Poszukiwanie największej kliki w grafie

Anna Stępień
Adam Stelmaszczyk

Spis treści

1 Zadanie

Kliką grafu nazywamy podgraf, w którym każde dwa wierzchołki są ze sobą połączone. Największą kliką nazywamy klikę o największej liczbie wierzchołków. Celem zadania jest implementacja wybranego algorytmu znajdującego największa klikę w grafie oraz analiza otrzymanych wyników.

2 Założenia

Wejściem dla algorytmu jest graf nieskierowany dany macierzą o n wierszach i n kolumnach:

$$\begin{matrix} q_{0,0} & q_{1,0} & \cdots & q_{n-1,0} \\ q_{0,1} & q_{1,1} & \cdots & q_{n-1,1} \\ \vdots & \vdots & \ddots & \vdots \\ q_{0,n-1} & q_{1,n-1} & \cdots & q_{n-1,n-1} \end{matrix}, q_{i,j} \in \{0,1\}, 0 \leqslant i,j < n$$

$q_{i,j}$ równe 0 oznacza, że wierzchołki i oraz j nie są połączone krawędzią. W przeciwnym razie, wierzchołki są połączone.

Macierz jest dana w pliku tekstowym, w którym kolejne $q_{i,j}$ w wierszu j są oddzielone co najmniej jednym znakiem białym. Przez znak biały rozumiemy spację lub tabulator. $q_{i,j}$ różne od 0 będą traktowane jak 1.

Wyjściem jest niepusty zbiór numerów wierzchołków, które tworzą największą klikę w podanym grafie. Wierzchołki numerujemy od 0 do $n-1$. W grafie może istnieć więcej niż jedna największa klika. W takim przypadku algorytm zwróci dowolną z nich.
Realizowana aplikacja będzie pracowała w trybie konsolowym, z ewentualną możliwością specyfikacji dodatkowych parametrów. W projekcie zostanie wykorzystany algorytm Brona–Kerboscha [1].

3 Algorytm

Algorytm Brona–Kerboscha jest rekurencyjnym algorytmem z nawrotami, który umożliwia poszukiwanie największych klik w zadanym grafie niezorientowanym.
Domyślnie algorytm zwraca wszystkie maksymalne kliki, tj. podzbiory wierzchołków grafu wejściowego do których nie można dodać już ani jednego wierzchołka. W algorytmie wprowadzona zostanie zmiana, dzięki której zwracana

będzie największa ze znalezionych klik, charakteryzująca się największą liczbą wierzchołków.

3.1 Pseudokod

Algorithm 1 Algorytm Brona–Kerboscha (wersja podstawowa)

```
1: function BRON_KERBOSCH(compsub, candidates, not)
2:     if candidates = ∅ and not = ∅ then
3:         return compsub                                  ▷ Maksymalna klika
4:     else
5:         for each v in candidates do
6:             candidates ← candidates \ {v}
7:             new_compsub ← compsub ∪ {v}
8:             new_candidates ← candidates ∩ neighbors(v)
9:             new_not ← not ∩ neighbors(v)
10:            BRON_KERBOSCH(new_compsub, new_candidates, new_not)
11:            compsub ← compsub ∪ {v}
12:        end for
13:    end if
14: end function
```

Algorithm 2 Algorytm Brona–Kerboscha (wersja rozszerzona)

```
1: compsub ← ∅
2: candidates ← V(G)
3: not ← ∅
4: maximum_clique ← ∅
5: function BRON_KERBOSCH(candidates, not)
6:     if candidates = ∅ and not = ∅ then
7:         if size(maximum_clique) < size(compsub) then
8:             maximum_clique ← compsub
9:         end if
10:        return maximum_clique                    ▷ Maksymalna klika
11:    else
12:        pivot ← maxdeg
13:        candidates_to_check ← candidates \ neighbors(pivot)
14:        for each v in candidates_to_check do
15:            compsub ← compsub ∪ {v}
16:            candidates ← candidates \ {v}
17:            new_candidates ← candidates ∩ neighbors(v)
18:            new_not ← not ∩ neighbors(v)
19:            BRON_KERBOSCH(new_candidates, new_not)
20:            compsub ← compsub \ {v}
21:            not ← not ∪ {v}
22:        end for
23:    end if
24: end function
```

3.2 Opis działania

Istotą działania przedstawionego algorytmu jest utrzymywanie trzech rozłącznych zbiorów: *compsub*, *candidates* oraz *not*.
Algorytm Brona–Kerboscha znajduje maksymalne kliki składające się ze wszystkich wierzchołków należących do zbioru *compsub*, niektórych należących do zbioru *candidates*, i z żadnego, który należy do zbioru *not*.
Poniżej przedstawiona została charakterystyka każdego ze zbiorów wykorzystywanych przez algorytm:

- *compsub*

  do zbioru należą wszystkie wierzchołki grafu, które tworzą powstającą klikę.

- *candidates*

do zbioru należą wierzchołki grafu, które mogą posłużyć do rozszerzenia zbioru *compsub*.

- *not*

  do zbioru należą te wierzchołki, które były już wcześniej wykorzystane do rozszerzenia zbioru *compsub*.

Należy zauważyć, iż wszystkie wierzchołki, które są połączone z każdym wierzchołkiem należącym do zbioru *compsub* znajdują się albo w zbiorze *candidates* albo *not*.

Zmodyfikowana wersja algorytmu Brona–Kerboscha wprowadza pojęcie wierzchołka zwrotnego (dalej oznaczanego *pivot*), który wybierany jest ze zbioru $candidates \cup not$ jako wierzchołek o największym stopniu.

W każdym rekurencyjnym wywołaniu algorytmu rozważane są wierzchołki należące do zbioru *candidates*. Jeśli zbiory *candidates* i *not* są puste, sprawdzane jest czy znaleziona klika (oparta na wierzchołkach ze zbioru *compsub*) jest większa od największej dotychczas znalezionej kliki. Jeśli tak, to znaleziona klika staje się maksymalną i jest zwracana przez algorytm, w przeciwnym wypadku zwracana jest największa dotychczas znaleziona klika.
W przypadku, gdy zbiory *candidates* i *not* nie są puste, dla każdego wierzchołka ze zbioru $candidates \setminus neighbors(pivot)$ następuje rekurencyjne wywołanie algorytmu, w którym bieżący wierzchołek v dodawany jest do zbioru *compsub* i usuwany ze zbioru *candidates*, a w zbiorach *candidates* i *not* pozostawiane są tylko te wierzchołki grafu, które są sąsiadami wierzchołka v. Następnie, wierzchołek v jest dodawany do zbioru *not* jako już wykorzystany do rozszerzenia kliki oraz usuwany ze zbioru *compsub*.

4 Struktury danych

5 Testy

5.1 Badanie poprawności zwracanych wyników

5.2 Badanie czasu wykonania dla różnych typów grafów

Referencje

[1] Coen Bron, Joep Kerbosch, *Algorithm 457: finding all cliques of an undirected graph*, Communications of the ACM, 16(9): 575–577, 1973.