Algorytmy Optymalizacji Dyskretnej

Laboratorium

Lista 4

Kinga Majcher
272354

Styczeń 2025

1 Zadanie 1 - Maksymalny przepływ w grafie

Celem zadania jest napisanie programu, który dla podanego k wygeneruje graf hiperkostki (opisany poniżej) oraz zaimplementowanie algorytmu Edmondsa-Karpa i przetestowanie go dla hiperkostek $H_k, k \in \{1, \ldots, 16\}$. Należy przeprowadzić eksperymenty pozwalające oszacować średnie wartości przepływu, liczby ścieżek powiększających oraz czas działania programu dla wszystkich wartości k.

1.1 Charakterystyka grafu wykorzystywanego w zadaniu

Wagę Hamminga $H(x)$ dla dowolnego ciągu bitowego x definiujemy jako liczbę jedynek w tym ciągu. Dodatkowo definiujemy $Z(x)$ jako liczbę zer w danym ciągu x. Obie te wartości dla ciągu k-elementowego przyjmują wartości ze zbioru $\{0, \ldots, k\}$.

Weźmy skierowaną hiperkostkę $H_k, k \in \{1, \ldots, 16\}$, czyli graf skierowany o $|N_k| = 2^k$ wierzchołkach, których etykietami są różne ciągi binarne długości k, a krawędzie łączą wierzchołki etykietowane ciągami różniącymi się na dokładnie jednej pozycji i prowadzą z wierzchołka o mniejszej wadze Hamminga do wierzchołka z większą wagą. Wiemy wówczas, że każdy wierzchołek jest początkiem lub końcem dokładnie k łuków. Wówczas wiemy, że w takim grafie liczba łuków wynosi $|A_k| = k \cdot 2^{k-1}$.

Pojemność każdej z krawędzi u_{ij} jest losowana niezależnie i jednostajnie ze zbioru $\{1, \ldots, 2^l\}$, gdzie $l = \max\{H(e(i)), Z(e(i)), H(e(j)), Z(e(j))\}$, gdzie $e(i)$ to wartość etykiety wierzchołka i.

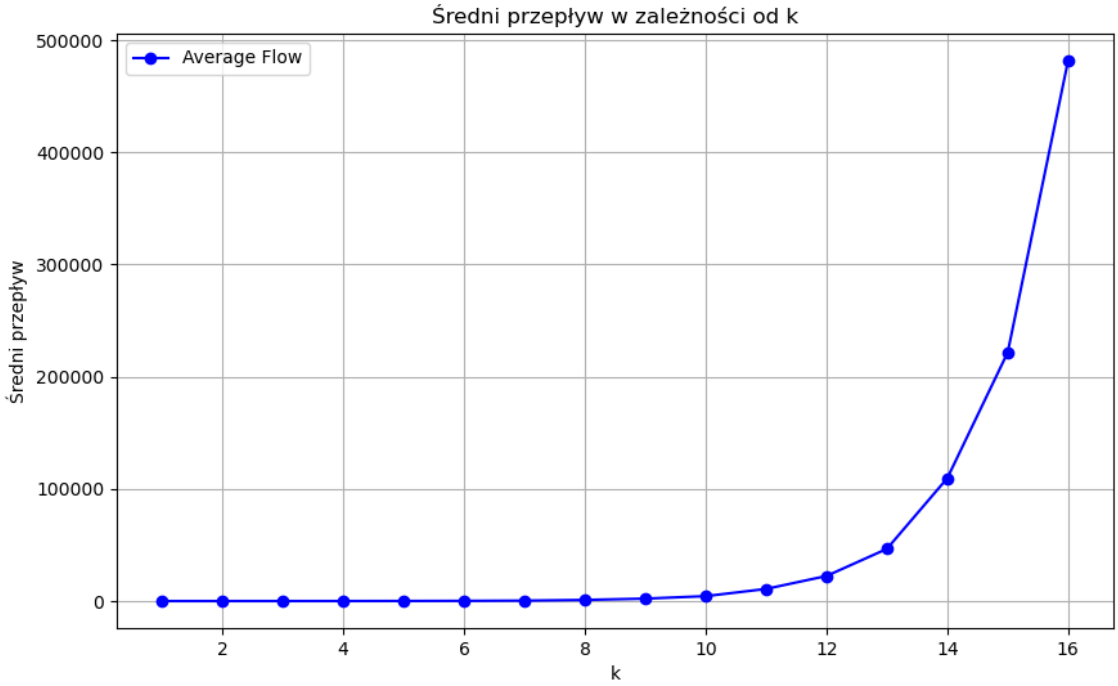
1.2 Idea algorytmu Edmondsa-Karpa

Zakładamy, że pojemności ścieżek w grafie są nieujemne. Tworzymy sieć residualną. Następnie algorytm Edmondsa-Karpa sprawdza, czy w sieci tej znajduje się ścieżka powiększająca z wierzchołka s (źródła) do wierzchołka t (ujścia). Jest to realizowane za pomocą algorytmu przeszukiwania wszerz (BFS). Jeśli taka ścieżka istnieje, to wyznaczany jest maksymalny przepływ przez tę ścieżkę (jest to najmniejsza wartość pojemności krawędzi w tej ścieżce; krawędź ta jest nazywana krawędzią krytyczną). Następnie aktualizowane są całkowity maksymalny przepływ w grafie i wartości w grafie residualnym. Algorytm wykonuje się do momentu, gdy nie istnieje więcej ścieżek powiększających w grafie residualnym.

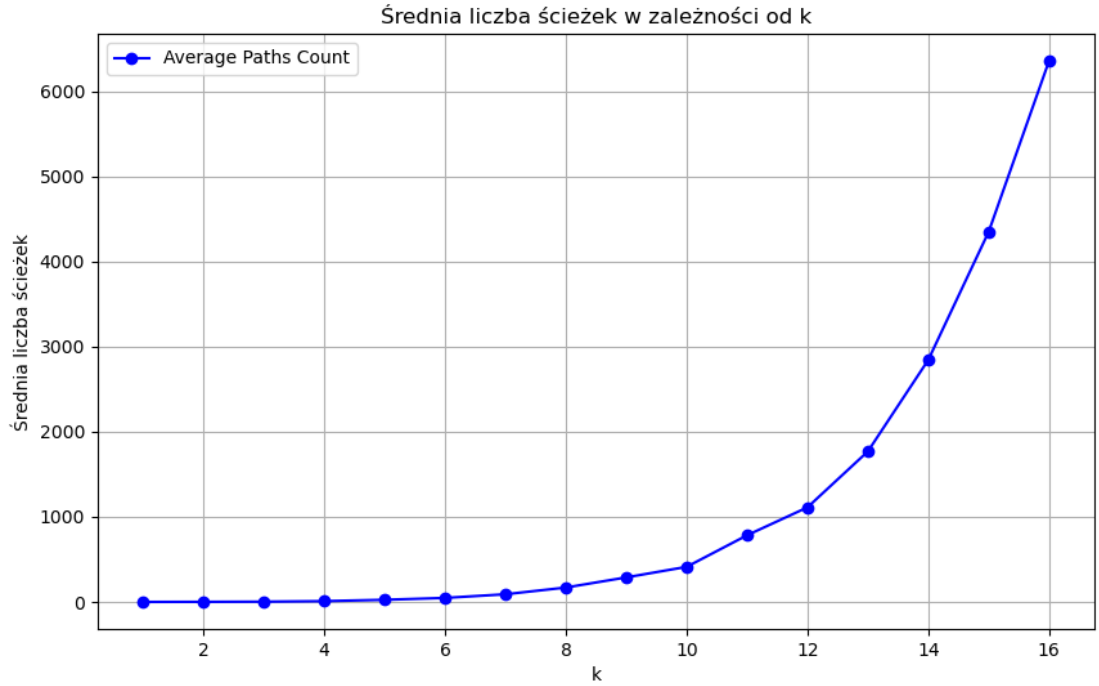
Algorytm ma złożoność worst-case $O(|N| \cdot |A|^2)$. BFS ma bowiem złożoność $O(|A|)$. Wiemy, że długość ścieżek powiększających w grafie po kolejnych iteracjach jest wartością rosnącą. Każda krawędź krytyczna w grafie residualnym znika z niego i w każda krawędź może się taką krawędzią stać, co daje nam $O(|A|)$. Każda krawędź może stać się krawędzią krytyczna w grafie residualnym maksymalnie $O(|N|)$ razy. Złożoność całego algorytmu wynosi więc $O(|N| \cdot |A|^2)$. W praktyce jednak tak duża złożoność jest rzadko osiągalna.

1.3 Testy i ich wyniki

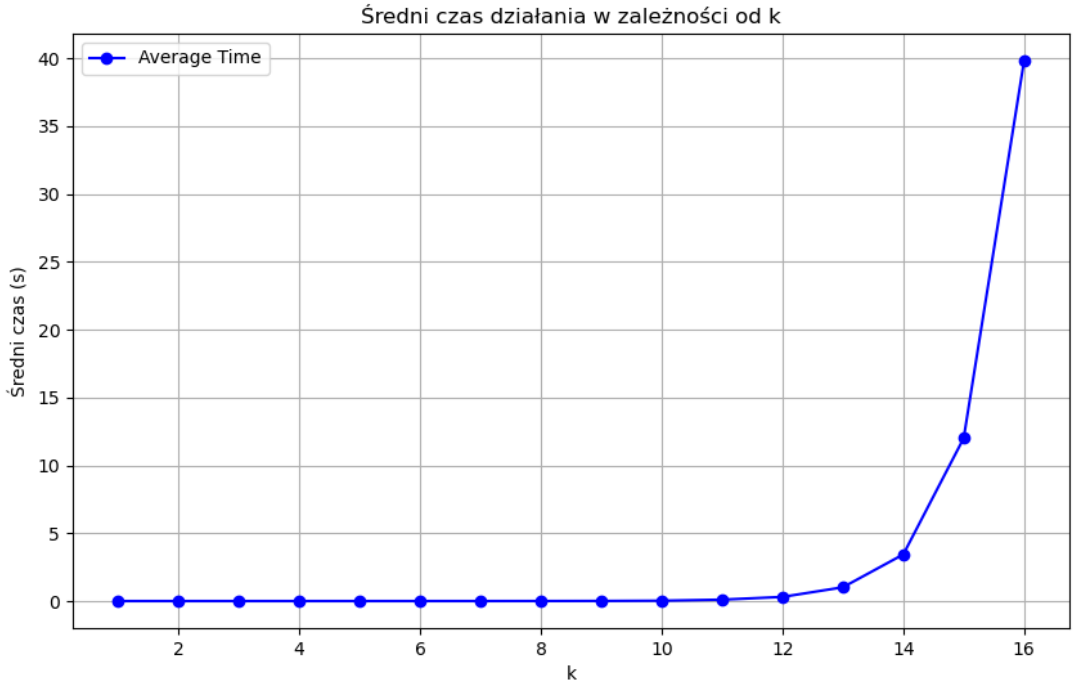
Testy algorytmu zostały wykonane na hiperkostkach $H_k, k \in \{1, \ldots, 16\}$. Dla każdej wartości k algorytm był testowany niezależnie 20 razy.



Rysunek 1: Średni przepływ w hiperkostce o wielkości k



Rysunek 2: Średnia liczba ścieżek powiększających w hiperkostce o wielkości k



Rysunek 3: Średni czas wykonywania programu dla hiperkostki o wielkości k

1.4 Wnioski

Jak można zauważyć, wraz z wzrostem wartości k rośnie każda z analizowanych wielkości. Jest to dość logiczna obserwacja, gdyż złożoność algorytmu wynosi $O(|N| \cdot |A|^2)$, co dla $|N| = 2^k$ i $|A| = k \cdot 2^{k-1}$ daje nam złożoność względem k $O(2^k \cdot k^2 \cdot 2^{2k-2}) = O(4^k)$. Algorytm Edmondsa-Karpa w przypadku hiperkostki ma więc złożoność wykładniczą względem k, co widać doskonale na wykresach. Zarówno wartość średniego przepływu, jak i średnia liczba ścieżek także mają złożoność wykładniczą względem k.

2 Zadanie 2 - Maksymalne skojarzenie w grafie

Celem zadania jest napisanie programu, który dla podanych parametrów wejściowych i i k wygeneruje odpowiedni graf (opisany poniżej) i obliczy wielkość skojarzenia o największym rozmiarze w tym grafie. Należy przeprowadzić eksperymenty pozwalające oszacować średnią wielkość maksymalnego skojarzenia i czas działania dla wartości $k \in \{2, \ldots, 10\}$ oraz $i \in \{1, \ldots, k\}$.

2.1 Charakterystyka grafu wykorzystywanego w zadaniu

Weźmy $k \in \mathbb{N}$ oraz $i \in \mathbb{N}$. Weźmy graf dwudzielny nieskierowany losowy mający dwa rozłączne podzbiory wierzchołków V_1 oraz V_2, każdy z nich o mocy 2^k. Krawędzie w tym grafie są generowane w taki sposób, że każdy wierzchołek z V_1 ma i sąsiadów z V_2 generowanych w sposób losowy niezależnie i jednostajnie.

2.2 Idea algorytmu

Maksymalne skojarzenie jest to taki zbiór wierzchołków w grafie, że żadne dwie krawędzie w tym zbiorze nie mają wspólnego wierzchołka.

Problem wyznaczania maksymalnego skojarzenia w grafie można sprowadzić do problemu wyznaczenia maksymalnego przepływu. W tym celu musimy zmodyfikować nieco graf dwudzielny.

1. Zastępujemy krawędzie w grafie krawędziami skierowanymi z V_1 do V_2.
2. Dodajemy dodatkowy wierzchołek, będący źródłem przepływu, połączony krawędziami skierowanymi z każdym z wierzchołków ze zbioru V_1.

3. Dodajemy dodatkowy wierzchołek, będący ujściem przepływu, z którym połączone są krawędziami skierowanymi wszystkie wierzchołki ze zbioru V_2.

4. Pojemność każdej z krawędzi w grafie ustawiamy na 1.

Na powyższym grafie uruchamiamy algorytm służący do wyznaczania maksymalnego przepływu.

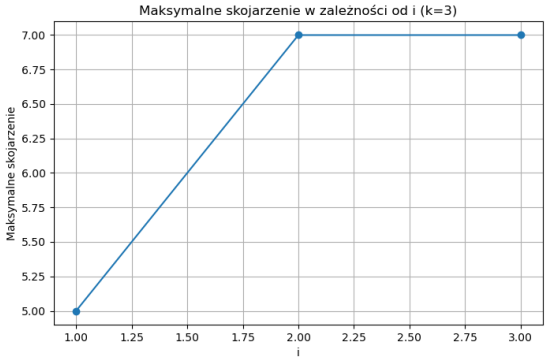
Metoda ta działa dlatego, że ze źródła do dowolnego wierzchołka z V_1 trafia tylko jedna jednostka przepływu. Również na dowolnej krawędzi łączącej wierzchołek z V_1 z wierzchołkiem z V_2 przepływ maksymalny może wynosić maksymalnie 1, a więc krawędź ta może być użyta tylko raz. Ze względu na to, że z danego wierzchołka z V_1 wypływa tylko jedna jednostka pojemności to po wybraniu krawędzi pozostałe nie mogą być już wybrane, co sprawia, że z żadnego wierzchołka z V_1 nie wybrano więcej niż jednej krawędzi. Podobne ograniczenia zachodzą w dalszej części grafu. W związku z tym, żadna z krawędzi w wyznaczonym zbiorze nie ma wspólnego wierzchołka, a więc zbiór ten jest maksymalnym skojarzeniem.

W celu wyznaczenia maksymalnego skojarzenia w grafie możemy więc wykorzystać zaprogramowany wcześniej algorytm Edmondsa-Karpa.

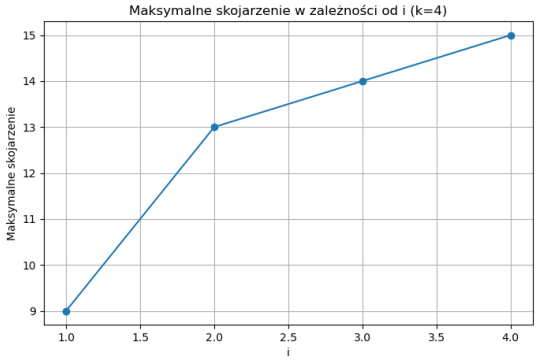
2.3 Testy i ich wyniki

Testy algorytmu zostały wykonane na grafach o $k \in \{1, \ldots, 10\}$ oraz $i \in \{1, \ldots, k\}$. Dla każdej wartości k algorytm był testowany niezależnie 20 razy.

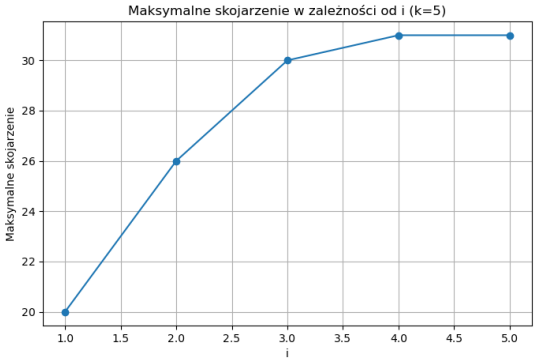
2.3.1 Wykresy maksymalnego skojarzenia



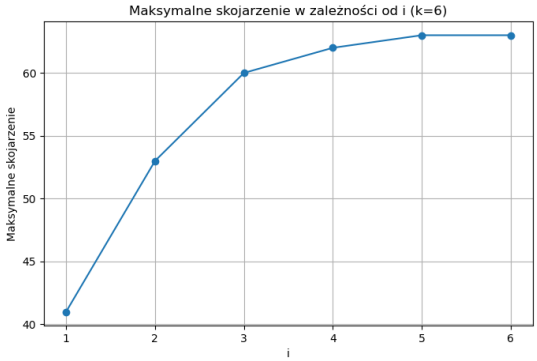
(a) Wykres dla $k = 3$.



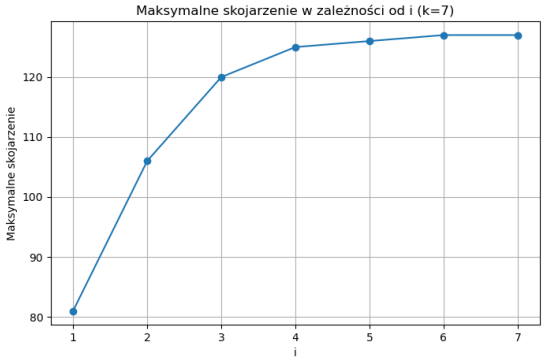
(b) Wykres dla $k = 4$.



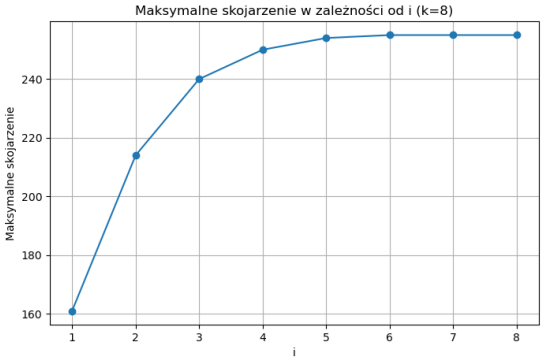
(c) Wykres dla $k = 5$.



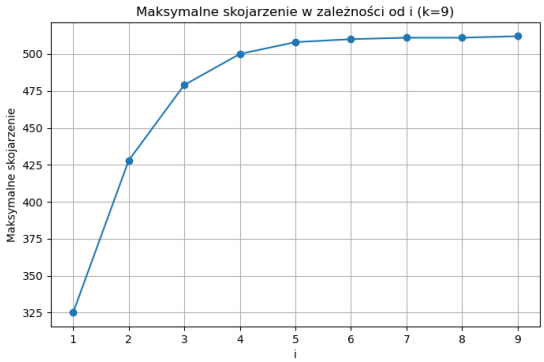
(d) Wykres dla $k = 6$.



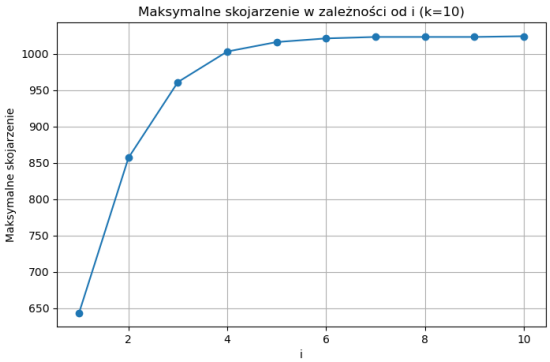
(a) Wykres dla $k = 7$.



(b) Wykres dla $k = 8$.



(c) Wykres dla $k = 9$.



(d) Wykres dla $k = 10$.

Rysunek 5: Wykresy przedstawiające wyniki dla różnych wartości k.

2.3.2 Wykresy czasu

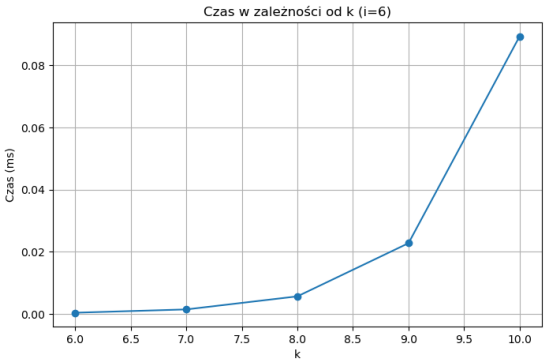
(a) Czas dla $i = 1$.

(b) Czas dla $i = 2$.

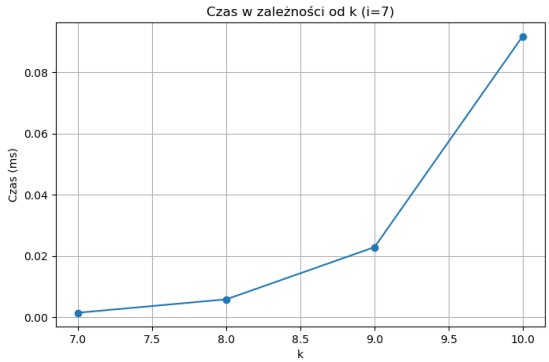
(a) Czas dla $i = 3$.

(b) Czas dla $i = 4$.

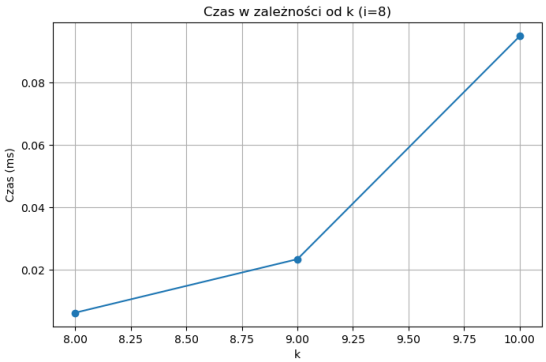
(c) Czas dla $i = 5$.



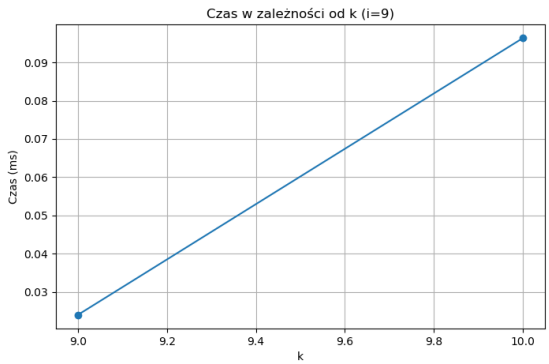
(d) Czas dla $i = 6$.



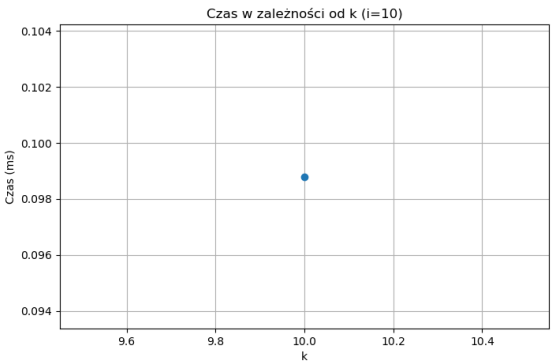
(e) Czas dla $i = 7$.



(f) Czas dla $i = 8$.



(g) Czas dla $i = 9$.



(h) Czas dla $i = 10$.

Rysunek 7: Wykresy przedstawiające czas dla różnych wartości i.

2.4 Wnioski

Patrząc na wykresy można łatwo zauważyć, że średnie maksymalne skojarzenie w zależności od i zbiega do wartości około 2^k. Liczba ta jest też maksymalnym możliwym skojarzeniem, jako, że $|V_1| = |V_2| = 2^k$, a z każdego wierzchołka może wychodzić maksymalnie jedna krawędź.

Już nawet dla $i = 1$ średnie maksymalne skojarzenie jest większe niż połowa maksymalnego, a wraz z wzrostem wartości i wartość ta zaczyna się do maksymalnej zbliżać, często osiągając ją dla $i = k$.

Czas, podobnie jak w zadaniu pierwszym, rośnie wykładniczo w stosunku do k. Moc zbioru wierzchołków w tym grafie to $2 \cdot 2^k + 2 = O(2^k)$. Moc zbioru krawędzi to natomiast $(2 + i) \cdot 2^k = O(2^k)$. Cała złożoność czasowa algorytmu wynosi więc $O(2^k \cdot 2^{k^2}) = O(4^k)$, a więc podobnie jak w zadaniu pierwszym.

Czas rośnie także wraz z wzrostem i, co również jest spodziewane, gdyż wartość ta odpowiada za liczbę krawędzi w grafie, a to ma wpływ na złożoność algorytmu.

3 Zadanie 3 - Modele programowania liniowego

Celem zadania jest uzupełnienie programów z poprzednich dwóch zadań o opcję generowania pliku z modelem programowania liniowego. Należy rozwiązać wygenerowane modele LP za pomocą solvera `glpk`, sprawdzić, czy zwraca takie same wartości maksymalnego przepływu oraz wielkości maksymalnego skojarzenia oraz porównać, który program liczy rozwiązanie szybciej.

Jako, że w obu zadaniach wykorzystujemy ten sam algorytm to sam model programowania liniowego będzie dla nich identyczny, jedyną różnicą jest wypełnienie macierzy G - w zadaniu 1 jest ona wypełniona pojemnościami krawędzi, jeśli krawędź istnieje, lub 0 jeśli nie, natomiast w zadaniu 2 będzie ona wypełniona 1 jeśli krawędź istnieje lub 0 jeśli nie.

3.1 Opis modelu

3.1.1 Dane

G_{ij} - macierz wypełniona pojemnościami krawędzi grafu (0 w sytuacji gdy krawędź nie istnieje)

3.1.2 Zmienna decyzyjna

Definiujemy zmienną decyzyjną f_{ij}, która reprezentuje przepływ na krawędzi $i \to j$. Wymuszamy, by w przypadku braku takiej krawędzi w grafie wartość f była ustawiona na 0.

3.1.3 Ograniczenia

- Przepływ na każdej z krawędzi nie może być większy niż pojemność krawędzi:

$$\forall_{(i,j)\in A} f_{ij} \leq G_{ij}$$

- Dla każdego wierzchołka suma przepływu wpływającego do niego musi być równa sumie przepływów wychodzących z niego:

$$\forall_{i\in\{1,\ldots,n\}, i\neq 1, i\neq n} \sum_{j=1}^{n} f_{ij} = \sum_{j=1} f_{ji}$$

- Przepływ na każdej krawędzi musi być nieujemny:

$$f_{ij} \geq 0$$

3.1.4 Funkcja celu

Chcemy zmaksymalizować przepływ w grafie:

$$\max \sum_{i=1}^{n} \sum_{j=1}^{n} f_{ij}$$

3.2 Testy i ich wyniki

3.2.1 Zadanie 1

- k = 2

  Maksymalny przepływ według algorytmu Edmondsa-Karpa: 3
  Maksymalny przepływ według solvera `glpk`: 3

  Czas wykonywania algorytmu Edmondsa-Karpa: 2.1211e-05s
  Czas wykonywania dla solvera `glpk`: 5.91278076171875e-5s

Krawędź	Przepływ wg. algorytmu	Przepływ wg. solvera
$0 \to 1$	1	1
$0 \to 2$	2	2
$1 \to 3$	1	1
$2 \to 3$	2	2

Tabela 1: Maksymalny przepływ wg. algorytmu Edmondsa-Karpa oraz solvera `glpk` dla $k = 2$

- k = 3

  Maksymalny przepływ według algorytmu Edmondsa-Karpa: 13
  Maksymalny przepływ według solvera `glpk`: 13

  Czas wykonywania algorytmu Edmondsa-Karpa: 2.3554e-05s
  Czas wykonywania dla solvera `glpk`: 8.106231689453125e-5s

Krawędź	Przepływ wg. algorytmu	Przepływ wg. solvera
$0 \to 1$	5	5
$0 \to 2$	6	6
$0 \to 4$	2	2
$1 \to 3$	1	1
$1 \to 5$	4	4
$2 \to 3$	4	4
$2 \to 6$	2	2
$3 \to 7$	5	5
$4 \to 5$	2	2
$4 \to 6$	0	0
$5 \to 7$	6	6
$6 \to 7$	2	2

Tabela 2: Maksymalny przepływ wg. algorytmu Edmondsa-Karpa oraz solvera `glpk` dla $k = 3$

- $k = 7$

  Maksymalny przepływ według algorytmu Edmondsa-Karpa: 525
  Maksymalny przepływ według solvera `glpk`: 525

  Czas wykonywania algorytmu Edmondsa-Karpa: 0.00251867s
  Czas wykonywania dla solvera `glpk`: 0.05552101135253906s

- $k = 10$

  Maksymalny przepływ według algorytmu Edmondsa-Karpa: 4620
  Maksymalny przepływ według solvera `glpk`: 4620

  Czas wykonywania algorytmu Edmondsa-Karpa: 0.0337379s
  Czas wykonywania dla solvera `glpk`: 31.606630086898804s

- $k = 11$

  Maksymalny przepływ według algorytmu Edmondsa-Karpa: 11777
  Maksymalny przepływ według solvera `glpk`: 11777

  Czas wykonywania algorytmu Edmondsa-Karpa: 0.113708s
  Czas wykonywania dla solvera `glpk`: 260.9325420856476s

- Dla wartości $k \geq 12$ działanie solvera zostawało unicestwione.

3.2.2 Zadanie 2

- $k = i = 2$

  Maksymalne skojarzenie według algorytmu Edmondsa-Karpa: 4
  Maksymalne skojarzenie według solvera `glpk`: 4

  Czas wykonywania algorytmu Edmondsa-Karpa: 2.4792e-05s
  Czas wykonywania dla solvera `glpk`: 0.00012493133544921875s

Skojarzenie wg. algorytmu	Skojarzenie wg. solvera
$1 \to 6$	$1 \to 6$
$2 \to 5$	$2 \to 5$
$3 \to 7$	$3 \to 7$
$4 \to 8$	$4 \to 8$

Tabela 3: Maksymalne skojarzenie według algorytmu Edmondsa-Karpa oraz solvera `glpk` dla $k = i = 2$

- $k = i = 3$

  Maksymalne skojarzenie według algorytmu Edmondsa-Karpa: 8
  Maksymalne skojarzenie według solvera `glpk`: 8

  Czas wykonywania algorytmu Edmondsa-Karpa: 3.9555e-05s
  Czas wykonywania dla solvera `glpk`: 0.0002779960632324219s

Skojarzenie wg. algorytmu	Skojarzenie wg. solvera
$1 \to 15$	$1 \to 15$
$2 \to 13$	$2 \to 13$
$3 \to 12$	$3 \to 12$
$4 \to 11$	$4 \to 11$
$5 \to 16$	$5 \to 16$
$6 \to 9$	$6 \to 9$
$7 \to 14$	$7 \to 14$
$8 \to 10$	$8 \to 10$

Tabela 4: Maksymalne skojarzenie według algorytmu Edmondsa-Karpa oraz solvera `glpk` dla $k = i = 3$

- $k = i = 7$

  Maksymalne skojarzenie według algorytmu Edmondsa-Karpa: 128
  Maksymalne skojarzenie według solvera `glpk`: 128

  Czas wykonywania algorytmu Edmondsa-Karpa: 0.00417851s
  Czas wykonywania dla solvera `glpk`: 0.5441708564758301s

- $k = i = 10$

Maksymalne skojarzenie według algorytmu Edmondsa-Karpa: 1024
Maksymalne skojarzenie według solvera `glpk`: 1024

Czas wykonywania algorytmu Edmondsa-Karpa: 0.10682s
Czas wykonywania dla solvera `glpk`: 450.0155770778656s

- Dla wartości $k = i \geq 11$ działanie solvera zostawało unicestwione.

3.3 Wnioski

Dla obu zadań solvery zwracają poprawne wartości maksymalnego przepływu, choć same szczegóły przepływu mogą się czasami różnić.

Pod względem czasu wykonywania, programy rozwiązujące problem za pomocą solvera były zdecydowanie wolniejsze niż te używające dedykowanego algorytmu - dla dużych wartości k solver pracował nawet kilka minut, podczas gdy algorytmowi Edmondsa-Karpa zajmowało to ułamki sekundy. Dla jeszcze większych wartości wyznaczenie rozwiązania za pomocą solvera było wręcz niewykonalne.

Choć korzystanie z solverów jest wygodne, to można powiedzieć, że w wielu przypadkach użycie dedykowanego algorytmu, jeśli takowy istnieje, będzie lepszym rozwiązaniem.