Sprawozdanie z listy nr 4
Algorytmy Optymalizacji Dyskretnej

Marek Świergoń (261750)

15 czerwca 2023, PWr, WIT INA

Celem tej listy było zaimplementowanie i przetestowanie różnych podejść do rozwiązywania problemów związanych z problemem maksymalnego przepływu sieci. W pierwszym zadaniu należało zastosować algorytm Edmondsa—Karpa do wyznaczania maksymalnych przepływów w hiperkostkach. Zadanie drugie dotyczyło wykorzystania tego samego algorytmu przy wyznaczaniu maksymalnego skojarzenia w nieskierowanych dwudzielnych grafach losowych. W trzecim zadaniu do rozwiązania tych samych problemów wykorzystać należało solver programowania liniowego. W tym celu programy z zadań 1 i 2 zostały wyposażone w generator modeli programowania liniowego.

Programy zostały napisane w języku Julia. Zadanie pierwsze zostało zrealizowane w pliku `zadanie1.jl`, zadanie drugie w pliku `zadanie2.jl`. Rozwiązanie zadania 3 również znajdują się w tych plikach, gdyż stanowi ich rozszerzenie o generator plików z modelami rozwiązującymi wyznaczanie maksymalnego przepływu w hiperkostce i maksymalnego skojarzenia w grafie dwudzielnym.

1 Algorytm Edmondsa—Karpa dla przepływów w skierowanej hiperkostce

Rozważany graf to hiperkostka H_k, $k \in \{1, \ldots, 16\}$. Zbiorem wierzchołków jest zbiór liczb $\{0, \ldots, 2^k - 1\}$. Wierzchołki te połączone są krawędziami w taki sposób, że każda krawędź łączy wierzchołki zaetykietowane liczbami różniącymi się o jeden bit w ich zapisie binarnym. Krawędzie te są skierowane w stronę wierzchołków mających większą wartość etykiety. Każda z krawędzi (łuków) ma określoną pojemność losowaną jednostajnie z przedziału $\{1, \ldots, 2^l\}$, $l = \max\{Z(e(i)), Z(e(j)), H(e(i)), H(e(j))\}$, gdzie $H(x)$ to liczba jedynek w zapisie binarnym liczby x, $Z(x)$ to liczba zer, zaś $e(x)$ to liczba etykietująca wierzchołek x, stanowiąca zapis binarny liczby wierzchołka.

1.1 Opis struktury przechowującej hiperkostkę

Hiperkostka H_k jest grafem rzadkim, mającym 2^k wierzchołków i $2^{k-1}n$ krawędzi. Z regularności i lematu o uściskach dłoni możemy zatem stwierdzić, że stopień każdego wierzchołka wynosi n. Oznacza to, że nieopłacalne będzie stosowanie struktur takich jak np. macierz sąsiedztwa. Lista sąsiedztwa mogłaby być pierwszym w miarę rozsądnym wyborem, ale można lepiej – do przechowania hiperkostki wystarczy tablica E rozmiaru $2^k \times k$. W polu E_{ij} znajdzie się informacja dotycząca krawędzi łączącej wierzchołek i z jego sąsiadem, który różni się na j-tym bicie. W języku Julia tablice numerowane są od 1, więc rozważane indeksy trzeba przesunąć w prawo o 1, czyli w E_{ij} znajdzie się pojemność krawędzi prowadzącej od wierzchołka $i-1$ do sąsiada różniącego się na $(j-1)$-szym bicie.

W takiej strukturze będziemy przechowywać **graf residualny** hiperkostki. W grafie tym przechowywana będzie informacja dotycząca przepływów, jakie są wyznaczane przez algorytm. Ze względu

na fakt, iż rozważana hiperkostka jest skierowana, to wstępnie połowa pól w tablicy jest wyzerowana, zaś połowa przechowuje pojemności krawędzi zgodne z wylosowanymi. Dzięki temu, że na wstępie nie istnieje żadna para krawędzi $((i,j),(j,i))$, nie potrzebujemy przechowywać oddzielnie przepływów – przepływ na danej krawędzi będzie równy wartości pojemności krawędzi odwróconej, znajdującej się w grafie residualnym.

1.2 Idea algorytmu

Algorytm na wstępie sprawdza czy w grafie residualnym znajduje się ścieżka powiększająca, czyli ścieżka $s \to \cdots \to t$ z pomocą której możliwe jest zwiększenie obecnego maksymalnego przepływu (każdy z łuków ścieżki ma pojemność niezerową). Realizuje to poprzez zastosowanie algorytmu BFS, który zapisuje w tablicy poprzedników najkrótszą ścieżkę powiększającą, jeśli taka istnieje. Gdy istnieje ścieżka powiększająca, to w tabeli znajduje się poprzednik wierzchołka t. Wtedy wyliczany jest maksymalny możliwy przepływ dla tej ścieżki, czyli minimum ze wszystkich pojemności łuków ścieżki w grafie residualnym. Po wyznaczeniu przepływu następuje zwiększenie całkowitego maksymalnego przepływu (na wstępie wynosi on 0) i aktualizacja wartości pojemności łuków w grafie residualnym (zmniejszenie pojemności łuków ścieżki o wartość dodanego przepływu i zwiększenie o tę samą wartość pojemności łuków powrotnych).

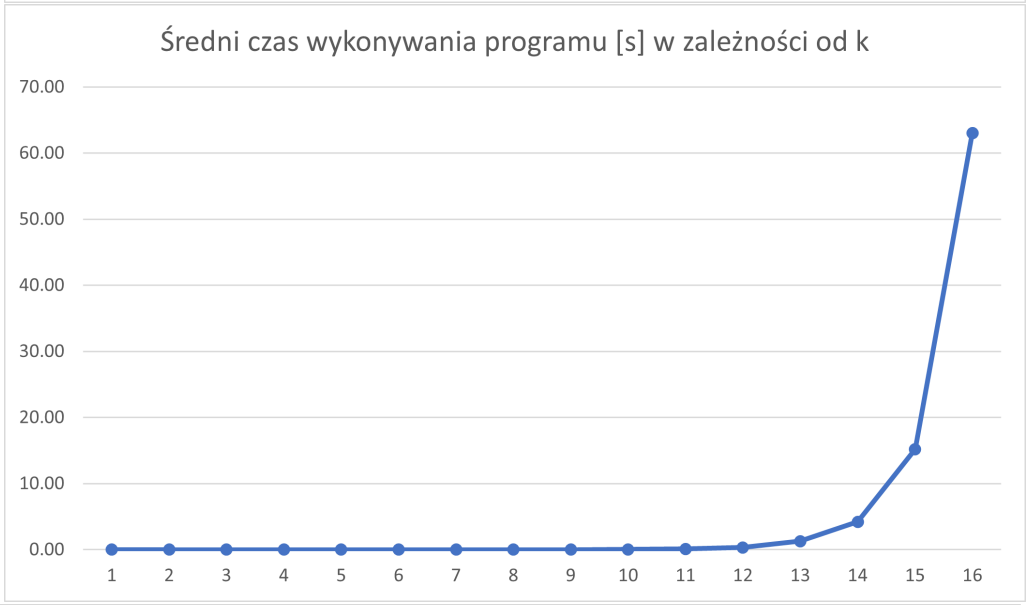
Działanie algorytmu kończy się, gdy już nie można znaleźć żadnej ścieżki powiększającej w grafie residualnym. Szczegóły przepływu, czyli ile przepływa przez każdy z łuków, możemy uzyskać patrząc na powstałe w grafie residualnym łuki odwrotne do tych w grafie pierwotnym. Wartości pojemności tych krawędzi w grafie residualnym są równe wartościom przepływów krawędzi „bazowych", ponieważ pierwotnie miały one pojemność równą 0 (nie istniały w grafie pierwotnym, więc ich pojemności były tylko zwiększane o tyle samo co przepływy na łukach „pierwotnych" hiperkostki).

Algorytm ma złożoność $O(nm^2)$. Wynika to z tego, że złożoność złożoność BFS to $O(m)$, zaś w grafie residualnym łącznie może wystąpić $O(nm)$ **krawędzi krytycznych**. Są to krawędzie znajdujące się na ścieżce powiększającej, których pojemność jest równa pojemności całej ścieżki powiększającej (czyli wyznaczają ograniczenie na przepływ po ścieżce powiększającej). Krawędzie te „znikają" po zwiększeniu przepływu i każda ścieżka powiększająca ma co najmniej jedną taką krawędź. Dowolna krawędź pierwotnego grafu może być krytyczna w grafie residualnym maksymalnie $n/2 = O(n)$ razy.

1.3 Empiryczne określenie średniego maksymalnego przepływu, czasu działania programu i średniej liczby ścieżek powiększających

Testy zostały przeprowadzone na hiperkostkach $H_k,\ k \in \{1,\ldots,16\}$. Dla każdego $k \in \{1,\ldots,14\}$ program był uruchamiany dla 1000 niezależnie wygenerowanych hiperkostek H_k (ich grafów residualnych). W przypadku $k \in \{15,16\}$ liczba hiperkostek została zmniejszona do 100 ze względu na dłuższy czas działania programu.

Poniższe wykresy prezentują wartości średniego maksymalnego przepływu, średniego czasu działania programu (w sekundach) i średniej liczby ścieżek powiększających wyznaczonych przez BFS podczas działania algorytmu.



Rysunek 1: Wykresy przedstawiające wyniki testów.

Każda z analizowanych wartości rośnie szybko przy zwiększaniu k. Czas działania programu zależy wykładniczo od k, ponieważ złożoność obliczeniowa algorytmu Edmondsa—Karpa wynosi $O(nm^2)$, zaś $n = 2^k$, $m = 2^{k-1}n = 2^{2k-1}$. Wielkość średniego maksymalnego przepływu rośnie szybciej niż średnia liczba ścieżek powiększających, ponieważ maksymalny przepływ zależy od liczby ścieżek powiększających (więcej ścieżek to więcej możliwości powiększenia przepływu) ale też od wartości pojemności krawędzi, których wartość oczekiwana rośnie w zależności liniowej od k (przy wzroście k o 1 rośnie również o 1 długość reprezentacji binarnej etykiety wierzchołka). Liczba ścieżek powiększających też zależy wykładniczo od k. Wynika to między innymi z tego, że w hiperkostce istnieje n wierzchołkowo rozłącznych ścieżek z s do t (dla tej i tylko tej pary wierzchołków można zastosować rozumowanie z ogólnej, nieskierowanej hiperkostki z użyciem twierdzenia Balinskiego i twierdzenia Mengera). Liczba ta może stanowić ograniczenie dolne na liczbę ścieżek powiększających, gdyż każda ze ścieżek ma pojemność o wylosowanej wartości większej od zera. Wartość ta rośnie jednak najwolniej spośród analizowanych.

Można zatem stwierdzić, że wyznaczanie maksymalnego przepływu jest problemem trudnym dla dużych wymiarów hiperkostek. Choć w zadaniu trzecim będzie można zauważyć, że algorytm Edmondsa—Karpa działa sprawniej od solvera z modelem LP, to i tak jest to proces czasochłonny dla dużego k.

2 Algorytm Edmondsa—Karpa przy wyznaczaniu maksymalnego skojarzenia

Celem zadania jest wyznaczenie maksymalnego skojarzenia dla nieskierowanego grafu dwudzielnego mającego dwa rozłączne podzbiory wierzchołków V_1 oraz V_2 o mocy 2^k. Każdy wierzchołek z V_1 ma i różnych sąsiadów z V_2 wybranych niezależnie i jednostajnie losowo. Dodatkowo program zwraca swój czas działania w sekundach i (opcjonalnie) szczegóły skojarzenia.

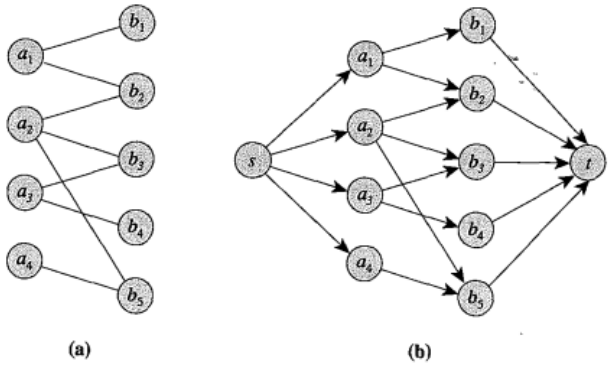
2.1 Idea redukcji problemu wyznaczenia maksymalnego skojarzenia do problemu wyznaczenia maksymalnego przepływu

Krawędzie pierwotnego grafu zamieniono na krawędzie skierowane w kierunku wierzchołków z V_2. Przed wierzchołkami z V_1 dodany został wierzchołek, który połączono krawędziami skierowanymi z wszystkimi wierzchołkami z V_1. Krawędzie te wychodzą z dodanego wierzchołka i mają pojemności równe 1. Wierzchołek ten będzie stanowić źródło przepływu.

Za wierzchołkami z V_2 został dodany kolejny wierzchołek, będący ujściem przepływu. Wierzchołek ten został połączony z wszystkimi wierzchołkami z V_2 za pomocą krawędzi wchodzących do tego wierzchołka i mających pojemność 1. Wszystkie krawędzie skierowane pomiędzy wierzchołkami z V_1 a wierzchołkami z V_2 również mają pojemność równą 1. Powstały graf razem z krawędziami odwrotnymi o zerowych pojemnościach stanowi graf residualny. Rozwiązanie dla tego grafu problemu maksymalnego przepływu wyznaczy maksymalne skojarzenie. Wynika to z faktu, iż do każdego z wierzchołków z V_1 może trafić tylko jedna jednostka przepływu (pojemności krawędzi ze źródła do wierzchołków V_1). Każda krawędź łącząca wierzchołek z V_1 z wierzchołkiem z V_2 może mieć przypływ wynoszący maksymalnie 1 (wzięta tylko raz), ze względu na jej ograniczenie pojemności. Z każdego z wierzchołka z V_2 tylko jedna jednostka przepływu może zostać przetransportowana do ujścia ze względu na ograniczenie pojemności krawędzi łączących z ujściem. W związku z tym wybrane krawędzie są wierzchołkowo rozłączne i dają maksymalne skojarzenie.

Dzięki przekształceniu problemu wyznaczenia maksymalnego skojarzenia na problem wyznaczenia maksymalnego przepływu możemy wykorzystać algorytm Edmondsa—Karpa do uzyskania rozwiązania. Należy dokonać jedynie drobnych zmian w implementacji aby dostosować go do nowej struktury grafu residualnego.

Poniższy rysunek obrazuje opisane wcześniej przekształcenia.



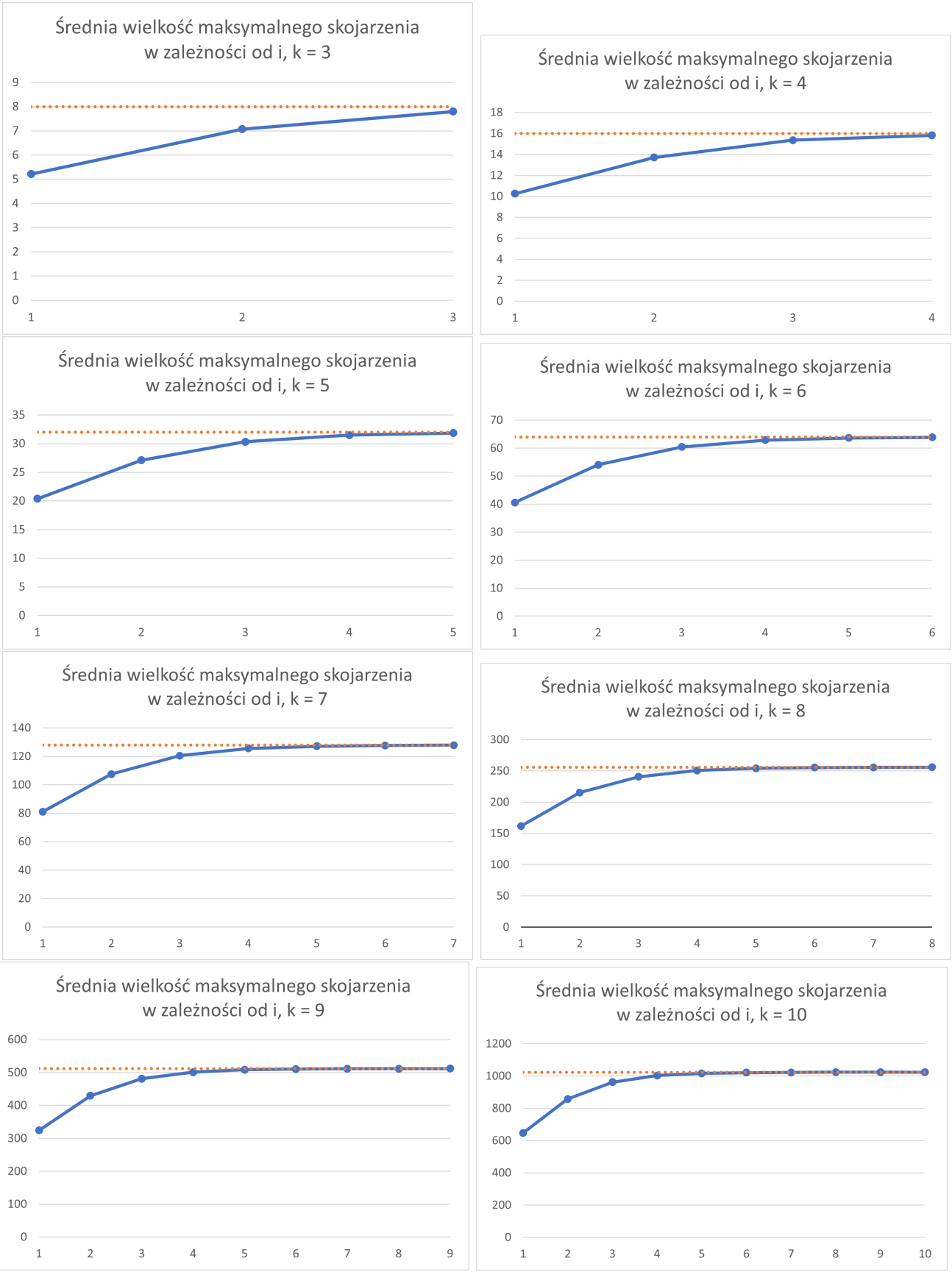
Rysunek 2: Przedstawienie graficzne redukcji problemu maksymalnego skojarzenia do problemu maksymalnego przepływu. Rysunek z książki Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications.* Prentice-Hall, Inc., USA, 1993.

2.2 Opis struktury przechowującej graf residualny

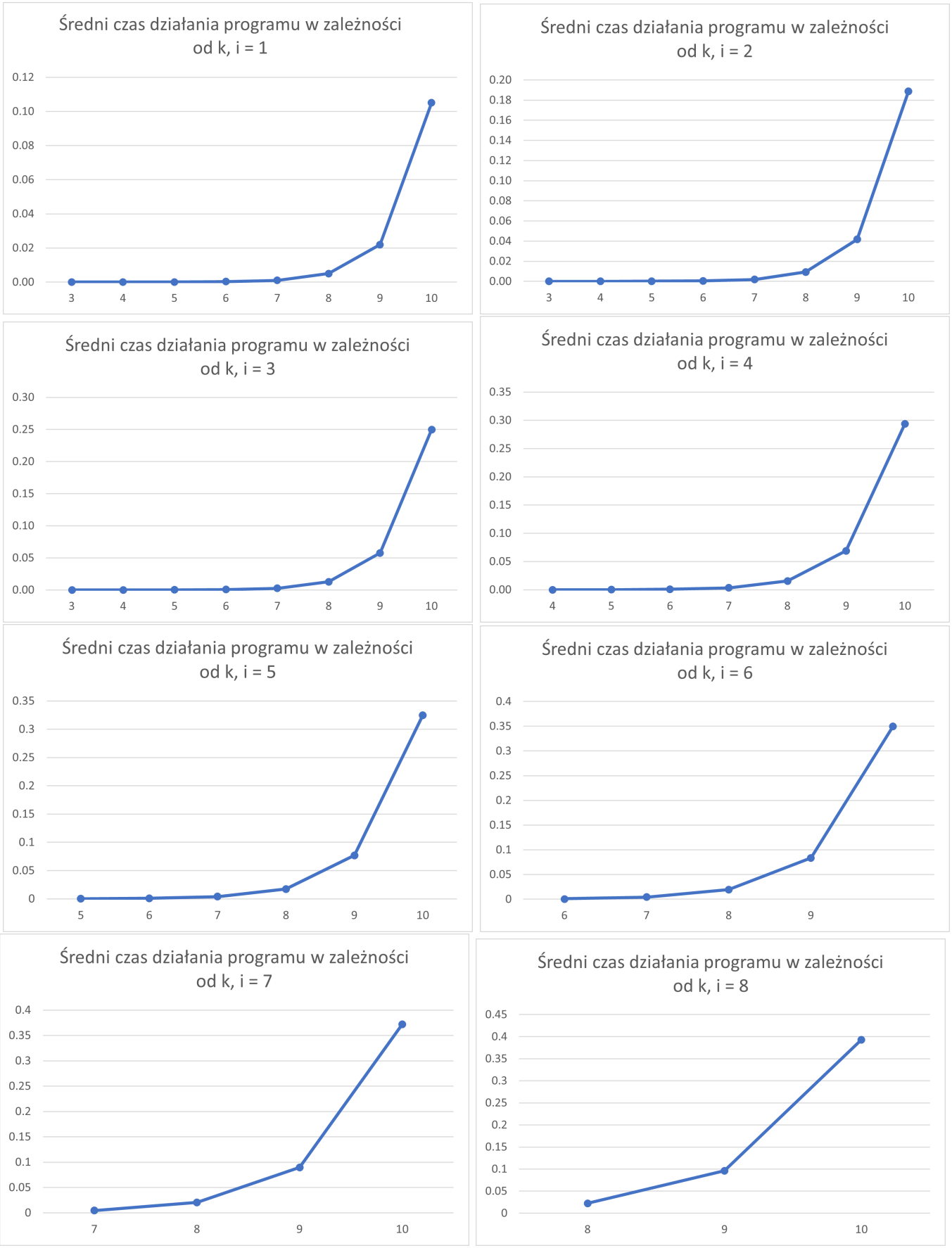
W strukturze przechowującej graf residualny znajduje się liczba wierzchołków, liczba k oraz wartości pojemności łuków grafu (wynoszące pierwotnie 1 dla krawędzi skierowanych w stronę ujścia i 0 dla krawędzi odwrotnych). Wartości pojemności łuków grafu residualnego przechowywane są w tablicy słowników. Indeksami tablicy są etykiety wierzchołków, z których wychodzą dane krawędzie. Kluczami w słowniku są z kolei etykiety wierzchołków, do których wchodzą te krawędzie. Jako wartość w słowniku przechowywana jest pojemność krawędzi. Graf residualny na wstępie jest grafem z krawędziami o niezerowych pojemnosciach skierowanymi w stronę ujścia, zatem nie jest konieczne stosowanie pól przechowujących przepływ – tak jak w zadaniu 1 będzie można go odczytać z wartości pojemności krawędzi odwrotnych.

2.3 Testy średniego maksymalnego skojarzenia i średniego czasu działania programu

Testy zostały przeprowadzone na grafach skierowanych o $k \in \{1, \ldots, 10\}$ oraz $i \in \{1, \ldots, k\}$. Poniżej zaprezentowane zostały wykresy przedstawiające zależność średniego maksymalnego skojarzenia od stopnia i oraz zależność średniego czasu działania programu od wielkości k.



Rysunek 3: Wykresy przedstawiające wartości średniego maksymalnego skojarzenia w zależności od i dla poszczególnych k.



Rysunek 4: Wykresy przedstawiające średni czas działania programu (w sekundach) w zależności od k dla poszczególnych $i \in \{1, \ldots, 8\}$.

Zauważyć można, że średnia wielkość maksymalnego skojarzenia bardzo szybko zbiega do liczby 2^k. Liczba ta stanowi maksymalne możliwe do uzyskania skojarzenie – w jego krawędziach zawarte są

wtedy wszystkie możliwe wierzchołki z rozważanych rozłącznych zbiorów wierzchołków V_1 i V_2 (o mocy właśnie 2^k każdy). W przypadku $k = i$ bardzo często uzyskane maksymalne skojarzenie jest równe 2^k. Nawet przy $i = 1$ średnia wielkość maksymalnego skojarzenia jest stosunkowo duża; w średnim przypadku maksymalne skojarzenie jest większe niż połowa górnego ograniczenia na maksymalne skojarzenie dla danej mocy zbiorów rozłącznych.

Tak jak w zadaniu 1, czas działania programu jest wykładniczy w stosunku do k. Wynika to z faktu, iż całkowita moc zbioru wierzchołków grafu residualnego wynosi $2^{k+1} + 2$. Porównując wykresy dla różnych i zauważyć można również, że średni czas działania programu rośnie wraz ze wzrostem i. Spowodowane jest to wzrostem liczby krawędzi, która silnie wpływa na czas działania algorytmu Edmondsa—Karpa. Liczba krawędzi w rozważanym grafie residualnym jest równa $(i + 2) \cdot 2^k$.

3 Modele LP dla zadań 1 i 2

Celem zadania jest stworzenie generatora modeli LP dla poprzednich dwóch zadań. Wygenerowane modele zapisywane są w osobnym pliku i można je uruchomić bezpośrednio z użyciem języka Julia i pakietu JuMP oraz solvera HiGHs.

3.1 Opis modelu LP oraz generatorów

Na wstępie do pliku zapisana zostaje informacja o wykorzystaniu JuMP i HiGHs. Następnie deklarowana jest dwuwymiarowa tablica G rozmiaru $2^k \times 2^k$. Znajdują się w niej pojemności krawędzi hiperkostki. Większość pól jest wyzerowana, ponieważ hiperkostka jest grafem rzadkim. Niestety taki narzut pamięci jest niezbędny do poprawnego zamodelowania problemu. Następnie do zmiennej n zapisana zostaje liczba rzędów tablicy G. Tworzymy macierz zmiennych, w której zmienna f_{ij} oznacza przepływ na krawędzi (i, j). Ze względu na to, że większość pól G jest wyzerowana, to można wymusić wartości zmiennych odpowiadających tym polom na równe 0. Dokonuje się tego z użyciem polecenia `fix(f[i,j], 0.0; force = true)`. Następnie zapisane zostają ograniczenia i funkcja celu:

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

subject to:

$$f_{ij} \leqslant u_{ij}$$

(u_{ij} to ograniczenie górne zapisane w G_{ij})

$$\sum_{j=1}^{n} f_{ij} = \sum_{j=1}^{n} f_{ji}, \quad i \in \{2, \ldots, n-1\}$$

$$f_{ij} \geqslant 0$$

W modelu nie jest konieczne ograniczenie na całkowitoliczbowość – bez niego wyniki też będą poprawne, całkowitoliczbowość wymuszą pojemności $u_{i,j}$ (które są równe 0 lub 1).

Po zapisaniu modelu wpisane zostały polecenia uruchomienia solvera i wyświetlania czasu oraz szczegółów przepływu.

Ten sam model będzie wykorzystany w generatorze w pliku do zadania 2. Tablica G ma wymiary $2^k + 2 \times 2^k + 2$ i ponownie przechowuje wartości pojemności łuków. Tak samo jak poprzednio, zerowe pojemności są wymuszane na zmiennych, dla których odpowiadająca wartość z G jest równa 0. Dalsze działanie generatora jest analogiczne do poprzedniego – jedyna istotna różnica to drukowanie szczegółów skojarzenia, czyli tylko krawędzi, dla których $f_{ij} = 1$.

3.2 Porównanie wyników i szybkości działania do rezultatów dedykowanego algorytmu

W przypadku obu zadań czas działania solvera jest znacząco dłuższy niż czas działania programu z algorytmem Edmondsa—Karpa. W pierwszym zadaniu uzyskanie rozwiązania dla $k > 12$ jest praktycznie niemożliwe. W zadaniu drugim średni czas działania programu dla $k = i = 10$ wynosił niewiele ponad pół sekundy. Solver nie zawsze był w stanie wyznaczyć rozwiązanie, czasami brakowało pamięci. Czas działania solvera był zwykle o parę rzędów wielkości dłuższy.

Choć obie metody zwracają dla tego samego grafu tą samą wartość maksymalnego przepływu (lub skojarzenia), to często różnią się szczegółami przepływu. Solver „dobiera" inne wartości przepływów dla poszczególnych krawędzi.