Wydział Matematyki
i Nauk Informacyjnych

POLITECHNIKA WARSZAWSKA

Teoria algorytmów i obliczeń

Projekt zaliczeniowy

Piotr Jacak
Jakub Kindracki
Wiktor Kobielski
Ernest Mołczan

Koordynator: prof. dr hab. inż. Władysław Homenda

Semestr zimowy 2025/2026

Spis treści

1 Wstęp

Niniejsza praca stanowi sprawozdanie z projektu zrealizowanego w ramach przedmiotu **Teoria algorytmów i obliczeń**. Przedmiotem badań są algorytmy operujące na multigrafach, ze szczególnym uwzględnieniem problematyki izomorfizmu podgrafów oraz minimalnych rozszerzeń grafów.

Głównym celem projektu jest opracowanie, analiza teoretyczna oraz implementacja algorytmów rozwiązujących dwa ściśle powiązane problemy. Pierwszym z nich jest weryfikacja, czy dany multigraf H jest izomorficzny z podgrafem multigrafu G. Drugim, kluczowym zagadnieniem, jest wyznaczenie *minimalnego rozszerzenia* multigrafu G do postaci G', która zawiera co najmniej jeden podgraf izomorficzny z H.

Realizacja powyższych celów wymagała formalnego zdefiniowania oraz uzasadnienia kilku fundamentalnych pojęć. W pracy zaproponowano autorskie lub bazujące na literaturze definicje:

- *rozmiaru multigrafu*,
- *metryki* w zbiorze multigrafów,
- *minimalnego rozszerzenia* multigrafu.

Pojęcia te stanowią podstawę do dalszej analizy algorytmicznej oraz oceny kosztu operacji.

W ramach pracy przeprowadzono analizę złożoności obliczeniowej opracowanych algorytmów. Zgodnie z założeniami projektu, w przypadku gdy algorytmy dokładne charakteryzują się złożonością wykładniczą, przedstawiono również propozycje algorytmów aproksymacyjnych o złożoności wielomianowej.

Niniejszy raport, oprócz formalnych definicji i analizy algorytmów, zawiera także opis przeprowadzonych testów obliczeniowych, dokumentację techniczną implementacji oraz wnioski końcowe.

2 Definicje pojęć

Definicja 1 (Graf)**.** Grafem nazywamy parę $G = (V, E)$, gdzie V jest zbiorem wierzchołków, a $E \subseteq V \times V = \{(u, v) : u, v \in V \wedge u \neq v\}$ jest zbiorem krawędzi. Dla każdej pary wierzchołków $u, v \in V$ istnieje co najwyżej jedna krawędź łącząca wierzchołki u i v.

Definicja 2 (Multigraf)**.** Multigrafem nazywamy graf, w którym pomiędzy dowolnymi dwoma różnymi wierzchołkami $u, v \in V$ może istnieć więcej niż jedna krawędź.

Definicja 3 (Izomorfizm grafów)**.** Dwa grafy $G_1 = (V_1, E_1)$ i $G_2 = (V_2, E_2)$ są izomorficzne, wtedy i tylko wtedy, gdy istnieje bijekcja $f : V_1 \to V_2$, taka że dla każdej krawędzi $(u, v) \in E_1$ zachodzi $(f(u), f(v)) \in E_2$. Definicja ta jest analogiczna dla multigrafów.

Definicja 4 (Podgraf)**.** Graf $H = (V_H, E_H)$ nazywamy podgrafem grafu $G = (V_G, E_G)$, wtedy i tylko wtedy, gdy $V_H \subseteq V_G$ oraz $E_H \subseteq E_G$. Definicja ta jest analogiczna dla multigrafów.

Definicja 5 (Graf atrybutowy)**.** Graf $G = (V, E, f)$ nazywamy grafem atrybutowym, gdzie V jest zbiorem wierzchołków, a $E \subseteq V \times V = \{(u, v) : u, v \in V \wedge u \neq v\}$ jest zbiorem krawędzi. Dla każdej pary wierzchołków $u, v \in V$ istnieje co najwyżej jedna krawędź łącząca wierzchołki u i v. $f : E \to \Sigma_E$ jest funkcją, przypisującą etykiety wszystkim krawędziom w grafie G.

Definicja 6 (Macierz sąsiedztwa)**.** Macierzą sąsiedztwa multigrafu $G = (V, E)$ nazywamy macierz A, której pole $A_{uv} = k$, wtedy i tylko wtedy, gdy istnieje k krawędzi $(u, v) \in E$. W przypadku gdy nie istnieje żadna krawędź pomiędzy wierzchołkami u i v, to $A_{uv} = 0$. Dla zwykłych grafów, macierz sąsiedztwa jest macierzą binarną.

3 Rozmiar multigrafu

4 Odległość multigrafów

5 Minimalne rozszerzenie multigrafu

5.1 Algorytm dokładny dla problemu izomorfizmu podgrafu

Mając dane dwa grafy G i H, chcemy znaleźć podgrafy G izomorficzne do H. Do rozwiązania tego problemu posłuży nam algorytm, który wykorzystuje procedurę Backtrackingu do sprawdzania struktury grafów.
Przed przejściem do algorytmu, zdefiniujmy sobie struktury przydatne nam do implementacji. Niech $n_G = |V(G)|$ - ilość wierzchołków w grafie G oraz $n_H = |V(H)|$ - ilość wierzchołków w Grafie H.

Opis algorytmu:

1. Inicjalizacja macierzy sąsiedztwa grafów G i H odpowiednio $S_G \in \mathbb{N}^{n_G \times n_G}$ i $S_H \in \mathbb{N}^{n_H \times n_H}$. Wartość $S[i,j]$, to ilość krawędzi pomiędzy i-tym, a j-tym wierzchołkiem dla danego grafu.

2. Inicjalizacja kandydatów - Zdefiniujmy sobie listę $mozliwe\_dopasowania$, $len(mozliwe\_dopasowania) = n_H$, gdzie pod i-tym indeksem, będziemy mieli listę możliwych dopasowań dla wierzchołka $i \in V(H)$.
   Algorytm Ullmana dla grafów prostych zakłada inicjalizację:
   $u \in V(G), u \in mozliwe\_dopasowania[i] \iff deg_G(u) \geq deg_H(i)$
   Jest ona działającą inicjalizacją dla multigrafów, jednak w celach optymalizacji algorytmu, możemy zmienić tę inicjalizację tak, aby zmniejszyć liczbę potencjalnych dopasowań, a co za tym idzie zmniejszyć liczbę gałęzi, które będzie musiał przejść algorytm. Możemy zauważyć, że w macierzach sąsiedztwa na głównej przekątnej pod indeksami $[i,i]$ znajduje się liczba pętli danego wierzchołka, zatem naszym warunkiem będzie także $S_G[i,i] \geq S_H[i,i]$. Biorąc to wszystko razem, otrzymujemy
   $u \in mozliwe\_dopasowania[i] \iff (deg_G(u) \geq deg_H(i)) \wedge (S_G[i,i] \geq S_H[i,i]))$

3. Dla każdej krawędzi, która istnieje między już dopasowanymi wierzchołkami z H, sprawdź czy istnieje krawędź między ich dopasowaniami z G i czy ilość krawędzi między dopasowaniami jest większa lub równa niż ilość krawędzi mię-

dzy wierzchołkami. Można to osiągnąć przez przejrzenie wszystkich par już dopasowanych wierzchołków i krawędzi między nimi. Jeśli nie, zwróć False

4. Sprawdź, czy wszystkie wierzchołki nie zostały już dopasowane. Jeśli tak, zwróć True.

5. Dla każdego wierzchołka $v \in mozliwe\_dopasowania[i]$, jeśli v *not in* $dopasowania$, przypisz $dopasowania[i] = v$ oraz wywołaj funkcję ponownie dla następnego wierzchołka $\in V(H)$ z przekazaną kopią. W przypadku wyniku True z tej funkcji, zwróć True, w przypadku False, $dopasowania[i] = null$ i przejdź do następnego kroku tej pętli.

6. W przypadku niedopasowania po wszystkich iteracjach pętli, zwróć False.

5.1.1 Dowód poprawności

Najpierw zbadajmy, czy algorytm dobrze inicjalizuje *mozliwe_dopasowania*. W tym celu rozbijmy wszystkie 3 warunki. Pierwszy warunek mówi o tym, że potencjalne dopasowanie v dla wierzchołka u, musi mieć stopień co najmniej równy stopniowi wierzchołka u. Gdyby tak nie było, w grafie G nie istniałaby co najmniej jedna krawędź wychodząca z v, która istniałaby w H i wychodziłaby z u, zatem v nie mogłoby być dopasowaniem dla u. Drugi warunek mówi o tym, że liczba pętli dla v musi być co najmniej równa liczbie pętli dla u. Idea jest taka sama jak warunku pierwszego, gdyby warunek nie był spełniony, nie istniałaby co najmniej jedna pętla da danego wierzchołka, a co za tym idzie, nie mógłby on być dopasowaniem dla u.

Dalej w algorytmie, przechodzimy po kolei po wierzchołkach z H. Najpierw sprawdzamy, czy struktura się zgadza dla tych wierzchołków, do których znaleźliśmy już dopasowania. Jeśli choć 1 krawędź istniejąca w H pomiędzy dwoma wierzchołkami nie będzie istnieć między ich dopasowaniami w G, algorytm wychodzi z tej ścieżki dopasowań i szuka innych, zatem działa poprawnie.

Następnie sprawdzamy wszystkie z możliwych dopasowań dla danego wierzchołka, zatem sprawdzając tak wszystkie wierzchołki, mamy pewność, że przejdziemy po wszystkich możliwych permutacjach.

5.1.2 Złożoność obliczeniowa

Zauważmy, że inicjalizacja *mozliwe_dopasowania* w taki sposób, że dla każdego wierzchołka $u \in V(H)$ możliwym dopasowaniem są wszystkie $v \in V(G)$, to algorytm przejdzie po wszystkich poddrzewach, zatem w przypadku pesymistycznym do 1

wierzchołka wykona n_G potencjalnych dopasowań, do drugiego $n_G - 1$, ..., a do n_H-tego, $(n_G - n_H + 1)$ dopasowań. Zatem mamy

$$\underbrace{(n_G)(n_G-1)\dots(n_G-n_H+1)}_{n_H \text{ razy}} \leq n_G^{n_H}$$

W każdej takiej pętli wykonujemy sprawdzenie, czy struktura grafu się zgadza, (krok 4). Zauważmy, że wykonamy tam i^2 porównań, gdzie i to indeks aktualnie obliczanego wierzchołka. Wiemy że $i < n_H$, zatem możemy ograniczyć tę operację: $i^2 < n_H^2$. W sumie możemy stwierdzić, że złożoność tego algorytmu wyniesie $O(n_G^{n_H} n_H^2)$

5.2 Algorytm aproksymacyjny do problemu izomorfizmu podgrafu

Do problemu można zastosować algorytm LeRP (Length-R Paths) opracowany przez Freda W DePiero oraz Davida Krouta [1]. Autorzy bezpośrednio stwierdzają, że nie zajmują się multigrafami. Zaznaczają natomiast, że multigrafy nie są ograniczeniem, ponieważ można stworzyć reprezentację multigrafu opisującą istnienie równoległych krawędzi za pomocą schematu kolorowania krawędzi. Zatem kompletny algorytm aproksymacyjny będzie składał się z dwóch kroków:

1. Redukcja: Transfomacja multigrafów wejściowych G_1 oraz G_2 na grafy atrybutowe G_1' oraz G_2'.

2. Aproksymacja: Wykorzystanie algorytmu LeRP (który obsługuje grafy atrybutowe) do stwierdzenia, czy graf G_1 jest izomorficzny z pewnym podgrafem P grafu G_2.

5.2.1 Transformacja multigrafu

Niech G_1 i G_2 będą multigrafami. Odpowiadające im grafy atrybutowe G_1' i G_2' konstruowane są w następujący sposób:

- Zbiory wierzchołków pozostają bez zmian ($V_1' = V_1, V_2' = V_2$).

- Dla każdej pary wierzchołków (u, v) w G_1 (i analogicznie w G_2): Jeśli między u a v w G_1 istnieje k równoległych krawędzi, to w G_1' tworzona jest pojedyncza krawędź (u, v) z atrybutem $k \in \mathbb{N}^+$.

5.2.2 Aproksymacja algorytmem LeRP

Fundamentalna zasada LeRP różni się od algorytmów backtrackingu (przykładowo algorytm Ullmanna). Zamiast próbować dopasować pełną strukturę sąsiedztwa krok po kroku, LeRP opiera się na założeniu, że o podobieństwie strukturalnym dwóch wierzchołków można wnioskować na podstawie porównania liczby ścieżek (*sygnatur*) o różnej długości (r - parametr algorytmu) w ich sąsiedztwie. Działanie algorytmu LeRP można podzielić na kilka etapów, realizowanych na grafach atrybutowych G'_1 i G'_2.

1. Obliczanie liczby ścieżek (pre-processing)
   Dla obu grafów G'_1 i G'_2 obliczane są potęgi ich macierzy sąsiedztwa, odpowiednio A^r i B^r aż do maksymalnej długości R. Wartość A^r_{ij} w macierzy A^r reprezentuje liczbę ścieżek o długości dokładnie r z wierzchołka i do wierzchołka j. W kontekście omówionej transformacji, macierz A jest macierzą, gdzie A_{ij} przechowuje atrybut k - liczbę równoległych krawędzi wcześniejszego multigrafu G_1. Obliczenie A^r polega na r-krotnym mnożeniu macierzy.

2. Porównanie strukturalne
   Dla każdej pary wierzchołków $g_{1i} \in G'_1$ i $g_{2k} \in G'_2$ algorytm porównuje ich struktury: sprawdza, do jakiej długości ścieżki ($r = 1, ..., R$) grafy wyglądają tak samo. Im dłużej są podobne, tym wyższy wynik podobieństwa r_{max}.

3. Mapping
   Algorytm zaczyna od pojedynczego ziarna mapowania - wstępnie przypisuje wierzchołek g_{1i} do wierzchołka g_{2k}. Następnie iteracyjnie dodaje do tego mapowania sąsiadów już zamapowanych wierzchołków, wybierając te, które maksymalizują wskaźnik podobieństwa i są spójne ze znalezionymi wcześniej.

LeRP opiera się na założeniu, że dopasowanie sygnatur ścieżek (taka sama liczba ścieżek o długości $1, 2, ..., R$ jest wystarczającym dowodem podobieństwa, aby utworzyć mapowanie.

Złożoność pesymistyczna algorytmu LeRP jest wielomianowa i wynosi $O(N^3 \cdot D^2 \cdot R)$, gdzie:

- N to liczba wierzchołków w grafach (zakładając, że oba grafy mają rozmiar rzędu N).
- D to średni stopień wierzchołków w grafach.

- R to maksymalna długość ścieżki brana pod uwagę.

Uzasadnienie złożoności: porównanie par wierzchołków wymaga N^2 porównań. Każdy wierzchołek ma średnio D sąsiadów, więc daje to D^2 dodatkowych operacji. Analiza różnych długości ścieżek to czynnik R. Podczas budowania dopasowania, algorytm jeszcze raz przechodzi po wszystkich kandydatach, a więc N jest kolejnym czynnikiem.

W kontekście tego algorytmu aproksymacyjnego nie rozważano formalnego dowodu poprawności. Dostarczono empiryczną gwarancję jakości - algorytm testowano na dużych zbiorach danych, wykazując, że algorytm konsekwentnie zwraca wyniki bliskie optimum w praktyce.

6 Testy

7 Podsumowanie

8 Bibliografia

Literatura

[1] Fred DePiero and David Krout. An algorithm using length-r paths to approximate subgraph isomorphism. *Pattern Recognition Letters*, 24(1):33–46, 2003.