Raport

Permutacje macierzy i ich wpływ na kompresje

Szymon Twardosz, Dominik Jeżów

4 stycznia 2024

1 Środowisko

Do wykonania ćwiczenia wykorzystaliśmy język python 3.11 wraz z następujęcymi bibliotekami numpy, matplotlib, sklearn, heapq, scipy

2 Temat zadania

Celem zadania było wygenerowanie macierzy o rozmiarze 2^{3k}, która opisuje topologię trójwymiarowej siatki. Po uzyskaniu macierzy, zadaniem było porównanie wzorca rzadkości tej macierzy z wzorcami powstałymi poprzez permutację źródłowej macierzy.
Zaimplementowane przez nas metody permutacji macierzy to:

- Minimum degree
- Culthill-McKee
- Reversed Culthill-McKee

Po przeprowadzeniu permutacji, konieczne było zbadanie efektów kompresji dla każdej z permutacji. macierzy.

3 Pseudokod

Kod 1: Minimum degree

```
M # macierz nXn
G = (V, E)  #Graf eliminacji
Permutate = [] #lista permutacji

while not visited node in G
    from V choose p with minimal degree
    visit p
    Permutate.append(p)
    actualize G
return Permutate
```

Kod 2: Culthill-McKee

```
M # macierz nXn
G = (V, E)  #Graf eliminacji
R = [x], where x is node with lowest degree

for i = 1,2,...
   if |R| >= n then break
   Ai = Adj(Ri) \ R #Construct the adjacency set Ai of R[i] node
   Ai.sort()
   R.append_all(Ai)
return R
```

Kod 3: Reversed Culthill-McKee

```
M # macierz nXn
G = (V, E)  #Graf eliminacji
return reverse(Culthill-McKee(M, G))
```

4 Ważniejsze fragmenty kodu

Kod 4: tworzenie grafu eliminacji

```
def graph(matrix):
n = len(matrix)
V = {}
for i in range(n):
    V[i] = set()
    for j in range(n):
        if matrix[i, j] != 0:
            V[i].add(j)
return V
```

Kod 5: metoda Minimum degree

```
def minimum_degree(matrix):
    n = len(matrix)
    V = graph(matrix)

    pq = [(len(edge), v) for v, edge in V.items()]
    heapify(pq)
    visited = [False for i in range(n)]
    permutation = []

    while pq:
        _, v = heappop(pq)
        if visited[v]:
```

```
            continue

        visited[v] = True
        permutation.append(v)

        for edge in V[v]:
            if not visited[edge]:
                V[edge].remove(v)
                heappush(pq, (len(V[edge]), edge))

    return permutation
```

Kod 6: metoda Culthill-McKee

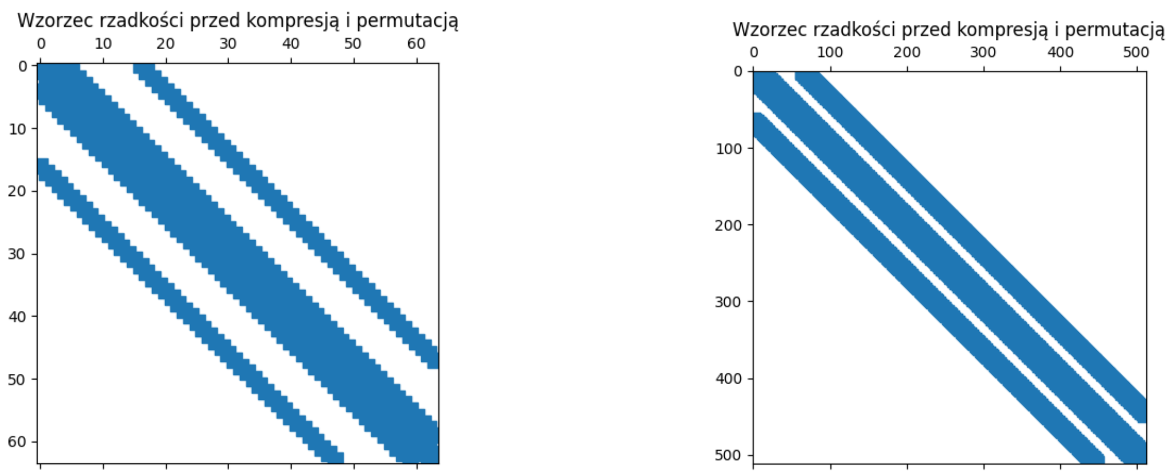
```
def cuthill_mckee(matrix):
    n = len(matrix)
    V = graph(matrix) # No need to sort because I use heapque


    all_vertex = [(-len(edge), v) for v, edge in V.items()] # reverse heap
    heapify(all_vertex)

    bfs_pq = [all_vertex[0]]
    visited = [False for i in range(n)]

    permutation = []
    while bfs_pq:

        # If graph is no consistent - bfs_pq empty but not all vertex visited
        while not bfs_pq and all_vertex:
            _, u = heappop(all_vertex)
            if not visited[u]: heappush(bfs_pq, (-len(V[u]), u))


        _, v = heappop(bfs_pq)

        if visited[v]:
            continue

        visited[v] = True
        permutation.append(v)

        for edge in V[v]:
            if not visited[edge]:
                heappush(bfs_pq, (-len(V[edge]), edge))

    return permutation
```

Kod 7: metoda reversed Culthill-McKee

```
def reversed_cuthill_mckee(matrix):
    return list(reversed(cuthill_mckee(matrix)))
```

5 Wyniki

5.1 Wzorce rzadkości macierzy

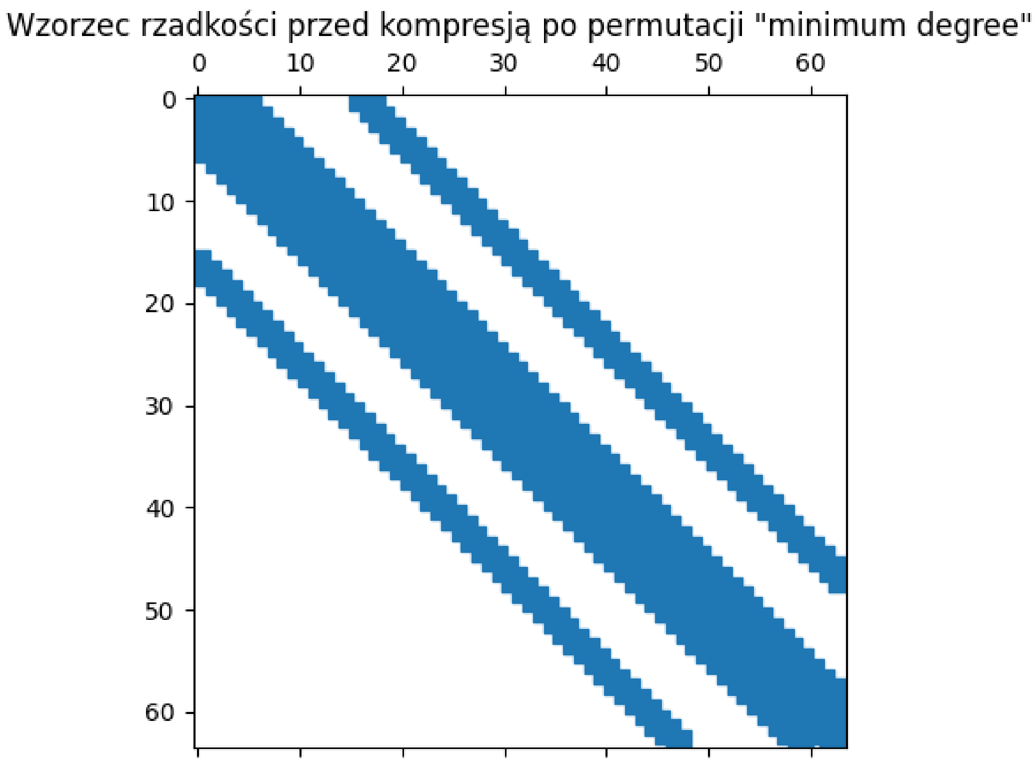


(a) k=2

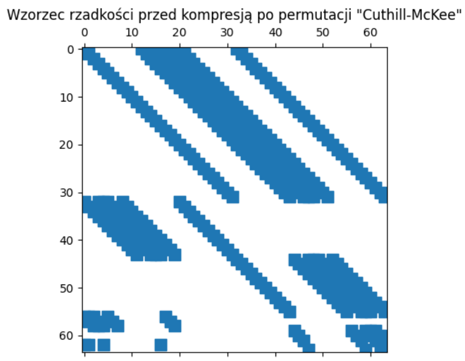
(b) k=3

Rysunek 1: Macierz orginalna dla różnych k

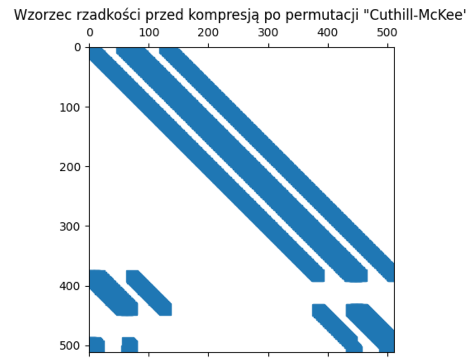
Dla zadanej klasy macierzy wejściowych, metoda permutacji "minimal degree" wykazuje odzwierciedlenie identycznościowe. To niepożądane zjawisko oznacza, że permutacja ta nie wprowadza istotnej zmiany w strukturze macierzy, zachowując jej podstawowe właściwości. Jest to problematyczne, gdyż celem permutacji jest zazwyczaj wprowadzenie pewnego stopnia rzadkości lub innego ułatwienia algorytmicznego, co w przypadku odzwierciedlenia identycznościowego nie zostaje osiągnięte.



Rysunek 2: permutacja minimal degree

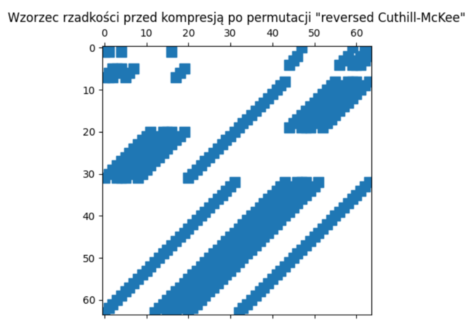


(a) k=2

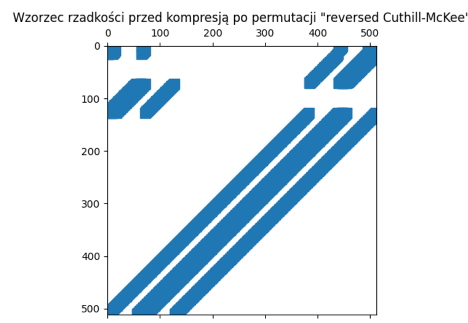


(b) k=3

Rysunek 3: permutacje Culthill-McKee



(a) k=2

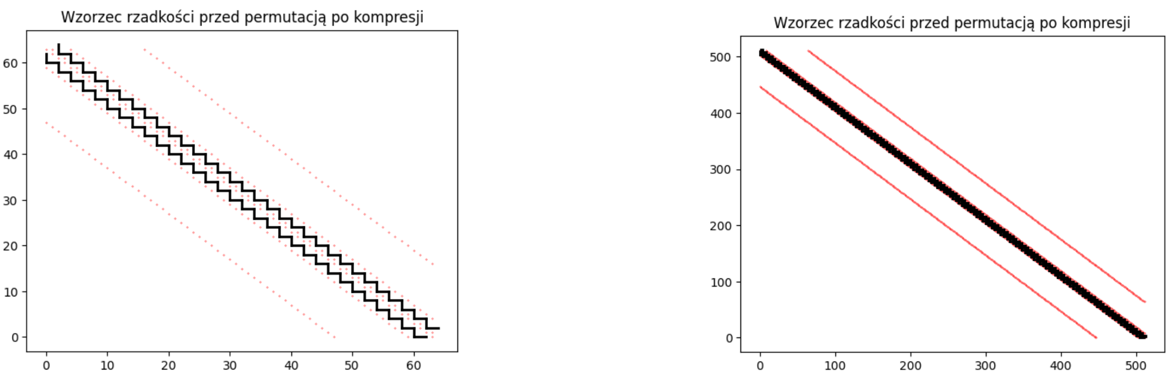


(b) k=3

Rysunek 4: permutacje reversed Culthill-McKee

Po obserwacji rysunków 3 i 4 można zauważyć, że dla mniejszych wartości k większa procentowa część została poddana permutacji.

5.2 Kompresja SVD



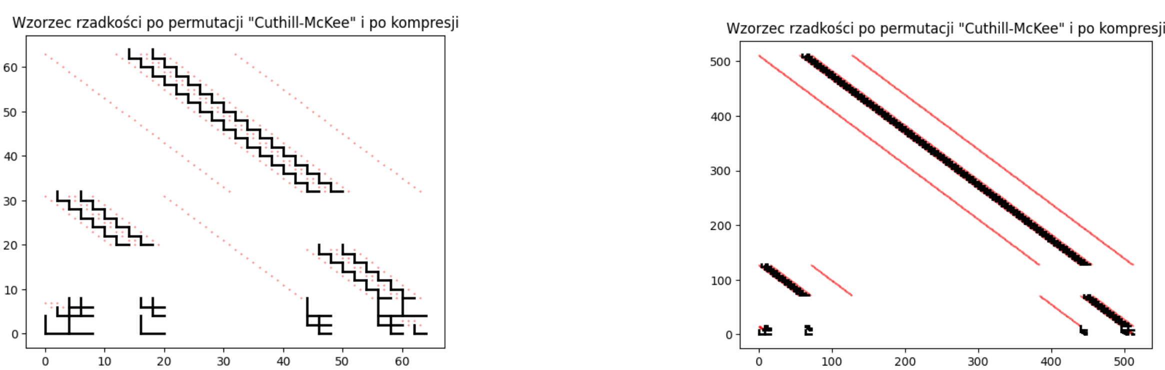
(a) k=2

(b) k=3

Rysunek 5: Kompresja SVD macierzy orginalnej

Zauważalne jest, że dla mniejszych wartości k znacznie bardziej wyraziste stają się obszary macierzy poddane kompresji. Dla przypadku k=3, na przekątnej widoczna jest czarna linia, wskazująca na liczne, mniejsze kompresje w tym obszarze.

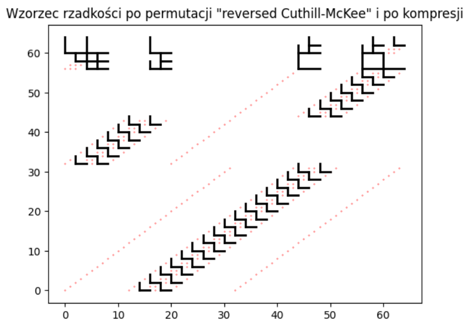
Ponadto, zauważalne jest, że nie udało się skompresować macierzy we wszystkich miejscach.



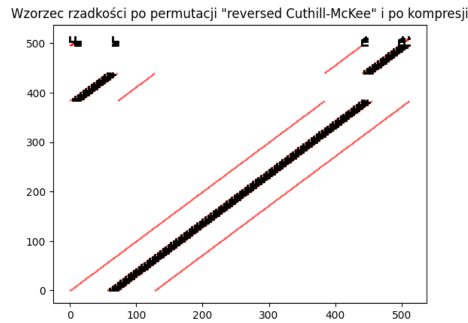
(a) k=2

(b) k=3

Rysunek 6: kompresjia po urzyciu metody Culthill-McKee



(a) k=2



(b) k=3

Rysunek 7: kompresjia po urzyciu metody reversed Culthill-McKee

Po zastosowaniu metod typu Culthill-McKee zauważalne są większe, oddzielone obszary, które zostały skompresowane. Ponadto, zauważamy większą ilość obszarów, które uległy kompresji (w dolnej części rysunku można dostrzec znacznie mniejszą liczbę czerwonych punktów).

6 Wnioski

- Metoda permutacji ”Minimum degree” okazała się bezużyteczna dla podanej klasy macierzy.
- Metody ”Culthill-McKee” oraz ”reversed Culthill-McKee” prowadzą do leprzej kompresji
- Macierz uzyskana za pomocą metody ”reversed Culthill-McKee” wygląda jak ta po zastosowaniu ”Culthill-McKee” obrócona o 180°
- Można zauważyć że dla mniejszego k większa częśc macierzy została przepermutowana.