

Debreceni Egyetem

Informatikai Kar

**Információterjedés szimulációja térben
dinamikus hálózatokon**

TDK dolgozat

Témavezető:

Kocsis Gergely
egyetemi adjunktus

Készítette:

Németh András
mérnök informatikus Msc
hallgató

Debrecen

2014

Tartalomjegyzék

1. Motiváció	2
2. Sejtautomaták és járványterjedési modellek	4
2.1. A sejtautomaták története és felépítése	4
2.2. Kétdimenziós sejtautomata	6
2.2.1. Soros és párhuzamos frissítés	8
2.3. Brown mozgás	9
2.4. Levy-eloszlású mozgás	9
2.5. Járványterjedés	10
2.5.1. SI modell	10
2.5.2. SIR modell	11
2.5.3. SIS modell	12
2.5.4. SIRS modell	12
3. A modell leírása és implementációja	14
3.1. Az általános modell leírása	14
3.2. Megvalósítás	15
4. Eredmények	20
4.1. Eredmények az SI modell esetén	20
4.2. Eredmények az SIS modell esetén	20
5. Összefoglalás	25
Köszönetnyilvánítás	30

1. fejezet

Motiváció

A valós világban lezajló terjedési folyamatok számítógépes vizsgálta hosszú évekre nyúlik vissza és a rogersi alapok óta szinte külön tudományként tartjuk számon [1]. TDK munkám során a terjedési folyamatok egy speciális változatával, az információterjedéssel foglalkozom térben dinamikus hálózatokon. Vizsgálataim eszközüül a jól ismert járványterjedési modellek módosított változatát választottam.

Dolgozatomban először ezeket a járványterjedési modelleket tekintem át, megvizsgálom sajátosságaikat és tulajdonságaikat. Korábbi kutatások azt bizonyították, hogy ezen modellek tökéletesen használhatóak más terjedési folyamatok vizsgálatára is [2]. Fő célkitűzésem ezen módszertanok alapján egy információterjedési rendszer szimulálása és kiértékelése.

A vezeték nélküli szenzoros technológiák rohamos fejlődésével, egyre inkább előtérbe kerül a hálózatok energiatakarékos fenntartása. Az általam alapul vett rendszerben több, folyamatosan mozgó egyed számára kell, friss információt biztosítani. Mivel viszonylag nagy számú és - a mobilitás miatt - kis méretű hordozható eszközről beszélünk, ezért egy állandó, nagy hatótávú kapcsolat nem lenne gazdaságos. Szükség van egy másik megoldásra. A felvetésem szerint, megfelelő technika használatával alacsony költségekkel és energiahasználattal megoldható ez a probléma. Az egyedek kis hatótávolságú jeladóval és vevővel rendelkeznek, melyek bizonyos időközönként egyszerre bekapcsolnak. Ezután az adott egyeden rögzített eszköz megvizsgálja, hogy van-e a közelében másik vevőegység,

amennyiben igen átküldi az általa birtokolt információkat.

Munkám során a fent leírt rendszert modellezem és elemeztem szimuláció segítségével. Kezdetben egy egyszerűbb modell segítségével próbálom meghatározni a rendszer alapvető viselkedését és tulajdonságait. Ehhez a modellhez egy olyan mozgási rendszert használok, mely sok tulajdonságában a Brown-mozgáshoz hasonlít. A szimulációt több beállításon is lefuttatom, az eredményeket összevetem, grafikonon ábrázolom és kiértékelem és ezek alapján próbálom megtalálni a legnagyobb hatékonysággal működő optimális rendszert.

Ezután hasonló vizsgálatokat végzek egy összetettebb, a Levy-eloszláson alapuló mozgási modellel rendelkező rendszer esetében, mely a kutatások szerint remekül leírja az egyes embercsoportok és az vadon élő állatok mozgását. Összevetem az így kapott eredményeket a korábbiakkal és ez alapján próbálom meghatározni, mennyiben változtat ez a rendszer viselkedésén. Munkám eredményeként sikerült megalkotni a fentiekben bemutatott modellt majd szimulációs programot fejleszteni mellyel hatékonyan vizsgálható. Az elkészült program segítségével sikerült néhány érdekes megállapítást tenni a modell által reprezentált rendszerekkel kapcsolatban.

2. fejezet

Sejtautomaták és járványterjedési modellek

2.1. A sejtautomaták története és felépítése

A sejtautomata, melynek kidolgozása Neumman János és Stanislaw Ulam nevéhez köthető, olyan matematikai módszer, mely alkalmas lehet biológiai rendszerek modellezésére [3]. A rendszer történetének egyik meghatározó lépése a Cambridge Egyetem professzorához, John Conway-hez fűződik. 1971-ben a Scientific American tudományos folyóiratban megjelent publikációjában egy különleges sejtautomata az úgynevezett *életjáték* modelljét írta le. Ez egy diszkrét, dinamikus rendszert vázolt fel, mely képes az önreprodukcióra [4]. Az elnevezés találó, hisz a rendszer egy, a felhasználó által meghatározott világban élő és működő, valamint a világban történő változásokra folyamatosan reagáló sejtek mozgását hivatott reprezentálni. A modell az úgynevezett "null személyes" játék kategóriába tartozik, vagyis a felhasználó, mindössze a játék elején adhat meg paramétereket, a program futása során mindent a beállítási értékek és a szabálygyűjtemény dönt el, vagyis itt a rendszer már nem befolyásolható. A felsorolt példák jól mutatják, hogy milyen sok területen alkalmazhatóak a sejtautomaták, beleértve az élővilágban lezajló eseményeket.

A sejtautomata tulajdonképpen sejtek, más néven egyedek, egy adott topológián elhelyezkedő csoportja. Ezek az egyedek véges számú állapotot vehetnek fel, melyet diszkrét időpillanatokban a saját és a szomszédos egyedek állapota alapján változtathatnak meg [5]. Ahhoz, hogy pontosan megértsük mit is értünk ez alatt, szükségünk van a rendszer összetevőinek ismeretére. A sejtautomata rendszer fő összetevői a következők:

- **Egyed**

A világban jelen levő szereplők megtestesítése. Minden sejtnak van valamiféle tulajdonsága, melyet a modellezésnél, a lehetséges tulajdonságok számával megegyező K féle állapottal jelölünk. [6]

- **Környezet**

Az egyedek, vagy más néven sejtek olyan csoportja, melyek meghatározott kapcsolatban vannak egymással. Egy ilyen sejt-együttes számos dimenzióban létezhet.

Egydimenziós sejtautomata esetén a sejtek egymás mellett vonalban helyezkednek el, így az egymás mellett lévő egyedek egymással kapcsolatban vannak.

Két dimenzió használata esetén egy sík területén elhelyezkedő sejtekről beszélhetünk, melyek egy négyzetrácson terülnek el.

Természetesen további dimenziók használatára is van lehetőség, ennek csak a rendelkezésre álló eszközök szabnak határt.

- **A szomszédsági viszony megállapítása**

A sejtautomata létrehozásánál elengedhetetlen megállapítani mely az egyes egyedek, milyen kapcsolatban állnak egymással, vagyis, hogy mely egyedek számítanak szomszédosnak. Más szavakkal, szükségünk van annak ismeretére, hogy a világ mely szereplői képesek interakcióba lépni egymással. Ez több tényezőtől is függ kezdve a dimenziók számától, egészen a szélső egyedeknél választott megoldásig. Egy ilyen kapcsolat lehet egy-, valamint kétirányú.

- **Szabályrendszer**

Miután elhelyeztük a világban az egyedeket, szükségünk van valamiféle rendszer felállítására, mely alapján eldönthetjük, hogy az adott egyed mi alapján változtatja meg saját állapotát. Ehhez bevezetünk egy olyan szabályrendszert, mely pontosan leírja, azoknak a megszorításoknak az összességét, melyeket minden egyednek be kell tartania a folyamat során.

A felület állapotváltozása tulajdonképpen két rétegre tagolható. Először is meg kell vizsgálnunk, mi történik sejt szinten(mikroszkopikus), azaz milyen változás megy végbe az érintett sejt eredeti állapota és a szomszédos sejtek állapotának függvényében. A másik az egész rendszer állapotváltozásai(makroszkopikus), melyet az előbb említett sejt szintű állapotváltozások összessége idéz elő.

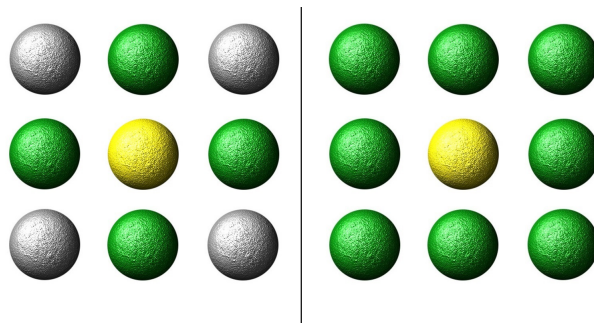
2.2. Kétdimenziós sejtautomata

Mivel dolgozatomban során kétdimenziós sejtautomatákkal fogok foglalkozni, ezért ezek működésére most részletesebben is kitérek.

A kétdimenziós sejtautomatát egy négyzetrács topológián elhelyezkedő egyedek összességének lehet tekinteni. A számítógépes implementálást tekintve ez optimális megoldás, hisz a négyzetrácsot megfejtethetjük egy két dimenziós tömbnek, melyben minden, értékkel rendelkező elem egy adott egyedet szimbolizál. Azokat a mezőket, melyeken nem tartózkodik egyed, valamilyen NULL értékkel jelölhetem.

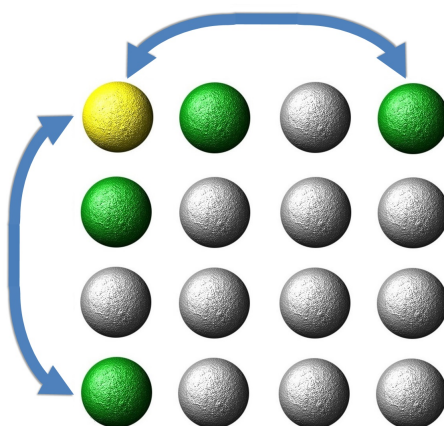
A fentebb említett szomszédsági viszonyok tekintetében több megoldás közül is választhatunk. A leggyakrabban használtak az úgynevezett *Neumann szomszédság*, mely négy és a *Moore szomszédság*, mely nyolc szomszédal dolgozik. Az első esetben az érintett egyed szomszédja az alatta, felette és két oldalt elhelyezkedő társa, míg utóbbinál az átlósan elhelyezkedőek is ide tartoznak, ahogy a 2.1 ábra is mutatja.

Minden négyzetrács topológia esetén felmerülő probléma a rács peremén fellépő szomszédsági viszony megállapítása. Ugyanis a rács szélein elhelyezkedő egyedek, a különböző



2.1. ábra. A sárga sejtet vizsgálom. A bal oldali ábrán a zöld sejtek jelentik a szomszédjait, Neumann szomszédság szerint. A jobb oldalon ugyanez a folyamat látható Moore szomszédság esetén.

szomszédszám miatt befolyásolhatják a szimuláció eredményét. Ezt a jelenséget nevezük peremhatásnak. Ezt legtöbb esetben az úgynevezett periodikus határfeltétellel oldják meg, mely a peremeken levő egyedeket egymás szomszédjainak felelteti meg. (lásd 2.2 ábra)



2.2. ábra. A sárga sejtet vizsgálom, Neumann-szomszédság szerint. A sejt szomszédjai zöld színnel vannak jelölve és látható, hogy két szomszéd a periodikus határfeltétel alapján a négyzetrács túlsó oldalán helyezkedik el.

Neumann szomszédság esetén ezt az alábbi módon lehet szemléltetni:

$$\begin{array}{cccccc}
 a_{11} & a_{12} & . & . & . & a_{1n} \\
 a_{21} & a_{22} & . & . & . & a_{2n} \\
 . & . & . & & & . \\
 . & . & & . & & . \\
 . & . & & & . & . \\
 a_{m1} & a_{m2} & . & . & . & a_{mn}
 \end{array}$$

ami egy $n * m$ méretű négyzetrácsot jelöl, ahol az n és az m pedig a két dimenzió mérete. Ebben az esetben az a_{11} egyednek szomszédjai, az a_{12} , a_{21} , a_{m1} és az a_{1n} egyedek.

2.2.1. Soros és párhuzamos frissítés

Egy sejtautomata szimuláció esetében fontos tényező a frissítés módjának megadása, azaz annak eldöntése, hogy a sejtek egy azonos időpillanatban módosítják-e állapotukat vagy sem [7]. A *párhuzamos frissítés (szinkron frissítés)* esetén a sejtek egy azon időpillanatban teszik ezt meg, azaz a t időpillanatban fennálló állapotot a $t - 1$ időpillanatban felálló értékek befolyásolják. *Soros frissítés (aszinkron frissítés)* esetén a sejtek állapota egymás után sorban, nem egy időben változik meg, így figyelembe veszik az eddig már lépett egyedek állapotát is. Ilyen esetben fontos megadnunk a sejtek lépésének sorrendjét, amely lehet teljesen véletlenszerű vagy szabályos.

Utóbbi esetben el kell döntenünk, hogy mit tekintjük egy időpillanatnak. Az egyik megoldás a lépésalapú, mely során egy egyed állapotváltozása számít egy lépésnek és az iteráció ilyen lépések gyűjteménye. Ebben az esetben az idő nincs egyértelműen definiálva. A másik megoldás az időalapú, amikor minden egyed számára megjelölünk egy időpontot, amikor az állapotváltozást véghezviheti. Attól függően, hogy melyik módszert választjuk, drasztikusan eltérő eredményeket kaphatunk, mely bizonyítja, hogy az egyedek lépésének sorrendje komoly hatással van a szimuláció eredményére.

2.3. Brown mozgás

1827-ben Robert Brown angol botanikus, vízben mozgó virágporszemcsék viselkedését vizsgálta. Az eközben tett megfigyelései alapján megállapította, hogy a gázokban és folyadékokban lebegő részecskék véletlenszerű mozgást végeznek. Ezenfelül maga a kifejezés egy matematikai modellt is jelöl, melyet a véletlen mozgás leírására használnak [12].

Albert Einstein 1905-ben egy - ezt a problémát vizsgáló - publikációt tett közvé, melyben magyarázatot adott erre a jelenségre. Eszerint a jelenség azért következik be, mert az oldószer molekulái az állandó hőmozgás miatt, a vizsgált részecskéket véletlenszerűen lökdösik [13]. A káosz elmélet is tartalmaz egy ehhez tartozó aspektust, melynek elnevezése *globális keveredés*. E szerint egy tipikus kezdőfeltételekkel induló rendszer, ha elég lépésszámot, adunk meg számára, minden lehetséges állapothoz közel kerül. A felfedezés hatalmas lökést adott a fizika tudományágának, hisz a Brown mozgás kitűnően alkalmazható egy-egy atomcsoport mozgásának leírására is.

Természetesen számunkra nem a fizikában elért eredmények miatt fontos ez a modell, hanem mert számos pontot megegyezik az általunk leírttal. A részecskék véletlenszerűen mozognak, ahogy azt a mi ágenseink is teszik. Fontos kiemelni azonban, hogy az általam használt modellben ezek az egyedek nem pattannak vissza egymásról.

2.4. Levy-eloszlású mozgás

Vizsgálatunk másik alapját egy olyan mozgási modell képezi, melyben az egyed, egy időpillanatban nagyobb, legfeljebb `dist_max` távolságot is megtehet. A távolság Levy-eloszlás szerint generálódik, mely egy folytonos valószínűség eloszlás [14]. Kidolgozása Paul Pierre Lévy francia matematikus nevéhez köthető. Az eloszlásnak megfelelően, a táv generálása során nagyobb valószínűséggel kapunk az alsó értékhez közeli számot, míg a maximum értéket, vagyis a `dist_max`-ot nagyon csekély valószínűséggel kaphatjuk eredményül.

2.5. Járványterjedés

A szociológiai kapcsolati hálókkal foglalkozó kutatások bebizonyították, hogy ezek számos közös pontot mutatnak a járványterjedési folyamatokkal. A betegség akár csak egy ötlet, vagy egy információ, egy adott, egymással kapcsolatban levő egyedek között terjed. Az ötletek és információk átadása során sok minden múlik az adott személy döntésein, szándékain, hogy eredményesen képes-e átadni az adott információt másoknak. Betegségek esetében a "döntési" folyamat teljesen különbözik ettől, hisz a terjedésnél rengeteg tényezőt szükséges figyelembe venni. Még maga a fertőzés módja is betegség függő. Van amelyik levegőben terjed, van amely érintés útján és van olyan is, amely csak közvetlenül a véráramba kerülve okoz problémát. Miután a kórokozók bekerültek a szervezetbe, elkezdenek sokszorozódni és egyre inkább kifejtik hatásukat. Az immunrendszer megpróbálja felvenni a vele harcot, eközben tünetek mutatkozhatnak. Általában végül vagy az egyik, vagy a másik győzedelmeskedik, mely gyógyulással, halállal, vagy hosszabb távú szimptómákkal jár. Elméletben ha pontosan meg akarjuk ismerni a betegségek terjedését, minden biológiai tényezőt figyelembe kell vennünk, azonban a valóságban ritkán történik efféle vizsgálat, annak összetettsége miatt. Azonban vannak sokkal követhetőbb és jobban vizsgálhatóbb módszerek is.

2.5.1. SI modell

Az SI modell a járványterjedési modellek legegyszerűbb fajtája, mely figyelmen kívül hagyja egy valódi betegség terjedésének számos aspektusát, de mégis megadja az alapot a további vizsgálatokhoz. Ebben a modellben egy sejt, más néven egy egyed mindössze két lehetséges állapotot vehet fel: fertőzött és egészséges. Az első állapotba, azok az egyedek tartoznak, amelyek egészségesek, azonban bármikor elkaphatják a fertőzést. Ezt a *susceptible*, vagyis a fogékony után, S -sel jelöljük. A másik állapotba azon egyedek kerülnek, amelyek már elkapták a betegséget. Őket az *infected*, vagyis fertőzött után I -vel jelöljük. Egy egészséges egyed abban az esetben betegedhet meg, ha valamilyen módon

kapcsolatba kerül egy fertőzött egyeddel. A modell szerint, ha valaki elkapja a betegséget, az a személy sohasem épül fel belőle [8].

A hagyományos megközelítés kimondja, hogy a szóban forgó egyedek között olyan kapcsolatrendszer áll fent, melyben egy adott egyed β valószínűséggel léphet kapcsolatba bármelyik másik egyeddel, egy adott időpillanatban. Természetesen ez nem realiztikus, hiszen a valóságban egy személy, mindennapjai során mindössze a népesség csak egy kis hányadával lép kapcsolatba. Jelölje $S(t)$ az egészséges, $X(t)$ pedig a fertőzött egyedek számát a t -edik időpillanatban. Továbbá tegyük fel, hogy egy időpillanatban az egyed átlagosan β véletlenszerűen választott egyeddel lép kapcsolatba. A betegség ugyebár csak abban az esetben terjed tovább, ha egy fertőzött egyed kapcsolatba kerül egy egészséggel. Ha a népesség száma n , akkor annak a valószínűsége, hogy egy véletlenszerűen választott egyed egészséges legyen S/n . Ebből következik, hogy egy fertőzött egyed átlagosan $\beta S/n$ egészséges egyeddel lép kapcsolatba, egy időpillanatban. Mivel összesen X fertőzött van, ezért az újonnan megfertőzöttek átlagos aránya $\beta SX/n$ lesz.

$$\frac{dX}{dt} = \beta \frac{SX}{n} \quad (2.1)$$

Eközben az egészséges egyedek aránya, ugyanilyen arányban csökken.

$$\frac{dS}{dt} = -\beta \frac{SX}{n} \quad (2.2)$$

2.5.2. SIR modell

A következő modell, mellyel foglalkoznunk kell az úgynevezett SIR, mely tulajdonképpen az előző modell kiterjesztése. Az SI esetében, ha egy egyed megfertőződik, a szimuláció végéig az is marad. Azonban a valóságban számos olyan betegség létezik, mely ellen az immunrendszer sikeresen fel tudja venni a harcot és a páciens meggyógyul. Sőt az is lehetséges, hogy így az adott személy, a továbbiakban immunissá válik. Ezért ebben a modellben bevezetjük az úgynevezett *recovered*- R , vagyis a meggyógyult állapotot. Az R állapotú egyedek nem fertőznek és nem is tudják újra elkapni a betegséget [9].

Természetesen olyan kimenetel is lehetséges, hogy a beteg meghal. Azonban a vizsgálat

szempontjából az, hogy a beteg meghalt, vagy meggyógyult számunkra teljesen érdektelen, hisz akár az egyik, akár a másik eset áll fent, a páciens mindenképp kiesik a megfertőzhető, vagy fertőző egyedek sorából. Ezért bizonyos esetekben az R állapotra, inkább *removed*, vagyis eltávolítottként hivatkoznak.

A modell két szakaszra bontható. Az első szakaszban az egyedek kapcsolatba lépnek egymással és megfertőződnek. Az egy időpillanatba eső kapcsolatok átlagos aránya egy egyedre tekintve továbbra is β . A második szakaszban a fertőzött egyedek meggyógyulnak, átlagosan γ aránnyal. Ennek felhasználásával meg tudjuk adni τ értékét, vagyis azt, hogy egy egyed átlagosan mennyi időt tölt fertőzött állapotban, mielőtt meggyógyulna. Bármelyik időpillanatra nézve, annak valószínűsége, hogy egy fertőzött egyed meggyógyul $\delta\tau\gamma$, míg annak, hogy nem így tesz $1 - \delta\tau\gamma$. Így annak a valószínűsége, hogy egy egyed τ idő eltelte után is fertőzött legyen:

$$\lim t(1 - \gamma\delta\theta)^{t/\beta\gamma} = e^{-\gamma t} \quad (2.3)$$

2.5.3. SIS modell

Az előző modell már figyelembe vette, a betegségből való kigyógyulás és az immunitás kialakulásának lehetőségét. Azonban olyan betegségek is vannak - mint például a megfázás - melynél a gyógyulást követően nem alakul ki immunitás, tehát a szervezet bármikor újra elkaphatja a kórt [10]. Ilyenkor ugyanúgy járunk el, mint az előző modell esetében, azzal a kivétellel, hogy az I állapotot, ismét S követi.

$$\frac{dx}{dt} = \beta sx - \gamma x \quad (2.4)$$

$$\frac{ds}{dt} = \gamma x - \beta sx \quad (2.5)$$

2.5.4. SIRS modell

Utolsóként az SIRS modellel foglalkozunk. Itt az egyedek gyógyulásuk után immunitást kapnak, akárcsak az SIR modellben, azonban itt az immunitás csak időleges, így egy adott

idő után újra S állapotba kerül [11]. Az immunitás elvesztésének átlagos arányát δ -val jelölöm.

$$\frac{dx}{dt} = \beta sx - \gamma x \quad (2.6)$$

$$\frac{ds}{dt} = \delta r - \beta sx \quad (2.7)$$

$$\frac{dr}{dt} = \gamma x - \delta r \quad (2.8)$$

A beállítási értékektől függően, ez a modell számos viselkedést tükrözhet. Lehet olyan variáció, ahol a betegség teljesen szétterjed, olyan ahol kihal és olyan is, ahol a világ egymást követő járványkitörésekkel jár.

3. fejezet

A modell leírása és implementációja

A TDK munka során a korábban már leírtakhoz hasonló információterjedéssel kapcsolatos jelenségeket vizsgálunk dinamikus hálózatokon. Több a felhozott példákhoz hasonló esetet találhatunk a való életben, a munka jelenlegi pontján azonban elsősorban arra koncentrálunk, hogy egy átfogó képet kapjunk a terjedési modellek viselkedéséről dinamikus hálózatokon. Bár vizsgálataim során nem foglalkoztam az SIR, valamint az SIRS modellekkel, de ezek későbbi implementálásának megkönnyítése érdekében, már az R állapotot is bevezettem a rendszerembe.

A modell bár nem teljesen egyezik meg, mégis sok hasonlóságot mutat az úgynevezett Brown mozgással, ezért az alábbiakban ezt is megvizsgálom.

3.1. Az általános modell leírása

A vizsgálatban szereplő modell egy $N \times N$ oldalhosszakkal rendelkező négyzetrácsot ír le. A kezdeti állapotban ezen a négyzetrácson véletlenszerűen elhelyezünk `agent_num` számú egyedet. Az egyedek az éppen használt modellnek megfelelően az alábbi állapotokat vehetik fel:

1. *S* - nem informált
2. *I* - informált
3. *R* - korábban informált

Az egyedek mozgásuk során a Brown-mozgás szabályait követve, véletlenszerűen mozdulhatnak el jobbra, balra, fel vagy le. Egy pozícióban több egyed is tartózkodhat. A periodikus határfeltételt használva, amennyiben egy egyed kilép a négyzetrács területéről - azaz valamely koordinátája nulla alá vagy $N - 1$ felé kerül -, abban az esetben ezeket a koordinátákat, a periodikus határfeltételnek megfelelően $N - 1$ -re vagy nullára módosítom. A korábban szemügyre vett rendszerek esetén a világ egyes mezői, mind egy-egy egyedet jelöltek. Az én általam felvázolt rendszer ettől eltér, itt a mező a világ egy pontját jelöli, mely lehet üres, valamint tartalmazhat egy, vagy akár több egyedet is.

3.2. Megvalósítás

A modell számítógépes implementálásához a C++ nyelvet használtam, mely képességiben tökéletesen megfelel számomra. Ennek ellenére tulajdonképpen egy C++ eszközt is felhasználó C programról van szó, ugyanis munkám során az utóbbira jellemző eljárás-orientáltságot részesítettem előnyben.

A program indításakor az alábbi paramétereket várja:

1. `N` - négyzetrács oldalhossza
2. `agent_num` - egyedek száma (százalékban megadva)
3. `begin_inf` - indulásnál a fertőzött száma (százalékban megadva)
4. `step` - iterációk (lépések) száma
5. `global_inf_time` - fertőzési idő
6. `global_imm_time` - immunitási idő
7. `dist_max` - maximális ugrási távolság (Levy-mozgás esetén)
8. `scale_file_name` - az egyes állapotok arányait tartalmazó fájl

A program egyik legmeghatározóbb eleme az **AGENT** struktúra, mely az egyedek számítógépes implementációja. Az **agent** hét paraméterrel rendelkezik: **ID** - az egyed azonosítója, **y,x** - az egyed koordinátái a tengelyek mentén, **agent_state** - az egyed jelenlegi állapota, **agent_prev_state** - az egyed előző időpillanatbeli állapota, **inf_time** - azt jelzi, hogy az egyed még hány iteráció elejéig marad fertőzött, **imm_time** - azt jelzi, hogy az egyed még hány iteráció elejéig marad immunis.

```
typedef struct agent{
    int ID;
    int y,x;
    char agent_state;
    char agent_t_state;
    int inf_time;
    int imm_time;
}AGENT;
```

A másik fontos struktúra a **FIELD**, mely a modellünk alapjául szolgáló világ, vagyis a négyzetrács egyes mezőinek megtestesítője. Két paraméterrel rendelkezik: **db** - azt mutatja, hány egyed tartózkodik az adott mezőben, **agent_list** - egy **integer** típusú lista, mely a mezőben tartózkodó egyedek azonosítóit tartalmazza.

```
typedef struct field{
    int db;
    int* agent_list;
}FIELD;
```

Mivel az egyes mezőkben tartózkodó egyedek száma a különböző vizsgálatok esetében nagymértékben eltérhet, ezért ez a struktúra dinamikusan tárolja az egyedek azonosítóit. Amennyiben új egyed érkezik, növeli a mezőben tárolható **int** értékek lehetséges számát eggyel, ellenkező esetben pedig csökkenti azt.

A beolvasott értéket alapján létrehozok egy **N*N** méretű **field[]** nevű két dimenziós **FIELD** típusú tömböt, mely a világot jelentő négyzetrácsként fog viselkedni és egy **agent_num** méretű **agent[]** nevű **AGENT** típusú tömböt, mely az egyedek listáját tartalmazza. Ezek mellett még létrehozok egy **inf_agent[]** **integer** típusú tömböt, mely a fertőzött egyedek azonosítóit tartalmazza.

Ezután szükségem lesz egyedekre, melyeket elhelyezhetek a "világban", vagyis a `field`-ben. Ezért a műveletért a `build_up_system()` algoritmus felel. Minden egyed számára legenerál két véletlen számot 0 és N között és ezeket adja meg annak kiindulási pozíciójaként. Ezenfelül megadja a soron következő azonosítószámot és az állapotot S -re, a fertőzöttségi számlálót 0-ra állítja. A `field[]` tömb, az előzőleg generált számoknak megfelelő sorában és oszlopában levő mező méretét megnöveli eggyel, ezt a `db` változóban is frissíti és az adott egyed azonosítóját, hozzá adja az `agent_list`-hez. Már rendelkezem egyedekkel és el is helyeztem őket, azonban az induláshoz még szükségem van fertőzött egyedekre, melyet a `gen_inf()` algoritmus biztosít számomra. Legenerál egy véletlen számot 0 és `agent_num` között, majd ellenőrzi, hogy az annak megfelelő azonosítóval rendelkező egyed fertőzött-e. Amennyiben igen, visszalép és újrakezdi a generálást, ha pedig nem akkor átállítja az adott egyed állapotát I -re.

Az egyed megfertőzéséhez a `add_to_inf()` elnevezésű algoritmust használom. Átadom neki a szóban forgó egyed azonosítóját, így átállítja annak állapotát fertőzöttre, a fertőzési számlálót beállítja a `global_inf_time`-nak megfelelően, valamint az `inf_agent[]` tömbhöz hozzáadja azt.

Ezzel be is fejeztem a rendszer felépítését és a kiindulási állapot létrehozását, így tovább léphetek az iterációs blokkra. Ennek két meghatározó algoritmus a `iterat()` és a `diag()`. Az előbbi felelős az egyedek helyzetének változtatásáért. Kezdetben lementi az adott egyed eredeti állapotát, majd generál egy véletlen egész számot 0 és 3 között, melyek az egyed lépésirányainak felelnek meg: 0 - bal, 1 - jobb, 2 - fel, 3 - le. Ennek megfelelően módosítom az egyed y és x értékeit. Amennyiben az így kapott koordináták kívül esnek a négyzetrács határain, abban az esetben a periodikus határfeltétel alapján áthelyezem őket. Ezután az `agent_prev_state`-be a későbbi vizsgálatok miatt, lementem az egyed jelenlegi állapotát.

Ez az első olyan pont, vagyis az egyedek helyzetének változtatása, ahol a Levy-mozgást implementáló program eltér ettől. Először Levy-eloszlás szerint generálunk egy, legfeljebb `dist_max` nagyságú számot, a `gen_dist()` segítségével. Ez fogja jelenteni az ugrás nagy-

ságát. Ezután szükségünk lesz még egy irányra, melyet egy 0 és 360 közötti, véletlen szám generálásával érek el. Ezekből az értékekből a megfelelő függvények használatával megkapom a célmezőt:

```
(int) pow((max_pow - min_pow) * drand48 ()+ min_pow, 1 / (exponent + 1));

px=dist*cos(irany);
py=dist*sin(irany);
px_=px > 0.0 ? (px+0.5) : (px-0.5);
py_=py > 0.0 ? (py+0.5) : (py-0.5);
```

Ezután foglalkozom az esetleges állapotváltoztatásokkal, a fertőzésen kívül. A program először csökkenti az egyedhez tartozó fertőzési számlálót eggyel. Amennyiben ez eléri a nullát, az gyógyulást jelent, vagyis ki kell húznom az egyedet a fertőzött egyedek listájáról. Ehhez a `sub_from_inf()` algoritmust viszi végbe az állapotváltoztatást. Ez SI modell esetén az *I*, SIR és SIRS modell esetén az *R* állapotot jelenti. Először a program végig megy a fertőzött egyedek azonosítóit tároló tömbön és törli azt onnan. Hogy a fentebb említett állapotváltást minden esetben egyszerűen tudjam ezt kezelni, ezért valójában minden esetben *R*-re állítom a vizsgált egyed állapotát, melyet a következő lépésben kezelek. Csökkentem az immunitási számlálót eggyel. Amennyiben ez eléri a nullát és az egyed állapota *R*, abban az esetben átállítom az egyed állapotát *S*-re, a `sub_from_imm()` algoritmus segítségével.

Mindezek után már csak azt kell ellenőriznem, hogy van-e új fertőzés a rendszerben, amelyet a fentebb is említett `diag()` algoritmussal vizsgálok. Ugyebár a fertőzött egyedek azonosítójának listáját már letároltam a `inf_agent[]` tömbben. Ez, azért fontos, hogy jobb teljesítményt legyek képes elérni, hisz így csak a fertőzött egyedek szomszédjait kell vizsgálnom. A program végigmegy ezen a tömbön, majd megnézi, azok *y* és *x* koordinátáit. Ez alapján megállapítja, hogy mely mezők tartalmazhatják az egyed szomszédjait. Itt van a második eltérés a Levy-mozgással dolgozó verzióhoz képest. Míg itt mező bal és jobb oldalán, valamint a felette és az alatta elhelyezkedő mezőket vizsgáltam, addig ott már az átlósan elhelyezkedő mezők is ebbe esnek bele. Ezek után amennyiben az

adott mező tartalmaz legalább egy egyedet, végigmegyek ezek listáján és amennyiben ezek állapota S , a korábban már használt `add_to_inf()` algoritmussal megfertőzőm azt. Fontos megemlíteni, hogy a program csak akkor veszi figyelembe a fertőzött egyedeket, ha azok `agent_prev_state`-ben tárolt, előző időpillanatbeli állapota is I volt, így elkerülve az egy időpillanatban végbemenő esetleges hullámszerű terjedést.

Ezzel megváltoztattam az egyed állapotát, de ezeket a változtatásokat még nem vittem fel a `field[]` tömbbe. Ehhez a `iterat_agent()`-et használom. Átadom neki az egyed azonosítóját, az eredeti és az új koordinátákat. Végigmegy a megfelelő mező egyedlistáján és ha megtalálja a megfelelő azonosítóval rendelkezőt törli azt onnan, csökkenti az egyedlistát és felveszi az új koordinátákkal jelölt mező listájába. Ezzel lezártam az iterációs blokkot is, melyet a maximális iteráció számnak megfelelően, *step*-szer hajtunk végre.

A program az eredmények kiértékelése végett több beépített, a vizsgálat szempontjából fontosabb értékek lementéséért felelős algoritmussal rendelkezik. Az egyik legalapvetőbb, a rendszer teszteléséért, valamint a kisebb rendszerek nyomon követéséért felelős `field_to_screen()`, mely a rendszer tartalmát megjeleníti a képernyőn. Bizonyos vizsgálatok elvégzéséhez szükségem van az egyedek pozíciójára és állapotára is, ehhez `field_to_file()` algoritmust használom fel. Minden lépésben végigmegy az `agent` tömbben levő egyedeken, kiolvassa azok x, y koordinátáit, valamint `state` állapotát és kimentí azt egy külön könyvtárba a lépés számának megfelelő néven. Mivel az rendszerben lévő állapotok arányainak elemzése csaknem minden vizsgálat alapját képezi, ezért ezt magába a `main()` függvénybe írtam bele. Lementi a jelenlegi időpillanat számát és az arányokat, a program elején, paraméterként megkapott fájlba.

4. fejezet

Eredmények

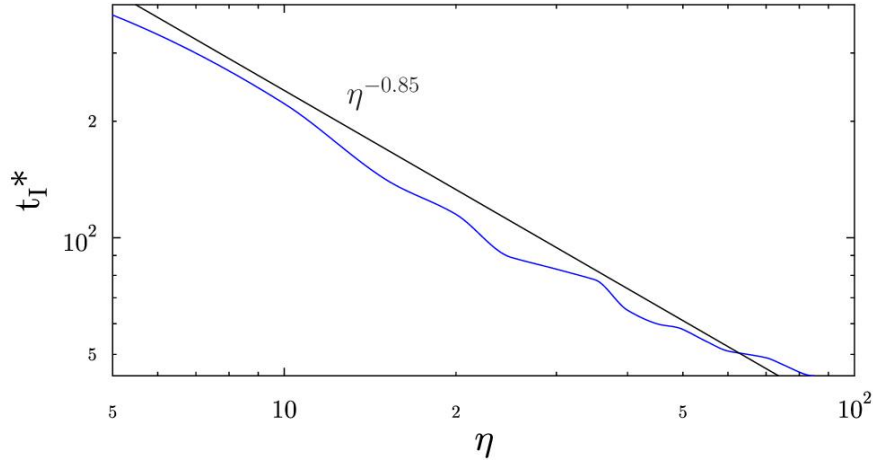
4.1. Eredmények az SI modell esetén

A rendszer jellemzőinek megismeréséhez, először a legegyszerűbb, SI modellen figyeltem meg, hogy a telítettség függvényében, mennyi idő szükséges, hogy az információ az egész rendszerben elterjedjen, azaz, hogy az egy homogén S állapotba kerüljön (4.1 ábra). A telítettség jelöli, hogy a rendszer méretéhez képest, hány százalék az egyedek száma. Vagyis $L=100$ rendszerméret és `agent_num=10%`-os telítettség esetén, 100 egyedről beszélhetünk. Az ábrán látszik, hogy minél inkább telített a rendszer η , annál korábbi időpillanatban t_I válik teljesen informálttá.

Az SI modell kiválóan alkalmas egyszerűbb rendszerek modellezésére, ugyanakkor számos olyan esettel is találkozhatunk, amikor az információt csak egy adott ideig tekinthetjük frissnek, azt az egyedeknek minél hamarabb tovább kell adnia, különben elavul. Az ilyen típusú rendszerek pontos modellezése nem egyszerű feladat, ugyanakkor, közel megfelelő eredményre juthatunk, ha az előzőnél csak egy kicsit bonyolultabb SIS modellt használjuk.

4.2. Eredmények az SIS modell esetén

Az SIS modellen végzett vizsgálataim során már rendszerek végállapotát veszem górcső alá, melyet a 200. időpillanatban határoztam meg. Megfigyeléseim szerint az általam



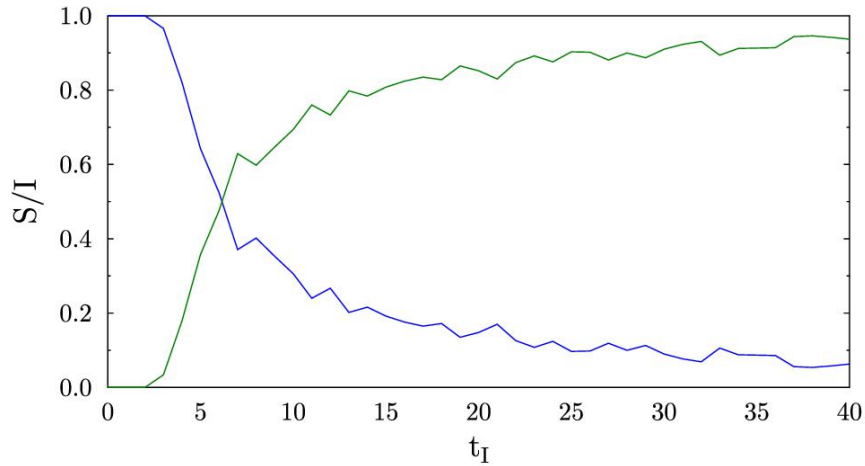
4.1. ábra. A rendszerben lévő összes egyed megfertőzéséhez szükséges időpillanatok t_I^* ábrázolása, változó telítettség η esetén. Vegyük észre, hogy a kétszer logaritmikus skálán kapott közel egyenes függvény alapján a homogén informált állapot eléréséhez szükséges idő hatványfüggvény szerint függ a telítettségtől.

végzett rendszerek esetén ennyi iterációs lépés elég ahhoz, hogy a rendszer közel állandó állapotban álljon be. Ebben a modellben az egyedek az információ megszerzése után csak egy előre definiált ideig képesek azt megtartani, utána elfelejtik (az információ elévül), azaz az egyedek visszaállnak az eredeti S állapotba.

Az elemzés során a következőkre voltam kíváncsi:

1. Egy adott kezdeti fertőzöttség esetén hogyan változik a terjedés jellege a fertőzöttségi idő változtatásával?

Ahogy azt a 4.2 ábra is mutatja alacsony fertőzési idő esetén a rendszer megmarad egészséges végállapotban, majd fokozatosan egy kevert állapotba megy át, ahol az információ nem terjed el a teljes hálózatban, de mégis jelen van. Nagyobb fertőzési idő mellett már a rendszer teljes informáltságát figyelhetjük meg. Az emelkedés egyenetlen, vannak hirtelen zuhanások, de általánosságban növekvő informáltságot mutat.



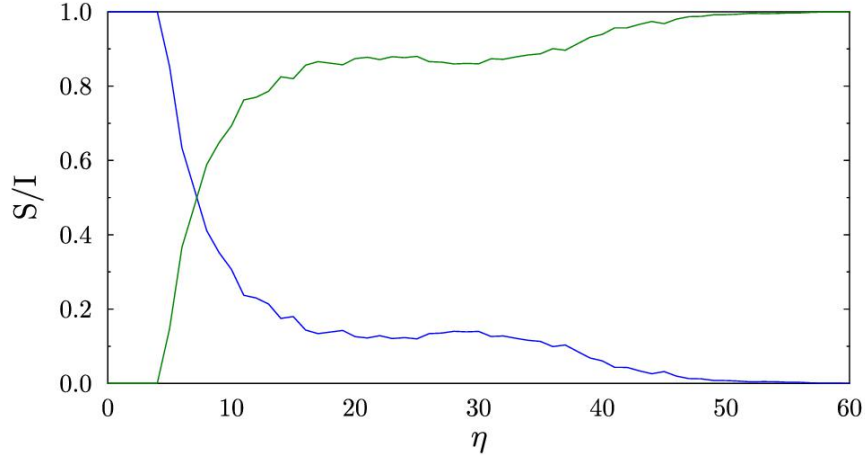
4.2. ábra. Az S és I állapotok arányainak ábrázolása, rögzített telítettség η , változó fertőzési idő t_I , valamint egyetlen kiinduló fertőzött egyed esetén. Az x tengelyen a fertőzési időt, az y tengelyen a százaléktérteket ábrázoltam. Kékkel jelöltem az S és zölddel az I értékeit.

2. Hogyan befolyásolja a terjedést a különböző egyedsűrűség (azonos fertőzöttségi idő és kezdeti fertőzöttség esetén)?

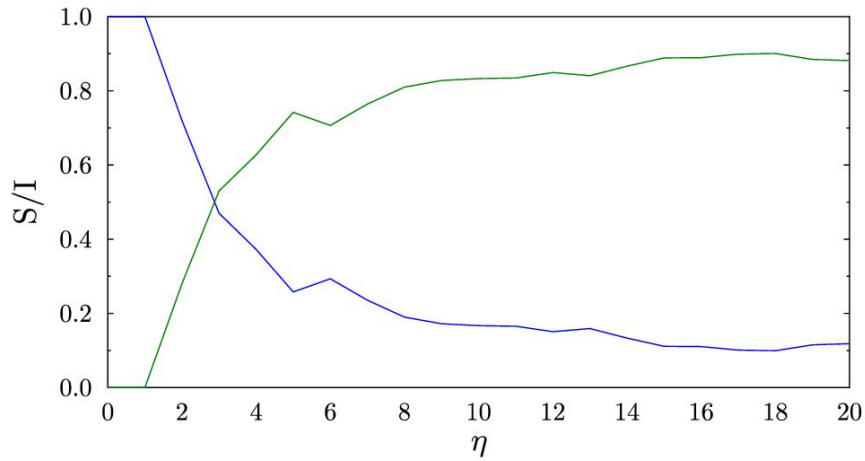
Akárcsak az előző vizsgálat esetében, az egyedsűrűség is hasonló módon befolyásolja a rendszer végállapotát. Alacsony telítettség esetén kihal a fertőzés, majd azt fokozatosan növelve, az informáltság is nagyobb méreteket ölt. Ahogy a 4.3 ábrán látszik, mindössze ezen állapotváltozások kezdete különbözik az előző rendszertől.

3. Milyen hatással van a terjedésre ,ha a természetben előforduló valós példákhoz igazodva az egyedek nem sima bolyongást végeznek, hanem Levy-mozgással változtatják a pozíciójukat.

További vizsgálatok tárgyául tervezem a két mozgástípus közötti átmenet feltérképezését, amire a Lévi mozgás modellje alapján lehetőség is van egy távolsági paraméter állításával. Ezt a paramétert 0-ra állítva pont a véletlen bolyongást kapom vissza, s a távolság növelésével lehet ebből a szélsőséges esetből indulva megfigyelni a rendszer viselkedésének változását. (4.4 ábra).

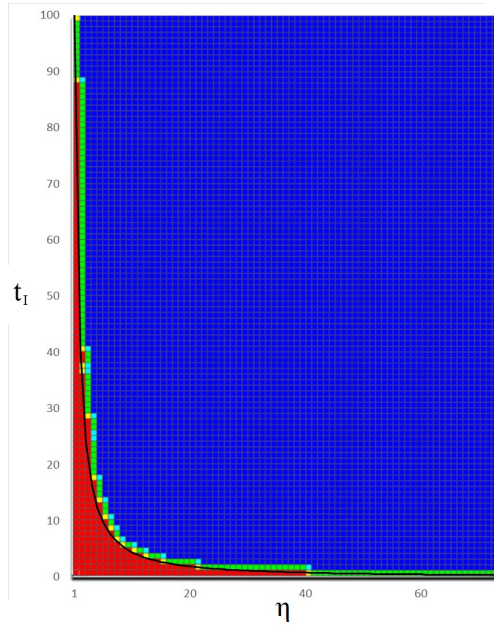


4.3. ábra. Az S és I állapotok arányainak ábrázolása, rögzített fertőzési idő, változó telítettség, és 5 százalékos kiinduló fertőzöttség esetén. Az x tengelyen a fertőzési időt, az y tengelyen a százaléktérteket ábrázoltam. Kékkel jelöltem az S és zölddel az I értékeit.



4.4. ábra. Az S és I állapotok arányainak ábrázolása *Levy-eloszlást* használó modell, rögzített fertőzési idő t_I , változó telítettség, 100-as ugrási határ és 5 százalékos kiinduló fertőzöttség esetén. Az x tengelyen a fertőzési időt, az y tengelyen a két állapot rendszerbeli arányát ábrázoltam. Kékkel jelöltem az S és zölddel az I értékeit.

A vizsgálataink egy érdekes eredménye, hogy sikerült a rendszer viselkedésének egy olyan kritikus állapotát megtalálni, ahol az egyedek hálózatán megjelenő információ nem hal ki, de el sem tud teljesen terjedni. Ahogy a 4.3 és a 4.4 ábrákon is látszik ez a szint különböző kezdeti beállítások esetén máshol található. Az eredményeket egy fázisdiagramon összegeztem (4.5 ábra). Ahogy az ábrán is látszik a paramétersíkot egy jól definiált határ mentén két részre lehet osztani. A $t_I = \eta^{-1.3} * 100$ görbétől az origó irányába haladva bármely paraméter mellett teljesen informálatlan végállapotot figyelhetünk meg. A határvonal mentén egy határozott köztes állapotba jutunk el, míg a görbétől a másik irányba elmozdulva mindig teljesen informált végállapotot kapunk. Ez az összefüggés segíthet annak megértésében, hogy a teljes informáltság megőrzése érdekében hogyan ellensúlyozhatjuk a résztvevő egyedek darabszámának csökkenését az információ aktualitásságát jelző idő növelésével, vagy fordítva.



4.5. ábra. A rendszerek végállapotainak ábrázolása, különböző telítettség és fertőzési idő értékpárok mellett. Pirossal jelöltem az informálatlan, zölddel és sárgával a részben informált, valamint a teljesen informált rendszereket. A kritikus beállításokon végigfutó görbe $t_I = \eta^{-1.3} * 100$ formában írható le.

5. fejezet

Összefoglalás

TDK dolgozatom elsődleges célja az volt, hogy megvizsgáljam, miként lehet felhasználni a járványterjedési modelleket, egy információ, adott rendszerben történő elterjedésének vizsgálatára.

Kezdetnek végigvettem a sejtautomaták fajtáit, melyből a számítógépes implementáció egyszerű lehetősége miatt a kétdimenziós sejtautomatát választottam. Megvizsgáltam ennek jellemzőit és sajátosságait. Külön figyelmet szenteltem a szomszédsági viszonyok és a peremen elhelyezkedő egyedek esetében felmerülő problémákra. Végigtekintettem az egyedek frissítése során használható lehetőségeket is. Ezekkel a jelenségekkel, a későbbiekben a gyakorlatban is szembesültem.

Az elméleti háttér megismerése után elkezdtem foglalkozni a gyakorlati megvalósítással. Elsőként egy egyszerűbb mozgási rendszert, az úgynevezett bolyongást használó modellt dolgoztam ki és implementáltam azt a C++ programozási nyelven. A modell véletlenszerűen bolyongó egyedek közötti információcserét vizsgálta. Az egyedeket egy külön erre a célra létrehozott **agent** struktúrával hoztam létre, melyben azok tulajdonságait is tároltam, külön kiemelve az egyed állapotát. A világot egy ilyen **agent**-ekből álló kétdimenziós tömb a **field** szimbolizálta. Folyamatosan nyilvántartottam a fertőzött egyedek azonosítójának listáját, hogy minél jobb teljesítményt érjek el. A szomszédsági esetek megvalósításánál a periodikus határfeltételt használtam. Minden lépésben figyeltem az különböző állapotok arányát és elmentettem azokat, külön erre a célra létrehozott

fájlokba.

Ezután egy, a valóvilágban gyakrabban megfigyelhető megoldást használtam. Ennél a rendszernél implementáltam az úgynevezett Levy-eloszlást követő mozgási modellt, mely bizonyítottan modellezi, akár egyes embercsoportok, akár a vadonélő állatok változó mozgását. A számítógépes megvalósítás során sikerült egy letisztult felépítésű programkódot létrehoznom, a korábbi verzió minimális változtatásával.

Az módosított Brown-mozgással rendelkező verzióban megvizsgáltam, hogy mennyiben befolyásolja a rendszer teljes megfertőzéséhez szükséges időt, a telítettség SI modell esetén, egyetlen egy informált egyedből kiindulva. Az eredményből kiderült, hogy a telítettség növelésével, az egyedek közötti kapcsolatok lehetséges számának megnövekedése miatt, hamarabb terjed el az információ a rendszerben. Ezt követően megvizsgáltam, hogy az SIS modell esetén, rögzített telítettség és az előzőhöz hasonlóan, egyetlen kiinduló informált egyeddel, mennyibe változik a végállapot, a fertőzési idő változtatásával. Ebből kiderült, hogy rendkívül alacsony fertőzési idő esetén, az egyedeknek nincs elég idejük átadni az információt, így a fertőzés még a szimuláció első időpillanataiban kihal. Azonban, ahogy fokozatosan növeltem a fertőzési időt, úgy kezdted növekedni a rendszer informáltsága is, mígnem, egy bizonyos határértéket meghaladva teljesen vagy közel informálttá nem vált.

Ezután mindkét verziónál megvizsgáltam, hogy az újrafertőzést is megengedő SIS modellenél, hogyan változik a fertőzöttség, ismét csak a kezdeti telítettség arányában. A tesztek kimutatták, hogy hasonló folyamat megy végbe mindkét esetben. Megfigyeltem, hogy a kezdeti fertőzöttség növelésével, az informált egyedek száma – kisebb statisztikus ingadozás mellett – folyamatos növekedést mutat. Ezután ugyanezt a modellt vettem alapul, ezúttal rögzített fertőzési idő és változó telítettség esetén. Vizsgálataim során itt is hasonló eredményre jutottam. Egy bizonyos értékig az információ rövid időn belül kihal a rendszerből, azonban egy bizonyos küszöbértéket meghaladva, egyre nagyobb lesz az informáltság mértéke. Ezt követően a Levy-eloszlást alapul vevő mozgási modell esetén megismételtem a mérést, melynél hasonló viselkedés figyelhető meg, mindössze máshol he-

lyezkednek el a kritikus pontok. Munkám részeként egy fázisdiagramon ábrázoltam, hogy a telítettség és a fertőzési idő függvényében milyen végállapotra számíthatunk a rendszer futtatásakor.

A munka folytatásaként tervezem egyrészt a Levy-mozgás további tanulmányozását, hogy folytonos képet kaphassunk arról, hogyan következik be a változás a két mozgástípus között. A programom rugalmasságából adódóan kis energia befektetéssel lehetőség nyílna az SIR és az SIRS modellek vizsgálatára is, amelyekből további a valós világra vonatkoztatható következtetéseket vonhatunk le.

Irodalomjegyzék

- [1] E.M. Rogers - *Diffusion of Innovations 5th edition*, Free Press Simon and Schuster, New York, 2003
- [2] M.E.J. Newman - *Networks - An introduction*, Oxford University Press, Oxford, 2010
- [3] von Neumann J. - *The general and logical theory of automata* L.A. Jefress, 1948
- [4] Conway J. H. - *Game of Life Scientific American 223 (October 1970): 120-123.*, 1970
- [5] Espericueta, R. - *Cellular automata dynamics* Bakerfield College, 1997
- [6] Macal C.M. and North M.J. - *Tutorial on agent-based modelling and simulation Journal of Simulation*, 2010
- [7] B. Schönfisch, A. de Roos - *Synchronous and asynchronous updating in cellular automata*, 1999
- [8] David Easley and Jon Kleinberg - *Networks, Crowds, and Markets: Reasoning about a Highly Connected World (pp 645-647)*, 2010
- [9] David Easley and Jon Kleinberg - *Networks, Crowds, and Markets: Reasoning about a Highly Connected World (pp 650-656)*, 2010
- [10] David Easley and Jon Kleinberg - *Networks, Crowds, and Markets: Reasoning about a Highly Connected World (pp 656-658)*, 2010

- [11] David Easley and Jon Kleinberg - *Networks, Crowds, and Markets: Reasoning about a Highly Connected World* (pp 659-660), 2010
- [12] Brownian motion http://en.wikipedia.org/wiki/Brownian_motion
Utoljára látogatva: 2014.11.16.
- [13] Einstein A. - *Investigations on the theory of the Brownian movement*, 1956 http://www.pitt.edu/~jdnorton/lectures/Rotman_Summer_School_2013/Einstein_1905_docs/Einstein_Brownian_English.pdf
Utoljára látogatva: 2014.11.16.
- [14] Lévy distribution http://en.wikipedia.org/wiki/L%C3%A9vy_distribution
Utoljára látogatva: 2014.11.16.

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani a témavezetőmnek, **Kocsis Gergely** egyetemi adjunktusnak, aki folyamatosan segített munkám során.