Eötvös Loránd Tudományegyetem

Informatikai Kar

Algoritmusok és Alkalmazásaik Tanszék

Sejtautomata szabályok klaszterezése az általuk generált textúrák alapján

Témavezető:
Tichler Krisztián
adjunktus, PhD

Szerző:
Wolosz András
programtervező informatikus, MSc

Budapest, 2024

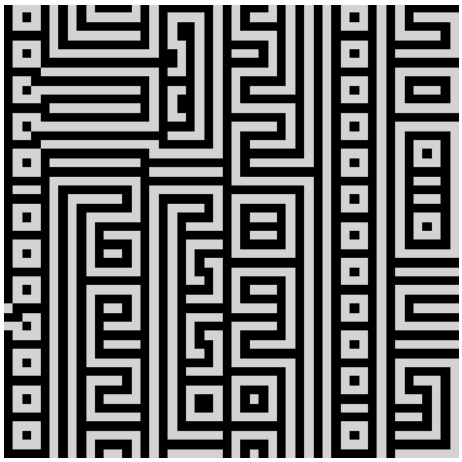
Tartalomjegyzék

1. fejezet

Bevezetés

Stephen Wolfram 1983-ban az elemi sejtautomaták szabályait négy különböző csoportba osztotta az alapján, hogy véletlenszerű kezdőkonfigurációból kiindulva milyen dinamikus viselkedés figyelhető meg a működésük során [1].

A diplomamunka célja egy ehhez hasonló csoportosítási módszer felállítása, amelyben a sejtautomaták dinamikus viselkedés helyett valamilyen egyéb tulajdonság – például vizuális megjelenés – szerint kerülnek csoportosításra.
A bemutatott módszer generikus, tehát különböző sejtautomaták vizsgálatára alkalmas. Ennek megfelelően a módszer teret ad arra, hogy a csoportosítás lépéseinek elvégzéséhez különböző informatikai eszközöket lehessen használni. A diplomamunka részét képezi egy konkrét sejtautomata analízise, konkrét eszközökkel, azonban bemutatásra kerülnek az alternatívan választható informatikai eszközök is, így prezentálva a módszer rugalmasságát.



1.1. ábra. `"0010000111101001"` 2D Wolfram kódú inkonzisztens szabály $T_{50,50}$-en

2. fejezet

CARCT

2.1. Sejtautomaták

1. Definíció. A végtelen sejtautomata egy végtelen hálózat, amelynek minden csúcspontjában egy-egy véges automata működik. A végtelen sejtautomaták:

- Egységesek: a hálózat csúcspontjaiban elhelyezkedő véges automaták azonos módon működnek.
- Szinkron módon működnek: minden csúcspontban található véges automata azonos időegységenként (iterációnként), egyszerre változtatja meg az állapotát, párhuzamosan.
- Lokálisak: minden véges automata a lokális környezetétől függően változtatja meg az állapotát [2].

A végtelen sejtautomata hálózata jellemzően valamilyen n-dimenziós kockarács, de bármilyen más rács is lehet, amelynek a csúcspontjaiban egységesen tudnak működni a véges automaták, például hatszög alapú rács.

A végtelen sejtautomaták vizsgálatát megnehezíti, hogy végtelen sok csúcspontja van az adott hálózatnak, ezért vezessük be a sejtautomata fogalmát.

2. Definíció. A sejtautomata a végtelen sejtautomatával analóg, azonban véges hálózaton helyezkedik el.

A legtermészetesebb módja egy sejtautomata megvalósításának, ha valamilyen tóruszszerű, önmagába záródó rácson helyezkedik el.

3. Definíció. A sejt a végtelen sejtautomata egy konkrét csúcspontja, illetve az ahhoz tartozó véges automata összefoglalóan [1].

4. Definíció. A sejtautomata szabályok azt határozzák meg, hogy egy sejtautomata sejtjeiben a véges automaták milyen lokális környezethez milyen állapotot rendelnek a sejthez a következő iterációban [1].

Megjegyzés: Ha egy sejt lokális környezete M darab különböző sejtet jelöl ki, és egy sejtnek N különböző állapota lehet, akkor N^M különböző szomszédság létezik. Ezekhez a szomszédságokhoz N állapot rendelhető, így összesen N^{N^M} szabály létezik. [3]

5. Definíció. A sejtautomata állapota egy adott t időpillanatban a sejtautomata sejtjeinek összessége [3].

A sejtautomaták végtelen időegységen keresztül iterálnak. Ez megnehezíti az általános vizsgálatukat, ezért vezessük be a textúra definícióját.

6. Definíció. A textúra vagy végállapot egy sejtautomata olyan állapota, amely után nem vizsgáljuk tovább a sejtautomata iterációit.

A továbbiakban a sejtautomaták textúráival foglalkozunk, vagyis azokkal az állapotokkal, amelyek a vizsgálat során számunkra releváns információkat tartalmaznak, és nem szükséges további iterációk kiszámítását elvégezni a következtetések levonásához.

A különböző sejtautomaták minden esetben megadhatóak matematikai leírással, működésük determinisztikus. Számítási modellként is felhasználhatók, valamint gyakorlati módon alkalmazhatók fizikai, biológiai vagy kémiai folyamatok szimulálására [1].

2.2. A CARCT általános bemutatása

Vezessük be a CARCT (Cellular Automaton Rules Clastering by Textures) generikus módszert, amelynek segítségével sejtautomaták szabályait lehet klaszterezni az általuk generált textúrák alapján. A módszer generikus, tehát különböző, szabadon választható algoritmusok alkalmazhatók a felhasználás során, valamint a konstansai is önkényesen beállíthatók. Mivel sejtautomatákból számtalan létezik különböző rácsokon, különböző állapotokkal és egyéb különböző sajátosságokkal [1], így az őket csoportosító módszernek is általánosnak kell lennie.

2.3. Pszeudokód

1. algoritmus CARCT

```
Info grouping = symmetricalRulesGrouping(rules)
rules = narrowRules(grouping)
List<Map<int, Features» consistents
for  int ruleIndex = 0; ruleIndex < rules.size(); ruleIndex++ do
   Array<State, numberOfRuleIteration> candidateStates
   int candidateStatesCounter = 0
   for  int ruleIterationIndex = 0;
      ruleIterationIndex < numberOfRuleIteration;
      ruleIterationIndex++  do
      State currentState = Automata::init()
      int persistanceCounter = 0
      int updateIndex = 0
      while updateIndex != numberOfUpdates do
         State oldState = currentState
         Automata::update(currentState, rules[ruleIndex])
         float similarityValue = similarityMetric(
                                 oldState, currentState, ...)
         if similarityValue > similarityThreshold then
            persistanceCounter++
            if persistanceCounter == persistanceLength then
               candidateStates[candidateStatesCounter] =
                              currentState
               candidateStatesCounter++
               break
            end if
         else
            persistanceCounter = 0
         end if
         updateIndex++
      end while
      if candidateStatesCounter == numberOfRuleIteration
      and isConsistent(candidateStates) then
         for  int ruleIterationIndex = 0;
      ruleIterationIndex < numberOfRuleIteration;
            ruleIterationIndex++  do
            Map<int, Features> consistent
            consistent[ruleIndex] = getFeatures(
                           candidateStates[ruleIterationIndex], ...)
            consistents.append(consistent)
         end for
      end if
   end for
   clusters = cluster(consistents, ...)
   clusters = regroup(clusters, grouping)
end for
```

2.4. A CARCT elemei

A CARCT pszeudokódjában piros színű kiemeléssel jelöljük azokat a függvényeket, amelyek generikusak, tehát minden esetben sajátos megvalósítást igényelnek. Kék színnel jelöljük a szabadon megválasztható paramétereket. Tekintsük meg ezeket részletesen.

2.4.1. Generikus függvények

- `symmetricalRulesGrouping(rules)`

  Bemenet: összes sejtautomata szabály.

  Kimenet: összevont sejtautomata szabályok halmazai.

  Működési elv: a létrehozott szabálycsoportok olyan szabályokat tartalmaznak, amelyek valamilyen logikai kapcsolat mentén összetartoznak. Jellemzően szimmetrikus szabályok együttesei. Minden szabály pontosan egy szabálycsoportban szerepel. Minden szabálycsoport legalább egy szabályt tartalmaz.

- `narrowRules(grouping)`

  Bemenet: sejtautomata szabályok halmazai.

  Kimenet: sejtautomata szabályok listája.

  Működési elv: a szabálycsoportokból valamilyen módon ki kell választani egy elemet. Ez lehet akár egy random elem is, vagy a szabálycsoport első eleme. Ha a szabálycsoportok megfelelően vannak megalkotva, akkor a kiválasztás folyamata drasztikus módon nem befolyásolhatja a klaszterezés végeredményét.

- `Automata::init()`

  Kimenet: sejtautomata állapot

  Működési elv: a sejtautomata hálózatának csomópontjaiban található véges automaták $t = 0$ időpillanatban vett állapotainak beállítása. Valamilyen random algoritmussal kell inicializálni. Ez jellemzően fehérzaj, de lehet Perlin-noise, vagy más random zajgenerátor.

- `Automata::update(currentState, rules[ruleIndex])`

  Bemenet:

  1. sejtautomata állapot
  2. sejtautomata szabály

Működési elv: szinkron módon módosítja a bemenetben megadott sejtautomata állapotot, a megadott sejtautomata szabály szerint.

- `similarityMetric(oldState, currentState, ...)`
  Bemenet:

  1. sejtautomata állapot
  2. sejtautomata állapot
  3. egyéb, az adott algoritmushoz szükséges paraméterek

  Kimenet: lebegőpontos szám.
  Működési elv: a két megadott sejtautomata állapothoz rendel egy lebegőpontos számot, ami valamilyen módon azt fejezi ki, hogy a két állapot mennyire hasonlít egymásra (vagy különbözik egymástól). Ennek különböző megvalósításai lehetnek, de minden esetben valamilyen kvantitatív összehasonlító metrikát kell alkalmazni.

- `isConsistent(candidateStates)`
  Bemenet: sejtautomata állapotok listája.
  Kimenet: bináris.
  Működési elv: a megadott sejtautomata állapotokat hasonlítja össze egymással. A `similarityMetric(oldState, currentState, ...)` algoritmusban alkalmazott metrikák alkalmazhatók itt is. Szabadon megközelíthető, hogy milyen feltétel mellett fogadjuk el több sejtautomata állapot hasonlóságát.

- `getFeatures(candidateStates[ruleIterationIndex], ...)`
  Bemenet:

  1. textúra.
  2. egyéb, az adott algoritmushoz szükséges paraméterek.

  Kimenet: tulajdonságok.
  Működési elv: a megadott textúrából nyer ki valamilyen tulajdonságokat, amelyek alkalmasak lesznek a klaszterezés elvégzéséhez.

- `cluster(consistents, ...)`

  Bemenet: szabály-tulajdonság kulcs-érték párosok.

  Kimenet: klaszterek.

  Működési elv: a megadott kulcs-érték párok alapján elvégzi a sejtautomata szabályok klaszterezését az általuk generált textúrák alapján. A klaszterek száma legfeljebb a kulcs-érték párok száma, és legalább egy.

- `regroup(clusters, grouping)`

  Bemenet:

  1. klaszterek.
  2. összevont sejtautomata szabályok halmazai.

  Kimenet: klaszterek.

  Működési elv: Az összevont sejtautomata szabályok halmazai alapján kibővíti a klaszterek taratalmát. A reprezentáns mellé elhelyezi a vele egy halmazba eső többi sejtautomata szabályt.

2.4.2. Konstansok

- `numberOfRuleIteration`: Azt határozza meg, hogy egy adott szabályt hányszor inicializálunk és folytatunk rajta iterálást annak érdekében, hogy ellenőrizzük, többszörösen is hasonló textúrákat generál-e.

  Feltételek: `numberOfRuleIteration` ≥ 2, `numberOfRuleIteration` $\in \mathbb{N}$

- `numberOfUpdates`: Azt határozza meg, hogy egy adott szabályt legfeljebb hány iteráción keresztül vizsgálunk.

  Feltételek: `numberOfUpdates` ≥ 1, `numberOfUpdates` $\in \mathbb{N}$

- `persistanceLength`: Azt határozza meg, hogy egy adott szabály hány egymást követő iteráción keresztül kell, hogy alig változó sejtautomata állapotokat vegyen fel.

  Feltételek: `persistanceLength` ≥ 1, `persistanceLength` $\in \mathbb{N}$

- `similarityThreshold`: Azt határozza meg, hogy a sejtautomata állapotokat összehasonlító algoritmus kimenete milyen küszöbérték felett (vagy alatt) jelent hasonlóságot.

  Feltételek: A sejtautomata állapotokat összehasonlító algoritmustól függenek.

2.5. A CARCT működése

2.5.1. Szimmetrikus szabályok összevonása

A `symmetricalRulesGrouping(rules)` függvény a szabályok halmazából alkot szabálycsoportokat. Különböző sejtautomatáknál előfrodulhat, hogy valamely szabályok a generált textúra szempontjából lezártat alkotnak. Ebben az esetben a szabályok valamilyen módon megfeleltethetőek egymásnak, így ezeket az analízis során egységesen kezelhetjük. Ezeket a szabályokat érdemes összevonni, ezzel a módszer ezt követő lépései leegyszerűsödhetnek, a végeredmény tisztább eredményt alkot majd, valamint számítási erőforrások szabadulhatnak fel.

A `narrowRules(grouping)` függvény a szabálycsoportokból valamilyen módon kiválaszt egy-egy konkrét szabályt. Ezek a kiválasztott szabályok reprezentálják majd a csoport többi tagját is a klaszterezés során. A klaszterezés lefutása után a `regroup(clusters, grouping)` függvény helyezi be a megfelelő klaszterekbe a reprezentáns mellé a csoport többi elemét.

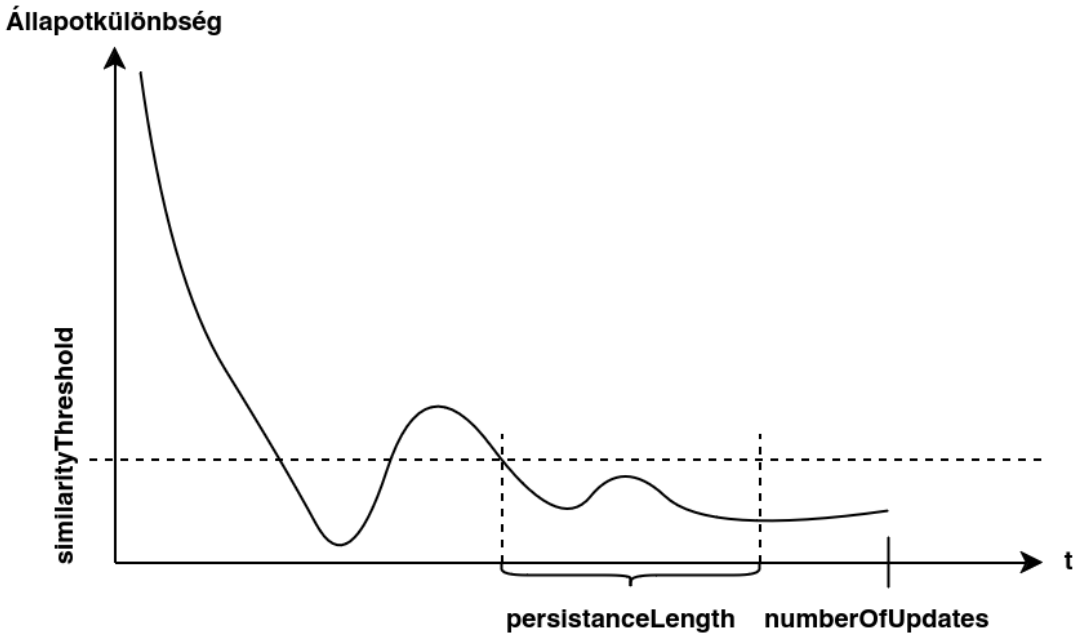
2.5.2. Konzisztens szabályok meghatározása

A reprezentáns szabályokat a következő módon vizsgáljuk `numberOfRuleIteration`, `numberOfUpdates`, `persistanceLength`, `similarityThreshold` paraméterek felett.

1. Az `Automata::init()` függvény segítségével inicializáljuk a sejtautomatát valamilyen random zajjal.

2. A sejtautomata minden állapotváltozása (`Automata::update(currentState, rules[ruleIndex])`) után vizsgáljuk meg, hogy mekkora a kvantitatív különbség a jelenlegi és az azt megelőző sejtautomata állapot között. Ennek a különbségnek a meghatározására alkalmazzuk a `similarityMetric(oldState, currentState, ...)` függvényt.

3. Ha legfeljebb `numberOfUpdates` állapotváltozás alatt megállapíthatjuk, hogy az automata az utolsó `persistanceLength` állapotában `similarityThreshold` küszöbérték alatt maradt, akkor elfogadjuk a szabályt. Egyébként a szabály inkonzisztens, a továbbiakban nem vizsgáljuk azt.



2.1. ábra. Szabály elfogadása

4. Ha elfogadtuk a szabályt, mentsük el az utolsó megvizsgált sejtautomata állapotot (textúrát) és ismételjük meg a vizsgálatot új kezdőállapotból.

5. Ha `numberOfRuleIteration`-szor elfogadtuk a szabályt, akkor tartsuk meg a szabályt. Ha nem, akkor vessük el azt.

6. Vizsgáljuk meg textúraanalízissel az elfogadott szabály `numberOfRuleIteration` darab elmentett textúráját. Ha kvantitaív módon hasonlóknak bizonyulnak (`isConsistent(candidateStates)`), nevezzük a szabályt konzisztensnek és tartsuk meg. Ha nem, akkor a szabály inkonzisztens és a továbbiakban nem vizsgáljuk.

Vegyük észre, hogy egy szabály inkonzisztens, ha

- az nem képes kvantitatív módon alig változó (stabil) textúrát generálni (3. lépés), vagy
- ha képes kvantitatív módon alig változó, stabil textúrát generálni, de nem képes erre többszörösen (5. lépés), vagy
- ha képes kvantitatív módon alig változó, stabil textúrát generálni többszörösen, de azok nem hasonlítanak egymáshoz kvantitatív módon (6. lépés).

Ezeket az inkonzisztens szabályokat nem vizsgáljuk, a klaszterezés során nem kerülnek feldolgozásra, a CARCT módszer nem nyilatkozik róluk.

2.5.3. Klaszterezés

7. Definíció. A klaszterezés az a folyamat, amely során N darab d-dimenziós objektumot K darab csoportba (klaszterbe) osztunk az őket jellemző tulajdonságok alapján, úgy, hogy az egy klaszterbe eső objektumok egymáshoz hasonlóak, és a külön klaszterbe eső objektumok egymástól különbözőek. Az elosztás során minden objektum pontosan egy klaszterbe kerül. $K > 0$, $K \in \mathbb{N}$ [4].

A konzisztens szabályok által generált textúrákból kinyert tulajdonságok a `getFeatures(candidateStates[ruleIterationIndex], ...)` függvény segítségével kerülnek feldolgozásra. Ezt a függvényt úgy kell megválasztani, hogy a textúrákból releváns információkkal szolgáljon.

Ezeket a tulajdonságokat kell felhasználni bemeneti adatként a `cluster(consistents, ...)` függvényben. Ennek a függvénynek kell a tulajdonságok alapján valamilyen algoritmus alapján csoportosítani a szabályokat.

3. fejezet

A CARCT alkalmazása

Alkalmazzuk a CARCT módszert egy konkrét sejtautomatára. Ehhez szükséges a generikus algoritmusokat konkrét algoritmusokkal helyettesíteni, valamint a konstansokat megfelelően beállítani. Mindenekelőtt azonban a kiválaszott sejtautomatát kell precízen definiálni.

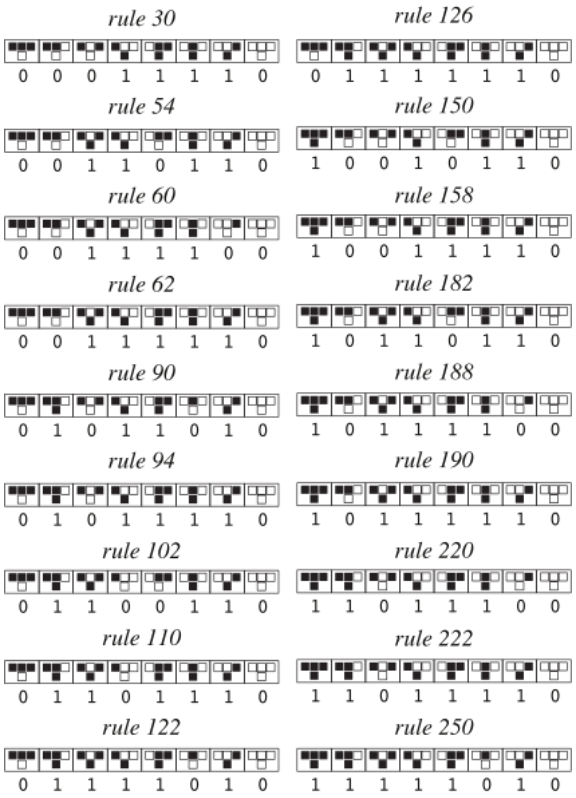
A CARCT alkalmazásának a bemutatásához definiáljuk az elemi kétdimenziós sejtautomatát. Ez a sejtautomata elemi sejtautomata kiterjesztése két dimenzióra.

3.1. Elemi sejtautomata ismertetése

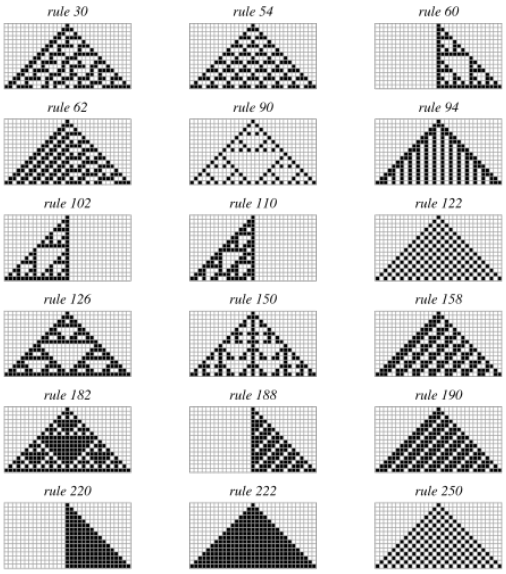
Az elemi sejtautomaták a legismertebb sejtautomaták. Egydimenziósak, tehát szalagszerű hálózaton működnek. Gyakran vizualizálják őket két dimenzióban, ahol az egyik tengely a hálózat (szalag) aktuális állapota, a másik tengely pedig a különböző t időpillanatokat reprezentálja [1].

Az elemi sejtautomaták sejtjei binárisan két különböző állapotot vehetnek fel. Egy sejt állapotát minden $t = n+1$ időpillanatban a $t = n$ időpillanatban vett saját állapota, valamint a két szomszédjának állapota határozza meg. Így összesen 2^3 szomszédság létezik, amelyekhez egyenként két fajta állapot rendelhető. Ez összesen $2^{2^3} = 256$ különböző szabályt határoz meg az elemi sejtautomatához [3].

Ezeket a szabályokat a Wolfram kóddal azonosítjuk, amely minden szabályhoz egy egész számot rendel 0-tól 255-ig [5].



3.1. ábra. Néhány elemi sejtautomata szabály definíciójának vizuális reprezentációja



3.2. ábra. Néhány elemi sejtautomata különböző szabályainak kétdimenziós vizuális reprezentációja, egy elütő sejt inicializációval[1]

[1]mathworld.wolfram.com [Hozzáférés: 2024. május 20.]

3.2. Elemi kétdimenziós sejtautomata

3.2.1. Hálózat

Jelölje C_n az n csúcsú körgráfot.

8. Definíció. A körgráf egy olyan gráf, amely pontosan egy körből áll [6].

Jelölje $G\square H$ G és H gráfok Descartes-szorzatát.

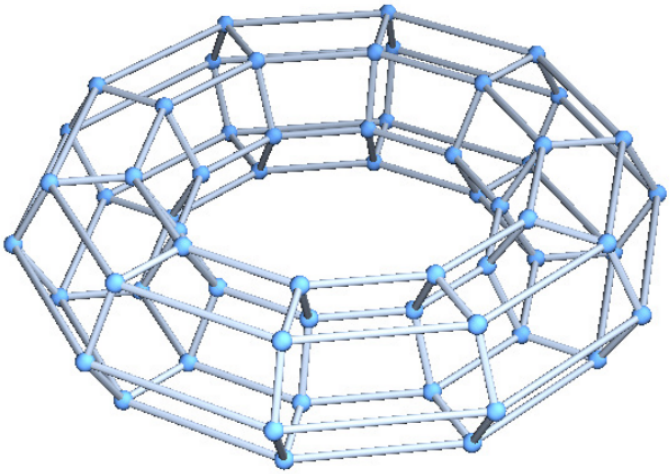
9. Definíció. G és H gráfok Descartes-szorzata egy olyan gráf, amelyre igaz, hogy

- $G\square H$ csúcshalmaza $V(G) \times V(H)$ Descartes-szorzata.
- (u, v) és (u', v') csak akkor szomszédosak $G\square H$ gráfban, ha
  - $u = u'$ és v szomszédos v'-vel H-ban, vagy ha
  - $v = v'$ és u szomszédos u'-vel G-ben [7].

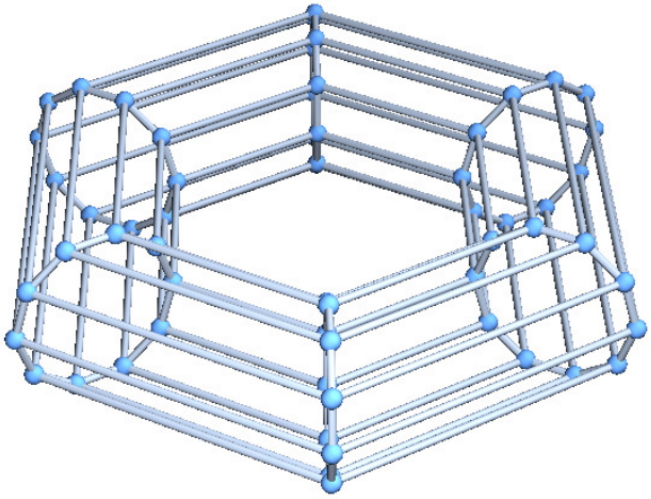
Megjegyzés: $G\square H$ kommutatív és asszociatív [7].
Jelölje $T_{m,n}$ a C_m és C_n körgráfokból alkotott tóruszrácsgráfot.

10. Definíció. $T_{m,n} = C_m\square C_n$ [8]

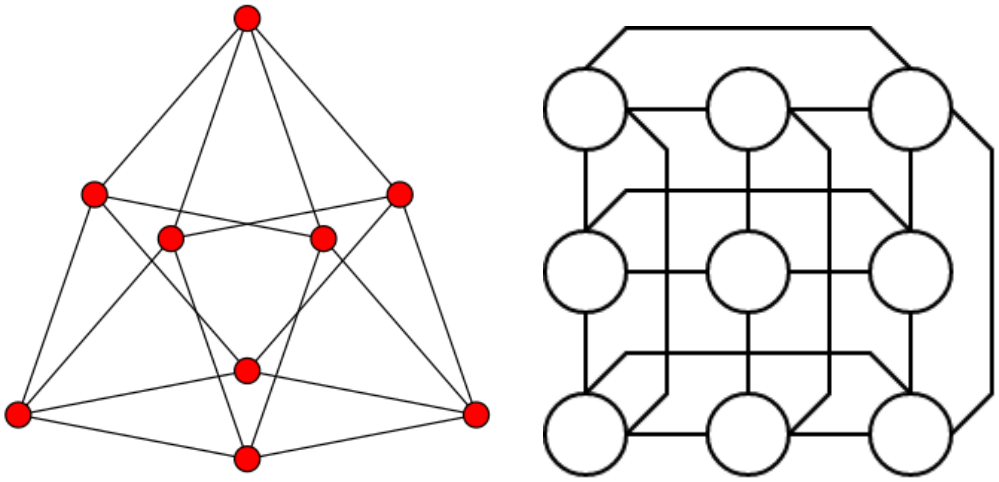
Megjegyzés: $T_{m,n}$ minden esetben 4-reguláris [8].



3.3. ábra. $T_{10,6}$ háromdimenziós vizuális reprezentációja



3.4. ábra. $T_{6,10}$ háromdimenziós vizuális reprezentációja



(a) Szabályos elrendezés[2] (b) Négyzetrácsszerű elrendezés

3.5. ábra. $T_{3,3}$ kétdimenziós vizuális reprezentációja

Az elemi kétdimenziós sejtautomata hálózatát $T_{32,32}$ alkotja.

3.2.2. Állapotok

Elemi kétdimenziós sejtautomata egy sejtje két különböző (bináris) állapotot vehet fel.

[2]mathworld.wolfram.com [Hozzáférés: 2024. május 20.]

3.2.3. Lokális környezet

Címkézés

Az elemi kétdimenziós sejtautomata sejtjeinek száma: $V(T_{32,32}) = 32*32 = 1024$. Minden csúcsot címkézzük meg egy egyedi (x, y) pár azonosítóval. $0 \leq x \leq 31, 0 \leq y \leq 31$.
Bővítsük ki a csúcsok címkézését. Az adott címkéhez tartozó csúcsokra a továbbiakban hivatkozhassunk a $(x * (k * 32), y * (k' * 32))$ címkékkel is. $k, k' \in \mathbb{N}$
A tóruszrácsgráf 4-reguláris [8]. Adott (x', y') csúcsot az alábbi csúcsokkal kapcsolja össze él, modulo 32:

- $(x' + 1, y')$
- $(x' - 1, y')$
- $(x', y' + 1)$
- $(x', y' - 1)$

S sejt lokális környezete

Jelölje (x'', y'') $T_{32,32}$ egy S sejtjét. Legyen S lokális környezete:

- $(x'' + 1, y'')$
- $(x'' - 1, y'')$
- $(x'', y'' + 1)$
- $(x'', y'' - 1)$

Megjegyzés: S nem alkotja a saját lokális környezetét.

3.2.4. Vizuális reprezentáció

A $T_{32,32}$ tóruszrácsgráfot, és ennek alapján az elemi kétdimenziós sejtautomata állapotait, textúráit a továbbiakban az alábbi konvenciók szerint jelenítjük meg:

- A 3.5b ábrán látható módon a csúcspontokat raszteresen, egy négyzetrácson jelenítjük meg.
- A csúcspontokban található sejtek formája négyzet.
- A csúcspontokban található sejtek állapota fekete vagy szürke.
- Az csúcsok egymáshoz vonatkoztatott helyzete címkézés szerint következetesen történik. Az (x, y) párral címkézett sejttől
  - keletre az $(x + 1, y)$;
  - délre az $(x, y + 1)$;
  - nyugatra az $(x - 1, y)$;
  - északra az $(x, y - 1)$ párral címkézett sejt jelenik meg.

3.2.5. Szabályok

Elemi kétdimenziós sejtautomata szabályainak száma

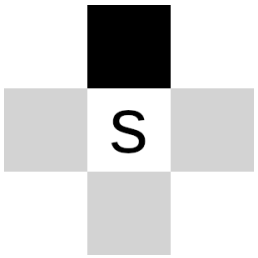
Az elemi kétdimenziós sejtautomata sejtjei két állapotot vehetnek fel. Egy S sejt lokális környezetét négy másik sejt alkotja, így egy sejtnek összesen 2^4 különböző lokális környezete lehet. Minden lokális környezet kétféle állapot közül rendel egyet az adott sejthez, így összesen $2^{2^4} = 65536$ különböző szabály létezik az elemi kétdimenziós sejtautomatához [3].

Lokális környezet kódolása

Vezessünk be egy kódolást a lokális környezet leírására. Egy (x, y) koordinátával címkézett S sejt négy szomszédját négy biten tároljuk, az alábbi táblázat szerint, nullától indexelve.

j. **bit**	**Lokális környezet eleme**
0.	$(x, y - 1)$ – észak
1.	$(x + 1, y)$ – kelet
2.	$(x, y + 1)$ – dél
3.	$(x - 1, y)$ – nyugat

3.1. táblázat. Lokális környezet kódolása



3.6. ábra. "1000" lokális környezet

2D Wolfram kód

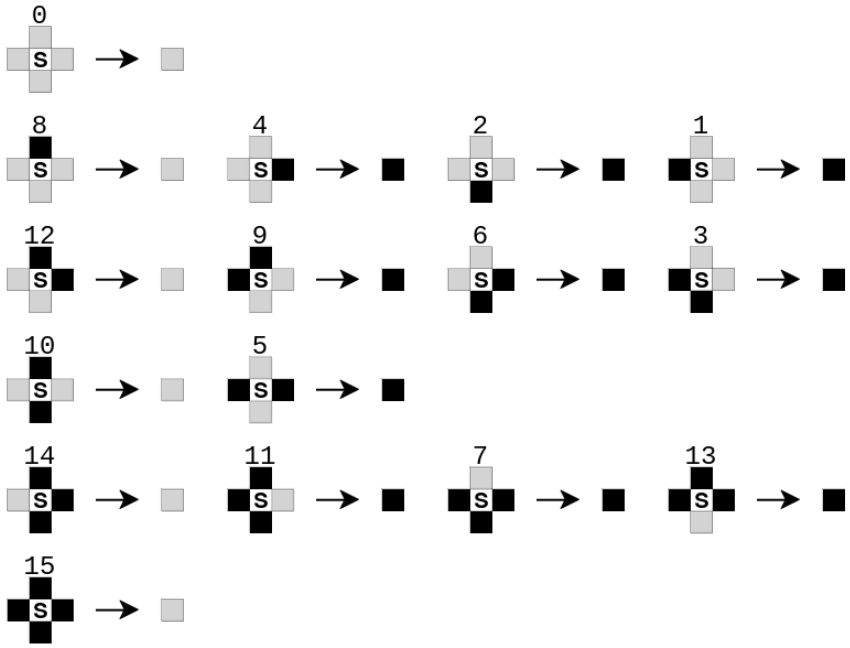
Vezessünk be egy, az elemi sejtautomaták Wolfram kódjához[5] hasonló leírást az elemi kétdimenziós sejtautomata különböző szabályainak tömör, jellemző megnevezésére. Nevezzük ezt 2D Wolfram kódnak.

Mivel 2^4 különböző lokális környezet lehetséges, ezért legyen minden 2D Wolfram kód $2^4 = 16$ bit hosszú. A kód i. bitje azt határozza meg, hogy melyik lokális környezethez rendelünk sejtállapotot. A lokális környezet binárisról decimálisra váltott értéke jelöli, hogy melyik i. bithez tartozik, nullától indexelve.

A kód i. bitjében felvett érték jelöli, hogy milyen sejtállapotot rendelünk az i. bitben meghatározott lokális környezethez.

i. **bit**	**Lokális környezet kódja**	**Hozzárendelt sejtállapot**
0.	"0000"	0
1.	"0001"	1
2.	"0010"	1
3.	"0011"	1
4.	"0100"	1
5.	"0101"	1
6.	"0110"	1
7.	"0111"	1
8.	"1000"	0
9.	"1001"	1
10.	"1010"	0
11.	"1011"	1
12.	"1100"	0
13.	"1101"	1
14.	"1110"	0
15.	"1111"	0

3.2. táblázat. "0111111101010100" 2D Wolfram kód



3.7. ábra. `"0111111101010100"` 2D Wolfram kód vizuálisan reprezentálva

3.3. CARCT módszer paramétereinek megadása elemi kétdimenziós sejtautomata textúráinak klaszterezéséhez

Az elemi kétdimenziós sejtautomata jellemzése és működésének leírása után elkezdődhet a CARCT módszer alkalmazása. Ehhez szükséges a konstansok beállítása, és a generikus algoritmusok megadása.

3.3.1. Konstansok

- `numberOfRuleIteration`: 10
- `numberOfUpdates`: 200
- `persistanceLength`: 50
- `similarityThreshold`: 0.2

3.3.2. Szimmetrikus szabályok meghatározása

Definiáljuk egy szabály transzformációit az alábbi módon.

11. Definíció. *A* szabály tengelyes tükrözése azon *B* szabály, amely 2D Wolfram kódja *A* 2D Wolfram kódjának a következő bitjeit cseréli fel:

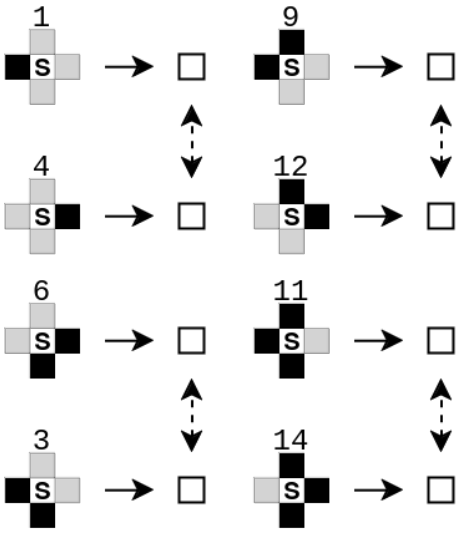
- 1. $\longleftrightarrow$ 4.
- 9. $\longleftrightarrow$ 12.
- 6. $\longleftrightarrow$ 3.
- 11. $\longleftrightarrow$ 14.

12. Definíció. *A* szabály 90°-os elforgatása azon *B* szabály, amely 2D Wolfram kódja *A* 2D Wolfram kódjának a következő bitjeit mozgatja a megadott bitekre:

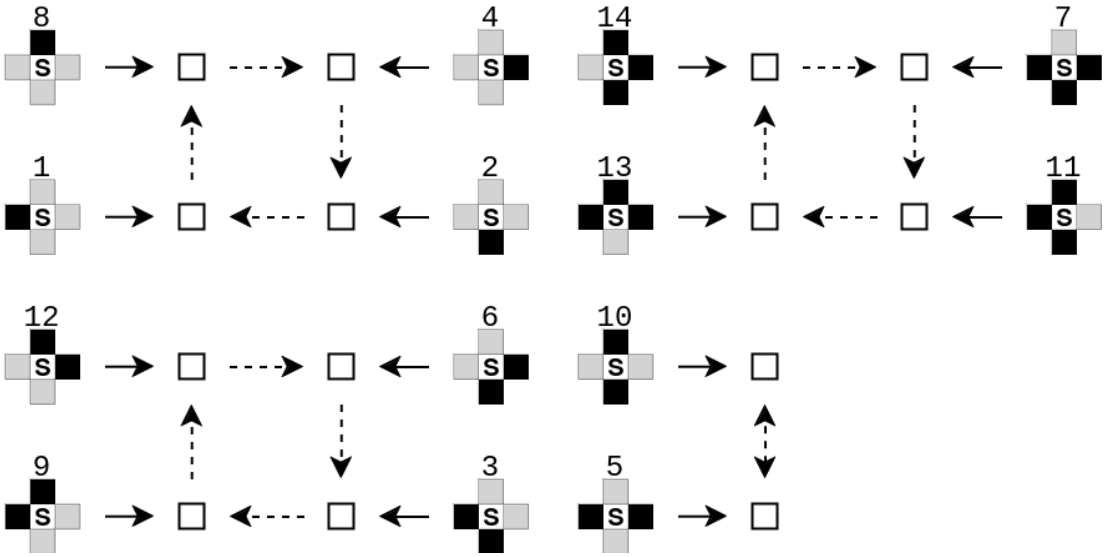
- 8. $\longrightarrow$ 4.
- 4. $\longrightarrow$ 2.
- 2. $\longrightarrow$ 1.
- 1. $\longrightarrow$ 8.
- 12. $\longrightarrow$ 6.
- 6. $\longrightarrow$ 3.
- 3. $\longrightarrow$ 9.
- 9. $\longrightarrow$ 12.
- 10. $\longleftrightarrow$ 5.
- 14. $\longrightarrow$ 7.
- 7. $\longrightarrow$ 11.
- 11. $\longrightarrow$ 13.
- 13. $\longrightarrow$ 14.

13. Definíció. *A* szabály invertálása azon *B* szabály, amely 2D Wolfram kódja *A* 2D Wolfram kódjának az összes bitjét negálja.

Ezek a definíciók természetesen következnek a vizuális reprezentációból (3.2.4). Az alábbi ábrákon az üres négyzetek az adott (akár 0, akár 1 értékű) sejtállapotot jelölik, a szaggatott vonalas nyilak pedig a bit elmozdulását ábrázolják.



3.8. ábra. Elemi kétdimenziós sejtautomata szabály tengelyes tükrözése

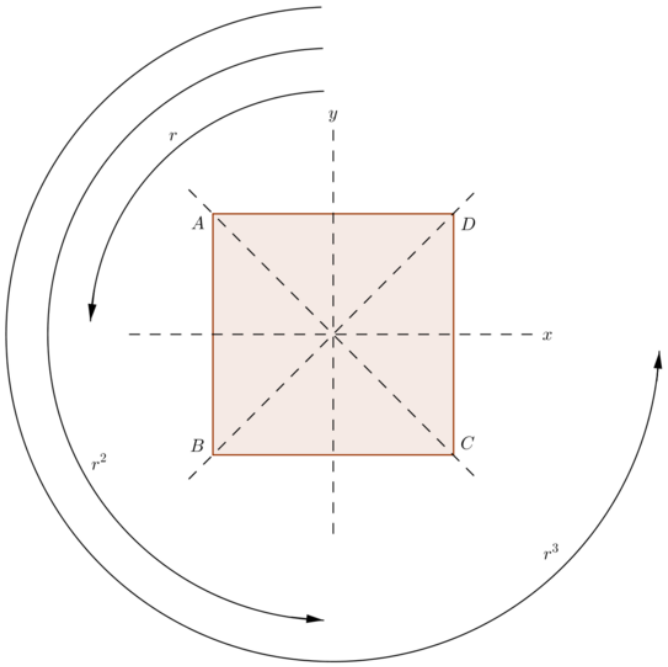


3.9. ábra. Elemi kétdimenziós sejtautomata szabály 90°-os elforgatása

Csoportelméletben egy $ABCD$ négyzet szimmetriáit D_4-el jelöljük [9], ezek a következők:

- e identitás,
- r, r^2, r^3 rendre $90°, 180°, 270°$ forgatások,
- $t_x = s, t_y$ rendre horizontális és vertikális tükrözések,
- t_{AC} AC csúcsok közötti tükrözés,
- t_{BD} BD csúcsok közötti tükrözés.

Ezen szimmetriák felírhatók a következő módon: e, r, r^2, r^3, s, rs, r^2s, r^3s [10].



3.10. ábra. ${D_4}^3$

r forgatás megfeleltethető az általunk definált forgatással, s tükrözés megfeleltethető az általunk definiált tükrözéssel. Az általunk definiált invertálás miatt egészítsük ki a transzformációkat a következő módon: $e, r, r^2, r^3, s, rs, r^2s, r^3s, i, ir, ir^2, ir^3, is, irs, ir^2s, ir^3s$.

A szimmetrikus szabályok csoportjai azok a halmazok legyenek, amelyek a szimmetria transzformációkra lezártat alkotnak.
Például a `"0100000000000000"` és a `"1111011111111111"` szabályok egy csoportba kerülnek, mert:

[3]proofwiki.org [Hozzáférés: 2024. május 20.]

1. `"0100000000000000"` $r \longrightarrow$
2. `"0000000010000000"` $r \longrightarrow$
3. `"0000100000000000"` $i \longrightarrow$
4. `"1111011111111111"`

A `"0000000000000000"` és az `"1000000000000000"` szabályok azonban külön csoportokba kerülnek, mert nem létezik a szimmetria transzformációknak olyan eleme, amellyel áttranszformálhatók lennének egymásba.

Így a `symmetricalRulesGrouping(rules)` a szabályok lényegileg különböző csoportjait hozza létre. Az eredeti $2^{2^4} = 65536$ szabály így 4856 csoportba lett osztva. A $SSIM_T$ (3.3.6) és a $GLCM_T$ (3.3.8) algoritmusok számításigényesek, ezért fontos, hogy ez az optimalizáció több mint tizedrészére csökkenti a megvizsgálandó szabályok halmazát.

3.3.3. Reprezentáns szabályok kiválasztása

Minden szabálycsoportból válasszuk ki azt a szabályt reprezentánsnak, amelyik 2D Wolfram kódja a binárisan vett legkisebb számként értelmezhető. Alternatíva lehetne a csoportból véletlenszerűen választani, vagy azt az elemet, amelynek a 2D Wolfram kódja egy adott állapotot (0 vagy 1) a legtöbb biten ábrázolja.

3.3.4. Automata inicializációja

Az elemi kétdimenziós sejtautomata $t = 0$ időpillanatban vett kezdőállapotát valamilyen pszeudo random algoritmussal generáljuk. A pszeudo random algoritmusok kevés erőforrást igényelnek, így hatékonyan alkalmazhatóak zaj generálására [11].

Alternatívaként alkalmazható Perlin-noise, vagy Simplex-noise algoritmusok is [12].

3.3.5. Automata interációja

Az elemi kétdiemnziós automata ismeretében triviális.

3.3.6. Sejtautomata állapotok kvantitatív összahasonlítása

A 3.2.4 részben leírtak szerint a gráfot raszteres, pixelgrafika-szerű módon ábrázoljuk, így felületesen akár képekként tekinthetünk az elemi kétdimenziós sejtautomata állapotaira.
Az $SSIM$ (Structural Similarity Index Measure) metrika szignálok struktúrájának az összehasonlítására lett kifejlesztve. Modellezi a biológia észlelést, így vizuális és audio tartalmak összevetésére alkalmas. Ezen kívül alkalmazzák képminőség vizsgálatára is [13]. Tekintsünk $SSIM$-re mint szürkeárnyalatos képszignálokat feldolgozó metrikára.

N darab pixelből álló x kép fényerőssége

$$\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i. \tag{3.1}$$

Szórás segítségével határozzuk meg x kép kontrasztját.

$$\sigma_x = \left(\frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \mu_x\right)^2\right). \tag{3.2}$$

Adjuk meg x és y képek kovarianciáját.

$$\sigma_{xy} = \frac{1}{N-1}\sum_{i=1}^{N}\left(x_i - \mu_i\right)\left(y_i - \mu_y\right). \tag{3.3}$$

A numerikus stabilitás érdekében határozzunk meg C_1 és C_2 konstansokat,

$$C_1 = (K_1 * L)^2 \tag{3.4}$$

$$C_2 = (K_2 * L)^2 \tag{3.5}$$

ahol $K_1 \ll 1, K_2 \ll 1$ és L a képen használható különböző szürkeárnyalatok számánál eggyel kevesebb.

Ekkor

$$SSIM(x,y) = \frac{\left(2\mu_x\mu_y + C_1\right)\left(2\sigma_{xy} + C_2\right)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(2\sigma_x^2 + \sigma_y^2 + C_2\right)} \tag{3.6}$$

x és y képekre [14].

Vezessük be $SSIM_T$-t, amely az $SSIM$ kiterjesztése tóruszrácsgráfokra.
A szignál intenzitásának és kontrasztjának kiszámítását nem kell módosítani, mert azok pontonként vizsgálják az adott szignált. A kovariancia azonban szignálpárok pontpárjaiként kerül kiszámításra. Habár a címkézés (3.2.3) mellett triviális, hogy mely pontok szomszédosak egymással (ahogy ez triviális képek esetében is), azonban a tóruszrácsgráfoknak nincs olyan egyértelműen meghatározható viszonyítási pontjuk, mint a képeknek, így nem egyértelmű, hogy mely pontpárokat kell egymáshoz rendelni.
Alkalmazzunk kereszt-korrelációt[15] ennek meghatározásához. Válasszunk ki egy-egy fix viszonyítási pontot mindkét tóruszrácsgráfon, majd számoljuk ki a kovariánst a fixpontok minden egymáshoz viszonyított lehetséges poziciójában. $SSIM_T$ kovarianciája legyen ezen értékek maximuma.

$$\sigma'_{xy} = \max_{O=1}^{N} \left(\frac{1}{N-1} \sum_{i=1}^{N} (x_i - \mu_i) \left((y_i + O) - \mu_y \right) \right). \tag{3.7}$$

Így

$$SSIM_T(x, y) = \frac{(2\mu_x\mu_y + C_1)\left(2\sigma'_{xy} + C_2\right)}{\left(\mu_x^2 + \mu_y^2 + C_1\right)\left(2\sigma_x^2 + \sigma_y^2 + C_2\right)}. \tag{3.8}$$

Mivel a különböző sejtállapotok száma 2, ezért $L = 1$. $N = 32 * 32$ a tóruszrácsgráf csúcsainak száma. Legyen

$$K_1 = 0.01,$$
$$K_2 = 0.03.$$

Az $SSIM$-nek létezik egy $MSSIM$ változata, amely egy kétdimenziós Gauss eloszlást alkalmazva (lényegében elmosva) rendel értéket a bemeneti képekhez. Részletesebb textúrákhoz jól alkalmazható lenne ez az algoritmus, mert nem érzékeny az elhanyagolható különbségekre. A $T_{32,32}$ azonban relatíve kicsi minta, amelynek az elmosása pont a lényegi különbségeket tünteté el a textúrákból [16].
További öszehasonlításra alkalmas metrika lehet az MSE (Mean squared error), illetve a $PSNR$ (Peak signal-to-noise ratio). Ezek széles körben elterjedt összehasonlító metrikák, amelyek implementálása egyszerű, és természetes fizikai jelentéssel bírnak, azonban az emberi észlelést nem veszik figyelembe, így a képekhez tartozó strukturális információt figyelmen kívül hagyják [13].

3.3.7. Konzisztens szabályok meghatározása

Páronként vessük össze az algoritmus bemenetének minden textúráját egymással $SSIM_T$ (3.3.6) metrika segítségével. Határozzunk meg egy konstans α értéket. Ha minden textúra pár $SSIM_T$ értéke nagyobb α értéknél, akkor konzisztensnek nevezhetjük a textúrákat generáló sejtautomata szabályt.
Legyen

$$\alpha = 0.09.$$

Minimum – vagy különbséget mérő a metrika esetén maximum – konstans meghatározása helyett alternatíva lehet egy konstans átlag meghatározása, amit a sejtautomata állapotpárok $SSIM_T$ értékének átlagosan kell meghaladnia.
Amennyiben sok sejtautomata állapotot kell összevetni (3.3.1), számítási erőforrásokat lehet csökkenteni minden pár összevetése helyett kevesebb pár összehasonlításával. Ezeket akár random módon is ki lehet választani.

3.3.8. Vizuális tulajdonságok kinyerése

A konzisztens szabályok alkotta textúrákat vizsgáljuk meg, és állítsunk elő belőlük a vizuális reprezenátció szempontjából (3.2.4) releváns információkat textúraanalízis segítségével. A textúraanalízis azon az ötleten alapszik, hogy egy bemeneti képet kisebb részekként, foltonként (angolul patch) kell vizsgálni [17].
Az információk kinyeréséhez alkalmazzunk másodfokú $GLCM$ (Gray-Level Co-Occurrence Matrix) mátrixot. Az n. fok azt jelöli, hogy a textúraanalízis során egyszerre az eredeti kép hány különböző pixele között vizsgálunk összefüggéseket. A (p_1, p_2) pixel pár első komponensére referenciaként, a második komponensére szomszédként hivatkozunk. p_1 és p_2 pixel között mindig d_x pixel távolság van horizontálisan, és d_y pixel távolság van vertikálisan. $p_1 \neq p_2$, így $d_x \neq d_y \neq 0$ [17].
Megjegyzés: mivel másodfokú a $GLCM$, ezért $d_x \leq x$ és $d_y \leq y$, ahol x és y rendre a bemeneti kép szélessége és hosszúsága.
A $GLCM$ mátrix $m*m$-es, ahol m a pixelek által felvehető szürkeárnyalatok száma. $GLCM$ mátrix egy adott eleme $GLCM_{i,j}$, ami a bemeneti képen i szürkeárnyalatú pixelektől d_x horizontális és d_y vertikális távolságra lévő j szürkeárnyalatú pixelek megjelenésének összege [17].

Például ha a bemeneti szürkeárnyalatos kép $\{0, 1, 2, 3\}$ értékű szürkeárnyalatokat vehet fel, akkor azt reprezentálhatjuk az alábbi módon.

0	0	1	1	1
0	0	1	1	1
0	2	2	2	2
2	2	3	3	3
2	2	3	3	3

3.11. ábra. Szürkeárnyalatos kép

Legyen $d_x = 1$, $d_y = 0$. Ekkor a $GLCM$ mátrix:

	0	1	2	3
0	2	2	1	0
1	0	4	0	0
2	0	0	5	2
3	0	0	0	4

3.12. ábra. $GLCM$

A pirossal jelölt oszlopban találhatók a referencia pixelek értékei, a kékkel jelölt sorban pedig a szomszéd pixelek értékei. A bekarikázott érték az 3.11 ábrán bekarikázott párok megszámlálásával lett értékül adva.

Megjegyzés: a $GLCM$ mátrix nem minden esetben szimmetrikus [17].

A $GLCM$ mátrix azonban még önmagában nem hordoz a bemeneti szürkeárnyalatos képről jól definiálható vizuális érzékeléssel kapcsolatos információkat. A következő összefüggésekkel nyerhetünk ki belőle releváns információkat, amelyek végeredményben a képpel kapcsolatos klaszterezhető tulajdonságokat is alkotják majd.

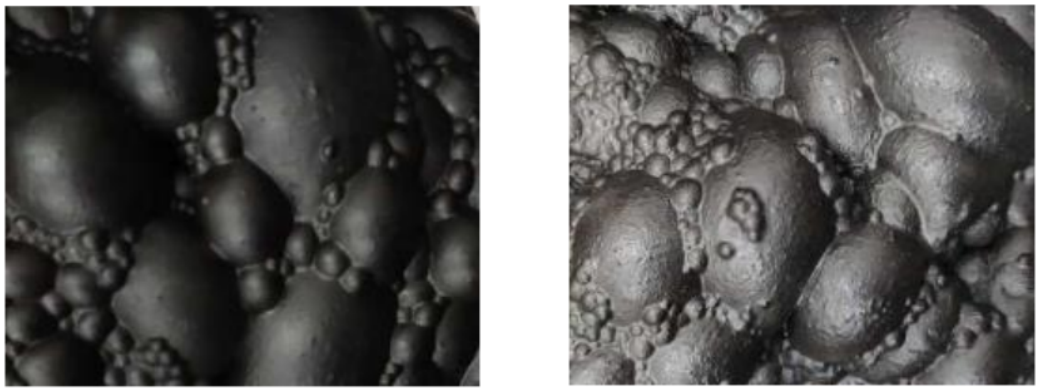
A kontraszt, a különbözőség és a uniformitás a $GLCM$ mátrix értékeit a diagonális viszonyában tekinti [17].

A kontraszt kiszámítható

$$\sum_{i,j=0}^{N-1} GLCM_{i,j} \left(i-j\right)^2 \tag{3.9}$$

módon.

Ha $i = j$ akkor a $GLCM$ mátrix főátlóján lévő azonos szürkeárnyalatú pixelpárokat vizsgáljuk, ekkor $i - j = 0$. A kontraszt exponenciálisan súlyozott, tehát exponenciálisan nő $i - j$ növekedésével.



(a) Alacsonyabb, 0.085 (b) Magasabb, 0.264

3.13. ábra. Kontraszt

Különbözőség az alábbi képlettel számítható ki:

$$\sum_{i,j=0}^{N-1} GLCM_{i,j} \left|i-j\right|. \tag{3.10}$$

A kontraszttal ellentétben nem exponenciálisan súlyozott, hanem lineárisan.

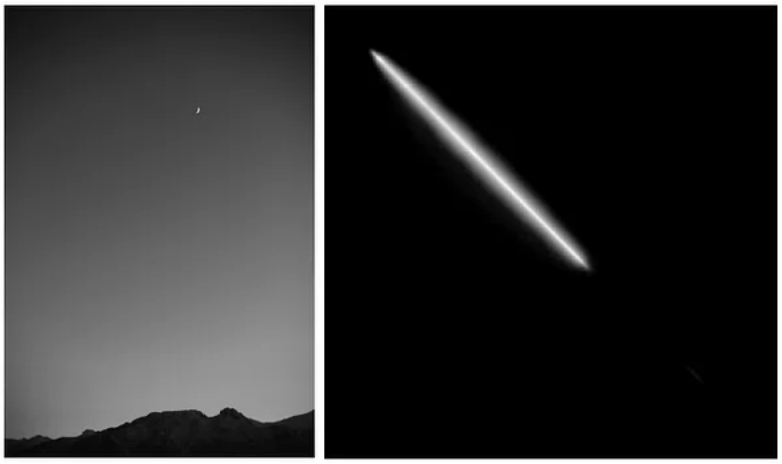
Az uniformitás súlyozása a kontraszttal ellentétben a diagonálistól ($i - j = 0$) távolodva csökken.

$$\sum_{i,j=0}^{N-1} \frac{GLCM_{i,j}}{1+\left(i-j\right)^2} \tag{3.11}$$

(a) Alacsonyabb, 0.928

(b) Magasabb, 0.998

3.14. ábra. Uniformitás



(a) Uniformabb kép

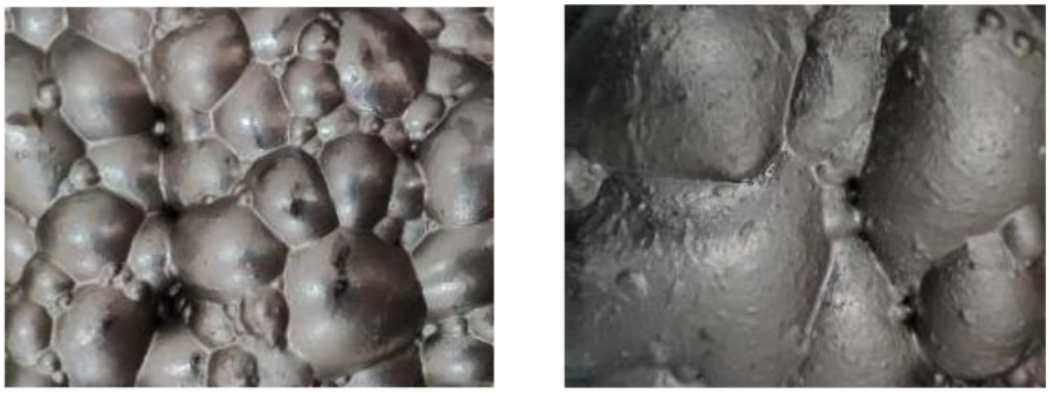
(b) Kontrasztosabb kép

3.15. ábra. *GLCM* mátrixok képekhez rendelve[4]

[4]towardsdatascience.com [Hozzáférés: 2024. május 20.]

Az energia nem a diagonális alapján kerül kiszámításra. A képek rendezettségét, textúrájuk egyenletességét fejezi ki az alábbi formulával.

$$\sqrt{\sum_{i,j=0}^{N-1} GLCM_{i,j}^2} \tag{3.12}$$



(a) Alacsonyabb, 0.361 (b) Magasabb, 0.862

3.16. ábra. Energia

Az itt felsoroltakon kívül többek között vizsgálható még $GLCM$ mátrix segítségével a képek korrelációja, entrópiája, átlaga és varianciája is [18].
Vezessük be $GLCM_T$-t, amely egy $GLCM$-hez hasonló mátrixot hoz létre, azzal a különbséggel, hogy az inputja nem kép, hanem $T_{n,m}$ töruszrácsgráf. Feleltessük meg $GLCM_{a,b}$ indexelését a töruszrácsgráf címkézésével (3.2.3).

Tekintsük meg az alábbi példát $T_{5,5}$ töruszrácsgráfon, $\{0, 1, 2, 3\}$ állapotokkal.

0	0	1	1	1
0	0	1	1	1
0	2	2	2	2
2	2	3	3	3
2	2	3	3	3

(a) Szürkeárnyalatos kép

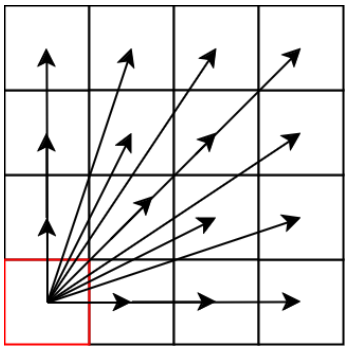
	0	1	2	3
0	2	2	1	0
1	2	4	0	0
2	1	0	5	2
3	0	0	2	4

(b) $GLCM_T$

3.17. ábra. $GLCM_T$ példa

Összehasonlítva a kép (véges négyzetrács) formátumot a töruszszerkezettel, megállapíthatjuk, hogy a másodfokú $GLCM_T$ mátrix elemeinek összege biztosan na-

gyobb, mint a másodfokú $GLCM$ mátrix elemeinek összege, mivel több szomszédság létezik $x * y$ méretű tóruszrácsgráfon, mint $x * y$ méretű négyzetrácson.
$x * y$ méretű négyzetrácson legalább egy pixelpárt meg kell vizsgálni, ha $d_x = x - 1$ és $d_y = y - 1$. Ekkor a kép két sarkában lévő pontot hasonlítjuk össze. Legfeljebb $(x - 1) * y$ párt hasonlíthatunk össze egy négyzetrácson, ha a (példában is látható) legkisebb $d_x = 1$ és $d_y = 0$ távolságokat választjuk. (Feltéve, hogy $x > y$. Ellenkező esetben legfeljebb $x * (y - 1)$ párt hasonlítunk össze.)
Tóruszrácsgráfon azonban minden esetben $x * y$ párt vizsgálunk a körkörös összeérések miatt, illetve d_x és d_y nincsenek maximalizálva, értékük tetszőleges egész szám lehet.
Mivel minden (x', y') pár mellett megvizsgáljuk a (y', x') párt is, ezért a $GLCM_T$ mátrix minden esetben szimmetrikus.
Minden bemeneti textúrát vizsgáljunk meg kontrasztra, különbözőségre, uniformitásra és energiára, az alábbi (d_x, d_y) párokra: (1, 1), (0, 1), (1, 0), (2, 2), (0, 2), (2, 0), (1, 2), (2, 1), (3, 3), (0, 3), (3, 0), (1, 3), (3, 1), (2, 3), (3, 2).



3.18. ábra. (d_x, d_y) párok

A konstansoknál (3.3.1) beállított `numberOfRuleIteration` = 10 textúrán 15 távolságra számolunk ki $GLCM_T$ mátrixot, majd azokat a mátrixokat 4 tulajdonságra vizsgáljuk. Tehát egy szabályhoz $10 * 15 * 4 = 600$ jellemzőt rendelünk.

3.3.9. Klaszterezés

A kinyert tulajdonságokat (3.3.8) valamilyen klaszterező algoritmussal csoportosítjuk. Legyen ez a K-Means algoritmus. Ez az klasszifikáló algoritmus önszervező (angolul unsupervised), tehát nincsenek megadott csoportok, amelyekbe el kell helyeznie az objektumokat, hanem az algoritmus határozza meg azokat. A csoportok száma azonban paraméterezhető, jelöljük ezt K-val. A K-Means távolság alapú al-

goritmus. [19]

Kezdetben K pontot választunk ki a klaszterek középpontjainak. Nevezzük ezeket centroidoknak. A centroidok inicializálásának több módja is van, az egyik ilyen a K pont véletlenszerű kiválasztása. Ez után kiszámítjuk a bemeneti elemek távolságát a centoridok mindegyikéhez. Minden pontot ahhoz az centroidhoz rendelünk, amelyik az adott ponthoz a legközelebb esik. Ezek után a centroidok pozícióját a hozzájuk tartozó pontok átlagos pozíciójához rendeljük. Addig ismételjük a pontok centroidokhoz rendelését, majd a centroidok újrapozícionálását, amíg a centroidok egy meghatározott küszöbértéknél kisebbet mozdulnak el a pozíció megváltoztatásakor.

Formálisan, ha adott n darab d-dimenziós pont

$$X_1 = (x_1 1, x_1 2, \ldots, x_1 d)$$

$$X_2 = (x_2 1, x_2 2, \ldots, x_2 d)$$

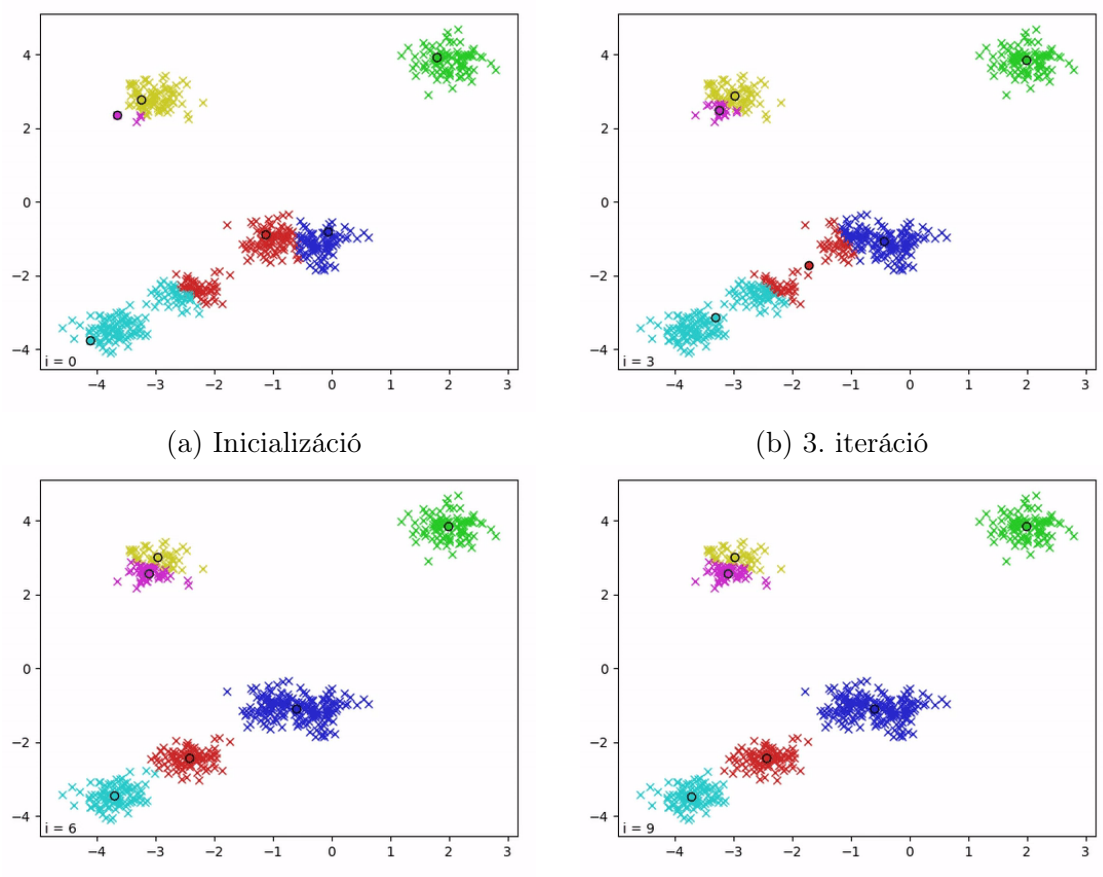
$$\ldots$$

$$X_n = (x_n 1, x_n 2, \ldots, x_n d)$$

akkor a cél $\{X_1, X_2, \ldots, X_n\}$ elhelyezése K darab klaszterbe. Jelöljük a klasztereket $\{C_1, C_2, \ldots C_K\}$-val. A K-Means algoritmus célja a $\mu_1, \mu_2, \ldots, \mu_K$ centroidokat úgy pozícionálni, hogy minimum távolság legyen a centroidok és a hozzájuk tartozó (velük egy klaszterbe eső) pontok között.

A K-Means az alábbi költségminimalizációt oldja meg [19]

$$\arg\min_{C} \sum_{i=1}^{K} \sum_{X_j \in C_i} \|X_j - \mu_i\|^2. \tag{3.13}$$



(a) Inicializáció

(b) 3. iteráció

(c) 6. iteráció

(d) 9. iteráció

3.19. ábra. K-Means klasztering

Legyen

$$K = 20$$

Megjegyzés: K megfelelő beállítása kulcsfontosságú az eredményes klaszterezéshez. Túl kicsi K esetén egy klaszterbe kerülhetnek egymástól lényegileg különböző objektumok, míg túl nagy K esetén lényegileg hasonló objektumok is külön klaszterekbe kerülhetnek [19].

3.3.10. Klaszterek kibővítése szimmetrikus szabályokkal

Minden reprezentás szabályból generáljuk le a hozzá tartozó szimmetrikus szabályokat a 3.3.2 fejezetben leírt módon a négyzet szimmetriáival, valamint a negálás transzformációival.

Alternatív és ekvivalens megoldás lehet a szimmetrikus szabályok összevonásánál (3.3.2) elmenteni az egymáshoz tartozó szabályokat. Ez esetben ebben a lépésben mindössze ismét össze kell vonni őket a hozzájuk tartozó reprezentánssal.

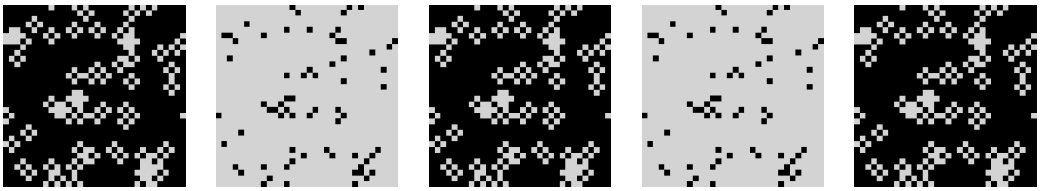
3.4. CARCT futtatása

Tekintsük meg a CARCT futtatásának néhány fontosabb részletét. A sejtautomata állapotok és textúrák megjelenítése a 3.2.4 fejezetben meghatározott módon történik.

3.4.1. Inkonzisztens szabályok

A 2.5.2 fejezetben bemutatott inkozisztens szabályokra tekintsünk néhány példát.

- Tekintsük meg azon szabályok egy csoportjának a `numberOfUpdates` − 4., `numberOfUpdates` − 3. . . . , `numberOfUpdates` iterációját, amelyek nem képesek legfeljebb `numberOfUpdates` lépés alatt `persistanceLength` egymást követő iteráción keresztül `similarityThreshold` $SSIM_T$ küszöbérték alatt maradni.



(a) -4. (b) -3. (c) -2. (d) -1. (e) `numberOfUpdates`.

3.20. ábra. `"0000000000000001"` 2D Wolfram kódú szabály sejtautomata állapotai

(a) -4. (b) -3. (c) -2. (d) -1. (e) `numberOfUpdates`.

3.21. ábra. `"0000000000010001"` 2D Wolfram kódú szabály sejtautomata állapotai

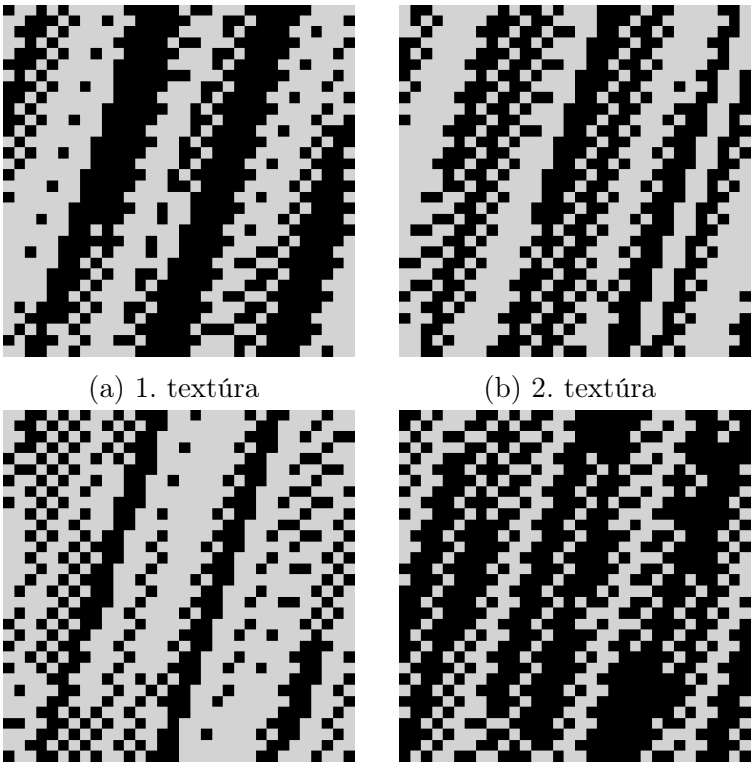
(a) -4. (b) -3. (c) -2. (d) -1. (e) `numberOfUpdates`.

3.22. ábra. `"0000000010000101"` 2D Wolfram kódú szabály sejtautomata állapotai

Ebben a lépésben 1422 szabályt vonunk ki az analízis alól.

Megjegyzés: Az oszcilláció önmagában még nem jelenti egy szabály kiesését, elképzelhető olyan eset is, amikor az oszcilláló sejtautomata állapotok kvantitatív módon hasonlók.

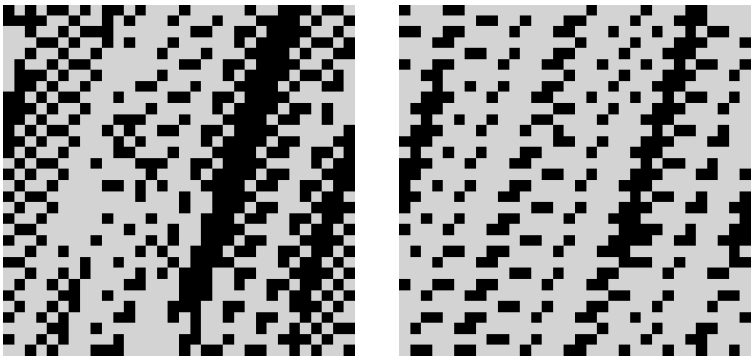
- Tekintsük meg azon szabályok egy csoportjának a textúráit, amelyek képesek legfeljebb `numberOfUpdates` lépés alatt `persistanceLength` egymást követő iteráción keresztül `similarityThreshold` $SSIM_T$ küszöbérték alatt maradni, azonban nem képesek erre többszörösen.



(a) 1. textúra (b) 2. textúra

(c) 3. textúra (d) 4. textúra (analízis vége)

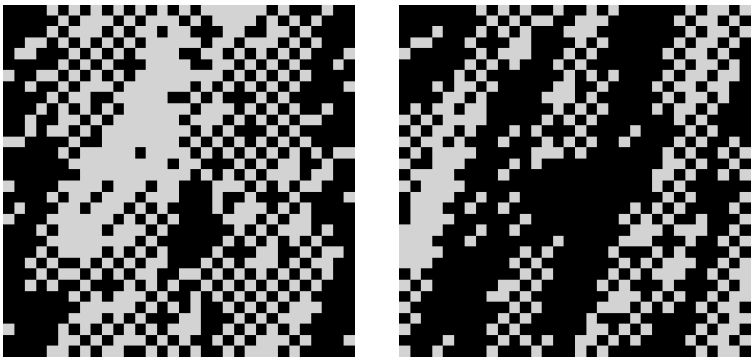
3.23. ábra. `"0000000010010001"` 2D Wolfram kódú szabály textúrái



(a) 1. textúra (b) 2. textúra

(c) 3. textúra (d) 4. textúra (analízis vége)

3.24. ábra. "0000000010011001" 2D Wolfram kódú szabály textúrái

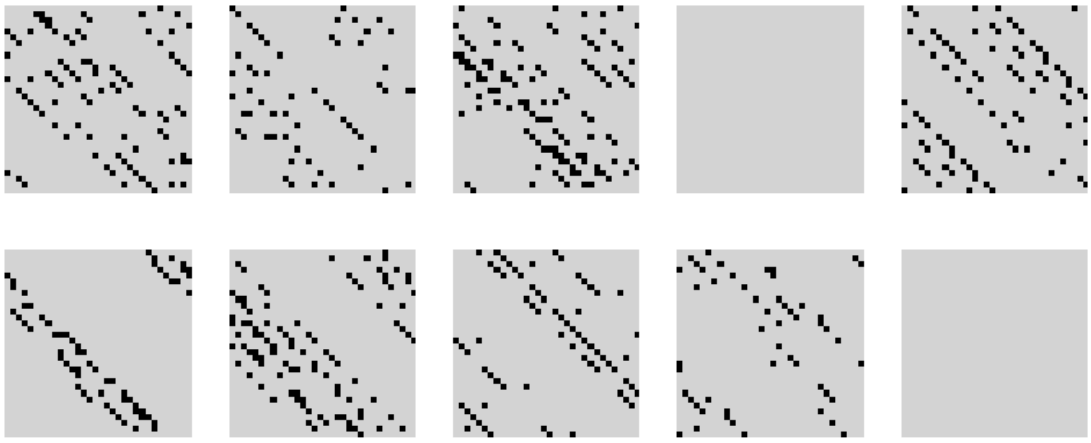


(a) 1. textúra (b) 2. textúra (analízis vége)

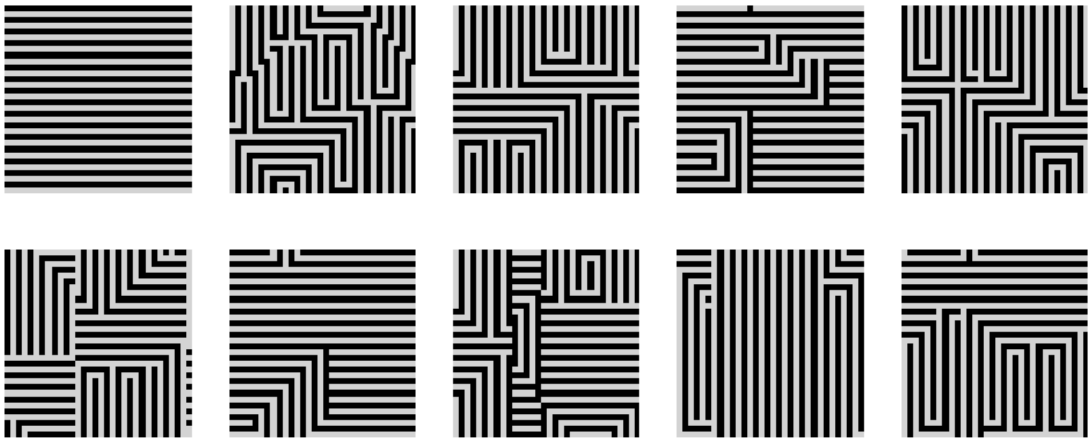
3.25. ábra. "0001000010010101" 2D Wolfram kódú szabály textúrái

Ebben a lépésben 188 szabályt vonunk ki az analízis alól.

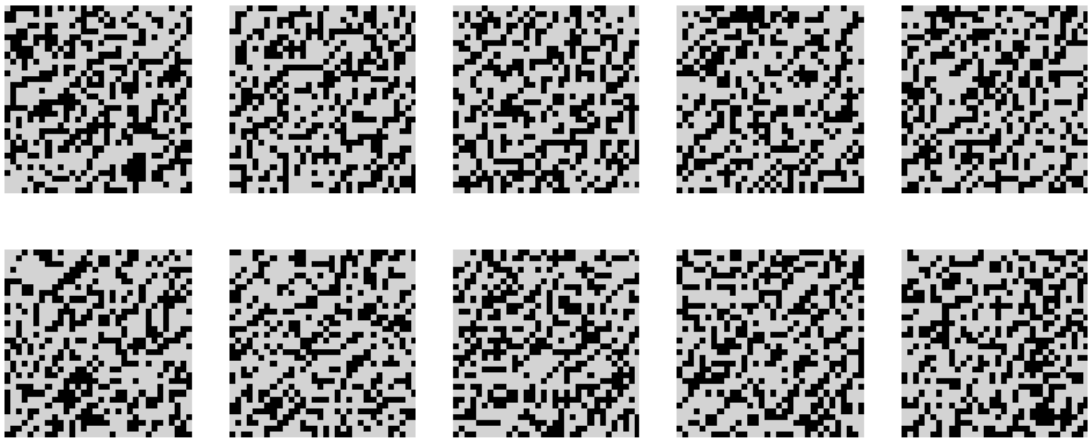
- Tekintsük meg azon szabályok egy csoportjának a textúráit, amelyek képesek legfeljebb `numberOfUpdates` lépés alatt `persistanceLength` egymást követő iteráción keresztül `similarityThreshold` $SSIM_T$ küszöbérték alatt maradni többszörösen, azonban mégis inkonzisztensek, mert a textúrák kvantitaív módon nem hasonlítanak egymásra (3.3.7).



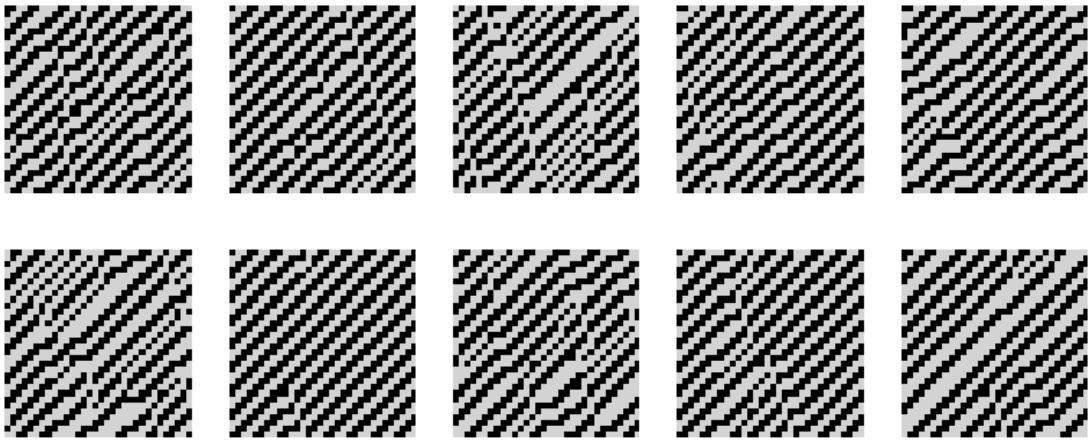
3.26. ábra. "0000000000010100" inkonzisztens szabály textúrái



3.27. ábra. "0000010101001100" inkonzisztens szabály textúrái



3.28. ábra. "0001010010001110" inkonzisztens szabály textúrái



3.29. ábra. "0001000110010000" inkonzisztens szabály textúrái

Ebben a lépésben 686 szabályt vonunk ki az analízis alól.

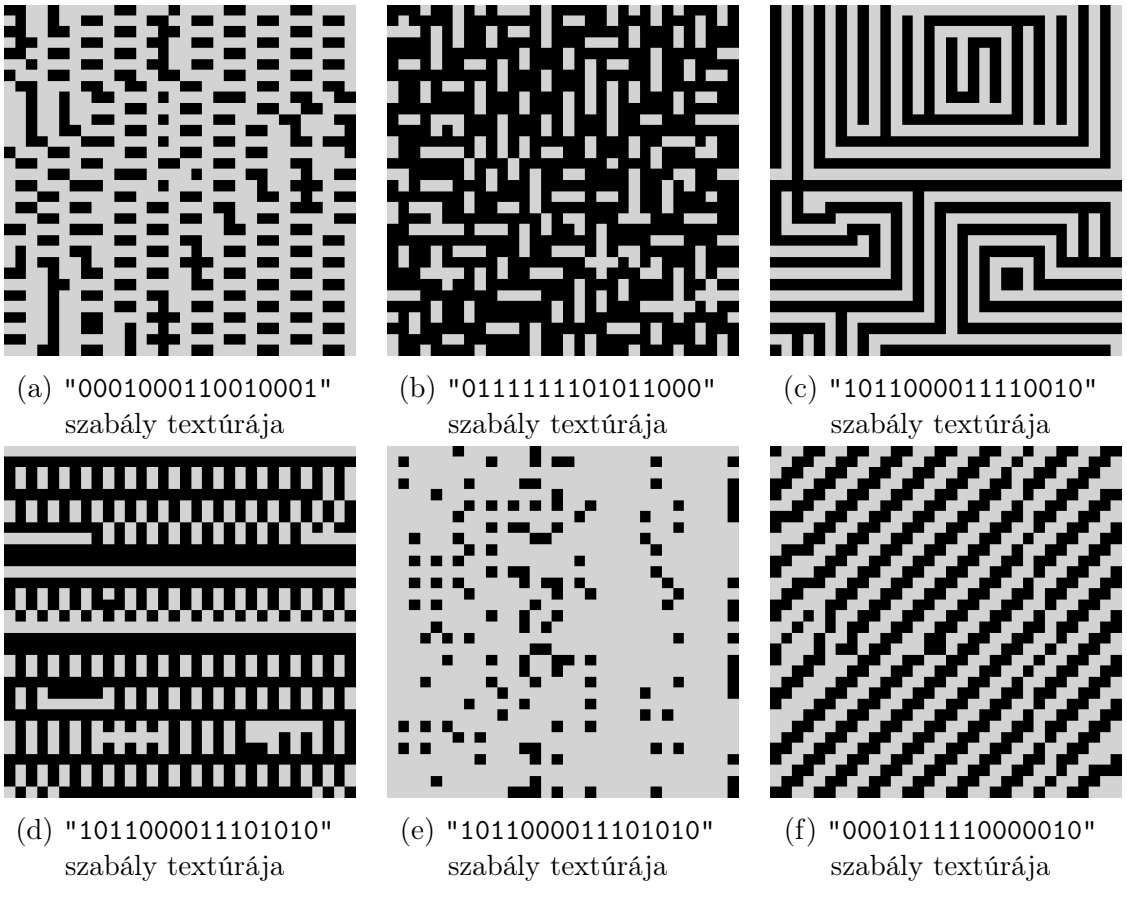
Végül 2560 konzisztens szabályt vizsgálunk meg.

3.4.2. Vizuális tulajdonságok összevetése

Az alábbi táblázatban három sejtautomata állapothoz tartozó $GLCM_T$ mátrixot mutatunk be $d_x = 0, d_y = 1$ távolságokkal. Továbbá feltüntetjük az ezekhez tartozó kontraszt, uniformitás, különbözőség és energia értékeket.

	(a)	(b)	(c)	(d)	(e)	(f)
GLCM	473 194 194 163	89 236 236 463	257 254 254 259	128 270 270 356	765 124 124 11	271 260 260 233
Kontraszt	0.378	0.460	0.496	0.527	0.242	0.507
Uniformitás	0.810	0.769	0.751	0.736	0.878	0.746
Különbözőség	0.378	0.460	0.496	0.527	0.242	0.507
Energia	0.557	0.564	0.500	0.524	0.766	0.500

3.3. táblázat. GLCM mátrixból kinyert vizuális tulajdonságok



(a) "0001000110010001" szabály textúrája

(b) "0111111101011000" szabály textúrája

(c) "10110000111110010" szabály textúrája

(d) "1011000011101010" szabály textúrája

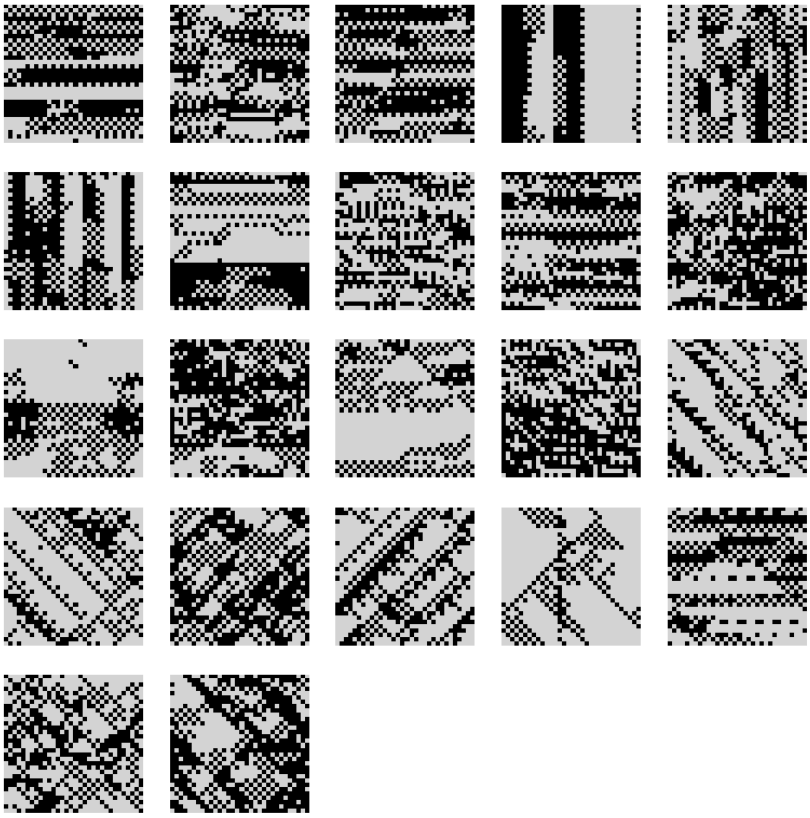
(e) "1011000011101010" szabály textúrája

(f) "00010111100000010" szabály textúrája

3.30. ábra. Textúrák

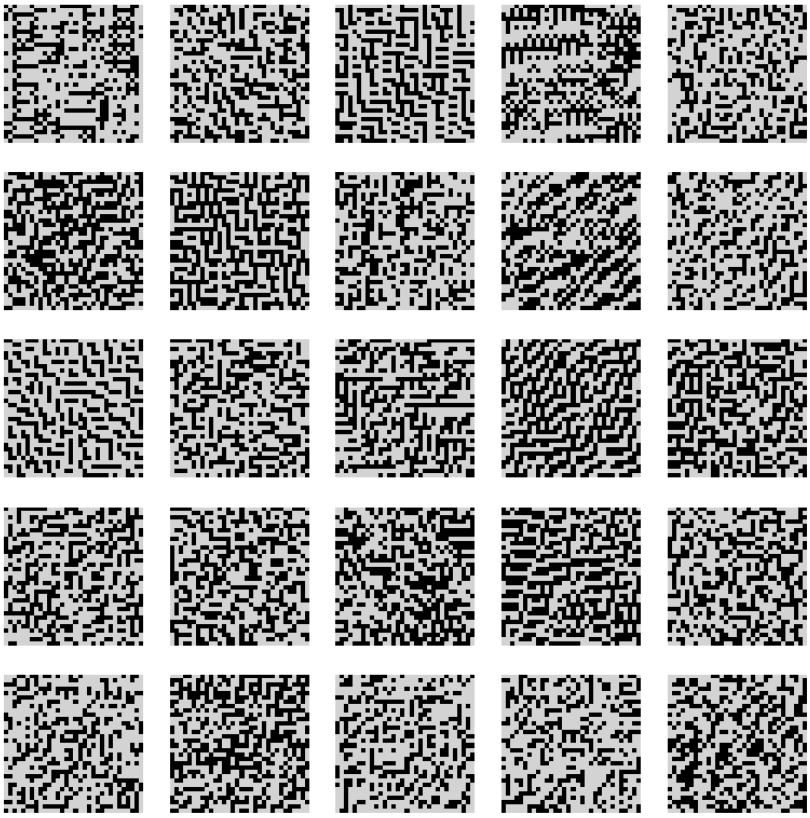
3.4.3. Klaszterezés eredménye

Tekintsük meg a klaszterezés eredményeit. A $K = 20$ klaszterből rendre 25 szabályt mutatunk, vagy ha a klaszter kevesebb mint 25 elemű, akkor az összeset, úgy, hogy egy adott szabályhoz tartozó 10 darab konzisztens textúra közül random választunk egyet.

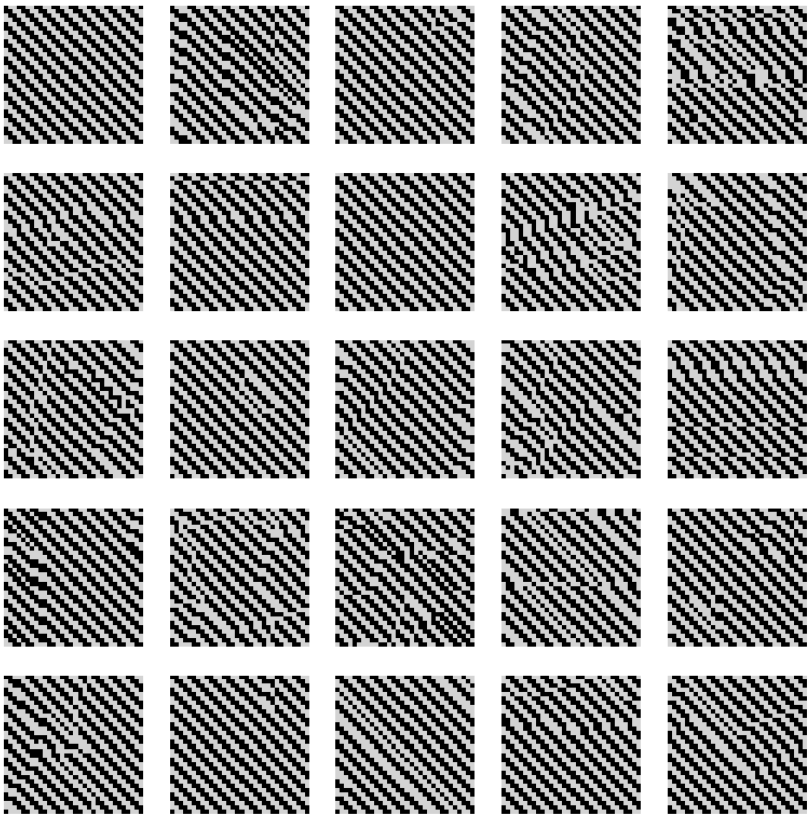


3.31. ábra. K_1

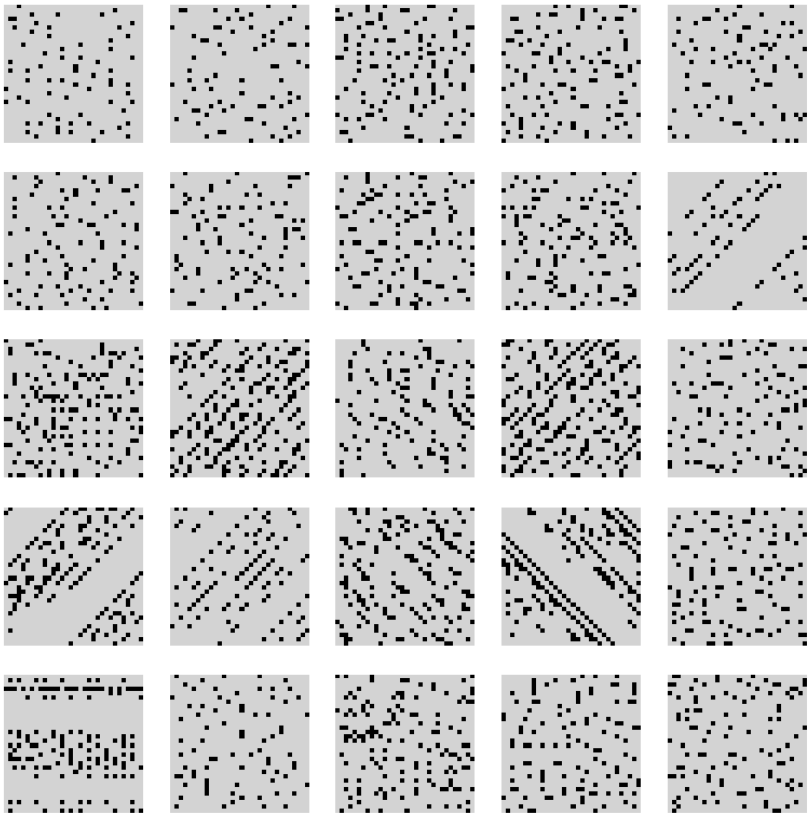
3.32. ábra. K_2



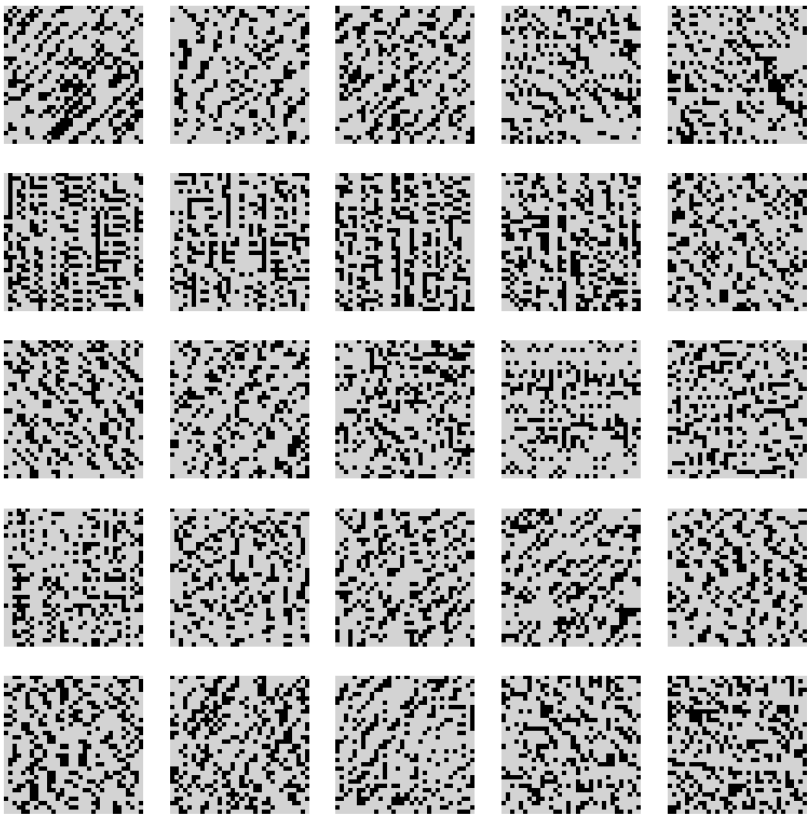
3.33. ábra. K_3



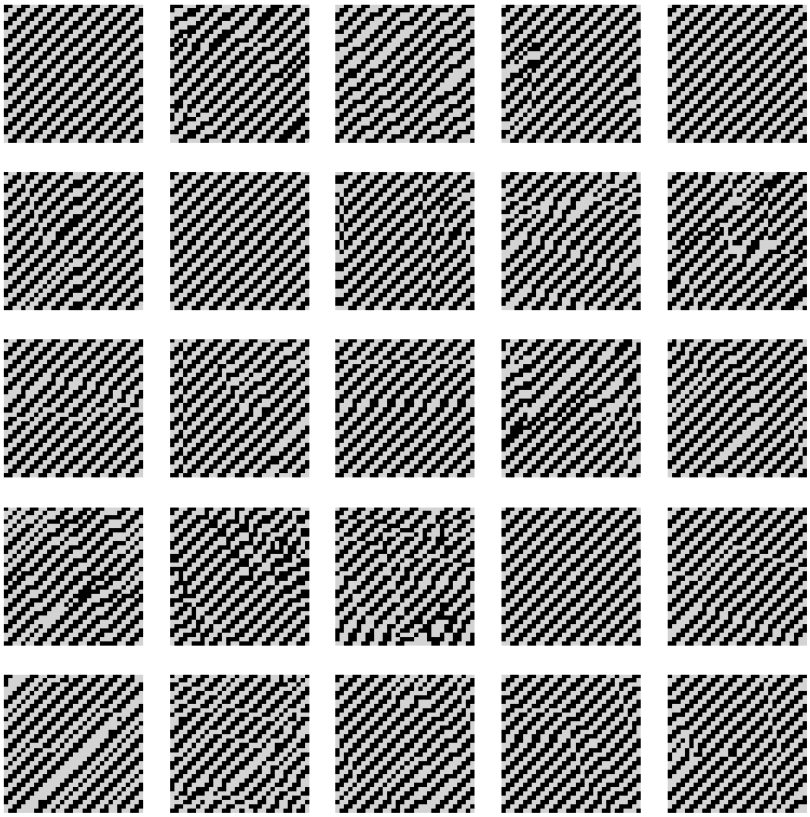
3.34. ábra. K_4



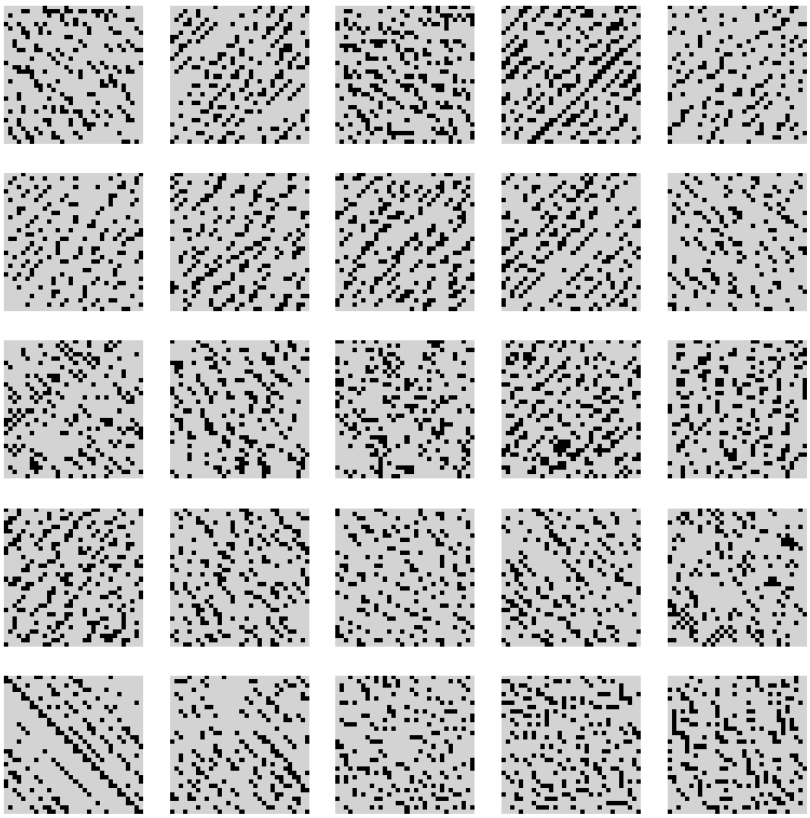
3.35. ábra. K_5



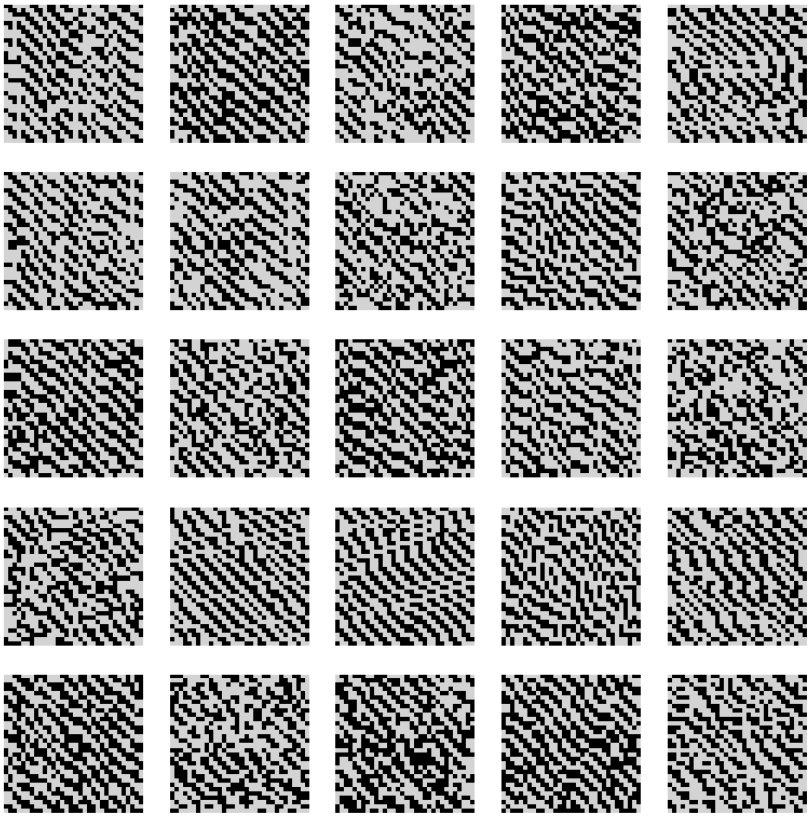
3.36. ábra. K_6



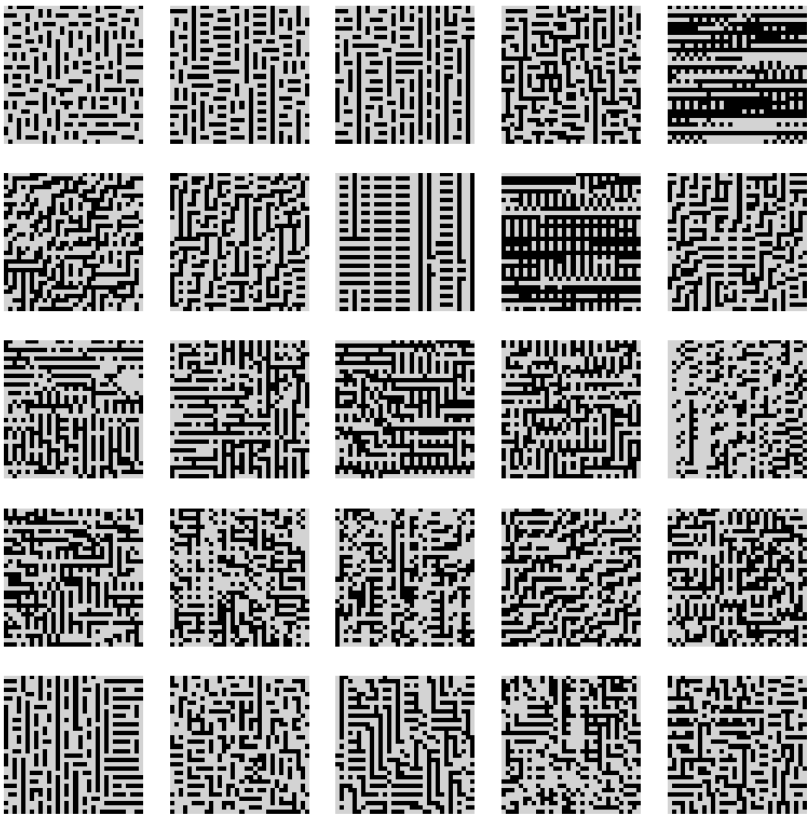
3.37. ábra. K_7



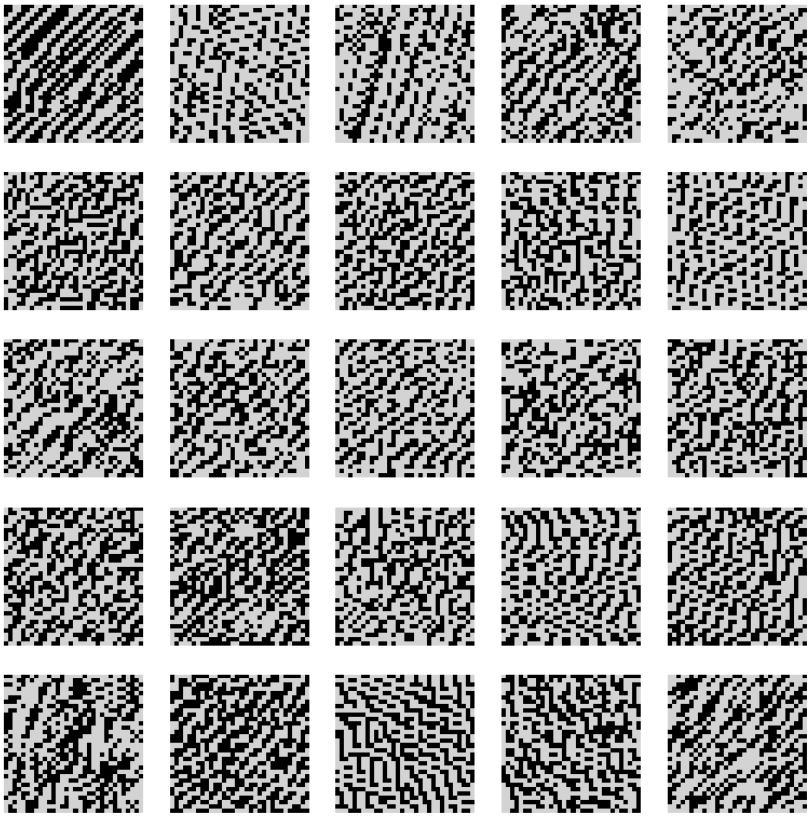
3.38. ábra. K_8



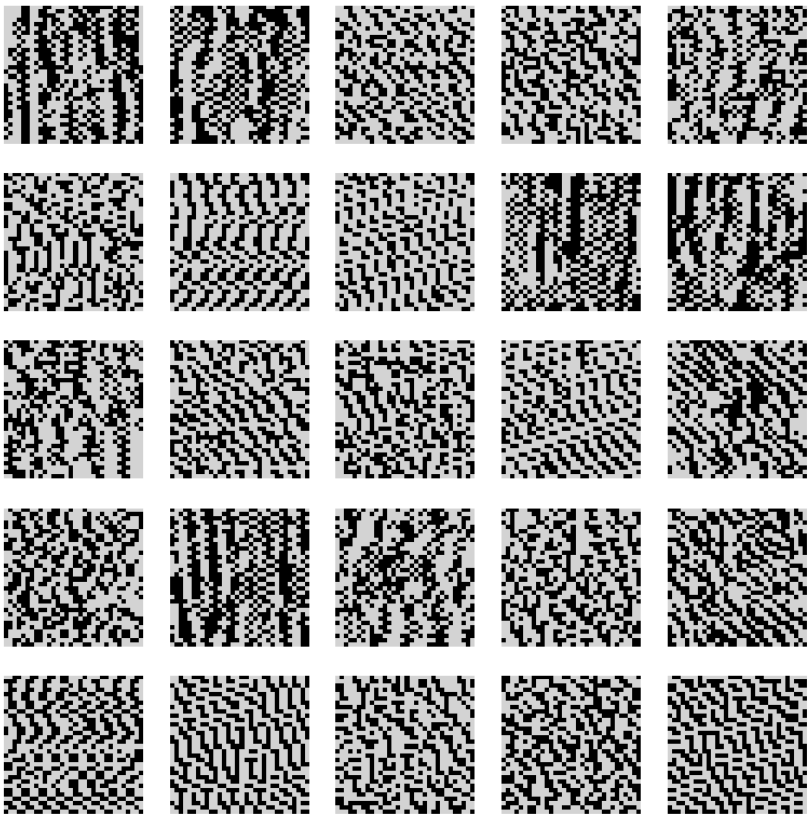
3.39. ábra. K_9



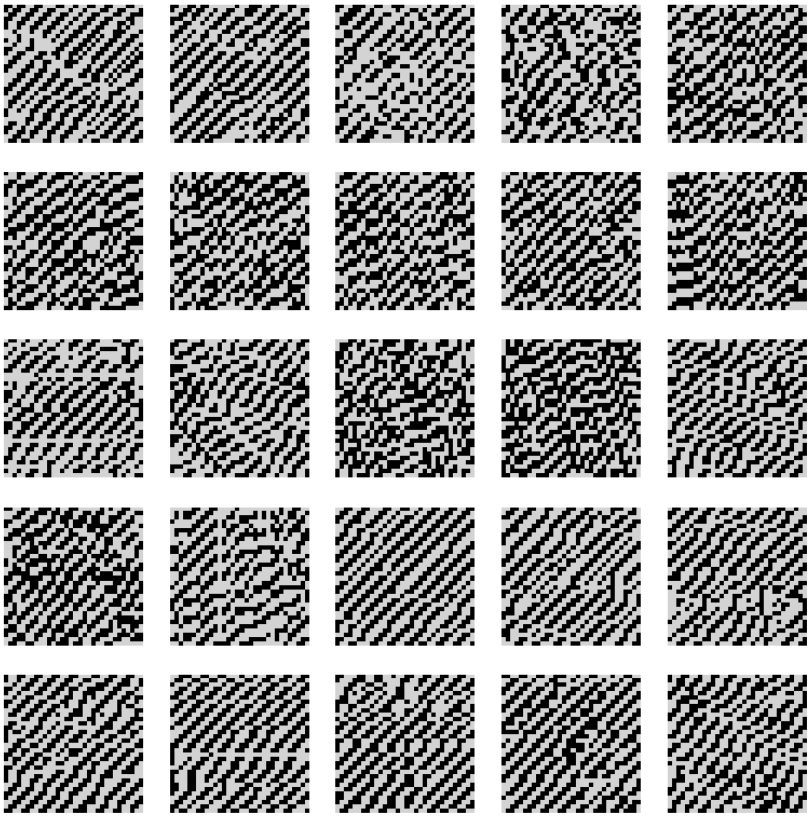
3.40. ábra. K_{10}



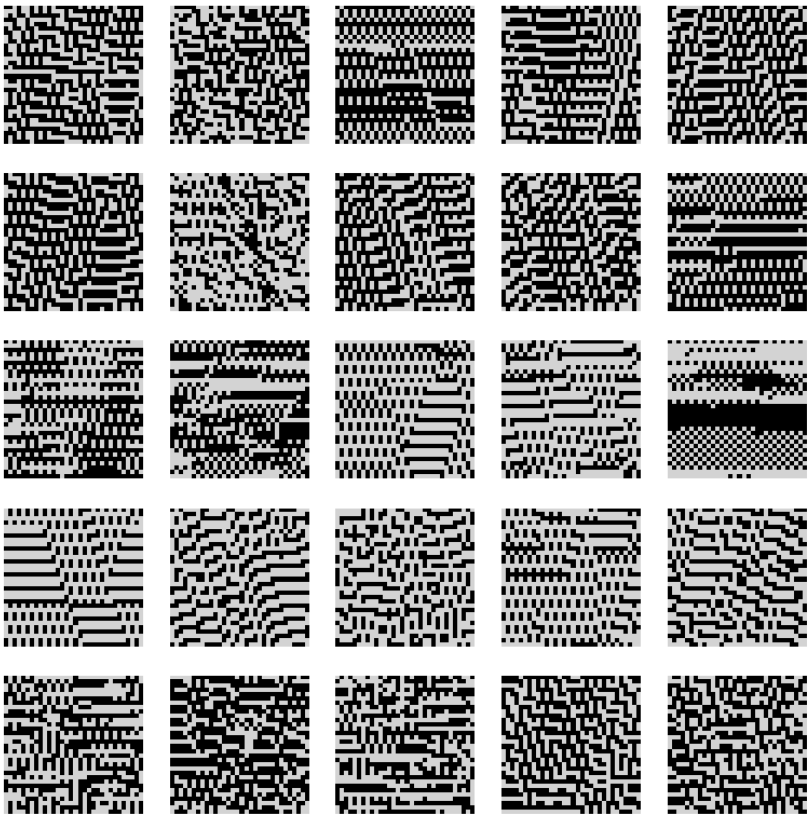
3.41. ábra. K_{11}



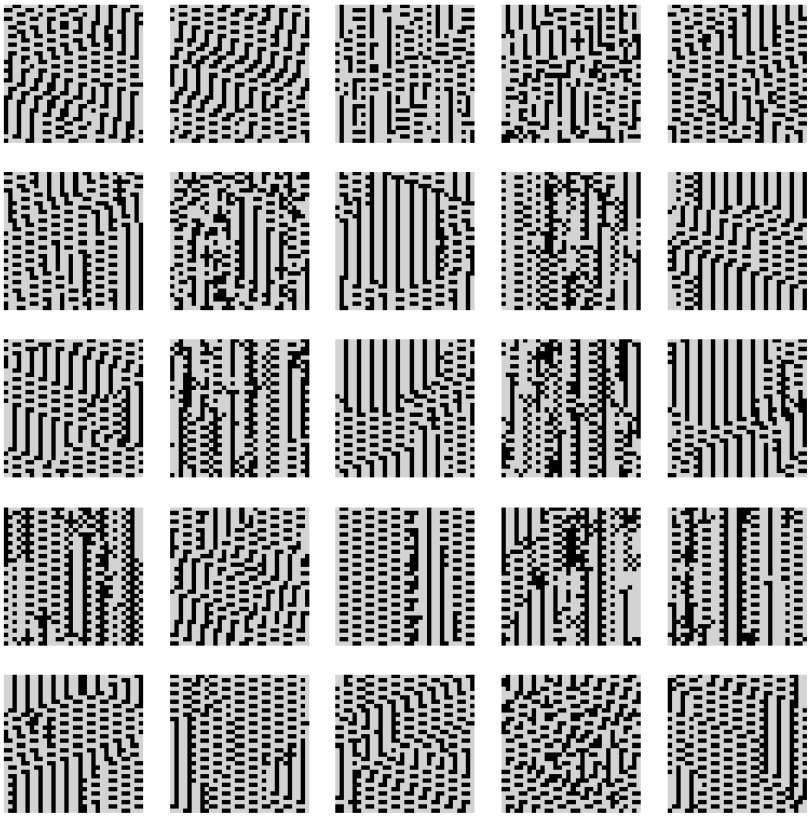
3.42. ábra. K_{12}



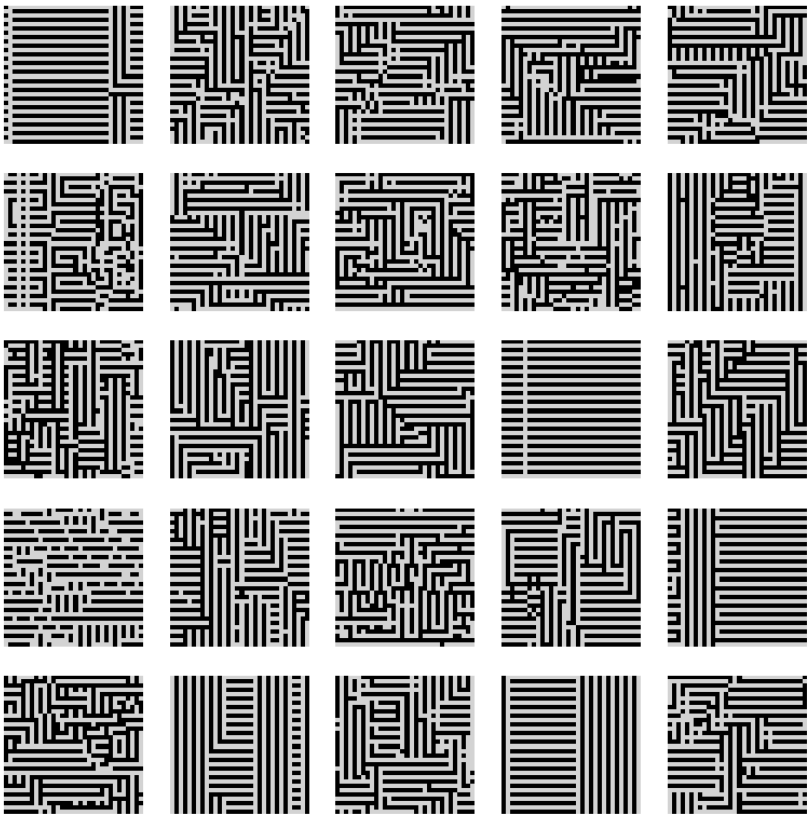
3.43. ábra. K_{13}



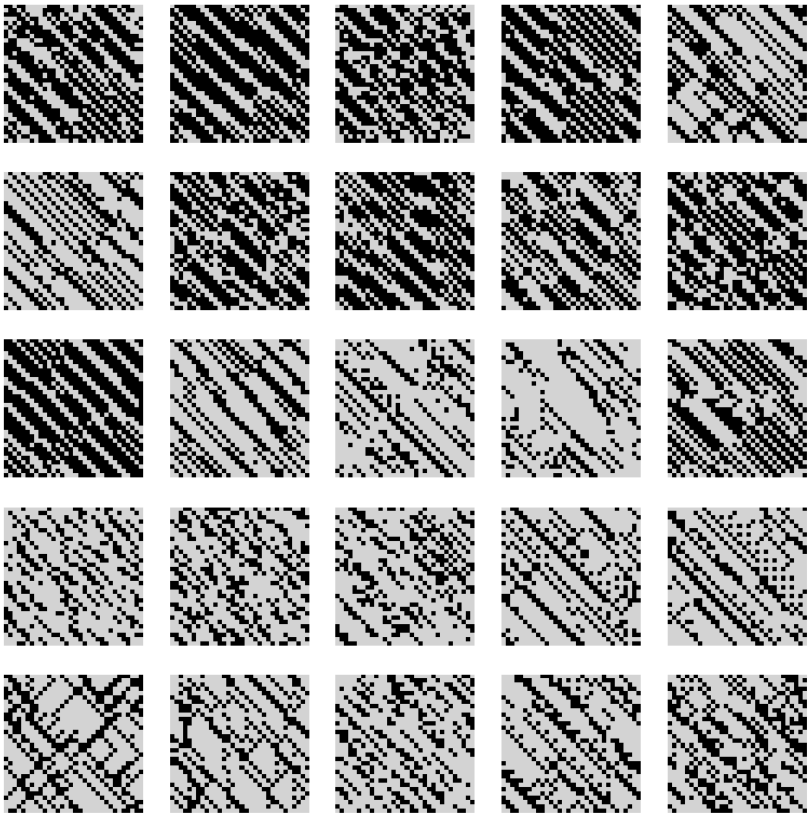
3.44. ábra. K_{14}



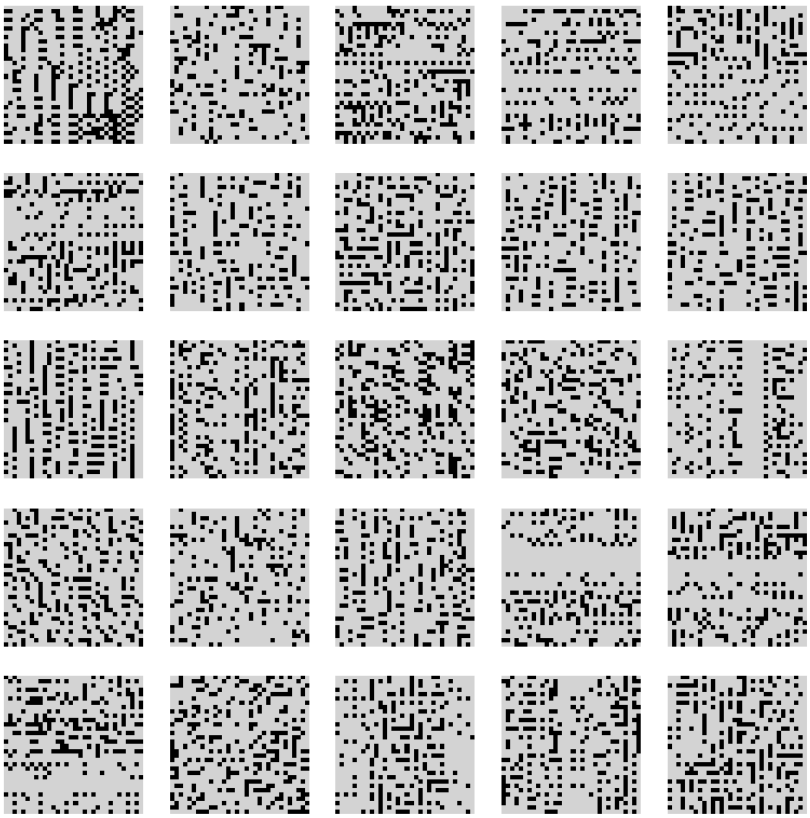
3.45. ábra. K_{15}



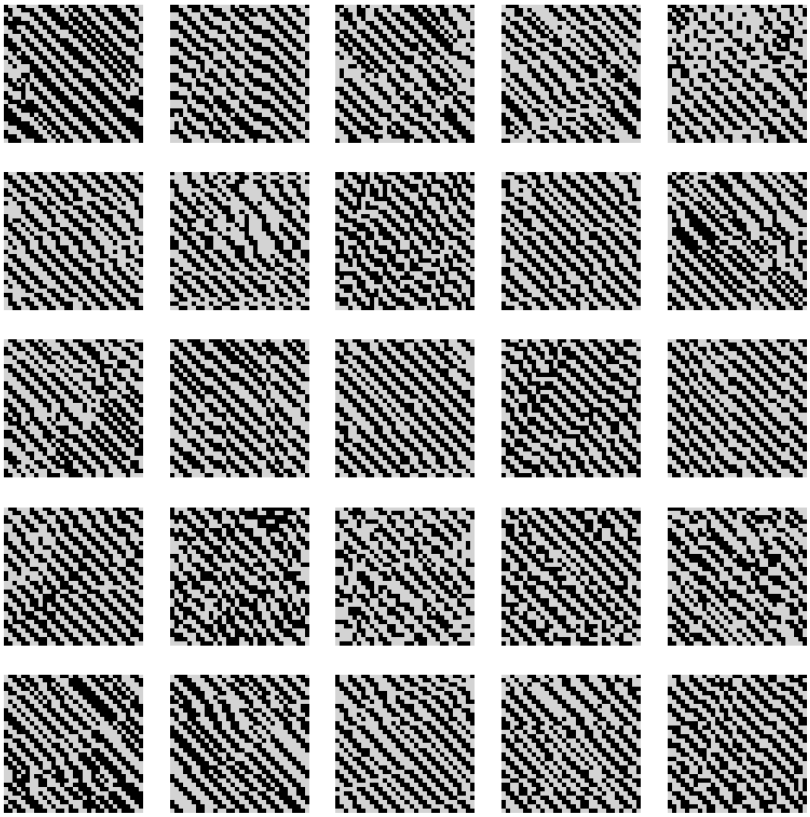
3.46. ábra. K_{16}



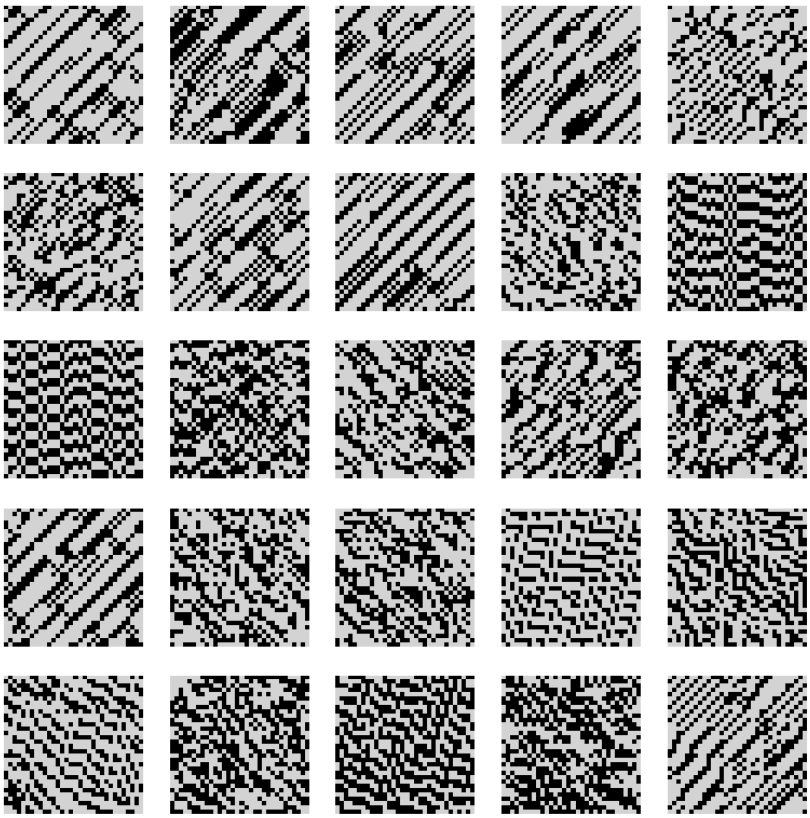
3.47. ábra. K_{17}



3.48. ábra. K_{18}



3.49. ábra. K_{19}



3.50. ábra. K_{20}

Az alábbi táblázatban látható, hogy az adott klaszterek hány (reprezentáns) szabályt tartalmaznak.

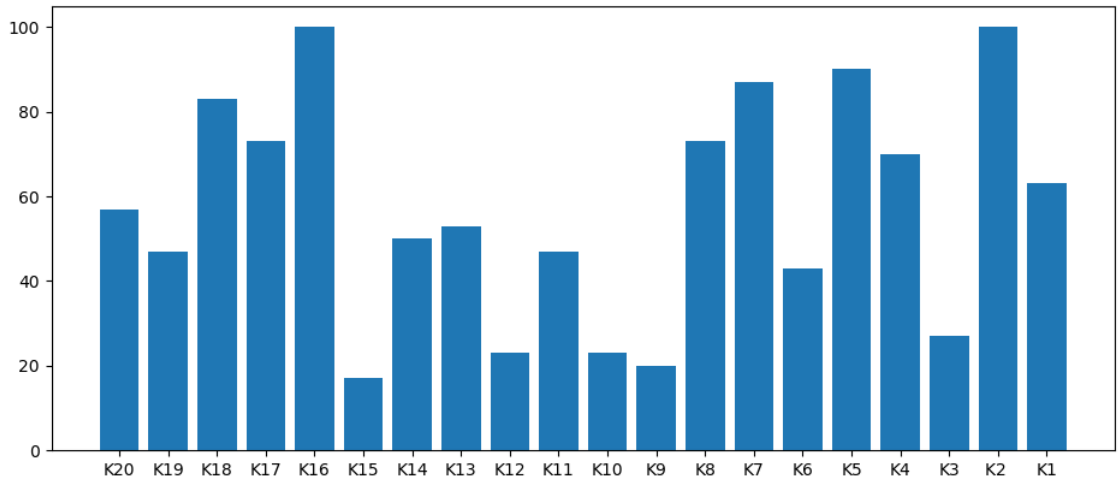
Klaszter index	**Reprezentáns szabályok száma**
K_1	22
K_2	232
K_3	311
K_4	86
K_5	189
K_6	221
K_7	69
K_8	244
K_9	109
K_{10}	135
K_{11}	128
K_{12}	134
K_{13}	70
K_{14}	49
K_{15}	44
K_{16}	31
K_{17}	51
K_{18}	136
K_{19}	58
K_{20}	239

3.4. táblázat. Reprezentáns szabályok száma klaszterenként

Az A függelékben hasonló módon bemutatjuk a $K = 10$ esetén kialakuló klasztereket is.

3.4.4. Tesztelés

A klaszterezés eredményei egy 10 fős mintán tesztelve lettek. A kiértékelés egy kvantitatívan mérhető feladat megoldásainak a vizsgálatával történt. A feladat a klaszterezéshez hasonló csoportosítás elvégzése volt, ahol a minden klaszterből kiválasztott 5 textúrához kellett a tesztalanyoknak 3-3 másik textúrát hozzárendelni.
Az alábbi diagramon látható, hogy a tesztalanyok melyik klaszterhez hány százalékban rendelték helyesen hozzá a textúrákat.



3.51. ábra. Klaszterekhez rendelt textúrák statisztikája

Fontos megjegyezni, hogy ha két különböző klaszter lényegileg hasonló elemeket tartalmaz (tehát K túl nagy), akkor a tesztalanyok könnyebben felcserélhetik a hozzájuk tartozó objektumokat. Ezen esetben a hibák tekinthetők úgy, hogy kétszer kerülnek megszámolásra.

A tesztalanyok eredményei a B függelékben találhatók.

4. fejezet

Összegzés

A diplomamunka tárgyalta a sejtautomatákat általánosságban, részletezte a hozzájuk tartozó definíciókat. Konkrétan ismertetve lett az elemi egydimenziós sejtautomata, majd azzal párhzamba állítva meghátároztuk az elemi kétdimenziós sejtautomata modelljét.
A szakdolgozat bemutatta a CARCT módszert és annak alkalmazását, mindeközben ismeretet nyújtott az $SSIM$ metrikáról, a $GLCM$ mátrixról, a vizuális információk kinyeréséről és azok K-Means klaszterezéssel való feldolgozásáról. Ezen kívül bemutatásra került a tóruszrácsgráf matematikai modellje.
Bevezetésre került a textúra definíciója. A Wolfram kóddal analóg módon definiáltuk a 2D Wolfram kód fogalmát. Meghatároztuk az $SSIM_T$ és a $GLCM_T$ általánosításokat tóruszra.

4.1. Kiértékelés

A dimplomamunkában prezentálva lett, hogy a módszer alkalmas sejtautomata szabályok klaszterezésére textúrák alapján. A bemutatott kétdimenziós elemi automatán ennek a vizuális érzékelésen alapuló példáját mutattuk be.
A tesztelés (3.4.4) bizonyította, hogy a klaszterezés a textúrák vizuális tulajdonságai szerint lényegében egybeesik az emberi percepcióval, így a CARCT alkalmazható erre. Ugyanakkor igaz, hogy a klaszterek némely esetekben hasonlóak. Ezek elkülönítése lehetséges más konstans értékek beállításával, vagy más algoritmusok kiválasztásával. Akár a klaszterek számának megváltoztatásával is módosítható, az A függelékben szerepel $K = 10$ klaszterezés során létrejött klaszterek szemléltetése.

4.2. Továbbfejelsztési lehetőségek

A módszer vizsgálata más automatáknak (például többdimenziós, vagy más rácson működő) más jellegű textúráit (például audió, vagy egyéb fizikai jellemzőit) nem érintette. Ebben az irányban még biztosan tovább vizsgálhatók a CARCT módszer keretei és korlátai.

További fejlesztési lehetőség lenne, ha a CARCT módszer kevesebb konstanssal, és kevesebb generikus algoritmussal működne. Nem magától értetődő, hogy ez megvalósítható, mert a sejtautomaták és azok textúrái rendkívül sokoldalúak tudnak lenni. A konstansok és a generikus algoritmusok csökkentése további vizsgálatokat és tervezést igényel.

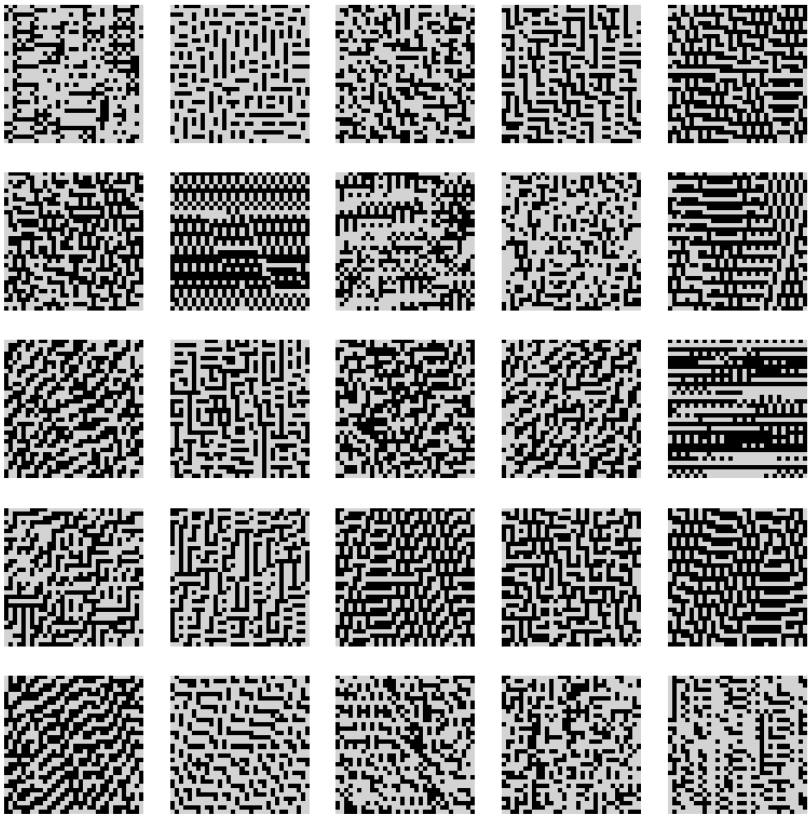
Köszönyetnyilvánítás

Szeretném megköszönni a sok segítséget Tichler Krisztián témavezetőmnek. Megjegyzéseivel és iránymutatásával nagyban hozzájárult a szakdolgozatom elkészültéhez. Az általa nyújtott szakmai tanácsok és konstruktív kritikák nemcsak a dolgozatom minőségét javították, hanem hozzájárultak szakmai fejlődésemhez is.

A. függelék

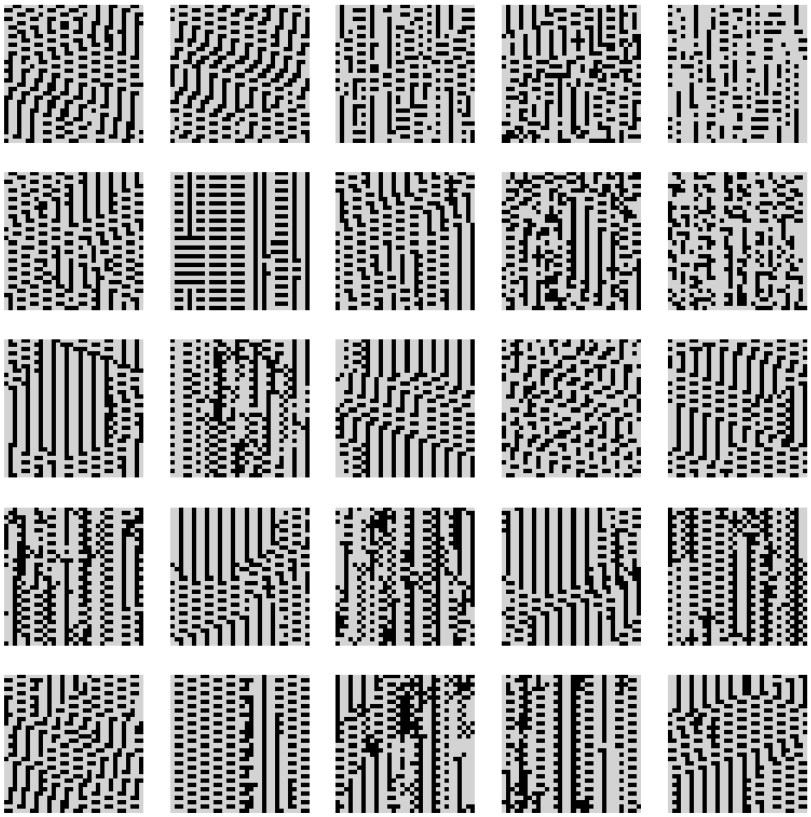
$K = 10$ klaszterezés

A 3.4.3 fejezetben leírt módon jelenítsük meg a klasztereket $K = 10$ darab klaszter esetén.

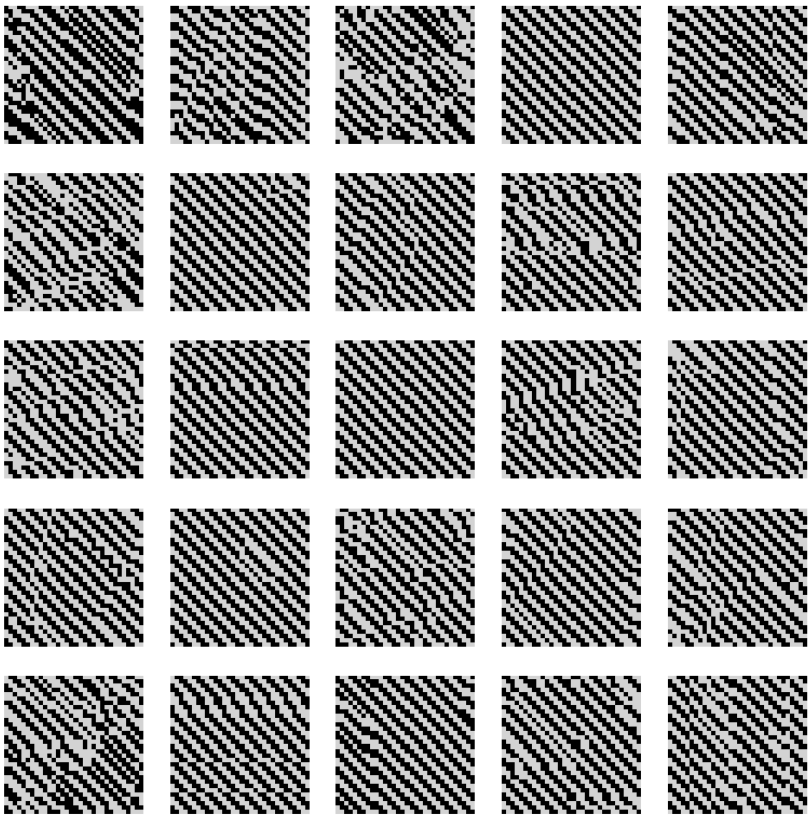


A.1. ábra. K_1'

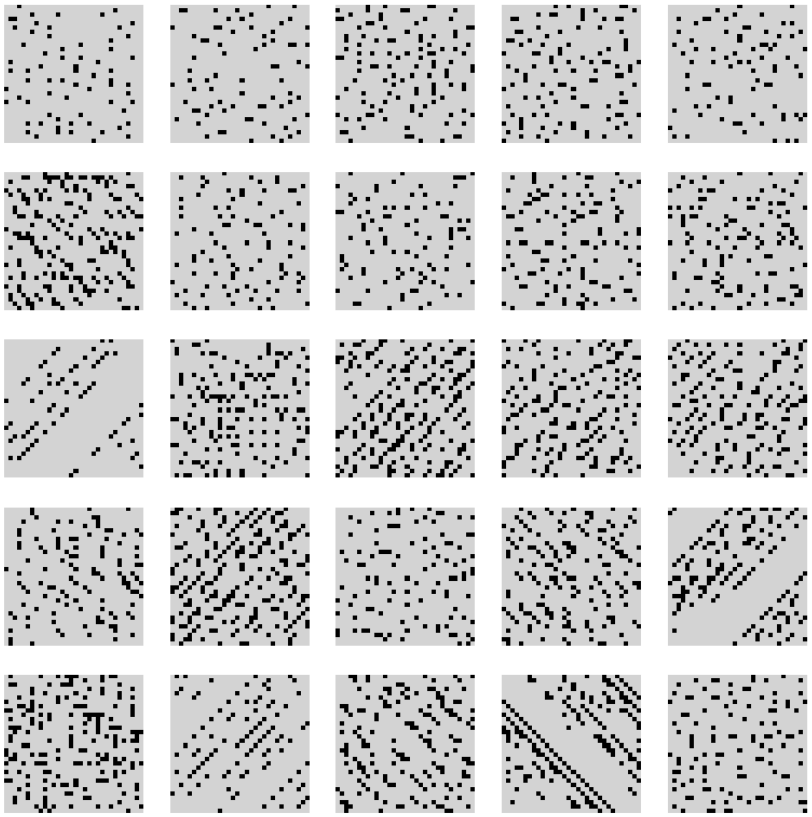
A.2. ábra. K_2'



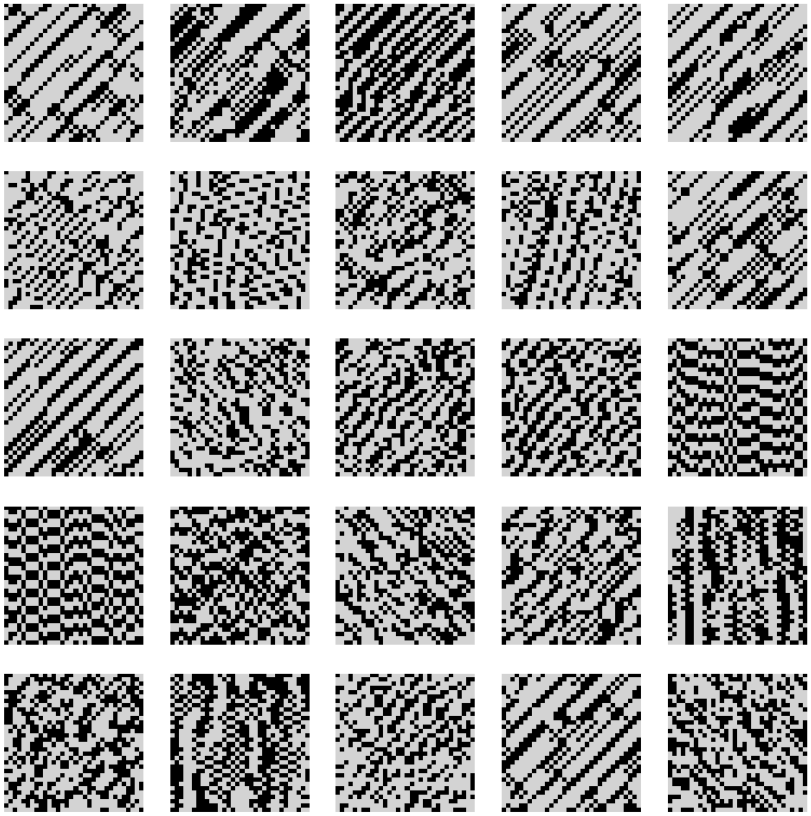
A.3. ábra. K_3'



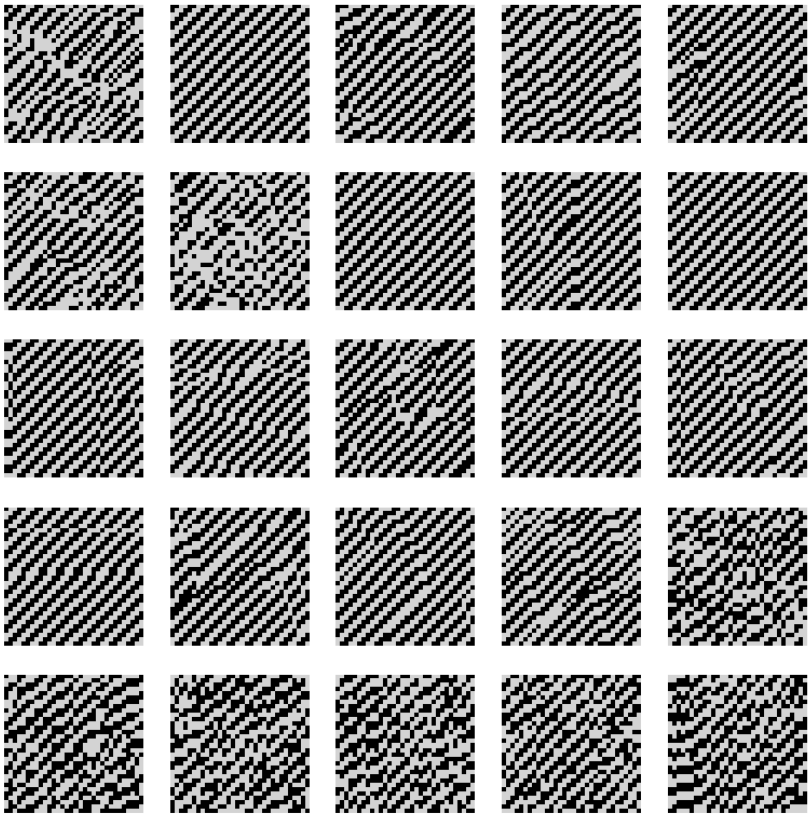
A.4. ábra. K'_4



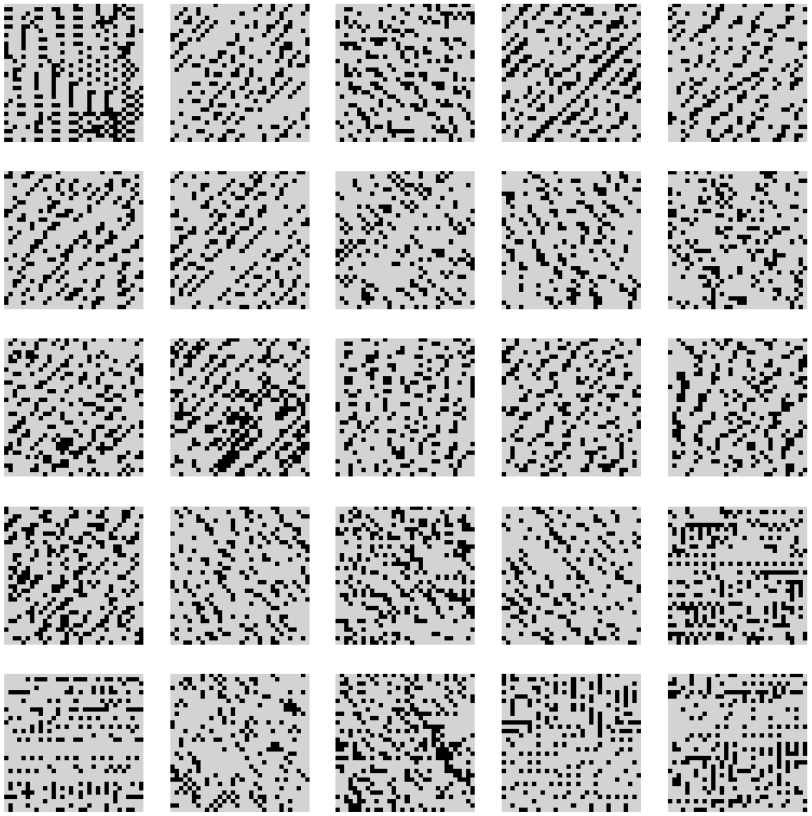
A.5. ábra. K'_5



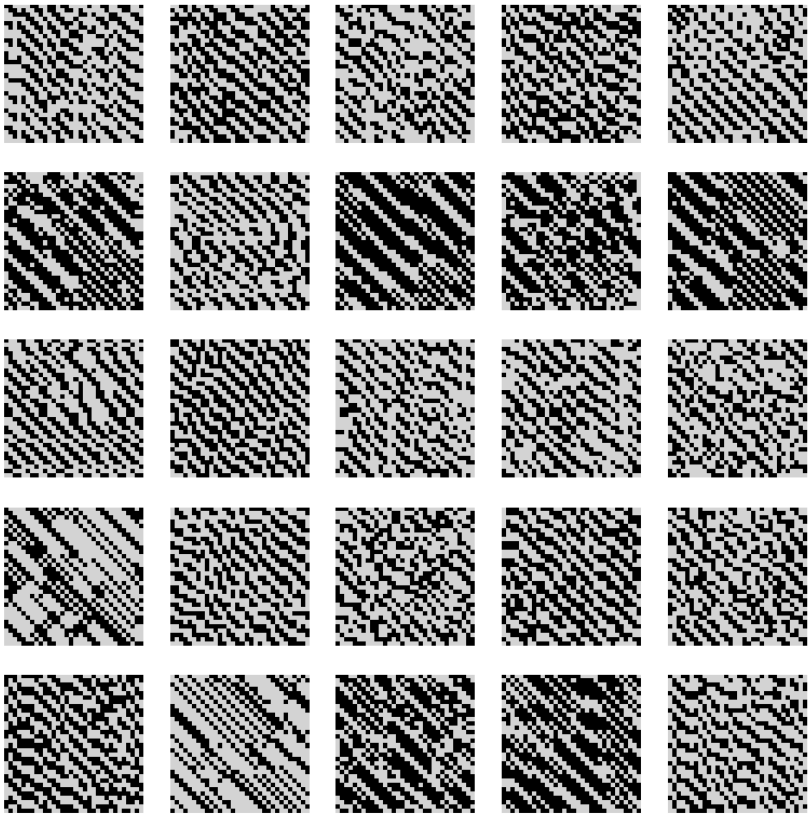
A.6. ábra. K'_6



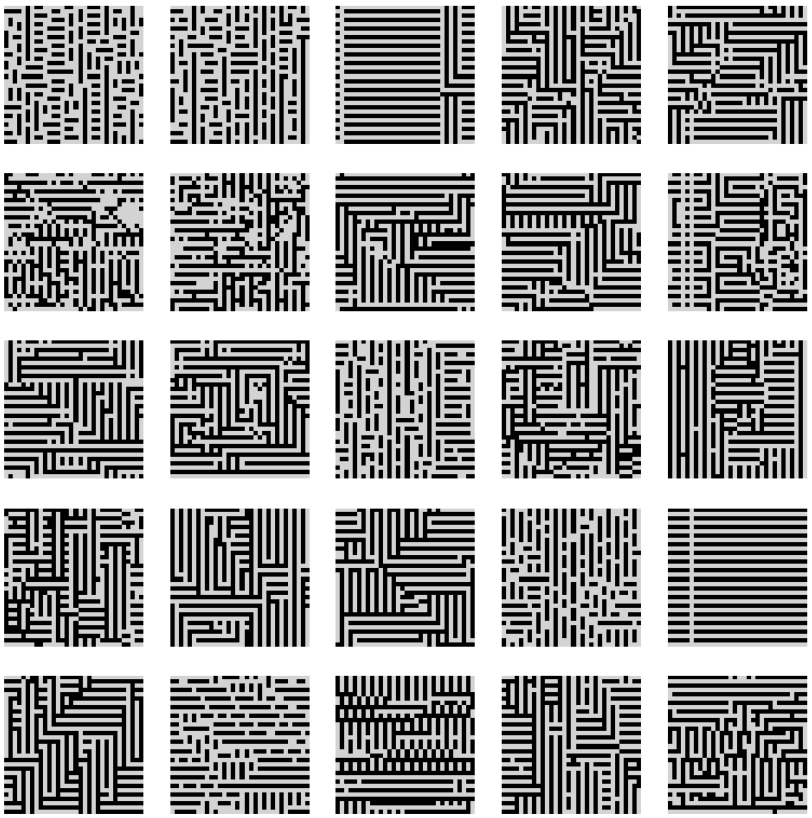
A.7. ábra. K'_7



A.8. ábra. K'_8



A.9. ábra. K'_9



A.10. ábra. K'_{10}

Klaszter index	**Reprezentáns szabályok száma**
K'_1	405
K'_2	232
K'_3	100
K'_4	119
K'_5	236
K'_6	623
K'_7	128
K'_8	497
K'_9	171
K'_{10}	49

A.1. táblázat. Reprezentáns szabályok száma klaszterenként

B. függelék

Tesztek eredményei

Klaszter	**Tesztalanyok helyes hozzárendelései**										**Százalékos eredmény**
K20	3	1	1	2	2	3	0	1	2	2	56,67%
K19	1	2	0	2	1	2	3	0	2	1	46,67%
K18	3	2	3	3	2	3	1	3	3	2	83,33%
K17	2	2	2	2	2	2	3	2	3	2	73,33%
K16	3	3	3	3	3	3	3	3	3	3	100,00%
K15	1	0	0	1	0	1	1	0	1	0	16,67%
K14	2	1	2	2	1	2	0	2	2	1	50,00%
K13	2	1	1	2	1	2	1	1	2	3	53,33%
K12	0	1	1	1	0	0	1	2	1	0	23,33%
K11	1	1	2	2	1	1	1	2	1	2	46,67%
K10	0	1	1	1	0	0	1	2	1	0	23,33%
K9	1	0	0	1	0	1	1	1	1	0	20,00%
K8	3	1	3	2	2	3	1	2	2	3	73,33%
K7	3	2	3	3	2	2	3	3	3	2	86,67%
K6	2	1	0	2	1	2	2	1	0	2	43,33%
K5	3	2	3	3	2	3	3	3	3	2	90,00%
K4	3	1	2	2	2	3	1	3	2	2	70,00%
K3	0	0	1	1	0	0	2	1	3	0	26,67%
K2	3	3	3	3	3	3	3	3	3	3	100,00%
K1	3	2	1	2	2	3	2	1	2	1	63,33%

Irodalomjegyzék

[1] Wolfram, Stephen. *A New Kind of Science*. English. Wolfram Media, 2002. ISBN: 1579550088. URL: https://www.wolframscience.com.

[2] J.L. Schiff. *Cellular Automata: A Discrete View of the World*. Wiley Series in Discrete Mathematics & Optimization. Wiley, 2011. ISBN: 9781118030639. URL: https://books.google.hu/books?id=uXJC2C2sRbIC.

[3] I. Bialynicki-Birula és I. Bialynicka-Birula. *Modeling Reality: How Computers Mirror Life*. OUP Oxford, 2004. ISBN: 9780191523991. URL: https://books.google.hu/books?id=GwpREAAAQBAJ.

[4] Absalom E. Ezugwu és tsai. „A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects". *Engineering Applications of Artificial Intelligence* 110 (2022), 104743. old. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2022.104743. URL: https://www.sciencedirect.com/science/article/pii/S095219762200046X.

[5] T. Ceccherini-Silberstein és M. Coornaert. *Cellular Automata and Groups*. Springer Monographs in Mathematics. Springer Berlin Heidelberg, 2010. ISBN: 9783642140341. URL: https://books.google.hu/books?id=N-LSFFaHTKwC.

[6] A. Bondy és U.S.R. Murty. *Graph Theory*. Graduate Texts in Mathematics. Springer London, 2009. ISBN: 9781848006638. URL: https://books.google.hu/books?id=5Z71jwEACAAJ.

[7] F. Harary. *Graph Theory*. Addison Wesley series in mathematics. Addison-Wesley, 1971. URL: https://books.google.hu/books?id=q8OWtwEACAAJ.

[8] Vladimir Rosenfeld. „The spectrum of the vertex quadrangulation of a 4-regular toroidal graph and beyond". *Journal of Mathematical Chemistry* 59 (2021. jún.). DOI: 10.1007/s10910-021-01254-2.

[9] Puneet Sharma. „Dihedral Group D4—A New Feature Extraction Algorithm”. *Symmetry* 12 (2020. ápr.), 548. old. DOI: 10.3390/sym12040548.

[10] T.W. Hungerford. *Algebra*. Graduate texts in mathematics. Holt, Rinehart és Winston, 1974. ISBN: 9780030860782. URL: https://books.google.hu/books?id=KvruAAAAMAAJ.

[11] Peter Kietzmann, Thomas C. Schmidt és Matthias Wählisch. „A Guideline on Pseudorandom Number Generation (PRNG) in the IoT”. *ACM Computing Surveys* 54.6 (2021. júl.), 1–38. ISSN: 1557-7341. DOI: 10.1145/3453159. URL: http://dx.doi.org/10.1145/3453159.

[12] Ken Perlin. „An image synthesizer”. *ACM Siggraph Computer Graphics* 19.3 (1985), 287–296. old.

[13] Zhou Wang és Alan C. Bovik. „Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures”. *IEEE Signal Processing Magazine* 26.1 (2009), 98–117. old. DOI: 10.1109/MSP.2008.930649.

[14] Zhou Wang és tsai. „Image quality assessment: from error visibility to structural similarity”. *IEEE Transactions on Image Processing* 13.4 (2004), 600–612. old. DOI: 10.1109/TIP.2003.819861.

[15] Feng Zhao, Qingming Huang és Wen Gao. „Image Matching by Normalized Cross-Correlation”. 2. köt. 2006. jún., II –II. old. DOI: 10.1109/ICASSP.2006.1660446.

[16] Zhou Wang, Ligang Lu és Alan C. Bovik. „Video quality assessment based on structural distortion measurement”. *Signal Processing: Image Communication* 19.2 (2004), 121–132. old. ISSN: 0923-5965. DOI: https://doi.org/10.1016/S0923-5965(03)00076-6. URL: https://www.sciencedirect.com/science/article/pii/S0923596503000766.

[17] Mryka Hall-Beyer. „GLCM Texture: A Tutorial v. 3.0 March 2017”. (2017. márc.). DOI: 10.13140/RG.2.2.12424.21767.

[18] Yateng Bai és Xiaoping Ma. „Coal quality prediction based on multi-feature fusion of flotation foam images”. (2020. máj.). DOI: 10.21203/rs.3.rs-29623/v1.

[19] M. Parsian. *Data Algorithms with Spark*. O'Reilly Media, 2022. ISBN: 9781492082330. URL: https://books.google.hu/books?id=VUdpEAAAQBAJ.

Ábrák jegyzéke

Táblázatok jegyzéke