poprzez dodanie do G_m^k wierzchołka o numerze $k+1$ i połączenie go z różnymi m niezależnie losowo wybranymi wierzchołkami $v_1, \ldots, v_m$ tak, aby:*

$$\mathbb{P}(v_i = u) = \frac{\deg_{G_m^k}(u)}{\sum\limits_{v \in V_{G_m^k}} \deg_{G_m^k}(v)} = \frac{\deg_{G_m^k}(u)}{2\left|E_{G_m^k}\right|} \qquad (u \in [k])\,. \tag{1.7}$$



Rys. 1.1: Wizualizacja procesu tworzenia grafu G_2^8 z G_2^2, będącego ścieżką P_1.

Model ten łączy w sobie wzrost liczby wierzchołków oraz preferencyjne przyłączanie, których efektem jest bezskalowość (ang. *scale-free*) grafu. Oznacza to, że wierzchołki mają bardzo zróżnicowane stopnie. Rozkład stopni opisuje rozkład potęgowy:

$$\mathbb{P}(\deg(v) = k) \propto k^{-\gamma}\,. \tag{1.8}$$

W przypadku modelu BA mamy $\gamma = 3$. Wierzchołki o dużym stopniu nazywamy hubami. Mechanizm przyłączania nowych wierzchołków odzwierciedla zasadę Pareta — niewielka liczba wierzchołków dominuje resztę pod względem ich stopnia, zaś duża liczba wierzchołków ma bardzo mały stopień.

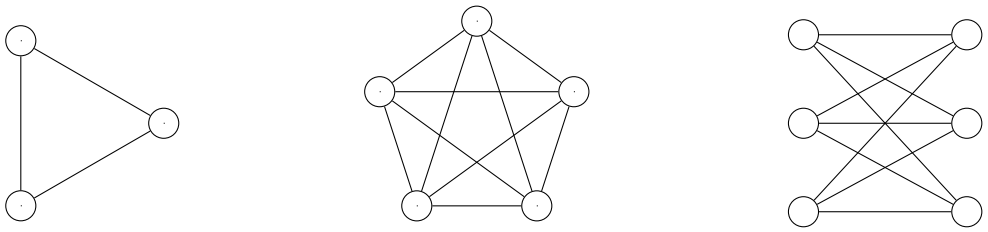
1.2. PLANARNOŚĆ

Sekcja ta odnosi się do grafów prostych. Teoria w niej opisana może zostać jednak zastosowana także do grafów skierowanych poprzez konwersję ich do grafów prostych — zamianę krawędzi skierowanych na nieskierowane, eliminację krawędzi wielokrotnych oraz usunięcie pętli.

Definicja 18. *Graf $G = (V, E)$ nazywamy planarnym, jeżeli możemy narysować go na płaszczyźnie tak, aby żadne dwie krawędzie się nie przecinały (dopuszczamy przecięcia krawędzi w wierzchołkach).*

Definicja 19. *Graf K_n jest grafem pełnym o n wierzchołkach.*

Definicja 20. *Graf $K_{n,n}$ jest grafem pełnym dwudzielnym o $2n$ wierzchołkach.*



Rys. 1.2: Grafy (od lewej strony) K_3, K_5 i $K_{3,3}$.

Twierdzenie 2. *Niech $G = (V, E)$ będzie grafem prostym, spójnym, planarnym o co najmniej 3 wierzchołkach. Wtedy:*

$$|E| \leq 3 \cdot |V| - 6 \,. \tag{1.9}$$

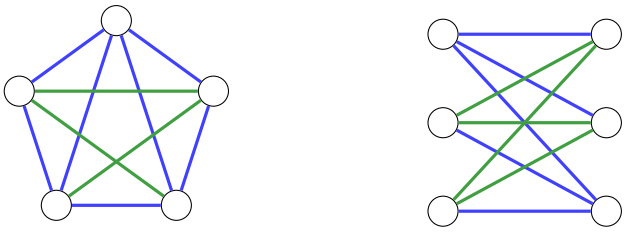
Definicja 21. *Elementarnym podpodziałem krawędzi $e \in E$ w grafie $G = (V, E)$ nazywamy zastąpienie jej drogą długości 2.*

Definicja 22. *Grafy G_1, G_2 nazywamy homeomorficznymi, jeśli istnieje graf G taki, że zarówno G_1 jak i G_2 da się otrzymać z G za pomocą skończonej sekwencji operacji elementarnego podpodziału. Mówimy, że G_1 i G_2 są podpodziałami G.*

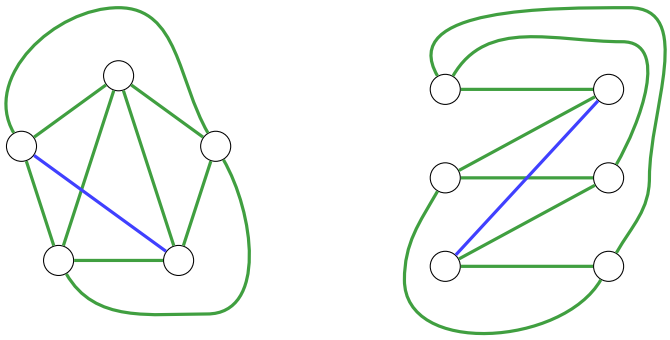
Twierdzenie 3 (Kuratowski)**.** *Graf G jest planarny, wtedy i tylko wtedy, gdy nie zawiera podgrafu homeomorficznego z $K_{3,3}$ lub z K_5.*

Definicja 23. *Liczbą przecięć grafu G nazywamy najmniejszą możliwą liczbę przecięć krawędzi w dowolnym rysunku grafu G na płaszczyźnie. Oznaczamy ją przez $cr(G)$ (ang. crossing number).*

Definicja 24. *Grubością grafu G nazywamy najmniejszą liczbę planarnych podgrafów (indukowanych przez rozłączne zbiory krawędzi) grafu G, na które można go podzielić. Oznaczamy ją przez $t(G)$ (ang. thickness).*



Rys. 1.3: Grafy K_5 i $K_{3,3}$ mają grubość 2. Kolory oznaczają podział na planarne podgrafy.



Rys. 1.4: Gdy spóbujemy narysować graf K_5 lub $K_{3,3}$ na płaszczyźnie, okazuje się, że możemy poprawnie osadzić wszystkie krawędzie z wyjątkiem jednej. Zatem $cr(K_5) = cr(K_{3,3)} = 1$.

Twierdzenie 4. *Jeżeli graf $G = (V, E)$ jest prosty, to:*

$$t(G) \geq \left\lceil \frac{|E|}{3 \cdot |V| - 6} \right\rceil . \tag{1.10}$$

Powyższe twierdzenie jest wnioskiem z Twierdzenia 2 — żaden monochromatyczny podgraf planarny nie może mieć więcej niż $3 \cdot |V| - 6$ krawędzi. Dodatkowo grubość grafu musi być liczbą naturalną.

1.3. PREFERENCYJNE KOLOROWANIE GRAFÓW

Preferencyjne kolorowanie krawędziowe grafu G polega na przypisaniu każdej krawędzi jednego koloru ze zbioru $\{1, 2, \ldots, c\}$ ($c \in \mathbb{N}$) tak, aby monochromatyczne podgrafy miały zapewnione pewną własność. W naszym przypadku tą własnością jest planarność. Celem takiego kolorowania jest użycie jak najmniejszej liczby kolorów c_{MIN}. Możemy zauważyć, że minimalna liczba kolorów odpowiada grubości grafu G. Problem wyznaczenia grubości grafu jest problemem NP-trudnym [11]. Z tego powodu stosuje się metody heurystyczne w celu uzyskania rozwiązania w zadowalającym czasie. W literaturze heurystyki opierają się na wybieraniu jak największego podgrafu planarnego aż do wyczerpania krawędzi w grafie początkowym. Problem maksymalnego podgrafu planarnego również jest problemem

NP-trudnym [10]. W literaturze obecnych jest kilka podejść mających na celu znalezienie jak najlepszego rozwiązania, używając także heurystyk [9]. Zostaną one przedstawione w następnym rozdziale.

Wspomniana wyżej minimalna liczba niezbędnych kolorów jest jedną z możliwych miar oceny jakości metody kolorowania. Innym podejściem jest określenie wskaźnika aproksymacji podgrafów, czyli stosunku liczby krawędzi w wyznaczonym podgrafie $H = (V_H, E_H)$ do maksymalnej liczby krawędzi w grafie planarnym o wierzchołkach jak w grafie $G = (V_G, E_G)$, z którego wybierany jest planarny podgraf:

$$\rho_G(H) = \frac{|E_H|}{3 \cdot |V_G| - 6} \,. \tag{1.11}$$

Odmiennym sposobem oceny jest zadanie maksymalnej liczby kolorów, wyznaczenie monochromatycznych podgrafów i sprawdzenie, ile z krawędzi w każdym nich należy usunąć, aby był on planarny. Wiąże się to z liczbą przecięć danego grafu G, tj. $cr(G)$.

2. ISTNIEJĄCE ROZWIĄZANIA

W tym rozdziale opiszę trzy podejścia do wyznaczania maksymalnych planarnych podgrafów zadanego grafu. Zostały one zebrane i opisane w pracy [9]. Następnie opiszę algorytm preferencyjnego kolorowania, bazujący na nich.

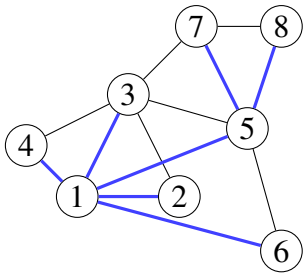
Definicja 25. *Drzewem rozpinającym grafu $G = (V, E)$ nazywamy drzewo, które zawiera wszystkie wierzchołki V, a zbiór krawędzi drzewa jest podzbiorem zbioru krawędzi E.*

Definicja 26. *Trójkątnym kaktusem nazywamy graf prosty, w którym wszystkie cykle są trójkątami K_3 i każda krawędź należy dokładnie do jednego cyklu.*

Definicja 27. *Struktura trójkątna to graf, w którym wszystkie minimalne cykle są trójkątami.*

2.1. DRZEWO ROZPINAJĄCE

Najprostszą heurystyką jest użycie drzewa rozpinającego. Jedną z jego własności jest acykliczność. Z tego powodu nie może zawierać ono podgrafu będącego podpodziałem K_5 lub $K_{3,3}$, ponieważ zawierają one co najmniej jeden cykl. W przypadku zastosowania tej metody otrzymujemy graf H o wierzchołkach jak w grafie G i $|V_G| - 1$ krawędziach. Zatem $\rho_G(H) = \frac{|V_G| - 1}{3(|V_G| - 2)} > \frac{1}{3}$. Drzewo rozpinające możemy znaleźć przy użyciu algorytmu DFS. Jego pseudokod został przedstawiony na Listingu 1. Polega on na eksploracji grafu, zaczynając od dowolnego wierzchołka, poprzez przechodzenie jak najdalej możliwą drogą, aż wróci się do odwiedzonego już wierzchołka. W implementacji z użyciem stosu wierzchołki dodawane są na jego szczyt w momencie odwiedzania ich nieodwiedzonego sąsiada(linie 13-14), a zdejmowane w momencie rozpoczynania procesu jego odwiedzania (linia 11). Aby wygenerować drzewo rozpinające, należy zapisywać krawędzie, które łączą obecnie odwiedzany wierzchołek i wierzchołek dodawany na stos (linia 15). Algorytm zakłada, że graf jest spójny. W przypadku niespójności należy użyć algorytmu dla każdej komponenty osobno. Ponieważ każdy wierzchołek jest odwiedzany tylko raz (zapewnia to struktura zainicjalizowana w liniach 4-7), to złożoność czasowa algorytmu wynosi $O(|V_G| + |E_G|)$. Złożoność pamięciowa jest liniowa względem liczby wierzchołków grafu, ponieważ na stos każdy wierzchołek trafia tylko raz. Drzewo rozpinające wyznaczone przy użyciu opisanego algorytmu zostało zaznaczone na Rysunku 2.1.



Rys. 2.1: Wizualizacja drzewa rozpinającego przykładowego grafu zaczynając od $v_0 = 1$. Jego krawędzie zostały zaznaczone na niebiesko.

Algorithm 1: DrzewoRozpinające

```
   Dane wejściowe: spójny graf prosty G = (V_G, E_G)
   Wynik: drzewo rozpinające graf G
 1 V_T ← V_G;
 2 E_T ← ∅;
 3 stos ← ∅;
 4 odwiedzone ← {};
 5 dla każdego v ∈ V_G wykonuj
 6 |   odwiedzone[v] = NIE;
 7 koniec
 8 umieść dowolny wierzchołek v_0 ∈ V_G na stosie;
 9 odwiedzone[v_0] = TAK;
   // symulacja DFS na stosie
10 dopóki stos ≠ ∅ wykonuj
11 |   v ← zdejmij szczyt stosu;
12 |   dla każdego u ∈ N_G(v) wykonuj
13 |   |   jeżeli odwiedzone[u] = NIE wtedy
14 |   |   |   umieść u na stosie;
15 |   |   |   E_T ← E_T ∪ {v, u};
16 |   |   |   odwiedzone[u] = TAK;
17 |   |   koniec
18 |   koniec
19 koniec
20 zwróć (V_T, E_T);
```

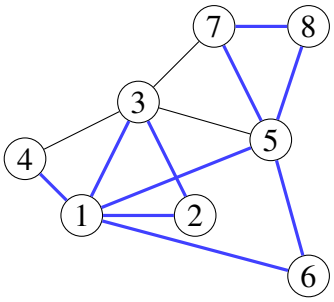
2.2. STRUKTURA „KAKTUS”

Kolejne rozwiązanie jest oparte na trójkątnych kaktusach i strukturach trójkątnych. Jego pseudokod został zaprezentowany na Listingu 2. Składa się ono z dwóch etapów. W pierwszym (linie 6-20) wybierany jest zachłannie maksymalny trójkątny kaktus. Następnie jest on rozszerzalny do struktury trójkątnej poprzez dodawanie pozostałych krawędzi, tak aby nie tworzyły one nowych cykli (linie 21-26).

Graf będący rezultatem tego algorytmu jest planarny, ponieważ zawiera cykle wyłącznie

o długości 3. Z tego powodu nie może zawierać podgrafu będącego podpodziałem K_5, który zawiera cykl długości co najmniej 5 oraz będącego podpodziałem $K_{3,3}$, którego cykle mają długości co najmniej 4. Dolne ograniczenie na $\rho_G(H)$ dla tego rozwiązania jest lepsze od poprzedniego i wynosi $\rho_G(H) \geq \frac{7}{18}$ i może zostać ulepszone do $\rho_G(H) \geq \frac{2}{5}$ poprzez niezachłanny wybór w pierwszym etapie algorytmu [6].

Ważnym elementem tego algorytmu jest struktura danych używana do śledzenia komponent w powstającym podgrafie. Dzięki niej będziemy mieli pewność, że stworzenie kolejnego trójkąta nie spowoduje powstania innych cykli. Powinna ona pozwalać na tworzenie zbiorów, efektywne ich łączenie oraz sprawdzanie, czy dwa elementy znajdują się w tym samym zbiorze. Dobrą strukturą do tego celu są zbiory rozłączne. Początkowo każdy wierzchołek należy do jednoelementowej komponenty. (inicjalizacja zbiorów rozłącznych odbywa się w liniach 3-5). Następnie dla każdego wierzchołka wybierane są trójkąty w ten sposób, aby każdy z ich wierzchołków znajdował się w innej komponencie. Należy to robić iteracyjnie, tak aby nie generować wszystkich trójkątów, a dopiero później sprawdzać ich przynależność do komponent - trzy zagnieżdżone pętle w liniach 6-20 „na bieżąco" sprawdzają przynależność do komponent i różność wierzchołków. Po znalezieniu takiego trójkąta aktualizowane są zbiory komponent i struktura trójkątna (linie 19-20). Gdy nie ma już kolejnych trójkątów do dodania, włączane są krawędzie o końcach w różnych komponentach i znów aktualizowane są zbiory komponent i struktura trójkątna (linie 25-26). Nie spowoduje to powstania cyklu, ponieważ wymagałoby to co najmniej dwóch krawędzi pomiędzy komponentami. Operacje na zbiorach rozłącznych mają w najgorszym przypadku logarytmiczną złożoność czasową względem liczby elementów w zbiorach, a ich amortyzowaną złożoność opisuje logarytm iterowany względem liczby wierzchołków. Funkcja ta rośnie bardzo wolno, zatem może być interpretowana w praktyce jako stała. Na złożoność czasową algorytmu składa się kilka czynników. Inicjalizacja zbioru komponent wykonuje się w $O(|V_G|)$. Przeglądanie trójkątów w najgorszym przypadku wyniesie $O(|V_G|^3)$, zaś końcowe dodawanie krawędzi $O(|E_G| \log(|V_G|))$, ponieważ każda aktualizacja zbioru komponent może wymagać czasu $O(\log(|V_G|))$. Wynik działania opisanego algorytmu, którego pseudokod został przedstawiony na Rysunku 2.2.



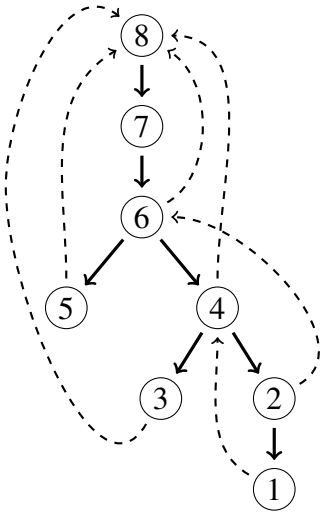
Rys. 2.2: Wizualizacja struktury kaktusowej na grafie bez wag. Wybrane trójkąty: {1,2,3}, {1,5,6}, {5,7,8} i dobrane krawędzie: {1,4} zostały zaznaczone na niebiesko.

Algorithm 2: StrukturaKaktusowa

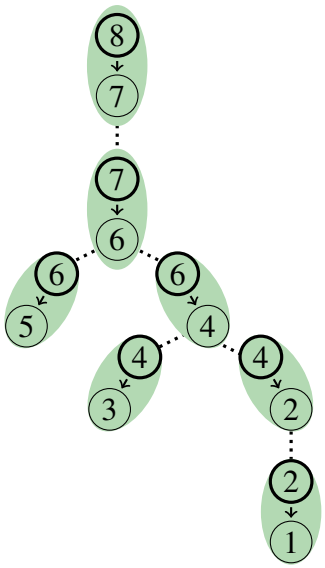
```
Dane wejściowe: spójny graf prosty G = (V_G, E_G)
Wynik: struktura trójkątna zawarta w G
1  V_C ← ∅;
2  E_C ← ∅;
3  komponenty ← {};   // struktura zbiorów rozłącznych
4  dla każdego v ∈ V_G wykonuj
5      dodaj zbiór {v} do komponenty;
   // wybór trójkątów
6  dla każdego t_1 ∈ V_G wykonuj
7      k_1 ← reprezentant zbioru, do którego należy t_1;
8      dla każdego t_2 ∈ N_G(t_1) wykonuj
9          k_2 ← reprezentant zbioru, do którego należy t_2;
10         jeżeli k_1 = k_2 wtedy
11             kontynuuj;
12         dla każdego t_3 ∈ N_G(t_1) wykonuj
13             jeżeli t_2 = t_3 wtedy
14                 kontynuuj;
15             k_3 ← reprezentant zbioru, do którego należy t_3;
16             jeżeli k_1 = k_3 lub k_2 = k_3 wtedy
17                 kontynuuj;
18             jeżeli {t_2, t_3} ∈ E_G wtedy
19                 E_C ← E_C ∪ {{t_1, t_2}, {t_1, t_3}, {t_2, t_3}};
20                 połącz zbiory zawierające t_1, t_2, t_3;
   // dobieranie krawędzi niepowodujących nowych cykli
21 dla każdego {v, u} ∈ E_G wykonuj
22     k_1 ← reprezentant zbioru do którego należy v;
23     k_2 ← reprezentant zbioru do którego należy u;
24     jeżeli k_1 ≠ k_2 wtedy
25         E_C ← E_C ∪ {{v, u}};
26         połącz zbiory zawierające v i u;
27 zwróć (V_C, E_C)
```

2.3. MODYFIKACJA TESTU PLANARNOŚCI

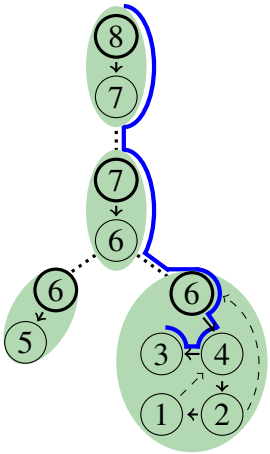
Następnym rozwiązaniem jest modyfikacja algorytmu Boyera—Myrvolda [5, 12] testującego planarność grafu. Opiera się on na konstrukcji i osadzaniu coraz to większych bloków złożonych z krawędzi grafu $G = (V_G, E_G)$ tak, aby zachować jego planarność. Ogólny pseudokod został przedstawiony na Listingu 3 a jego szczegóły obrazują Rysunki 2.3a, 2.3b, 2.3c, 2.3d, 2.3e, 2.3f.



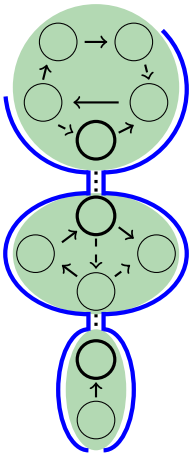
(a) Wizualizacja drzewa przejścia algorytmu DFS - liczby reprezentują odwrotny porządek.



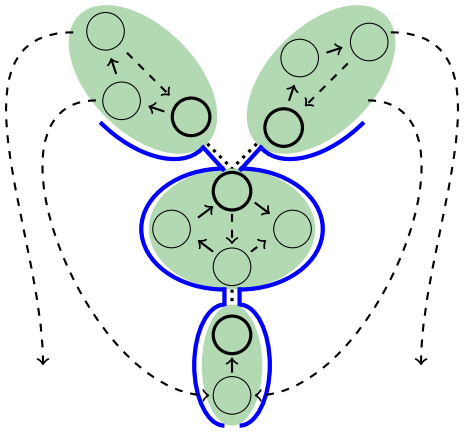
(b) Wizualizacja początkowych bloków z wybranymi korzeniami(pogrubione wierzchołki).



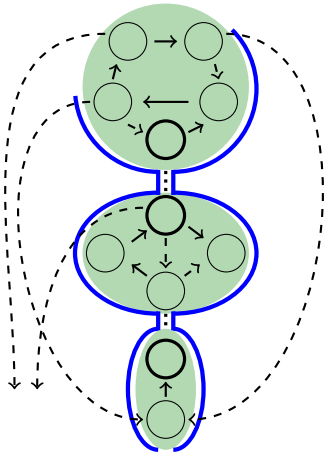
(c) Przykład potencjalnej drogi z wierzchołka 8 do 3. Strzałki przerywane oznaczają *backedges*.



(d) Przykład dwóch kompatybilnych dróg - nie dzielą one wspólnych krawędzi.



(e) Przykład dwóch kompatybilnych dróg - pochodzą z różnych bloków zewnętrznych.



(f) Przykład dwóch niekompatybilnych dróg.

Pierwszym krokiem jest wyznaczenie odwrotnego porządku wierzchołków grafu przy przejściu go algorytmem DFS oraz krawędzi powrotnych (ang. *backedges*). Przedstawia to rysunek Rysunek 2.3a a w pseudokodzie odpowiadają linie 1-3. Kolejno tworzone są bloki reprezentujące krawędzie w drzewie DFS. Dla każdego z nich ustalany jest korzeń bloku — wierzchołek o największym indeksie w nim zawartym. Zostało to zwizualizowane na Rysunek 2.3b. Inicjalizowane jest także osadzenie grafu G — $\tilde{G}$ (linia 4). Następnie dla każdego wierzchołka v w ustalonym wcześniej odwrotnym porządku odbywają się cztery fazy — osadzanie bloków reprezentujących krawędzie drzewa wychodzące z aktualnego wierzchołka (linie 6-8), wyszukiwanie dróg (linie 9-11), osadzanie bloków reprezentujących znalezioną drogę (linie 12-14) oraz sprawdzenie, czy osadzenie bloków i krawędzi powrotnych z nim związanych nie powoduje żadnego konfliktu (linie 15-19).

Wyszukiwanie dróg polega na znalezieniu wszystkich dróg w obecnie osadzonych blokach z wierzchołka v do wierzchołków połączonych krawędziami powrotnymi we wcześniej wyznaczonym drzewie. Wierzchołki tworzące te drogi muszą spełniać następujące własności:

1. muszą znajdować się na granicy dotychczas powstałych bloków,
2. wierzchołki będące korzeniami bloków, muszą być dzieckiem wierzchołka rozcinającego w bloku nadrzędnym (wierzchołkiem rozcinającym nazywamy wierzchołek występujący w więcej niż jednym bloku),
3. z wyjątkiem początkowego i końcowego wierzchołka nie mogą być one wierzchołkami zewnętrznymi w grafie stworzonym przez dotychczas wybrane bloki.

Przykład takiej drogi został pokazany na Rysunek 2.3c. W wyniku tej operacji może zostać wybrana więcej niż jedna droga. Wówczas należy sprawdzić, czy wybrane drogi są ze sobą kompatybilne tj., czy nie dzielą ze sobą żadnej krawędzi. Niekompatybilność może wystąpić tylko, gdy dwie drogi przechodzą przez ten sam blok. Załóżmy, że dwie drogi przechodzą przez ten sam blok. Wówczas:

1. jeśli drogi nie dzielą krawędzi w tym bloku to nie dzielą krawędzi w żadnym innym bloku (Rysunek 2.3d),
2. jeśli drogi przechodzą przez dwa różne zewnętrzne bloki (blok zewnętrzny zawiera wierzchołek zewnętrzny w obecnie osadzonych już blokach), to nie dzielą żadnej krawędzi rozpatrywanego bloku (Rysunek 2.3e),
3. jeśli drogi nie dzielą krawędzi rozpatrywanego bloku, a korzeń tego bloku jest różny od korzenia aktualnie rozpatrywanego wierzchołka, to korzeń bloku nie jest zewnętrznym wierzchołkiem obecnego osadzenia bloków (Rysunek 2.3f).

Faza osadzania polega na przechodzeniu bloków z poprzedniej iteracji wzdłuż wyznaczonych dróg i łączeniu bloków, które ścieżka zawiera. Oprócz tego, do powstałego bloku należy dodać krawędzie powrotne o końcu w aktualnie rozważanym wierzchołku.

Algorithm 3: TestPlanarnościBoyerMyrvold

```
Dane wejściowe: graf prosty G = (V_G, E_G)
Wynik: informacja o planarności i osadzenie grafu lub
       znaleziony podgraf Kuratowskiego
wykonaj algorytm DFS na grafie G;
zapisz kolenność odwiedzania wierzchołków;
odwróć kolejność odwiedzonych wierzchołków;
zainicjalizuj osadzenie grafu G - G̃;
dla każdego v w wyznaczonej kolejności wykonuj
    dla każdego dziecka c wierzchołka v w drzewie DFS wykonuj
        dodaj komponent reprezentujący krawędź drzewa (v, c) do G̃
    koniec
    dla każdego w połączonego w drzewie DFS krawędzią powrotną
     z v wykonuj
        wyszukaj ścieżki w G pomiędzy v oraz w;
    koniec
    dla każdego dziecka c wierzchołka v w drzewie DFS wykonuj
        osadź drogę pomiędzy v i c w G̃;
    koniec
    dla każdego w połączonego w drzewie DFS krawędzią powrotną
     z v wykonuj
        jeżeli (v, w) ∉ G̃ wtedy
            // nie udało osadzić się wszystkich backedges
            zwróć (NIEPLANARNY, G̃);
        koniec
    koniec
koniec
zwróć (PLANARNY, G̃);
```

Jeśli w opisanych wyżej fazach udało się znaleźć drogi dla każdej pary wierzchołków połączonych krawędzią powrotną, to możliwe jest poprawne osadzenie wierzchołków, a co za tym idzie — graf jest planarny. W przeciwnym wypadku graf nie jest planarny.

Modyfikacja przedstawionego testu planarności polega na przerwaniu testu w momencie niemożliwości osadzenia kolejnej krawędzi powrotnej oraz zwróceniu aktualnego osadzenia bloków.

Złożoność czasowa tego algorytmu jest liniowa względem wielkości grafu, ponieważ algorytm DFS ma liniową złożoność czasową, proces znajdywania poprawnych dróg w grafie przegląda każdą krawędź dwa razy, a każdy wierzchołek i krawędź osadzane są tylko raz. Ważnym elementem algorytmu jest struktura danych służąca do łączenia i przechodzenia bloków. Musi ona pozwalać na efektywne przechodzenie po wierzchołkach bloków w różnych kierunkach (tj. zgodnie i przeciwnie z ruchem wskazówek zegara) i odwracanie

kolejności elementów. Taką zaletę ma struktura danych *bicomp* opisana w pracy [12]. Struktura ta opiera się na liście cyklicznej. Każdy jej element zawiera wskaźniki na dwa elementy sąsiednie. Dodatkowo pamiętany jest wskaźnik do elementu ostatnio odwiedzonego podczas przechodzenia listy, na podstawie którego można określić kierunek, w którym chce się listę obecnie przechodzić. Pozwala to na zmianę „orientacji" listy w czasie stałym. Złożoność pamięciowa także jest liniowa względem wielkości grafu. Wpływa na to użycie algorytmu DFS oraz struktura danych zarządzająca blokami.

2.4. MAKSYMALIZACJA PODGRAFU

Przedstawione wyżej rozwiązania wybierają podgraf, który jest planarny. Mogą jednak istnieć jeszcze krawędzie z grafu początkowego, których dodanie nie będzie przeszkodą dla planarności podgrafu. W tym celu możemy dla każdej niewybranej krawędzi sprawdzić, czy po jej dodaniu planarność zostanie zachowana. Jeśli tak, włączamy ją do podgrafu. Złożoność czasowa takiego procesu wynosi $O(|V_G|^2)$, ponieważ test planarności możemy wykonać w czasie $O(|V_G| + |E_G|) = O(|V_G| + m \cdot |V_G|) = O(|V_G|)$, a potencjalnie odpowiednich krawędzi jest $O(|E_G|) = O(m \cdot |V_G|) = O(|V_G|)$. Pseudokod tego rozwiązania został pokazany na Listingu 4.

Algorithm 4: MaksymalizacjaPodgrafu

```
Dane wejściowe: bazowy graf prosty G = (V_G, E_G)
Wynik: wybrany planarny podgraf H = (V_H, E_H)
Wynik: maksymalny planarny podgraf H*
1 V_H* ← V_H;
2 E_H* ← E_H;
3 dla każdego e ∈ E_G \ E_H wykonuj
4 |   E_H* ← E_H* ∪ {e};
5 |   jeżeli jestPlanarny(H*) = NIE wtedy
6 |   |   E_H* ← E_H* \ {e};
7 |   koniec
8 koniec
9 zwróć (V_H*, E_H*);
```

2.5. PREFERENCYJNE KOLOROWANIE

Znając już heurystyki zwracające planarny podgraf zadanego grafu, możemy opisać metodę preferencyjnego kolorowania. Polega ona na wybieraniu planarnego podgrafu jedną z trzech wyżej opisanych heurystyk i uzupełnieniu go procedurą maksymalizacji. Następnie krawędzie wybranego podgrafu usuwamy z grafu początkowego i ustawiamy im wspólny, unikalny kolor. Procedurę powtarzamy aż do braku pozostałych krawędzi.

Algorithm 5: PreferencyjneKolorowanie

```
Dane wejściowe: graf prosty G = (V_G, E_G)
Wynik: kolorowanie krawędzi grafu  c : E_G → ℕ
c ← ∅;
G' ← G; // algorytm działa na kopii grafu wejściowego
k ← 1; // licznik kolorów
dopóki E_G' ≠ ∅ wykonuj
    H ← planarny podgraf G' wybrany jedną z heurystyk;
    H* ← maksymalizacja podgrafu H;
    E_G' → E_G' \ E_H*;
    dla każdego e ∈ E_H* wykonuj
        c[e] ← k;
    koniec
    k ← k + 1;
koniec
zwróć c;
```

Procedura ta została przedstawiona na Listingu 5. Jej złożoność obliczeniowa zależy od wybranej heurystyki i liczby niezbędnych kolorów.

3. OCZEKIWANA LICZBA PODPODZIAŁÓW $K_{3,3}$ I K_5 W GRAFIE G_m^n

W tym rozdziale wyznaczymy oczekiwaną liczbę podpodziałów $K_{3,3}$ i K_5 w grafie BA G_m^n. Na podstawie pracy „Mathematical results on scale-free random graphs" [4] wyznaczymy oczekiwaną liczbę podgrafów $K_{3,3}$ i K_5 w G_m^n. Następnie rozszerzymy to na liczbę podpodziałów grafów $K_{3,3}$ i K_5 w nim zawartych.

Twierdzenie 5 (Bollobás)**.** *Niech $G_1^n = (V_G, E_G)$ będzie grafem BA, $H = (V_H, E_H)$ będzie grafem skierowanym o wierzchołkach ze zbioru V_G, krawędziach skierowanych od wierzchołka o wyższym indeksie do wierzchołka o niższym indeksie oraz o stopniach wyjściowych nie większych niż 1. Wówczas prawdopodobieństwo, że krawędzie E_H występują w G_1^n wynosi:*

$$p_H = \prod_{v \in V_H} \deg_H^{\mathrm{IN}}(v)! \prod_{(u,v) \in E_H} \frac{1}{2\sqrt{uv}} \exp\left(O\left(\sum_{v \in V_H} C_H(v)^2 / v \right) \right), \tag{3.1}$$

gdzie $C_H(v) = |\{(u, w) \in E_H : u \leq v \leq w\}|$.

Uwaga: *w wersji dokładnej pierwszy produkt odnosi się wyłącznie do wierzchołków, do których prowadzi co najmniej jedna krawędź. W celu uproszczenia zapisu zostało zastąpione to wszystkimi wierzchołkami. Nie powoduje to zmiany wyniku, ponieważ $0! = 1$.*

Aby zastosować Twierdzenie 5 do grafu o parametrze $m > 1$ i n wierzchołkach, możemy zamodelować go jako graf z parametrem 1 zamiast m i mn wierzchołkami, a następnie utworzyć grupy po m wierzchołków, biorąc je po kolei i traktując każdą grupę jako pojedynczy wierzchołek grafu wyjściowego. Tak zdefiniowane grupowanie może doprowadzić do powstania pętli w grafie wynikowym, które są niedozwolone w przyjętej przez nas definicji. Jednakże odpowiedni wybór krawędzi w grafie H może temu zapobiec.

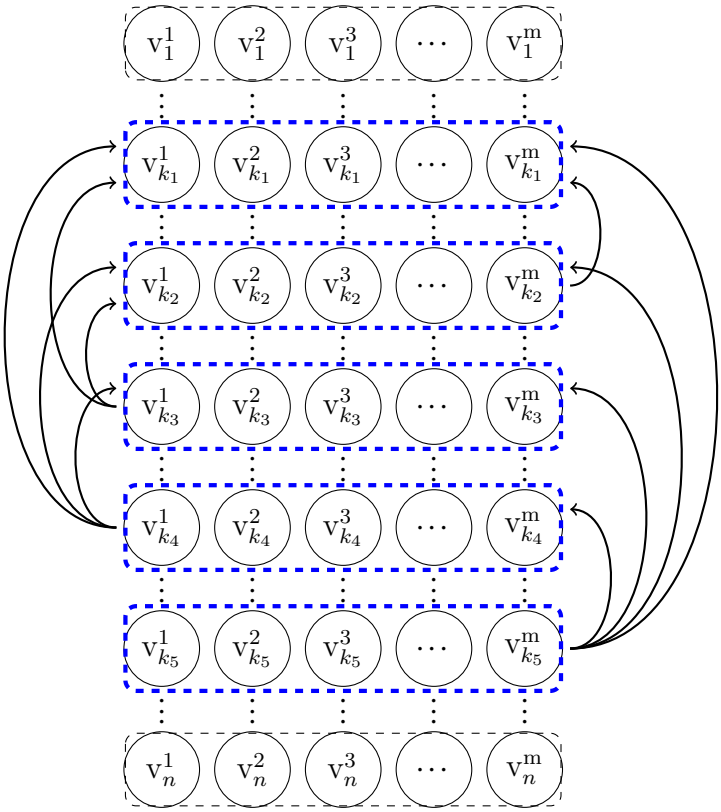
3.1. OCZEKIWANA LICZBA K_5

Weźmy graf G_1^{mn} i pogrupujmy jego wierzchołki w n kolejnych m-elementowych grup. Zdefiniujmy klasę $\mathcal{H}$ reprezentującą skierowane grafy pełne o 5-ciu wierzchołkach i następujących własnościach:

- każdy wierzchołek należy do innej grupy

- krawędzie zorientowane są tak, aby prowadziły od wierzchołków o większym indeksie do tych o indeksie mniejszym
- krawędzie grafu były zawarte w zbiorze krawędzi grafu G_1^{mn}
- stopnie wyjściowe nie są większe niż 1.

Rysunek 3.1 przedstawia rozdział wierzchołków G_1^{mn} na grupy, wybór grup oraz jedyne możliwe poprowadzenie krawędzi między nimi - krawędzie łączą wierzchołki znajdujące się wewnątrz grup, na które wskazują strzałki.



Rys. 3.1: Wizualizacja podziału wierzchołków grafu G_1^{mn} na grupy oraz wyboru krawędzi odpowiadających podgrafowi K_5 w G_m^n.

Chcemy wyznaczyć oczekiwaną liczbę elementów klasy $\mathcal{H}$ oraz ich stopnie wejściowe, które będą niezbędne w dalszych obliczeniach. W tym celu musimy znaleźć liczbę możliwości wyboru wierzchołków w grupach, których dotyczą krawędzie. Wybierzmy jeden z elementów klasy $\mathcal{H}$ i rozważmy możliwe rozkłady krawędzi wewnątrz grup.

1. Rozkład krawędzi wyjściowych wierzchołków wewnątrz grupy k_5 wybieramy na $m(m-1)(m-2)(m-3)$ sposobów.
   Nie mają one krawędzi wejściowych zatem wszystkie stopnie wejściowe są równe 0.
2. Rozkład krawędzi wyjściowych wierzchołków wewnątrz grupy k_4 wybieramy na $m(m-1)(m-2)$ sposobów.
   Rozkład ich krawędzi wejściowych możemy wybrać na m sposobów, stąd grupa będzie zawierała jeden wierzchołek o stopniu 1, a reszta będzie równa 0.

3. Rozkład krawędzi wyjściowych wierzchołków wewnątrz grupy k_3 wybieramy na $m(m-1)$ sposobów.

   Rozkład ich krawędzi wejściowych możemy wybrać na:
   - $m(m-1)$ sposobów - 2 stopnie wejściowe równe 1, reszta 0
   - m sposobów - jeden stopień wejściowy równy 2, reszta 0
4. Rozkład krawędzi wyjściowych wierzchołków wewnątrz grupy k_2 wybieramy na m sposobów.

   Rozkład ich krawędzi wejściowych możemy wybrać na"
   - $m(m-1)(m-2)$ sposobów - 3 stopnie wejściowe równe 1, reszta 0
   - $3m(m-1)$ sposobów - 1 stopień wejściowy równy 2, 1 stopień równy 1, reszta 0
   - m sposobów - jeden stopień wejściowy równy 3, reszta 0
5. Wierzchołki wewnątrz grupy k_1 nie mają krawędzi krawędzi wyjściowych.

   Rozkład ich krawędzi wejściowych możemy wybrać na:
   - $m(m-1)(m-2)(m-3)$ sposobów - 4 stopnie wejściowe równe 1, reszta 0
   - $6m(m-1)(m-2)$ sposobów - 1 stopień wejściowy równy 2, 2 stopnie równe 1, reszta 0
   - $4m(m-1)$ sposobów - 1 stopień wejściowy równy 3, 1 stopień równy 1, reszta 0
   - $3m(m-1)$ sposobów - 2 stopnie równe 2
   - m sposobów - jeden stopień wejściowy równy 4, reszta 0

Niech $H = (V_H, E_H)$ oznacza graf z klasy $\mathcal{H}$. Ustalmy indeksy grup $1 \leq k_1 < k_2 < k_3 < k_4 < k_5 \leq n$ oraz numery wierzchołków połączonymi krawędziami: $(v_{k_5}^{l_1}, v_{k_4}^{l_1}), (v_{k_5}^{l_2}, v_{k_3}^{l_2}), (v_{k_5}^{l_3}, v_{k_2}^{l_3}), (v_{k_5}^{l_4}, v_{k_1}^{l_4}), (v_{k_4}^{l_5}, v_{k_3}^{l_5}), (v_{k_4}^{l_6}, v_{k_2}^{l_6}), (v_{k_4}^{l_7}, v_{k_1}^{l_7}), (v_{k_3}^{l_8}, v_{k_2}^{l_8}), (v_{k_3}^{l_9}, v_{k_1}^{l_9}), (v_{k_2}^{l_{10}}, v_{k_1}^{l_{10}})$ - wierzchołek v_k^l oznacza l-ty wierzchołek z grupy k. Wiemy, że indeksy wierzchołków w grupie k są ograniczone z dołu przez $m(k-1)$ i z góry przez mk.

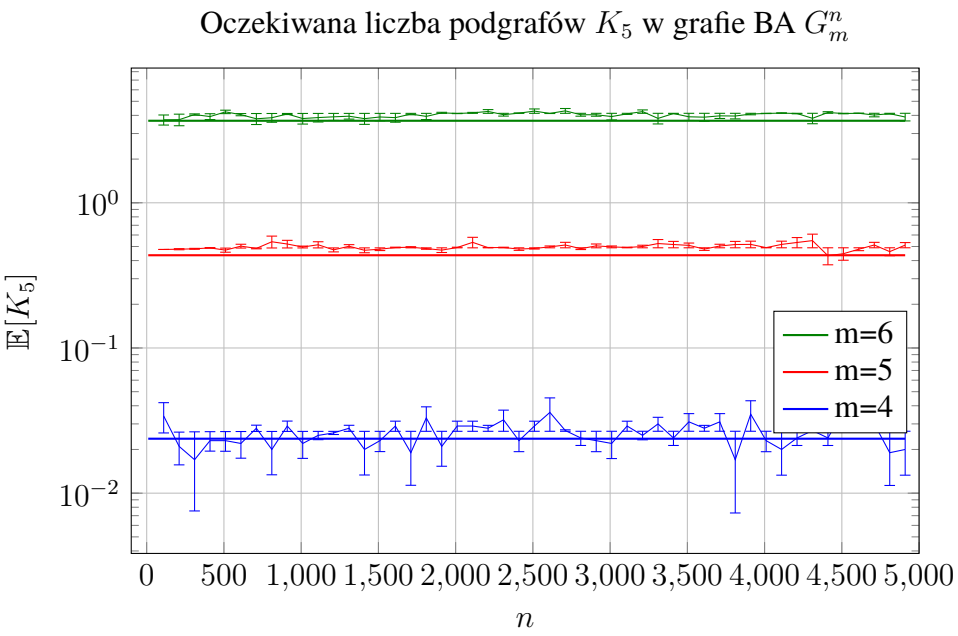
Czynnikiem niezbędnym do obliczenia wzoru 3.1 jest iloraz $C_H(v)^2/v$. Ponieważ H jest ustalonym grafem, to wartości $C_H(v)$ dla poszczególnych wierzchołków wewnątrz grup będą stałe. Wówczas:

$$\prod_{(u,v)\in E_H} 2\sqrt{uv}\exp\left(O\left(\sum_{v\in V_H} C_H(v)^2/v\right)\right) \sim 2^{10}m^{10}k_1^2k_2^2k_3^2k_4^2k_5^2 \tag{3.2}$$

Teraz możemy wyznaczyć już oczekiwaną liczbę grafów w klasie $\mathcal{H}$. W tym celu musimy zsumować prawdopodobieństwa wystąpienia każdego z nich w grafie G_1^{mn}. Prawdopodobieństwa zależą głównie od stopni wejściowych w rozpatrywanym przypadku. Wyznaczenie ich liczby polega na wymnożeniu odpowiednich wyrażeń opisujących liczbę rozdziałów wierzchołków wewnątrz grup. Jest to zadanie pracochłonne, dlatego ta część obliczeń została wykonana przy pomocy skryptu w pakiecie Mathematica. Został on zaprezentowany w Dodatku A. Pozostało jeszcze uwzględnić różne wybory indeksów grup k_1, k_2, k_3, k_4, k_5. Niech $\mathcal{K}_{n,t}$ oznacza zbiór grup indeksów takich, że $1 \leq k_1, k_2, \ldots, k_t \leq n$, zaś $\mathcal{K}_{n,t}^{\mathrm{ord}}$ ozna-

cza zbiór grup uporządkowanych indeksów takich, że $1 \leq k_1 < k_2 < \ldots < k_t \leq n$. Wówczas:

$$\begin{aligned}
\mathbb{E}[K_5] &\sim \sum_{\mathcal{K}^{\text{ord}}_{n,5}} \frac{(m^2-9)(m^2-4)^2(m^2-1)^3}{1024m^2(k_1k_2k_3k_4k_5)^2} \sim \\
&\sim \left(\frac{1}{5!}\right)^2 \sum_{\mathcal{K}_{n,5}} \frac{(m^2-9)(m^2-4)^2(m^2-1)^3}{1024m^2(k_1k_2k_3k_4k_5)^2} = \\
&= \frac{(m^2-9)(m^2-4)^2(m^2-1)^3}{14745600m^2}(H_n^{(2)})^5 \sim \\
&\sim \frac{(m^2-9)(m^2-4)^2(m^2-1)^3}{14745600m^2}\,(\zeta(2))^5
\end{aligned} \tag{3.3}$$

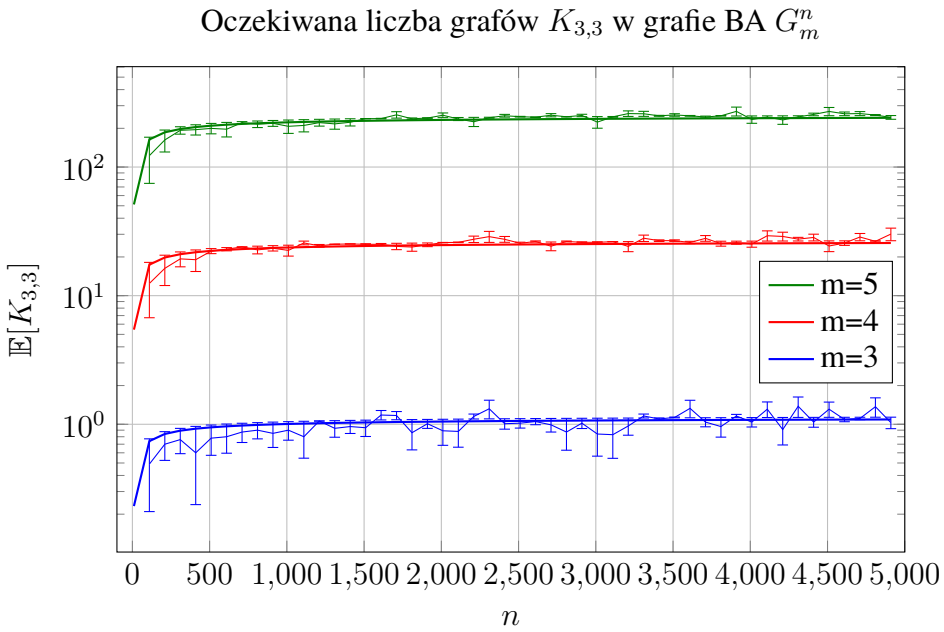


Rys. 3.2: Porównanie oczekiwanej liczby podgrafów K_5 w grafie G_m^n(linia ciągła) z wynikami otrzymanymi eksperymentalnie (krzyżyki) w 100 powtórzeniach. Oś pionowa jest wyskalowana logarytmicznie. Słupki pionowe oznaczają odchylenie standardowe.

3.2. OCZEKIWANA LICZBA $K_{3,3}$

Analogicznie do przedstawionego w poprzedniej sekcji toku rozumowania możemy przeprowadzić obliczenia w celu znalezienia oczekiwanej liczby podgrafów $K_{3,3}$ w G_m^n. W tym przypadku obliczenia są dłuższe, ponieważ zamiast jednego możliwego połączenia grup krawędziami mamy ich 10. Szczegóły obliczeń znajdują się w Dodatku B.

$$\mathbb{E}[K_{3,3}] \sim \frac{(m^2-2)^2(m^2-1)^2(5m^4-5m^2+4)}{2211840\sqrt{5}m^3}(H_n^{(3/2)})^6 \sim$$
$$\sim \frac{(m^2-2)^2(m^2-1)^2(5m^4-5m^2+4)}{2211840\sqrt{5}m^3}(\zeta(3/2))^6 \tag{3.4}$$



Rys. 3.3: Porównanie oczekiwanej liczby podgrafów $K_{3,3}$ w grafie G_m^n(linia ciągła) z wynikami otrzymanymi eksperymentalnie(krzyżyki) w 100 powtórzeniach. Oś pionowa jest wyskalowana logarytmicznie. Słupki pionowe oznaczają odchylenie standardowe.

Na wykresach 3.2, 3.3 widać małą rozbieżność pomiędzy teorią a eksperymentem. W celu ich zrozumienia przeprowadzono symulacje numeryczne poszczególnych części wzorów. Największa różnica pojawia się w szacowaniu oczekiwanej liczby C_H - dla małych liczb wierzchołków ma ona wartość bliższą 1. Dodatkowo, przy sumowaniu wartości dla każdego zbioru indeksów wartość numeryczna jest 3-krotnie większa niż ta wynikająca ze wzoru.

3.3. OCZEKIWANA LICZBA PODPODZIAŁÓW GRAFU F

Podpodziały pewnego grafu $F = (V_F, E_F)$ powstają poprzez dodawanie wierzchołków stopnia 2 na istniejących krawędziach. Niech $v_1, v_2, \ldots, v_k$ będą wierzchołkami podpodziału F. Spośród jego wierzchołków wybieramy $|V_F|$ wierzchołków, które będą odpowiadać wierzchołkom grafu F - robimy to na $\binom{k}{|V_F|}$ sposobów. Następnie pozostałe wierzchołki rozdzielamy na $|E_F|$ grup, które tworzą krawędzie - robimy to na $\binom{k-|V_F|+|E_F|-1}{|E_F|-1}$

sposobów. Każdy podział daje nam rozbicie $k - |V_F| = k_1 + k_2 + \ldots + k_{|E_F|}$, gdzie k_i oznacza liczbę wierzchołków, które zostaną przydzielone do i-tej krawędzi. Następnie podział na dane rozbicie możemy zrealizować na $\frac{(k-|V_F|)!}{k_1!k_2!\ldots k_{|E_F|}!}$ sposobów. Wierzchołki w każdej krawędzi i możemy ustawić na $k_i!$ sposobów. Zatem różnych podpodziałów grafu F o k wierzchołkach jest:

$$\begin{aligned}\#F_k =& \binom{k}{|V_F|}\binom{k-|V_F|+|E_F|-1}{|E_F|-1}\frac{(k-|V_F|)!}{k_1!k_2!\ldots k_{|E_F|}!}k_1!k_2!\ldots k_{|E_F|}! = \\ =& \binom{k}{|V_F|}\binom{k-|V_F|+|E_F|-1}{|E_F|-1}(k-|V_F|)!\end{aligned} \tag{3.5}$$

Weźmy teraz graf G_1^{mn} i pogrupujmy jego wierzchołki w n m-elementowych grup. Następnie wybierzmy k z nich - stworzą one jeden z $\#F_k$ podpodziałów grafu F. Ustalmy krawędzie pomiędzy grupami tak, aby odpowiadały zadanemu podpodziałowi. Przez $H = (V_H, E_H)$ oznaczmy graf skierowany z odpowiednio wybranymi wierzchołkami z grup - tak aby stopnie wyjściowe nie przekraczały 1 i zapewnione były odpowiednie połączenia pomiędzy grupami - oraz krawędzie im odpowiadające ($V_H \subset [mn]$). Wówczas wierzchołki będące w grupach tworzących drogi w podpodziale mogą mieć:

1. jeden stopień wejściowy równy 1 i jeden stopień wyjściowy równy 1, gdy jeden sąsiad ma wyższy, a drugi niższy indeks
2. dwa stopnie wyjściowe równe 1, gdy obaj sąsiedzi mają niższe indeksy
3. jeden stopień wejściowy równy 2 albo dwa stopnie wejściowe równe 1, gdy obaj sąsiedzi mają wyższe indeksy

W 1. przypadku mamy m^2 możliwości, w 2. przypadku mamy $m(m-1)$ możliwości, a w 3. przypadku mamy m albo $m(m-1)$ możliwości wyboru wierzchołków w grupach. Oczekiwana liczba wierzchołków w każdej z trzech wyżej opisanych kategorii jest sobie równa i wynosi: $\frac{k-|V_F|}{3}$, ponieważ indeksy wierzchołka i jego sąsiadów mogą być w relacji ze sobą na 6 różnych sposobów i po dwa z nich spełniają każdy z warunków. Zatem oczekiwana liczba wyborów wierzchołków w grupach tak, aby wszystkie stopnie wejściowe nie były większe od 1 wynosi:

$$(m^2)^{\frac{k-|V_F|}{3}} \cdot (m(m-1))^{\frac{k-|V_F|}{3}} \cdot (m(m-1))^{\frac{k-|V_F|}{3}} \sim m^{2(k-|V_F|)}\,. \tag{3.6}$$

Komplementarnym przypadkiem jest sytuacja, gdy $\frac{k'}{3}$ wierzchołków ma stopień wejściowy równy 2. Oczekiwana liczba takich wyborów wynosi:

$$(m^2)^{\frac{k-|V_F|}{3}} \cdot (m(m-1))^{\frac{k-|V_F|}{3}} \cdot m^{\frac{k-|V_F|}{3}} \sim m^{\frac{5}{3}(k-|V_F|)} \tag{3.7}$$

Rozkład stopni pozostałych wierzchołków i ich liczby można przenieść z analizy danego grafu F.

W celu oceny prawdopodobieństwa wystąpienia danego podpodziału w G_1^{mn} należy także określić liczbę $C_{F_k}(v)$, dla $v \in V_H$. Dla każdego wierzchołka v mamy v wierzchołków o indeksach od niego niewiększych oraz $mn - v + 1$ wierzchołków o indeksach od niego niemniejszych. W rezultacie dostajemy prawdopodobieństwo $\frac{v}{mn} \cdot \frac{mn - v + 1}{mn - 1}$ zdarzenia, w którym jedna z krawędzi $(u, w) \in E_F$ spełnia $u \leq v \leq w$. Zatem oczekiwana liczba takich krawędzi dla całego grafu F_k wynosi:

$$\mathbb{E}[C_{F_k}(v)] = \frac{|E_{F_k}|}{mn} \sum_{v=1}^{mn} \frac{v(mn - v + 1)}{(mn)(mn - 1)} = |E_{F_k}| \frac{(mn + 1)(mn + 2)}{6(mn)(mn - 1)} \,. \tag{3.8}$$

Możemy teraz oszacować ostatni czynnik Twierdzenie 5:

$$\mathbb{E}\left[\sum_{v \in V_{F_k}} C_{F_k}(v)^2 / v\right] \sim \left(\frac{|E_{F_k}|}{6}\right)^2 \frac{|V_{F_k}|}{mn} H_{mn} \sim \frac{|E_{F_k}|^2 |V_{F_k}|}{6} \frac{\log(mn)}{mn} \tag{3.9}$$

Ustalmy indeksy grup $1 \leq l_1, l_2, \ldots, l_k \leq n$ ($\mathcal{L}_{n,k}$ oznacza zbiór grup) i obliczmy oczekiwaną liczbę podpodziałów grafu F z wykorzystaniem k wierzchołków w G_1^{mn}.

$$\begin{aligned}
\mathbb{E}[F_k] &=_\Theta \sum_{\mathcal{L}_{n,k'}} \#F_k \cdot \left(m^{2k'} \cdot (1!)^{\frac{k'}{3}} + m^{\frac{5}{3}k'} \cdot (2!)^{\frac{k'}{3}}\right) \cdot \frac{\exp\left(k^3 \frac{\log(mn)}{mn}\right)}{2^{k'} m^{k'} l_1 \cdot \ldots \cdot l_{k'}} \cdot \mathbb{E}[F] \\
&= \sum_{\mathcal{L}_{n,k'}} \#F_k \frac{m^{k'} + m^{\frac{2}{3}k'} \cdot 2^{\frac{k'}{3}}}{2^{k'} \cdot l_1 \cdot \ldots \cdot l_{k'}} \cdot \exp\left(k^3 \frac{\log(mn)}{mn}\right) \cdot \mathbb{E}[F] \\
&=_\Theta \#F_k \cdot \frac{m^{k'} + m^{\frac{2}{3}k'} \cdot 2^{\frac{k'}{3}}}{2^{k'}} \cdot \exp\left(k^3 \frac{\log(mn)}{mn}\right) \cdot \mathbb{E}[F] \cdot \log(n)^{k'} \,,
\end{aligned} \tag{3.10}$$

gdzie $k' = k - |V_F|$.

Zauważmy, że w tym przypadku rozważaliśmy dowolne ciągi indeksów, a nie tylko ciągi rosnące. Wynika to z tego, że podczas wyznaczania oczekiwanej liczby podpodziałów zostało to rozważone. Ostatecznie możemy wyrazić oczekiwaną liczbę podpodziałów H w grafie F jako sumę:

$$\mathbb{E}[H] = \sum_{i=|V_F|}^{n} \mathbb{E}[F_i] \tag{3.11}$$

Wyznaczmy teraz wartości dla konkretnych grafów F.

H = podpodział K_5

$$\mathbb{E}[H] =_\Theta \sum_{k=5}^{n} \frac{k^5(k+4)!}{5!\cdot 9!} \cdot \frac{m^{k-5} + m^{\frac{2}{3}(k-5)} \cdot 2^{\frac{k-5}{3}}}{2^{k-5}} \cdot \exp\left(k^3 \frac{\log(mn)}{mn}\right) \cdot \mathbb{E}[K_5] \cdot \log(n)^k$$

$$=_\Theta \sum_{k=5}^{n} \frac{k^9}{k!} \cdot \left(\left(\frac{m}{2}\right)^{k+5} + \left(\frac{m}{2}\right)^{\frac{2k+20}{3}}\right) \cdot \exp\left(k^3 \frac{\log(mn)}{mn}\right) \cdot \left(H_n^2\right)^5 \cdot \log(n)^{k-5} \tag{3.12}$$

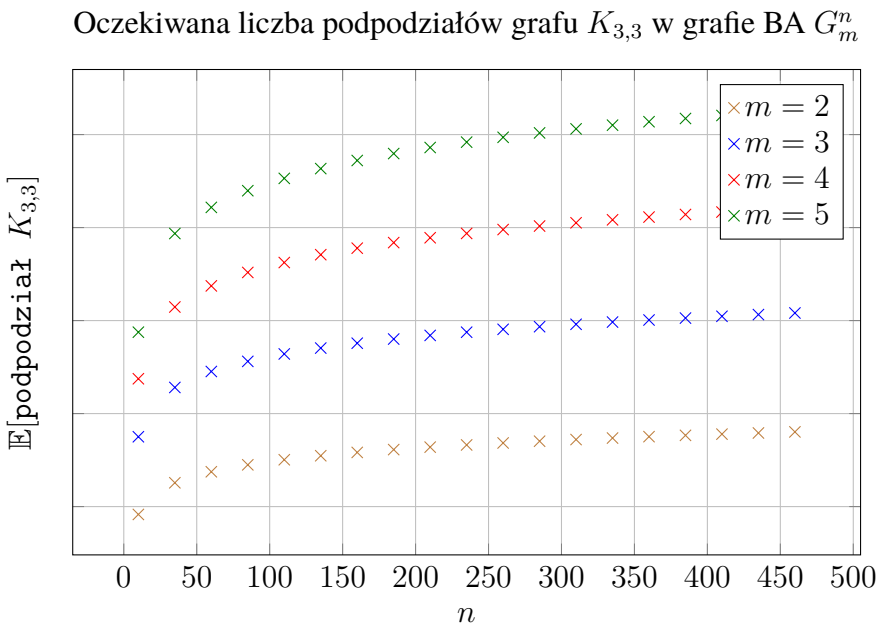
H = podpodział $K_{3,3}$

$$\mathbb{E}[H] =_\Theta \sum_{k=6}^{n} \frac{k^6(k+2)!}{6!\cdot 8!} \cdot \frac{m^{k-6} + m^{\frac{2}{3}k-6} \cdot 2^{\frac{k-6}{3}}}{2^{k-6}} \cdot \exp\left(k^3 \frac{\log(mn)}{mn}\right) \cdot \mathbb{E}[K_{3,3}] \cdot \log(n)^k$$

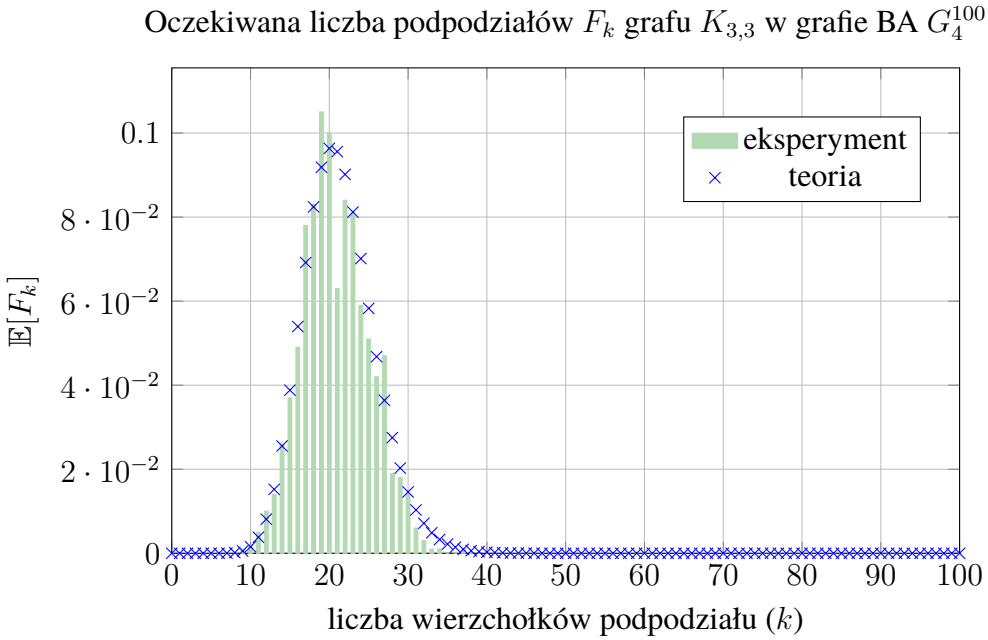
$$=_\Theta \sum_{k=6}^{n} \frac{k^8}{k!} \cdot \left(\left(\frac{m}{2}\right)^{k+3} + \left(\frac{m}{2}\right)^{\frac{2k+15}{3}}\right) \cdot \exp\left(k^3 \frac{\log(mn)}{mn}\right) \cdot \left(H_n^{\frac{3}{2}}\right)^6 \cdot \log(n)^{k-6} \tag{3.13}$$

3.4. OBSERWACJE I WNIOSKI

1. Oczekiwana liczba grafów K_5 w grafie BA jest wielokrotnie mniejsza niż grafów $K_{3,3}$.
2. Wraz ze wzrostem liczby wierzchołków w grafie, oczekiwana liczba grafów K_5 i $K_{3,3}$ w nim zawartych rośnie wolno i jest ograniczona z góry. Wynika to z Twierdzenia 1.
3. Liczba podpodziałów zadanego grafu zależy od liczby wierzchołków i parametru m grafu G_m^n, w którym je zliczamy. Tendencje Wzoru 3.13 zostały zwizualizowane na Rysunku 3.4. Nie przedstawia on dokładnych wartości, ponieważ wynik został uzyskany w notacji Θ.
4. Rozkład oczekiwanej liczby podpodziałów o zadanej liczbie wierzchołków nie jest jednostajny. Osiąga duże wartości dla podpodziałów o małej liczbie wierzchołków. Zjawisko to można zaobserwować na Rysunku 3.5. Przedstawia on eksperymentalnie wyznaczony rozkład liczby podpodziałów o zadanej liczbie wierzchołków oraz teoretyczne przewidywania. Ponieważ teoretyczna formuła jest w postaci Θ to wartości eksperymentalne i teoretyczne zostały znormalizowane. Eksperyment wykorzystał algorytm testowania planarności grafu Boyera-Myrvolda. Zwraca on informację, czy graf jest planarny oraz, w przypadku negatywnej odpowiedzi, podpodział powodujący nieplanarność. Pozwala to na sprawdzenie wielkości jednego z potencjalnie wielu podpodziałów w wygenerowanym grafie.



Rys. 3.4: Wizualizacja teoretycznych wartości oczekiwanej liczby podpodziałów $K_{3,3}$ w grafie G_m^n. Oś pionowa jest wyskalowana logarytmicznie.



Rys. 3.5: Eksperymentalne wyznaczenie oczekiwanej liczby podpodziałów grafu $K_{3,3}$ o zadanej liczbie wierzchołków w grafie BA G_4^{100}. Wykonano 1000 powtórzeń.

3.5. ODCHYLENIE STANDARDOWE WYKONANYCH EKSPERYMENTÓW

Definicja 28. *Odchylenie standardowe jest miarą określającą rozproszenie wartości w zbiorze danych $x_1, \ldots, x_n$ względem ich średniej arytmetycznej $\overline{x}$. Wyraża je wzór:*

$$\sigma_x = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})} \,. \tag{3.14}$$

Twierdzenie 6 (Centralne Twierdzenie Graniczne)**.** *Niech (Ω, σ, P) będzie przestrzenią probabilistyczną, a $X_1 \ldots, X_n$ będą niezależnymi zmiennymi losowymi zdefiniowanymi na Ω z tą samą wartością oczekiwaną μ i tym samym skończonym odchyleniem standardowym σ oraz $S_n = \sum_{i=1}^{n} X_i$. Wówczas, gdy $n \to \infty$ to znormalizowana suma S_n zbiega do rozkładu normalnego (według rozkładu):*

$$Z_n \overset{d}{=} \frac{S_n - \mathbb{E}[S_n]}{\sigma_{S_n}} \overset{d}{=} \frac{S_n - n\mu}{\sigma\sqrt{n}} \overset{d}{=} \frac{\overline{X} - \mu}{\frac{\sigma}{\sqrt{n}}} \xrightarrow{d} \mathcal{N}(0,1) \,. \tag{3.15}$$

W celu lepszego zrozumienia oczekiwanej liczby K_5 i $K_{3,3}$ zawartych w grafie Barabási—Albert, możemy określić odchylenie standardowe. Opisuje ono, jak bardzo wartości zmiennej losowej są rozproszone wokół jej wartości oczekiwanej. W celu jego wyznaczenia zastosujemy Twierdzenie 6. Dla każdej liczby badanych wierzchołków przeprowadzono serię eksperymentów wyznaczających liczbę danych grafów. W każdej z nich możemy wybrać pewną liczbę n różnych podzbiorów wartości i ich średnie traktować jako niezależne zmienne losowe o tej samej wartości oczekiwanej i tym samym skończonym odchyleniu standardowym. Następnie obliczyć odchylenia standardowe w podzbiorach. Wówczas na tej podstawie możemy wyznaczyć odchylenie standardowe zmiennej losowej opisującej liczbę K_5 czy $K_{3,3}$ w grafie BA. Wartości te zostały zaznaczone na wykresach 3.2 i 3.3.

4. WSKAŹNIK PLANARNOŚCI

Analizując obliczenia przeprowadzone w poprzednim rozdziale, możemy zauważyć, że parametry wpływające na prawdopodobieństwo wystąpienia podpodziału powodującego nieplanarność grafu można rozpatrywać w dwóch skalach - lokalnej i globalnej. W skali lokalnej ważny jest rozkład stopni w grafie - wierzchołki o większej liczbie sąsiadów mogą na więcej sposobów uczestniczyć w stworzeniu podpodziału. W skali globalnej kluczowa jest liczba wierzchołków n oraz parametr m grafu Barabási—Albert G_m^n.

Na podstawie określonych lokalnych i globalnych parametrów możemy określić lokalną i globalną metrykę, które następnie posłużą do ulepszenia opisanych w drugim rozdziale algorytmów i zaproponowania nowej metody kolorowania grafu.

4.1. DEFINICJA

Weźmy spójny graf $G_m^n = (V, E)$ oraz krawędź $\{u, v\} \in E$. Zdefiniujmy funkcje:

$$\tau_{G_m^n}(\{u, v\}) = \deg_{G_m^n}(u) \cdot \deg_{G_m^n}(v) \tag{4.1}$$

$$\tau(G_m^n) = \left(\sum_{\{u,v\} \in E} \tau_{G_m^n}(\{u, v\}) \right) \cdot m \cdot \log(n) \tag{4.2}$$

Tak zdefiniowane funkcje przyjmują wartości dodatnie. Wartości większe w przypadku $\tau_{G_m^n}(\{u, v\})$ oznaczają, że krawędź $\{u, v\}$ ma większe prawdopodobieństwo bycia użytą w podgrafie będącym podpodziałem $K_{3,3}$ lub K_5. Wartości funkcji $\tau(G_m^n)$ dla grafu wskazują, jak duże są szanse na wystąpienie podpodziału powodującego nieplanarność w grafie. Odnoszą się one do własności lokalnej oraz wyprowadzonych w poprzednim rozdziale formuł (3.12, 3.13). Zaniechane zostały potęgi, ponieważ jak wynika z obserwacji, nie zmieniają się one w znaczny sposób przy małej zmianie liczby n.

4.2. ZASTOSOWANIE

Metryka zdefiniowana dla krawędzi grafu może zostać użyta jako waga w heurystykach opisanych w drugim rozdziale, zaś ta zdefiniowana dla całego grafu może posłużyć do oceny różnych konfiguracji grafu podczas włączania do niego kolejnych krawędzi. Na tej podstawie mogą być wybierane ich kolory. W następnych sekcjach zostaną opisane usprawnienia opisanych wcześniej metod oraz zostaną przedstawione nowe.

4.2.1. Usprawnienie heurystyki — drzewo rozpinające

Heurystyka drzewa rozpinającego zakładała wybór dowolnego drzewa rozpinającego danego grafu G_m^n. Zamiast dowolnego drzewa możemy wybrać drzewo minimalne, czyli takie, którego suma wag krawędzi będzie najmniejsza. Wybraną wagą dla krawędzi $\{u, v\} \in E_{G_m^n}$ będzie wartość $\tau_{G_m^n}(u, v)$. Znajdywanie drzewa warto rozpocząć w „centrum" grafu - od wierzchołka o największym stopniu. Takie ulepszenie powoduje, że drzewo składa się z krawędzi, które minimalizują prawdopodobieństwo powstania problematycznych podpodziałów przy późniejszym maksymalizowaniu podgrafu.

Algorithm 6: MinimalneDrzewoRozpinające

```
Dane wejściowe: spójny graf prosty G = (V_G, E_G)
Wynik: drzewo rozpinające graf G
V_T ← V_G;
E_T ← ∅;
kolejka ← ∅; // kolejka priorytetowa
odwiedzone ← {};
dla każdego v ∈ V_G wykonuj
    odwiedzone[v] = NIE;
koniec
v_0 ← 1;
dla każdego v ∈ V_G wykonuj
    jeżeli deg_G(v) > deg_G(v_0) wtedy
        v_0 ← v;
    koniec
koniec
umieść parę (NULL, v_0) w kolejce z wagą 0;
dopóki kolejka ≠ ∅ wykonuj
    (u, v) ← wyciągnij parę o minimalnej wadze;
    jeżeli odwiedzone[v] = NIE wtedy
        odwiedzone[v] = TAK;
        E_T ← E_T ∪ {u, v};
        dla każdego w ∈ N_G(v) wykonuj
            jeżeli odwiedzone[w] = NIE wtedy
                c ← deg_G(v) · deg_G(w);
                umieść parę (v, w) w kolejce z wagą c;
            koniec
        koniec
    koniec
koniec
zwróć (V_T, E_T);
```

Jednym z algorytmów znajdujących minimalne drzewo rozpinające jest algorytm Prima. Pseudokod tego rozwiązania prezentuje się na Listingu 6. Jest to algorytm zachłanny,

który przy użyciu kolejki priorytetowej, iteracyjnie wybiera kolejne krawędzie grafu, aż do momentu uzyskania drzewa rozpinającego zawierającego wszystkie wierzchołki grafu (warunek z linii 15). Pierwsza do kolejki trafia krawędź „wirtualna". Jak wspomniano wcześniej, zawiera ona wierzchołek o największym stopniu (jego wybieranie odbywa się w liniach 8-13). Następnie wkładane są do niej krawędzie prowadzące do nieodwiedzonych wierzchołków wraz z odpowiadającymi im wagami (linie 20-25). Krawędzie wyciągane są z niej i przetwarzane w kolejności rosnących wag - zapewnia to działanie kolejki priorytetowej (linia 16). Złożoność czasowa tego algorytmu zależy od implementacji kolejki priorytetowej. W przypadku kopca binarnego wynosi ona $O(|E_{G_m^n}| \log(|V_{G_m^n}|))$, ponieważ operacje na kopcu zajmują logarytmiczny czas względem liczby elementów na nim, a aktualizacja kopca odbywa się dla każdej krawędzi grafu. Obliczanie wag krawędzi (linia 22) odbywa się w czasie stałym. Złożoność pamięciowa jest liniowa względem liczby wierzchołków grafu - maksymalnej liczby wierzchołków znajdujących się w kolejce.

4.2.2. Usprawnienie heurystyki — struktura „kaktus"

Konstruowanie struktury „kaktus" zaczyna się od wyboru trójkątów K_3. W opisanej wcześniej wersji, kolejność ich wyboru była dowolna. Usprawnienie polega na wyborze ich w kolejności rosnących sum wag krawędzi trójkątów. To samo dotyczy kolejności wyboru krawędzi łączących niepołączone wierzchołki z cyklami. Zaktualizowany algorytm wykorzystujący tę heurystykę został opisany na Listingu 7. Znaczącą zmianą jest generowanie wszystkich trójkątów (linie 6 - 13), a dopiero później dodawanie ich w posortowanej (linia 14) kolejności oraz sortowanie krawędzi grafu przed ich dodawaniem (linia 26). Motywacja tego działania jest taka sama jak w poprzednim przypadku. Modyfikacja ta potrzebuje sortowania, co zwiększa złożoność czasową działania algorytmu o ten czynnik. Przykładowo, algorytm sortowania przez scalanie ma złożoność czasową w najgorszym przypadku $O(n \log(n))$, gdzie n jest liczbą elementów do posortowania.

4.2.3. Usprawnienie maksymalizacji planarnego podgrafu

Procedura maksymalizacji podgrafu także może zostać poprawiona dzięki użyciu zdefiniowanych wag krawędzi. Mogą one posłużyć do ustalenia kolejności włączania kolejnych krawędzi do podgrafu. Najskuteczniejsze okazuje się włączanie krawędzi w kolejności malejących wartości wag. Intuicją stojącą za taką kolejnością jest to, że krawędzie dodawane później w mniejszym stopniu wpływają na brak planarności. Zmieniony algorytm prezentuje się na Listingu 8. Od poprzedniej wersji różni się dodaniem sortowania krawędzi w linii 4. Zmiana ta generuje dodatkową złożoność czasową, której powodem jest sortowanie krawędzi opisane w poprzedniej podsekcji.

Algorithm 7: WażonaStrukturaKaktusowa

```
Dane wejściowe: spójny graf prosty G = (V_G, E_G)
Wynik: struktura trójkątna zawarta w G
1  V_C ← ∅;
2  E_C ← ∅;
3  komponenty ← {};
4  dla każdego v ∈ V_G wykonuj
5      komponenty[v] ← v;
6  trójkąty ← {};
7  dla każdego t_1 ∈ V_G wykonuj
8      dla każdego t_2 ∈ N_G(t_1) wykonuj
9          dla każdego t_3 ∈ N_G(t_1) wykonuj
10             jeżeli t_2 = t_3 wtedy
11                 kontynuuj;
12             jeżeli {t_2, t_3} ∈ E_G wtedy
13                 trójkąty ← trójkąty ∪ {(t_1, t_2, t_3)};
14 sortuj(trójkąty, rosnąco, klucz:
     (t_1, t_2, t_3) → deg_G(t_1) · deg_G(t_2) + deg_G(t_1) · deg_G(t_3) + deg_G(t_2) · deg_G(t_3));
15 dla każdego (t_1, t_2, t_3) ∈ trójkąty wykonuj
16     k_1 ← komponenty[t_1];
17     k_2 ← komponenty[t_2];
18     k_3 ← komponenty[t_3];
19     jeżeli k_1 ≠ k_2 oraz k_1 ≠ k_3 oraz k_2 ≠ k_3 wtedy
20         E_C ← E_C ∪ {{t_1, t_2}, {t_1, t_3}, {t_2, t_3}};
21         l_2 ← komponenty[t_2];
22         l_3 ← komponenty[t_3];
23         dla każdego v ∈ V_G wykonuj
24             jeżeli komponenty[v] = l_2 ∨ komponenty[v] = l_3 wtedy
25                 komponenty[v] ← k_1;
26 sortuj(E_G, rosnąco, klucz: (v_1, v_2) → deg_G(v_1) · deg_G(v_2));
27 dla każdego {v, u} ∈ E_G wykonuj
28     jeżeli komponenty[v] ≠ komponenty[u] wtedy
29         E_C ← E_C ∪ {{v, u}};
30         k ← komponenty[v];
31         l ← komponenty[u];
32         dla każdego w ∈ V_G wykonuj
33             jeżeli komponenty[w] = l wtedy
34                 komponenty[w] = k;
35 zwróć (V_C, E_C)
```

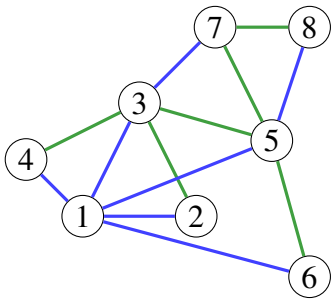
Algorithm 8: WażonaMaksymalizacjaPodgrafu

Dane wejściowe: bazowy spójny graf prosty $G = (V_G, E_G)$
Wynik: wybrany planarny podgraf $H = (V_H, E_H)$
Wynik: maksymalny planarny podgraf H^*

1 $V_{H^*} \leftarrow V_H$;
2 $E_{H^*} \leftarrow E_H$;
3 $E \leftarrow E_G \setminus E_H$;
4 sortuj(E, malejąco, klucz: $(v_1, v_2) \to \deg_G(v_1) \cdot \deg_G(v_2)$);
5 dla każdego $e \in E$ wykonuj
6 $\quad E_{H^*} \leftarrow E_{H^*} \cup \{e\}$;
7 $\quad$ jeżeli jestPlanarny(H^*) = NIE wtedy
8 $\quad\quad E_{H^*} \leftarrow E_{H^*} \setminus \{e\}$;
9 $\quad$ koniec
10 koniec
11 zwróć (V_{H^*}, E_{H^*});

4.3. NATURALNE KOLOROWANIE

Metodą kolorowania niekorzystającą z wybierania maksymalnych planarnych podgrafów może być metoda bazująca na strukturze grafu Barabási—Albert. Polega ona na przypisaniu każdej z m krawędzi, które są dodawane do grafu wraz z każdym nowym wierzchołkiem innego koloru. Założeniem umożliwiającym takie działanie jest to, aby ustalony graf początkowy można było rozłożyć na nie więcej niż m planarnych podgrafów. Wówczas opisane kolorowanie wygeneruje m drzew - nie biorąc pod uwagę krawędzi w grafie początkowym - a drzewa są planarne. Kolory mogą, ale nie muszą być ustalane w trakcie generowania grafu. Gdy graf jest już wygenerowany, możemy przejść po jego wierzchołkach i różne kolory ustalić krawędziom prowadzącym do wierzchołków o mniejszym indeksie. Narzut czasowy tej metody w przypadku kolorowania w trakcie generowania grafu jest stały — przydzielamy kolory m krawędziom. W przypadku kolorowania już wygenerowanego grafu złożoność jest liniowa względem liczby krawędzi w grafie - każdą z nich odwiedzamy dwa razy.



Rys. 4.1: Wizualizacja naturalnego kolorowania grafu G_2^8 dwoma kolorami.

4.4. KOLOROWANIE GRAFU Z ZADANĄ LICZBĄ KOLORÓW

Opisane wyżej podejścia nie wykorzystują globalnych parametrów grafu. Możemy zmodyfikować zadanie kolorowania w taki sposób, że zadajemy liczbę kolorów, jakie możemy użyć, i chcemy tak rozdzielić krawędzie, aby zminimalizować liczbę *crossing edges*. Wówczas wraz z rozrastaniem się grafu możemy oceniać metrykę dla każdego monochromatycznego podgrafu indukowanego przez krawędzie o jednym z zadanych kolorów i tak wybierać kolor nowej krawędzi, aby te metryki minimalizować. Ponieważ tworzące się podgrafy mogą mieć zróżnicowaną gęstość, warto zastąpić parametr m średnim stopniem wierzchołka. Ważną cechą metryki τ dla grafu jest łatwość, z jaką możemy ją aktualizować. Dodanie kolejnej krawędzi wymaga aktualizacji czynników związanych z sąsiadami jej końców oraz tych związanych z globalnymi własnościami grafu. Z uwagi na stały średni stopień wierzchołka w grafie BA taka aktualizacja metryki w średnim przypadku będzie wykonywała się w czasie stałym.

Problemem może być brak spójności podgrafów. W przypadku jej braku, należałoby śledzić każdą komponentę osobno, co mogłoby spowodować liniowy czas aktualizacji metryki za każdym razem. W celu rozwiązania tego problemu można przyjąć regułę nadrzędną, która ustala kolor krawędzi na ten, który powoduje spójność któregoś podgrafu w przypadku jej braku. Gdy spójność każdego z podgrafów będzie już zapewniona, wybór odbywać się będzie na podstawie wartości metryk. W zdecydowanej większości, kolor wybrany na podstawie braku planarności pokrywałby się z tym wybranym na podstawie metryki. Aby spójność każdego z podgrafów dało się zachować, liczba krawędzi dodawana z każdym wierzchołkiem musi być większa od zadanej liczby kolorów. Jest to założenie, które nie ogranicza nam zastosowania algorytmu, ponieważ naturalne kolorowanie opisane w poprzedniej sekcji dowodzi, że więcej niż m kolorów do poprawnego kolorowania nie potrzeba. Ważnym czynnikiem jest sprawdzanie spójności podgrafów, które może zostać zrealizowane przy użyciu struktury danych zbiorów rozłącznych, opisanych wcześniej. W związku z tym złożoność czasowa tego algorytmu jest w średnim przypadku liniowa względem wielkości grafu początkowego.

Pseudokod algorytmu został przedstawiony na Listingu 9. Zaczyna się on od inicjalizacji grafów, zbiorów rozłącznych oraz wartości metryk dla każdego z kolorów (linie 2 - 6). Następnie dla każdej krawędzi aktualizowana jest waga metryki dla każdego z kolorów (linie 8 - 16). Kolejną fazą jest sprawdzenie, czy krawędź łączy różne komponenty w grafie o danym kolorze (linie 17 - 23) — jeśli tak, to jest to szukany kolor. W przeciwnym przypadku, wybierany jest ten kolor, któremu odpowiada graf o najmniejszej wartości metryki (linie 24 - 26). Na koniec aktualizowane są zmienne dotyczące wybranego koloru (linie 27 - 30).

Algorithm 9: PreferencyjneKolorowanieZadanąLiczbąKolorów

Dane wejściowe: graf BA $G = (V_G, E_G)$
Dane wejściowe: liczba kolorów k
Wynik: kolorowanie krawędzi grafu $c : E_G \to [k]$

$c \leftarrow \emptyset$;
dla każdego $i \in [k]$ wykonuj
  $H_i = (V_{H_i}, E_{H_i}) \leftarrow (\emptyset, \emptyset)$; // podgraf o krawędziach koloru i
  $K_i \leftarrow \{\}$; // struktura zbiorów rozłącznych podgrafu i
  $S_i \leftarrow 0$; // suma wartości τ dla krawędzi z podgrafu i
koniec
dla każdego $\{u, v\} \in E_g$ wykonuj
  dla każdego $i \in [k]$ wykonuj
    dla każdego $w \in N_{H_i}(u)$ wykonuj
      $S_i^* \leftarrow S_i - \deg_{H_i}(w) \cdot \deg_{H_i}(u) + \deg_{H_i}(w) \cdot (\deg_{H_i}(u) + 1)$;
    koniec
    dla każdego $w \in N_{H_i}(v)$ wykonuj
      $S_i^* \leftarrow S_i^* - \deg_{H_i}(w) \cdot \deg_{H_i}(v) + \deg_{H_i}(w) \cdot (\deg_{H_i}(v) + 1)$;
    koniec
    $S_i^* \leftarrow S_i^* + (\deg_{H_i}(u) + 1) \cdot (\deg_{H_i}(v) + 1)$;
  koniec
  $i^* \leftarrow -1$;
  dla każdego $i \in [k]$ wykonuj
    jeżeli u, v nie są w jednym zbiorze w K_i wtedy
      połącz zbiory z v, u w K_i;
      $i^* \leftarrow i$;
    koniec
  koniec
  jeżeli $i^* \neq -1$ wtedy
    $i^* \leftarrow \underset{i \in [k]}{\operatorname{argmin}} \; S_i^* \cdot \frac{|E_{H_i}|}{|V_{H_i}|} \cdot \log |V_{H_i}|$;
  koniec
  $V_{i^*} \leftarrow V_{i^*} \cup \{u, v\}$;
  $E_{i^*} \leftarrow E_{i^*} \cup \{\{u, v\}\}$;
  $S_{i^*} \leftarrow S_{i*}^*$;
  $c[\{u, v\}] \leftarrow i^*$;
koniec
zwróć c;

5. PORÓWNANIE ALGORYTMÓW

W tym rozdziale porównamy wcześniej opisane algorytmy: wybierania największego planarnego podgrafu, kolorowania grafu w taki sposób, aby zachować planarne monochromatyczne podgrafy przy użyciu minimalnej liczby kolorów, oraz kolorowania grafu z określoną liczbą kolorów, aby zminimalizować liczbę przecięć krawędzi w podgrafach. Algorytmy zostały zaimplementowane w języku *C++*. Struktura grafu korzysta z list sąsiedztwa. W celu testowania planarności grafu została użyta biblioteka *OGDF* [8]. Zbadana zostanie jakość generowanych rozwiązań, a także złożoność czasowa ich działania.

Niezbędnym elementem tej analizy jest metoda generowania grafu Barabási—Albert, która zostanie przedstawiona i zbadana w następnej sekcji.

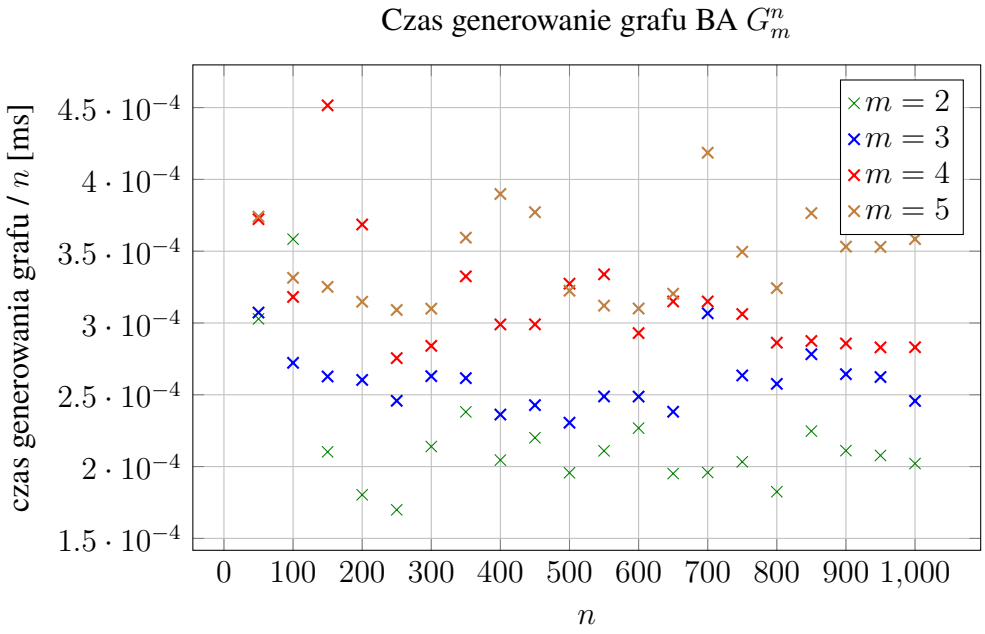
5.1. GENEROWANIE GRAFU LOSOWEGO BARABÁSI—ALBERT

Sposób efektywnego generowania grafu Barabási—Albert G_m^n o zadanych parametrach m oraz n został opisany w pracy Vladimira Batagelj oraz Ulrika Brandes „Efficient generation of large random networks" [3]. Tutaj został przedstawiony na Listingu 10. Polega on na stworzeniu tablicy o długości $2mn$, której każde dwa kolejne elementy reprezentują wierzchołki połączone krawędzią. W rozważanym modelu prawdopodobieństwo wybrania wierzchołka do stworzenia nowej krawędzi jest proporcjonalne do jego stopnia, dlatego potrzebujemy efektywnego sposobu na wybór takiego wierzchołka. Możemy zauważyć, że dla rozpatrywanego wierzchołka o indeksie i ($0 \leq i < n$), liczba wystąpień indeksu j ($0 \leq j < i$) w tablicy jest równa aktualnemu stopniowi wierzchołka j w grafie stworzonym na wierzchołkach $\{0, \ldots, i\}$ i krawędziach opisanych przez $2mi$ pierwszych elementów tablicy. Zatem dla każdego indeksu wierzchołka, zaczynając od 0 i kończąc na $(n-1)$, należy wybrać sekwencyjnie, losowo i niezależnie, m wierzchołków i dodać takie pary do generowanej tablicy (linie 3-7). Dla i-tego wierzchołka i j-tej krawędzi parę dla wierzchołka i wybieramy z zakresu 0-$(2mi+j)$. Na koniec należy przenumerować indeksy tak, aby były zawarte w zbiorze $[n]$ oraz stworzyć na podstawie tablic zbiór wierzchołków (linie 9-12) i krawędzi (linie 13-16). Taka implementacja wymaga liniowej względem wielkości grafu pamięci — pamiętamy tablicę rozmiaru $2mn$. Ma także liniową względem wielkości grafu złożoność pamięciową — dla każdego z n wierzchołków losujemy m końców nowych krawędzi. Rysunek 5.1 przedstawia eksperymentalnie zbadaną złożoność czasową działania tego algorytmu. Jego wyniki pokrywają się z teoretycznymi rozważaniami.

Algorithm 10: GenerowanieGrafuBA

Dane wejściowe: liczba wierzchołków n
Dane wejściowe: minimalny stopień wierzchołka m
Wynik: graf BA $G = (V_G, E_G)$

```
1  M ← []; // tablica o długości 2mn
2  dla każdego v = 0 ... (n − 1) wykonuj
3  |  dla każdego i = 0 ... (m − 1) wykonuj
4  |  |  M[2(vm + i)] ← v;
5  |  |  wylosuj r ze zbioru {0, ..., 2(vm + i)};
6  |  |  M[2(vm + i)] ← M[r];
7  |  koniec
8  koniec
9  V_G ← ∅;
10 dla każdego i = 1 ... n wykonuj
11 |  V_G ← V_G ∪ {i};
12 koniec
13 E_G ← ∅;
14 dla każdego i = 0 ... (mn − 1) wykonuj
15 |  E_G ← E_G ∪ {M[2i] + 1, M[2i + 1] + 1};
16 koniec
17 zwróć (V_G, E_G);
```



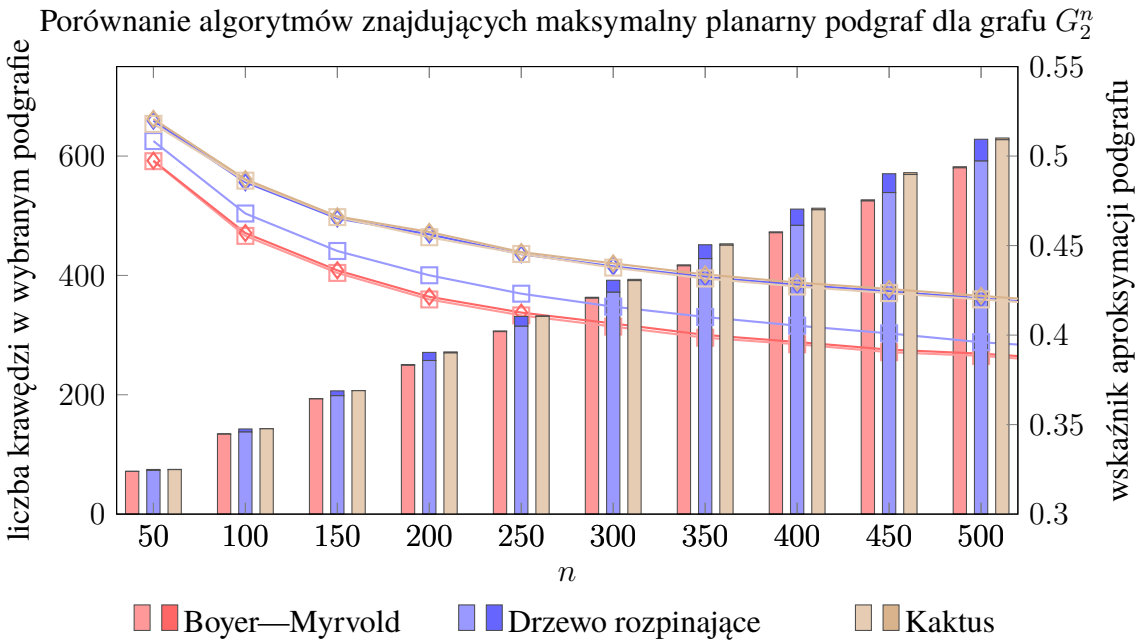
Rys. 5.1: Eksperymentalne wyznaczenie złożoności obliczeniowej algorytmu generowania grafu BA G_m^n. Dla każdego punktu wykonano 100000 powtórzeń. Oś pionowa reprezentuje czas w milisekundach podzielony przez liczbę wierzchołków n.

5.2. WYBÓR MAKSYMALNEGO PLANARNEGO PODGRAFU

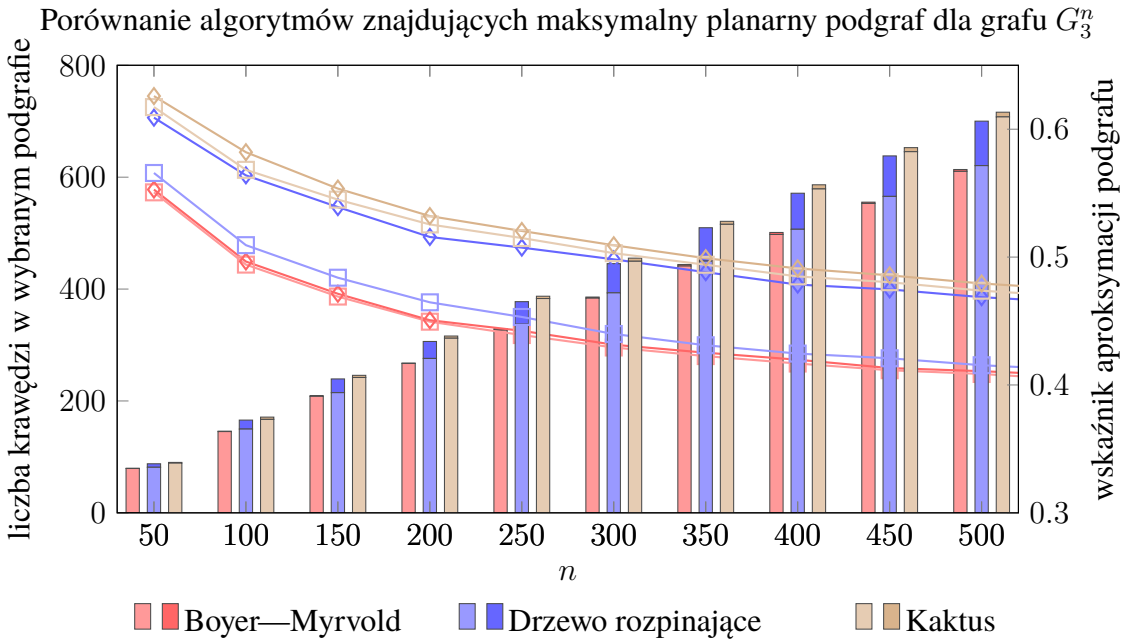
W obecnej sekcji porównamy jakość generowanych rozwiązań oraz czas ich generowania przez algorytmy wybierające maksymalny planarny podgraf, które zostały przedstawione w 2-im i 4 rozdziale. Został wykonany eksperyment, który polegał na wygenerowaniu po 100 grafów BA G_m^n o różnych wartościach m i n oraz sprawdzeniu, jak duże planarne podgrafy mogą zostać wybrane. Zastosowano jedną z trzech opisanych heurystyk oraz zmaksymalizowano liczbę krawędzi w znajdowanym podgrafie opisanym algorytmem.

Wyniki jakości generowanych rozwiązań zostały przedstawione na Rysunkach 5.2, 5.3, 5.4, 5.5. Lewa oś opisuje liczbę krawędzi w podgrafie i odnosi się do słupków na wykresie, a prawa oś opisuje wskaźnik aproksymacji podgrafu i odnosi się do krzywych na wykresie. Kwadraty na nich oznaczają wyniki metod pierwotnych, a romby metod wzbogaconych o wskaźnik planarności. Jaśniejsze kolory symbolizują wynik uzyskany przez rozwiązanie pierwotne, a kolory ciemniejsze symbolizują wynik uzyskany przez algorytm wzbogacony o metrykę planarności (w przypadku serii) albo zysk z zastosowania metryki planarności (w przypadku słupków). Łatwo ocenić, że najgorsze wyniki osiąga algorytm bazujący na teście planarności Boyera—Myrvolda, a najlepsze algorytm wykorzystujący strukturę „kaktus". Zastosowanie wskaźnika planarności najbardziej poprawia wyniki uzyskiwane przez algorytm korzystający z drzewa rozpinającego — podnosi liczbę krawędzi w planarnym podgrafie o kilkadziesiąt/kilkaset w zależności od parametru m. Liczba ta rośnie wraz ze wzrostem tego parametru. Znacznie mniejszy zysk obserwowany jest w przypadku rozwiązania zaczynającego od wyboru struktury kaktusowej — wraz ze wzrostem parametru m zysk rośnie od kilkunastu do kilkudziesięciu krawędzi. Zysk w postaci kilku dodatkowych krawędzi obserwowany jest dla rozwiązania bazującego na teście planarności. Wskaźnik aproksymacji oczywiście odzwierciedla wyniki mierzone w liczbie krawędzi. Warto jednak zauważyć, że w każdym z przypadków jego wartość maleje wraz ze wzrostem liczby wierzchołków n.

Wyniki porównujące czas działania algorytmów zostały przedstawione na Rysunkach 5.6, 5.7, 5.8, 5.9. Krzyżyki na nich oznaczają czasy działania algorytmów w wersji pierwotnej, a plusy czasy działania algorytmów wzbogaconych o wskaźnik planarności. Asymptotyki otrzymane eksperymentalnie pokrywają się z przewidywaniami teoretycznymi z kilkoma wyjątkami. Heurystyka, będąca modyfikacją testu planarności Boyera—Myrvolda, w eksperymentach potwierdza liniową złożoność czasową. Rośnie ona wraz z wartościami parametru m. Wyjątkiem jest wartość 2. Może to wynikać z tego, że prawdopodobieństwo wystąpienia podpodziału K_5 czy $K_{3,3}$ w grafie BA o tym parametrze jest małe. W przypadku heurystyki bazującej na drzewie rozpinającym i minimalnym drzewie rozpinającym wyniki teoretyczne i praktyczne pokrywają się. Odpowiednio są to złożoności $O(|V_{G_m^n}|)$ oraz $O(|V_{G_m^n}| \log(|V_{G_m^n}|))$. Eksperymenty nie pokryły się z teorią dla algorytmów bazujących na strukturze „kaktus". Średni czas ich działania w eksperymencie prezentuje asymptotykę

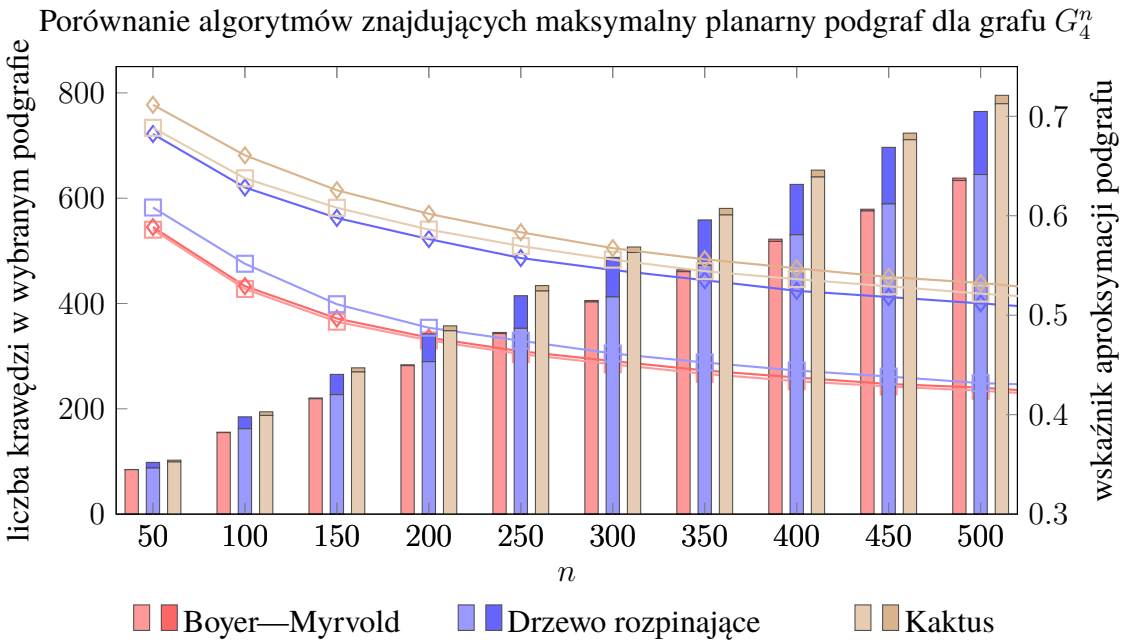


Rys. 5.2: Eksperymentalne porównanie algorytmów generowania maksymalnych podgrafów planarnych dla grafu BA G_2^n.

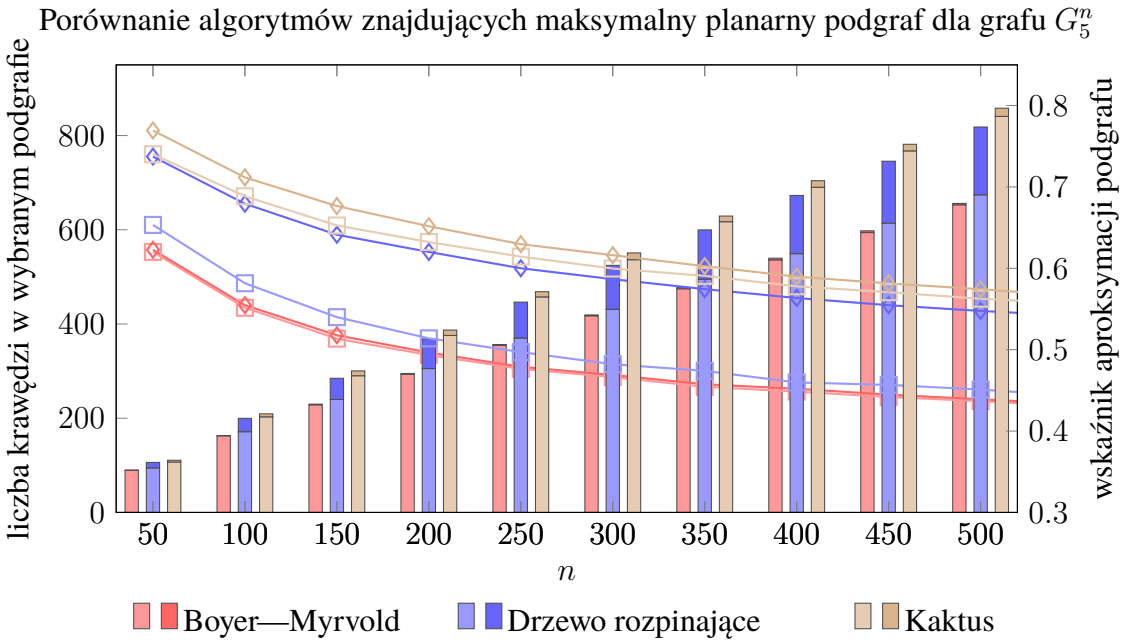


Rys. 5.3: Eksperymentalne porównanie algorytmów generowania maksymalnych podgrafów planarnych dla grafu BA G_3^n.

$|V_{G_m^n}|(\log|V_{G_m^n}|)^2$. Rozbieżności mogą być powodowane przez fakt, że średni stopień w badanym grafie jest stały, przez co liczba trójkątów jest znacząco mniejsza, niż sześcian liczby wierzchołków w grafie. Ponadto użycie wag w celu wyznaczenia kolejności wybierania trójkątów wymaga sortowania, co dodaje kolejny narzut czasowy — jest to widoczne na wykresie. Ostatni wykres badający czas zaproponowanych algorytmów bada rozwiązania

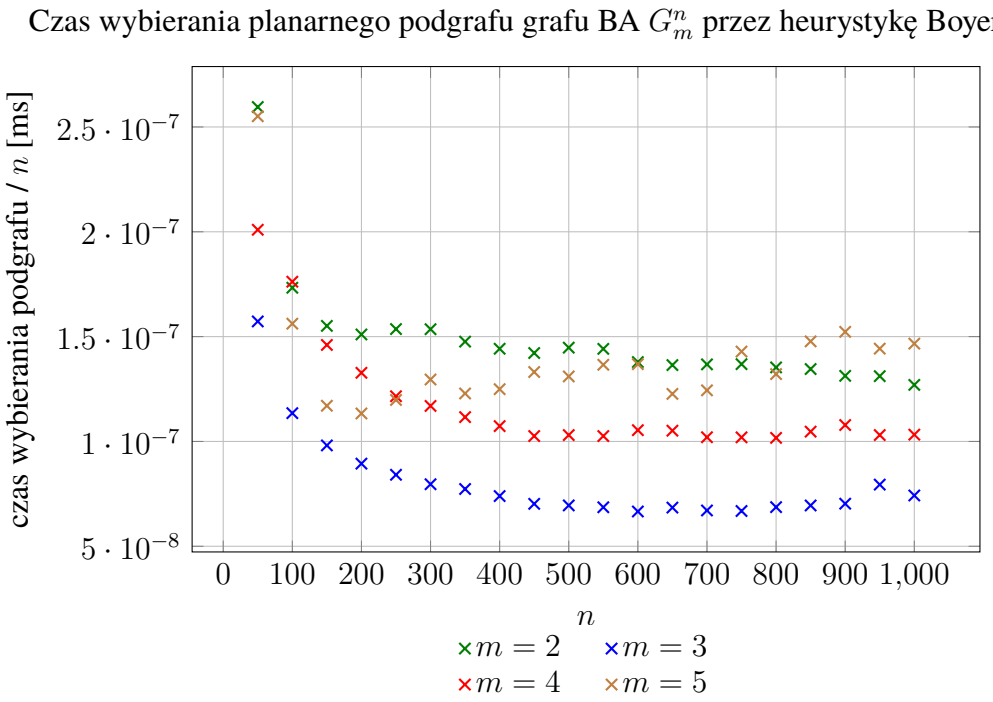


Rys. 5.4: Eksperymentalne porównanie algorytmów generowania maksymalnych podgrafów planarnych dla grafu BA G_4^n.

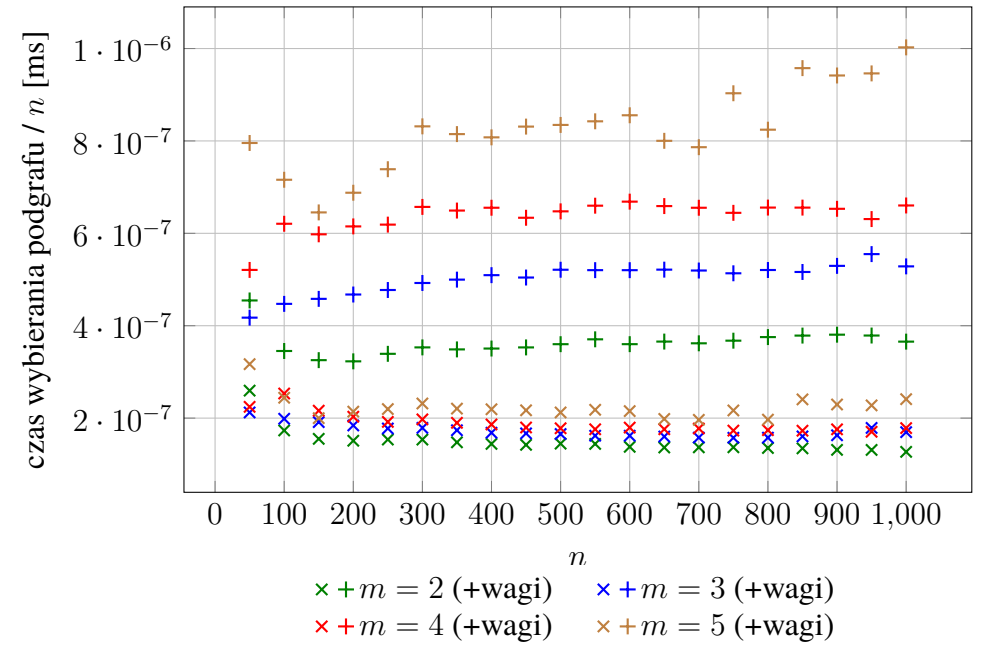


Rys. 5.5: Eksperymentalne porównanie algorytmów generowania maksymalnych podgrafów planarnych dla grafu BA G_5^n.

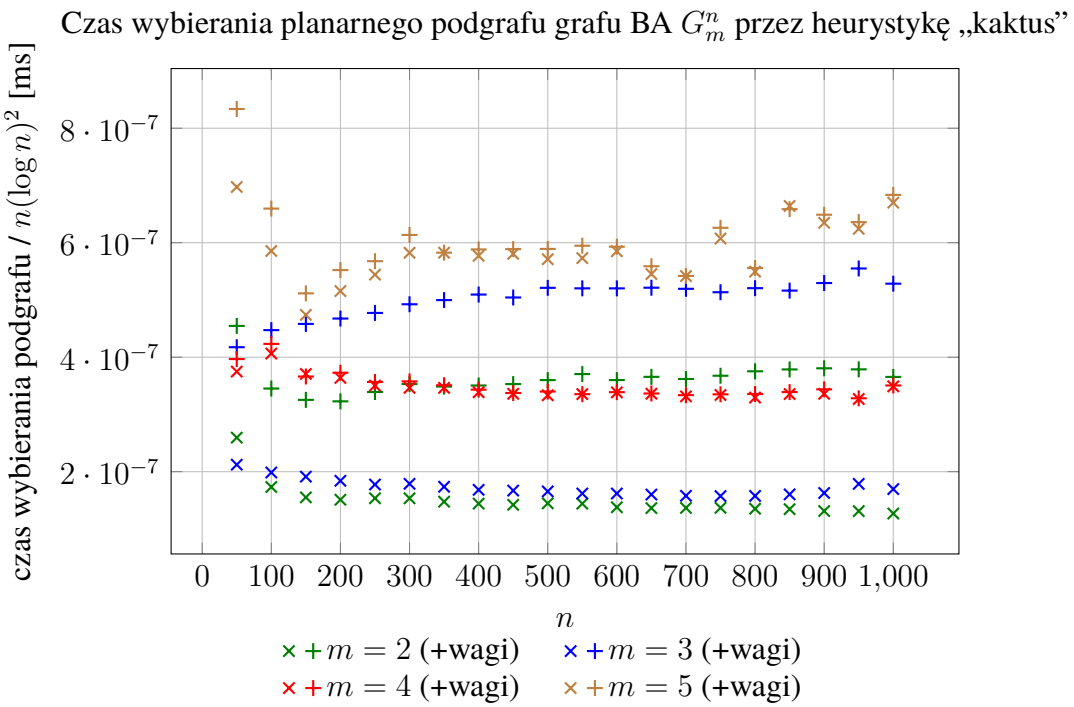
wybierające maksymalny planarny podgraf, czyli użycie heurystyki w celu wybrania planarnego podgrafu i zmaksymalizowanie go. W tym przypadku wyniki teoretyczne oraz te wyznaczone eksperymentalnie są zbieżne i prezentują kwadratową względem liczby wierzchołków w grafie asymptotykę. Najszybciej w średnim przypadku działa metoda korzystająca z drzewa rozpinającego, a najwolniej ta modyfikująca test planarności.



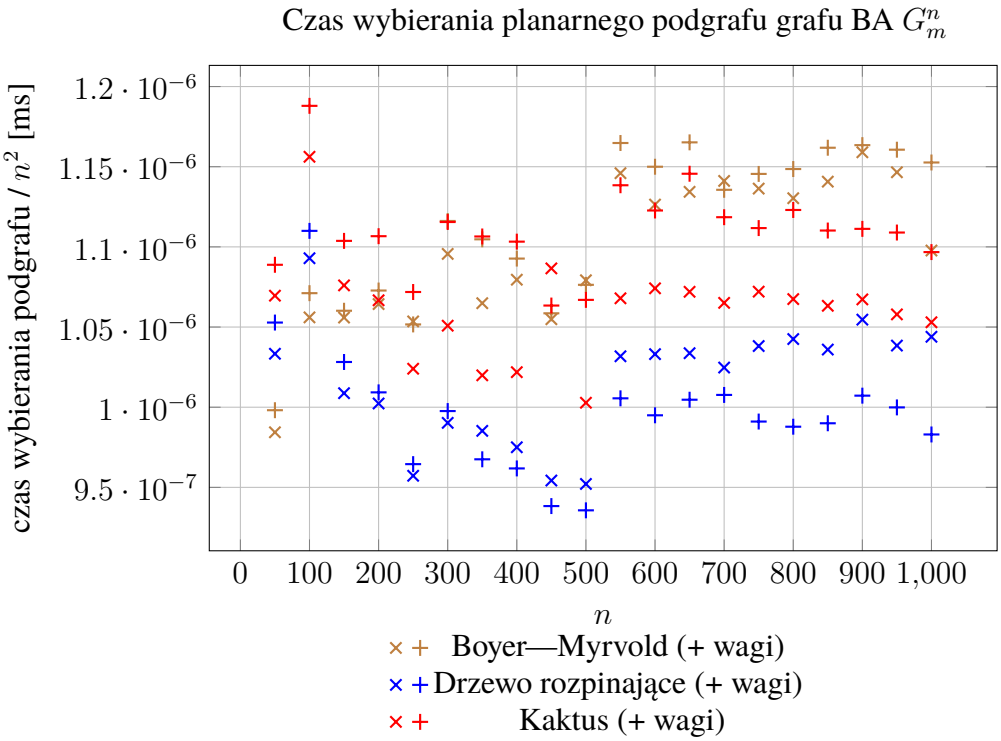
Rys. 5.6: Eksperymentalne wyznaczenie złożoności obliczeniowej heurystyki Boyer–Myrvold w grafie BA G_m^n. Oś pionowa reprezentuje czas w milisekundach podzielony przez liczbę wierzchołków n.



Rys. 5.7: Eksperymentalne wyznaczenie złożoności obliczeniowej heurystyki drzewo rozpinające w grafie BA G_m^n. Oś pionowa reprezentuje czas w milisekundach podzielony przez liczbę wierzchołków n.



Rys. 5.8: Eksperymentalne wyznaczenie złożoności obliczeniowej heurystyki „kaktus” w grafie BA G_m^n. Oś pionowa reprezentuje czas w milisekundach podzielony przez liczbę wierzchołków i logarytm w potędze 2 — $n(\log n)^2$

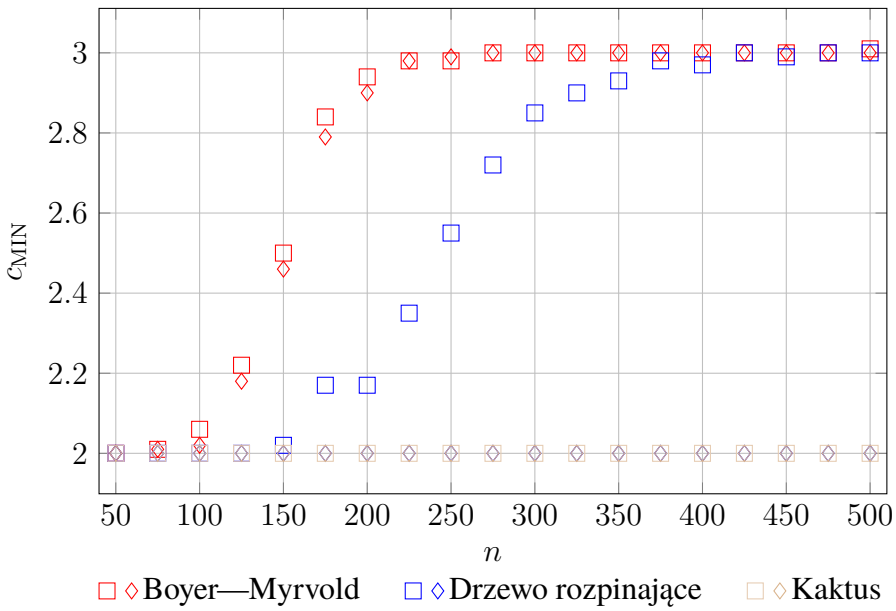


Rys. 5.9: Eksperymentalne wyznaczenie złożoności obliczeniowej algorytmu wybierania maksymalnego planarngeo podgrafu grafu BA G_4^n. Oś pionowa reprezentuje czas w milisekundach podzielony przez kwadrat liczby wierzchołków n^2.

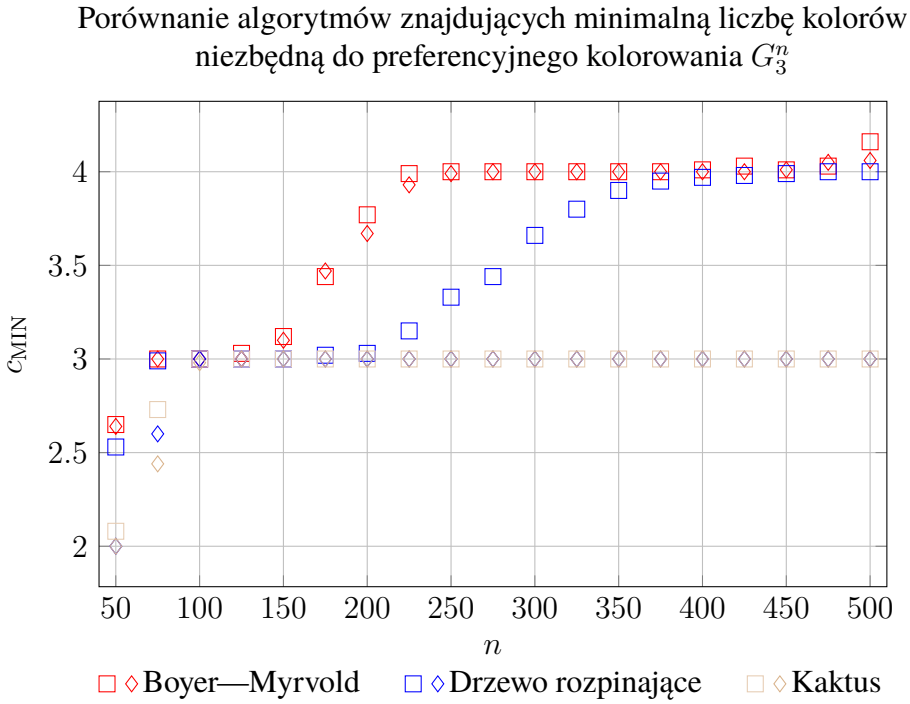
5.3. KOLOROWANIE MINIMALNĄ LICZBĄ KOLORÓW

Ważnym wskaźnikiem oceny jakości algorytmów jest minimalna liczba kolorów niezbędna, do określonego w części teoretycznej, preferencyjnego kolorowania. Wiąże się ona także z grubością kolorowanego grafu. Został wykonany eksperyment analogiczny do tego z poprzedniej sekcji, w którym zamiast wielkości wybranego podgrafu, zbadana została minimalna liczba kolorów. Wyniki tego eksperymentu zostały przedstawione na Rysunkach 5.10, 5.11, 5.12. Przedstawiają one po dwie serie wyników dla każdej heurystyki. Serie z kwadracikami oznaczają metodę pierwotną, zaś te z rombami metodę wzbogaconą o wskaźnik planarności. Ranking jakości algorytmów w tym porównaniu kształtuje się analogicznie do tego z poprzedniego eksperymentu. Warto zauważyć, że dla każdej wartości m, metoda bazująca na „kaktusach", radzi sobie znacznie lepiej od pozostałych. Gdy wartości m rosną, metoda bazująca na drzewie rozpinającym pokazuje swoją wyższość nad metodą, opartą o test planarności. Wyraźnie widać, że dla pewnych przedziałów liczb wierzchołków w grafie G_m^n, wpływ wskaźnika planarności jest znaczący (z wyjątkiem metody bazującej na teście planarności, która znacząco się nie poprawia). Bardzo dobrze obrazuje to Rysunek 5.12, gdzie dla $n \in [100, 250]$ ulepszony algorytm generuje rozwiązania, które potrzebują średnio jednego koloru mniej do pokolorowania grafu w poprawny, opisany na początku, sposób. Takie przedziały powtarzają się dla kolejnych wartości n, a wraz ze wzrostem n i m, różnice liczby kolorów rosną.

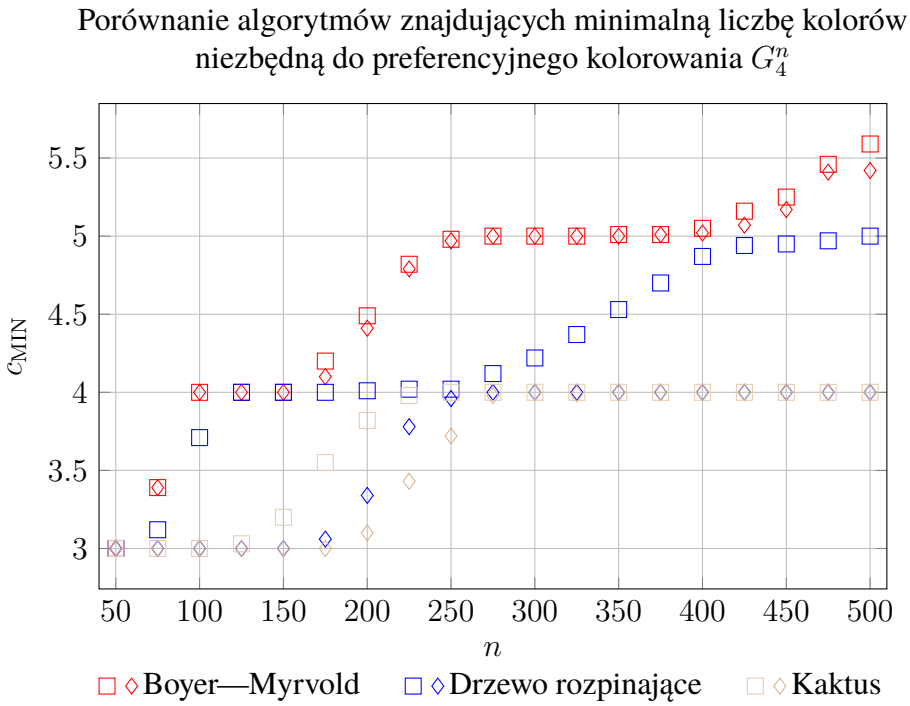
Porównanie algorytmów znajdujących minimalną liczbę kolorów niezbędną do preferencyjnego kolorowania G_2^n



Rys. 5.10: Eksperymentalne porównanie algorytmów wyznaczania minimalnej liczby kolorów niezbędnych do preferencyjnego kolorowania grafu BA G_2^n.



Rys. 5.11: Eksperymentalne porównanie algorytmów wyznaczania minimalnej liczby kolorów niezbędnych do preferencyjnego kolorowania grafu BA G_3^n.



Rys. 5.12: Eksperymentalne porównanie algorytmów wyznaczania minimalnej liczby kolorów niezbędnych do preferencyjnego kolorowania grafu BA G_4^n.

Aby dokładniej prześledzić zmianę minimalnej liczby kolorów, powstała Tabela 5.1. Przedstawia ona zakresy liczby wierzchołków n, w których dany algorytm potrzebuje co najmniej c_{MIN} kolorów do preferencyjnego kolorowania grafu G_m^n. Symbol ∞ oznacza, że nie ustalono górnej granicy przedziału — jest ona powyżej kilkudziesięciu tysięcy. Łatwo zauważyć, że algorytmy bazujące na teście planarności radzą sobie najgorzej z rozważanym problemem, algorytm bazujący na dowolnym drzewie rozpinającym daje lepsze efekty, ale dopiero jego wzbogacenie o indeks planarności daje najbardziej jakościowe wyniki. Z problemem najlepiej radzi sobie algorytm korzystający ze struktury „kaktus" a dodanie do niego wag powoduje uzyskanie najlepszych wyników w całym eksperymencie. Ważną obserwacją jest fakt, że dla grafów o dużej liczbie wierzchołków algorytmy używające heurystyki „kaktus" i minimalnego drzewa rozpinającego z przedstawionymi wagami wymagają liczby kolorów równej parametrowi m. Jest to taki sam rezultat jak ten otrzymany w wyniku naturalnego kolorowania opisanego na koniec 2-go rozdziału pracy. Dla grafów o kilkudziesięciu/kilkuset wierzchołkach, w zależności od wartości m, możliwe jest kolorowanie, które potrzebuje mniej niż m kolorów. W tych przypadkach użycie zdefiniowanego indeksu planarności znacząco podnosi jakość rozwiązania.

m	c_{MIN}	Boyer—Myrvold		drzewo rozpinające		struktura „kaktus"	
		bez wag	z wagami	bez wag	z wagami	bez wag	z wagami
2	2	$[19, 146]$	$[19, 149]$	$[19, 240]$	$[19, \infty]$	$[19, \infty]$	$[19, \infty]$
	3	$[147, 980]$	$[150, 1000]$	$[241, \infty]$	–	–	–
3	2	$[14, 49]$	$[14, 49]$	$[14, 49]$	$[14, 49]$	$[14, 49]$	$[14, 49]$
	3	$[50, 170]$	$[50, 180]$	$[50, 270]$	$[50, \infty]$	$[50, \infty]$	$[50, \infty]$
	4	$[171, 600]$	$[181, 620]$	$[271, 1800]$	–	–	–
4	2	$[12, 27]$	$[12, 29]$	$[12, 27]$	$[12, 33]$	$[12, 32]$	$[12, 37]$
	3	$[28, 78]$	$[30, 79]$	$[28, 82]$	$[34, 227]$	$[33, 190]$	$[38, 244]$
	4	$[79, 198]$	$[80, 207]$	$[83, 338]$	$[228, \infty]$	$[190, \infty]$	$[244, \infty]$
	5	$[199, 480]$	$[208, 520]$	$[339, 1350]$	–	–	–
	6	$[481, 1250]$	$[521, 1300]$	$[1351, 2900]$	–	–	–
5	2	$[12, 21]$	$[12, 21]$	$[12, 21]$	$[12, 24]$	$[12, 22]$	$[12, 25]$
	3	$[22, 48]$	$[22, 50]$	$[22, 51]$	$[25, 79]$	$[23, 70]$	$[26, 87]$
	4	$[49, 136]$,	$[51, 138]$	$[52, 142]$	$[80, 500]$	$[71, 378]$	$[88, 588]$
	5	$[136, 231]$	$[139, 238]$,	$[143, 396]$	$[501, \infty]$	$[379, \infty]$	$[589, \infty]$
	6	$[232, 478]$	$[239, 483]$	$[397, 1200]$	–	–	–
	7	$[479, 1150]$	$[484, 1200]$	$[1201, 6300]$	–	–	–

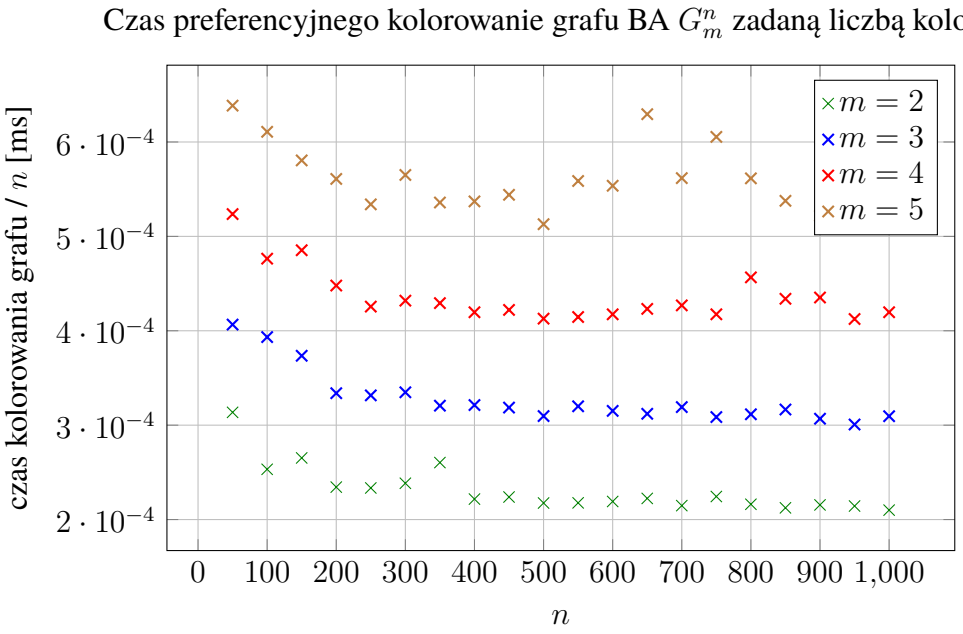
Tabela 5.1: Zakresy liczby wierzchołków n, w których dany algorytm potrzebuje co najmniej c_{MIN} kolorów do preferencyjnego kolorowania grafu G_m^n.

5.4. KOLOROWANIE ZADANĄ LICZBĄ KOLORÓW

Na koniec sprawdzimy jakość algorytmu preferencyjnego kolorowania z zadaną liczbą kolorów. Przeprowadzono eksperyment, który polegał na wygenerowaniu po 100 grafów BA G_m^n o różnych wartościach m i n oraz zadanej liczbie kolorów z zakresu $[m]$. Sprawdzono czas działania programu kolorującego podgrafy oraz zakresy liczby wierzchołków, w których średnia liczba przecięć grafu jest ustalona. Wyznaczanie jej polegało na usuwaniu podzbiorów krawędzi o rozmiarach od 1 do n i sprawdzaniu, czy bez nich graf jest planarny.

W wyniku eksperymentu wyznaczono średni czas kolorowania. Przedstawia to wykres na Rysunku 5.13. Potwierdza on złożoność wyznaczoną teoretycznie — $O(|V_{G_m^n}|)$.

W celu porównania zakresów liczby wierzchołków zestawiono je w Tabeli 5.2 z parametrem grafu m, zadaną liczbą kolorów c oraz maksymalną średnią liczbą *crossing edges*. Symbol ∞ ma takie samo znaczenie jak w poprzedniej tabeli. Wynika z niej, że opracowana metoda daje wyniki nie gorsze, niż kolorowanie naturalne, które ograniczało liczbę potrzebnych kolorów dla grafu G_m^n do m. Ponadto dla grafów o kilkudziesięciu/kilkuset wierzchołkach generuje kolorowanie mniejszą liczbą kolorów, z tym że powoduje średnio kilka przecięć w każdym z monochromatycznych podgrafów. Wyniki tej metody są porównywalne do wyników metody korzystającej z heurystyki Boyer—Myrvold. Algorytmy bazujące na drzewach rozpinających i „kaktusach" generują lepsze wyniki. Jednak te trzy metody mają większą złożoność czasową, co w przypadku dużych grafów może być problemem.



Rys. 5.13: Eksperymentalne wyznaczenie złożoności obliczeniowej algorytmu preferencyjengo kolorowania grafu BA G_m^n z zadaną liczbą kolorów $m - 1$. Dla każdego punktu wykonano 1000 powtórzeń. Oś pionowa reprezentuje czas w milisekundach podzielony przez liczbę wierzchołków n.

m	c	średnia liczba przecięć grafu cr					
		0	(0,1]	(1,2]	(2,3]	(3,4]	(4,5]
2	1	$[0, 10]$	$[10, 16]$	$[17, 22]$	$[23, 28]$	$[29, 34]$	$[35, 42]$
	≥ 2	$[0, \infty]$	–	–	–	–	–
3	1	$[0, 7]$	$[8, 11]$	$[12, 13]$	$[14, 15]$	$[15, 16]$	$[17, 17]$
	2	$[0, 14]$	$[15, 30]$	$[31, 49]$	$[50, 64]$	$[65, 82]$	$[83, 98]$
	≥ 3	$[0, \infty]$	–	–	–	–	–
4	1	$[0, 6]$	$[7, 8]$	$[9, 9]$	$[10, 10]$	$[11, 11]$	$[12, 12]$
	2	$[0, 6]$	$[7, 14]$	$[15, 19]$	$[20, 23]$	$[24, 25]$	$[26, 26]$
	3	$[0, 12]$	$[13, 30]$	$[31, 57]$	$[58, 71]$	$[72, 94]$	$[95, 109]$
	≥ 4	$[0, \infty]$	–	–	–	–	–
5	1	$[0, 5]$	$[6, 6]$	$[7, 7]$	$[8, 8]$	$[9, 9]$	$[10, 10]$
	2	$[0, 8]$	$[9, 12]$	$[13, 16]$	$[17, 19]$	$[20, 21]$	$[22, 23]$
	3	$[0, 10]$,	$[11, 19]$	$[20, 43]$	$[44, 59]$	$[60, 71]$	$[72, 89]$
	4	$[0, 14]$	$[15, 43]$,	$[44, 113]$	$[114, 153]$	$[154, 182]$	$[182, 204]$
	≥ 5	$[0, \infty]$	–	–	–	–	–

Tabela 5.2: Zakresy liczby wierzchołków n, w których algorytm preferencyjnego kolorowania c kolorami generuje średnio co najwyżej cr przecięć grafu G_m^n.

6. ZASTOSOWANIE

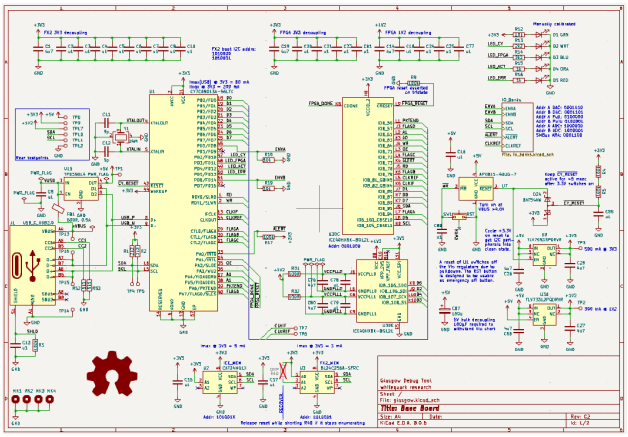
Przytoczona w poprzednich rozdziałach teoria i opisane algorytmy znajdują kilka zastosowań. Według mnie najciekawszym i najlepiej oddającym istotę problemu jest transformacja schematu elektronicznego (Rysunek 6.1) na projekt płytki drukowanej PCB (ang. *printed circuit board*) (Rysunek 6.3). Polega ona na zamianie abstrakcyjnych połączeń logicznych pomiędzy komponentami elektronicznymi w rzeczywiste połączenia elektryczne, które są możliwe do wykonania na płytce drukowanej.

Schemat elektroniczny możemy przedstawić jako graf prosty (Rysunek 6.2), w którym wierzchołki reprezentują komponenty, a krawędzie — połączenia między nimi. Planarność tak zdefiniowanego grafu jest jego najważniejszą własnością, ponieważ fizyczne połączenia nie mogą się przecinać — doprowadzałoby to do zwarć czy nieodpowiedniego funkcjonowania układu. Jednakże duża liczba wierzchołków i krawędzi w skomplikowanych układach powoduje, że zazwyczaj takie grafy planarne nie są. W celu rozwiązania tego problemu stosuje się dwie metody. Jedną z nich są płytki wielowarstwowe, czyli próba rozłożenia grafu na jak najmniejszą liczbę planarnych podgrafów, a następnie połączenie ich w całość przy użyciu przelotek(połączeń pomiędzy różnymi warstwami płytki). Inną metodą jest zastosowanie „mostów", czyli dodatkowych połączeń, które umożliwiają poprowadzenie połączenia „nad" już istniejącym połączeniem na jednej warstwie, korzystając z innej warstwy. Rozwiązania te odnoszą się do badanej w pracy grubości grafu oraz liczby jego przecięć. Warstwy odnoszą się do monochromatycznych podgrafów generowanych przez opisane algorytmy.

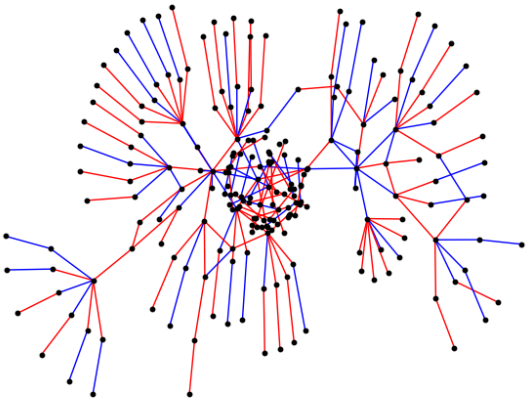
Rodzina grafów, na której badane były zaproponowane algorytmy, nie była przypadkowa, ponieważ jak się okazuje w [7], schematy elektroniczne z powodzeniem mogą być modelowane przez grafy losowe Barabási—Albert. Wykazują one własności bezskalowości oraz „preferencyjnego przyłączania". Jest to zgodne z intuicją, ponieważ wiele schematów zawiera kilka układów scalonych wykonujących główne zadania oraz bardzo dużo elementów „pomocniczych" (rezystory, kondensatory, cewki), odpowiadających za poprawne działanie tych pierwszych.

Z wyżej przytoczonych argumentów oraz wyników uzyskanych w rozdziale 5 wynika, że dokonane usprawnienia, mające swoje uzasadnienie w strukturze grafów BA, mają realny wpływ na proces tworzenia płytek drukowanych. Mogą przyczynić się do ich optymalizacji, zarówno pod względem kosztów produkcji (wraz z każdą kolejną warstwą cena PCB rośnie), jak i parametrów elektronicznych układu (wydłużanie długości ścieżek pomię-

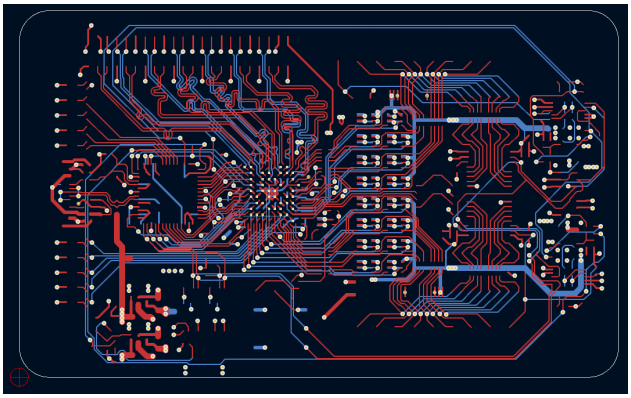
dzy komponentami w wyniku dodatkowych „mostów” wpływa negatywnie na szybkość transmisji sygnałów, zużycie energii czy poziom zakłóceń sygnału).



Rys. 6.1: Przykładowy schemat elektroniczny [1] - screen z programu KiCad.



Rys. 6.2: Graf reprezentujący powyższy schemat pokolorowany preferencyjnie.



Rys. 6.3: Układ dwuwarstwowej płytki drukowanej [1] - screen z programu KiCad.

PODSUMOWANIE

Celem pracy było opracowanie algorytmów do kolorowania krawędziowego grafów losowych Barabási—Albert, które przy zachowaniu planarności monochromatycznych podgrafów minimalizują liczbę kolorów lub przy zadanej liczbie kolorów minimalizują liczbę przecięć każdego z jednokolorowych podgrafów.

Przedstawiono trzy metody obecne w literaturze: zmodyfikowany test planarności Boyer–Myrvold, drzewo rozpinające oraz struktura „kaktus". Następnie została obliczona oczekiwana liczba podpodziałów grafów K_5 oraz $K_{3,3}$, które odgrywają główne role w Twierdzeniu Kuratowskiego o planarności grafu. Z przeprowadzonych obliczeń wynika, że oczekiwana liczba grafów K_5 i $K_{3,3}$ w grafie BA jest ograniczona z góry. Wraz ze wzrostem liczby wierzchołków, ich liczba rośnie tak jak liczby harmoniczne, rzędu odpowiednio: 2 i $\frac{3}{2}$, czyli $\frac{|E|}{|V|}$. Dodatkowo, rozkład oczekiwanej liczby podpodziałów o zadanej liczbie wierzchołków nie jest jednostajny. Osiąga duże wartości dla podpodziałów o liczbie wierzchołków stanowiącej nieduży ułamek wszystkich wierzchołków w grafie.

Kolejnym krokiem była definicja wskaźnika planarności. Został on zdefiniowany lokalnie dla krawędzi, jak i globalnie - dla całego grafu. Na podstawie lokalnego wskaźnika planarności zmodyfikowano wcześniej opisane algorytmy - użyto go do wyznaczania kolejności przetwarzania krawędzi. Opisano także alternatywne metody kolorowania : kolorowanie korzystające z globalnej definicji wskaźnika dla zadanej liczby kolorów oraz podejście korzystające z procesu generowania grafu Barabási—Albert. To ostatnie pozwala ograniczyć maksymalną liczbę kolorów niezbędnych do kolorowania grafu, co przekłada się na ograniczenie jego grubości: $t(G_m^n) \leq m$. Kolejnym etapem było eksperymentalne porównanie opisanych wcześniej metod i skonfrontowanie ich z teorią. Wynika z nich, że zaproponowane modyfikacje algorytmów znacząco zwiększały liczbę krawędzi w wybieranych maksymalnych planarnych podgrafach, nie zwiększając ich złożoności - przed i po modyfikacji wynosiła ona $O(|V_{G_m^n}|^2)$. Wyniki te przełożyły się na zmniejszenie niezbędnej liczby kolorów potrzebnych do preferencyjnego kolorowania. Najlepszą bazą do maksymalizacji podgrafu okazała się heurystyka „kaktus", a najgorszą modyfikacja testu planarności. Złożoność czasowa metod alternatywnych jest niższa i wynosi $O(|V_{G_m^n}|)$, ale algorytmy te generują gorsze rozwiązania.

Na koniec przedstawiono zastosowanie preferencyjnego kolorowania w procesie transformacji schematu elektronicznego na projekt płytki drukowanej.

BIBLIOGRAFIA

[1] Dostępne online (6.12.2024): `https://github.com/GlasgowEmbedded/glasgow/tree/main/hardware/boards/glasgow`.

[2] Barabási, A., Albert, R., *Emergence of scaling in random networks*, tom 9781400841356 (Princeton University Press, United States, 2011), str. 349–352. Publisher Copyright: © 1999 The American Physical Society.

[3] Batagelj, V., Brandes, U., *Efficient generation of large random networks*, Physical Review E—Statistical, Nonlinear, and Soft Matter Physics. 2005, tom 71, 3, str. 036113.

[4] Bollobás, B., Riordan, O.M., *Mathematical results on scale-free random graphs*, Handbook of graphs and networks: from the genome to the internet. 2003, str. 1–34.

[5] Boyer, J.M., Myrvold, W.J., *Simplified o (n) planarity by edge addition*, Graph Algorithms Appl. 2006, tom 5, str. 241.

[6] Călinescu, G., Fernandes, C.G., Finkler, U., Karloff, H., *A better approximation algorithm for finding planar subgraphs*, Journal of Algorithms. 1998, tom 27, 2, str. 269–302.

[7] i Cancho, R.F., Janssen, C., Solé, R.V., *Topology of technology graphs: Small world patterns in electronic circuits*, Physical Review E. 2001, tom 64, 4, str. 046119.

[8] Chimani, M., Gutwenger, C., Jünger, M., Klau, G.W., Klein, K., Mutzel, P., *The open graph drawing framework (ogdf).*, Handbook of graph drawing and visualization. 2013, tom 2011, str. 543–569.

[9] Chimani, M., Klein, K., Wiedera, T., *A note on the practicality of maximal planar subgraph algorithms*, w: *Graph Drawing and Network Visualization: 24th International Symposium, GD 2016, Athens, Greece, September 19-21, 2016, Revised Selected Papers 24* (Springer, 2016), str. 357–364.

[10] Garey, M.R., Johnson, D.S., *Computers and intractability*, tom 174 (freeman San Francisco, 1979).

[11] Mansfield, A., *Determining the thickness of graphs is np-hard*, Mathematical Proceedings of the Cambridge Philosophical Society. 1983, tom 93, 1, str. 9–23.

[12] Patrignani, M., *Planarity testing and embedding.* 2013.

SPIS RYSUNKÓW

SPIS PSEUDOKODÓW

SPIS TABEL

Dodatki

A. SKRYPT OBLICZAJĄCY SYMBOLICZNIE OCZEKIWANĄ LICZBĘ K_5 I $K_{3,3}$ W GRAFIE G^n_m

Dodatek przedstawia skrypt, który posłużył do wygenerowania formuł określających oczekiwaną liczbę grafów K_5 oraz $K_{3,3}$ w grafie Barabási—Albert G^n_m. Pierwsza funkcja służy do wyznaczania oczekiwanej liczby podgrafów o ustalonych wierzchołkach i krawędziach, których wynikiem jest pewien zbiór stopni wyjściowych. Generuje ona wszystkie możliwe dobory rozkładów wierzchołków w podanych grupach. Dwa kolejne listingi wizualizują wyznaczanie wartości docelowych.

```
(*
cases_ - lista, ktora reprezentuje grupy wierzcholkow;
    jej elementami sa listy, ktore zawieraja pary -
    liczba przyporzadkowan krawedzi wierzcholkom
    oraz stopnie wejsciowe wynikajace z takiego rozkladu

prob_ - funkcja, ktora oblicza prawdopodobienstwo
    wystapienie danego grafu w grafie BA
*)
SumUp[cases_, prob_] := Module[
    {combinations, results},

    combinations = Tuples[cases];

    results = Times @@@ (Table[
        {
            Times @@ combination[[All, 1]],
            prob[Flatten[combination[[All, 2]]]]
        },
        {combination, combinations}
    ]);

    Total[results]
]
```

```
K5SubgraphProbability[degrees_, m_] :=
    (Times @@ (Factorial /@ degrees)) / (2^10 * m^10)

K5ExpNumCoeff[m_] :=
    FullSimplify[SumUp[
        {
            { { m*(m-1)*(m-2)*(m-3), {} } },
            { { m*m*(m-1)*(m-2), {1} } },
            { { m*(m-1)*m*(m-1), {1,1} },
              { m*m*(m-1),{2} } },
            { { m*(m-1)(m-2)*m, {1,1,1} },
              { 3*m*(m-1)*m, {1,2} },
              { m*m, {3} } },
            { { m*(m-1)*(m-2)*(m-3), {1,1,1,1} },
              { 6*m*(m-1)*(m-2), {1,1,2} },
              { 4*m*(m-1), {1,3} },
              { 3*m*(m-1), {2,2} },
              { m, {4} } }
        },
        K5SubgraphProbability[#, m] &
    ]]

K5ExpectedNumber[n_, m_] :=
    K5ExpNumCoeff[m] * HarmonicNumber[n, 2]^5 / (5!)^2
```

```
K33SubgraphProbability[degrees_, m_] :=
    (Times @@ (Factorial /@ degrees)) / (2^9 * m^9)

K33ExpNumCoeff1[m_] := FullSimplify[SumUp[
    {
        { { m*(m-1)*(m-2), {} } },
        { { m*(m-1)*(m-2), {} } },
        { { m*(m-1)*(m-2), {} } },
        { { m*(m-1)*(m-2), {1,1,1} },
          { 3*m*(m-1), {1,2} },
          { m, {3} } },
        { { m*(m-1)*(m-2), {1,1,1} },
          { 3*m*(m-1), {1,2} },
          { m, {3} } },
        { { m*(m-1)*(m-2), {1,1,1} },
          { 3*m*(m-1), {1,2} },
          { m, {3} } }
    },
    K33SubgraphProbability[#, m] &
]]

K33ExpNumCoeff2[m_] := FullSimplify[SumUp[
    {
        { { m*(m-1)*(m-2), {} } },
        { { m*(m-1)*(m-2), {} } },
        { { m*m*(m-1), {1,1} },
          { m*m, {2} } },
        { { m*(m-1)*m, {1} } },
        { { m*(m-1)*(m-2), {1,1,1} },
          { 3*m*(m-1), {1,2} },
          { m, {3} } },
        { { m*(m-1)*(m-2), {1,1,1} },
          { 3*m*(m-1), {1,2} },
          { m, {3} } }
    },
    K33SubgraphProbability[#, m] &
]]

K33ExpNumCoeff3And4[m_] := FullSimplify[SumUp[
    {
        { { m*(m-1)*(m-2), {} } },
```

```
        { { m*(m-1)*m, {1} } },
        { { m*(m-1)*m, {1} } },
        { { m*(m-1)*m, {1} } },
        { { m*(m-1)*(m-2), {1,1,1} },
          { 3*m*(m-1), {1,2} },
          { m, {3} } },
        { { m*(m-1)*(m-2), {1,1,1} },
          { 3*m*(m-1), {1,2} },
          { m, {3} } }
    },
    K33SubgraphProbability[#, m] &
]]

K33ExpNumCoeff5And10[m_] := FullSimplify[SumUp[
    {
        { { m*(m-1)*(m-2), {} } },
        { { m*(m-1)*(m-2), {} } },
        { { m*m*(m-1), {} },
          { m*m, {2} } },
        { { m*m*(m-1), {1,1}},
          { m*m, {2} } },
        { { m*m*(m-1), {1,1} },
          { m*m, {2} } },
        { { m*(m-1)*(m-2), {1,1,1} },
          { 3*m*(m-1), {1,2} },
          { m, {3} } }
    },
    K33SubgraphProbability[#, m] &
]]

K33ExpNumCoeff6And7And8And9[m_] := FullSimplify[SumUp[
    {
        { { m*(m-1)*(m-2), {} } },
        { { m*(m-1)*m, {1} } },
        { { m*(m-1)*m, {1} } },
        { { m*m*(m-1), {1,1} },
          { m*m, {2} } },
        { { m*m*(m-1), {1,1} },
          { m*m, {2} } },
        { { m*(m-1)*(m-2), {1,1,1} },
          { 3*m*(m-1), {1,2} },
```

```
            { m, {3} } }
    },
    K33SubgraphProbability[#, m] &
]]

K33ExpectedNumber[n_, m_] :=
    FullSimplify[Total[
        {
            K33ExpNumCoeff1[m],
            K33ExpNumCoeff2[m],
            2 * K33ExpNumCoeff3And4[m],
            2 * K33ExpNumCoeff5And10[m],
            4 * K33ExpNumCoeff6And7And8And9[m]
        }
    ]] * HarmonicNumber[n, 3/2]^6 / (6!)^(3/2)
```

B. WYZNACZANIE OCZEKIWANEJ LICZBY $\mathbf{K_{3,3}}$ W GRAFIE $\mathbf{G^n_m}$

W tym dodatku przybliżymy szczegóły wyznaczania oczekiwanej liczby podgrafów $K_{3,3}$ w grafie G^n_m. Z uwagi na inną topologię grafu, w porównaniu do rozważanej w trzecim rozdziale kliki, liczba różnych połączeń między grupami jest większa.

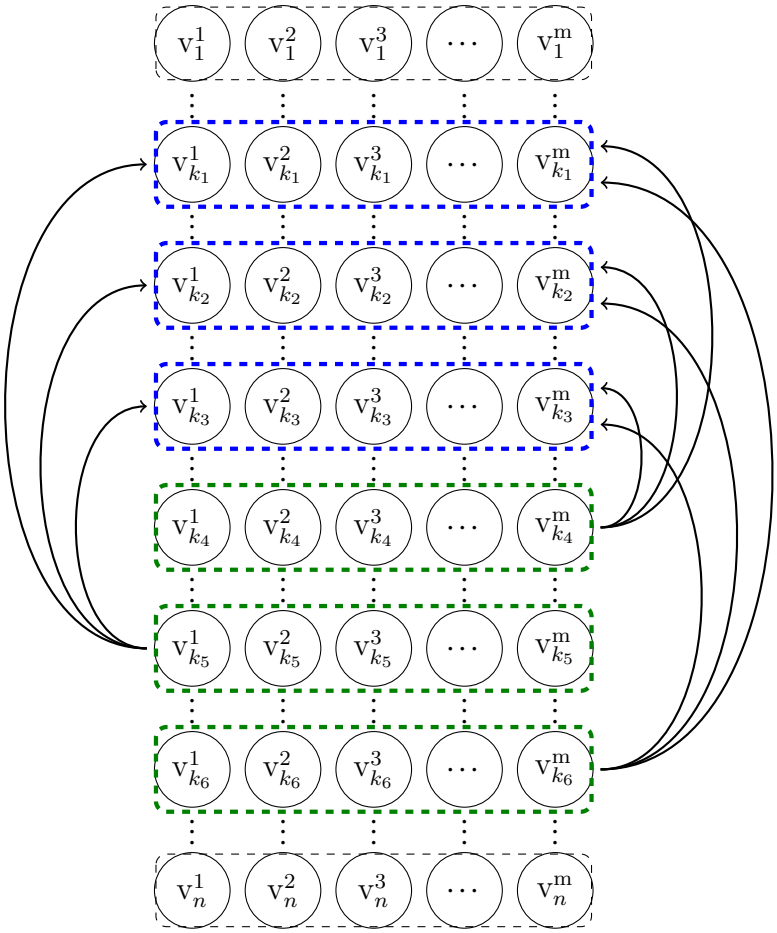
Analogicznie zaczynamy od wyboru grafu G_1^{mn}, podzielenia jego wierzchołków na m, n-elementowych kolejnych grup i zdefiniowaniu klasy $\mathcal{H}$ reprezentującej skierowane grafy pełne dwudzielne o 6-ciu wierzchołkach i następujących własnościach:

- każdy wierzchołek należy do innej grupy,
- krawędzie zorientowane są tak, aby prowadziły od wierzchołków o większym indeksie do tych o indeksie mniejszym,
- krawędzie grafu były zawarte w zbiorze krawędzi grafu G_1^{mn},
- stopnie wyjściowe nie są większe niż 1.

Rysunek B.1 przedstawia rozdział wierzchołków G_1^{mn} na grupy, wybór grup oraz jedne z możliwych 10-ciu poprowadzeń krawędzi między nimi (scenariusz 1) - krawędzie łączą pewne wierzchołki znajdujące się wewnątrz grup, na które wskazują strzałki. Partycje oznaczono kolorami niebieskim i zielonym.

Wybieramy teraz jeden element z klasy $\mathcal{H}$ i wyznaczamy liczbę rozdziałów wierzchołków wewnątrz grup i odpowiadające im stopnie wejściowe tych wierzchołków. Rozdziały te zależą także od tego, które grupy chcemy ze sobą połączyć. Grupy oznaczmy indeksami $1 \leq k_1 < k_2 < k_3 < k_4 < k_5 < k_6 \leq n$.

1. Wybór partycji: $A_1 = \{k_1, k_2, k_3\}$ oraz $B_1 = \{k_4, k_5, k_6\}$
   Rozkład krawędzi wyjściowych wierzchołków wewnątrz grup z k_4, k_5, k_6 wybieramy na $m(m-1)(m-2)$ sposobów. Rozkład krawędzi wejściowych możemy wybrać dla grup k_3, k_2, k_1 na
   - $m(m-1)(m-2)$ sposobów - 3 stopnie wejściowe równe 1, reszta 0
   - $3m(m-1)$ sposobów - 1 stopień wejściowy równy 2, 1 stopień równy 1, reszta 0
   - m sposobów - jeden stopień wejściowy równy 3, reszta 0
2. Wybór partycji: $A_2 = \{k_1, k_2, k_4\}$ oraz $B_2 = \{k_3, k_5, k_6\}$
   Rozkład krawędzi wyjściowych wierzchołków wewnątrz grup k_5 i k_6 wybieramy na $m(m-1)(m-2)$ sposobów, w grupie k_3 na $m(m-1)$ sposobów, w grupie k_4 na m sposobów.



Rys. B.1: Wizualizacja podziału wierzchołków grafu G_1^{mn} na grupy oraz jednego z możliwych wyborów krawędzi odpowiadających podgrafowi $K_{3,3}$ w G_m^n.

Rozkład krawędzi wejściowych dla grup k_2, k_1 możemy wybrać na

- $m(m-1)(m-2)$ sposobów - 3 stopnie wejściowe równe 1, reszta 0
- $3m(m-1)$ sposobów - 1 stopień wejściowy równy 2, 1 stopień równy 1, reszta 0
- m sposobów - jeden stopień wejściowy równy 3, reszta 0

Rozkład krawędzi wejściowych dla grupy k_4 możemy wybrać na

- $m(m-1)$ sposobów - 2 stopnie wejściowe równe 1, reszta 0
- m sposobów - jeden stopień wejściowy równy 2, reszta 0

Rozkład krawędzi wejściowych dla grupy k_3 możemy wybrać na m sposobów - jeden wierzchołek o stopniu 1.

3. Wybór partycji: $A_3 = \{k_1, k_2, k_5\}$ oraz $B_3 = \{k_3, k_4, k_6\}$

   Rozkład krawędzi wyjściowych wierzchołków wewnątrz grupy k_6 wybieramy na $m(m-1)(m-2)$ sposobów, wewnątrz grup k_5, k_4, k_3 na $m(m-1)$ sposobów.

   Rozkład krawędzi wejściowych dla grup k_2, k_1 możemy wybrać na

   - $m(m-1)(m-2)$ sposobów - 3 stopnie wejściowe równe 1, reszta 0
   - $3m(m-1)$ sposobów - 1 stopień wejściowy równy 2, 1 stopień równy 1, reszta 0
   - m sposobów - jeden stopień wejściowy równy 3, reszta 0

Rozkład krawędzi wejściowych w każdej z grup k_5, k_4 , k_3 możemy wybrać na m sposobów - po jednym wierzchołku o stopniu 1.

4. Wybór partycji: $A_4 = \{k_1, k_2, k_6\}$ oraz $B_4 = \{k_3, k_4, k_5\}$
   Rozkład krawędzi wyjściowych wierzchołków wewnątrz grupy k_6 wybieramy na $m(m-1)(m-2)$ sposobów, wewnątrz grup k_5, k_4, k_3 na $m(m-1)$ sposobów.
   Rozkład krawędzi wejściowych dla grup k_2, k_1 możemy wybrać na
   - $m(m-1)(m-2)$ sposobów - 3 stopnie wejściowe równe 1, reszta 0
   - $3m(m-1)$ sposobów - 1 stopień wejściowy równy 2, 1 stopień równy 1, reszta 0
   - m sposobów - jeden stopień wejściowy równy 3, reszta 0

   Rozkład krawędzi wejściowych w każdej z grup k_5, k_4 , k_3 możemy wybrać na m sposobów - po jednym wierzchołku o stopniu 1.
5. Wybór partycji: $A_5 = \{k_1, k_3, k_4\}$ oraz $B_5 = \{k_2, k_5, k_6\}$
   Rozkład krawędzi wyjściowych wierzchołków wewnątrz grup k_6, k_5 wybieramy na $m(m-1)(m-2)$ sposoby, wewnątrz grup k_4, k_3, k_2 na m sposobów.
   Rozkład krawędzi wejściowych dla grup k_3, k_1 możemy wybrać na
   - $m(m-1)(m-2)$ sposobów - 3 stopnie wejściowe równe 1, reszta 0
   - $3m(m-1)$ sposobów - 1 stopień wejściowy równy 2, 1 stopień równy 1, reszta 0
   - m sposobów - jeden stopień wejściowy równy 3, reszta 0

   Rozkład krawędzi wejściowych dla grup k_4, k_3, k_2 możemy wybrać na
   - $m(m-1)$ sposobów - 2 stopnie wejściowe równe 1, reszta 0
   - m sposobów - jeden stopień wejściowy równy 2, reszta 0
6. Wybór partycji: $A_6 = \{k_1, k_3, k_5\}$ oraz $B_6 = \{k_2, k_4, k_6\}$
   Rozkład krawędzi wyjściowych wierzchołków wewnątrz grupy k_6 wybieramy na $m(m-1)(m-2)$ sposobów, wewnątrz grup k_5, k_4 na $m(m-1)$ sposobów, wewnątrz grup k_3, k_2 na m sposobów.
   Rozkład krawędzi wejściowych dla grup k_3, k_1 możemy wybrać na
   - $m(m-1)(m-2)$ sposobów - 3 stopnie wejściowe równe 1, reszta 0
   - $3m(m-1)$ sposobów - 1 stopień wejściowy równy 2, 1 stopień równy 1, reszta 0
   - m sposobów - jeden stopień wejściowy równy 3, reszta 0

   Rozkład krawędzi wejściowych dla grup k_3, k_2 możemy wybrać na
   - $m(m-1)$ sposobów - 2 stopnie wejściowe równe 1, reszta 0
   - m sposobów - jeden stopień wejściowy równy 2, reszta 0

   Rozkład krawędzi wejściowych w każdej z grup k_5, k_4 możemy wybrać na m sposobów - po jednym wierzchołku o stopniu 1.
7. Wybór partycji: $A_7 = \{k_1, k_3, k_6\}$ oraz $B_7 = \{k_2, k_4, k_5\}$
   Rozkład krawędzi wyjściowych wierzchołków wewnątrz grupy k_6 wybieramy na $m(m-1)(m-2)$ sposobów, wewnątrz grup k_5, k_4 na $m(m-1)$ sposobów, wewnątrz grup k_3, k_2 na m sposobów.
   Rozkład krawędzi wejściowych dla grupy k_1 możemy wybrać na

- $m(m-1)(m-2)$ sposobów - 3 stopnie wejściowe równe 1, reszta 0
- $3m(m-1)$ sposobów - 1 stopień wejściowy równy 2, 1 stopień równy 1, reszta 0
- m sposobów - jeden stopień wejściowy równy 3, reszta 0

Rozkład krawędzi wejściowych dla grup k_3, k_2 możemy wybrać na

- $m(m-1)$ sposobów - 2 stopnie wejściowe równe 1, reszta 0
- m sposobów - jeden stopień wejściowy równy 2, reszta 0

Rozkład krawędzi wejściowych w każdej z grup k_5, k_4 możemy wybrać na m sposobów - po jednym wierzchołku o stopniu 1.

8. Wybór partycji: $A_8 = \{k_1, k_4, k_5\}$ oraz $B_8 = \{k_2, k_3, k_6\}$
   Rozkład krawędzi wyjściowych wierzchołków wewnątrz grupy k_6 wybieramy na $m(m-1)(m-2)$ sposobów, wewnątrz grup k_5, k_4 na $m(m-1)$ sposobów, wewnątrz grup k_3, k_2 na m sposobów.
   Rozkład krawędzi wejściowych dla grupy k_1 możemy wybrać na
   - $m(m-1)(m-2)$ sposobów - 3 stopnie wejściowe równe 1, reszta 0
   - $3m(m-1)$ sposobów - 1 stopień wejściowy równy 2, 1 stopień równy 1, reszta 0
   - m sposobów - jeden stopień wejściowy równy 3, reszta 0

   Rozkład krawędzi wejściowych dla grup k_3, k_2 możemy wybrać na
   - $m(m-1)$ sposobów - 2 stopnie wejściowe równe 1, reszta 0
   - m sposobów - jeden stopień wejściowy równy 2, reszta 0

   Rozkład krawędzi wejściowych w każdej z grup k_5, k_4 możemy wybrać na m sposobów - po jednym wierzchołku o stopniu 1.
9. Wybór partycji: $A_9 = \{k_1, k_4, k_6\}$ oraz $B_9 = \{k_2, k_3, k_5\}$
   Rozkład krawędzi wyjściowych wierzchołków wewnątrz grupy k_6 wybieramy na $m(m-1)(m-2)$ sposobów, wewnątrz grup k_5, k_4 na $m(m-1)$ sposobów, wewnątrz grup k_3, k_2 na m sposobów.
   Rozkład krawędzi wejściowych dla grupy k_1 możemy wybrać na
   - $m(m-1)(m-2)$ sposobów - 3 stopnie wejściowe równe 1, reszta 0
   - $3m(m-1)$ sposobów - 1 stopień wejściowy równy 2, 1 stopień równy 1, reszta 0
   - m sposobów - jeden stopień wejściowy równy 3, reszta 0

   Rozkład krawędzi wejściowych dla grup k_3, k_2 możemy wybrać na
   - $m(m-1)$ sposobów - 2 stopnie wejściowe równe 1, reszta 0
   - m sposobów - jeden stopień wejściowy równy 2, reszta 0

   Rozkład krawędzi wejściowych w każdej z grup k_5, k_4 możemy wybrać na m sposobów - po jednym wierzchołku o stopniu 1.
10. Wybór partycji: $A_{10} = \{k_1, k_5, k_6\}$ oraz $B_{10} = \{k_2, k_3, k_4\}$
    Rozkład krawędzi wyjściowych wierzchołków wewnątrz grup k_6, k_5 wybieramy na $m(m-1)(m-2)$ sposoby, wewnątrz grup k_4, k_3, k_2 na m sposobów. Rozkład krawędzi wejściowych dla grupy k_1 możemy wybrać na
    - $m(m-1)(m-2)$ sposobów - 3 stopnie wejściowe równe 1, reszta 0

- $3m(m-1)$ sposobów - 1 stopień wejściowy równy 2, 1 stopień równy 1, reszta 0
- m sposobów - jeden stopień wejściowy równy 3, reszta 0

Rozkład krawędzi wejściowych dla grup k_4, k_3, k_2 możemy wybrać na

- $m(m-1)$ sposobów - 2 stopnie wejściowe równe 1, reszta 0
- m sposobów - jeden stopień wejściowy równy 2, reszta 0

Niech $H = (V_H, E_H)$ oznacza graf z klasy $\mathcal{H}$. Ustalmy indeksy grup $1 \leq k_1 < k_2 < k_3 < k_4 < k_5 < k_6 \leq n$ i wybierzmy jeden z opisanych wyżej scenariuszy. Wówczas, analogicznie jak w trzecim rozdziale, mamy:

$$\prod_{(u,v)\in E_H} 2\sqrt{uv}\exp\left(O\left(\sum_{v\in V_H} C_H(v)^2/v\right)\right) \sim 2^9 m^9 k_1^{\frac{3}{2}} k_2^{\frac{3}{2}} k_3^{\frac{3}{2}} k_4^{\frac{3}{2}} k_5^{\frac{3}{2}} k_6^{\frac{3}{2}} \tag{B.1}$$

Końcowy krok polega na zsumowaniu wszystkich prawdopodobieństw. Tutaj także główne obliczenia zostały wykonane przy użyciu skryptu, który znajduje się w Dodatku A. Różnicą jest fakt, że mamy 10 różnych scenariuszy i musimy zsumować wyniki w nich uzyskane.

$$\begin{aligned}
\mathbb{E}[K_{3,3}] &\sim \sum_{\mathcal{K}^{\mathrm{ord}}_{n,6}} \frac{(m^2-2)^2(m^2-1)^2(5m^4-5m^2+4)}{512m^3(k_1k_2k_3k_4k_5k_6)^{3/2}} \sim \\
&\sim \left(\frac{1}{6!}\right)^{\frac{3}{2}} \sum_{\mathcal{K}_{n,6}} \frac{(m^2-2)^2(m^2-1)^2(5m^4-5m^2+4)}{512m^3(k_1k_2k_3k_4k_5k_6)^{3/2}} = \\
&= \frac{(m^2-2)^2(m^2-1)^2(5m^4-5m^2+4)}{2211840\sqrt{5}m^3}(H_n^{(3/2)})^6 \sim \\
&\sim \frac{(m^2-2)^2(m^2-1)^2(5m^4-5m^2+4)}{2211840\sqrt{5}m^3}\left(\zeta(3/2)\right)^6
\end{aligned} \tag{B.2}$$