Algorytmy Macierzowe
Sprawozdanie 5
Reordering w celu optymalizacji kompresji macierzy

Przemek Węglik
Szymon Paszkiewicz

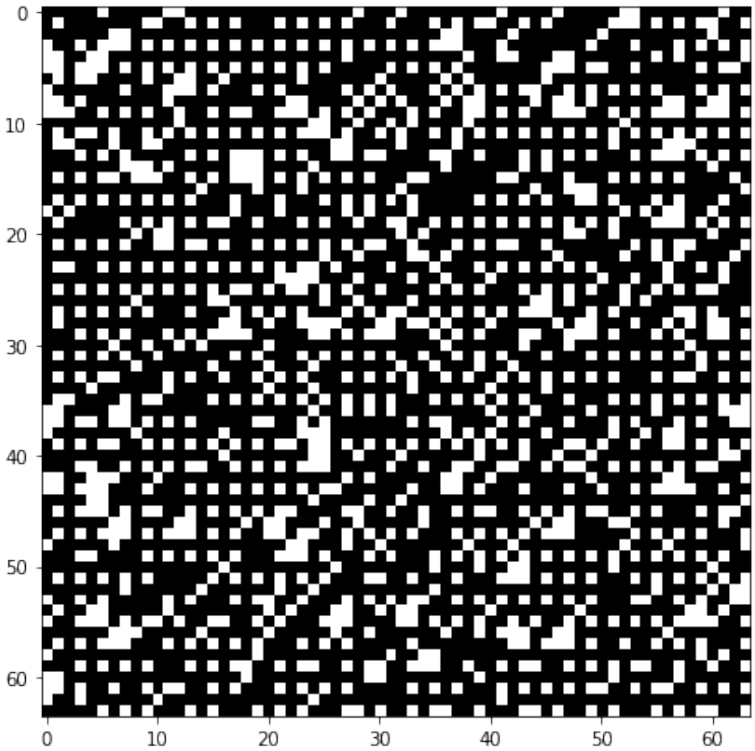
11 lutego 2023

1 Fragmenty kodu

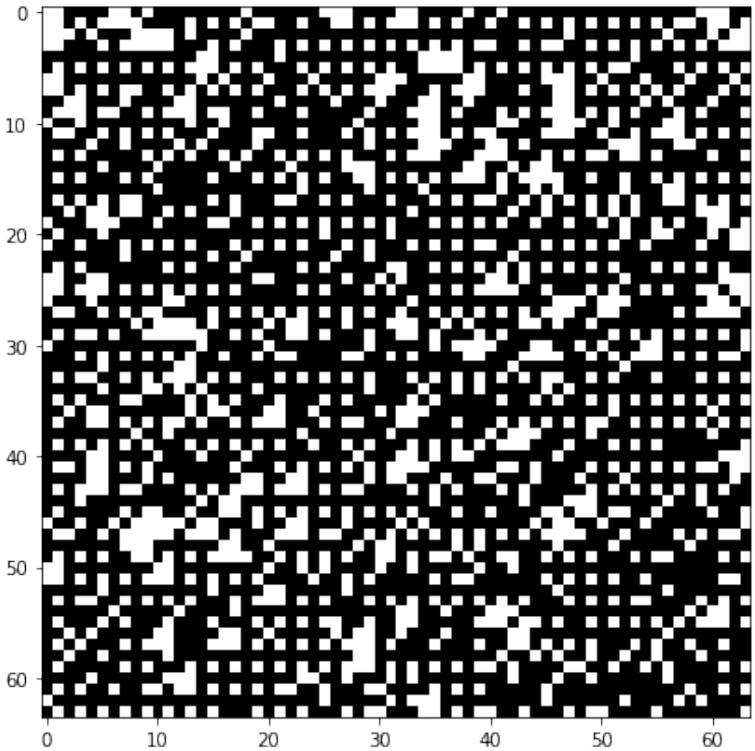
Funkcja realizująca minimum degree reordering:

```
def minimum_degree_transformation(input_matrix: np.ndarray) -> np.ndarray:
E = list(zip(*np.nonzero(input_matrix)))
graph = [{i} for i in range(input_matrix.shape[0])]
for e in E:
    graph[e[0]].add(e[1])
    graph[e[1]].add(e[0])

order = []
for _ in range(len(graph)):
    v = -1
    min_deg = len(input_matrix)

    for j, s in enumerate(graph):
        if len(s) != 0 and len(s) < min_deg:
            v = j
            min_deg = len(s)

    order.append(v)
    for s in graph:
        if v in s:
            s.remove(v)
    graph[v].clear()

inverted = [None for _ in range(len(order))]
for i, o in enumerate(order):
    inverted[o] = i

result = np.zeros(input_matrix.shape)
for coord in E:
    result[inverted[coord[0]]][inverted[coord[1]]] = input_matrix[coord]

return result
```

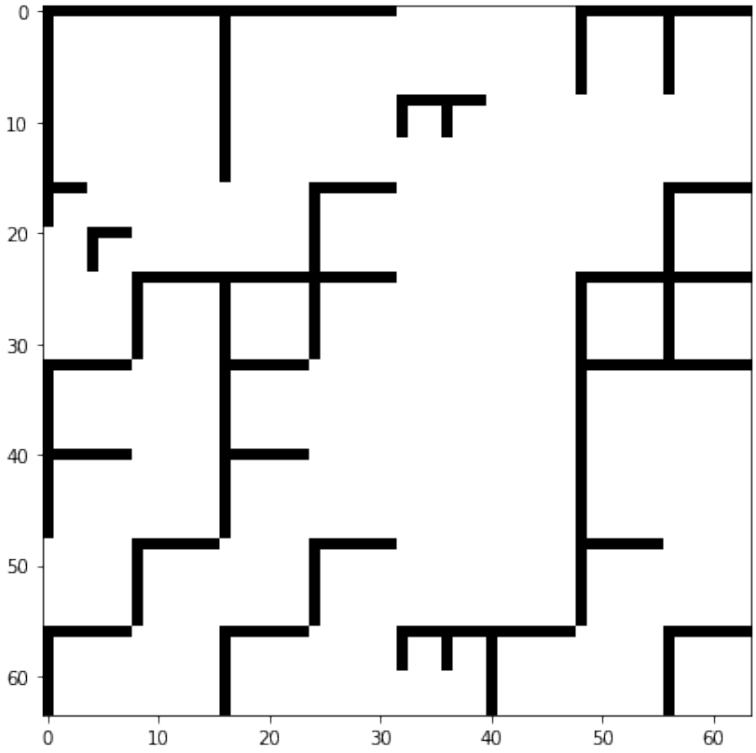
2 Rysunki



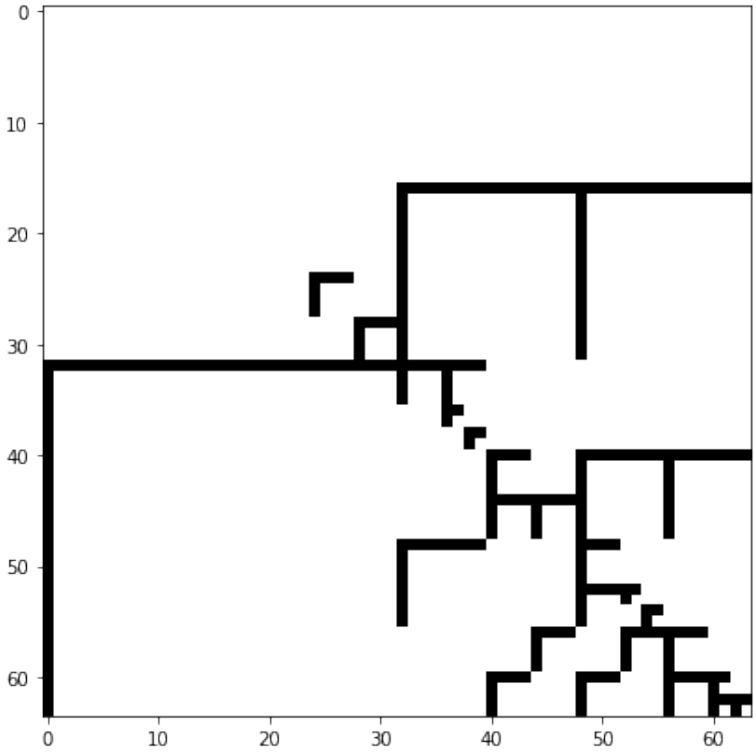
Rysunek 1: Macierz gęsta (0.5) przed reorderingiem



Rysunek 2: Macierz gęsta (0.5) po reorderingu



Rysunek 3: Macierz rzadka (0.01) przed reorderingiem



Rysunek 4: Macierz rzadka (0.01) po reorderingu

3 Wnioseki

Widzimy, że dla macierzy gęstych stosowanie reorderingu mija się z celem i nie jest efektywne. Z kolei podczas używania macierzy rzadkich, możemy uzyskać znaczą optymalizację. Im rzadsza macierz tym lepsze rezultaty daje reordering.