Laboratorium 4

Permutacje macierzy i ich wpływ na kompresje

Szymon Twardosz, Dominik Jeżów

6 stycznia 2024

1 Środowisko

Do wykonania ćwiczenia wykorzystaliśmy język python 3.11 wraz z następującymi bibliotekami numpy, matplotlib, sklearn, math, collections

2 Temat zadania

Celem zadania było wygenerowanie macierzy o rozmiarze 2^{3k} (dla k = 2,3,4) 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.
Wszystkie kompresję przeprowadzone zostały dla następujących parametrów: rank równy 2 oraz sigma równa medianie wartości własnych macierzy pierwotnej.

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
visited = [False for i in range(|V|)]
queue = []
Permutate = []

for vertex in sorted(V)
    if vertex not visited
        queue = [vertex]
        BFS(G, queue, permutation, visited)
return Permutate
```

Kod 3: BFS

```
queue # kolejka BFS
visited # struktra pomocnicza
G = (V, E) # Graf

while queue not empty
    v = queue.pop()
    permutation.append(v)

    for neighbour in sorted(G[v])
        if neighbour not visited
            queue.append(neighbour)
```

Kod 4: Reversed Culthill-McKee

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

4 Ważniejsze fragmenty kodu

Kod 5: 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
return V
```

Kod 6: metoda Minimum degree

```
def minimum_degree(matrix):
  n = len(matrix)
  V = graph(matrix)
  pq = set([v for v in V.keys()])

  permutation = []

  for i in range(n):
      min_v, min_degree = None, inf

      for v in pq:
          if len(V[v]) < min_degree:
              min_degree = len(V[v])
              min_v = v

      permutation.append(min_v)

      pq.remove(min_v)

      for u in V:
          V[u].discard(min_v)

      for u in V[min_v]:
          V[u] = (V[u].union(V[min_v])).difference(set([u, min_v]))

  return permutation
```

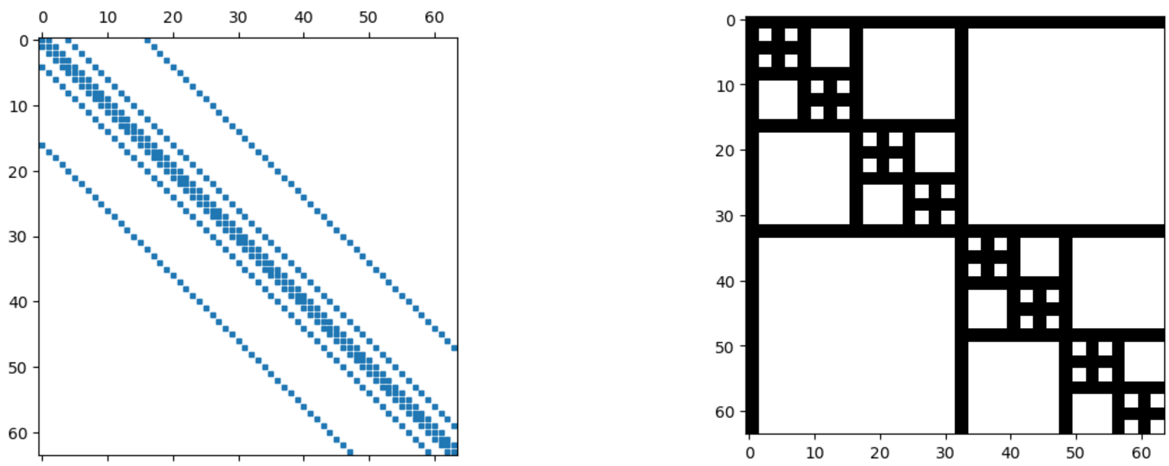
Kod 7: metoda Culthill-McKee

```
def cuthill_mckee(matrix):
  def BFS():
      while queue:
          v = queue.popleft()

          if visited[v]: continue

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

```
            for u in sorted(G[v], key=lambda x: len(G[x])):
                if not visited[u]:
                    queue.append(u)



    n = len(matrix)
    G = graph(matrix) # No need to sort because I use heapque

    visited = [False for _ in range(n)]
    queue = deque()
    sorted_nodes = sorted([(len(edges), v) for v, edges in G.items()], key=lambda x: x[0

    permutation = []
    for prior, node in sorted_nodes:
        if not visited[node]:
            queue.append(node)
            BFS()
    return permutation
```

Kod 8: metoda reversed Culthill-McKee

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



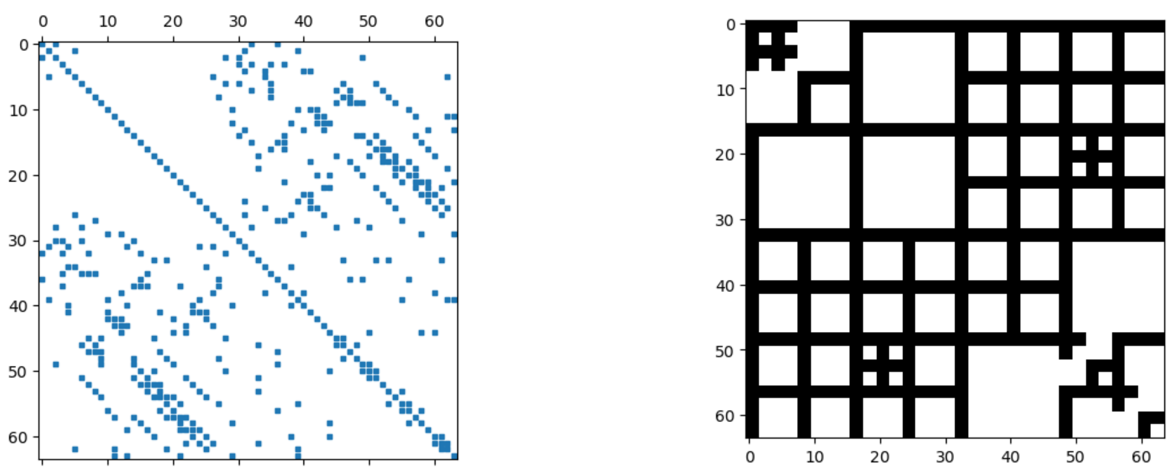
(a) Wzorzec rzadkości przed permutacją

(b) Wzorzec rzadkości po permutacji

Rysunek 1: Brak permutacji

5 Wyniki

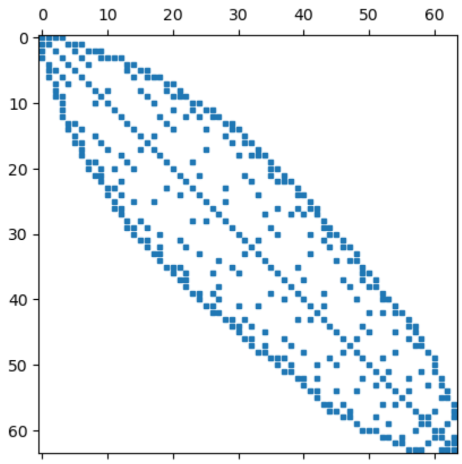
5.1 K = 2



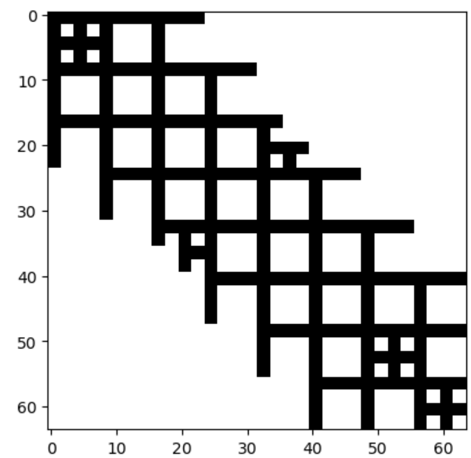
(a) Wzorzec rzadkości przed permutacją

(b) Wzorzec rzadkości po permutacji

Rysunek 2: Permutacja minimum degree

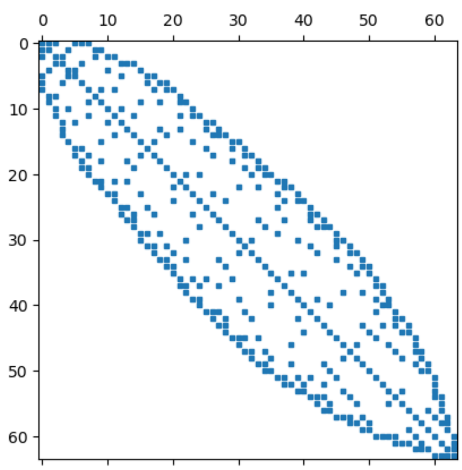


(a) Wzorzec rzadkości przed permutacją

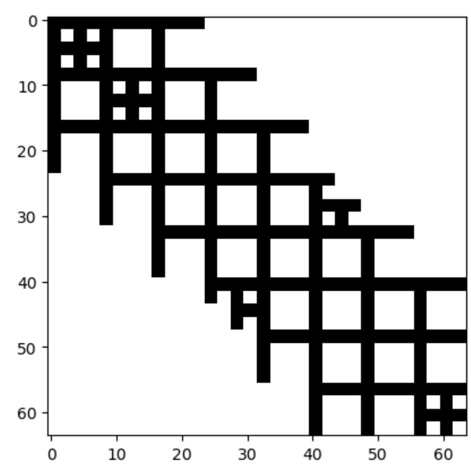


(b) Wzorzec rzadkości po permutacji

Rysunek 3: Permutacja Cuthill-McKee

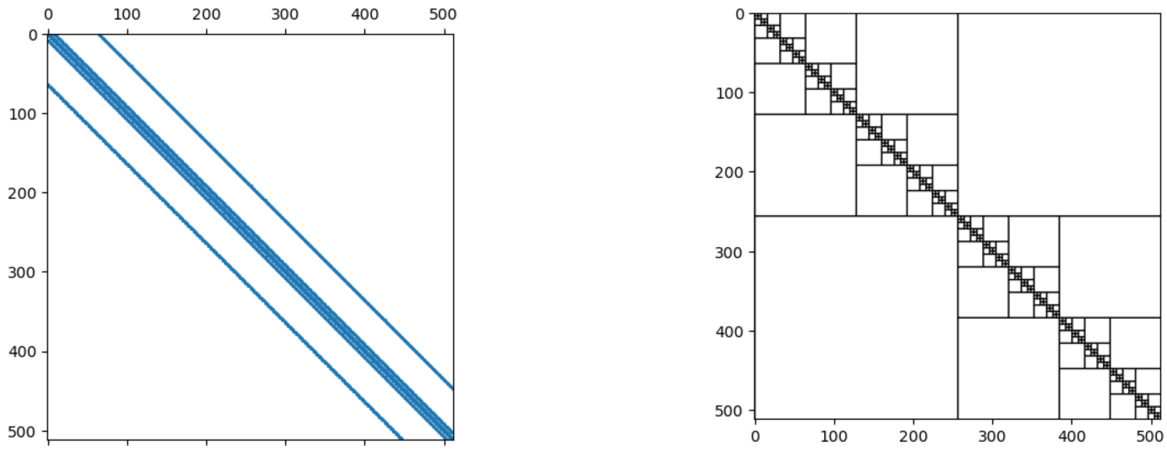


(a) Wzorzec rzadkości przed permutacją



(b) Wzorzec rzadkości po permutacji

Rysunek 4: Permutacja Reverse Cuthill-McKee

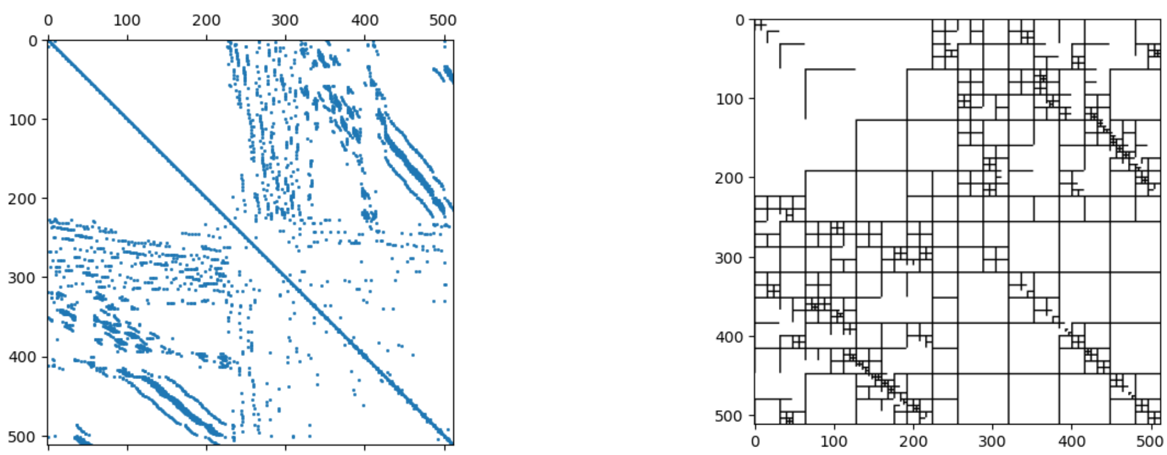


(a) Wzorzec rzadkości przed permutacją

(b) Wzorzec rzadkości po permutacji

Rysunek 5: Brak permutacji

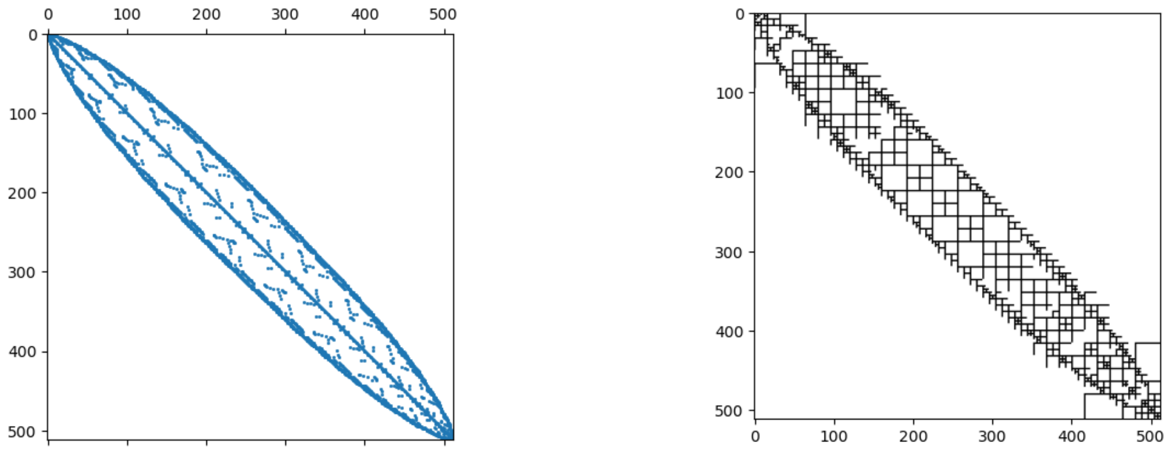
5.2 K = 3



(a) Wzorzec rzadkości przed permutacją

(b) Wzorzec rzadkości po permutacji

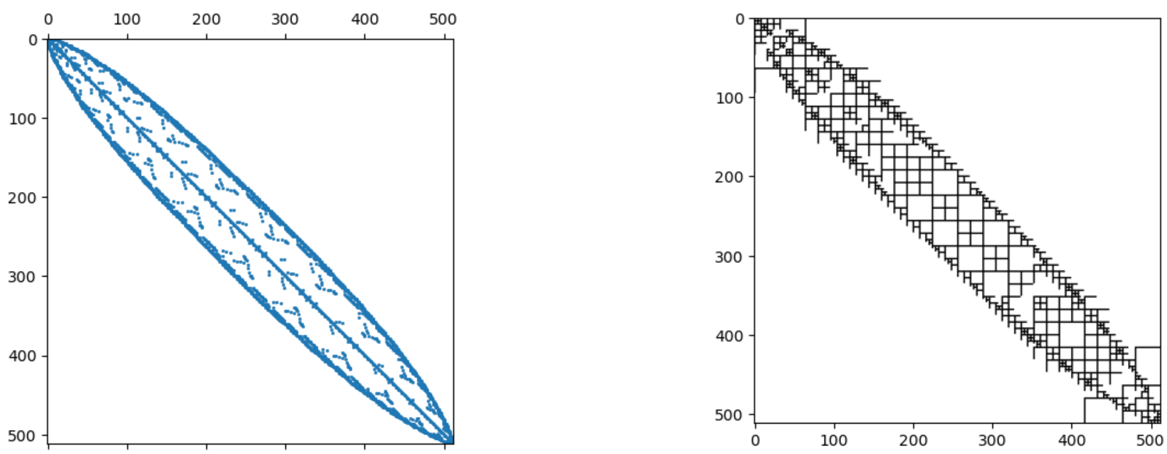
Rysunek 6: Permutacja minimum degree



(a) Wzorzec rzadkości przed permutacją

(b) Wzorzec rzadkości po permutacji

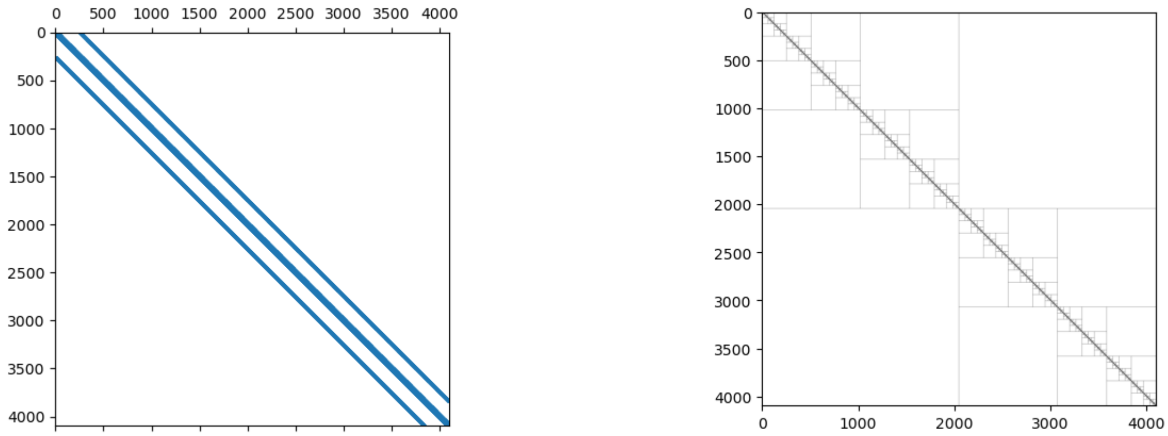
Rysunek 7: Permutacja Cuthill-McKee



(a) Wzorzec rzadkości przed permutacją

(b) Wzorzec rzadkości po permutacji

Rysunek 8: Permutacja Reverse Cuthill-McKee

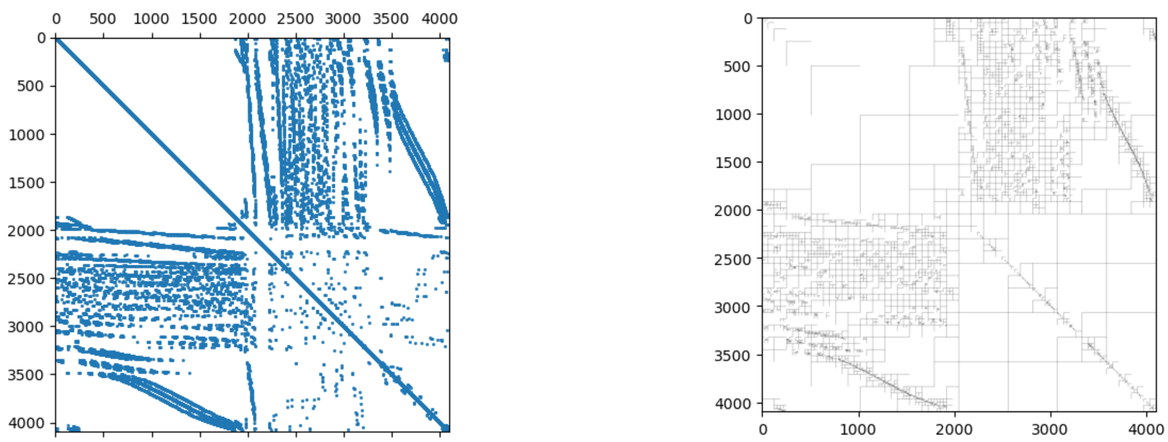


(a) Wzorzec rzadkości przed permutacją

(b) Wzorzec rzadkości po permutacji

Rysunek 9: Brak permutacji

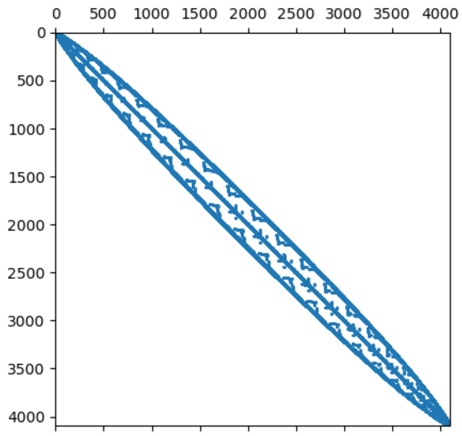
5.3 K = 4



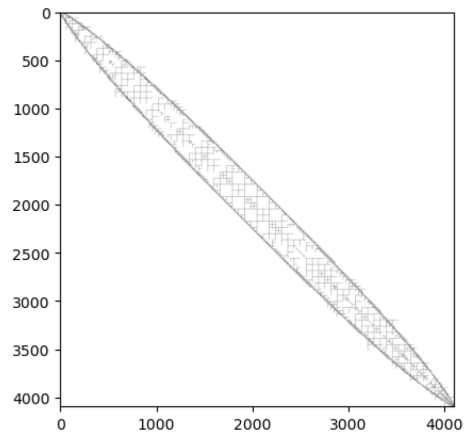
(a) Wzorzec rzadkości przed permutacją

(b) Wzorzec rzadkości po permutacji

Rysunek 10: Permutacja minimum degree

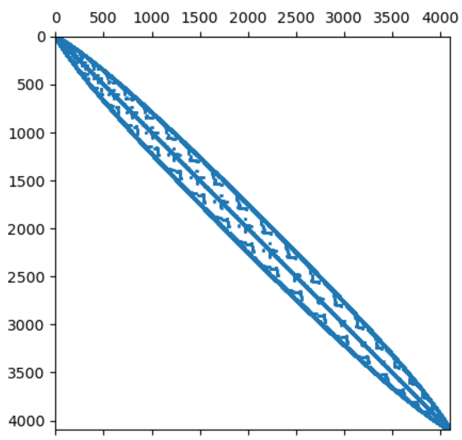


(a) Wzorzec rzadkości przed permutacją

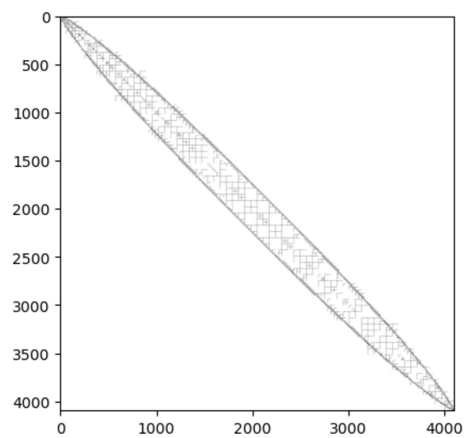


(b) Wzorzec rzadkości po permutacji

Rysunek 11: Permutacja Cuthill-McKee



(a) Wzorzec rzadkości przed permutacją



(b) Wzorzec rzadkości po permutacji

Rysunek 12: Permutacja Reverse Cuthill-McKee

6 Stopień kompresji

Stopień kompresji mierzony był korzystając z następującego wzoru

$S = T/|M|$, gdzie
T - liczba elementów w drzewie, które są różne od zera M - liczba wszystkich elementów w oryginalnej macierz

Zatem czym mniejsza jest ta metryka, tym macierz jest lepiej skompresowana (zajmuję mniej miejsca)

	K	2	3	4
Algorytm permutacji	------------	------------	------------	------------
Brak	------------	0.335	0.065	0.0012
Minimum Degree	------------	0.458	0.125	0.0294
Cuthill-McKee	------------	0.319	0.077	0.0115
Reverse Cuthill-McKee	------------	0.319	0.077	0.0115

Rysunek 13: Stopień kompresji a użyte algorytmy permutacji

7 Wnioski

1. Dla macierzy opisującej topologię trójwymiarowej siatki algorytm kompresji działa najefektywniej, gdy nie modyfikujemy kolejności wierszy/kolumn macierzy.
2. Macierze przepermutowane metodą Cuthill-McKee/Reverse Cuthill-McKee lepiej radzą sobię z kompresją, niż algorytm Minimum degree.
3. Dla wszystkich rozmiarów macierzy obie metody - Cuthill-McKee i Reverse Cuthill-McKee zwracają dokładnie ten sam stopień kompresji.
4. Stopień kompresji maleje wraz z większającym się rozmiarem macierzy wejściowej