A Magyar Tudomány Napja Erdélyben
3. Matematika és Informatika Alkalmazásokkal Konferencia

GRANULÁRIS ANYAGOK SZIMULÁCIÓJA PENTAMER RÉSZECSKE-MODELLEL

Témavezető:
DR. LIBÁL ANDRÁS

BABEŞ-BOLYAI TUDOMÁNYEGYETEM,
MATEMATIKA ÉS INFORMATIKA KAR,
MAGYAR MATEMATIKA ÉS
INFORMATIKA INTÉZET

Szerző:
ZÖLDE ATTILA

BABEŞ-BOLYAI TUDOMÁNYEGYETEM,
MATEMATIKA ÉS INFORMATIKA KAR,
INFORMATIKAI RENDSZEREK
OPTIMALIZÁLÁSA SZAK,
MESTERKÉPZÉS, 2. ÉV

Tudományos kollaborátor:
DR. YAIR SHOKEF

TEL-AVIV UNIVERSITY, ISRAEL,
FACULTY OF ENGINEERING,
SCHOOL OF MECHANICAL ENGINEERING

Kolozsvár, 2012. November 9-11.

TARTALOMJEGYZÉK

1 GRANULÁRIS ANYAGOK

A granuláris anyagok mindenhol jelen vannak a minket körülvevő világban. Jelentős szerepet játszanak a mezőgazdaságban (növényvédő és rovarirtó szerek), az iparban (homok, cement), sőt még a gasztronómiában is (cukor, borsó, burgonya). A teljesítmény orientált világunkban kulcsfontosságú ezek hatékony szállítása, tárolása és feldolgozása. Ez a magyarázat arra, hogy már évtizedek óta folynak kutatások a granuláris anyagok területén.[1]

Ezek olyan rendszerek, amelyek nagyszámú (10^4-10^{15}), makroszkopikus (10 μm - 10 m) részecskéből állnak; más néven szemcsés anyagoknak is nevezzük őket. Ezekre a rendszerekre láthatunk példát az 1-es ábrán: műanyag labdák, kavics, lencse és szezámmag.[2] Fizikai leírásuk egyáltalán nem triviális, mivel teljesen másképp viselkednek, mint a folytonos közegként tekinthető anyagok. Ez azért van, mert a rendszert alkotó szemcsék mérete elég nagy ahhoz, hogy a hőmozgásnak ne legyen jelentős szerepe. Így a rendszert alkotó részecskékre az energia-eloszlás nem olyan egyenletes, mint a termodinamikában megszokott rendszerek esetén, vagyis a hőmozgás általgoló, homogenizáló szerepe nem érvényesül; a granuláris anyagokra jellemzőek a stabilan fenntartható heterogén állapotok.

1. ábra: Példák granuláris anyagokra:
műanyag labdák, kavics, lencse és szezámmag

Azt az állapotot, amelyben a részecskék annyira sűrűn helyezkednek el, hogy az önálló részecskék mozgása nem lehetséges, illetve csak akkor lehetséges, hogyha egyszerre nagyon sok részecske mozdul meg, 'jammed' állapotnak nevezzük. A granuláris anyagok ebben az állapotban szilárd testként viselkednek. Ebből az állapotból a hőmérséklet növelésével, illetve külső erőhatással lehet kihozni a rendszert, például földrengés (rázás) hatására az eddig

szilárd anyagként viselkedő homok alap egy ház alatt ismét folyékonnyá válhat, és így a ház elsüllyedhet benne.

2 GRANULÁRIS ANYAG-MODELL

A legegyszerűbb granuláris anyagokat szimuláló modell gömb (illetve két dimenzióban kör) alakú részecskékkel dolgozik. Ebben a modellben az állapotok, noha stabil konfigurációknak néznek ki, valójában csak metastabil állapotok, mert ha bizonyos erővel hatunk rá (rázás, nyírás) átmennek a lehető legsűrűbb konfigurációba, a hcp (hexagonal close packed) illetve két dimenzió esetén, a háromszögrácsba.

A valódi granuláris anyagok esetén a részecskék közötti súrlódás miatt fennmaradhatnak nem szabályos állapotok is. Az ilyen 'jammed' állapotok ellenállnak bizonyos mértékű erőhatásoknak. Ezen állapotok stabilitását hőmérséklet, sűrűség és külső erőhatás függvényében a Liu-Nagel fázisdiagram írja le.[3] Így a valódi granuláris anyagokra nem jellemző a modellekben megfigyelhető kristályosodás (a lehető legsűrűbb konfiguráció elérése), hanem stabil 'jammed' állapotokkal rendelkeznek, amelyeknek kisebb a sűrűsége.

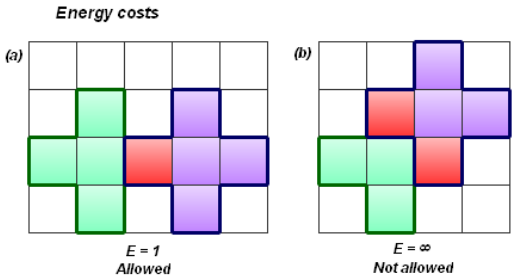
A célunk az, hogy olyan granuláris részecskemodellel dolgozzunk, ahol létrejöhetnek ilyen stabil 'jammed' állapotok, ahol nem kristályosodik ki a rendszer. A gömbökből vagy körökből álló rendszerek esetén, ahhoz hogy ilyen inhomogén állapotok létrejöjjenek, kénytelenek a rendszert két különböző nagyságú részecske keverékeként előállítani (úgy kísérletileg, mint modellezés szempontjából).

3 PENTAMER RÉSZECSKÉK

Valódi granuláris anyagok esetén sokszor a részecskék alakja nem szabályos, és a közöttük fellépő súrlódási és kontakt erők ezen valódi rendszerek esetén megengedik a stabil 'jammed' állapotok kialakulását. Mi egy olyan modellt szeretnénk felépíteni, amelyben ezek az állapotok természetes módon létrejöhetnek. Ezért választottuk a pentamer alakú részecskét, amelyek homogén részecske méret mellett is lehetővé teszik ennek a viselkedésnek a reprodukálását. Ebben a modellben a részecskék öt, kereszt alakban elhelyezett négyzetből állnak és egy négyzetrácson mozoghatnak. Ezt a modell Eisenber és Baram javasolta 2000-ben.[4]

4 HARD ÉS SOFT PENTAMER RÉSZECSKÉK

Mi úgy fejlesztettük tovább ezt a merev részecskékből álló modellt (hard), hogy megengedjük azt, hogy a részecskék részlegesen átfedjék egymást (egy bizonyos energiaköltég mellett). Ezek a soft pentamer részecskék a következő szabály szerint fedhetik át egymást.



2. ábra: Átfedések energiaköltségei

a). Két pentamer részecske egymásra csúszása egyetlen négyzet átfedésével. Ezeket az átfedéseket megengedjük a soft rendszer eseten. b). Pentamerek kettős átfedése. Ez az egyszerű modellben tiltott.

Csak a pentamer 4 lábán történhet átfedés, ami egyértelműen kizárja a teljes átfedést. Továbbá 2 pentamernek legtöbb 1-1 lába fedheti át egymást. Nem megengedett, hogy egyszerre 2-2 láb legyen átfedésben. Erre azért van szükség, hogy a modell minél egyszerűbb legyen. Látható, hogy egy pentamerre nézve a maximális átfedések száma 4, amihez 4 másik pentamerre van szükség.

Az egymásra csúszás energiaköltséggel jár, ami befolyásolja az esemény bekövetkezésének valószínűségét (amikor a részecskéket mozgatjuk, vagy letesszük). Az energia mellett a hőmérséklet is befolyásolja ezt a valószínűséget a Boltzmann eloszlásnak megfelelően:

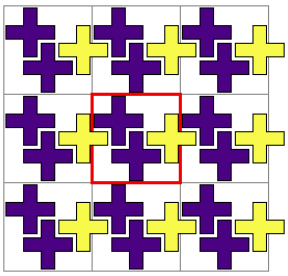
$$P(A) = e^{-\frac{E}{T}}$$

Ez a hőmérséklet a szimuláció során kap jelentőséget, és az annak felel meg, hogy a rendszer mennyire tudja kipróbálni a magasabb energiával járó állapotokat.

5 SZIMULÁCIÓ SOFT PENTAMER RÉSZECSKÉKKEL

A szimuláció egy Monte Carlo szimuláció. A kezdeti állapotban a szimuláció terét alkotó négyzetrács üres. Annak érdekében, hogy ez a tér ne táguljon ki a szimuláció során, biztosítjuk a periodikus határfeltételt (periodic boundary condition). Ezt úgy érjük el, hogy a négyzetrács köré elhelyezzük annak másolatait. Így a négyzetrács szélén elhelyezkedő

pentamer lábai, melyek egyik oldalon a rácson kívülre kerülnének, megjelennek a másik oldalon.

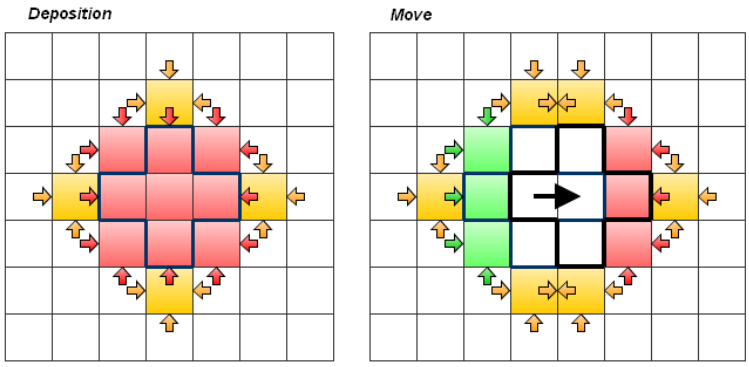


3. ábra: Periodikus határfeltételek a pentamer rendszer esetén
A rendszert körbevesszük annak másolataival, így az egyik oldalon kilógó részecskék megjelennek a szemközti oldalon.

Mielőtt részleteznénk a szimuláció lépéseit, nézzük meg, hogy milyen következményekkel jár egy pentamer elhelyezése, illetve elmozgatása.

Elhelyezés (deposition) után annak a valószínűsége, hogy a piros háttérrel jelölt cellára kerüljön egy újabb pentamer középpontja, nulla (4-es ábra). Ezekre a cellákra már nem helyezhetünk el több pentamert. Ugyanígy a piros nyilakkal jelölt elmozgatások valószínűsége is nulla lesz. A sárga háttérrel jelölt cellákra való elhelyezéshez szükséges energia egy egységgel megváltozik. Ennek következtében az ezekre a cellákra való elhelyezés Boltzmann képlet szerint számolt valószínűsége is megváltozik. Ugyanígy megváltozik a sárga nyilakkal jelölt elmozgatások valószínűsége is, mert ebben az esetben is más átfedésekkel kell számoljunk az illető mozdulathoz.

Elmozgatás (move) esetén hasonló a helyzet, azonban megtörténhet, hogy felszabadulnak bizonyos helyek, esetleg már rég eltörölt elmozgatások lesznek újra lehetségesek. A zöld háttérrel jelölt cellák felszabadult helyeket jelentenek: ezekre a cellákra újabb pentamereket helyezhetünk el akkor, hogyha a szomszédos cellákban található pentamerek ezt megengedik. A zöld nyilakkal jelölt elmozgatások szintén felszabadulhatnak, azonban itt is figyelembe kell vennünk a szomszédos pentamereket. A zöld, sárga és piros nyilak eseten az ábra megmutatja az összes lehetőséget ami előfordulhat egy ilyen általános esetben, azonban ezt a szimulációban egy specifikus helyzetre alkalmazva, nem minden nyíl menten változik meg az elmozdulás valószínűsége, mivel nem minden helyen van egy pentamer amely elmozdulhat. Így egy specifikus esetben csak azoknak az elmozdulásoknak a valószínűsége változik meg ahol már jelen volt egy pentamer és az illető elmozdulás el volt tárolva a mozgatások listájában (lásd a 6. fejezetnél).



4. ábra: Az elhelyezés/elmozgatás hatása a szomszédos pentamerekre

a) Egy új pentamer elhelyezésekor létrejövő változások: a piros négyzetekre nem lehet többé pentamert tenni, a piros nyilak menten tilossá válik a mozgatás illetve a sárga négyzeteken változik a depozíció valószínűsége, a sárga nyilak menten változik a mozgatás valószínűsége

b) Egy mozgatáskor létrejövő változások: a piros és sárga négyzetek illetve nyilak a fent említett jelentéssel bírnak, míg a zöld négyzetek esetén felszabadulhat a depozíció, illetve a zöld nyilak menten szabaddá válhatnak elmozdulások

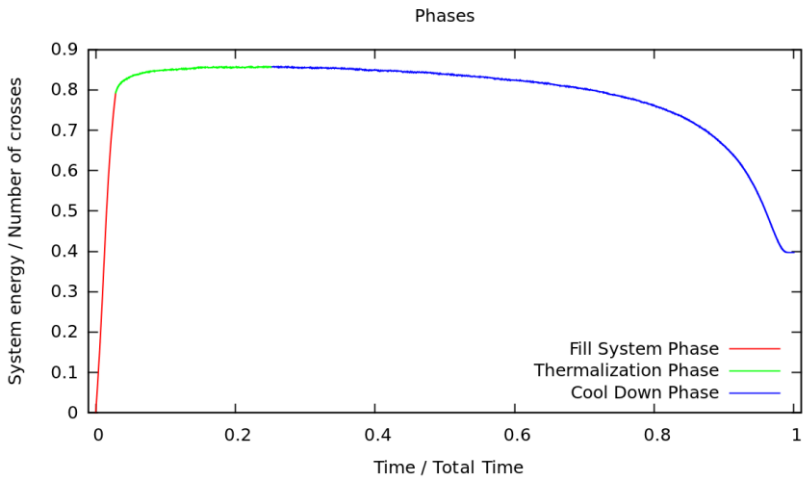
A szimuláció három fázisból áll. Először szükséges feltölteni a rendszert, majd egy termalizálási szakasz következik, végül a hűtési folyamat juttatja el a rendszert egy stabil alapállapotba.

Feltöltés (Fill System): A cél egy előre meghatározott sűrűség (ρ) elérése egy adott hőmérsékleten (T_0). Ezt új pentamerek elhelyezésével, illetve a már elhelyezettek elmozgatásával érjük el. A folyamat során az elhelyezés egy adott p valószínűséggel fog bekövetkezni, míg az elmozgatás $(1-p)$ valószínűséggel. Egy újabb pentamer pozíciójának meghatározása véletlenszerűen történik, minden egyes cella (amely a pentamer középpontját határozza meg) ugyanolyan valószínűséggel kerülhet kiválasztásra. Abban az esetben, ha egy már foglalt cella kerül újra kiválasztásra, a lépést visszautasítjuk (rejection). Hasonló a helyzet az elmozgatásnál is. Szintén egyenletes valószínűséggel történik a kiválasztás, és abban az esetben, ha a választott elmozdulás nem megvalósítható, ugyancsak visszautasítjuk a lépést.

Termalizálás (Thermalization): A kezdeti hőmérsékleten tartva a rendszert, véletlenszerű elmozgatásokat végzünk. Erre a köztes lépésre azért van szükség, hogy a rendszer energiájának kiegyensúlyozását biztosítsuk mielőtt megkezdődne a hűtési folyamat.

Hűtés (CoolDown): A rendszer hőmérsékletének hűtésével egyidőben, a termalizálási fázishoz hasonlóan, szintén véletlenszerű elmozgatásokat végzünk. A hőmérséklet hűtése lineáris, egy hűtési arány (r) határozza meg a lehűlés gyorsaságát ($T = T_0 - r \cdot time$). Ez a

legidőigényesebb fázis mind közül. A cél a rendszer energiájának tanulmányozása a hőmérséklet függvényében.



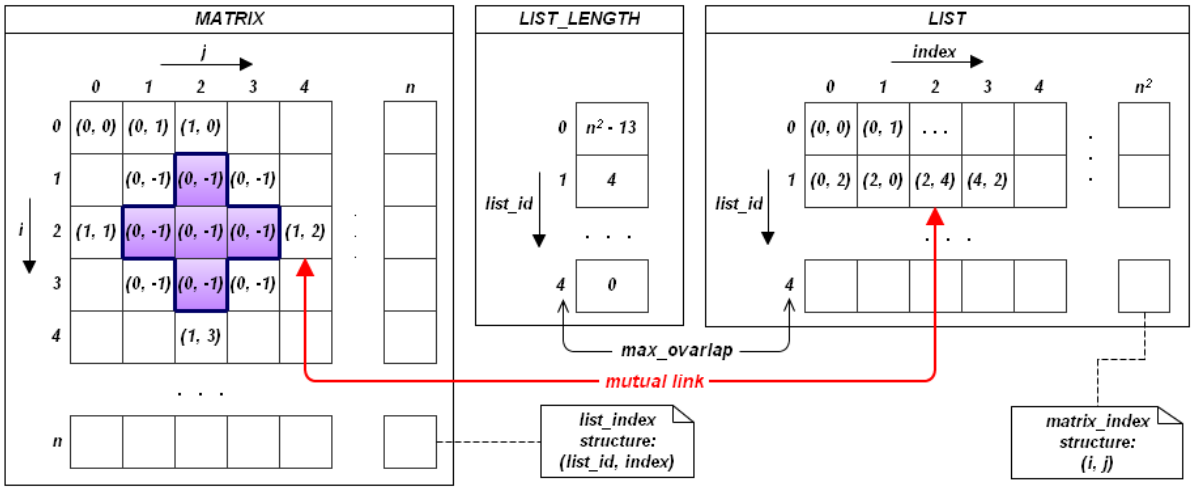
5. ábra: A szimuláció fázisai

A rendszer energiája idő függvényében. A piros a rendszer feltöltése, a zöld a termalizálási szakasz, a kék a hűtési szakasz közben történő energiaváltozás.

6 AZ ALKALMAZÁS

Röviden összefoglalva a szimuláció paraméterei a következők: N – rendszerméret, T_0 – kezdeti hőmérséklet, ρ – az elérni kívánt sűrűség (a sűrűség értékei 0 és 0.2 között mozoghat, mivel a következőképpen számoljuk ki: *pentamerek_száma* / N^2), p – elhelyezés valószínűsége, r – hűtési arány.

Az accept-reject algoritmus hátránya a sűrűség növekedésével jelentkezik, mivel egyre több olyan lépés generálódik, melyet el kell utasítani. Ez nagyban megnöveli a program futási idejét. Az optimalizálás érdekében egy rejection-free algoritmust implementáltunk. Az accept-reject algoritmus egyetlen előnye, hogy egy új elhelyezés illetve elmozdulás meghatározása konstans idő alatt történnik. Hogy elkerüljük az elutasításokkal járó fölösleges műveleteket, de mégis megtartsuk az accept-reject algoritmus előnyét, szükség volt egy olyan adatszerkezet megtervezésére, mely lehetőséget biztosít a lehetséges szabad helyek / elmozdulások tárolására, ugyanakkor konstans időn belől egy új meghatározására. Ha csak a rendelkezésre álló szabad helyekből illetve elmozgatásokból választjuk ki az újat, egyértelműen egyetlen lépést sem kell elutasítanunk. Figyelembe kell vennünk azt, hogy egy pentamerre nézve a maximális átfedések száma 4. Továbbá azt is, hogy az átfedésekkel járó energia függvényében, az adott esemény különböző valószínűséggel következhet be. Az adatszerkezet, mely eleget tesz a fenti elvárásoknak, a következőképpen épül fel.



6. ábra: Szabad helyeket tároló adatszerkezet

A *list_index* struktúrákból álló *Matrix* és a *matrix_index* struktúrákból álló *List* közötti kölcsönös egymásra hivatkozás. A *List_length* a listák aktuális hosszait tárolja.

Három fő kompenens alkotja.

Matrix: Ez egy *N* x *N* - es mátrix, mely *list_index* típusú struktúrákból áll. A struktúra két adattagból tevődik össze (*list_id* és *index*), mely pontosan meghatároz egy listát, azon belül pedig egy konkrét elemet. A negatív index (-1) azt jelzi, hogy az elem nem található meg egyetlen listában sem. Esetünkben *n* a rendszer méretét adja meg. A mátrixra azért van szükség, hogy egyszerűen meg tudjuk határozni az elhelyezésnél/elmozgatásnál tárgyalt szomszédos cellákat.

List_length: Egy 5 elemből álló lista, mely a listák aktuális hosszát tárolja.

List: Öt listát foglal magában, melyek *matrix_index* típusú elemekből állank. Ez a struktúra szintén két adattagot tartalmaz (*i* es *j*), melyek a mátrix egy cellájára hivatkoznak. Ahogy a piros nyíl is jelzi, létrejön a *matrix* és a *list* között egy kölcsönösen egymásra való hivatkozás. A listák maximális hossza N^2. Mindegyik lista olyan *matrix_index*-eket tartalmaz, melyekre ha el szeretnénk helyezni egy pentamert pontosan a *list_id* értékével megegyező energiára van szükségünk.

Egy elem hozzáadása illetve törlése esetén meg kell őrizni az értékek helyességét. Egy új elem kiválasztása a következőképpen történik. Generálunk egy véletlenszámot 0 és $\sum_{i=0}^{4} list\_length_i \cdot e^{-\frac{i}{T}}$ között, mely meghatároz egy *list_id*-t, majd egy másikat 0 és $list\_length_{list\_id}$ között mely meghatároz egy *index*-et. Ez a páros egyértelműen beazonosítja a kiválasztott elemet és ez a műveletsor konstans idő alatt elvégezhető.

A mozgatásokat tároló adatszerkezet hasonló felépítésű. A mátrix illetve a lista kibővül egy irányt (direction) meghatározó adataggal.

Maga az alkalmazás C++ programozási nyelvben van implementálva és 6 jól elkülöníthető részből áll (lást 7-es ábra):

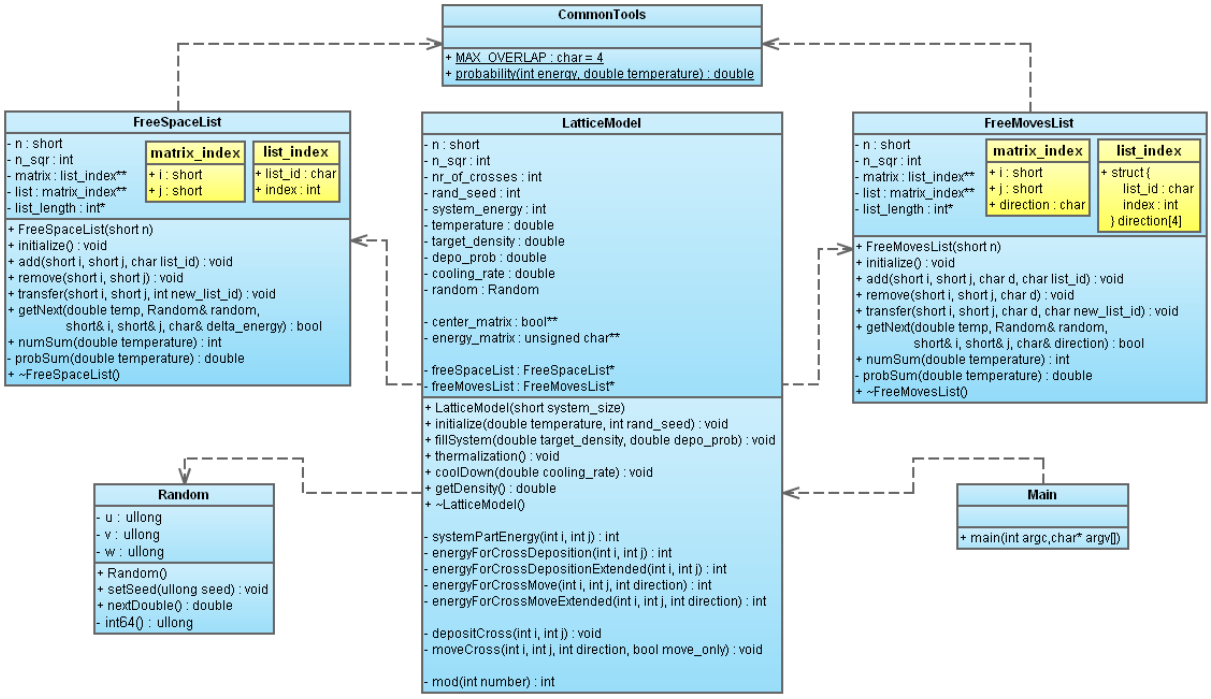
Random: Egy véletlenszám-generátor osztály, melynek nagyobb a periódusa, mint a C++ beépített véletlenszám-generátoráénak és gyorsabb is annál.[5]

FreeSpaceList és *FreeMoveList*: A szabad helyeket, illetve mozgásokat tároló adatszerkezetek. Felépítésükben nagyon hasonlóak, mindkét osztály ugyanazokata függvényeket implementálja. Ezen függvények segítségével beszúrhatunk, törölhetünk vagy akár lekérhetünk egy újabb elemet. A műveletek bonyolultsága O(1) (konstans futási idejű függvények).

CommonTools: A két adatszerkezet által közösen használt funkcionalitásokat biztosítja, ilyen például egy esemény valószínűségének kiszámítása.

LatticeModel: Magát a szimuláció terét biztosító osztály. Tartalmazza a pentamerek manipulálásához szükséges metódusokat (elhelyezés, elmozgatás) és a szimuláció magját képező 3 fázis implementációját.

Main: A fő függvényt tartalmazó osztály. Gondoskodik a bemeneti adatok helyességéről és a szimulációt végző függvények meghívásáról az adott paraméterekkel.



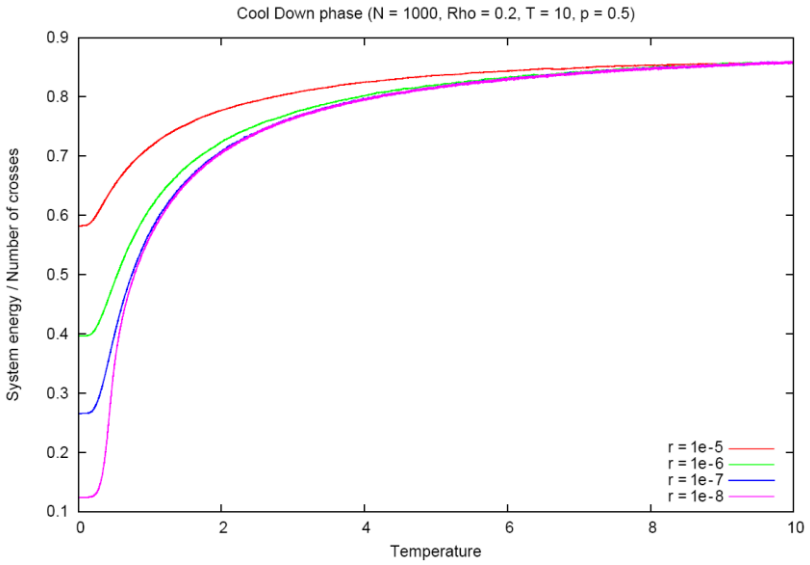
7. ábra: *CrossesModel* - Osztálydiagram

A programban szereplő osztályok: a négyzetrácsot, a mozgatásokat és szabad helyeket tartalmazó osztály, illetve a többi rész kapcsolata és felépítése

7 EREDMÉNYEK

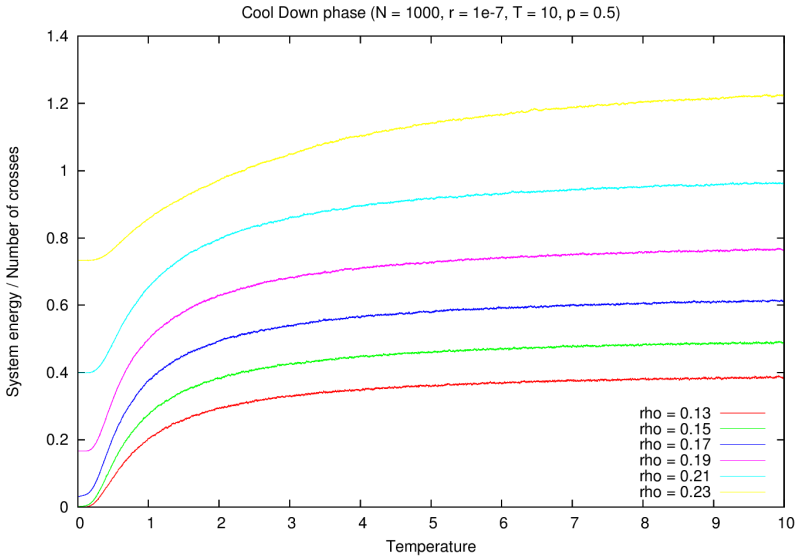
A hőmérsékletet ha nullára állítjuk, visszakapjuk az eredeti hard modellt. Ebben az esetben nem érhető el tetszőleges sűrűség. p = 1 esetén egy adott sűrűségnél a rendszer eléri a 'jammed' állapotot. Ez a határ ρ = 0.139-nél található. A p paraméter csökkentésével egyre nagyobb sűrűséget ér el a rendszer, p = 0.1 esetén már ρ = 0.172. Ezek az eredmények megegyeznek az eddig közölt eredményekkel.

A hőmérséklet növelésével tetszőleges sűrűséget elérhetünk (a részecskék egymásra csúszása miatt). Itt fontos szerepet kap a hűtési folyamat, azt vizsgáljuk, hogy a különböző hűtési arányok függvényében milyen hőmérsékleten, illetve milyen energiaszinteken fagy be a rendszer ('jammed'). Jelenleg a legoptimálisabb rendszerméret N = 1000, mivel ez már elég jó statisztikákat biztosít, de még nem túl hosszú a futási idő. A 8-as ábrán különböző r értékekre látható a hűtési folyamat eredménye. A 9-es ábrán pedig különböző ρ értékekre.



8. ábra: Hűtés, különböző r-re.

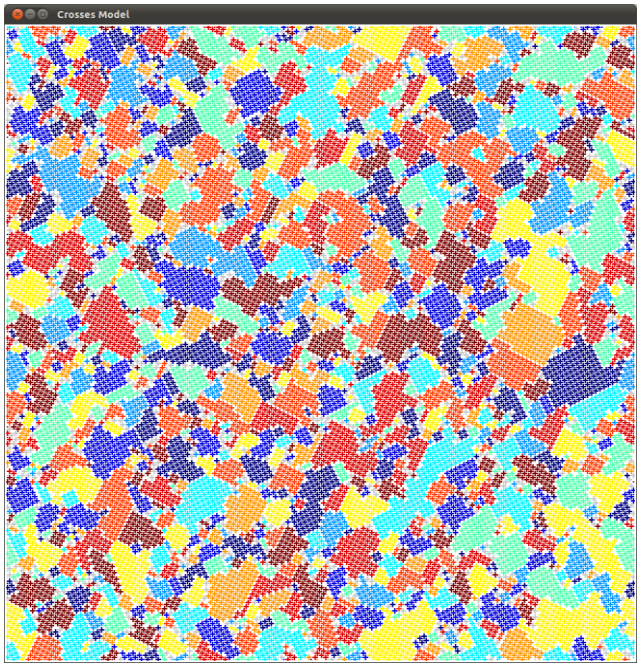
A rendszer energiája a hőmérséklet függvényében. A különböző színek különböző hűtési arányt jelölnek.



9. ábra: Hűtés, különböző ρ-ra

A rendszer energiája a hőmérséklet függvényében. A különböző színek különböző sűrűséget jelölnek.

A szemléletesség kedvéért a 10-es ábrán egy kisebb méretű rendszer látható, amelyben kristályok alakultak ki. A különböző színek különböző kristályokat jelölnek. A hideg és meleg színek a kristályok kiralitását különböztetik meg. Ez azt jelenti, hogy a meleg színek esetén a szomszédos pentamerek x irányban 1 és y irányban 2 távolságra vannak az aktuális pentamertől. Hideg színek eseten pedig x irányban 2 és y irányban 1 távolságra.



10. ábra: Apróbb kristályok egy rendszeren belül

A különböző színek különböző kiralitással rendelkező és különböző pozíciót elfogalaló kristályszemcséket jelölnek (meleg színek 1:2 kiralitast, a hideg színek 2:1 kiralitást jelzik).

8 KÖVETKEZTETÉS

A jól megtervezett adatszerkezeteknek és hatékony algoritmusoknak köszönhetően az eddig szimulált rendszerekhez képest több nagyságrend előrelépést értünk el, ennek következtében sokkal jobb statisztikákat is.

Van egy működőképes szimulációnk/modellünk granuláris anyagokra, amivel különböző statikus és majd a későbbiek során dinamikus tulajdonságokat fogunk vizsgálni.

9 IRODALOMJEGYZÉK

1. Tegzes Pál, Koltai János, Ábel Dániel *Granuláris anyagok* Egyetemi jegyzet ELTE (2010)

2. http://it.wikipedia.org/wiki/File:Granular_matter_examples.PNG

3. A. J. Liu, S. R. Nagel *Jamming is not just cool any more*, Nature **396**, 21 (1998)

4. E Eisenberg, A Baram *Random closest packing in a 2D lattice model*
J. Phys. A **33**, 1729-34 (2000)

5. W. Press, et. al. *Numerical Recipies*, page 341-343, Cambridge University Press (2007)