Teoria Algorytmów i Obliczeń

Algorytm dokładny oraz dwa algorytmy aproksymacyjne znajdowania maksymalnego spójnego izomorficznego podgrafu dwóch grafów

Dokumentacja i Raport

Kazimierz Wojciechowski

12 listopada 2018

Spis treści

Rozdział 1

Instrukcja uruchomienia

1.1 Wymagania systemowe

- System Operacyjny Windows, macOS oraz Linux na których można zainstalować platformę .NET Core 2.0.
- Platforma .NET Core 2.0
- Jeżeli użytkownik ma system operacyjny Windows ale nie ma platformy .NET Core to może uruchomić niezależną wersję programu (`Application.exe` zamiast `Application.dll`)

1.2 Uruchamianie z parametrami z poziomu powłoki systemowej

Program można uruchomić na dwa sposoby w zależności czy jest dostępna platforma .NET Core. Jeżeli adres URI grafu zawiera znaki białe to należy obłożyć ten parametr w cudzysłów.

- `dotnet Application.dll parametr1 "parametr2" parametr3 ...`
- `Application.exe "parametr1" parametr2 parametr3 ...`

Po uruchomieniu programu bez parametrów lub po ukończonych obliczeniach można wprowadzić parametry uruchomieniowe. Ponieważ adresy URI często zawierają białe znaki, należy oddzielić kolejne parametry średnikiem.

Kolejność i rodzaj parametrów są wyjaśniane po uruchomieniu programu bez parametrów jako pomoc w pierwszym użyciu parametrów oraz jako przypomnienie ich prawidłowego używania.

Rozdział 2

Wstęp

2.1 Definicja problemu

Problem polega na znalezieniu maksymalnego izomorficznego spójnego podgrafu[1] dwóch niepustych niekoniecznie spójnych grafów bez pętli $G = (V_G, E_G)$ i $H = (V_H, E_H)$.

Zgłębiając się w szczegóły, problem sprowadza się do znalezienia jednej (z możliwie wielu) maksymalnej względem pewnego monotonicznego[2] kryterium $\kappa : (V, E) \to \mathbb{R}$ (w implementacji algorytmów $\kappa : \mathbb{N}^2 \to \mathbb{R}$, a parametrami są liczba wierzchołków oraz liczba krawędzi w izomorficznym podgrafie) różnowartościowej funkcji $f : V_G \to V_H$, której dziedziną jest podzbiór wierzchołków grafu G a przeciwdziedziną jest podzbiór wierzchołków grafu H. Ponadto, funkcja powinna spełniać warunek lokalnej izomorficzności:

$$\forall\, u, v \in D_f \quad \{u, g\} \in E_G \iff \{f(u), f(g)\} \in E_H$$

Funkcję spełniającą powyższe kryteria będziemy nazywać mapą.

[1] Algorytm dokładny praktycznie bez spowolnienia liczy także niespójne podgrafy. Analiza złożoności w następnym rozdziale dotyczy wariantu algorytmu znajdującego tylko spójny podgraf. Pierwszy algorytm aproksymacyjny został zaprojektowany z myślą wyłącznie o znajdowaniu spójnego podgrafu izomorficznego. Natomiast drugi algorytm aproksymacyjny ma takie same możliwości co algorytm dokładny.

[2] Wartościowanie jest niemalejące względem zarówno liczby wierzchołków jak i krawędzi w izomorficznym podgrafie. Dzięki takiemu ograniczeniu funkcji wartościującej algorytm dokładny oraz drugi aproksymacyjny wykonują mniej obliczeń.

Rozdział 3

Analiza problemu

3.1 Spostrzeżenia

Istnieje niepusta mapa

Skoro grafy G oraz H są niepuste to zawsze istnieje przynajmniej jedna mapa – wystarczy założyć izomorficzność jednego, dowolnego wierzchołka ze zbioru V_G oraz jeden, także dowolnego wierzchołka z V_H. W ten sposób powstanie jak najbardziej poprawny izomorficzny podgraf.

Istnieje przynajmniej jedna maksymalna mapa

Istnieje co najmniej jedna maksymalna mapa według kryterium wartościującego κ. Prawidłowych map dla danych dwóch grafów jest skończenie wiele, stąd możemy w skończonym czasie znaleźć wszystkie takie prawidłowe mapy, następnie dokonać wartościowania wszystkich poprawnych map zgodnie z ustalonym kryterium aby następnie wybrać jedną mapę z niepustego zbioru maksymalnych map.

Definicja

Niech $M^{\kappa}_{G,H}$ będzie zbiorem maksymalnych map zgodnie z kryterium κ między grafem G a grafem H.

Własność redukcji dla maksymalnej mapy

Mapa $m^{\kappa}_{G,H} \in M^{\kappa}_{G,H}$ nadal pozostaje maksymalna po usunięciu dowolnej liczby wierzchołków z G nienależących do dziedziny tej mapy. Analogicznie jest z przeciwdziedziną mapy oraz grafem H. Bardziej formalnie:

$$\left(m^{\kappa}_{G,H} \in M^{\kappa}_{G,H} \ \wedge \ D_{m^{\kappa}_{G,H}} \subseteq G' \subseteq G \ \wedge \ \complement_{m^{\kappa}_{G,H}} \subseteq H' \subseteq H\right) \Longrightarrow m^{\kappa}_{G,H} \in M^{\kappa}_{G',H'}$$

W szczególności jeżeli w dziedzinie maksymalnej mapy dla G oraz H nie znajdziemy pewnego wierzchołka (nazwijmy go v_G^k) to prawidłową mapę dla G oraz H możemy równie dobrze znaleźć badając grafy $G^* := \left(V_G - \{v_G^k\},\ E_G \cap (V_G - \{v_G^k\})^2\right)$ oraz H.

Zakładając, że wierzchołek nie należy do dziedziny maksymalnej mapy $m_{G,H}^\kappa$ możemy badać graf G bez tego wierzchołka. Jeśli jednak należy do mapy wtedy wartość funkcji $m(v_G^k)$ musi być określona, ten wniosek doprowadzi nas do konstrukcji algorytmu dokładnego.

3.1.1 Szkic algorytmu dokładnego

Niech m będzie jednym z elementów zbioru $M_{G,H}^\kappa$. Weźmy dowolny wierzchołek $v_G^0 \in V_G$. Wierzchołek v_G^0 albo należy do dziedziny m albo do niej nie należy. Jeśli należy to wtedy $m(v_G^0) \in V_H$, a jeśli nie należy to $m \in M_{G,H}^\kappa \Longrightarrow m \in M_{G-\{v_G\},H}^\kappa$.

Najpierw rozpatrzmy rekurencyjnie problem podmieniając G na $G^* = G - \{v_G\}$ (możemy wtedy równie dobrze odnaleźć m dla problemu o mniejszym rozmiarze).

Następnie rozpatrzymy rekurencyjnie wszystkie możliwe wartościowania mapy $m(v_G) = v_H \in V_H$.

Skoro dziedzina m jest spójna w sensie podgrafu G^* to warto rozpatrywać kolejne wartościowania mapy według takiego porządku aby każdy kolejny rozpatrywany wierzchołek $v_{G^*}^{\text{candidate}}$ oraz jego izomorficzny odpowiednik $v_H^{\text{candidate}}$ był połączony z już dotychczas wybudowaną dziedziną (czyli należy do zbioru tzw. Otoczki).

W ten sposób upewnimy się, że dziedzina (odpowiednio przeciwdziedzina) będzie w każdym kroku spójnym podgrafem indukowanym grafu G^* (odpowiednio H) czyli tym bardziej spójnym podgrafem indukowanym G. Rozpatrując poprawność izomorfizmu wierzchołków v_G oraz v_H sprawdzamy czy są identycznie połączone z już zbudowaną dziedziną D_f^k oraz odpowiadającą przeciwdziedziną w k-tym kroku (głębokość rekursji):

$$\forall d \in D_f^k \quad \{v_G, d\} \in E_{G^*} \subseteq E_G \iff \{v_H, m(d)\} \in E_H$$

Dzięki powyższemu warunkowi mamy pewność, że konstruowana mapa w każdym kroku iteracji spełnia definicję prawidłowej mapy.

3.1.2 Dokładny algorytm w języku naturalnym

Uwagi

Wiele implementacyjnych szczegółów zostanie pominiętych ze względu na czytelność.

Niezmienniki rekursji

1. W trakcie rekursji będziemy usuwać i przywracać usunięte wierzchołki z grafu G stąd poniższe odwołania do G^* dotyczą także podgrafu grafu G oprócz pełnego G.

2. W każdym wywołaniu rekursji wspólny izomorficzny podgraf S_{G^*} oraz S_H z mapą $m : S_{G^*} \to S_H$ posiadają niepuste zbiory wierzchołków.

3. Zbiór Otoczki (dla grafu G^* i podgrafu S_{G^*}) $O_{S_{G^*},G^*}$ nazywamy wszystkie wierzchołki z $V_{G^*} - V_{S_{G^*}}$ które są połączone co najmniej jedną krawędzią z jakimkolwiek wierzchołkiem z $V_{S_{G^*}}$. Analogiczne definiujemy zbiór Otoczki dla grafu H i podgrafu S_H.

4. Zbiór Pozostałych wierzchołków (dla grafu G^* i podgrafu S_{G^*}) nazywamy zbiór wszystkich wierzchołków ze zbioru $V_{G^*} - V_{S_{G^*}} - O_{S_{G^*},G^*}$. Są to wierzchołki nie należące do wspólnego podgrafu ani do Otoczki tzn. nie istnieje krawędź bezpośrednio łącząca te wierzchołki z rozpatrywanym wspólnym izomorficznym podgrafem S_{G^*}. Analogiczne definiujemy zbiór Pozostałych wierzchołków dla grafu H i podgrafu S_H.

5. Poprzez zagłębienie się w rekursji rozumiemy analizę wszystkich sensownych mapowań oraz zapamiętanie najbardziej wartościowej ograniczając się przez już dotychczas ustalone ograniczenia (np. kluczowe ograniczenie dot. lokalnej izomorficzności już zbudowanego podgrafu S_{G^*} i S_H).

Rekursja

1. Zakładam, początkowo, że pewne dwa wierzchołki (jeden z V_{G^*} oraz jeden z V_H) są izomorficzne. Następnie zagłębiam się w rekursji spełniając wymagania niezmienników odpowiednio manipulując zbiorami Otoczek oraz Pozostałych dla obu grafów: G^* oraz H. Jeśli napotkam w czasie rekursji lepszą niż dotychczas zapamiętaną mapę, to zapisuję ją do globalnej dla algorytmu zmiennej, którą na koniec rekursji zwracam jako wynik.

2. Rekursja polega na:

   (a) Sprawdzeniu czy którakolwiek z Otoczek jest pusta. Jeśli tak, to sprawdzam maksymalność znalezionej dotychczas mapy oraz jeśli jest wyżej wartościowana zapamiętuję najlepszą znalezioną mapę. Następnie wychodzę z rekursji.

   (b) Jeśli liczba wierzchołków w obu grafach G^* i H oraz liczba krawędzi dają nadzieję na znalezienie lepszego wyniku to kontynuujemy rozważania w ramach tego poziomu rekursji czyli przechodzimy do kolejnego kroku.

(c) Wybieramy dowolnego kandydata z Otoczki O_{S_G,G^*}, który ma najmniejszy stopień[1] spośród wszystkich wierzchołków ze zbioru otoczki. Następnie poprawiamy zbiory Otoczek oraz Pozostałych względem grafu G^*.

(d) Dla każdego poprawnego kandydata z otoczki $O_{S_H,H}$ to znaczy lokalnie izomorficznego[2] zagłębiamy się w rekursji zakładając izomorficzność kandydata z H oraz z G^* z poprzedniego punktu. Dodajemy oba wierzchołki do im odpowiadających podgrafów S_{G^*} oraz S_H. Powracając z rekursji oraz ponownie się w niej zagłębiając dbamy o spełnienie niezmiennika dotyczącego Otoczek oraz Pozostałych.

(e) Powróciwszy z ostatniej prawidłowej rekursji opisanej powyżej ponownie zagłębiam się w rekursji tym razem zakładając że nasz najlepszy kandydat z punktu 2.(c) nie należy do maksymalnej mapy[3]. Oznacza to, że usuwamy go z grafu G^* i zagłębiamy się w rekursji bez tego wierzchołka. Po powrocie z rekursji z powrotem dodajemy ten wierzchołek do G^*.

(f) Cofamy wszystkie poprawki dokonane w punkcie 2.(c) tak, aby przywrócić stan wszystkich Otoczek i Pozostałych do stanu pierwotnego to znaczy jakiego zastaliśmy wchodząc na ten poziom rekursji.

3. Wykluczamy z rozważania wierzchołek wybrany w punkcie 1 to znaczy z grafu G^* usuwamy wybrany wierzchołek wraz z powiązanymi z nim krawędziami i następnie powtarzamy krok 1 i zagłębiamy się w kroku 2 dla mniejszego grafu G^* oraz nienaruszonego grafu H. Jeśli mniejszy graf G^* już nie ma szans na poprawę wyniku (jest pusty lub zawiera mniej wierzchołków i krawędzi nie pozostawiając nadziei na poprawę wyniku) wtedy z monotoniczności kryterium κ możemy zaprzestać zbędnych obliczeń które już nie poprawią wyniku.

[1] Dla grafów o kilkunastu wierzchołkach i gęstości 90%-99% oraz wartościowania $\kappa(G) = |V_G|$ dzięki właśnie takiemu kryterium wyboru wierzchołka z Otoczki O_{G^*} czas obliczeń trwa najczęściej dziesiątki/setki milisekund zamiast minut. Dla wartościowania zawierającego liczbę krawędzi jak np. $\kappa(G) = |V_G| + |E_G|$ czasami zdarzają się przypadki (np. $|V| = 18, \rho = 93\%$) dla których czas obliczeń trwa długie minuty. Drugi algorytm aproksymacyjny na pocieszenie najczęściej oblicza 100% odpowiedzi na tego typu podchwytliwe grafy. Oznacza to, że poprawny wynik jest znajdowany wcześnie, tylko dowodzenie jego maksymalności trwa tak długo.

[2] $(s^{\text{nowy}}, s_i) \in E_{S_{G^*}} \iff (h^{\text{nowy}}, m(s_i)) \in E_{S_H}$ czyli istnieje połączenie z wierzchołkami z podgrafu S_H wtedy i tylko wtedy gdy istnieje połączenie z im izomorficznymi odpowiednikami z S_{G^*} inaczej trywialnie nie mogą to być wierzchołki izomorficzne nawet po dodaniu dowolnej liczby wierzchołków do obu podgrafów tzn. zwiększając wymagania wgłąb rekursji. Będzie to zawsze nieprawidłowa mapa między nieizomorficznymi podgrafami.

[3] Gdyby należał, to by był izomorficzny z pewnym wierzchołkiem z Otoczki O_H, a ten przypadek już rozważyliśmy w punkcie 2.(d)

4. Po zakończonych obliczeniach zwracam najlepszy dotychczas znaleziony wynik oraz mapę której wartość jest maksymalna.

3.1.3 Pierwszy algorytm aproksymacyjny w języku naturalnym

Opis zmagań

Zaimplementowałem wersję algorytmu aproksymacyjnego wykonującego zero, jeden lub dowolną liczbę kroków naprzód algorytmem dokładnym po czym został wybierany wierzchołek, który dawał najlepszy lokalnie wynik.

Wykonanie choć jednego kroku algorytmem dokładnym nie dawało praktycznej przewagi nad algorytmem wykonującym zero kroków[4].

Jeśli było kilka wierzchołków o takim samym wyniku to był on wybierany na podstawie liczności sąsiadów w oryginalnym grafie oraz liczbie połączeń z już istniejącym izomorficznym podgrafem. Wartościowania te były bardzo zróżnicowane: były to wszystkie kombinacje do trzeciego poziomu złożenia mnożeń, sum, maksimów, minimów, potęg powyższych czterech liczb dla kandydatów. Poddałem więc w wątpliwość ograniczanie się do wybierania maksymalnego wierzchołka według jakiegokolwiek kryterium. Algorytm losujący bez zagłębiania się dawał lepsze jakościowo wyniki w tym samym czasie co wykonujący kilka kroków algorytm dokładny ze statystykami.

Algorytm aproksymacyjny

1. Poniższa iteracja zostaje powtórzona (z różnym ziarnem dla generatora liczb losowych) w liczbie proporcjonalnej do maksimum z rozmiaru dwóch grafów $\max\{|V_G|, |V_H|\}$. Wybierane jest maksimum spośród znalezionych wyników.

   (a) Algorytm początkowo wybiera losowo parę wierzchołków i zakłada ich izomorficzność.

   (b) Spośród Otoczek zarówno O_G jak i O_H algorytm losuje kolejną izomorficzną parę pod warunkiem że są lokalnie izomorficzne. Jeśli wylosowana para nie jest lokalnie izomorficzna, to losowana jest kolejna para. Jeśli nie ma możliwości wyboru kolejnej izomorficznej pary to algorytm kończy działanie zwracając mapę wraz z jej wartościowaniem.

   (c) Jeżeli udało się znaleźć lokalnie izomorficzną parę to algorytm w kolejnym kroku zakłada izomorficzność znalezionej pary i aktualizuje zbiory Otoczek oraz Pozostałych oraz przechodzi do kroku 1.(b)

[4] Zero kroków sprawdza się do poprzestania na sprawdzeniu lokalnej izomorficzności dwóch kandydatów.

3.1.4 Drugi algorytm aproksymacyjny w języku naturalnym

Wstęp

Algorytm polega na ograniczeniu przeszukiwań drzewa rozwiązań przez pewną stałą zależną od początkowego grafu G oraz H, czyli krok 2 opisany w algorytmie dokładnym zostanie wykonany co najwyżej skończoną wcześniej ustaloną liczbę razy ale dopiero od pewnej ustalonej głębokości.

Różnice między algorytmem dokładnym a aproksymacyjnym

1. Z każdym uruchomieniem procedury Rekursji w kroku 2 algorytmu dokładnego zostaje zmniejszona określona liczba pod warunkiem, że przekroczyliśmy pewną liczbę oznaczającą głębokość od której należy liczyć kroki.

2. Jeżeli znajdujemy się na niskiej głębokości to znaczy nie przekraczającej głębokość podaną w parametrze wywołania to wtedy resetujemy licznik ograniczający liczbę wywołań rekursji.

3. Przed krokiem 2.(a) na samym początku sprawdzam czy ta liczba nie jest wyzerowana. Jeśli jest wyzerowana to przechodzimy natychmiast do kroku sprawdzania i ewentualnego zapamiętania znalezionej mapy dokładnie tak jak jest opisane w kroku 2.(a). Następnie powracamy z rekursji tak długo aż licznik nie zostanie zresetowany lub kiedy już zupełnie wyjdziemy z rekursji.

Rozdział 4

Analiza złożoności

4.1 Algorytm dokładny

W implementacji posługuję się słownikami i zbiorami [1] i zakładam, że operacja na takich strukturach danych trwa co najwyżej $\mathcal{O}(1)$[2]. Tam, gdzie dane się nie zmieniają staram się używać tablic wielowymiarowych.

Oznaczenia

Funkcję całkowitego kosztu nazywam funkcję $\mathscr{C}$. Notacja $\mathcal{O}$ "dużego O" oznacza złożoność co najwyżej rzędu funkcji określonej w nawiasie (oszacowanie górne).

Funkcja $R : \mathbb{N}^3 \to \mathbb{N}$ szacuje odgórnie złożoność czasową kluczowej procedury algorytmu. Poniżej zostają objaśnione kolejne parametry funkcji szacującej R:

- parametr g oznacza liczbę wierzchołków w lokalnym grafie G^*.
- parametr h nie ulega zmianie w czasie działania algorytmu i jest równy liczbie wierzchołków grafu H.
- parametr s oznacza liczbę wierzchołków już dotąd zbudowanego podgrafu izomorficznego: $|V_{S_{G^*}}|$.

[1] Oba zaimplementowane jako tablica z haszowaniem.

[2] Nawet gdyby operacja trwała $\mathcal{O}(\log |V_{G^*}|)$ czy $\mathcal{O}(\log |V_H|)$ to i tak złożoność wzrośnie o kwadrat logarytmu czyli o coś praktycznie niezauważalnego.

$$
\begin{aligned}
R(g,h,s) &= \mathcal{O}(g-s) + (h-s)(\mathcal{O}(s) + \mathcal{O}(h-s) + R(g,h,s+1)) + R(g-1,h,s) \\
R(g,h,s) &= \mathcal{O}(h^2+g) = c \cdot h^2, \quad \text{dla } g = s+1 \\
R(g,h,s) &= \mathcal{O}(1), \quad \text{dla } g = s \\
\mathscr{C}(G,H) &= \mathcal{O}\left(\sum_{k=1}^{|V_G|} |V_H| \cdot (k + |V_H| + |E_H| + |E_G| + R(k,|V_H|,0))\right) \\
&= \mathcal{O}\left(\sum_{k=1}^{|V_G|} |V_H| \cdot R(k,|V_H|,1)\right) \\
&= \mathcal{O}\left(|V_H| \cdot \sum_{k=1}^{|V_G|} R(k,|V_H|,1)\right)
\end{aligned}
$$

Powyższe oszacowania są wyznaczone zgodnie z przetestowaną implementacją algorytmu w celu ogarnięcia wszystkich niuansów wynikających z potrzeby poprawnego działania algorytmu.

Niech m oznacza rozmiar maksymalnego wspólnego podgrafu względem monotonicznego wartościowania κ.

W szczególnym przypadku rozpisanym poniżej rekursja zagłębi się co najwyżej do poziomu $s = m = 5$.

g\s	1	2	3	4	5	6	7	8
1	$\mathcal{O}(1)$							
2	$c \cdot \lvert V_H\rvert^2$	$\mathcal{O}(1)$						
3	$c \cdot \lvert V_H\rvert^3$	$c \cdot \lvert V_H\rvert^2$	$\mathcal{O}(1)$					
4	$c \cdot \lvert V_H\rvert^4$	$c \cdot \lvert V_H\rvert^3$	$c \cdot \lvert V_H\rvert^2$	$\mathcal{O}(1)$				
5	$c \cdot \lvert V_H\rvert^5$	$c \cdot \lvert V_H\rvert^4$	$c \cdot \lvert V_H\rvert^3$	$c \cdot \lvert V_H\rvert^2$	$\mathcal{O}(1)$			
6	$c \cdot \lvert V_H\rvert^6$	$c \cdot \lvert V_H\rvert^5$	$c \cdot \lvert V_H\rvert^4$	$c \cdot \lvert V_H\rvert^3$	$c \cdot \lvert V_H\rvert^2$	0 [3]	0	0
7	$6c \cdot \lvert V_H\rvert^6$	$5c \cdot \lvert V_H\rvert^5$	$4c \cdot \lvert V_H\rvert^4$	$3c \cdot \lvert V_H\rvert^3$	$2c \cdot \lvert V_H\rvert^2$	0	0	0
8	$21c \cdot \lvert V_H\rvert^6$	$15c \cdot \lvert V_H\rvert^5$	$10c \cdot \lvert V_H\rvert^4$	$6c \cdot \lvert V_H\rvert^3$	$3c \cdot \lvert V_H\rvert^2$	0	0	0
9	$56c \cdot \lvert V_H\rvert^6$	$35c \cdot \lvert V_H\rvert^5$	$20c \cdot \lvert V_H\rvert^4$	$10c \cdot \lvert V_H\rvert^3$	$4c \cdot \lvert V_H\rvert^2$	0	0	0
10	$126c \cdot \lvert V_H\rvert^6$	$70c \cdot \lvert V_H\rvert^5$	$35c \cdot \lvert V_H\rvert^4$	$15c \cdot \lvert V_H\rvert^3$	$5c \cdot \lvert V_H\rvert^2$	0	0	0
11	$252c \cdot \lvert V_H\rvert^6$	$126c \cdot \lvert V_H\rvert^5$	$56c \cdot \lvert V_H\rvert^4$	$21c \cdot \lvert V_H\rvert^3$	$6c \cdot \lvert V_H\rvert^2$	0	0	0
12	$462c \cdot \lvert V_H\rvert^6$	$210c \cdot \lvert V_H\rvert^5$	$84c \cdot \lvert V_H\rvert^4$	$28c \cdot \lvert V_H\rvert^3$	$7c \cdot \lvert V_H\rvert^2$	0	0	0

[3] $R(g,h,s)$ dla $s > m$ nigdy się nie wykona więc jego koszt obliczenia jest zerowy.

$$\begin{aligned}
\mathscr{C}(G,H) &= \mathcal{O}\left(\sum_{k=2}^{m}|V_H|^k + \sum_{k=m+1}^{|V_G|}\binom{k-1}{m}|V_H|^{m+1}\right)\\
&= \mathcal{O}\left(\frac{|V_H|^{m+1}-|V_H|^2}{|V_H|-1} + |V_H|^{m+1}\sum_{k=m}^{|V_G|-1}\binom{k}{m}\right)\\
&= \mathcal{O}\left(|V_H|^{m+1}\left(1+(m+1)+\frac{(m+2)(m+1)}{2!}+\cdots+\frac{(|V_G|-1)\ldots(|V_G|-m)}{m!}\right)\right)\\
&= \mathcal{O}\left((|V_H|\cdot|V_G|)^{m+1}\right)
\end{aligned}$$

Podejrzewam, że w ogólności dla $g, s \in \mathbb{N} \quad g > m \geqslant s$ współczynnik jest równy

$$W(g,s) = \binom{g-s}{m-s+1}$$

Uzasadnienie:

$$W(g,s) = W(g{-}1,s) +^{4} W(g,s{+}1) = \binom{g-1-s}{m-s+1} + \binom{g-(s+1)}{m-(s+1)+1} = \binom{g-s}{m-s+1}$$

Z powyższej analizy wynika, że krytycznym dla wydajnego rozwiązania problemu okazuje się ograniczenie przeszukiwania drzewa rozwiązań do takich podgrafów, których rozmiar nie przewyższa maksymalnego.

Praktyka pokazuje, że najsprawniej obcinane są obliczenia kiedy usuwamy wierzchołki z mniejszego grafu (krawędziowo, nie wierzchołkowo) czyli kiedy zachodzi nierówność $|E_G| \leqslant |E_H|$.

Gdyby jeden z grafów np. G miał stałą, ustaloną liczbę wierzchołków to złożoność algorytmu dokładnego można by było ograniczyć z góry przez wielomian $\omega(h) = \mathcal{O}(h^{|V_G|+1})$ gdzie parametrem h jest liczba wierzchołków grafu H.

Dolne ograniczenie

W przypadku poszukiwania maksymalnego spójnego izomorficznego podgrafu oraz szczęśliwego dobierania wierzchołków-kandydatów dolne ograniczenie złożoności obliczeniowej (wynikające z implementacji algorytmu) wynosi:

$$\left(|V_G|^2 + |V_H|^2\right) m$$

Gdzie m to rozmiar maksymalnego[5] izomorficznego spójnego podgrafu obu grafów.

[4] Mnożąc przez $|V_H|$ otrzymujemy ten sam rząd wielkości stąd sumujemy współczynniki z sąsiednich komórek

[5] względem liczby wierzchołków

4.2 Pierwszy algorytm aproksymacyjny

Nawiązując do opisu w języku naturalnym w centralnej części algorytmu mamy pewną liczbę iteracji (maksymalnie m)[6] z której każda ma złożoność $\mathcal{O}(D^3)$, gdzie D jest większym z $|V_G|$ oraz $|V_H|$.

Powyższą, centralną część algorytmu powtarzamy maksymalnie m razy. Stąd całkowite górne oszacowanie złożoności algorytmu aproksymacyjnego wynosi $\mathcal{O}(m \cdot D^3)$. Jeśli w pesymistycznym przypadku założymy, że izomorficzny podgraf jest rzędu liczby wierzchołków grafów w problemie wtedy złożoność wynosi $\mathcal{O}(D^4)$.

Powyższą procedurę użytkownik powtarza tyle razy ile uważa za stosowne, może to być liczba stała a może to być liczba proporcjonalna do pewnej kombinacji liczby wierzchołków obu grafów. Wtedy złożoność całego algorytmu odpowiednio wzrośnie.

Dolnym ograniczeniem dla centralnej części algorytmu w przypadku szczęśliwego doboru lokalnie izomorficznych kandydatów wynosi $\mathcal{O}(D^2 \log D)$. Dla całego algorytmu należy wymienioną złożoność przemnożyć przez liczbę powtórzeń centralnej części – dokładnie tak, jak przemnożyliśmy górne ograniczenie przez liczbę powtórzeń losowania. Logarytm wynika z sortowania podczas każdej iteracji. Teoretycznie można się pozbyć logarytmu stosując algorytm Fisher–Yates'a, ale generator liczb losowych nie jest idealny, więc większą losowość i w konsekwencji lepsze wyniki jednak daje posortowanie kandydatów względem przypisanych im losowych wartości.

Poprawność pierwszego algorytmu aproksymacyjnego wynika z konstrukcji poprawnego izomorficznego, spójnego podgrafu w każdym kroku iteracji.

4.3 Drugi algorytm aproksymacyjny

Ponieważ liczba ograniczająca zmniejsza się z każdym wywołaniem procedury, która sama z siebie trwa co najwyżej $\mathcal{O}(|V_G|^2 + |V_H|^2)$. Kolejne wywołania tej procedury możemy jednoznacznie przyporządkować każdej liczbie naturalnej w kolejności od danej stałej aż do jedności. Jeśli chcielibyśmy skorzystać z modułu obliczającego składowe niespójne to wtedy złożoność pojedynczego kroku rozpoczynającego obliczenie kolejnej składowej niespójnej można ograniczyć z góry przez $\mathcal{O}\left(|V_G||V_H|^2\left(|V_G| + |V_H|\right)\right)$. Każdą iterację rozpoczynamy z daną stałą dla każdej początkowej izomorficznej pary wierzchołków.

Jedne z lepszych wyników dla różnych wartościowań biorąc pod uwagę czas obliczeń daje stała

$$C(G, H) = c \cdot (\min\{|V_G|, |V_H|\} + \min\{|E_G|, |E_H|\})$$

[6] m to rozmiar maksymalnego wspólnego podgrafu względem liczby wierzchołków

Gdzie stała c może być dowolna, w praktyce dobrym kompromisem między wydajnością a jakością aproksymacji jest wybranie stałej z przedziału $[1, 100]$. Stała równa jedności daje dosyć zadowalające wyniki, ale za to obliczenia przebiegają błyskawicznie. Złoty środek to stała rzędu $[10, 50]$. Stała równa 100 pozwala pokryć naprawdę większość ścieżek obliczeń prowadzących do maksymalnego podgrafu – szczególnie dla wymagającego wartościowania $\kappa(G) = |V_G| + |E_G|$.

Niech d oznacza głębokość od której (włącznie) ma być liczona liczba wywołań rekursji, może to być dowolna nieujemna liczba. W praktyce lepiej jest jednak zwiększać parametr c niż d.

Zatem jeśli mamy do czynienia z problemem znalezienia niekoniecznie spójnego podgrafu izomorficznego górne ograniczenie złożoności algorytmu wynosi:

$$\mathcal{O}\left(|V_G||V_H| \cdot (|V_G|^2 + |V_H|^2)^d \cdot |V_G||V_H|^2(|V_G| + |V_H|) \cdot (\min\{|V_G|, |V_H|\} + \min\{|E_G|, |E_H|\})\right) =$$
$$\mathcal{O}\left(|V_G|^2|V_H|^3(|V_G|^2 + |V_H|^2)^d(|V_G| + |V_H|)(\min\{|V_G|, |V_H|\} + \min\{|E_G|, |E_H|\})\right)$$

Jeśli jednak problem dotyczy wyłącznie spójnych podgrafów izomorficznych wtedy złożoność wynosi:

$$\mathcal{O}\left(|V_G||V_H| \cdot (|V_G|^2 + |V_H|^2)^d \cdot (|V_G| + |V_H|^2) \cdot (\min\{|V_G|, |V_H|\} + \min\{|E_G|, |E_H|\})\right)$$

Upraszczając zapisy zakładamy, że $D = \max\{|V_G|, |V_H|\}$ oraz, że $|E_G| = \mathcal{O}(D^2), |E_H| = \mathcal{O}(D^2)$. Poniżej zostają wymienione mniej dokładne ograniczenia z góry w zależności od ograniczenia co do spójności rozwiązania:

- rozwiązanie niespójne $\mathcal{O}\left(D^{8+2d}\right)$
- rozwiązanie spójne $\mathcal{O}\left(D^{6+2d}\right)$

Dolne ograniczenie złożoności procedury poszukującej izomorficzny podgraf spójny jest identyczne z ograniczeniem dla algorytmu dokładnego.

Poprawność drugiego algorytmu aproksymacyjnego wynika z wykonywania podzbioru podprocedur algorytmu dokładnego. Analogicznie do pierwszego algorytmu aproksymacyjnego – w każdym kroku iteracji tworzymy poprawny izomorficzny podgraf.

Rozdział 5

Testy poprawności

5.1 Pierwszy zestaw testów

Pierwszy zestaw testów składa się z problemu znalezienia maksymalnego spójnego izomorficznego podgrafu dwóch izomorficznych grafów o liczbie wierzchołków od 1 do 10.

- Sprawdzany jest rozmiar znalezionego podgrafu (czy jest równy maksymalnej składowej spójności)
- Sprawdzana jest izomorficzność, czyli równoważność istnienia lub nieistnienia krawędzi między izomorficznymi wierzchołkami
- Sprawdzana jest różnowartościowość mapy m

5.2 Drugi zestaw testów

Drugi zestaw, podobny do pierwszego, zostaje sprawdzany za pomocą algorytmu poszukiwania niekoniecznie spójnych podgrafów.

Rozmiar (osobno liczba wierzchołków oraz liczba krawędzi) znalezionego niekoniecznie spójnego podgrafu powinien być równy oryginalnemu grafu, w szczególności jeśli graf oryginalny jest niespójny.

Sprawdzanie niespójności sprzyja wykryciu dodatkowych błędów implementacyjnych gdyż moduł poszukiwania niespójnych składowych korzysta z rekurencyjnego zagłębienia algorytmu szukania spójnych składowych.

Z grafu G zostają usuwane kolejno wierzchołki i sprawdzany jest nowy rozmiar znalezionego izomorficznego podgrafu oraz nowa liczba krawędzi. W tym zestawie testów również sprawdzane są przypadki gdy graf G jest mniejszy niż H oraz się w nim zawiera (nawet jeśli G nie jest spójny).

5.3 Trzeci zestaw testów

Trzeci zestaw testów dotyczy losowych grafów o nierównych licznościach zbiorów wierzchołków:

- dla $|V_G|$: 1-10
- dla $|V_H|$: 4-40 (czterokrotność liczby wierzchołków grafu $|V_G|$)

Sprawdzana jest wyłącznie różnowartościowość m oraz izomorficzność znalezionego podgrafu. Nie jest jednak sprawdzana maksymalność m.

Sprawdzana jest poprawność algorytmu dokładnego oraz obu algorytmów aproksymacyjnych dla wyżej wymienionych grafów.

5.4 Czwarty zestaw testów

Graf G składa się z dwóch dostatecznie dużych klik o niekoniecznie równych rozmiarach oraz z jednego łańcucha o długości[1] 5 łączącego jeden wierzchołek jednej kliki z innym wierzchołkiem z drugiej kliki. Graf H jest tworzony identycznie z tą różnicą, że łączący kliki łańcuch jest krótszy: ma długość 4 krawędzi.

Niech i oznacza liczbę wierzchołków w jednej klice, a j w drugiej. Wówczas: $|V_G| = i + j + 4$, $|E_G| = \binom{i}{2} + \binom{j}{2} + 5$ oraz $|V_H| = i + j + 3$, $|E_H| = \binom{i}{2} + \binom{j}{2} + 4$.

Dodatkowo wierzchołki grafu są permutowane aby uniknąć podobieństwa między indeksami wierzchołków V_G i V_H.

Sprawdzana jest maksymalna liczba krawędzi – oczywiście dla algorytmu znajdującego maksymalny niekoniecznie spójny izomorficzny podgraf z naciskiem na *niekoniecznie spójny*. Powinna ona wynosić $\binom{i}{2} + \binom{j}{2} + 2$. Liczba wierzchołków izomorficznego podgrafu powinna wynosić $i + j + 2$. Sprawdzana jest również izomorficzność oraz różnowartościowość znalezionego mapowania.

Algorytm pomyślnie przechodzi testy dla $4 \leqslant i \leqslant 10$ oraz $3 \leqslant j \leqslant i$.

5.5 Piąty zestaw testów

Podobnie do czwartego zestawu tworzony jest graf G oraz H tylko tym razem środek dłuższego łańcucha zostaje złączony tak, że powstaje na samym środku trójkąt o trzech wierzchołkach. Algorytm szukania spójnego podgrafu izomorficznego powinien odnaleźć spójny izomorficzny podgraf równy mniejszemu grafu czyli równy większemu po wykluczeniu odstającego wierzchołka z trójkąta na środku łańcucha w większym grafie.

[1] Liczba krawędzi pomiędzy wierzchołkami należącymi do różnych klik.

5.6 Szósty zestaw testów

Dla każdego z wartościowań $\kappa(G) = |V_G|$, $\kappa(G) = |V_G| + |E_G|$ oraz $\kappa(G) = |V_G| \cdot |E_G|$ dla losowych grafów o gęstości $\rho = 10\%, 20\%, \ldots, 90\%$ oraz liczbie wierzchołków $i = 1, 2, \ldots, 10$ dla grafu G oraz $j = 1, 2, \ldots, i-1$ dla grafu H sprawdzam, czy wynik pierwszego algorytmu aproksymacyjnego (losowanie wierzchołków) nie przekracza wyniku algorytmu dokładnego. Dysponując dostatecznie dużymi zasobami czasu algorytm aproksymacyjny uzyskuje wyniki rzędu 90% - 100% ale nigdy więcej niż 100% (względem wyniku algorytmu dokładnego). Uważam to za niebanalny, wręcz liczący się argument przemawiający za poprawnością algorytmu dokładnego. Warto też wspomnieć, że dysponując dostatecznie dużymi zasobami czasowymi drugi algorytm aproksymacyjny w końcu zwróci poprawny wynik, a pierwszy będzie zbiegał do poprawnego rozwiązania.

5.7 Kolejne, mnogie zestawy testów

Zestawy $1-3$ zostały powtórzone dla różnych gęstości grafów, 10%, 50%, 70% oraz 90%.

Zostały wykonane również inne testy o większej liczności (w sumie 7,5 mln) dla mniejszej[2] liczby wierzchołków (1-5).

5.8 Wyniki testów

Wszystkie algorytmy przechodzą odpowiadające im testy poprawnościowe.

[2]Wśród większych grafów jest o wiele trudniej znaleźć podchwytliwy przypadek który i tak najpewniej występuje w mniejszym grafie.

Rozdział 6

Rzeczywista wydajność algorytmów oraz jakość aproksymacji algorytmów aproksymacyjnych

Okazuje się, że można uzyskać znaczny wzrost wydajności rezygnując z modułu parsującego wejście do kryterium wartościowania[1]. Największą wydajność osiąga się korzystając z już skompilowanej funkcji wartościującej (funkcja lambda C#).

Kolejnym zabiegiem jest zastosowanie obliczeń równoległych. Niekorzystne ścieżki są skutecznie pomijane w drzewie obliczeń. Algorytm przypomina aukcję bez ograniczenia czasowego gdzie nabywcami są wątki szukające rozwiązania (każdy wątek rozważa przypadki z ograniczeniem izomorficzności pewnej określonej pary $(v^0_{G^*}, v^0_H)$).

[1]Dokładniej: uruchomić moduł tylko wtedy kiedy to będzie konieczne np. dla bardziej skomplikowanych wartościowań.

6.1 Jakość aproksymacji oraz wydajność pierwszego algorytmu

Obliczenia trwały od kilku ($|G| = |H| \leqslant 13$) do kilkudziesięciu milisekund dla $|G| = |H| \leqslant 21$. Jakość aproksymacji jest zaprezentowana poniżej w formie tabeli dla różnych wartościowań (większym grafom przysługiwało więcej czasu na obliczenia stąd dla mniejszych przypadków obliczenia trwały milisekundy ale nie zawsze ze stuprocentową skutecznością):

Wartościowanie $\kappa(G) = |V_G|$

$\|G\|$	$\|H\|$	$\rho = 5\%$	$\rho = 15\%$	$\rho = 25\%$	$\rho = 35\%$	$\rho = 45\%$	$\rho = 55\%$	$\rho = 65\%$	$\rho = 75\%$	$\rho = 85\%$	$\rho = 95\%$
4	4	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
5	5	100.0%	50.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
6	6	100.0%	100.0%	100.0%	100.0%	80.0%	100.0%	100.0%	100.0%	100.0%	100.0%
7	7	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	80.0%	80.0%	100.0%	100.0%
8	8	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
9	9	100.0%	100.0%	100.0%	100.0%	100.0%	85.7%	100.0%	100.0%	100.0%	100.0%
10	10	100.0%	100.0%	87.5%	100.0%	87.5%	100.0%	100.0%	77.8%	100.0%	100.0%
11	11	100.0%	88.9%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	90.0%	100.0%
12	12	50.0%	100.0%	100.0%	100.0%	88.9%	100.0%	100.0%	100.0%	100.0%	100.0%
13	13	100.0%	87.5%	100.0%	100.0%	88.9%	88.9%	100.0%	100.0%	90.9%	100.0%
14	14	50.0%	100.0%	100.0%	100.0%	100.0%	90.0%	90.0%	100.0%	100.0%	92.3%
15	15	100.0%	100.0%	100.0%	90.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
16	16	100.0%	90.9%	100.0%	100.0%	100.0%	100.0%	100.0%	90.9%	100.0%	100.0%
17	17	100.0%	100.0%	100.0%	100.0%	100.0%	90.9%	90.9%	84.6%	92.3%	93.3%
18	18	100.0%	100.0%	91.7%	90.9%	90.0%	90.9%	91.7%	92.3%	92.9%	94.1%
19	19	100.0%	100.0%	91.7%	91.7%	100.0%	90.9%	91.7%	92.3%	92.9%	100.0%
20	20	100.0%	93.3%	100.0%	91.7%	100.0%	100.0%	91.7%	92.3%	86.7%	100.0%
21	21	100.0%	92.9%	100.0%	100.0%	100.0%	90.9%	91.7%	92.9%	93.8%	100.0%

Wartościowanie $\kappa(G) = |V_G| + |E_G|$

$\|G\|$	$\|H\|$	$\rho = 5\%$	$\rho = 15\%$	$\rho = 25\%$	$\rho = 35\%$	$\rho = 45\%$	$\rho = 55\%$	$\rho = 65\%$	$\rho = 75\%$	$\rho = 85\%$	$\rho = 95\%$
4	4	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
5	5	100.0%	33.3%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
6	6	100.0%	100.0%	100.0%	100.0%	72.7%	100.0%	100.0%	100.0%	100.0%	100.0%
7	7	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	71.4%	100.0%	100.0%
8	8	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
9	9	100.0%	100.0%	100.0%	100.0%	100.0%	80.0%	100.0%	100.0%	100.0%	100.0%
10	10	100.0%	100.0%	88.2%	93.3%	90.5%	85.0%	100.0%	100.0%	97.6%	100.0%
11	11	100.0%	88.2%	93.3%	100.0%	90.9%	88.0%	100.0%	100.0%	84.3%	100.0%
12	12	100.0%	100.0%	100.0%	100.0%	100.0%	96.2%	96.4%	100.0%	100.0%	100.0%
13	13	100.0%	86.7%	100.0%	91.3%	88.0%	96.4%	100.0%	100.0%	86.4%	100.0%
14	14	100.0%	85.7%	100.0%	91.3%	95.8%	87.2%	85.0%	100.0%	86.7%	87.4%
15	15	100.0%	95.0%	90.9%	96.2%	96.8%	100.0%	97.8%	93.8%	100.0%	100.0%
16	16	100.0%	90.5%	100.0%	100.0%	90.9%	94.1%	97.8%	87.3%	100.0%	100.0%
17	17	100.0%	100.0%	96.3%	96.6%	100.0%	88.0%	89.8%	83.5%	88.8%	88.6%
18	18	100.0%	100.0%	92.9%	100.0%	86.8%	92.9%	87.3%	89.5%	88.5%	89.9%
19	19	100.0%	100.0%	87.5%	84.4%	89.7%	90.9%	85.9%	86.4%	88.7%	100.0%
20	20	100.0%	96.9%	100.0%	94.1%	90.2%	84.8%	81.7%	93.1%	76.9%	99.3%
21	21	100.0%	90.3%	93.8%	94.1%	100.0%	91.7%	84.1%	87.4%	88.8%	90.3%

6.2 Jakość aproksymacji oraz wydajność drugiego algorytmu

Na potrzeby testów wydajnościowych czas obliczeń drugiego algorytmu był identyczny jak czas obliczeń pierwszego algorytmu aproksymacyjnego (tak naprawdę najpierw uruchomiłem drugi algorytm a następnie pierwszemu nakazałem tak długo liczyć jak długo trwały obliczenia drugiego algorytmu). Stała określająca maksymalną liczbę wywołań Rekursji wynosiła zaledwie $c = 3$ (zwykle wskazane jest aby $c = 30$ lub $c = 300$). Liczba wywołań była liczona od głębokości $d = 0$. Zatem górna złożoność algorytmu wynosi $\mathcal{O}(D^6)$.

Wartościowanie $\kappa(G) = |V_G|$

$\|G\|$	$\|H\|$	$\rho = 5\%$	$\rho = 15\%$	$\rho = 25\%$	$\rho = 35\%$	$\rho = 45\%$	$\rho = 55\%$	$\rho = 65\%$	$\rho = 75\%$	$\rho = 85\%$	$\rho = 95\%$
4	4	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
5	5	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
6	6	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
7	7	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
8	8	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
9	9	100.0%	100.0%	85.7%	100.0%	85.7%	85.7%	100.0%	100.0%	100.0%	100.0%
10	10	100.0%	100.0%	87.5%	85.7%	87.5%	100.0%	100.0%	100.0%	100.0%	100.0%
11	11	100.0%	100.0%	100.0%	100.0%	87.5%	87.5%	100.0%	100.0%	100.0%	100.0%
12	12	100.0%	100.0%	100.0%	100.0%	88.9%	100.0%	100.0%	100.0%	100.0%	100.0%
13	13	100.0%	100.0%	100.0%	88.9%	88.9%	88.9%	88.9%	100.0%	100.0%	100.0%
14	14	100.0%	100.0%	90.0%	88.9%	100.0%	90.0%	80.0%	90.0%	100.0%	100.0%
15	15	100.0%	90.0%	100.0%	100.0%	90.0%	100.0%	100.0%	100.0%	100.0%	100.0%
16	16	100.0%	100.0%	100.0%	90.0%	90.0%	100.0%	100.0%	90.9%	100.0%	100.0%
17	17	100.0%	91.7%	90.9%	100.0%	90.0%	90.9%	90.9%	92.3%	100.0%	100.0%
18	18	100.0%	92.3%	91.7%	90.9%	90.0%	81.8%	91.7%	92.3%	100.0%	100.0%
19	19	100.0%	90.9%	83.3%	83.3%	90.9%	90.9%	91.7%	92.3%	100.0%	100.0%
20	20	100.0%	86.7%	92.3%	83.3%	90.9%	90.9%	91.7%	92.3%	100.0%	100.0%
21	21	100.0%	92.9%	92.3%	91.7%	83.3%	90.9%	91.7%	100.0%	93.8%	94.4%

Wartościowanie $\kappa(G) = |V_G| + |E_G|$

$\|G\|$	$\|H\|$	$\rho = 5\%$	$\rho = 15\%$	$\rho = 25\%$	$\rho = 35\%$	$\rho = 45\%$	$\rho = 55\%$	$\rho = 65\%$	$\rho = 75\%$	$\rho = 85\%$	$\rho = 95\%$
4	4	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
5	5	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
6	6	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
7	7	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
8	8	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
9	9	100.0%	100.0%	85.7%	100.0%	87.5%	85.0%	100.0%	100.0%	100.0%	100.0%
10	10	100.0%	100.0%	88.2%	93.3%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
11	11	100.0%	100.0%	100.0%	100.0%	90.9%	88.0%	100.0%	100.0%	100.0%	100.0%
12	12	100.0%	100.0%	90.0%	100.0%	100.0%	100.0%	96.4%	100.0%	98.0%	100.0%
13	13	100.0%	100.0%	94.4%	87.0%	92.0%	96.4%	90.9%	100.0%	100.0%	100.0%
14	14	100.0%	95.2%	90.5%	91.3%	95.8%	87.2%	75.0%	86.7%	100.0%	100.0%
15	15	100.0%	90.0%	86.4%	84.6%	83.9%	100.0%	100.0%	100.0%	100.0%	100.0%
16	16	100.0%	100.0%	91.3%	69.7%	100.0%	85.3%	89.1%	87.3%	100.0%	100.0%
17	17	100.0%	91.3%	85.2%	89.7%	96.9%	86.0%	83.7%	87.3%	100.0%	100.0%
18	18	100.0%	88.9%	85.7%	77.4%	89.5%	85.7%	88.9%	89.5%	100.0%	100.0%
19	19	100.0%	84.0%	84.4%	78.1%	82.1%	90.9%	89.1%	87.7%	100.0%	100.0%
20	20	100.0%	81.3%	93.3%	73.5%	78.0%	84.8%	83.3%	95.8%	100.0%	100.0%
21	21	100.0%	83.9%	81.3%	79.4%	74.4%	85.4%	82.5%	97.9%	89.6%	90.3%

Wartościowanie $\kappa(G) = |E_G|$, podgraf niekoniecznie spójny ($c = 40$)

$\|G\|$	$\|H\|$	$\rho = 5\%$	$\rho = 15\%$	$\rho = 25\%$	$\rho = 35\%$	$\rho = 45\%$	$\rho = 55\%$	$\rho = 65\%$	$\rho = 75\%$	$\rho = 85\%$	$\rho = 95\%$
4	4	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
5	5	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
6	6	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
7	7	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
8	8	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
9	9	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
10	10	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
11	11	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
12	12	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
13	13	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
14	14	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
15	15	100.0%	100.0%	100.0%	93.3%	100.0%	92.0%	100.0%	100.0%	100.0%	100.0%
16	16	100.0%	100.0%	100.0%	100.0%	91.7%	96.0%	100.0%	100.0%	88.2%	100.0%
17	17	100.0%	100.0%	100.0%	100.0%	88.0%	93.8%	88.1%	100.0%	100.0%	100.0%
18	18	100.0%	100.0%	93.3%	94.7%	92.0%	96.9%	87.8%	100.0%	100.0%	100.0%
19	19	100.0%	100.0%	93.8%	90.5%	92.6%	91.2%	97.6%	88.9%	98.8%	100.0%
20	20	100.0%	100.0%	88.9%	83.3%	92.9%	84.6%	97.8%	100.0%	100.0%	100.0%
21	21	100.0%	93.3%	89.5%	83.3%	93.5%	95.1%	88.5%	97.1%	98.2%	100.0%

6.3 Porównanie empiryczne algorytmów aproksymacyjnych

Dokonując empirycznych porównań wyników dla grafów o liczbie wierzchołków 40 oraz 30 dla gęstości $\rho = 1\%, 2\%, 3\%, \ldots, 100\%$ o identycznym czasie wykonania można dostrzec tendencję: dla $\rho < \frac{1}{2}$ lepsze wyniki daje pierwszy algorytm aproksymacyjny (losowy dobór wierzchołków), natomiast dla $\rho > \frac{1}{2}$ lepszy wynik daje drugi algorytm aproksymacyjny (ograniczona rekursja). Jest to argument przemawiający za skutecznością obu algorytmów aproksymacyjnych, gdyż żaden nie dominuje całkowicie nad drugim oraz (dla grafów o mniejszym rozmiarze) wyniki aproksymacji są zbliżone do dokładnych, a są one obliczane w naprawdę krótkim czasie.

6.4 Pomiar czasu obliczeń algorytmu dokładnego

Wartościowanie $\kappa(G) = |V_G|$

$\|V_G\|$	$\|V_H\|$	$\rho = 5\%$	$\rho = 15\%$	$\rho = 25\%$	$\rho = 35\%$	$\rho = 45\%$	$\rho = 55\%$	$\rho = 65\%$	$\rho = 75\%$	$\rho = 85\%$	$\rho = 95\%$
4	4	28.7ms[2]	0.4ms	0.1ms	0.2ms	0.2ms	0.1ms	0.1ms	0.3ms	0.2ms	0.2ms
5	5	0.2ms	0.2ms	0.2ms	0.3ms	0.3ms	0.2ms	0.2ms	0.2ms	0.3ms	0.2ms
6	6	0.3ms	0.2ms	0.2ms	0.4ms	0.2ms	0.3ms	0.2ms	0.2ms	0.5ms	0.2ms
7	7	0.3ms	0.4ms	0.3ms	0.4ms	0.4ms	0.3ms	0.6ms	0.2ms	0.3ms	0.3ms
8	8	2.7ms	0.3ms	0.4ms	0.4ms	0.5ms	0.9ms	0.8ms	0.7ms	0.4ms	0.4ms
9	9	1.4ms	0.4ms	0.4ms	0.8ms	1.2ms	1.2ms	1.1ms	1.5ms	0.6ms	0.6ms
10	10	0.3ms	0.7ms	1.5ms	1.5ms	2.2ms	3.2ms	2.3ms	6.7ms	1.9ms	1.0ms
11	11	1.4ms	0.8ms	2.0ms	3.4ms	9.1ms	6.3ms	6.2ms	3.6ms	1.2ms	2.3ms
12	12	0.5ms	3.1ms	3.3ms	10.5ms	13.6ms	15.3ms	16.5ms	19.2ms	3.7ms	3.4ms
13	13	0.6ms	0.9ms	6.0ms	17.8ms	33.1ms	27.9ms	59.1ms	12.3ms	9.7ms	1.9ms
14	14	0.7ms	2.2ms	12.1ms	47.3ms	69.4ms	60.7ms	62.8ms	27.1ms	31.9ms	16.0ms
15	15	2.1ms	14.0ms	72.8ms	118.2ms	201.1ms	165.3ms	122.1ms	83.7ms	46.0ms	35.4ms
16	16	1.2ms	4.5ms	77.2ms	258.7ms	241.0ms	264.6ms	258.8ms	119.7ms[3]	172.0ms	58.5ms
17	17	1.3ms	6.6ms	186.2ms	667.4ms	755.7ms	789.0ms	797.0ms	415.7ms	74.3ms	3670.3ms
18	18	1.1ms	27.7ms	283.0ms	1453.8ms	1802.5ms	1457.3ms	1661.1ms	812.8ms	187.0ms	613.9ms
19	19	1.3ms	45.8ms	842.4ms	4091.6ms	4571.3ms	3843.0ms	3560.8ms	2454.2ms	5336.1ms	2241.8ms
20	20	2.7ms	149.0ms	2554.2ms	8807.4ms	9476.2ms	8149.8ms	4050.0ms	4602.2ms	2028.0ms	956.5ms
21	21	3.0ms	480.4ms	6839.3ms	20942.4ms	24650.0ms	14162.4ms	8252.4ms	11931.2ms	3701.1ms	108.1ms

Wartościowanie $\kappa(G) = |V_G| + |E_G|$

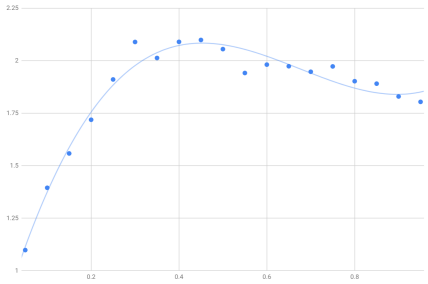
$\|V_G\|$	$\|V_H\|$	$\rho = 5\%$	$\rho = 15\%$	$\rho = 25\%$	$\rho = 35\%$	$\rho = 45\%$	$\rho = 55\%$	$\rho = 65\%$	$\rho = 75\%$	$\rho = 85\%$	$\rho = 95\%$
4	4	39.2ms	0.3ms	0.1ms	0.1ms	0.1ms	0.2ms	0.2ms	0.2ms	0.6ms	0.3ms
5	5	0.2ms	0.2ms	0.2ms	0.2ms	0.3ms	0.3ms	0.2ms	0.2ms	0.2ms	0.2ms
6	6	0.2ms	0.2ms	0.2ms	0.3ms	0.3ms	0.2ms	0.2ms	0.3ms	0.5ms	0.2ms
7	7	0.2ms	0.2ms	0.2ms	0.3ms	0.4ms	0.3ms	0.3ms	0.3ms	0.2ms	0.3ms
8	8	1.1ms	0.2ms	0.3ms	0.5ms	0.4ms	0.7ms	0.9ms	1.0ms	0.7ms	1.6ms
9	9	0.3ms	0.3ms	0.4ms	1.2ms	1.3ms	3.1ms	0.9ms	1.0ms	0.7ms	0.7ms
10	10	1.0ms	0.5ms	0.6ms	1.3ms	3.7ms	3.3ms	4.4ms	3.7ms	1.7ms	1.2ms
11	11	0.7ms	0.8ms	1.9ms	3.1ms	7.7ms	9.0ms	4.9ms	2.5ms	1.9ms	1.8ms
12	12	0.4ms	2.1ms	3.2ms	10.6ms	19.6ms	24.6ms	22.8ms	10.7ms	4.2ms	3.5ms
13	13	0.5ms	0.8ms	5.3ms	23.5ms	41.8ms	41.3ms	77.4ms	16.0ms	8.9ms	2.4ms
14	14	1.3ms	2.2ms	15.7ms	60.0ms	72.4ms	92.4ms	75.1ms	51.4ms	44.4ms	23.2ms
15	15	2.1ms	6.0ms	120.9ms	186.6ms	292.6ms	286.6ms	194.5ms	86.7ms	53.4ms	41.4ms
16	16	1.1ms	4.8ms	78.0ms	272.5ms	490.8ms	519.4ms	281.9ms	157.0ms	216.4ms	79.6ms
17	17	1.6ms	8.1ms	340.1ms	811.0ms	1183.8ms	971.4ms	840.8ms	587.4ms	550.3ms	3237.2ms
18	18	4.5ms	32.5ms	569.9ms	1816.9ms	2886.3ms	2466.5ms	2032.3ms	1262.5ms	275.8ms	537.7ms
19	19	3.8ms	62.2ms	1866.5ms	4444.5ms	6434.7ms	4347.5ms	4213.7ms	3138.9ms	5704.6ms	2329.7ms
20	20	1.9ms	179.7ms	4064.0ms	8214.8ms	13142.0ms	9342.7ms	7539.2ms	6759.3ms	2146.5ms	912.4ms
21	21	6.3ms	485.7ms	11264.7ms	26690.9ms	25684.0ms	15287.6ms	9548.2ms	12874.2ms	3997.9ms	142.9ms

[2]Pierwsze uruchomienie zawsze wiąże się z niewielkim opóźnieniem.

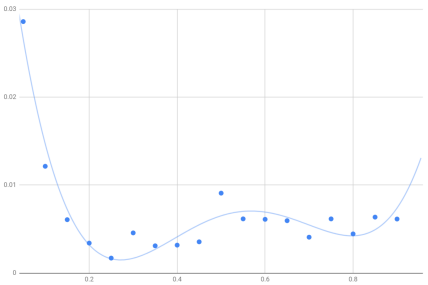
[3]Na stan z pierwszego laboratorium obliczenia algorytmu dokładnego dla szesnastu wierzchołków trwały 15 sekund, to dwa rzędy wielkości dłużej niż teraz

Wartościowanie $\kappa(G) = |E_G|$, podgraf niekoniecznie spójny

$\|V_G\|$	$\|V_H\|$	$\rho = 5\%$	$\rho = 15\%$	$\rho = 25\%$	$\rho = 35\%$	$\rho = 45\%$	$\rho = 55\%$	$\rho = 65\%$	$\rho = 75\%$	$\rho = 85\%$	$\rho = 95\%$
4	4	0.2ms	0.1ms	0.2ms	0.2ms	0.2ms	0.4ms	0.4ms	0.2ms	0.3ms	0.4ms
5	5	0.2ms	1.1ms	0.1ms	0.3ms	0.3ms	0.3ms	0.4ms	0.2ms	0.2ms	0.3ms
6	6	0.2ms	0.3ms	0.2ms	0.3ms	0.3ms	0.3ms	0.4ms	0.3ms	0.4ms	0.3ms
7	7	0.2ms	0.3ms	0.5ms	0.5ms	1.4ms	0.7ms	0.7ms	0.5ms	0.3ms	0.3ms
8	8	0.2ms	0.4ms	0.9ms	0.5ms	3.9ms	0.7ms	0.9ms	1.9ms	0.9ms	0.6ms
9	9	1.4ms	0.8ms	1.1ms	1.7ms	2.5ms	1.9ms	1.5ms	1.1ms	1.8ms	0.6ms
10	10	1.2ms	1.6ms	2.1ms	3.0ms	5.5ms	3.6ms	5.3ms	3.8ms	1.3ms	1.0ms
11	11	3.3ms	4.7ms	3.8ms	7.7ms	12.6ms	11.6ms	6.0ms	4.1ms	2.5ms	1.8ms
12	12	1.2ms	18.2ms	13.7ms	14.7ms	33.3ms	36.9ms	26.7ms	10.9ms	11.7ms	3.2ms
13	13	0.5ms	15.8ms	21.7ms	45.2ms	51.8ms	44.1ms	57.0ms	17.8ms	11.9ms	4.5ms
14	14	27.8ms	73.1ms	65.0ms	62.3ms	65.3ms	87.5ms	90.8ms	31.2ms	17.9ms	12.3ms
15	15	9.9ms	78.1ms	179.3ms	292.6ms	376.6ms	202.8ms	216.4ms	98.5ms	52.1ms	35.8ms
16	16	41.9ms	175.9ms	411.6ms	442.4ms	484.8ms	467.3ms	275.9ms	165.5ms	163.1ms	68.6ms
17	17	38.8ms	410.7ms	1065.1ms	1104.9ms	925.2ms	888.9ms	662.6ms	335.3ms	351.5ms	3142.2ms
18	18	55.8ms	750.9ms	1823.7ms	2535.1ms	2878.5ms	2674.5ms	2157.8ms	1263.8ms	314.4ms	619.6ms
19	19	36.1ms	1474.7ms	4366.1ms	5343.5ms	6549.7ms	4937.6ms	4130.6ms	3252.1ms	6336.5ms	2412.4ms
20	20	90.5ms	2208.0ms	11777.8ms	9799.3ms	14831.9ms	9464.8ms	6911.1ms	5777.7ms	1978.6ms	1033.9ms
21	21	88.9ms	12432.0ms	41411.7ms	32175.4ms	29763.7ms	18176.3ms	12089.6ms	14493.8ms	5964.5ms	145.0ms



Rysunek 6.1: Estymacja podstawy potęgi

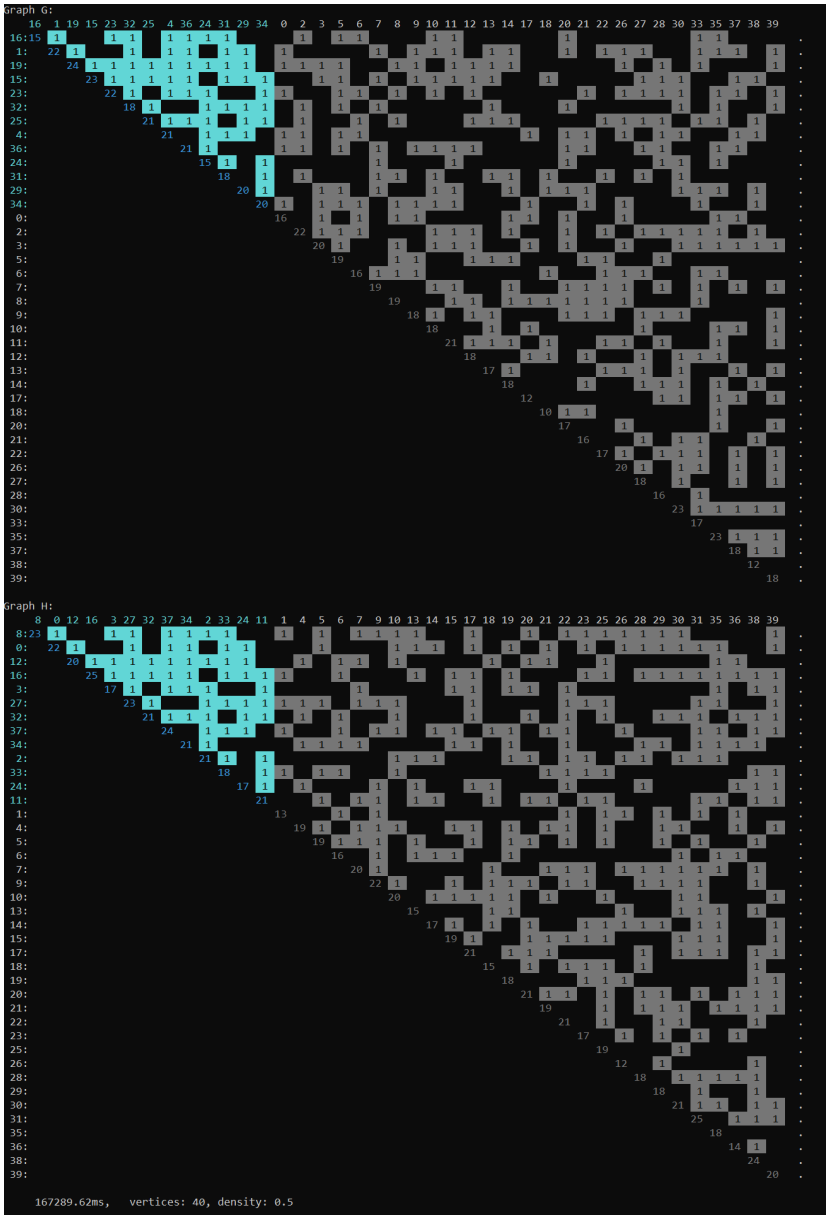


Rysunek 6.2: Estymacja stałego współczynnika dla $\rho \geqslant 10\%$

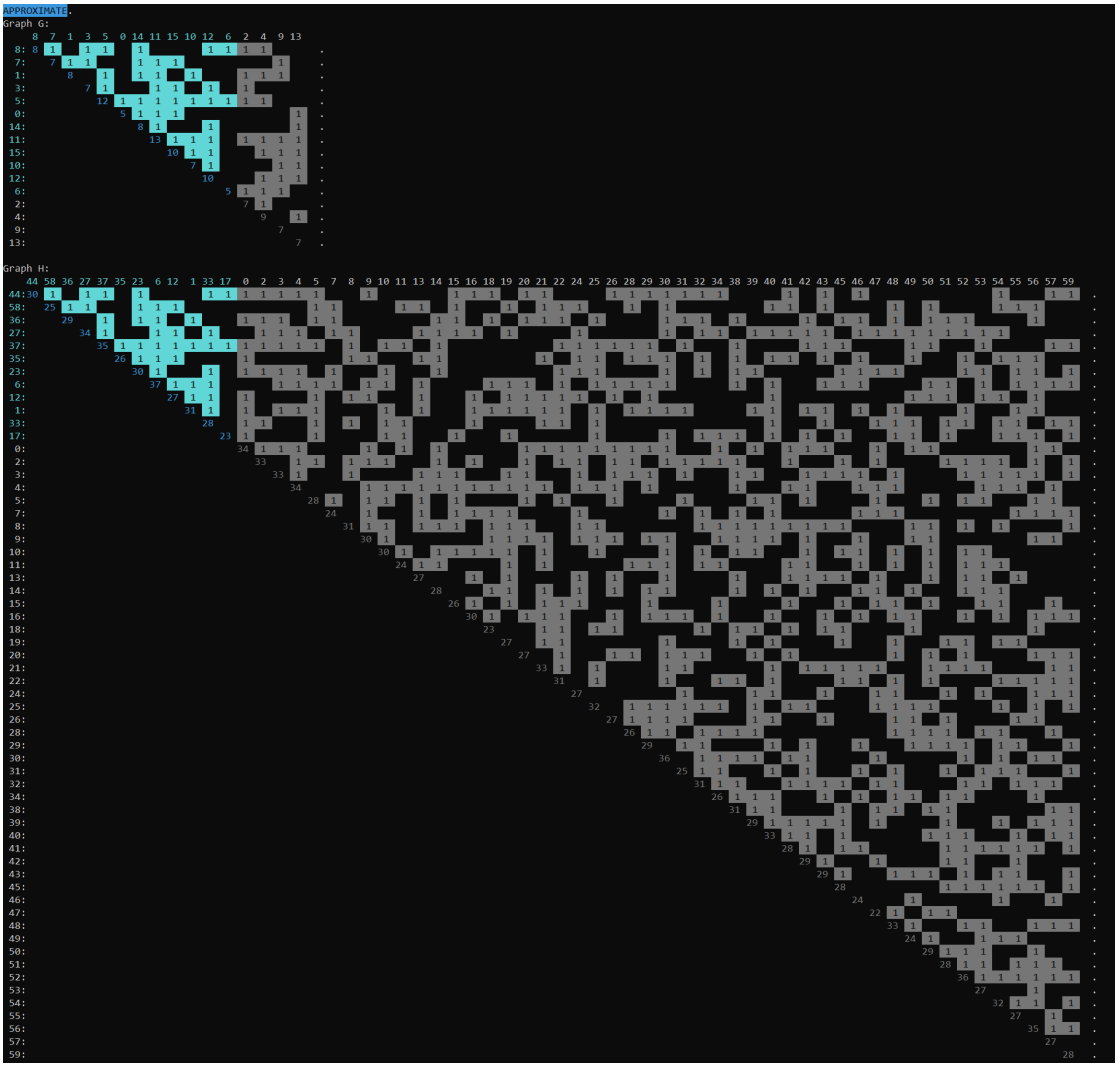
Dla wartościowania $\kappa(G) = |V_G|$ czas obliczeń w milisekundach dla grafów G i H o tej samej liczbie wierzchołków $|V_G| = |V_H| = n,\ n \in \{4, 5, \ldots, 21\}$ oraz tej samej gęstości[4] $\rho \in [5\%, 95\%]$ można oszacować za pomocą funkcji (wyjątek dla $\rho = 5\%$, wtedy stały współczynnik wynosi 0.44 gdyż czas obliczeń przypomina funkcję wielomianową bardziej niż wykładniczą):

$$T(n, \rho) := \left(0.0442 - 0.417\rho + 1.4\rho^2 - 1.85\rho^3 + 0.848\rho^4\right) \cdot \left(0.802 + 6.81\rho - 11.3\rho^2 + 5.59\rho^3\right)^n$$

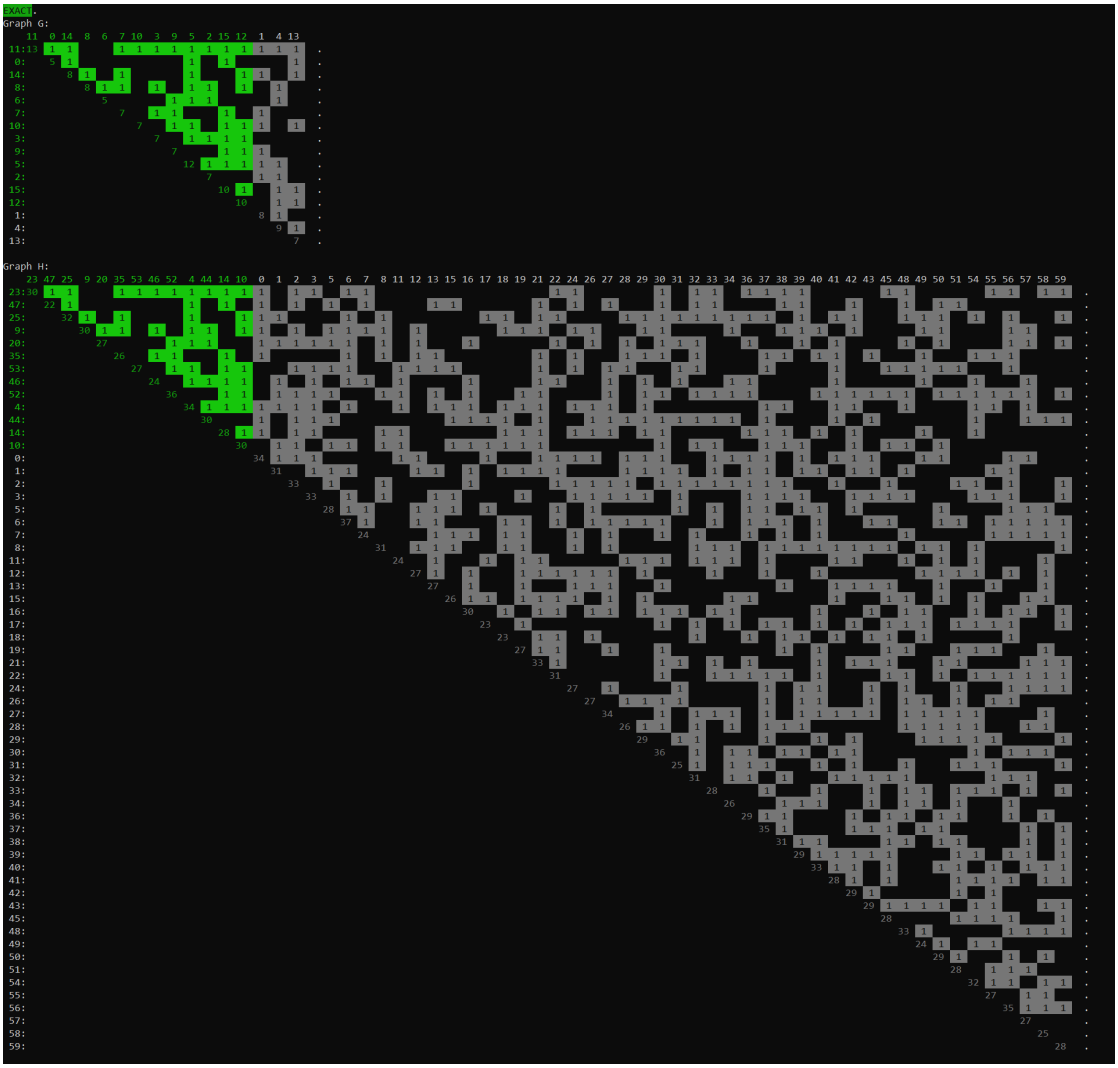
[4]Prawdopodobieństwo istnienia krawędzi w grafie podczas jego tworzenia. Docelowo jest przybliżeniem ułamka $|E_G| \div \binom{|V_G|}{2}$.



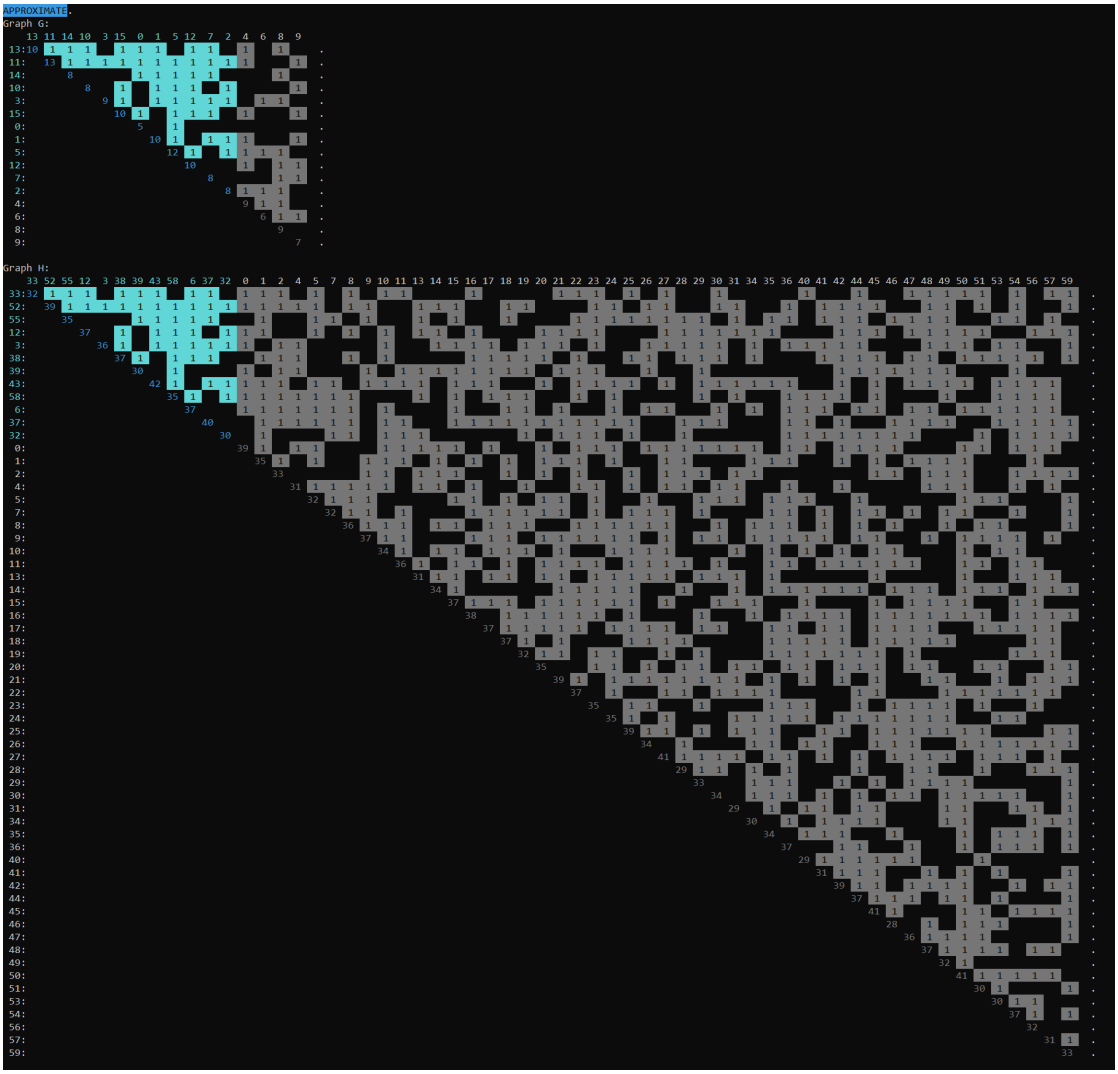
Rysunek 6.3: Wynik aproksymacji drugiego algorytmu dla $|V_G| = 40, |V_H| = 40, \rho = 50\%$. Liczby na diagonali oznaczają stopień danego wierzchołka w oryginalnym grafie.



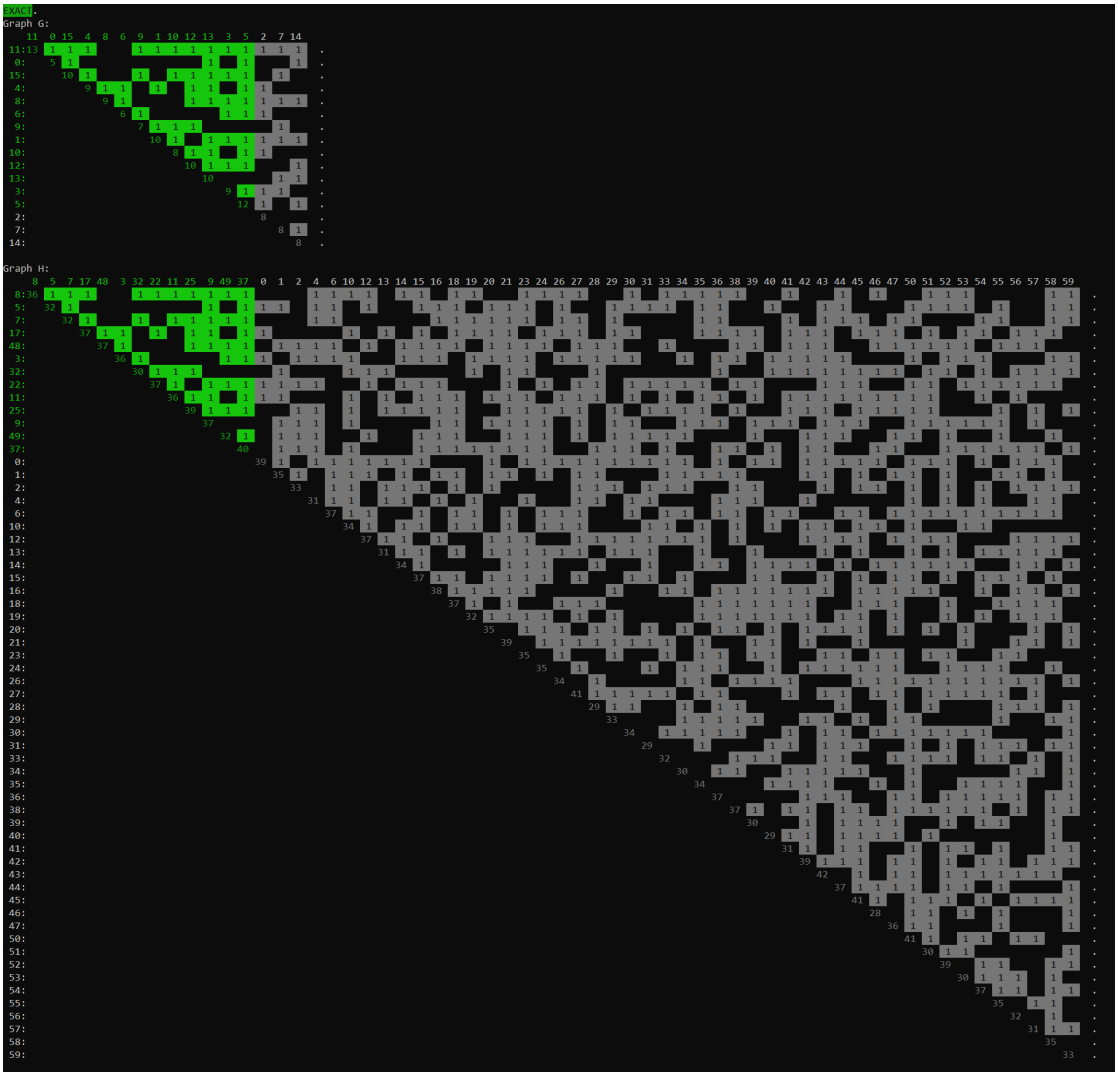
Rysunek 6.4: Wynik aproksymacji pierwszego algorytmu dla $|V_G| = 16, |V_H| = 60, \rho = 50\%$



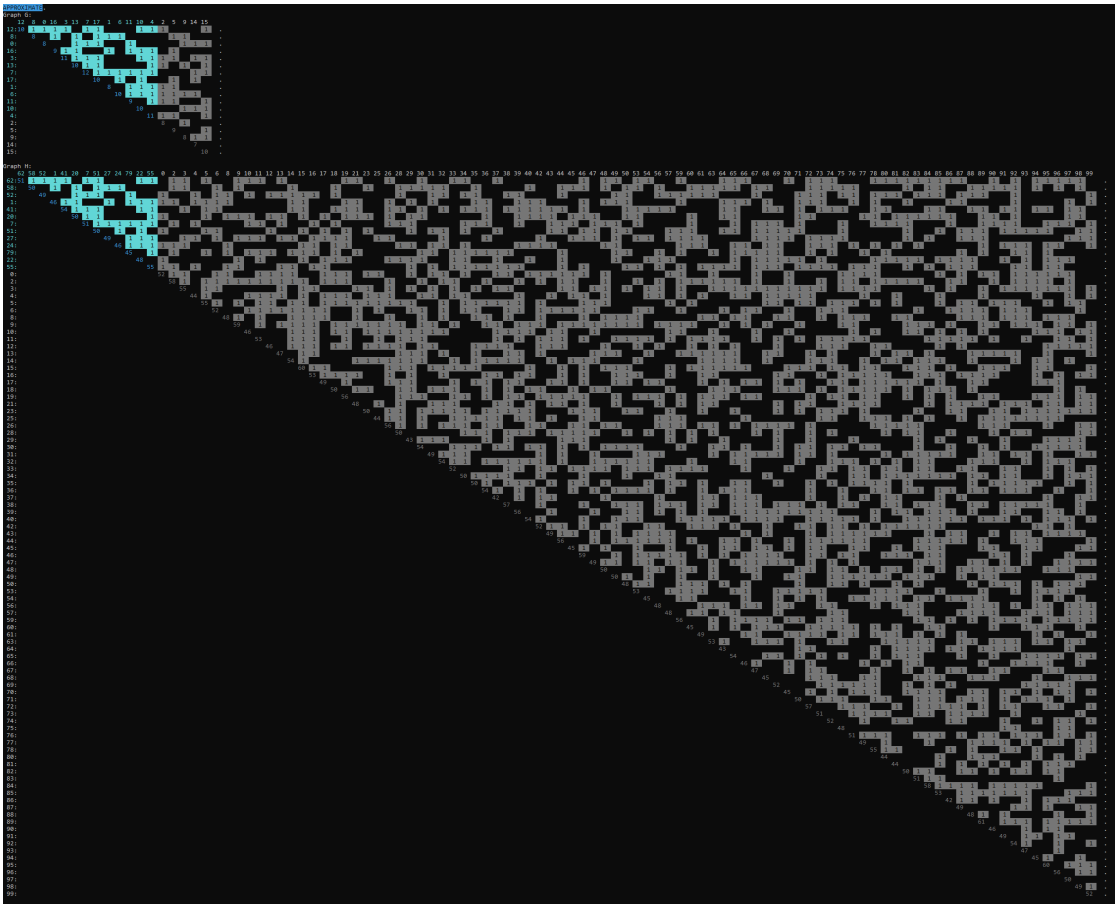
Rysunek 6.5: Wynik algorytmu dokładnego dla $|V_G| = 16, |V_H| = 60, \rho = 50\%$



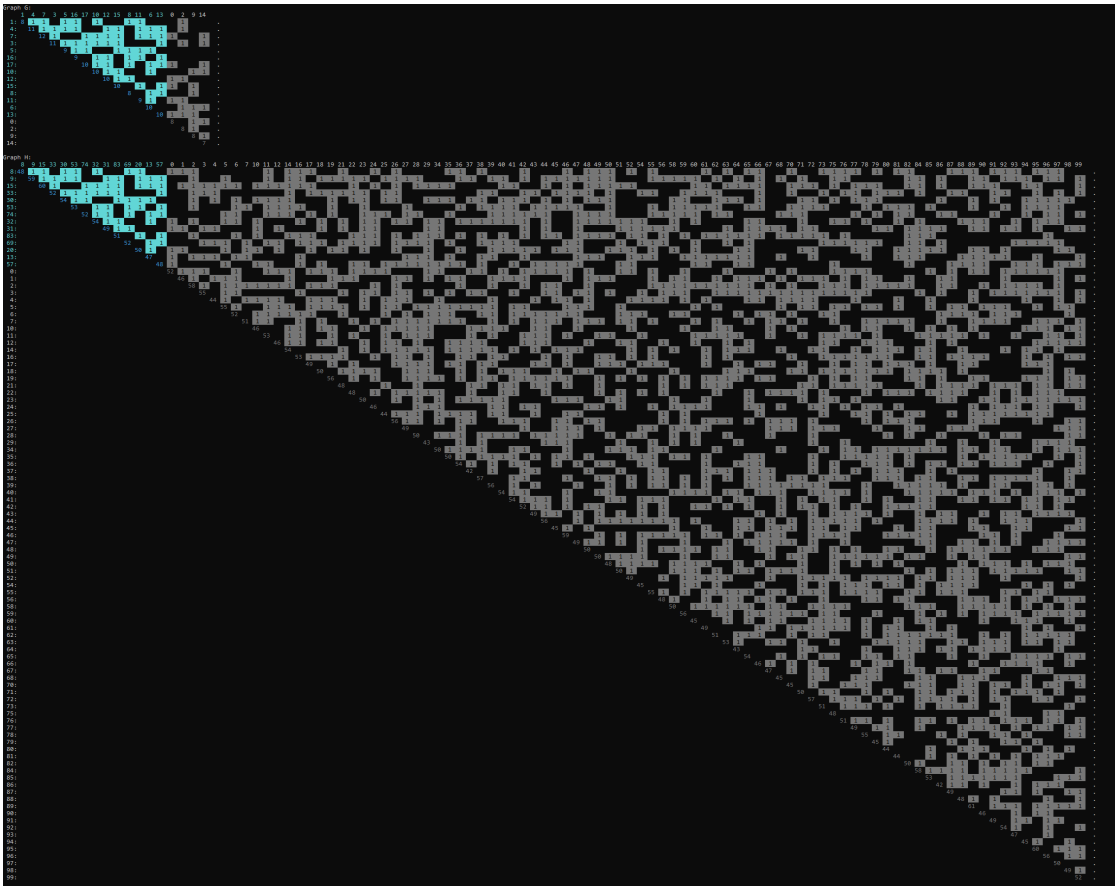
Rysunek 6.6: Wynik aproksymacji pierwszego algorytmu dla $|V_G| = 16, |V_H| = 60, \rho = 60\%$



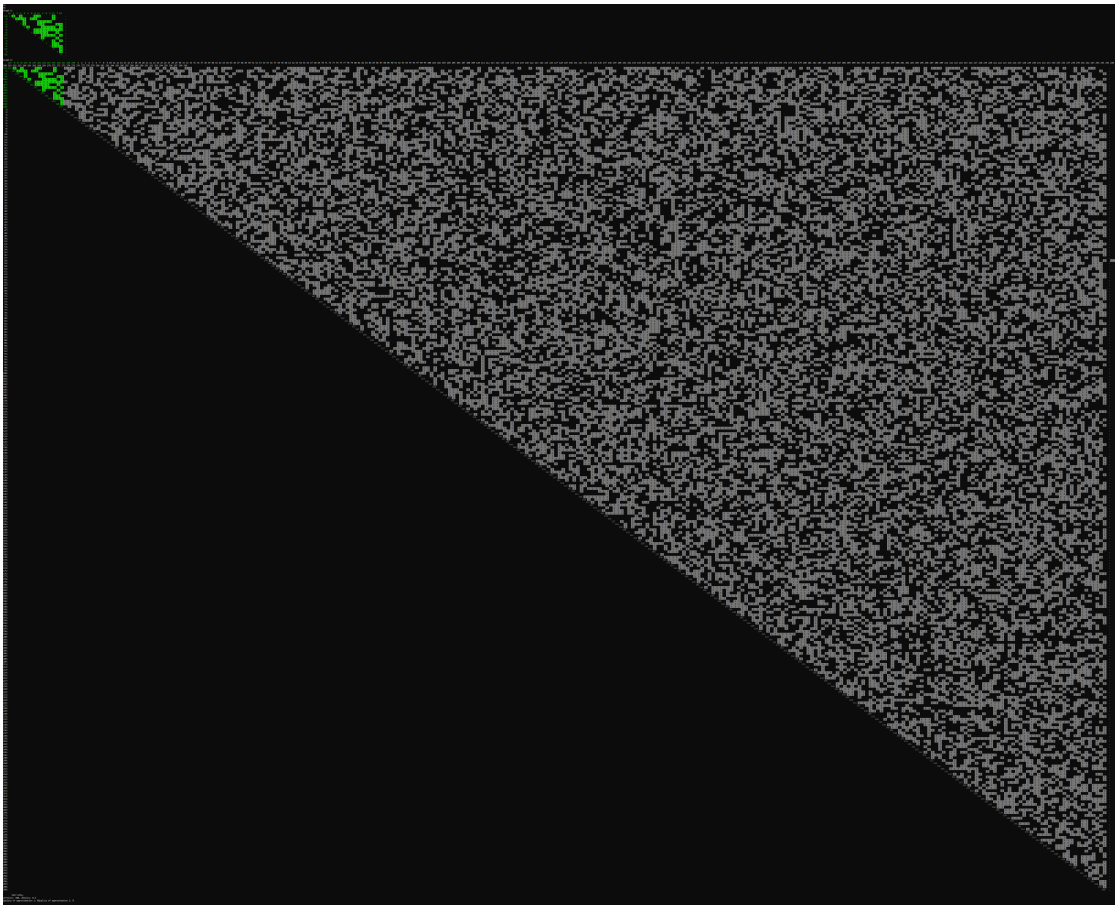
Rysunek 6.7: Wynik algorytmu dokładnego dla $|V_G| = 16, |V_H| = 60, \rho = 60\%$



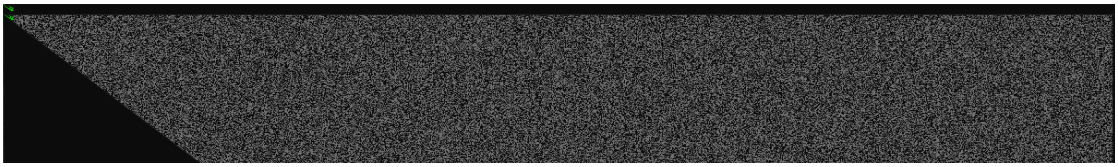
Rysunek 6.8: Wynik aproksymacji pierwszego algorytmu dla $|V_G| = 18, |V_H| = 100, \rho = 50\%$



Rysunek 6.9: Wynik aproksymacji drugiego algorytmu dla $|V_G| = 18, |V_H| = 100, \rho = 50\%$



Rysunek 6.10: Wynik algorytmu dokładnego dla $|V_G| = 15, |V_H| = 300, \rho = 50\%$



Rysunek 6.11: Wynik algorytmu dokładnego dla $|V_G| = 10, |V_H| = 1200, \rho = 50\%$. Zamieszczona jest górna część konsoli w całej szerokości. Czas wypisywania wyniku na konsolę wielokrotnie przekraczał czas obliczeń.

Powyższe szacowania dotyczą ściśle pojedynczych, konkretnych grafów dla ustalonego n, σ. Dla tej samej gęstości oraz liczbie wierzchołków czas obliczeń różni się w zależności od wzajemnego umiejscowienia krawędzi obu grafów.

Głęboka wrażliwość na wewnętrzną strukturę obu grafów podkreśla optymalność dokładnego algorytmu.