Automaty komórkowe

http://home.agh.edu.pl/malarz/dyd/ak/

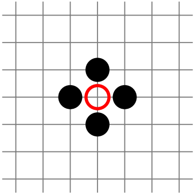
v. 2.718281828459045235360287

Sieci i otoczenia. Proste i złożone

Krzysztof Malarz

4 maja 2024

SQ-1 [1, 2] I



Rysunek: SQ-1

SQ-1 [1, 2] II

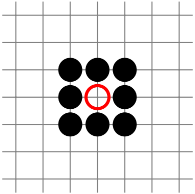
Listing 1: boundaries() dla SQ-1

```
void boundaries()
{
  int i,j;
  for (i=0; i<N; i++) {
// 1nn core:
     nn[i][0] = (N+i +1)%N;
     nn[i][1] = (N+i -1)%N;
     nn[i][2] = (N+i +L)%N;
     nn[i][3] = (N+i -L)%N;
// 1nn left border:
     if (i%L==0)      nn[i][1] = (N+i+L -1)%N;
// 1nn right border:
     if ((i+1)%L==0) nn[i][0] = (N+i-L +1)%N;
  }
```

SQ-1 [1, 2] III

```
}
```

SQ-1,2 [1, 2] I



Rysunek: SQ-1,2

SQ-1,2 [1, 2] II

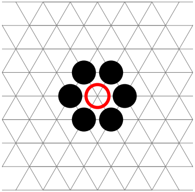
Listing 2: `boundaries()` dla SQ-2

```
void boundaries()
{
  int i,j;
  for (i=0; i<N; i++) {
// 2nn core:
     nn[i][0] = (N+i +L+1)%N;
     nn[i][1] = (N+i +L-1)%N;
     nn[i][2] = (N+i -L+1)%N;
     nn[i][3] = (N+i -L-1)%N;
// 2nn left border:
     if(i%L==0) {
         nn[i][1] = (N+i+L +L-1)%N;
         nn[i][3] = (N+i+L -L-1)%N; }
// 2nn right border:
```

SQ-1,2 [1, 2] III

```
    if((i+1)%L==0) {
       nn[i][0] = (N+i-L +L+1)%N;
       nn[i][2] = (N+i-L -L+1)%N; }
  }
}
```

TR-1 [2, 3] I



Rysunek: TR-1

TR-1 [2, 3] II

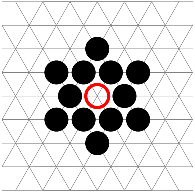
Listing 3: boundaries() dla TR-1

```
void boundaries()
{
int i,j;
for (i=0; i<N; i++){
// ---------------------------------------------------
// 1NN:
      nn[i][0] = (N+i -L  )%N;
      nn[i][1] = (N+i -L+1)%N;
      nn[i][2] = (N+i -1  )%N;
      nn[i][3] = (N+i +1  )%N;
      nn[i][4] = (N+i +L-1)%N;
      nn[i][5] = (N+i +L  )%N;
// 1NN:
      if (i%L==0) {
```

TR-1 [2, 3] III

```
          nn[i][2] = (N+L+i -1  )%N;
          nn[i][4] = (N+L+i +L-1)%N; }
        if ((i+1)%L==0) {
          nn[i][1] = (N-L+i -L+1)%N;
          nn[i][3] = (N-L+i +1  )%N; }
// ----------------------------------------------------
  }
}
```

TR-1,2 [2, 3] I



Rysunek: TR-1,2

TR-1,2 [2, 3] II

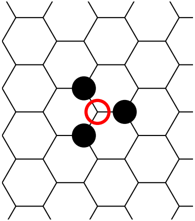
Listing 4: boundaries() dla TR-2

```
void boundaries()
{
int i,j;
for (i=0; i<N; i++){
// -------------------------------------------------------
// 2NN:
     nn[i][0] = (N+i -2*L+1)%N;
     nn[i][1] = (N+i -L-1   )%N;
     nn[i][2] = (N+i -L+2   )%N;
     nn[i][3] = (N+i +L-2   )%N;
     nn[i][4] = (N+i +L+1   )%N;
     nn[i][5] = (N+i +2*L-1)%N;
// 2NN:
     if (i%L==0) {
```

TR-1,2 [2, 3] III

```
      nn[i][1] = (N+L+i -L-1  )%N;
      nn[i][5] = (N+L+i +2*L-1)%N;
      nn[i][3] = (N+L+i +L-2  )%N; }
    if (i%L==1) {
      nn[i][3] = (N+L+i +L-2  )%N; }
    if ((i+2)%L==0) {
      nn[i][2] = (N-L+i -L+2  )%N; }
    if ((i+1)%L==0) {
      nn[i][0] = (N-L+i -2*L+1)%N;
      nn[i][4] = (N-L+i +L+1  )%N;
      nn[i][2] = (N-L+i -L+2  )%N; }
// ---------------------------------------------
  }
}
```

HC-1 [2, 4] I



Rysunek: HC-1

HC-1 [2, 4] II

Listing 5: boundaries() dla HC-1

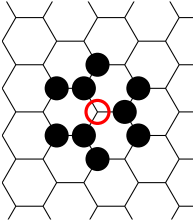
```
void boundaries() {
int i,row,col;

for(row=0; row<L; row++)
for(col=0; col<L; col++) {
   i=row*L+col;
// 1NN core:
   nn[i][0] = (N+i -1)%N;
   nn[i][1] = (N+i +1)%N;
   if(row%2==0) {
      if(col%2!=0)
         nn[i][2] = (N+i +L)%N;
      else
         nn[i][2] = (N+i -L)%N; }
```

HC-1 [2, 4] III

```
    else {
        if(col%2==0)
            nn[i][2] = (N+i +L)%N;
        else
            nn[i][2] = (N+i -L)%N; }
// 1NN left border:
    if(i%L==0)
        nn[i][0] = (N+L+i -1)%N;
// 1NN right border:
    if((i+1)%L==0)
        nn[i][1] = (N-L+i +1)%N;
    }
}
```

HC-1,2 [2, 4] I



Rysunek: HC-1,2

HC-1,2 [2, 4] II

Listing 6: `boundaries()` dla HC-2

```
void boundaries() {
int i,row,col;

for(row=0; row<L; row++)
for(col=0; col<L; col++) {
    i=row*L+col;
// 2NN core:
    nn[i][0] = (N+i -2  )%N;
    nn[i][1] = (N+i +2  )%N;
    nn[i][2] = (N+i +L-1)%N;
    nn[i][3] = (N+i +L+1)%N;
    nn[i][4] = (N+i -L-1)%N;
    nn[i][5] = (N+i -L+1)%N;
// 2NN left border:
```

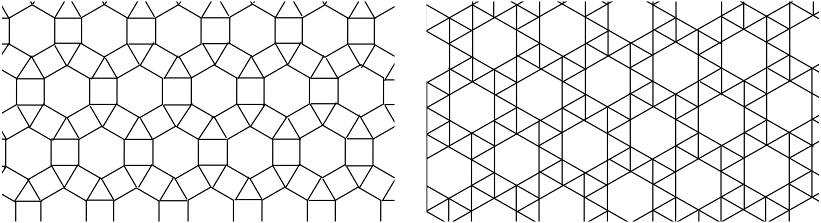
HC-1,2 [2, 4] III

```
    if(i%L==0) {
        nn[i][0] = (N+L+i -2  )%N;
        nn[i][2] = (N+L+i +L-1)%N;
        nn[i][4] = (N+L+i -L-1)%N; }
    if(i%L==1) {
        nn[i][0] = (N+L+i -2  )%N; }
// 2NN right border:
    if((i+1)%L==0) {
        nn[i][1] = (N-L+i +2  )%N;
        nn[i][3] = (N-L+i +L+1)%N;
        nn[i][5] = (N-L+i -L+1)%N; }
    if((i+2)%L==0) {
        nn[i][1] = (N-L+i +2  )%N; }
    }
}
```

Sieci Archimedesa I

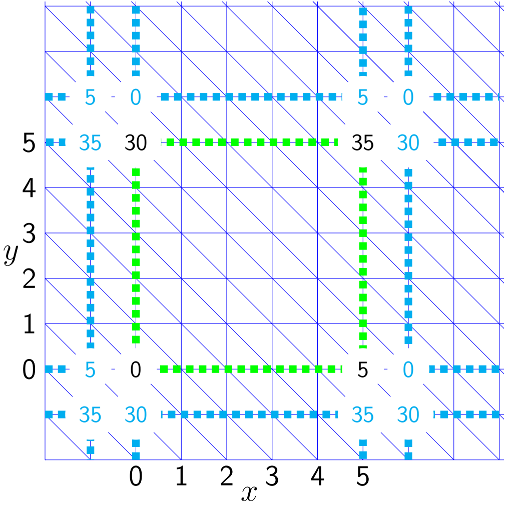
Siatki Archimedesa to grafy translacyjne wierzchołków, które można osadzić na płaszczyźnie tak, aby każda ściana była wielokątem foremnym [5]. Kepler pokazał, że istnieje dokładnie jedenaście takich grafów. Nazwy sieci nadawane są według wielokątów przypadających na dany wierzchołek. Liczby boków tych wielokątów są wymieniane w takiej kolejności aby utowrzona sekwencja była możliwie najmniejsza w porządku leksykograficznym. W ten sposób siatka kwadratowa otrzymuje nazwę $(4, 4, 4, 4)$, w skrócie (4^4), plaster miodu nazywa się (6^3), a siatka Kagomé to $(3, 6, 3, 6)$.

Sieci Archimedesa II

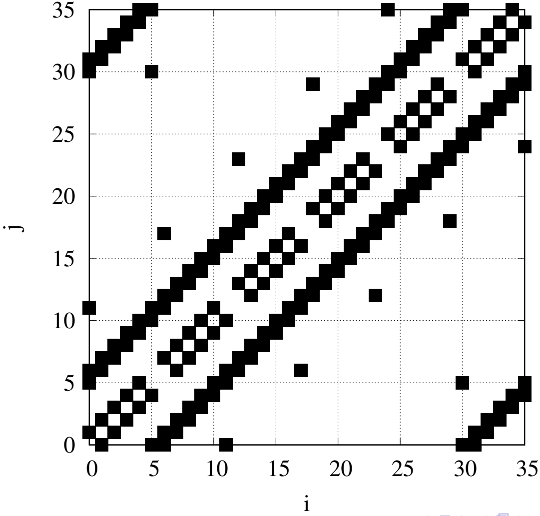


Rysunek: Sieci Archimedesa $(3, 4, 6, 4)$ i $(3^4, 6)$

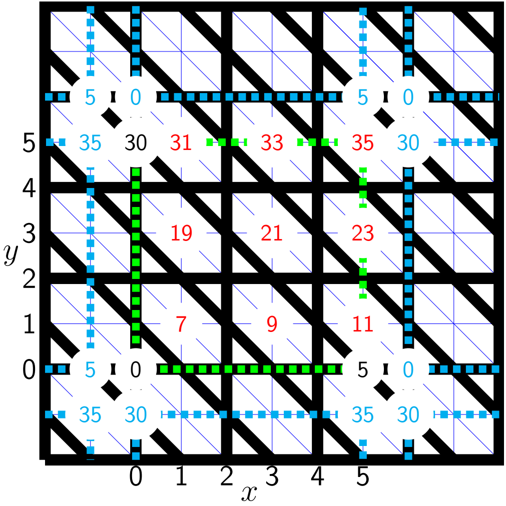
Siatka (3^6) (trójkątna), $k = 6$



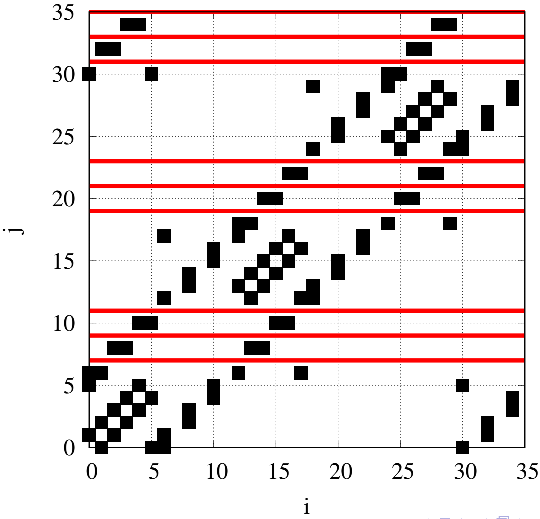
Siatka (3^6) (trójkątna), $k = 6$



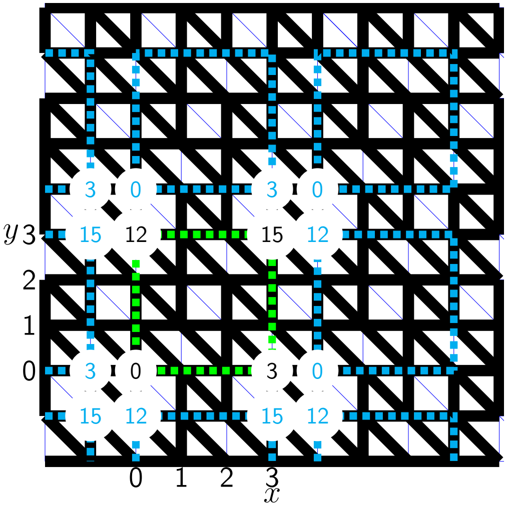
Siatka $(3, 6, 3, 6)$ (kagomé), $k = 6$



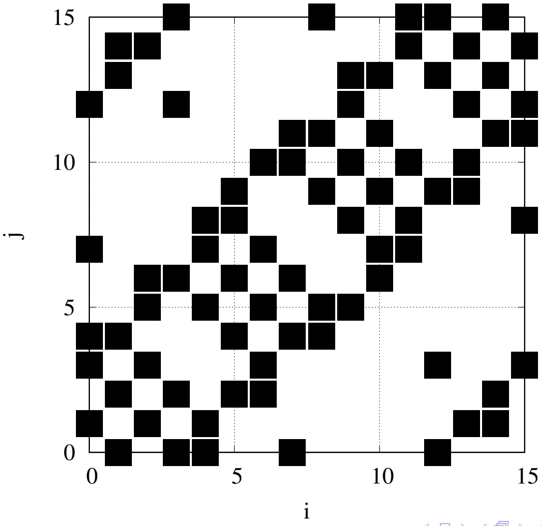
Siatka $(3, 6, 3, 6)$ (kagomé), $k = 6$



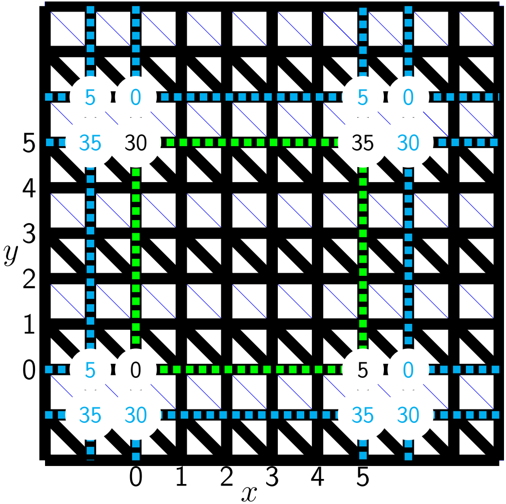
Siatka $(3^2, 4, 3, 4)$, $k = 5$



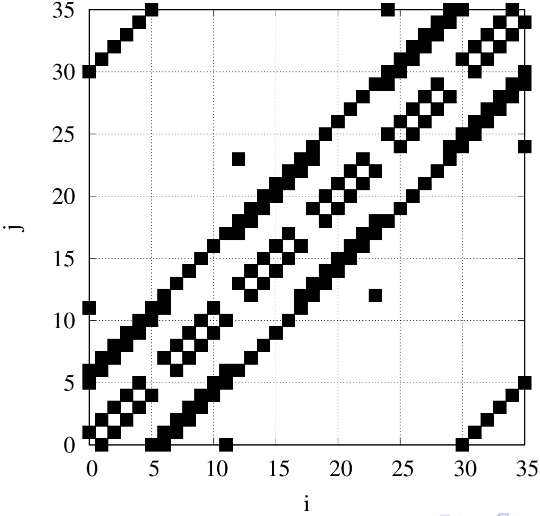
Siatka $(3^2, 4, 3, 4)$, $k = 5$



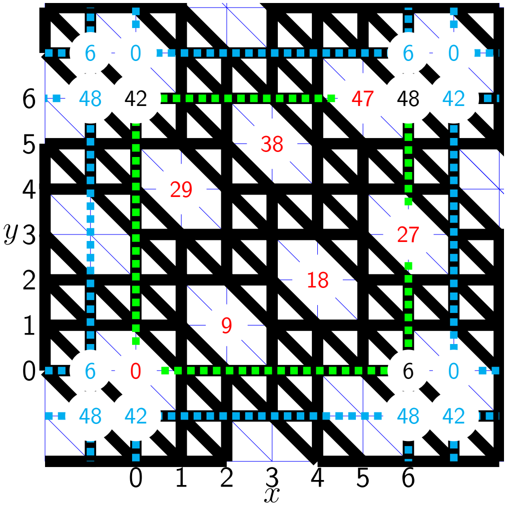
Siatka $(3^3, 4^2)$, $k = 5$



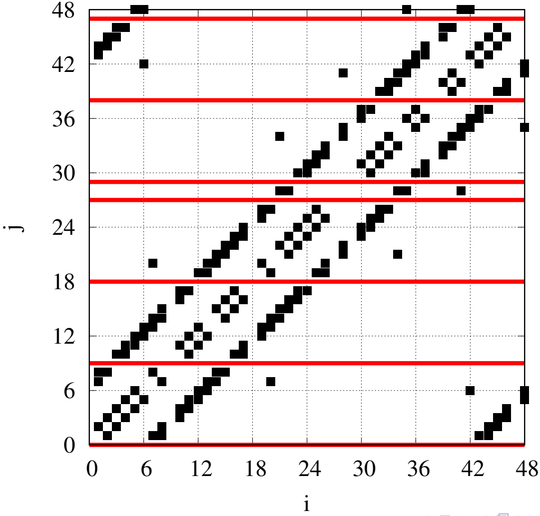
Siatka $(3^3, 4^2)$, $k = 5$



Siatka $(3^4, 6)$, $k = 5$



Siatka $(3^4, 6)$ lattice, $k = 5$



Kody I

Listing 7: Procedure for triangular (3^6) lattice

```
SUBROUTINE triangular(A)
!! ----------------------------------------
USE par
IMPLICIT NONE
INTEGER, DIMENSION(0:N-1,0:Z-1) :: nn
INTEGER, DIMENSION(0:N-1,0:N-1) :: A
INTEGER :: i, j
!! ----------------------------------------
DO i=0, N-1
  nn(i,0) = MOD(N+i -W   ,N)
  nn(i,1) = MOD(N+i -W+1 ,N)
  nn(i,2) = MOD(N+i -1   ,N)
  nn(i,3) = MOD(N+i +1   ,N)
  nn(i,4) = MOD(N+i +W-1 ,N)
  nn(i,5) = MOD(N+i +W   ,N)
```

Kody II

```fortran
  IF(MOD(i,W)==0) THEN
    nn(i,2) = MOD(N+W+i -1   ,N)
    nn(i,4) = MOD(N+W+i +W-1,N)
  ENDIF
  IF(MOD(i+1,W)==0) THEN
    nn(i,1) = MOD(N-W+i -W+1,N)
    nn(i,3) = MOD(N-W+i +1   ,N)
  ENDIF
ENDDO
!! -------------------------------------------
A=0
DO i=0,N-1
DO j=0,Z-1
  A(i,nn(i,j))=1
ENDDO
ENDDO
!! -------------------------------------------
```

Kody III

```
END SUBROUTINE triangular
```

Listing 8: Procedure for Archimedean $(3^3, 4^2)$ lattice

```
SUBROUTINE al_33344(A)
!! -----------------------------------------
USE par
IMPLICIT NONE
INTEGER, DIMENSION(0:N-1,0:Z-1) :: nn
INTEGER, DIMENSION(0:N-1,0:N-1) :: A
INTEGER :: i, j, x, y
!! -----------------------------------------
DO y=0,W-1,2
DO x=0,W-1
  i=x+y*W
  nn(i,0) = MOD(N+i -1  ,N)
  nn(i,1) = MOD(N+i +1  ,N)
```

Kody IV

```
   nn(i,2) = MOD(N+i -W   ,N)
   nn(i,3) = MOD(N+i +W   ,N)
   nn(i,4) = MOD(N+i +W-1,N)
   IF(MOD(i,W)==0) THEN
      nn(i,0) = MOD(N+W+i -1   ,N)
      nn(i,4) = MOD(N+W+i +W-1,N)
   ENDIF
   IF(MOD(i+1,W)==0) THEN
      nn(i,1) = MOD(N-W+i +1,N)
   ENDIF
ENDDO
ENDDO
!! ----------------------------------
DO y=1,W-1,2
DO x=0,W-1
   i=x+y*W
   nn(i,0) = MOD(N+i -1   ,N)
```

Kody V

```
  nn(i,1) = MOD(N+i +1  ,N)
  nn(i,2) = MOD(N+i -W  ,N)
  nn(i,3) = MOD(N+i +W  ,N)
  nn(i,4) = MOD(N+i -W+1,N)
  IF(MOD(i,W)==0) THEN
    nn(i,0) = MOD(N+W+i -1,N)
  ENDIF
  IF(MOD(i+1,W)==0) THEN
    nn(i,1) = MOD(N-W+i +1,N)
    nn(i,4) = MOD(N-W+i -W+1,N)
  ENDIF
ENDDO
ENDDO
!! ----------------------------------
A=0
DO i=0,N-1
DO j=0,Z-1
```

Kody VI

```
  A(i,nn(i,j))=1
ENDDO
ENDDO
!! ---------------------------------------
END SUBROUTINE al_33344
```

Listing 9: Procedure for kagomé $(3, 6, 3, 6)$ lattice

```
SUBROUTINE tri_2_kagome(A)
!! ---------------------------------------
USE par
IMPLICIT NONE
INTEGER, DIMENSION(0:N-1,0:N-1) :: A
INTEGER :: i, j, x, y
!! ---------------------------------------
DO y=1,W,2
DO x=1,W,2
```

Kody VII

```
  i=x+W*y
  DO j=0,N-1
    A(i,j)=0
    A(j,i)=0
  ENDDO
ENDDO
ENDDO
!! ------------------------------------
END SUBROUTINE tri_2_kagome
```

Kody VIII

Listing 10: Procedure for maple leaf $(3^3, 6)$ lattice

```
SUBROUTINE tri_2_maple_leaf(A)
!! ------------------------------------------
USE par
IMPLICIT NONE
INTEGER, DIMENSION(0:N-1,0:N-1) :: A
INTEGER :: i, j, k, x, y
!! ------------------------------------------
DO y=0,W-1
DO k=1,S
  x = MOD(2*y+7*k,W)
  i = x + W*y
  DO j=0,N-1
    A(i,j)=0
    A(j,i)=0
```

Kody IX

```
   ENDDO
ENDDO
ENDDO
!! ---------------------------------------
END SUBROUTINE tri_2_maple_leaf
```

Listing 11: Procedure for Archimedean $(3^2, 4, 3, 4)$ lattice

```
SUBROUTINE tri_2_al33434(A)
USE par
IMPLICIT NONE
INTEGER, DIMENSION(0:N-1,0:Z-1) :: nn
INTEGER, DIMENSION(0:N-1,0:N-1) :: A
INTEGER :: i, x ,y
!! ---------------------------------------
!! neighbors on triangular lattice
DO i=0, N-1
```

Kody X

```
  nn(i,0) = MOD(N+i -W  ,N)
  nn(i,1) = MOD(N+i -W+1,N)
  nn(i,2) = MOD(N+i -1  ,N)
  nn(i,3) = MOD(N+i +1  ,N)
  nn(i,4) = MOD(N+i +W-1,N)
  nn(i,5) = MOD(N+i +W  ,N)
  IF(MOD(i,W)==0) THEN
    nn(i,2) = MOD(N+W+i -1  ,N)
    nn(i,4) = MOD(N+W+i +W-1,N)
  ENDIF
  IF(MOD(i+1,W)==0) THEN
    nn(i,1) = MOD(N-W+i -W+1,N)
    nn(i,3) = MOD(N-W+i +1  ,N)
  ENDIF
ENDDO
!! -----------------------------------------
!! adjacency matrix for (3^2,4,3,4) lattice:
```

Kody XI

```
DO  x=0,W-1
DO  y=0,W-1
   i=x+y*W
   IF(MOD(x,W).EQ.0  .AND.  MOD(y,W).EQ.0)
       A(i,nn(i,1))=0
   IF(MOD(x,W).EQ.1  .AND.  MOD(y,W).EQ.0)
       A(i,nn(i,5))=0
   IF(MOD(x,W).EQ.2  .AND.  MOD(y,W).EQ.0)
       A(i,nn(i,1))=0
   IF(MOD(x,W).EQ.3  .AND.  MOD(y,W).EQ.0)
       A(i,nn(i,5))=0

   IF(MOD(x,W).EQ.0  .AND.  MOD(y,W).EQ.1)
       A(i,nn(i,4))=0
   IF(MOD(x,W).EQ.1  .AND.  MOD(y,W).EQ.1)
       A(i,nn(i,0))=0
   IF(MOD(x,W).EQ.2  .AND.  MOD(y,W).EQ.1)
```

Kody XII

```
      A(i,nn(i,4))=0
   IF(MOD(x,W).EQ.3 .AND. MOD(y,W).EQ.1)
      A(i,nn(i,0))=0

   IF(MOD(x,W).EQ.0 .AND. MOD(y,W).EQ.2)
      A(i,nn(i,5))=0
   IF(MOD(x,W).EQ.1 .AND. MOD(y,W).EQ.2)
      A(i,nn(i,1))=0
   IF(MOD(x,W).EQ.2 .AND. MOD(y,W).EQ.2)
      A(i,nn(i,5))=0
   IF(MOD(x,W).EQ.3 .AND. MOD(y,W).EQ.2)
      A(i,nn(i,1))=0

   IF(MOD(x,W).EQ.0 .AND. MOD(y,W).EQ.3)
      A(i,nn(i,0))=0
   IF(MOD(x,W).EQ.1 .AND. MOD(y,W).EQ.3)
      A(i,nn(i,4))=0
```

Kody XIII

```
  IF(MOD(x,W).EQ.2 .AND. MOD(y,W).EQ.3)
     A(i,nn(i,0))=0
  IF(MOD(x,W).EQ.3 .AND. MOD(y,W).EQ.3)
     A(i,nn(i,4))=0
ENDDO
ENDDO
END SUBROUTINE tri_2_al33434
```

3D, SC-1, $k = 6$ I

Najbliżsi sąsiedzi na sieci kubicznej prostej w $i \pm 1$, $i \pm L$ oraz $i \pm L^2$

4D, SC-1, $k = 8$ I

Najbliżsi sąsiedzi na sieci hiperkubicznej prostej w $i \pm 1$, $i \pm L$, $i \pm L^2$ oraz $i \pm L^3$

Sieci rosnące I

W przygotowaniu na podstawie [6].
At each growth step i a node is added and linked to M nodes selected among $i-1$ preexisting nodes. Selection of a target node, k, is given by a specified probability $p(k,i)$ [6].

- $p(k,i) = 1/(i-1) \quad \rightarrow$ exponential degree distribution.
- $p(k,i) = [1+q(k,i)/M]/2i \quad \rightarrow$ scale-free degree distribution.

(Do własności strukturalnych sieci bezskalowych wrócimy przy okazji dyskusji zagadnień z socjofizyki).

[1] K. Malarz. "Random site percolation thresholds on square lattice for complex neighborhoods containing sites up to the sixth coordination zone". *Physica A* **632**.1 (2023), 129347.

[2] K. Malarz. "Universality of percolation thresholds for two-dimensional complex non-compact neighborhoods". *Physical Review E* **109**.3 (2024), 034108.

[3] K. Malarz. "Percolation thresholds on triangular lattice for neighbourhoods containing sites up to the fifth coordination zone". *Physical Review E* **103**.5 (2021), 052107.

[4] K. Malarz. "Random site percolation on honeycomb lattices with complex neighborhoods". *Chaos* **32**.8 (2022), 083123.

[5] Wikipedia. *Euclidean tilings by convex regular polygons*.

[6] K. Malarz and K. Kułakowski. “Matrix representation of evolving networks”. *Acta Physica Polonica B* **36**.8 (2005), 2523–2536.