ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

**NEUMANN JÁNOS
INFORMATIKAI KAR**

5. sz. melléklet

SZAKDOLGOZAT

OE-NIK Hallgató neve: **Bolya Ádám Bence**

2022 Hallgató törzskönyvi száma: **T/005580/FI12904/N**

A feladat

Sejt- és tumorbiológia jelenségek kutatása során egyre gyakrabban használnak számítógépes modelleket egy-egy jelenség alaposabb megértéséhez. A vizsgált probléma jellegétől függően ilyen számítógépes megközelítés lehet például az ágens-alapú szimuláció, amikor a rendszert alkotó egyedek (sejtek) jellemzőit és adott körülmények melletti viselkedésüket az őket reprezentáló ágensek egyedi paraméterek beállításával lehet elérni. A sejtek különféle életjelenségeinek (pl. növekedés, osztódás, sejthalál) ágens alapú modellbe építésére jelenleg több megközelítés ismert, melyek implementálása sokszor komoly szakmai felkészültséget kíván a felhasználótól.
A szakdolgozat célja egy olyan ágens-alapú sejtszimulációs környezet megtervezése, implementálása és tesztelése, ahol a sejtek különféle életjelenségei az ágensek tetszőleges, a felhasználó által előre definiálható állapotainak, valamint azok között előre megadott szabályok szerint bekövetkező átmeneteknek felelnek meg. Az elkészített szimulációs környezet alkalmas kell legyen néhány, az irodalomban fellelhető megközelítés megvalósítására is.

A dolgozatnak tartalmaznia kell:

- az ágens-alapú sejtbiológiai modellezés fontosabb irodalmi forrásainak rövid összegzését, illetve néhány hasonló célú modell bemutatását és elemzését,
- az ágensek állapotainak és állapotátmeneteinek reprezentálására alkalmas megközelítések összefoglalását, valamint a kiválasztott állapotmodell ismertetését,
- az elkészítendő szimulációs környezet megtervezését, az alkalmazott részmodellek bemutatását és szoftveres implementációjuk részletes ismertetését,
- az elkészült szimulációs szoftver tesztelését, illetve egy szabadon választott, az irodalomban elérhető állapotmodell megvalósításának vizsgálatát,
- a rendszer lehetséges továbbfejlesztési irányainak ismertetését.

ÓBUDAI EGYETEM * Neumann János Informatikai Kar * 1. Budapest

Ph.

.....
Dr. Vámossy Zoltán
intézetigazgató

A szakdolgozat elévülésének határideje: **2024. május 15.**
(OE TVSz 55.§ szerint)

A dolgozatot beadásra alkalmasnak tartom:

.....
külső konzulens

Kiss Dániel
.....
intézményi konzulens

ÓBUDAI EGYETEM
ÓBUDA UNIVERSITY

Neumann János Informatikai Kar

7. sz. melléklet

HALLGATÓI NYILATKOZAT

Alulírott hallgató kijelentem, hogy a szakdolgozat saját munkám eredménye, a felhasznált szakirodalmat és eszközöket azonosíthatóan közöltem. Az elkészült szakdolgozatomban található eredményeket az egyetem és a feladatot kiíró intézmény saját céljára térítés nélkül felhasználhatja.

Budapest, 2022. május 13.

Bolya Ádám
.....
hallgató aláírása

Tartalomjegyzék

1. BEVEZETÉS

Manapság a szimulációk világában élünk: például a legtöbb cég nem kezd bele bonyolultabb üzleti manőverbe előzetes számítások, de akár komplexebb előrejelző modellek futtatása, és az eredmények alapos elemzése nélkül. Hasonló modellek és szimulációk természetesen nem csak a gazdasági életben, hanem mára szinte minden természettudományos és műszaki területen jelen vannak. Az utóbbi évtizedek egyik legdinamikusabban fejlődő ága a számítógépes biológia, amely az élő rendszerek különböző szerveződési szintjein megfigyelhető jelenségekre keres válaszokat számítógépes modellek használatával.

A számítógépes biológiában gyakran alkalmazott megközelítések egyike az ágensalapú modellezés, amely persze nem kizárólag ezen a tudományterületen kerül elő. A technikával először egy dokumentumfilmben találkoztam. Az epizód egy felhőkarcoló tervezését tárgyalta, ahol a fő szempont a gyors kimenekítés volt esetleges vészhelyzet (például tűzvész) esetén. Az embereket szimbolizáló ágensek a bennük definiált szabályrendszernek megfelelően mindig a legrövidebb útvonalon próbáltak kijutni az épületből. Ahol az útvonal összeszűkült, tömeg alakult ki, így lelassult a kimenekítés. A szimulációs eredmények alapján később a mérnököknek lehetősége volt a terveken úgy korrigálni, hogy a kimenekítés hatékonyabban mehessen végbe. Érdekes volt látni, hogy az igen egyszerű szabályrendszerrel felruházott ágensek önmagukban optimális utat találtak, viszont egy talán nem várt következménye a kollektív viselkedésüknek, hogy tömegben akadályozták és lökdösték egymást, pontosan reprezentálva így a pánikoló embertömeget.

Az ágensalapú modellezés és szimuláció gyakorlatilag minden olyan területen használható, ahol egy jelenség megértése a rendszert alkotó részegységek kollektív viselkedésén keresztül értelmezhető. Az elmúlt évek megmutatták, hogy a megközelítés a járványterjedésben is hatékonyan alkalmazható [6]. Az ágensalapú modellekről általánosan elmondható, hogy a modellezett rendszer viselkedését elsődlegesen az ágensek jellemzői, illetve a bennük definiált szabályrendszer határozza meg. A megközelítést azonban épp az teszi hatékonnyá, hogy ezek az egyedek kölcsönhatásba lépnek a rendszer további egyedeivel, illetve az egyed környezetével is, és ezek mind módosíthatják az egyed állapotát, amely a kölcsönhatásokon keresztül kihat a többi egyedre, és így tovább. Tehát a modellezett rendszer kollektív viselkedését megérthetjük az azt felépítő egyedek tipikusan egyszerűbb viselkedése alapján, valamint előrejelzéseket tehetünk az egyedi szabályok módosításából következő komplex viselkedésre.

Sejtbiológia, tumorbiológia vagy molekuláris biológia területén gyakran használnak úgynevezett *in vitro* (tenyésztőedényben végrehajtott) kísérleteket a kutatók. A tenyésztőedényes kísérletek eredményeiből valamelyest következtetni lehet például

arra, hogy egy gyógyszerjelölt anyagnak milyen hatása lesz az élő szervezetre, mielőtt azt közvetlenül élő szervezetbe adva tesztelnék. Mivel egy számítógépes szimulációhoz képest a tenyésztőedényes kísérletek is arányaiban költségesek, így van értelme annak, hogy in vitro sejtes kísérletek előkészítéséhez számítógépes, például ágensalapú modellt felhasználva végeznek becsléseket a kutatók. A probléma ezzel viszont legtöbbször az, hogy a biológiatudományok területén dolgozó kutatók érthető módon sokszor nem rendelkeznek olyan szintű programozási ismeretekkel, amely egy ilyen modell létrehozásához szükséges. Tapasztalatok szerint gyakran annak sem látják hasznát, hogy az idejüket egy olyan létező rendszer elsajátításába fektessék, amely bármilyen szintű programozói ismeretet feltételez.

Ahogy a következő fejezetben látni fogjuk, számos ágensalapú modell érhető el a szakirodalomban, amelyek jól alkalmazhatók különféle biológiai problémák esetén. Közös jellemzőjük azonban, hogy a közzétett formában lényegében csak az adott problémára használható, és a kódbázis módosítása, de akár a szimulációk testre szabása is komoly felkészültséget kíván.

A fentiek alapján dolgozatomban azt a célt tűzöm ki, hogy létrehozzak egy specifikus programozói ismeretek nélkül is használható, intuitív felhasználói felülettel rendelkező alkalmazást, amelyben különféle tenyésztőedényes kísérleteket lehet szimulálni. Célom, hogy az edény fizikai jellemzői mellett könnyen definiálható legyenek mind a sejteket reprezentáló ágensek, mind a tenyésztőedényben jelen lévő kémiai anyagok jellemzői, és azok egymásra hatásának módja. Az irodalomban elérhető megközelítések áttekintése és elemzése után olyan modelleket valósítok meg a rendszerem komponenseiként, amelyeknél elsődleges az egyszerűség, a könnyű alkalmazhatóság a specifikussággal szemben. Az ágensek viselkedésének leírására lehetőség lesz úgynevezett állapotok egyszerű bevezetésére, amelyben a felhasználó eldöntheti, milyen szabályok szerint viselkedjen az adott állapotban lévő ágens. Ez a megközelítés nagy rugalmasságot biztosít, hiszen a rendszerben nincs előre meghatározva a szabályrendszer, így a szoftver kellően általánosan használható *in vitro* kísérletek leírására. Az állapotok reprezentálására és a közöttük történő váltások modellezésére áttekintem az elterjedtebb megközelítéseket, majd a flexibilis alkalmazhatóságot szem előtt tartva hozok döntést. A modellezett kémiai anyagok termelődésének, terjedésének és bomlásának (felvételének) matematikai modelljeinek áttekintése után ugyancsak az egyszerűséget szem előtt tartva végzem az implementációt.

Az elkészült szoftver teszteléséhez néhány, a szakirodalomban közölt biológiai kísérletet veszek alapul, és megvalósítom azok megfelelőit a saját rendszeremben. A szoftver gyakorlati alkalmazhatóságát a szimulációs eredmények kvalitatív és kvantitatív összevetésével állapítom meg.

2. ÁGENSALAPÚ SZIMULÁCIÓS MÓDSZEREK

Egyed vagy ágens alapú modelleket több tudományág szimulációinál használnak. A legfőbb felhasználási terület a biológia. Egyed alapú modellel szimulálható például járványok terjedése, állat-növény interakciók, populációk evolúciós változása, a mikrobiológia területén a génkifejezés vagy az immunrendszer. Egyed alapú modelleket használnak közgazdasági szimulációknál is.

2.1. Az egyed jellemzői

Az egyed alapú modellek középpontja az egyed maga és az egyedek sokasága.

Az egyedekről elmondható, hogy van saját működésük, nem csupán adattagok. A működésük szabályokon alapszik. Ezek a szabályok lehetnek egyszerű switch-case szerkezetek, de bonyolultabb esetekben általában differenciálegyenletek.

Az egyedek általában csak a mikrokörnyezetükkel lépnek aktívan interakcióba. Ez legtöbbször a szomszédos egyedeket jelenti. Ha van a szimulációban makrokörnyezet, azt az ágensek nem változtatják meg maguk (a makrokörnyezet viszont reagálhat a jelenlétükre), a környezet tulajdonságait csak érzékelni tudják.

A biológiai szimulációkban előfordulhat, hogy az egyedek szaporodnak. Ezt két módon tehetik; ha sejt szimulációról van szó, akkor osztódnak és az egyetlen szülő egyed génkészletét kapják meg az utódok, ha magasabb rendű életformát szimulálunk, akkor a szülő egyedek génkészletét valamilyen szabály alapján kombináljuk, és ezt kapják meg az utódok.

Az egyedek belő működésének kifejeződése az egyedek állapota. Az egyedek véges állapottal rendelkeznek, az állapotátmeneteket a viselkedésük határozza meg.

2.2. A makrokörnyezet

A komplexebb szimulációkban megjelenik a makrokörnyezet. Ez a környezet az egyedek mindegyikére hatást gyakorol. A makrokörnyezetet jelképezheti egy egyszerű környezeti változó pl antibiotikum koncentráció, vagy bonyolultabb szerkezet.

Bonyolultabb makrokörnyezetre akkor van szükség, ha fizikai szimuláció is társul az egyedekhez. A makrokörnyezete ilyenkor egy kontinuum tér, ami lefedi az egyedek által bejárható teret. A kontinuum környezeti értékeket differenciálegyenlettel lehet megadni. A környezet az egyenletben megadott módon változhat az egyedek jelenléte, sűrűsége stb. alapján.

2.3. Fizikai jellemzők

Előfordulhat, hogy a szimuláció során szükség van az egyedek fizikai helyének, alakjának mozgásának szimulációjára.

A legegyszerűbb esetben az egyedek valamilyen rácsszerkezeten foglalnak helyet, például két- vagy háromdimenziós mátrixban. Ez a megoldás előnyös a könnyű megvalósítás és a jól és egyszerűen párhuzamosíthatóság miatt. A szomszédság könnyen definiálható. A rácsszerkezet hátránya, hogy korlátozza a lehetséges haladási irányokat.

A másik megoldás az off-lattice szimuláció. Ilyenkor nincs rácsszerkezet, az egyedek szabadon mozoghatnak a térben. Az egyedek koordinátái kontinuum függvények, a mozgásukat differenciálegyenlettel lehet leírni. Az off-lattice módszer lehetővé teszi az egyedek formájának a definícióját. A legtöbb példában az egyedek gömb alakúak. A módszer hátránya, hogy az egyedek mozgásának és koordinátáinak kiszámítása gépigényesebb, illetve nehezebb definiálni az egyedek közötti szomszédságot.

2.4. Egyszerű modellek

Az egyszerű modellek csak a vizsgált jellemzőket tartalmazzák, általában egyszerű matematikai és evolúciós szabályok irányítják a populációt.

Chisholm és Motta a gyógyszer-rezisztencia kialakulását modellezte, Chisholm a rákos sejtekben, Motta pedig az E. coli baktériumokban.

Chisholm modellje egymástól függetlenül egy ágens alapú és egy integrál-differenciál egyenlet alapú számítást is használ [1]. Mindkét esetben a kezdő rákos sejt populáció PC9 fenotípusú. Ezt a kezdő populációt folyamatosan cytotoxikus (sejtosztódást megállító) gyógyszerrel kezeljük.

A sejtek két fő jellemzője a túlélési esély és a szaporodási valószínűség. A PC9 típusú sejt alacsony túlélési esélyű, de gyorsan osztódik. A DTP típusnak nagy a túlélési esélye, de lassan szaporodik. A DTEP típusnak mindkét tulajdonsága magas értékű.

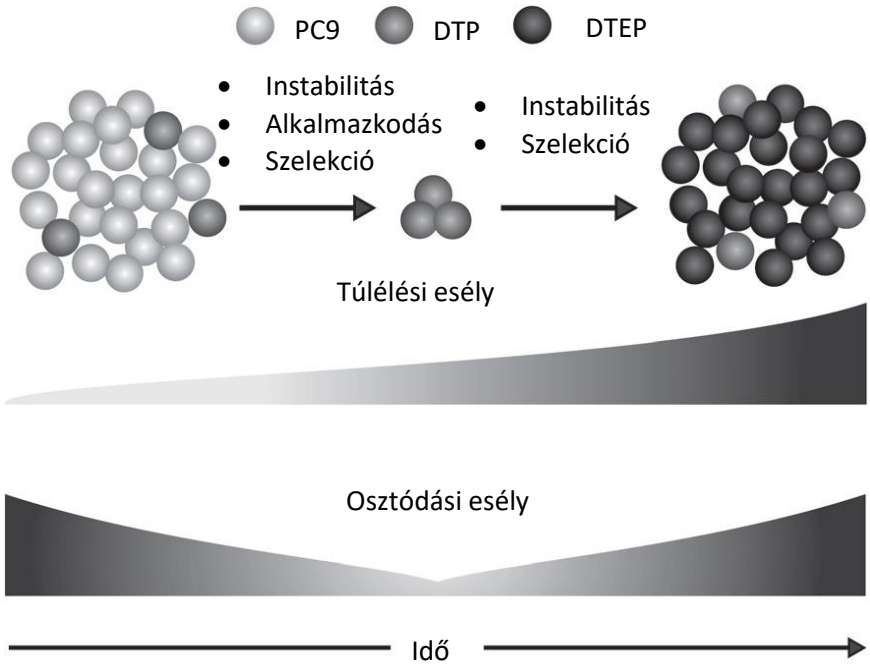
A legfőbb modellezett folyamatok: szelekció, stressz által kiváltott alkalmazkodás, nem genetikus úton történő tulajdonság változás. A szelekciót az osztódási valószínűség p() és a sejthalál valószínűség d() segítésével szimuláljuk. Ezek a valószínűségek a vizsgált sejt tulajdonságaitól és a sejt t idő-beli mikrokörnyezetétől függnek. Feltételezzük, hogy d() nem függ p()-től. Azt is feltételezzük, hogy a magas osztódási esély fenntartása költséges a sejtre nézve, így a sejt csökkenti a p() értékét. Végül feltételezzük, hogy p() függ a teljes sejtállomány nagyságától, így szimulálva a sejtek közötti versengést.

A stressz kiváltotta alkalmazkodás sebessége v() és függ a környezettől. Gyógyszer jelenléte nélkül nincs alkalmazkodás.

A nem genetikus mutációkat Brown-mozgás jelleggel szimulálták.

A szimuláció diszkrét időben történik. A sejtek minden körben osztódnak, elhalnak, vagy nyugalmi állapotban maradnak.

Integrálegyenletes formában a populáció egy háromváltozós függvény formájában létezik, ahol x a normalizált túlélési esély, y a szaporodási esély, t az idő, a függvény értéke adott pontban pedig a sejtek száma.



1. ábra. A modell eredményei egybeesnek a kísérletekben megfigyelt jelenségekkel. A sejtek PC9 fenotípusból DTP majd DTEP-vé alakulnak. Chisholm ábrájának reprodukciója [1]

Motta modellje nagymértékben hasonlít Chisholméra. A szimulált E.coli baktériumoknak is csak két fő tulajdonságát vizsgáljuk, az antibiotikum eltávolítására használt efflux pumpák számát és a pumpák hatékonyságát [9].

A populáció egy sejthalmaz, amiben minden sejtet egy egyenletrendszer vezérel. A sejtek szaporodnak vagy elpusztulnak a bennük tárolt antibiotikum és tápanyag koncentráció függvényében.

A sejtek különbözőségét a két paraméterük változása adja. A pumpák hatékonysága genetikus eredetű tulajdonság, mutáció hatására tud változni. A pumpák száma az antibiotikum jelenlététől függ, ugyanis az antibiotikum a génkifejezést módosítja a methylation nevű folyamat során.

A sejtek tulajdonságait egy Gauss-eloszlással generálja a modell ahol az eloszlás paraméterei az adott sejt paraméterei.

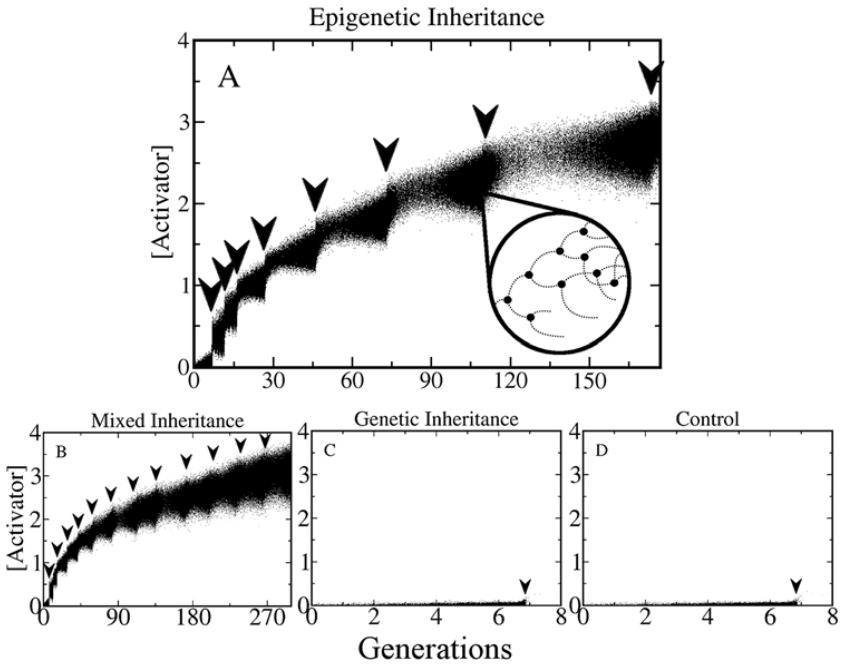
Motta négy kísérletet végzett:

- Kontrol szimuláció: Nincs öröklés, csak véletlenszerű mutáció.
- Genetikus öröklés: Szülő-gyermek kapcsolat a sejtek között csak az efflux pumpák hatékonyságát érinti (csak genetikus öröklés van).
- Epigenetikus öröklés: Szülő-gyermek kapcsolat csak a pumpák számát érinti
- Kevert öröklés: Mindkét tulajdonság örökölhető

Kísérlet:

Amikor a populáció eléri az 5000-es elemszámot, növeljük az antibiotikum koncentrációt.

Eredmények:



2. ábra. A populáció antibiotikum-ellenálló része túlél, a kevésbé ellenálló része elpusztul Forrás: [9]

2.5. Komplex modellek

Az előzőeknél komplexebb modelleket használtak Cytowski, Macklin és Waclaw cikkeiben. Ezek a modellek az ágensek vizsgált jellemzőin kívül a térbeli helyüket, méretüket, mozgásukat, fenotípusukat és az egyedek közötti interakciót is figyelembe vették.

2.5.1.Waclaw és munkatársai modellje

Waclaw a rákos sejtek terjedését és a genetikai felépítéüsket vizsgálta. [10] Kísérletek kimutatták, hogy a normális sejt populációk genetikai szempontból heterogének, míg a rákos tumort alkotó sejtek génállománya homogén és csak a tumor növekedésének kései fázisában jelennek meg mutációk. A modell megmutatja, hogy a rövidtávú szóródás és a sejtek gyors cseréje lehet a sejtek keveredésének az oka. Akár egy sejt hasznos mutációja elég ahhoz, hogy a sejt utódai orvosilag szignifikáns időn belül elfoglalják a tumor nagy részét. Ez a folyamat rövid idő alatt kemoterápia-ellenállóvá tumorok kialakulásához vezethet.

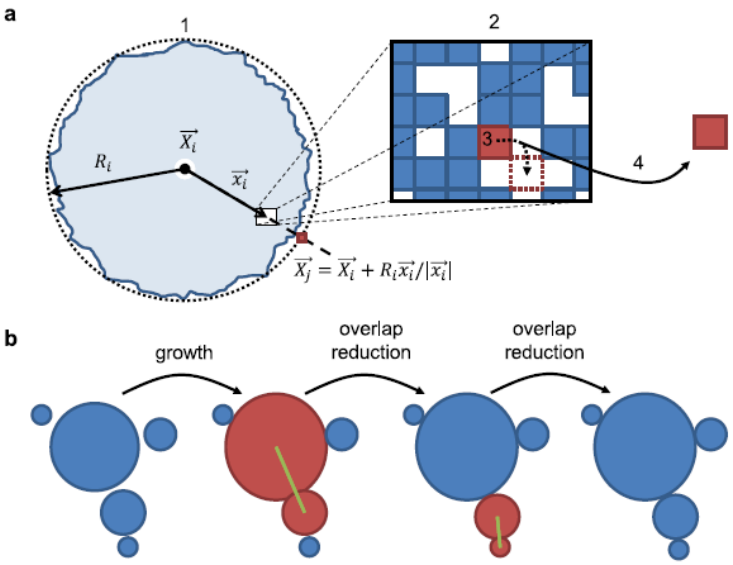
Térbeli modell

Egy tumor sejtjeit gömbök jelképezik, amik egy 3D-s négyzetrácson foglalnak helyet. Ez a Moore-szomszédság, egy gömbnek 26 szomszédja lehet. Minden ágensnek ismerjük a helyét és a bekövetkezett mutációkat, amik lehetnek semlegesek (passenger), gyorsíthatják a reprodukciót (driver), vagy növelhetik az ellenálló képességet (resistance carrying). Mindhárom mutáció csak egyszer alakulhat ki egy adott sejtben. Minden osztódásnál Poisson eloszlás szerint kaphat az utód sejt valamely mutációt.

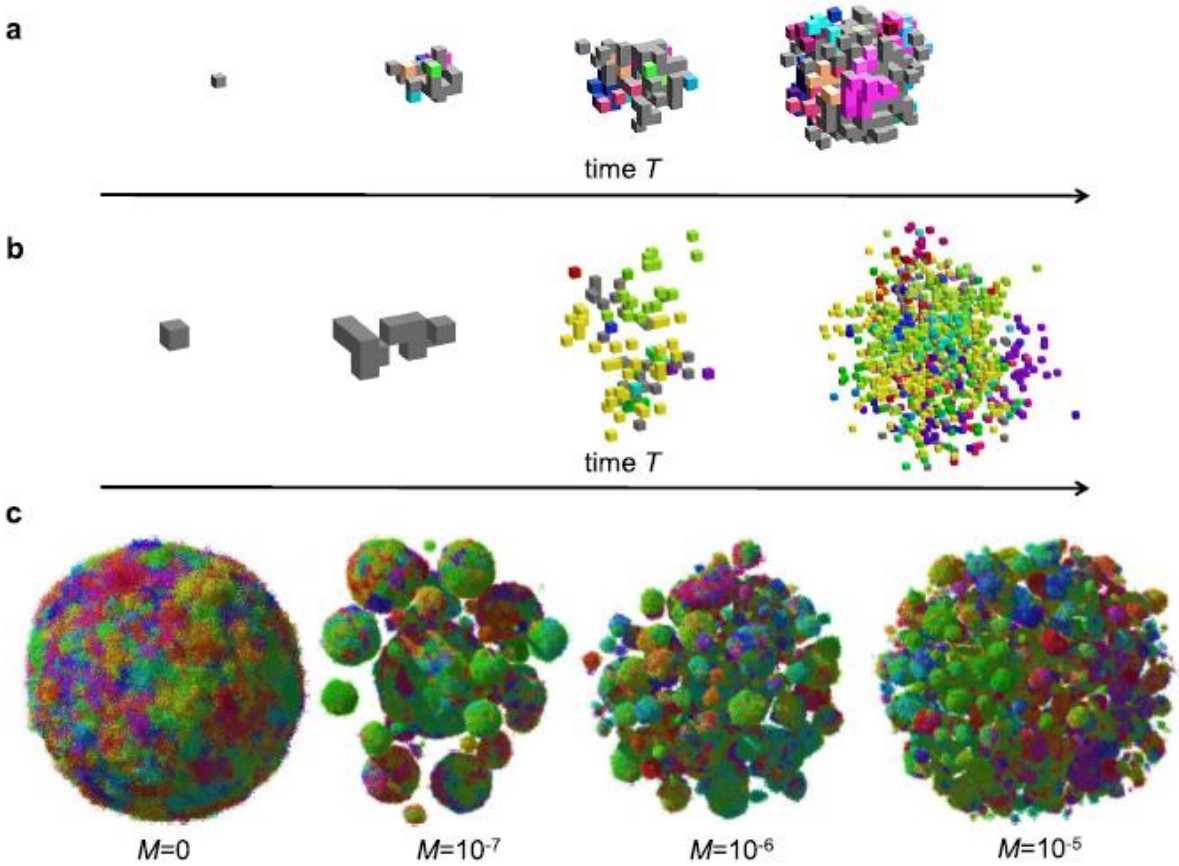
A szóródást úgy modellezték, hogy az utód sejt bizonyos valószínűséggel egy közeli üres helyre vándorol. A sejtek átfedését elkerülendő, ha ez megtörténik, a két átfedő sejtet a középpontjaik vonalában eltávolítja egymástól egy algoritmus.

A modell paraméterei:

- b: születési/osztódási ráta
- d: sejthalál ráta
- s: szelekciós előny (mennyit gyorsít a sejten a driver gén)
- y, yd, yr: Mutációs valószínűség (passenger, driver, resistance)
- M: szóródási valószínűség



3. ábra. A modell térbeli szerkezete és az átfedés megszüntető algoritmus működése [10]



4. ábra. a, b: Néhány snapshot a szimulációból d=0 és d=0,9 értéknél. c: 10^7 sejtű szimulációk d=0,9 értékkel és genetikai összetétel szerint színkódolva. [10]

2.5.2. Macklin és munkatársai modellje

Macklin modellje az angolul ductal carcinoma in situ (DCIS) jelenséget vizsgálja [8]. A DCIS egy daganat, amiből agresszív mellrák alakulhat ki. Olyan modellel szimulálták a DCIS -t, ahol a sejtmozgást mechanikai erők idézik elő. Potenciálfüggvényekkel szimulálták a különböző méretű és fenotípusú sejtek és az alap membrán közötti interakciókat. Minden sejt fenotípusát a génkészlettől és a mikrokörnyezettől függő sztochasztikus folyamatok határozzák meg.

A modell célja, hogy egy adott páciens adataira lehessen hangolni a modellt és ezzel egyedi előrejelzést adni a rák kialakulásáról.

Modell

Az egyedek egy rácsszerkezet nélküli térben foglalnak helyet, és szabadon mozoghatnak.

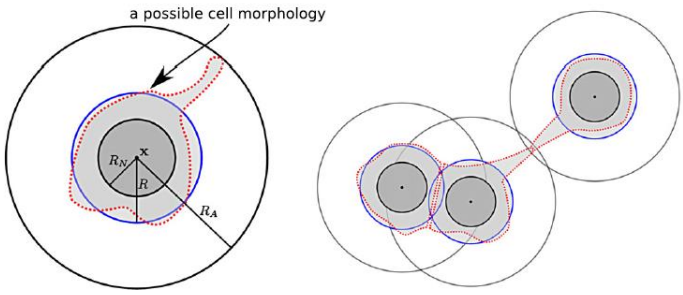
Minden egyed rendelkezik x pozícióval, v sebességgel, V teljes térfogattal, Vc cisztoplazma térfogata és Vn sejtmag térfogattal. A sejtek alakjával nem számol a modell (körnek tekinti őket), de a sejtek sugarát kiszámolhatjuk a térfogatukból a következő képletekkel;

V=4/3*Pi*R^3, ahol R a teljes sejt sugara és

V=4/3*Pi*Rn^3, ahol Rn a sejtmag sugara

A sejtnek van egy Ra interakciós távolsága, amin belül vonzani tud más sejteket.

A sejtek átfedhetik egymást, ezzel szimulálva az összenyomhatóságukat, de a minél jobban összenyomódik egy sejt, annál jobban ellenáll a nyomóerőnek.



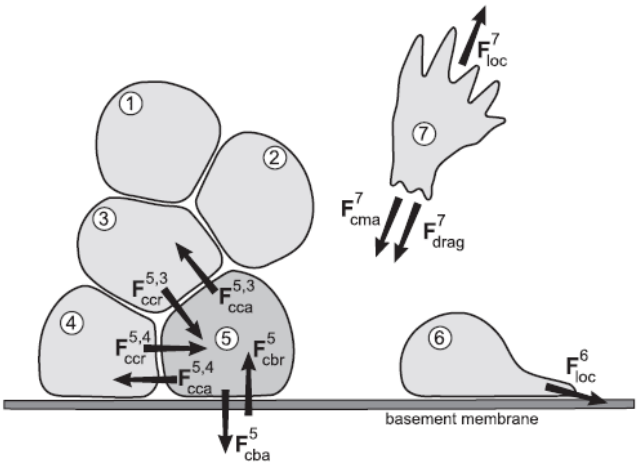
5. ábra Egy sejt fontos méretadatai és sejtek lehetséges formái interakció közben [8]

A sejtekre sokféle erő hathat:

- Sejtek közötti vonzóerő: $\mathbf{F_{cca}}$
- Sejtek és ECM közötti erő: $\mathbf{F_{cma}}$
- Sejtek és basement membrane közötti erő: $\mathbf{F_{cba}}$

- Sejtek közötti taszító erő: $\mathbf{F_{ccr}}$
- Sejt és BM közötti taszító erő: $\mathbf{F_{cbr}}$
- Mozgatóerő: $\mathbf{F_{loc}}$
- Közegellenállás: $\mathbf{F_{drag}}$

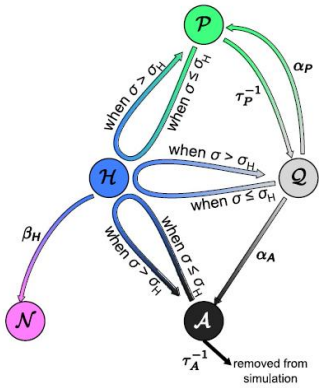
$$m_i \dot{\mathbf{v}}_i = \overbrace{\sum_{\substack{j=1 \\ j \neq i}}^{N(t)} (\mathbf{F}^{ij}_{\mathrm{cca}} + \mathbf{F}^{ij}_{\mathrm{ccr}})}^{\text{cell-cell interactions}} + \overbrace{\mathbf{F}^{i}_{\mathrm{cba}} + \mathbf{F}^{i}_{\mathrm{cbr}}}^{\text{cell-BM interactions}} + \overbrace{\mathbf{F}^{i}_{\mathrm{cma}} + \mathbf{F}^{i}_{\mathrm{drag}}}^{\text{cell-medium interactions}} + \mathbf{F}^{i}_{\mathrm{loc}}.$$



6. ábra Egy sejtre ható erők és vizualizációjuk [8]

A sejtek különböző állapotokat tudnak felvenni:

- Nyugalmi állapot (Q)
- Osztódás (P)
- Hypoxia (H)
- Necrosis (N)
- Programozott sejthalál (A)



7. ábra. Az állapotok között megadott feltételek megléte mellett tud váltani a sejt [8]

2.5.3. Cytowski és munkatársai modellje

Cytowski modellje párhuzamos végrehajtásra és szuperszámítógépekre optimalizált. A cél, hogy szövet nagyságrendű sejt populációt szimuláljon. [2]

A sejteknek van koordinátájuk, sebességük és sugaruk. Tudják egymást „lökdösni” és tudnak osztódni. Egy egyszerű szabályrendszer alapján váltanak fenotípust.

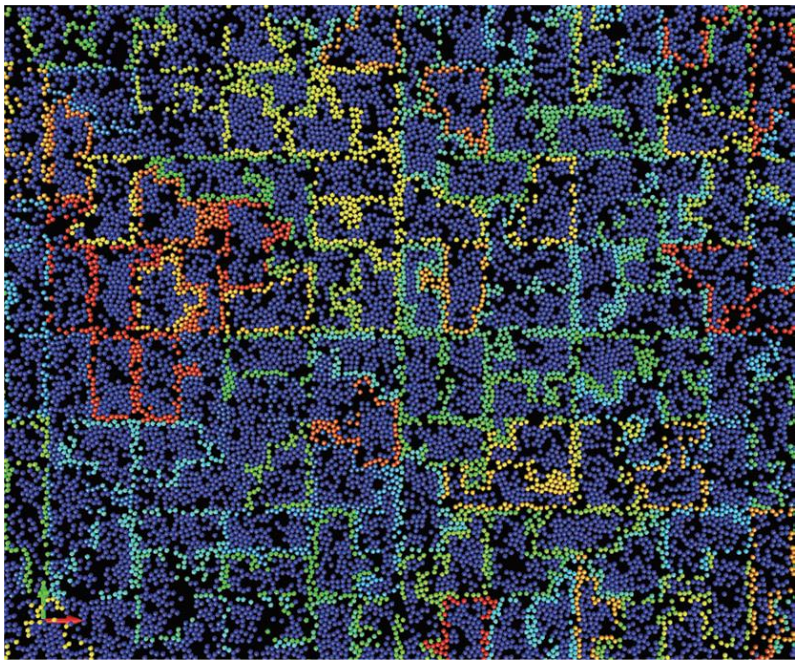
A kulcsszó itt a párhuzamosítás. A populációt egy Peano-Hilbert space filling curves algoritmus dekomponálja, majd párhuzamos folyamatoknak szétosztjuk a komponenseket.

A sejtek minden folyamatban egy iteratív algoritmus fa szerkezetbe szervezi, így gyorsabb megtalálni egy sejt szomszédait.

Párhuzamosan elindul egy adatcsere a régiók között, míg a helyi régió sejtjei megkeresik a szomszédjaikat és kiszámolják a potenciál és sűrűség függvényeket, ami alapján később mozogni fognak.

Miután befejeződött az adatcsere, a helyi régió sejtjei a más folyamatokból érkezett adatokat felhasználva újabb szomszédokat keresnek és rájuk is kiszámolják a mozgáshoz szükséges függvényeket.

Végül a helyi régió minden sejtje megvizsgálja a fenotípus váltási feltételeket és a korábban kiszámolt függvények alapján mozgatóerőt számolnak és új pozícióba mozognak.



8. ábra. A kékkel jelölt területek a dekomponált régiók, míg a színes részek a régiók határterületei, ahol adatcsere történik a szomszédos régiókkal. Forrás: [2]

2.6. Hibrid modellek

A hibrid, vagy többrétegű, modellek a fizikai és genetikus szimuláción kívül tartalmaznak egy makrokörnyezetet jelképező plusz réteget. Ez lehet egy kontinuum vagy diszkrét tér. Lehet ezen kívül egy vagy több globális változó, ami a környezet, mint egész állapotát vagy tulajdonságát írja le.

2.6.1. Walker és munkatársai modellje

Walker modellje a bőrsejtek kialakulását szimulálja. A sejteknek van fenotípusa, x,y és z koordinátája, mérete. A fenotípus váltást egyszerű szabályrendszer irányítja. [11]

Bár a sejteknek három koordinátája van, a modellezett tér egy kétdimenziós mátrix. A sejtek nőni, osztódni, a szubsztráthoz vagy más sejtekhez ragadni, terjedni, vándorolni és elpusztulni tudnak.

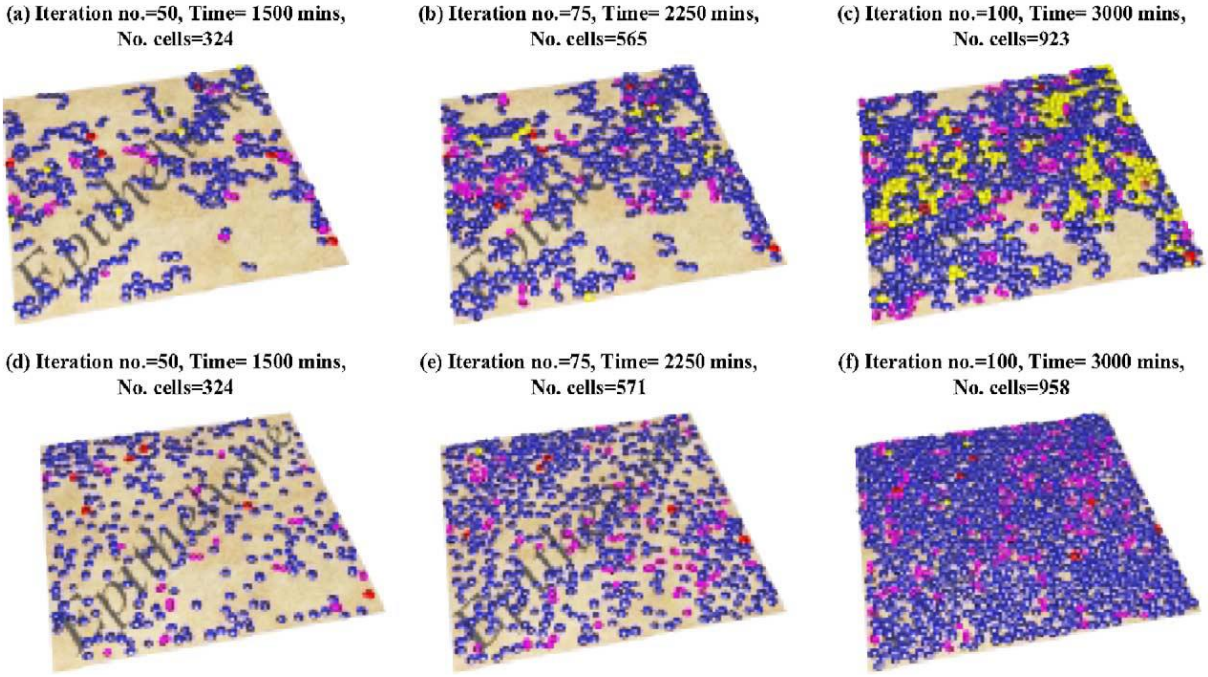
Két fontos környezeti változó van. Az egyik egy kommunikációs mátrix, ami NxN méretű, ahol N a sejtek száma. Ez egy szomszédsági mátrix, ahol a sejtek regisztrálni tudják, hogy melyik más sejtekhez „ragadtak oda”. A másik környezeti tényező a sejtek környezetében lévő kalciumion-koncentráció. Ez az érték a sejtek összekapcsolódásának valószínűségét határozza meg.

Ha két sejt összekapcsolódott, egymás felé mozognak, amíg a köztük lévő távolság 0 nem lesz, ezután megállnak. Ha két sejt összekapcsolódott, attól még nem kizárt, hogy más sejtekhez is kapcsolódjanak.

Fontos mechanika a sejtek vándorlása. Minden sejt, ami már rákapcsolódott a szubsztrátra, de más sejtekkel még nem létesített kapcsolatot szabadon mozoghat a modellben. Minden mozgó sejt egyenesen halad, de véletlenszerűen irányt válthat. Ha akadály álla sejt előtt, 30°-al változtat az irányán, majd újra megpróbál mozogni. Ha ez nem sikerül, a sejt mozdulatlan marad.

A modell célja replikálni a kalcium koncentráció függvényében történő változásokat a populációban.

Eredmények:



9. ábra (a-c): Normál kalcium koncentráció, összekapcsolódó sejtekből felépült „kolóniák”

(d-f): Alacsony kalcium szint mellett a sejtek még nagy sűrűség mellet sem kapcsolódnak. Forrás: [11]

2.6.2. Jeon és munkatársai modellje

Jeon egy négyzetrács nélküli hibrid diszkrét kontinuum modellt alkotott, hogy szimulálja a rákos sejtek, az ECM és a basement membrane közötti interakciókat, különös figyelemmel a tumor terjedési formájára. [4]

A modell kontinuum része

ECM: sejtek közötti tér és a határoló membránok képezik. A határoló membrán akadályt képez a sejteknek, míg az ECM stabil alapzatot és a sejtek közötti kommunikáció médiumát jelenti. Az ECM egy kontinuum tér, a koncentrációját a következő egyenlet írja le:

$$\frac{\partial e}{\partial t} = -\alpha m e,$$

Ahol alfa az ECM roncsolási rátája az MDE által

MDE: mátrixroncsoló enzim. Ezt a rákos sejtek választják ki, hogy át tudják törni a határoló membránt és a környező ECM-be jussanak. Mivel ez mind az ECM-et és a BM-et is roncsolja, ezért az MDE mind segíti, mind gátolja a rák terjedését. Az MDE koncentrációt a következő egyenlet írja le:

$$\frac{\partial m}{\partial t} = D_{\mathrm{m}} \nabla^2 m + \beta c - \gamma m,$$

Tápanyag (Oxigén): A sejtek túléléséhez, növekedéséhez és terjedéséhez elengedhetetlen az oxigén jelenléte. A modell feltételezi, hogy az oxigénsűrűség az ECM sűrűségétől függ. Az is feltételezett, hogy az oxigén diffundál az ECM-be, illetve természetes módon lebomlik. Az oxigén diffúziót leíró függvény a következő:

$$\frac{\partial n}{\partial t} = D_{\mathrm{n}} \nabla^2 n + \delta e - \varphi c - \lambda n,$$

A modell diszkrét része

A sejtek vándorlását sokszor egy persistent random walk motion-nel lehet leírni, ami a Langevin egyenletből származik.

$$m \frac{d\vec{v}_{\mathrm{i}}(t)}{dt} = -\xi \vec{v}_{\mathrm{i}}(t) + \vec{f}_{\mathrm{i}}^{\mathrm{R}}(t) + \vec{f}_{\mathrm{i}}^{\mathrm{D}}(t),$$

Sejtműködések:

Egy sejt két utódsejtté osztódik, ha a szülő sejt elérte az osztódási életkort és van hely két utódsejtnek. Ha nincs hely az osztódásra, a sejt nyugalmi állapotba kerül és fele olyan gyorsan fogyasztja környezetéből a tápanyagot.

Sejthalál, mutáció:

A sejtek elpusztulnak, ha az oxigénszint bizonyos érték alá csökken továbbra is elfoglalva a helyüket. A sejt genomját a következő értékek adják: a sejt osztódási életkora, oxigénfogyasztás, MDT termelési sebesség, kapcsolódási helyek száma. Az öröklés és véletlenszerű mutáció szimulált.

Kapcsolódási helyek (sticky sites):

Minden sejt felszínén meghatározott mennyiségű, egyenletesen elosztott, „ragadós" felület található, ami vonzza más sejtek ragadós részeit, ezzel szimulálva a sejtek közötti interakciót.

2.7. Az irodalomkutatás eredményinek összefoglalása

Az irodalomkutatás során megvizsgáltam a forrásokban szereplő modelleket, figyelmet fordítva a modellezés céljára, a modellben szereplő ágensek felépítésére, interakcióikra más ágensekkel, és a környezet szimulációjára.

A modellezés célja mindegyik forrás esetén egyedi. A kutatók minden esetben egy konkrét jelenség modellezésére keresték a megoldást, nem használtak általános célú, harmadik féltől származó szimulátort és maguk írták meg a modell szoftveres implementációját. Ennek megfelelően az ágensek és a hozzájuk tartozó szabályrendszerek implementációjában figyelhető meg a legnagyobb változatosság. Általában az ágensek viselkedése valamilyen egyszerű szabályhoz kötött, az adott ágens állapotának megfelelően. Az ágensek állapotát egyszerű számszerű adatként vagy valamilyen bonyolultabb szerkezetként lehet ábrázolni.

A környezet szimulációjának két fő iránya van. Ezek a folytonos teret használó modellek és a diszkrét teret használó modellek. A diszkrét teres modellek előnye a jó párhuzamosíthatóság és a nagy elemszámú kísérletek gyors futtatása. A folytonos teres modell viszont lehetőséget ad bonyolultabb mechanikai jellegű szimulációra, például a rugalmas ütközés vagy a sejtek változó alakjának szimulációjára.

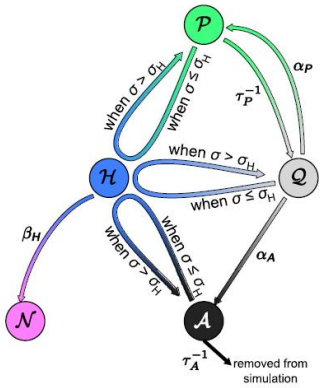
Az általam készített szoftver térbeli szimulációs részében a gyorsaságra törekedve rácsszerkezetet érdemes használni. Az ágensek viselkedésének leírásához pedig előre definiált, de a felhasználó által paraméterezhető viselkedésformákat, például mozgás, osztódás, anyagfelvétel, használhatnék.

3. ÁLLAPOTÁTMENETEK

A fenti cikkek példáiból is látszik, hogy a sejteket szimuláló ágens alapú szimulációk egyik sarokpontja a sejtek állapota, illetve az állapotok közötti átmenetek dinamikája. Az állapotok diszkrétek és egyértelműen meghatározzák, hogy az ágens éppen hol jár az életciklusában. Általában két állapot közötti átmenetnek valamilyen valószínűsége vagy sebessége van, illetve előfordulhat, hogy egy adott állapotból való kilépés vagy adott állapotba való belépés előfeltételhez kötött.

3.1. Ábrázolási lehetőségek

Ahhoz, hogy állapotokat és átmeneteket be lehessen építeni egy szoftverbe, valamilyen adatszerkezet részeként kell őket eltárolni. Fontos szempont az adatméret, a változtathatóság és a könnyű mentés, illetve betöltés lehetősége. Mindkét módszer, amit be fogok mutatni, gyakorlatilag egy gráf adatszerkezetet valósít meg, hiszen ez a leg kézenfekvőbb tárolási módja az állapotátmeneteknek.



3.1.1. Állapotátmenet-mátrix

Ez az ábrázolás egy szomszédsági mátrixot használ. A táblázat fejlécei tartalmazzák az állapotokat, a táblázat többi része pedig az átmenetek valószínűségét. Ez egy diagonál mátrix, azaz szimmetrikus az átlóra. Az átló azonban nem feltétlenül áll zérusokból, ugyanis itt az átló az adott állapotban maradás valószínűségét tartalmazza.

Egyszerűen kivitelezhető megoldás, viszont, ha nagy az állapotok száma, de nem sűrű a kialakított mátrix, akkor helyet pocsékolunk a megoldással. Ezen felül minden új állapot egy sorral és egy oszloppal bővíti a táblázatot, tehát a lefoglalt terület négyzetesen nő az állapotok függvényében.

Könnyű elmenteni és betölteni a mátrixot egyszerűbb, pl. CSV, formátum felhasználásával. Ha viszont a felhasználó a mentett állományt szeretné szerkeszteni, akkor szüksége lesz valamilyen táblázatkezelőre, ami támogatja a CSV formátumot, vagy kénytelen lesz maga bogarászni egy szövegfájlban.

3.1.2. Objektumorientált megközelítés

Ez a megközelítés három osztály segítségével valósítja meg az állapotátmenet gráfot. A gráf osztály foglalja magába az állapotokat. Az állapot osztálynak egy, vagy több, átmenet adattagja van. Előfordulhat az is, hogy egy állapot példánynak nincs átmenet adattagja; ekkor az állapot egy végállapot. Az átmenet pedig tartalmaz egy mutatót a cél állapotra.

Ennek a megoldásnak a szerkezete közelebb van a reprezentálandó gráfhoz. A mérete az adattagok számával lineárisan nő. Új állapot és átmenet felvétele is könnyű, csak hozzá kell adni egy új elemet a megfelelő listához.

A mentés és betöltés funkciók ebben a megoldásban sokkal bonyolultabbak. Két lehetőség van erre:

Kiíratni bináris fájlba a példányok memóriaképét, de ebben az esetben a felhasználó nem tudja szerkeszteni.

Adott formátumú leíró fájlba, pl. JSON formátumba, kiírni a szerkezetet. Így a felhasználó is létre tud hozni saját gráfokat és betölteni azokat a programba. Ehhez viszont szükség van szerializáló és deszerializáló könyvtárakra, amik növelik a kódbázist.

3.2. Matematikai háttér

A dolgozatom célja egy populáció szinten viszonylag pontosan működő szimuláció létrehozása. Ez matematikai jellegű problémákat vet fel. A probléma pedig az, hogy az egyszerűsítés következtében elvesznek az ágensek állapotátmeneteinek az okai. Látjuk, hogy a sejtek egy rész ilyen állapotban van, más része meg olyanban, de nem tudjuk miért kerültek pont abba az állapotba.

Egy valós sejt működése rengeteg inputtól függ; oxigén és tápanyag szintek, hormonok, szomszédos sejtek kémiai jelei és még sok más. Egy valós sejt esetében felállíthatók ha, akkor feltételek a sejt működésével kapcsolatban. Egy általam szimulált sejt esetén nem.

Kézenfekvő megoldás a problémára, hogy véletlennek vesszük az állapotok közötti átmenet okait és a populáció egészének a viselkedésére koncentrálunk.

Egy Markov-láncként leképezett matematikai modell jó megoldást ad a problémára.

3.2.1. Markov-láncok

Egy Markov-láncot szemléletesen egy irányított, súlyozott gráfként lehetne ábrázolni, ahol a csúcsok az állapotok, az élek az adott irányba lehetséges átmenetet jelzik, az élsúlyok pedig az átmenet valószínűségét jelentik. [7]

Fontos megemlíteni a Markov-tulajdonságot, ami röviden azt jelenti, hogy a Markov-láncnak nincs emlékezete. Az adott állapotból való kilépés, vagy ottmaradás, csakis az adott állapotból kiinduló átmenetektől függ, a valószínűséget nem befolyásolják a múltbeli lépések. A tulajdonság egyik érdekes következménye, hogy egy adott állapotból egy több állapottal ”arrébb” lévő állapotba való lépés valószínűsége is kiszámítható.

Még egy fontos tulajdonsága van a Markov-láncoknak. Egy Markov-láncban nem kötelező ”kezdőpontot” megadni, meg lehet adni egy kezdeti eloszlást is, ami alapján kisorsolásra kerül a kezdőpont. Ez azért fontos, mert így egy nagyobb méretű seed populációt is létre lehet hozni a szimulációban, a sejtek állapoteloszlása pedig megfelel annak, mint ha egy ágensből növesztettük volna a populációt.

Azért a Markov-láncos megoldást választom a feladat megoldásához, mert viszonylag kevés paramétert kell beállítani – gyakorlatilag csak az állapotátmenetek valószínűségét – és mert populáció szinten pontos megoldást ad.

Hátránya a módszernek, hogy a valószínűségeket kézzel kell beállítani és lefuttatni velük a szimulációt, hogy megbizonyosodjunk a helyes működésről. Vagy egy másik, már kipróbált modell értékeit kell beírni.

4. FELADAT SPECIFIKÁCIÓ, A KIALAKÍTANDÓ MODELL

A dolgozat során egy olyan grafikus alkalmazást hozok létre, ami alkalmas általános célú sejtszimulációra és az ebből a szimulációból generált adatok megjelenítésére.

A szoftver célja az, hogy in vitro biológiau kísérletek előtt a kutatóknak ne kelljen értékes időt elpazarolniuk azzal, hogy felépítenek egy szimulációs modellt egy kísérlet megvalósíthatóságának vizsgálatára. Előzetes tervezési céllal akár igénybe vehetnének egy olyan szoftvert, amiben igény szerint hozhatnak létre sejteket saját működéssel, készíthetnek környezetet az egyedi sejteknek és adott paraméterek mellett futtathatnának szimulációkat. Mindezeknek a paramétereknek a beállítása pár percet venne igénybe a napok vagy hetek helyett, amennyi időbe egy teljesen új szimulációs modell elkészítése kerülne.

A szimulátor négy fő komponense a következők:

- Felhasználói Felület
- Szimuláció
- Vizualizáció
- Adatgenerátor

A Felhasználói Felületen tudja a felhasználó felvinni a rendszerbe a sejtek lehetséges állapotait és az állapotok közötti átmeneteket. Lehetőség van több állapotátmenet-gráfot is létrehozni, ezáltal többféle sejtpopuláció is létezhet egy környezetben.

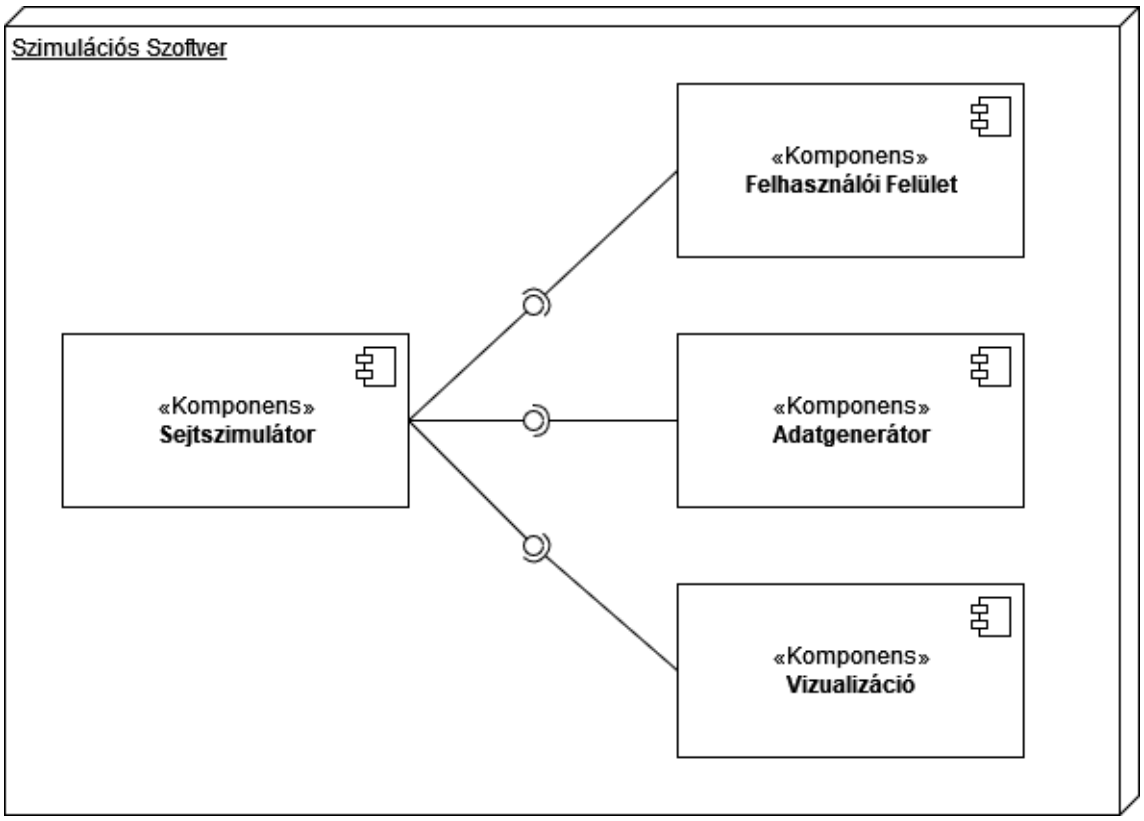
A felhasználó beállíthatja a ”pálya” méretét, a szimuláció időosztását és a szimuláció hosszát. A környezetnek megadható a szélessége és magassága. A környezet szélére automatikusan fal települ, ami megakadályozza a sejtek szimulált térről való leesését. Igény szerint fal további falak létrehozásával meghatározott alakúra alakítható a ”játéktér”.

A Szimulációs komponens tartalmazza a program magját. Fontos szempont a szimuláció működésével kapcsolatban a sebesség, az erre alkalmas számításokat párhuzamosítva végzi a szimulációs mag.

Az átmenetgráfokon kívül beállítható az időosztás, mind a sejtek, mind a diffúziós modell tekintetében, a szimulációnak helyet adó négyzetrács szélessége, magassága és a négyzetrács osztása, tehát a rácsméret.

Kétrétegű a szimuláció, egy anyagtranszport modellt is tartalmaz, ami a sejtek környezetében található különféle vegyi anyagok koncentrációját és forrásait tartja számon.

Az Adatgenerátor komponens a szimuláció során beállítható időközönként összegzést ad a szimuláció állapotáról. pl az ágensek száma és eloszlása, vagy az adott szimulációs ciklus számítási ideje. Ezeket az adatokat elmenti egy fájlba.



10. ábra. A szoftvercsomag komponensei

A Vizualizációs komponens az adatokból képsort készít. Az ágenseket kis négyzetekként jelöli a program, a színük pedig az aktuális állapotuktól függ.

11. ábra. A szoftver munkamenetének rövid összefoglalása

4.1. A komponensek kialakítása

Szó volt eddig a komponensek szerepéről, most részletesebben kitérek a komponensek kialakítására.

4.1.1. Állapot reprezentáció

Nagyon fontos része a modellnek az, hogy hogyan reprezentálom az állapotokat.

Az állapot azonosítója a neve. Ezt a felhasználó adhatja meg, a megjelenítésen és azonosításon kívül nincs szerepe. Minden állapot kap egy számot is, ez a vizualizáló komponens miatt szükséges, e szerint a szám szerint rendel majd színt az ágensekhez a megjelenítési folyamat során.

Minden állapot kap négy boolean típusú tulajdonságot is, ezek fogják befolyásolni az ágens viselkedését, amíg az adott állapotban van.

A Motile flag lehetővé teszi a sejt számára a mozgást.

A Proliferative flag engedélyezi az osztódást.

A Dead flag olyan állapotot jelez, amilyen állapotú sejt eltávolítható a szimulációból (például a programozott sejthalálon áteső sejt ilyen flaggel ellátott állapotba kerül).

A Default flag az adott állapotátmenet-gráf alapértelmezett állapotát jelöli. Minden gráfban, ahol létezik osztódó flaggel ellátott állapot, lennie kell egy alapértelmezett állapotnak. Az újonnan létrehozott sejtek ebben az állapotban kezdenek.

Az állapot tulajdonságai közé tartozik egy szótár, ami az adott állapot anyagfelvételét reprezentálja. Minden állapot különféle vegyi anyagokat vehet fel különféle rátával.

Az állapot ezen felül tartalmaz egy listát a belőle kiinduló állapotátmenetekkel feltöltve.

4.1.2. Állapotátmenet reprezentáció

Az állapotátmenetek is rendelkeznek névvel, bár ez itt is csak azonosításra szolgál.

Az állapotátmenet ezen felül tartalmaz két Állapot típusú változót, a Forrás állapotot és a cél állapotot.

Matematikai szempontból a legfontosabb tulajdonságai az átmenetnek a Valószínűség és Esély tulajdonságok. A Valószínűség meghatározza az átmenet valószínűségét a Forrásból a Célba. Az Esély változó a tényleges sorsolásnál, azaz a következő állapot meghatározásánál jön a képbe.

A Valószínűséget egy Hill függvény és az adott helyen lévő vegyi anyag koncentráció alapján számíthatjuk ki. A felhasználó megadja a vegyi anyagot és a Hill függvény két paraméterét: a Hill együtthatót és az EC50 paramétert.

Az adott állapotból kifelé mutató átmenetek Valószínűségeinek a normalizálásával kapjuk meg az Esély értékeket.

Egy példa: A-ból B-be 40% valószínűség van az átmenetre, A-ból C-be 25%. Az átmenetszámítás során egy random generátor 0 és egy közötti számot generál. Ebben a példában, ha 0,4 alatti számot generálok, a B állapotba lépek, ha 0,65 és 0,4 közötti számot, akkor C állapotba lépek, ha 0,65 fölötti szám jön ki, A állapotban maradok.

Ebben a példában A->B átmenet valószínűsége 0,4 az esélye 0,4; A->C átmenet valószínűsége 0,25 esélye 0,65.

4.1.3. Diffúziós modell

A modellemben a vegyi anyagok terjedését a hőegyenlet egy numerikus módszerével, az FTCS sémával oldottam meg.

Az FTCS séma a szomszédos cellák energiaszintjéből (itt vegyi anyag koncentrációjából) és a saját energiaszintjéből képez egy differenciát és ennek a differenciának a segítségével számítja ki az adott cellába ki és beáramló energiát.

$$u_{i,j}^{k+1} = \gamma(u_{i+1,j}^{k} + u_{i-1,j}^{k} + u_{i,j+1}^{k} + u_{i,j-1}^{k} - 4u_{i,j}^{k}) + u_{i,j}^{k}$$

Az egyenlet bal oldala az adott cellába a következő időpillanatban jelen lévő koncentráció.

Az egyenlet jobb oldalán a zárójelben a szomszédos cellák koncentrációja látható egy normalizáló taggal (-4u…). A zárójel jobb oldalán található tag a jelenlegi időpillanatban a jelenlegi cella koncentráció.

A zárójel előtti együtthatót a következő módon számoljuk.

$$\gamma = \alpha \frac{\Delta t}{\Delta x^2}$$

Ahol alfa a hővezetési vagy diffúziós együttható, delta T az időosztás, delta X pedig a négyzetrács mérete.

Ezzel a módszerrel egy probléma van, az instabilitás. Ez a módszer ugyanis csak a következő feltétel mellett stabil:

$$\Delta t \leq \frac{\Delta x^2}{4\alpha}$$

Ez azt jelenti, hogy csak kis időosztás mellett lesz stabil a közelítés. Egy részmegoldás lehet ha a hővezetési együtthatót csökkentjük, ettől viszont ”viszkózusabb” lenne a szimulálni kívánt vegyi anyag.

Egy másik megoldás lehet az, hogy ha külön kezeljük a diffúziós modellt és a sejtek működését. Így a diffúziós modellt megfelelő sűrűséggel lehet frissíteni anélkül, hogy a sejtek számításigényes műveleteit is el kelljen végezni.

Azért

választottam a második módszert, mert viszonylag egyszerű, és nem kell kompromisszumokba bocsátkozni a két rendszer különbsége miatt.

4.1.4. A sejtek mozgása és osztódása

A szimulált sejtek két legfőbb mechanikája az állapotváltás után a mozgás és az osztódás.

A sejtek a helyüket egy négyzetrácson foglalják el és csak ezen mozoghatnak a szomszéd cellákba való átlépéssel. A mozgás valószínűsége függ a sejt sebességétől és a négyzetrács osztásától. A mozgás iránya véletlenszerű, de egy sejt nem fog olyan cellába mozogni, amiben már található másik sejt.

Az osztódás hasonlít a mozgáshoz, azonban itt elmozdulás helyett a szülő sejt egy új sejtet helyez a szomszédos szabad cellába.

Ahhoz, hogy osztódás menjen végbe, két feltételnek kell teljesülnie. A sejtnek osztódásra alkalmas állapotban kell lennie és az élettartama során el kellett töltenie egy megadott időhosszt osztódásra kész állapotban, ezzel szimulálva a sejt életciklusát, amiben végig kell járnia bizonyos fázisokat az osztódás megkezdése előtt.

Mind a mozgás, mind az osztódás esetében a sejtek négycellás szomszédságot vizsgálnak meg.

4.1.5. Párhuzamosítás

Előzetes tapasztalataim alapján elég a versenyhelyzet nélkül párhuzamosítható részeket egy .Net keretrendszer által biztosított megoldással. A feltételnek megfelelő műveletek a vegyi anyag diffúzió számítás és az ágensek anyagfelvételének szimulációja.

4.1.6. Kimeneti fájlformátumok

Az adatgenerátor által előállított adatokat egyszerű CSV formátumban menti el a program, hogy később Excellel vagy hasonló táblázatkezelőkkel és adatelemző szoftverekkel egyszerűen meg lehessen nyitni.

5. IMPLEMENTÁCIÓ

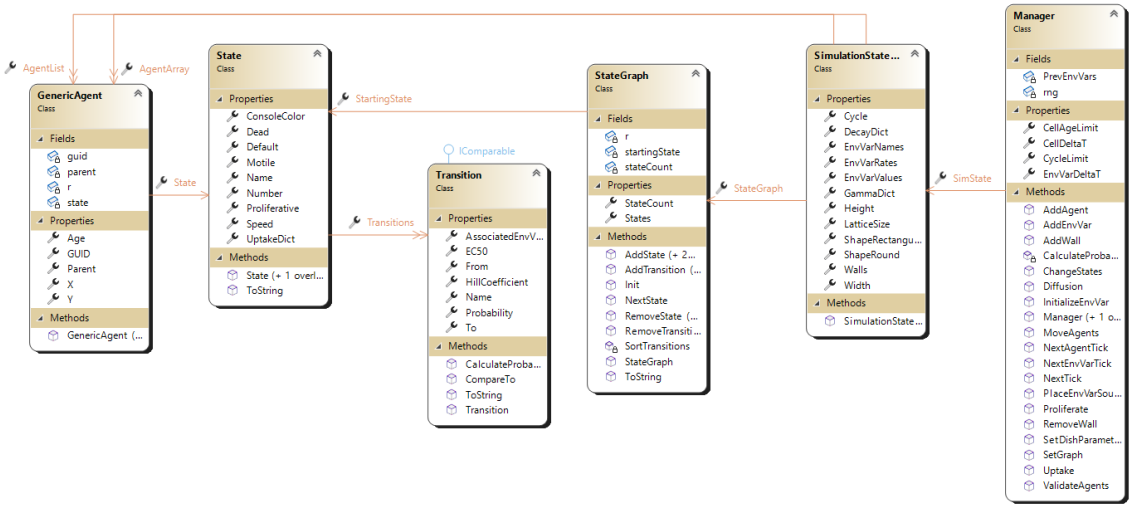
Ebben a fejezetben a szoftverem fő komponenseinek a szerkezetét és működését fogom kifejteni. A felhasznált keretrendszereken és patterneken felül a párhuzamosítási és adatszerkezeti lehetőségekről is írni fogok.

5.1. Szimulációs mag

A szimulációs komponens a szoftver alapvető egysége. Ez a komponens tartalmazza az osztályokat, amivel leírható egy teljes állapotátmenet-gráf és ez a komponens végzi a szimulációs számításokat, valamint belépési pontot biztosít a többi komponenssel való interakció számára.

5.1.1 Osztályok

A szimulációs mag osztályai a következő osztálydiagram szerint épülnek fel:



12. ábra. A szimulációs mag osztálydiagramja

Ahhoz, hogy egy állapotátmenet-gráfot felépítsünk a következő osztályokra van szükség:

- StateGraph: Ez az osztály reprezentálja az állapotátmenet-gráfot. Tartalmaz egy gráfcsúcsokat reprezentáló, State osztályokat tartalmazó gyűjteményt. A gráf feladata egy adott State input alapján meghatározni a következő állapotot.
- State: Ez az osztály reprezentálja a gráf egy csúcsát és egyben leírja az ebben az állapotban lévő ágensek viselkedését. Ezt négy logikai változó és egy sebesség

változó segítségével valósítom meg. A State része még egy szótár változó is, ami az adott állapotban lévő ágens anyagfelvételének az arányait tárolja.

- Transition: Ha a State osztály a gráf csúcsa, úgy a Transition osztály a gráf éle. Az osztályban tárolt Hill és EC50 együtthatók és az átmenethez köthető vegyi anyag koncentráció alapán kiszámolható az átmenet valószínűsége, amit tekinthetünk a gráfél súlyának.
- GenericAgent: Az ágens maga, ami a szimulált térben helyet foglal, mozog, a leírt szabályrendszer szerint viselkedik. Koordinátái, életkora, egyedi azonosítója és egy State változója van. Az ágens rendelkezik egy szülőre mutató változóval is, ha az ágens osztódással keletkezett. Ez felhasználható egy adott sejt családfájának a feltérképezésére.

A fent felsorolt osztályok fő feladata az adattárolás és leíróként viselkedés. Ezért kevés és főleg adminisztratív jellegű függvény része ezeknek az osztályoknak, például konstruktorok és a StatGraph esetében az állapotokat rögzítő és eltávolító függvények. A szimuláció főbb feladatainak a működtetéséért a Manager osztály a felelős.

5.1.2. SimulationStateObject

A SimulationStateObject osztály feladata a szimuláció egy állapotának a leírása. A Manager osztály a szimuláció folyamán ezen az objektumon végzi el a változtatásokat. Ez az objektum tárolja az összes ágenst, a vegyi anyagokat leíró változókat, a szimulációs terület tulajdonságait és az állapotátmenet-gráfot, ami alapján az ágensek viselkedése meghatározható.

A szimulációs terület leírói a következők:

- Cycle: A szimulációs terület életkora, leírja, hogy milyen régen fut a szimuláció. A mértékegysége másodperc.
- Height és Width: a szimulációs terület magassága és szélessége.
- LatticeSize: a szimulált négyzetrács osztásköze, a mozgásnál kap szerepet.
- ShapeRound és ShapeRectangular: a szimuláció előkészítéséhez segédváltozók, ezek határozzák meg, hogy milyen alakban határolják majd körbe falak a szimulációs területet.
- Walls: egy logikai változókkal feltöltött mátrix, meghatározza, hogy hol találhatóak falak a szimulált területen. A falak mind az ágensek, mind a vegyi anyagok számára akadályként viselkednek.

Ezeket a változókat a szimuláció elején állítja be a felhasználó vagy az automatika és a működés folyamán az értékeik nem változnak. A többi számítás használja fel az értékeiket.

A vegyi anyagok működéséhez a következő változókra van szükség:

- EnvVarNames: Egy lista, ami tartalmazza a vegyi anyagok neveit
- EnvVarRates: Egy háromdimenziós mátrix; az első két dimenzió a térbeli kiterjedés szélessége és magassága. A harmadik dimenzió a vegyi anyagok számától függ. Minden vegyi anyag külön rétegén létezik a mátrixnak. Ez a mátrix a vegyi anyagok termelődésének a rátáit tartalmazza. A pozitív értékek az adott pozícióban növelik a koncentrációt, a negatív értékek csökkentik.
- EnvVarValues: Ez is egy háromdimenziós mátrix, itt az egyes vegyi anyagok koncentrációi találhatók.
- GammaDict: Egy szótár változó, ami minden vegyi anyaghoz hozzárendel egy Gamma értéket. A gamma a diffúzióban játszik szerepet, a hővezetési együtthatóból számolható ki és az adott vegyi anyag terjedési sebességének egy együtthatója.
- DecayDict: Ebben a szótárban a vegyi anyagok bomlási rátája található. Minden anyaghoz hozzárendelhető külön ráta.

Az ágensek tárolásához egyszerre két adatszerkezetet használ a szoftver, így egyesítve mindkettő előnyeit.

Az egyik adatszerkezet egy láncolt lista, a programban AgentList néven. A listára azért van szükség, mert a szimuláció több lépésében is végig kell iterálni az összes ágensen valamilyen vizsgálat céljából. A lista adatszerkezet pedig pont erre való, gyorsan lehet bejárni.

A másik adatszerkezet egy kétdimenziós tömb, AgentArray néven. A tömböt szomszédsági vizsgálatokra használom, mert adott koordináták elérésére ez az adatszerkezet gyorsabb. Megvan az az előnye is, hogy nem tudok túlindexelni, hiszen hibát dob a program, és ezzel nem tudok nem létező helyekre véletlen ágenst mozgatni.

Eredetileg csak a lista szerkezetet használtam, de belefutottam két problémába.

Az első, hogy a lista szerkezetben nagyon gyors folyamat iterálni, de szomszédságvizsgálatot végezni a lista hosszával arányosan egyre lassabb folyamat. Erre lett megoldás a tömbhasználat, viszont két helyen voltam lénytelen fenntartani a konzisztenciát.

A második probléma pedig az, hogy a listák hossza változhat adott műveletek során. Például, ha egy sejt elpusztul, akkor a lista rövidebb lesz egyel, de az iteráló ciklus ezt nem tudja lereagálni, így túl fog indexelni a tömbön. Ennek a megoldása ilyen műveletek során egy másolat lista készítése. A lemásolt listán iterálok végig, de az igazi listán végzem a műveleteket.

5.1.3. Manager

A Manager osztály feladata a felhasználó felé biztosítani egy interface-t. A Manager végzi a SimulationStateObject létrehozását, felparaméterezését és előkészítését az ágensek fogadására.

A létrehozás több lépésből áll, mivel a felhasználó nem tudja egyszerre megadni az összes adatot.

Az első lépés a létrehozás után a felparaméterezés. A Manager inicializálja az adatszerkezeteket, beállítja a szimulált terület méretét, majd falakat helyez a terület széleire, így megakadályozandó a túlindexelés és a vegyi anyagok pályáról lévő elszivárgása.

A következő lépésben tetszőleges számú vegyi anyagot vehet fel a felhasználó, majd a művelet végén a Manager inicializálja a vegyi anyagok tömb alapú változóit. Ezeknek a dimenzióit később nem lehet megváltoztatni, így új vegyi anyag sem vehető fel a lépés után.

A felhasználó ezután létrehozhatja az állapotátmenet gráfot, majd ágenseket és vegyi anyag forrásokat helyezhet el a szimulált térben.

A következő feladat az ágensek számának validálása. A Manager ellenőrzi, hogy az ágenslistában és -tömbben ugyanazon számú ágens található.

A validáció után következik a fő funkcionalitása a Managernek, maga a szimuláció. A szimulációnak két hívható függvénye van. A NextEnvVarTick() a vegyi anyagok szimulációjának számolja ki a következő állapotát, míg a NextAgentTick() az ágensek oldalán teszi ugyanezt.

Azért kellett a két műveletet szétválasztani, mert a vegyi anyagokat, hogy a modelljük stabil legyen, sokkal kisebb időosztással kell frissíteni. Az ágensek ugyanakkor sokkal lassabb frissítési aránnyal is pontos eredményt adnak, mivel a sejtek általában sokkal ritkábban érnek el szignifikáns változást. Ez percek, de inkább órák időléptékében értendő.

A következő fejezetekben a főbb szimulációs folyamatokat fejtem ki.

5.1.4. Mozgás

A mozgás alap algoritmusa egyszerű megközelítést használ. Végigiterál az összes sejten, megvizsgálja, hogy az adott sejt mobilis-e, aztán megpróbálja elmozdítani a sejtet.

A mozgás irányát véletlenszerűen generálja, majd megvizsgálja az ágenstömb-beli, megfelelő irányú szomszédságot. Ha szabad a cél cella, akkor a tömbben lemásolja a sejtet a célba, majd az eredeti helyről törli. Az ágens listában pedig frissíti az adott ágens megfelelő koordinátáját.

A mozgás érdekesebb aspektusa a sebesség, ugyanis egy rácshálón foglalnak helyet az ágensek, csak diszkrét pozíciót vehetnek fel. A problémát úgy oldottam meg, hogy az ágens helyváltoztatásának a valószínűségét számolom ki, aztán véletlengenerálok egy számot. Ha a valószínűség kisebb a random számnál, akkor lépek, ha nem, akkor pedig marad a helyén a sejt.

A valószínűség meghatározásához a következő paramétereket használom fel:

- A sejt aktuális állapotának a sebessége
- A sejtekre vonatkozó időosztás
- Szimulált terület négyzetrácsának a nagysága

A képlet pedig a következő:

Valószínűség = Sebesség * Időosztás / Négyzetrács mérete

5.1.5. Állapotátmenet

Az állapotátmenet algoritmus a mozgáséhoz hasonló egyszerűségű. Végigiterál egy másolat ágens listán - amire azért van szükség, mert az elpusztult sejteket el kell távolítani – és mindegyikre meghívja a valószínűség normalizáló függvényt. A normalizálást követően pedig meghívja a Gráf következő állapotot meghatározó függvényét.

A normalizálásra azért van szükség, mert adott állapotból akárhány átmenet mutathat más állapotokba és az átmenetek valószínűsége egyenként egy nulla és egy közé eső szám. A Gráf következő állapotot kiszámító függvénye viszont szintén egy nulla és egy közé eső össz valószínűséget vár az átmenetektől, vagyis a valószínűségek összege nem haladhatja meg az egyet.

5.1.6. Osztódás

Az osztódás során az algoritmus letesztel minden osztódásra kész állapotban lévő sejtet, hogy elérték-e az osztódáshoz szükséges életkort. Ha ez teljesül, akkor megvizsgálja a sejt négy irányú szomszédságát és ha talál helyet az új sejtnek, akkor lehelyez egy ágenst az alapértelmezett állapotban.

Az életkort azért kell tesztelni, mert a sejtek létrejöttüktől kezdve átmennek különböző növekedési fázisokon és csak egy adott fázis után lesznek osztódóképesek. Ezt a

folyamatot nem szimulálom le, de szimbolizálom ezzel a növekedési időtartammal. Az időtartam beállítható a szimulációs terület létrehozásakor.

5.1.7. Diffúzió és Anyagfelvétel

A vegyi anyagokat egy háromdimenziós mátrixban tárolja a program. A harmadik dimenzió mérete a vegyi anyagok számától függ. A koncentrációt százalékos formátumban tárolom, nulla és egy közé eső számokként.

A diffúziós algoritmus bejárja a vegyi anyag mátrix mindhárom dimenziójának az összes elemét. A Gammadict szótár segítségével és az FTCS módszerrel kiszámolja a diffúzió eredményét.

A feladat jól párhuzamosítható, mert az aktuális koncentrációt csak az előző időpillanatban jelenlévő koncentrációból számolja az algoritmus, ezáltal nem tud versenyhelyzet kialakulni. Akár az összes cella kaphatna saját szálat.

Sarkalatos pontja a diffúziónak a falak kérdése, mert nem lehet őket egyszerűen maximális vagy minimális értékre állítani, mert akkor forrásként vagy elnyelőként működnének a falak. Ehelyett a falak alatti értéket mindig a legközelebbi nem fal szomszéd értékéhez igazítom, így a falak nem befolyásolják a kísérlet eredményeit.

Az anyagfelvétel sejtszinten történik. A vegyi anyag mátrix megfelelő pontján, a sejt adott anyaghoz tartozó felvételi értékével csökkentem a koncentrációt.

5.2. Adatgenerátor

Az adatgenerátor az Utils osztály néhány statikus függvényét jelenti. Az adatgenerátor célja, hogy strukturált módon állítson össze egy adathalmazt, amit a szimuláció futása alapján állít össze.

A PrintSimState függvény egy SimulationStateObject alapján képez adatokat. Kiírja egy, az egyszerűség kedvéért CSV formátumú, fájlba a következőket:

- Ciklusszám
- Ágensek száma
- Ágensek eloszlása állapot szerint
- A ciklus kiszámításához elhasznált idő
- Vegyszerek átlagos koncentrációja vegyszerenként lebontva

Minden ciklus adata egy új sorba kerül. A program minden futása új fájlt generál, a fájlnév pedig a kísérlet indításának az ideje.

A PrintAgentsInArea függvény célzottabb elemzést tesz lehetővé. A függvény paramétereként megadható két koordináta, amiket felhasználva a függvény összeszámolja a koordináták által leírt négyszögbe eső ágensek számát.

Az előző függvényéhez hasonló fájlba kerül eltárolásra az eredmény. A fájlnév a -processed végződést kapja.

5.3 Vizualizáció

A vizualizációért az Utils osztály egy különleges függvénye felel. Paraméternek egy SimulationStateObject objektumot, egy pixelben adott méretet és az adott szimulációs ciklus sorszámát veszi be.

A szimuláció indításakor az aktuális időpontnak megfelelő nevű mappa készül, ahova a függvény képsort fog előállítani.

Minden egyes ciklusban a státusz objektum alapján rekreálja a szimulált területet egy kép formájában. A képen négyszögekként jelennek meg az ágensek és a falak. Az ágensek színe az állapotuktól függően egy színskálából lesz kiválasztva.

Ez a képkészítő függvény a System.Drawing csomagot használja, ami a GDI+ könyvtárhoz ad hozzáférést.

5.4 Felhasználói felület

13. ábra. Reprezentatív képernyőfotó a program főképernyőjéről szimuláció közben.

A felhasználói felületet a Windows Presentation Foundation keretrendszert felhasználva készítettem el. A szimuláció előkészítése során a felhasználó intuitív módon, több ablakon keresztül, tudja a különböző paramétereket beállítani.

A felhasználót egy több fázison átívelő folyamaton vezeti végig a program. Az aktuális fázist bal lent a Status feliratnál jelzi a felület. Minden fázis végén a Next Phase gombra kattintva a felhasználó átléphet a következőbe. Ha a gomb nem elérhető, akkor a felhasználó még nem végzett el egy tovább haladáshoz szükséges feladatot.

A fázisok a következők:

- NoDish: Még nincs beállítva a szimulációs felület
- EnvVarEdit: Most adhatóak a szimulációhoz a vegyi anyagok. Legalább egy anyagnak lennie kell a továbbhaladáshoz.
- GraphEdit: Most hozható létre a gráf a megfelelő menüből.
- MapEdit: Most helyezhetők az ágensek és az anyag források a játéktérre.
- Simulation: A szimuláció fut.

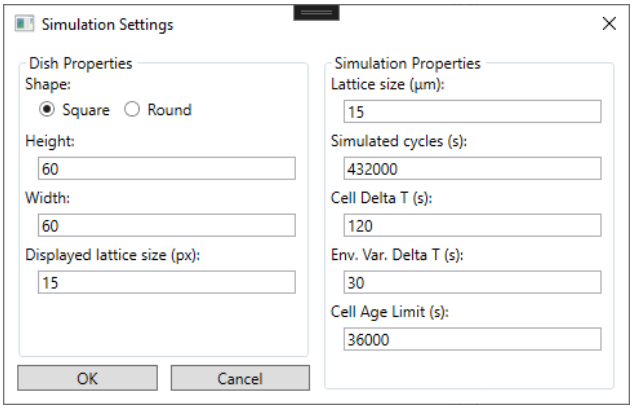
Az ablakokat a legtöbb esetben az MVVM patternnek megfelelően készítettem el. Minden ablak saját viewmodellel rendelkezik, ami a lokális működéshez szükséges kódon kívül csak adatokat tartalmaznak.

6. TESZTELÉS ÉS VALIDÁCIÓ

Ebben a fejezetben bemutatom, hogy a felhasználónak milyen lehetőségei vannak a program használata során, kitérve a különféle ablakok szerepére, a szimuláció fázisaira és a kimeneti adatok szerkezetére. A fejezet második felében pedig egy konkrét kísérlet valós és szimulált eredményeit fogom összehasonlítani.

6.1. Felhasználási lehetőségek

Egy kísérlet előkészítésének az első lépése a szimulált terület paramétereinek a beállítása. Ezt a Dish (mint petri-csésze) menü New Dish pontjában lehet beállítani.

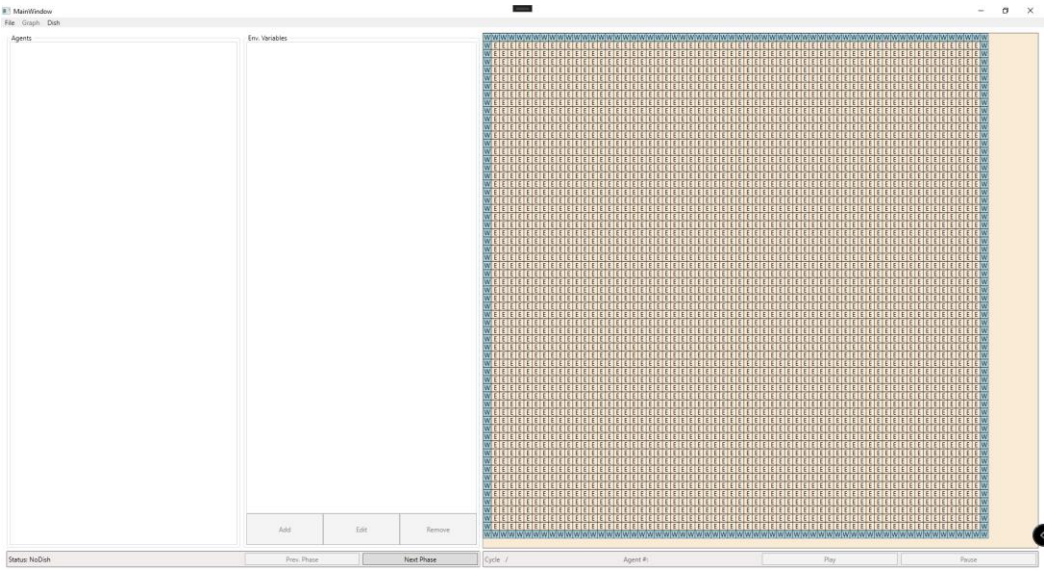


14. ábra. A szimuláció tulajdonságait beállító felület

Az ablak Dish Properties-zel jelölt felében az adatszerkezeti dimenziókat lehet beállítani, mint a szimulált tér alakja, szélessége, magassága és egy négyzetrács megjelenítési mérete. Az osztásméret mikrométerben, az időegységek másodpercben adhatók meg.

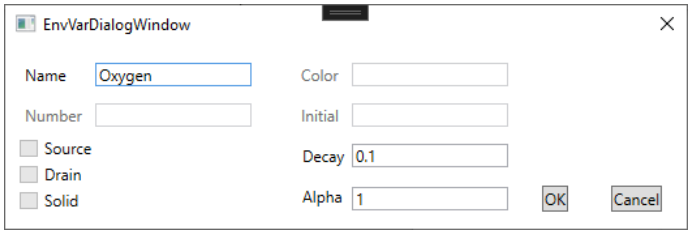
A Simulation Properties tulajdonságok pedig a szimuláció logikai paraméterei. Beállítható a négyzetrács logikai mérete, a szimulált időtartam, a sejtek és vegyi anyagok időosztása és az osztódáshoz szükséges minimális sejt életkor.

Ha a felhasználó elégedett a beállításokkal, akkor az OK gombra kattintva véglegesítheti azokat. Az ablak bezárul és a fő ablakban megjelenik a szimulált tér. A kék színű, W-vel jelölt négyszögek falak, míg az E-vel jelölt négyszögek egyelőre üresek.



15. ábra. A főképernyő a szimuláció létrejötte után

A következő fázisban adhat hozzá a felhasználó vegyi anyagokat. Az Env. Variables. lista alatt elérhetővé válik az Add gomb. Rákattintva a következő ablakban meg lehet adni egy vegyi anyag tulajdonságait.



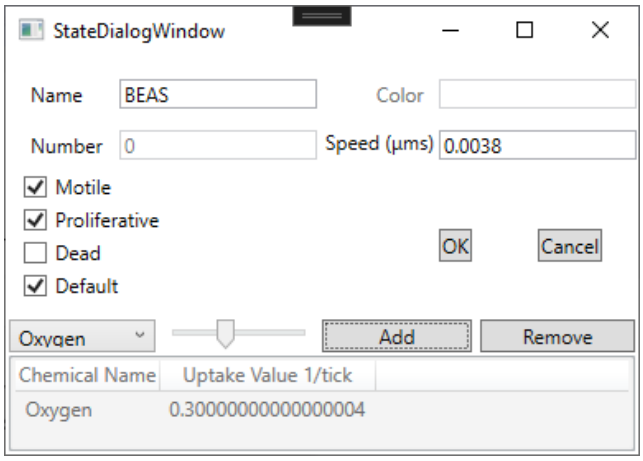
16. ábra. Egy vegyi anyag tulajdonságait felvevő ablak

A név tulajdonság azonosítás céljából adható meg. A Decay tulajdonság a vegyi anyag természetes bomlási aránya százalékos formában. Itt a példában minden időosztásban 10%-kal csökken a koncentráció. Az Alpha tulajdonság hővezetési együttható, ami a diffúzió sebességével arányos.

Az OK gombra kattintva a vegyi anyag hozzá lesz adva a listához.

A következő fázis a Gráf beállítása. Ez a Graph menü New Graph pontjában érhető el. Az új ablakban állapotok és átmenetek hozhatók létre.

Egy állapot létrehozása a következő felületen történik.



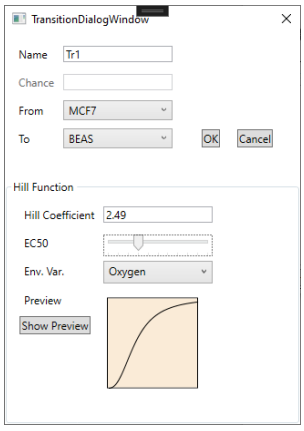
17. ábra. Egy állapot tulajdonságainak a beállítása

A névvel azonosítható az állapot. Ha a Motile checkbox be van jelölve, akkor meg lehet adni a sejt sebességét. A sebesség az ágens elmozdulásának a valószínűségét növeli, a mértékegysége mikrométer per másodperc. A Proliferative checkbox osztódási lehetőséget biztosít az ebben az állapotban lévő sejtnek. A Dead checkbox eltávolításra jelöli azt a sejtet, ami ebben az állapotban van. A Default checkbox pedig ezt az állapotot jelöli meg alapértelmezettként.

Az ablak alján lévő legördülő menüből kiválasztott vegyi anyagot a csúszkával megadott ütemben veszi majd fel a sejt, ha a felhasználó hozzáadja az állapot szótárához az Add gombbal.

Az OK gombbal véglegesíthető az állapot.

Egy átmenetet a következő felületen lehet beállítani.



18. ábra. Egy állapot-átmenet tulajdonságai

A start és cél állapotokat a From és To legördülő menükből lehet kiválasztani. Az átmenet valószínűségét a Hill függvény paramétereinek változtatásával lehet beállítani. A Hill együttható a függvény meredekségét befolyásolja, míg az EC50 az 50%-os

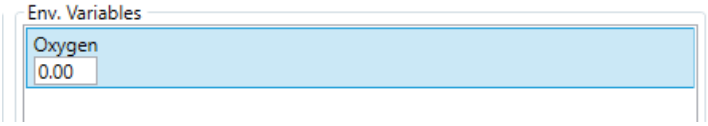
reakció helyét jelöli ki. Az Env. Var. legördülő menüből pedig a vegyi anyag választható ki, aminek a koncentrációja alapján fog a Hill függvény valószínűséget képezni. A Hill függvény értelmezési tartomány és értékkészlete is nulla és egy közé eső szám. Az X tengely a vegyi anyag koncentrációját, az Y tengelye a reakció (itt az átmeneti valószínűség) százalékos esélyét fejezi ki. A Hill együttható negatív számra való beállítása egy inverz Hill függvényt eredményez.

A felhasználónak nem kötelező átmeneteket megadni, minimum egy állapotnak azonban lennie kell. Így homogén sejtek halmazán lehet kísérletezni.

Az állapotok és átmenetek véglegesítése után a felhasználó a következő fázisba léphet.

A fő képernyőn megjelentek az állapotok. Az állapotokat kiválasztva és a szimulált térre kattintva helyezhetőek el ágensek a szimulációban. Az elhelyezett ágens a kiválasztott állapottal fog rendelkezni.

Ha a felhasználó egy vegyi anyagot választ ki a jobb oldali oszlopból, akkor a szimulált teret megjelenítő ábrán az ágensek helyett a kiválasztott vegyi anyag forrásai jelennek meg.



19. ábra. Egy vegyi anyag forrás elhelyezéshez kiválasztva

A listában a vegyi anyag neve alatt található egy beviteli mező. Ez a mező a vegyi anyag forrás termelési sebességét adja meg. A mértékegysége arány per vegyi anyag időosztása, azaz az adott rácsponton a vegyi anyag koncentrációja minden időosztásban az adott számmal fog nőni. A szimulált térre kattintva helyezheti el a felhasználó a forrást az ágensekhez hasonlóan.

Az ágensek és vegyi anyagok lehelyezése után a felhasználó elindíthatja a szimulációt. A szimuláció során a képernyő időközönként frissül, így láthatja a felhasználó a sejtek mozgását. A frissítés viszonylag nagy időközönként történik, ez a WPF limitációja, viszont az adatgenerátor és vizualizációs szoftverrész minden egyes ágensekhez tartozóidőosztásban rögzíti az adatokat.

Ezek az adatok a logs mappában találhatók. A kísérlet indításának az időpontja a fájlnév, illetve a képsort tartalmazó mappa neve.

A „data” végződésű fájlból eloszlási adatokat lehet kinyerni, a „processed” végűből pedig egy megadott területen lévő ágensszámot. A képek látványos, színes reprezentációi a szimulációnak, a színek az ágensek állapotait jelölik.

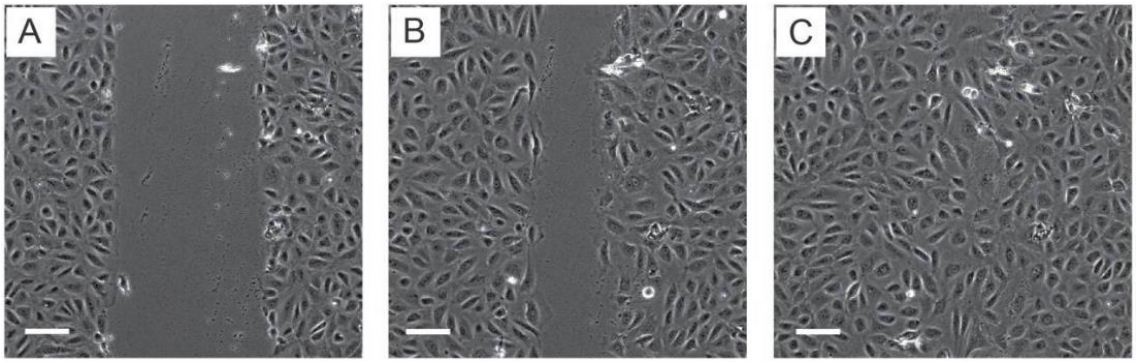
A szoftver jelenlegi verziójában a szimuláció befejezése után új szimulációt kezdeni csak a program újraindításával és újra konfigurálásával lehet. Egy jövőbeli verzióban mentés-betöltés rendszerrel egészítem majd ki a programot.

6.2. Wound healing assay

6.2.1. A kísérlet biológiai háttere

A wound healing assay (magyarul nagyjából sebgyógyulási eljárásnak fordítható) egy általánosan alkalmazott in vitro kísérleti eljárás, amely különféle sejttípusok mozgékonyságának karakterizálására alkalmazható. Az eljárás lényege a következő.

A tenyésztőedénybe kiültetett sejteket konfluens (közel teljesen benőtt) állapotig a protokoll szerinti körülmények között tartják. Ezt követően *sebet* ejtenek a sejtekkel benőtt tenyésztőedény felszínén azzal, hogy egy sztenderd szélességű eszközzel (ez általában egy pipettahegy) megkarcolják az edény felszínét, ezzel eltávolítva onnan egy fix szélességű sávban a sejteket (lásd a 20. ábrát). A sejtek mozgásának következtében a seb általában 10-30 óra alatt záródik, a folyamatot rendszeres időközönként készített mikroszkópos felvételek elemzésével (amelyet manapság gyakran automatizáltan végeznek) követik nyomon. A sebzáródás sebességéből következtetni lehet a sejttenyészet egyedeinek átlagos mozgékonyságára: a motilisabb sejtek hamarabb, a kevésbé motilisak (pl. valamilyen hatóanyaggal kezelt sejtek) később zárják a sebet.



20. ábra. A sebzáródás folyamata a wound healing assay során. (A) Kezdeti seb, (B) részben záródó seb, (C) a seb teljes záródása. Forrás: [5]

6.2.2. A kísérlet modellje és szimulációja

A kísérlet szimulációjához a [5] forrásban leírt protokollt vettem alapul. Az ábrán látható elrendezés közelítéséhez egy 64 × 32 egység méretű virtuális tenyésztőedényt hoztam létre. A szimulált tér osztását 15 mikrométerre állítottam, így a szimulált terület mérete nagyjából megfelel a kísérletéhez. A szimulált időintervallumot öt napra, azaz 50400 szimulált ciklusra állítottam be. A sejtek időosztása 120 másodperc. Vegyi

anyagra és osztódásra nem volt szükség a szimuláció során, így mindegy, hogy mennyi a vegyi anyag időosztás és a sejtek életkor limitje.

Mivel a következő fázisba lépéshez szükséges legalább egy vegyi anyag, ezért hozzáadtam egy dummy vegyi anyagot a listához, de nem használtam és nem is helyeztem le forrást.

Két sejttípust hasonlítottam össze a kísérlet során, a gyorsabb BEAS és a lassabb MCF7 típusokat. Mindkét sejt beállításainál bejelöltem a Motile flaget.

A BEAS sejtek 14 mikrométert tesznek meg óránként, míg az MCF7-es sejtek pedig 7-et. Ezeket az értékeket át kellett váltani mikrométer per másodpercre.

A BEAS sejt sebessége 0.0038, az MCF7 sejté pedig 0.0019-re lett állítva.

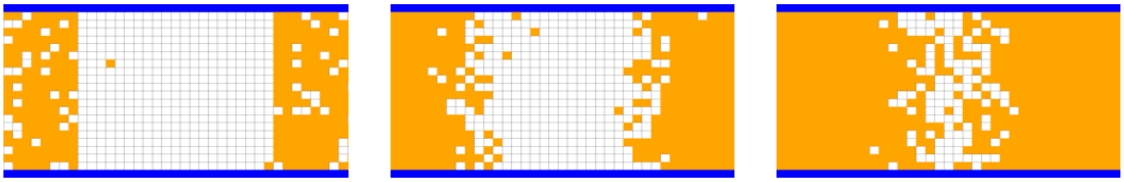
Állapotátmenetet nem kellett beállítani.

Az ágensek lehelyezését a szimulációs térbe egy seed függvénnyel végeztem, mivel a WPF felületen ennyi ágenst manuálisan lehelyezni hosszabb időbe telt volna, mint a kísérlet futtatása. A seed függvény a szimulált terület két szélső harmadára 85 százalékos telítettséggel helyezett el ágenseket.

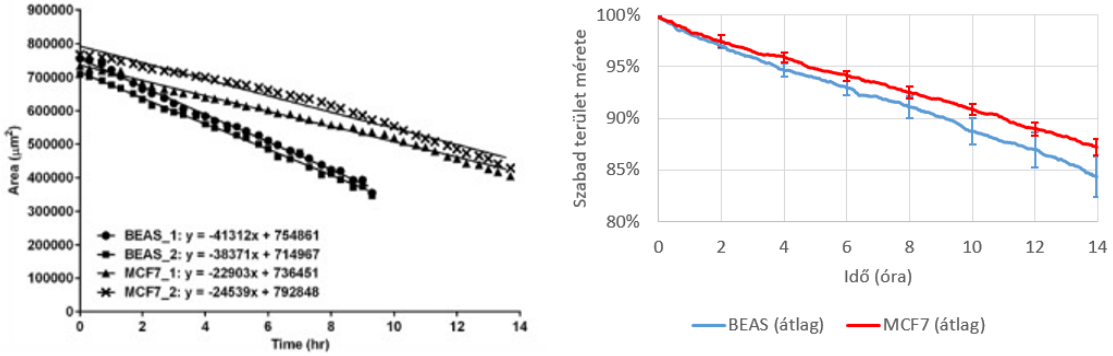
6.2.3. Szimulációs eredmények és diszkussziójuk

Az előbbiek alapján felépített szimulációt több ismétléssel futtattam le, ezzel csökkentve a véletlen hatását, amely a felhasznált szimulációs technika sajátossága. A szimulációk során szabályos időközönként kimeneti fájlba mentettem az ágensek jellemzőit, köztük a pozíciót. Ezek alapján lehetőségem volt megadni, hogy a sebet szimbolizáló, kezdetben üres sávot az idő előrehaladtával milyen mértékben töltik ki a véletlenszerűen mozgó ágensek. A kinyert adatsorok a 22. ábrán láthatóak, összehasonlításképp az eredeti biológiai kísérletből kapott adatokkal együtt. A szimulációs eredmények egyfelől jelentős kvantitatív különbséget mutatnak a biológiai kísérlethez képest, ugyanis a sebzáródás mértéke közel sem éri el a valóságban tapasztaltat. Másfelől ugyanakkor hűen visszaadja az kísérletben kapott kvalitatív különbséget a két sejttípus között, vagyis a valóságban lassabbnak karakterizált sejttípus a szimulációban is lassabban zárja a sebet.

A kvantitatív eltérés megmagyarázható a modell egyszerűségével: a valóságban a sejtek nem diszkrét pozíciókat elfoglaló pontok, amelyek üres szomszédos rácspontokra léphetnek, a mozgásuk ennél jóval bonyolultabb dinamikát követ (az alakjuk dinamikusan változik, képesek összenyomódni), ezen felül a környezetüket pásztázva valószínűleg a mozgásukra sem a teljesen véletlenszerű viselkedés a jellemző, hanem inkább az üresebb térrészek felé történő orientáció.



21. ábra. Reprezentatív ábra a sebzáródás assay szimulációjából. A sebet jelképező, kezdetben közel üres sávot az idő előrehaladtával folyamatosan kitöltik a sejtek.

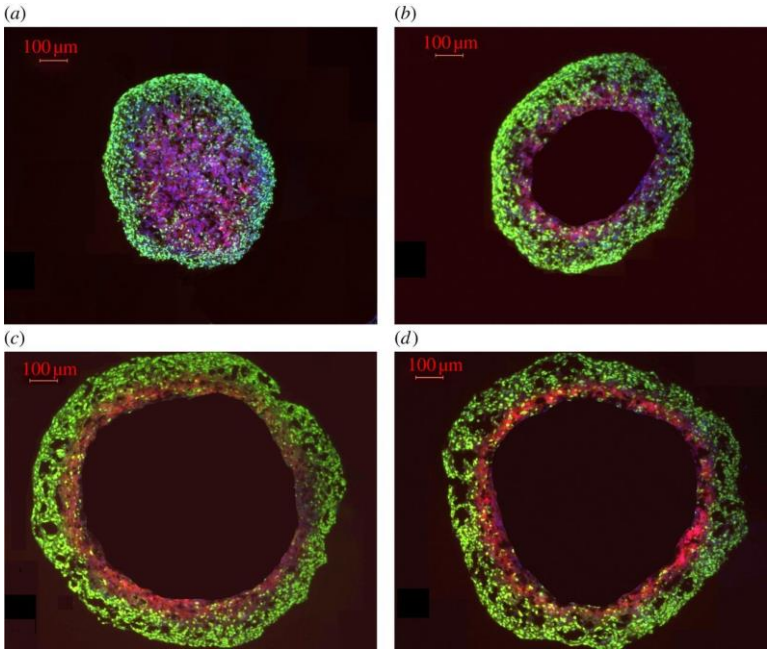


22. ábra. BEAS és MCF7 sejttípusok sebzáródási kísérleteiből kapott adatsor (balra, [5] alapján, módosítva), valamint a szimuláció eredménye (jobbra). A jobb oldali grafikon függőleges tengelye nem 0%-nál kezdődik, mivel a szimuláció nem tudta kvantitatívan visszaadni a valódi kísérletnél megfigyelt dinamikát, de a két sejttípus közötti minőségi különbség látható (n=4, átlag ± sztenderd hiba)

6.3. Viable rim kialakulása

6.3.1. A kísérlet biológiai háttere

Tenyésztőedényben kialakuló mesterséges tumor szferoidok egyik jellegzetes tulajdonsága az ún. hypoxiás mag kialakulása. Ez a mag a tumor belső térrészében lévő, oxigénhiányos sejtekből álló szövetrész, ami azért alakul ki, mert a sejtek felhasználják a tumor felszíne felől a belső régióba diffúzióval bekerülő oxigént, amely mérettől függően egy idő után már nem képes megfelelő mennyiségben a szövet belsejébe jutni. Az oxigénben kellően gazdag felszíni sávot szokás *viable rim*-ként, vagyis élő sejteket tartalmazó gyűrűként hivatkozni. A hypoxiás régiók kialakulása a klinikumban azért lényeges jelenség, mert a hypoxiás állapotban lévő sejtek sokkal jobban ellenállnak a sugárkezelésnek, mint az oxigénnel jól ellátott sejtek [3]. A tumor belseje felé haladva az oxigénkoncentráció olyan alacsonyra esik, hogy a tumor közepe anoxiás állapotba, azaz teljesen oxigénhiányos állapotba kerül, a tumor közepén lévő sejtek így idővel elpusztulnak.



23. ábra. A tumor szferoid kezdeti állapotban (a) normál sejteket tartalmaz. A későbbi képeken (b-d) jól látható az ún. *viable rim* (a zöld színű gyűrű), a hypoxiás sejtek (piros) és az anoxiás régió (fekete) a tumor magjában. Forrás: [3]

6.3.2. A kísérlet modellje és szimulációja

A szimulált kísérletet egy 50 × 50-es térben hajtottam végre, a tér osztását 15 mikrométerre állítottam. A szimulált időintervallum három nap, az időosztás a sejtek részéről 120 másodperc, a vegyi anyagok időosztása 30 másodperc volt.

Az oxigént reprezentáló képzeletbeli anyagot egy 1.0 négyzetmikrométer/másodperc diffúziós együtthatójú, 0,0001 1/s bomlású vegyi anyagként hoztam létre. Az oxigén a szimulált tér szélein folyamatosan termelődik 0.5 1/s aránnyal, a kísérlet kezdetén a koncentrációja az egész térben 100%-os.

Az ágenseknek két állapotot határoztam meg. A normál állapotot N betűvel jelöltem, a kimeneti képeken sárga a színe. Normál állapotban az ágensek képesek mozgásra 0,0013 mikrométer/másodperc átlagsebességgel. Az ebben az állapotban lévő sejtek 0,001 1/s rátával veszik fel az azonos cellában található oxigént. Normál állapotban osztódhatnak az ágensek, ha az életkoruk elérte a 72000 másodpercet.

A hypoxiás állapotú sejt jele H, a színe a képi kimeneten zöld. Ezek a sejtek nem mozognak, nem veszik fel az oxigént és nem osztódnak.

A fenti két állapot között kétirányú átmenet van. A normálból hypoxiás állapotba vezető átmenet Hill együtthatója -2.49 az oxigén koncentrációhoz kapcsolódva, az EC50

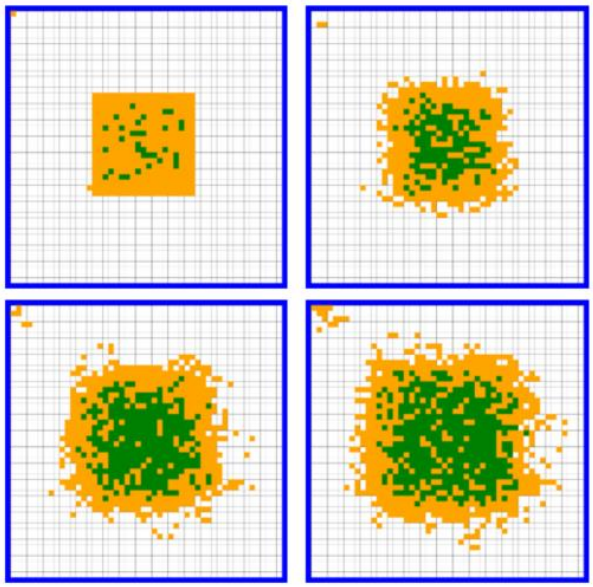
paramétere 10. Az ellentétes irányú, hypoxiából normál állapotba vezető átmenet paraméterei hasonlóak. Hill együttható 2.49, EC50 értéke 10. Röviden az oxigén koncentrációval fordítottan arányosan kerülhet egy ágens hypoxiás állapotba és az oxigénszint növekedése tudja ebből az állapotból kimozdítani.

A kezdeti sejttelep a szimulált tér közepére helyezett, 20 × 20-as méretű, normál állapotú sejtekből álló csoport. A paramétereket a [3] irodalmi forrás alapján, illetve experimentális módon állítottam be és finomhangoltam.

6.3.3. Szimulációs eredmények és diszkussziójuk

Az előző kísérlet szimulációjához hasonlóan most is több ismétléssel futtattam szimulációkat a véletlen hatásának csökkentésére. Az eredmények megjelenítéséhez a sejtek jellemzőit (pozíció, állapot) adott időközönként kimeneti fájlokba mentettem, amelyek alapján később megjeleníthetők és elemezhetők az eredmények.

Az oxigént reprezentáló képzeletbeli anyag diffúziós együtthatóját a valódi oxigénhez képest több nagyságrenddel kisebbnek választottam, amire a választott időlépés, valamint az alkalmazott FTCS numerikus közelítés miatt volt szükség, a módszer ekkor volt használható. A további paraméterek arányos módosításával (finomhangolásával) a szimuláció kvalitatív egyezést mutatott a valódi kísérleti eredményekkel, vagyis megfigyelhető volt a hypoxiás mag kialakulása (24. ábra). Az is látható volt a megjelenített eredményeken, hogy a *viable rim* mérete közel azonos maradt a sejttelep (szferoid szelet) tényleges méretétől függetlenül, ahogyan az a valódi kísérletekben is megfigyelhető volt.



24. ábra. A szimuláció folyamán generált képi adatok. A normál állapotú sejtek színe sárga, a hypoxiás sejtek színe zöld. A normál sejtek alkotják a *viable rim*-et.

7. TOVÁBBFEJLESZTÉSI LEHETŐSÉGEK

Ebben a fejezetben összefoglalom a mind a szimulációs, mind a felhasználói élménybeli hiányosságokat, majd javítási vagy kiegészítési ötleteimet is kifejtem.

7.1. Limitációk

A szimulációs modell legjelentősebb korlátozottsága a négyzetrács alapú modell használata. Négyzetrácson valóban gyorsabb számolni és egyszerűbb, diszkrét függvények használhatók, de így kénytelen voltam feláldozni bizonyos szintű pontosságot. A négyzetrács hátránya viszont egyben az előnye is, hiszen a szimulált négyzetrács méretének a beállításával legalább tudatosan meghatározhatom a pontatlanság mértékét.

A modell másik hátránya, ami az előző fejezetben is megmutatkozott az az ütközéseket szimuláló alrendszer hiánya. A kísérlet azért nem vezetett pontos eredményre, mert a sejtek csak önerőből tudtak mozogni. A valóságban azonban sok esetben az ütközéseknek is nagy szerepe van, hiszen így a sejtek nem csak mozognak, hanem a szomszédjaik is „lökdösik” őket, így nagyobb átlagsebességre tennének szert.

A felhasználói felületnek is vannak korlátosságai. Az egyik ilyen a sebessége. A felület a WPF keretrendszert használja, ami szöveges felületek és grafikák gyors megjelenítésére kiválóan alkalmas. Ebben az esetben viszont nem csak megjeleníteni kell az ágenseket, hanem a szimulációs térre le is kell őket helyezni. Ehhez kattintásérzékelés szükséges, ráadásul az üres helyek is valójában objektumok a megjelenített pályán. Mivel nem akartam megszegni az MVVM patternt, ezért ezek az objektumok mind külön paranccsal rendelkeznek, amit végrehajtanak, ha rájuk kattintok. A probléma az, hogy nagy pálya esetén, ahol sok objektumot kell frissíteni minden egyes kattintás után, aztán adatkötéssel meghatározni a helyzetüket és kirajzolni őket, nagyon lassú a frissítési idő. Több másodperces időről van szó. Ez egy reszponzív programhoz túl lassú.

Az ágenseket ezen felül csak egyesével lehet lehelyezni, ami nagy ágensszám esetén nem hatékony megoldás.

Talán a legnagyobb Quality of Life hiányosság a mentési-betöltési rendszer hiánya. Így minden egyes kísérletet manuálisan kell felparaméterezni, az ágenseket pedig kézzel kell elhelyezni. Ez az egyező paraméterekkel rendelkező, többször megismételt kísérleteknél érezhető leginkább.

7.2. Fejlesztési lehetőségek

A legelső probléma, amit megoldanék, az a mentési rendszer hiánya. A programomat egy szerializáló alrendszerrel egészíteném ki, ami JSON vagy valami hasonló leíró

formátumban el tudná menteni külön az állapotátmenet gráfot és esetleg egy kezdő állapotot, hogy csak egyszer kelljen a szimuláció kezdő feltételeit megadni.

A kezelőfelület lassúságát úgy javíthatnám ki, hogy megtöröm az MVVM pattern szabályait és alacsonyabb szinten, code-behind-ból végezném a képernyő frissítését. Elválaszthatnám a kattintásérzékelést a megjelenítéstől. Esetleg másik keretrendszert használhatnék az ablakkezelésre, ami gyorsabb a WPF-nél.

A szimulációs modell pontosságán javíthatna, ha átállnék egy négyzetrács nélküli modellre. Egy ilyen modellben könnyebb lenne implementálni a sejtek közötti kölcsönhatásokat, viszont folytonos függvényekkel kellen számolni, ami processzorigényesebb művelet.

Felhasználói interakció szempontjából lehetne fejlesztés az állapotátmenet valószínűségének a megadásához egy felhasználó által írt képlet használata, így nagyobb szabadságot biztosítva a felhasználónak és akár több vegyi anyag is felhasználható lenne egy képleten belül.

8. ÖSSZEGZÉS

Amióta megfelelő szintre fejlődött az informatika, nehéz elképzelni olyan mérnöki, gazdasági, ipari vagy más problémát, ahol a probléma szimulációval történő felderítése, esetleg megoldása ne jelenne meg. Nincs ez máshogy a sejtbiológia területén sem. Általában a szimulációs modelleket élő szervezetekben vagy in vitro kísérletek során tapasztalt működések replikálására használják.

A sejtbiológia területén az jelent technikai jellegű és szoftverfejlesztési kihívást, hogy nincs általános célú szimulációs szoftver, így azt a kutatóknak kell sajátkezűleg megtervezni és implementálni. Mivel a kutatók általában nem programozók, így sok időbe telhet nekik elsajátítani a megfelelő ismereteket és még több idő megalkotni a modellt.

A szakdolgozatom célja az volt, hogy megalkossak egy könnyen használható, általános célú, gyors és megfelelő nagyságrenden belül pontos sejtszimulációs szoftvert. A szakirodalom tanulmányozása és a modellezési lehetőségek felfedezése után megterveztem és elkészítettem egy intuitív felhasználói felülettel rendelkező, felhasználó által személyre szabható és paraméterezhető szimulátort. A szimulátor elkészítése után az irodalomkutatás során megismert két kísérlet szimulált megvalósításával képet kaptam a szimulátor pontosságáról és limitációiról. A paramétereket a szakirodalom alapján állítottam be, majd experimentális módon finomhangoltam.

A megvalósított szoftver több irány mentén is továbbfejleszthető. Ha maradok az alkalmazott, diszkrét térben történő szimuláció mellett, akkor a futási teljesítmény növelését a párhuzamosítás szintjének emelésével lehetne elérni. Erre az egyik lehetőség a klaszterezés, azaz a tér kisebb részegységekre osztása és külön-külön szimulálása. A klaszterezés ezen felül lehetővé tenné nagyobb számú ágens szimulációját.

Egy másik irány a rács nélküli térben történő szimuláció az ágensek részéről, ahol megvalósíthatók olyan mechanikai jellegű interakciók, mint a rugalmas ütközés, az ágensek fizikai alakjának megváltozása, vagy esetleg az ágensek egymáshoz kötődése.

A felhasználói élményt reszponzívabb felülettel és többféle adat exportálásával, illetve a beállított paraméterek és szimulációs állapotok mentésével lehetne javítani.

9. SUMMARY

Since the IT industry advanced to the appropriate level, it is hard to imagine solving engineering, economic or industrial problems without at least preliminary simulations. Such is the case in the field of cell biology. Generally, simulations are used to replicate processes observed in living organisms or in *in vitro* experiments.

In cell biology, the challenge comes from the fact that there are no general purpose simulators. Researchers are forced to design and implement their own softwares. As researchers in this field are seldom programmers, it takes a considerable amount of time for them to learn the appropriate skills and even more to implement a working model.

In the thesis my goal was to create an easy to use, general purpose cell simulator that is also fast and accurate to a certain degree. After studying the corresponding technical literature, particularly the aspects of modeling, I designed and implemented my own simulator. This software is a user friendly, customizable, out of the box style simulator. Many parametres can be set, and the user has opportunities to define other ones. After completing the simulator, I used two experiments described in the literature to ascertain the accuracy and limitations of my software by replicating those experiments *in silico*. The parametres were set based on literature, and later fine-tuned by experimentation.

The complete product could be iproved upon in many different ways. If I stick to the already implemented discrete space simulation, I could increase performance increasing the level of parallelism. I could achieve it by using clustering. In short, I would divide the simulated space into regions and simulate them in a parallel fashion. Furthermore, by using clustering, I could increase the number of simulated agents significantly.

Another direction would be to abandon the discrete space in favour of a continuum space simulation, at least concerning the simulation of agents. This would create significant overhead and would reduce parallelism, but would also allow for more mechanically complex simulations such as elastic collisions and squished together cells with amorphous forms. It would also allow interactions between cells like adhesion.

User experience could be improved by a more responsive user interface and providing the opportunity to save load and export parameters and states of a simulation.

IRODALOMJEGYZÉK

[1] Chisholm, R. H., Lorenzi, T., Lorz, A., Larsen, A. K., de Almeida, L. N., Escargueil, A., Clairambault, J.: Emergence of Drug Tolerance in Cancer Cell Populations: An Evolutionary Outcome of Selection, Nongenetic Instability, and Stress-Induced Adaptation, *Cancer Res*, Vol. 75, Issue 6, 2015, pp. 930-939

[2] Cytowski, M., Szymanska, Z.: Large-Scale Parallel Simulations of 3D Cell Colony Dynamics, *Computing in Science & Engineering*, Vol. 16, Issue 5, 2014, pp. 86-95.

[3] Grimes, R., Kelly, C., Bloch, K., Partridge, M.: A method for estimating the oxygen consumption rate in multicellular tumour spheroids, *R. Soc. Interface.* 11, 2014

[4] Jeon, J., Quaranta, V., Cummings, P. T.: An Off-Lattice Hybrid Discrete-Continuum Model of Tumor Growth and Invasion, *Biophysical Journal*, Vol. 98, 2010, pp. 37-47.

[5] Jonkman, JE., Cathcart, JA., Xu, F., et al. An introduction to the wound healing assay using live-cell microscopy. *Cell Adh Migr.* 8(5), 2014

[6] Kerr, CC., Stuart, RM., Mistry, D., Abeysuriya, RG., Rosenfeld, K., et al. Covasim: An agent-based model of COVID-19 dynamics and interventions. PLOS Computational Biology 17(7), 2021

[7] Kleinrock L.: Queueing Systems, Volume I, *Wiley-Interscience*, 1975

[8] Macklin, P., Edgerton, M. E., Thompson, A. M., Cristini, V.: Patient-calibrated agent-based modelling of ductal carcinoma in situ (DCIS): from microscopic measurements to macroscopic predictions of clinical progression, *Journal of Theoretical Biology*, Vol. 301, 2012, pp. 122-140.

[9] Motta, S. S., Cluzel, P., Aldana, M.: Adaptive Resistance in Bacteria Requires Epigenetic Inheritance, Genetic Noise, and Cost of Efflux Pumps, *PLoS ONE*, Vol. 10(3), 2015

[10] Waclaw, B., Bozic, I., Pittman, M. E., Hruban, R. H., Vogelstein, B., Nowak, M. A.: A spatial model predicts that dispersal and cell turnover limit intratumour heterogeneity, *Nature*, Vol. 525, 2015, pp. 261-264

[11] Walker, D. C., Southgate, J., Hill, G., Holcombe, M., Hose, D. R., Wood, S.M. Mac Neil, S., Smallwood, R. H.: The epitheliome: agent-based modelling of the social behaviour of cells, *BioSystems*, Vol. 76, 2004, pp 89-100.