XX. reál- és humántudományi Erdélyi Tudományos Diákköri Konferencia (ETDK) Kolozsvár, 2017. május 18-21.

Optimalizáció és modularitás evolúciós hálókban

Szerző:

Lénárt Levente

Babeş-Bolyai Tudományegyetem, Kolozsvár, Fizika Kar, fizika-informatika szak, alapképzés, III. év

Témavezető:

dr. Lázár Zsolt egyetemi adjunktus,
Babeş-Bolyai Tudományegyetem, Kolozsvár, Fizika Kar, Magyar Fizika Intézet

Tartalomjegyzék

Kivonat

Biológiai rendszerek és egyéb komplex, adaptív jelenségek jól modellezhetők egyszerű feladatokat megoldó, időben fejlődő hálózatokkal.

A megoldás pontossága és időigénye, illetve a hálózat által felhasznált egyéb erőforrások mértékének viszonya sok esetben magyarázza az élő rendszerek túlélési stratégiáit.

Jelen dolgozatban egy egyszerű hálózat genetikus algoritmus szerinti evolúcióját vizsgálom. Feltérképezem a hálózat által megoldandó feladat nehézsége és a hálózat mérete, illetve a megoldás hatékonysága közötti összefüggéseket. Ennek kapcsán tanulmányozom a fenti tényezők és a hálózat moduláris szerkezetének viszonyát.

Bevezető

A természetben megjelenő biológiai és más adaptív rendszerek viselkedése komplex, evolúciója nehezen írható le. Az egymással és a környezettel kölcsönható biológiai rendszerek időbeli fejlődése direkt módon nehezen kezelhető. Azonban jól lehet modellezni ezeket időben fejlődő, evolúciós hálózatokkal [1], melyek viselkedése úgy minőségileg, mint számszerűen összecseng természetben észlelt jelenségekkel.

Komplex rendszerek viselkedésére sok esetben, mint valamely optimalizációs feladatként tekinthetünk [2, 3]. Optimalizációs feladatok megoldását általában többrétű megszorítások mellett keressük [3]. Egy jó példa erre a biológiából, az agy, melyben a szinapszisok költségének függvényében kell létrejöjjön a neuronhálózat [2].

Komplex rendszerek esetén gyakran érdekes, hogy miként tud reagálni a változásokra, mennyire jól tud alkalmazkodni ezekhez [4, 5]. Biológiai rendszerek esetén ez létfontosságú, úgy az egyedek, mint a fajok túlélési képessége nő az adaptivitással [5, 6].

Nagyon sok komplex rendszer az optimalizációs feladatra moduláris megoldást ad, mely gyakran hierarchikus strukturát mutat [1]. Biológiai rendszerek esetén: aminosav, fehérje , sejt, stb. A társadalom, az emberi tudás felépítése is ilyen. Technológiáink is ilyenek. Jelenlegi elképzelések szerint az egyes modulok különböző funkcióra való szakosodása nagy mértékben növeli a rendszer adaptivitását. Ugyanakkor a modularitás járulékos többletköltségekkel jár [7].

A kutatásom célja komplex rendszerekben történő optimalizáció, adaptivitás és modularitás kapcsolatának feltérképezése, jól meghatározott mennyisegek és ezek közti viszony leírásán keresztül, egy neurális hálozati modell segítségével. Előre elvárom azt, hogy a fitness előre gyorsan, majd lassan növekedjen [1], és a fitness növekedésével nőjjön a kihasználtság is.

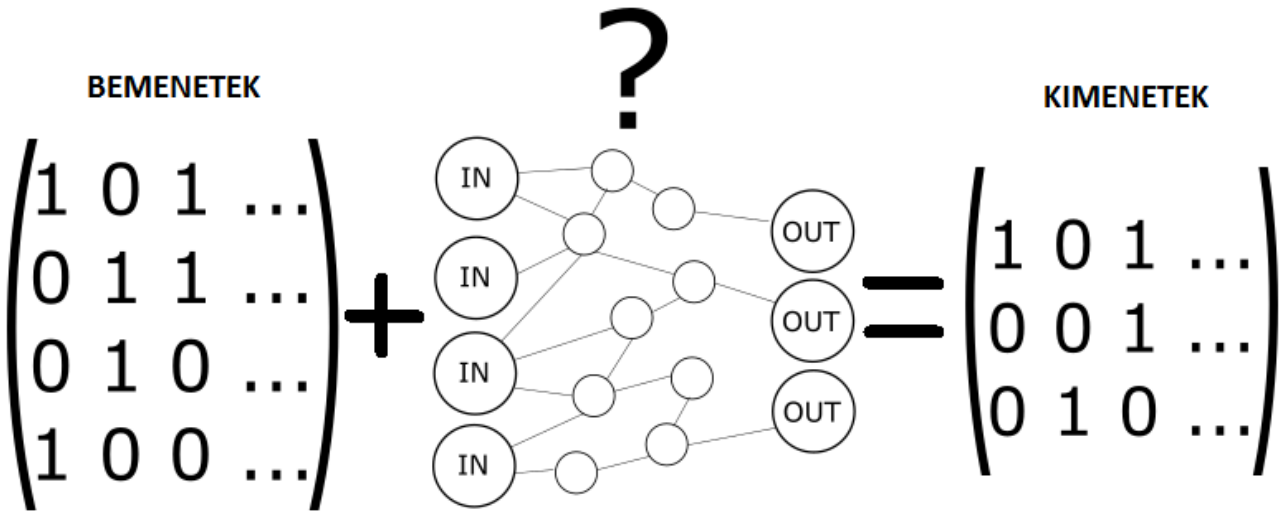
A dolgozatban foglaltakhoz való hozzájárulásom lefedi a szakirodalom áttekintését, szimulációs programok megtervezését és megírását, ezek futtatását, az eredmények kiértékelését, ábrák elkészíté-

sét és a dolgozat megírását.

1. fejezet

Optimalizáció neurális hálókban

Optimalizáció alatt azt értjük, hogy keressük a legjobb elemet a lehetséges elemek halmazából [2, 3]. A mi munkánkban ezt speciális neurális hálózatokon végezzük, ahol a csomópontok logikai függvények (kapuk), főként NAND, bináris számokkal dolgoznak, két bemenetűek és az élek elrendezése változó, mivel evolúciós hálók ezek. Ezeknek a hálóknak vannak kitüntetett csomópontjai, melyek nem logikai kapuk, hanem be- és kimenetek szerepét tölti be a hálózatban: a bemeneteknek értéket adva, a háló kimenetein értékek jelennek meg. Ezen hálózat optimalizációja alatt azt értjük, hogy valamilyen feladatra minél jobb megoldást adjon [1, 8, 9].



1.1. ábra. Háló optimalizációja: keressük azt a hálót, mely minél inkább teljesíti a feladatot, azaz a megfelelő bemeneti értékekre az elvárt kimeneti értékeket adja

A 1.1 ábrán látható, hogy létezik egy BEMENETEK és egy KIMENETEK nevű mátrix, mely

együttesen definiálja az optimalizációs feladatot: Keressük azt a hálót, mely a bemenetek mátrix oszlopait sorra véve, az oszlop elemeit a hálónak bemenetekként adva, a háló kimenetein a kimenetek mátrix megfelelő oszlopát adja vissza.

Értelmezhetünk egy olyan mennyiséget, melyet ***fitnessz***nek nevezünk, ami nem mást fejez ki, mint hogy egy hálózat mennyire optimális az adott feladatra [8, 9]. Ezt egy 0 és 1 közti racionális számmal írhatjuk le, és úgy számítható ki, hogy végig vesszük az összes oszlopot a bemenetek mátrixon, és megszámoljuk, hogy a hálózat kimenetein hány érték talál a feladatban szereplő kimeneti mátrix értékeivel, és ezt elosztjuk a kimeneti mátrix elemeinek számával [1]. Mondhatjuk úgy is, hogy a jó kimenetek és az össz kimenetek aránya a fitnessz.

1.1. Genetikus algoritmus

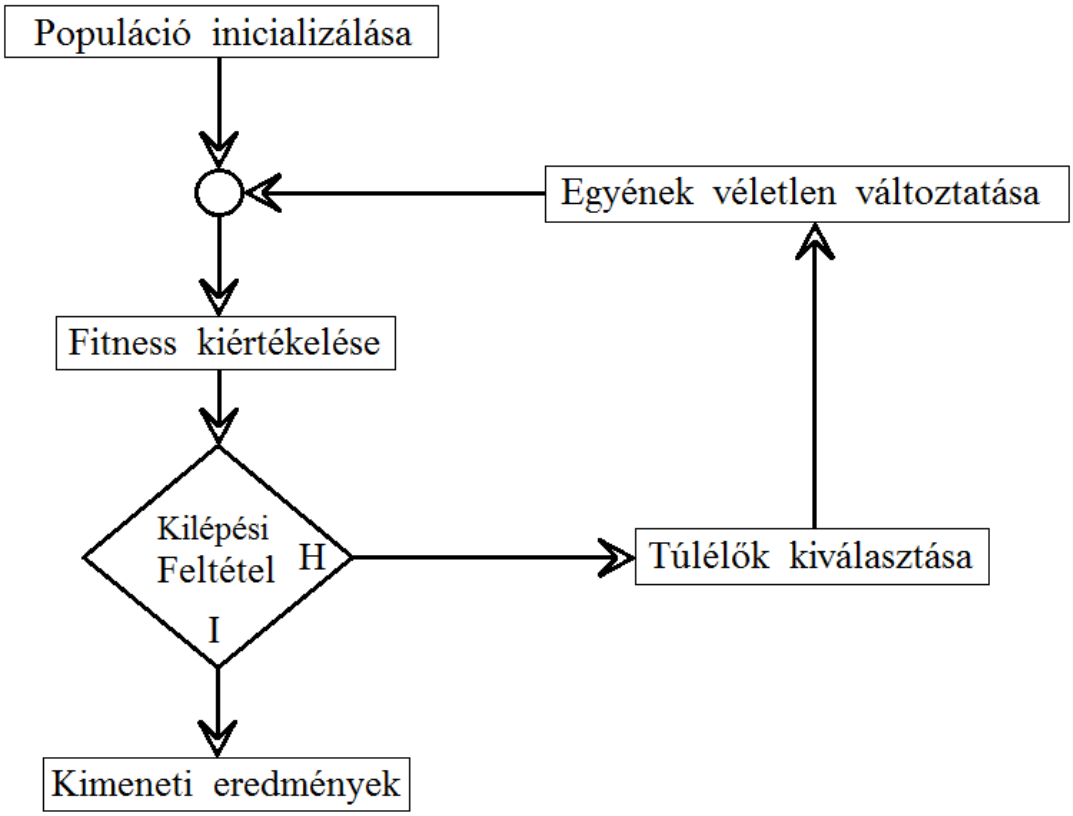
Genetikus algoritmusok alatt olyan keresési technikák egy osztályát értjük, melyekkel optimumot vagy egy adott tulajdonságú elemet lehet keresni [8, 9, 10]. A genetikus algoritmusok speciális evolúciós algoritmusok [10, 11, 12].

Az 1.2 ábrán látható genetikus algoritmus egyfajta evolúciós algoritmus. Ez a biológiából kölcsönzött evolúcióra épít, modellezi az evolúciót, és alkalmazási lehetőséget nyújt más témabeli szimulációkra és optimumfeladatok megoldására [11, 12, 13]. A genetikus algoritmus főbb lépései:

- Inicializáljuk a populációt
- Ciklus
  - Kiértékeljük a fitnesst
  - Ha a kiszabott feltétel teljesül, akkor vége a programnak
  - Adott módon túlélőket jelölünk ki
  - Az egyéneket véletlen módon változtatjuk (mutáljuk)

Ez a módszer azért jó, mert benne van a véletlenszerűség (egyének véletlen változtatása), mely a természet egyik alap mozgatója [12, 13], így a megoldás természetes szelekció révén (túlélők kiválasztása) konvergál az optimum felé.

A módszer egyik fő előnye, hogy a számítástechnikában előforduló problémák egy nagyon széles osztályára alkalmazható, ugyanakkor általában nem használ területfüggő tudást, így akkor is működ-



1.2. ábra. Genetikus algoritmus: inicializáljuk a populációt, majd egy ciklusban kiértékeljük a fitnesszt, ha a kilépési feltétel nem teljesül, túlélőket választunk ki, majd véletlenszerűen megváltoztatjuk őket, és újból ciklus, amíg a kilépési feltétel igaz nem lesz, ezután kiértékeljük az eredményt.

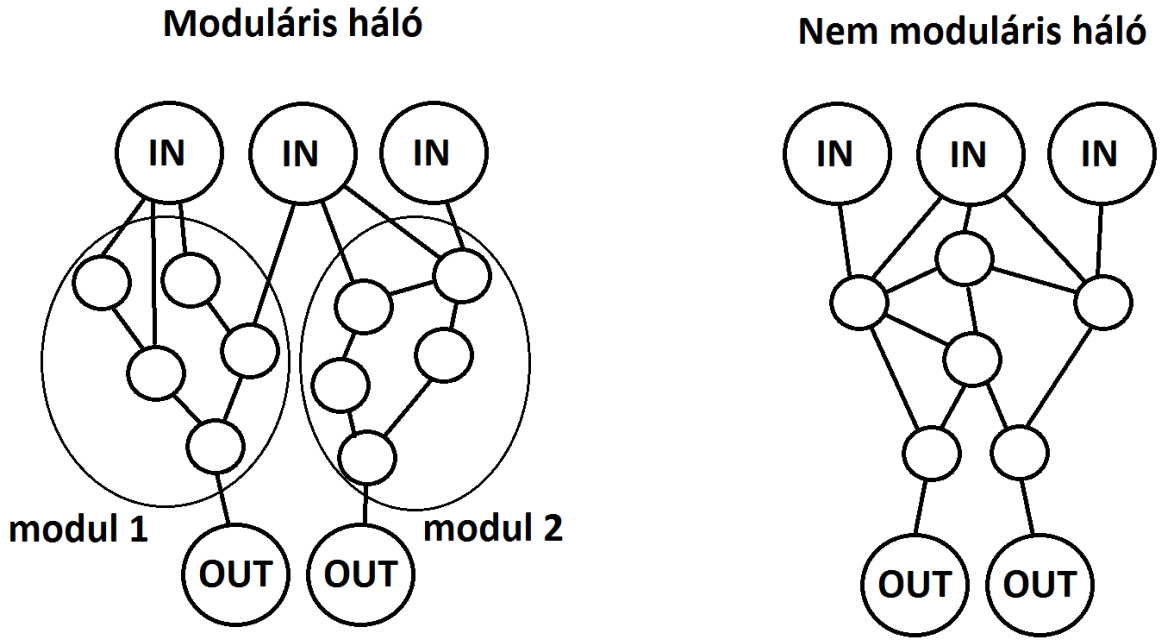
dik, ha a feladat struktúrája kevéssé ismert [10, 12]. Alkalmazása: pl.: a gráf színezési problémában, gépi tanulásban, stb. [11, 13].

1.2. Modularitás

Hogy megértsük a modularitás fogalmát, nézzük a 1.3 ábrát, ahol láthatjuk, hogy mi a különbség egy moduláris és egy kevésbé moduláris hálózat között.

Látható, hogy a moduláris hálókban jobban csoportosíthatóak a csomópontok, strukturálisan egyszerűbb a hálózat [7, 14]. Modulokra, klaszterektre osztható, melyek jobbára független részfeladatokat látnak el. Ezekben a klaszterekben az élsűrűség nagyobb. A klaszterek más-más feladatot látnak el, más a funkcionalitásuk [14, 15].

Egy hálózat fent említett tulajdonságainak jellemzésére vezették be a modularitási mértéket a



1.3. ábra. Modularitás neurális hálóban: jól elkülöníthető, sűrűbben kötött csomópont alhalmazok vannak, melyek alfunkciókra szakosodnak (bal oldali ábra)

továbbiakban egyszerűen modularitást, melyet a hálózat heterogenitásának számszerű kifejezését adja. Ennek egyik fajta kiszámítására nézzük a következő egyenletet:

$$Q = \frac{1}{(2m)} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{(2m)} \right] \delta(c_v, c_w)$$

Ahol v és w a csomópontok indexei, m a kötések száma, A a szomszédsági mátrix, k a csomópont fokszáma. A fenti egyenlet segítségével a Q modularitás számszerűen meghatározható. Az egyenletben szereplő tagokat nem részletezem, az egyenlet csupán formailag való ismertetése volt a cél.

Kutatásom során egy hálózat modularitásának kiszámítására az "igraph" nevű csomagot használom fel.

1.3. Modularitás spontán evolúciója neurális hálókban

A dolgozat egyik alappillére a Kashtan és Alon munkássága [1]. Csoportjuk neurális hálózati és elektronikai áramköri kapcsolási modell segítségével végeztek szimulációkat. A cikkben hangsúlyos figyelmet kapott a modularitás. Fő módszerük a 1.3.1 szakaszban tárgyalt genetikus algoritmuson

alapult.

1.3.1. Genetikus algoritmus evolúciós hálókra

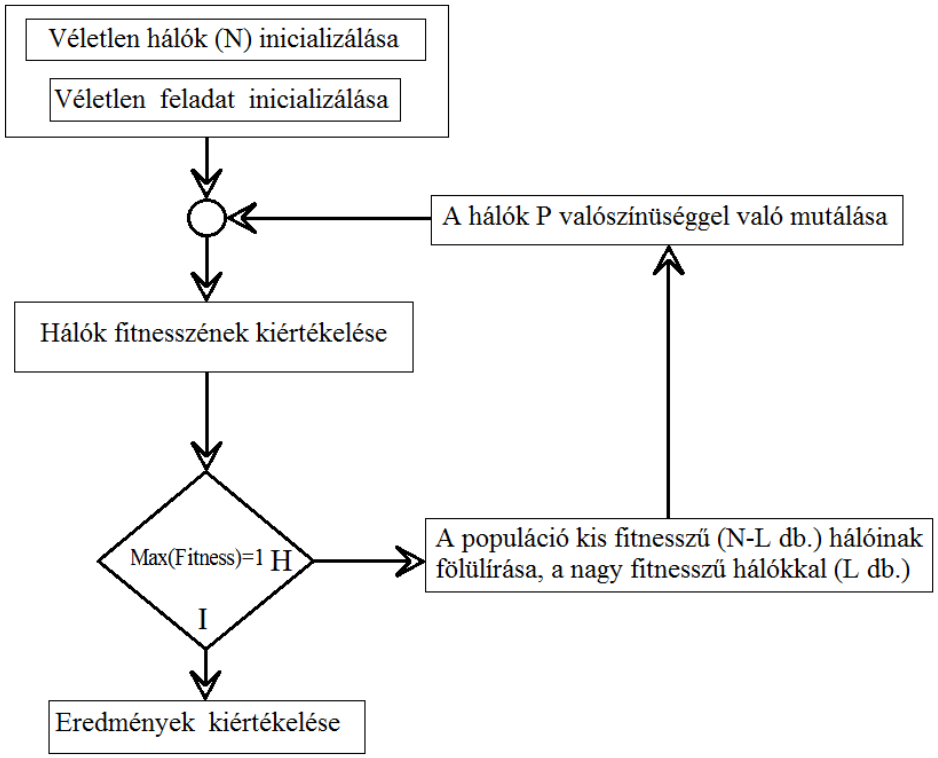
Az algoritmust N véletlenszerű, ám lerögzített csomópontszámú (paraméter), be- és kimenet számú háló inicializálásával kezdjük. Jellemzően N = 1000 hálóval dolgozunk, a háló bemenetei N_{in} egy-egy szimulációban más és más, paraméterként adjuk meg úgy, ahogy az N_{out} kimenetek számát és a csomópontok számát is. A bemenetek számához kötött a véletlen feladat inicializálása, ahol $2N_{in}$ számú oszlop (variáció) jelenik meg, N_{in} számú sor a bemenetekre és Nout számú sor a kimenetekre (ha mátrixokkal dolgozunk). Ciklusban: mind az N = 1000 háló fitnessét értékeljük ki a feladat és a hálók függvényében, melyek hálónként egy 0 és 1 közötti racionális szám lesz Ha ezen fitness-ek (N db fitness, minden hálóra egy-egy) maximuma nem éri el az 1-et (Max(Fitness) !=1), akkor: Az L (L = 300) legjobb fitnesszű hálót kiválasztjuk és sokszorosítjuk N-re (1000-re), majd: $P(P = 70$ %) valószínűséggel egyenként minden hálót mutálunk (átkötünk egy-egy élet). Ismételjük a ciklust addig, amíg a maximális fitness 1 nem lesz (Max(Fitness)=1) Végül kiértékelhetjük az eredményeket. A fent említett algoritmus és paraméterek (N = 1000, P = 70 %, L = 300, stb.) az [1] forrás alapján lettek kiválasztva.

1.3.2. Korábbi eredmények

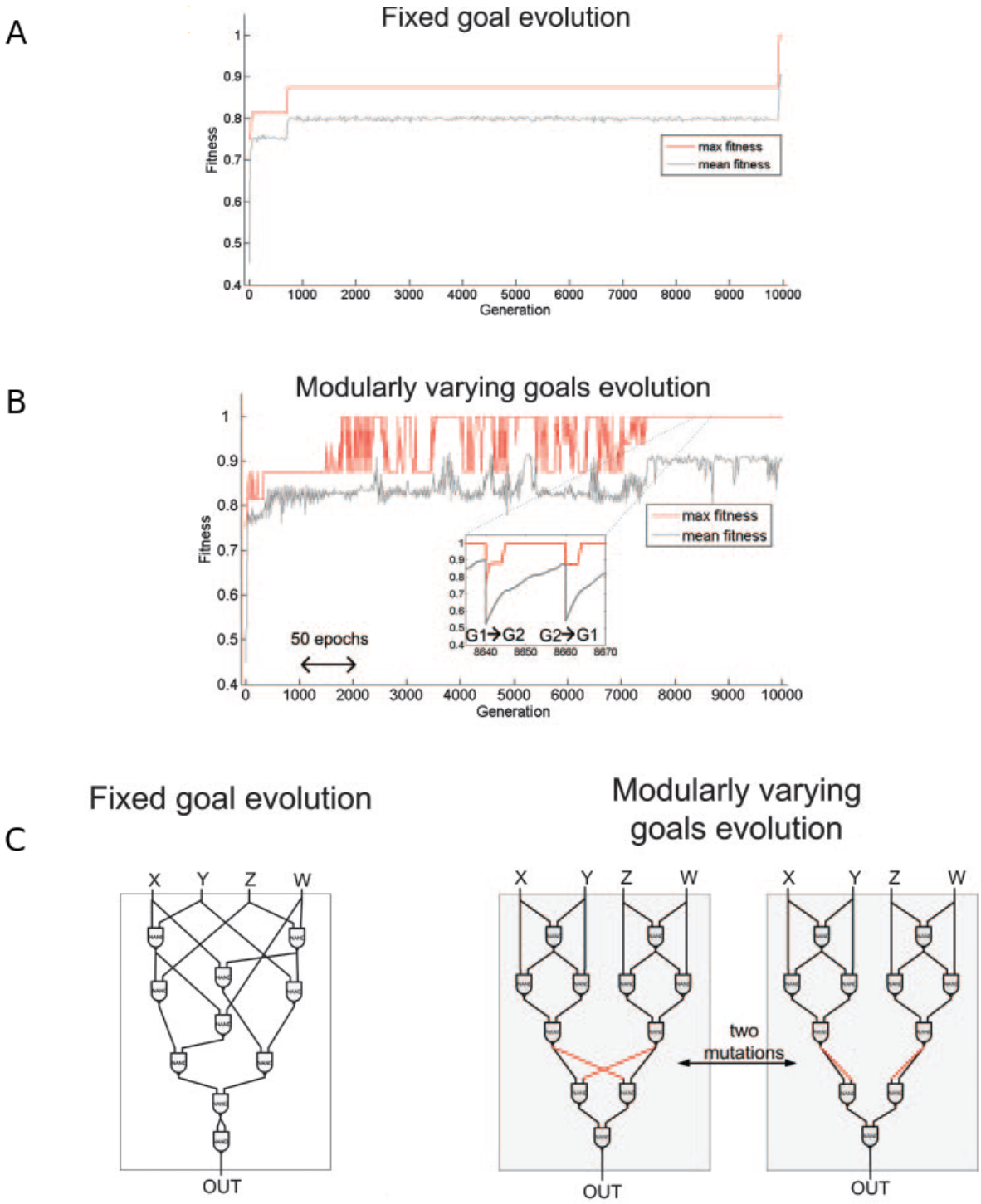
A cikkbeli szimulációk és lényeges eredmények a következők voltak:

Ha a szimuláció ideje alatt állandó feladatra próbáltak hálózatokat optimalizálni genetikus algoritmussal, akkor a megoldás időben előre gyorsan, majd csak lassan konvergált a tökéleteshez.

Az előzővel ellentétben, ha úgy optimalizálták a hálókat egy feladatra, hogy a feladat adott ciklussal váltakozott két feladat közt, akkor a tapasztalat a következő volt: Sokkal gyorsabban létre jött a megoldás. És azt tapasztalták, hogy az előző, fix feladatos, szimulációval ellentétben, itt a végén a hálózat moduláris lett sőt, csupán kevés mutáció (átkötés) révén az egyik feladatra optimális háló, immár a másik feladatra lesz optimalizálva. Mindkét esetben a szimuláció paraméterei: 1000 háló, 300 túlélő, hálónkénti 0.7 valószinűséggel mutálva, 11 kapuval, 4 bemenettel és 1 kimenettel. A váltott feladatnál 20 generációnkénti váltogatás. Ezekhez nézzük az 1.5 ábrát, melyet az [1] forrásból kölcsönöztünk.



1.4. ábra. Genetikus algoritmus evolúciós neurális hálókra: a sajátos genetikus algoritmus, hálók és feladat inicializálása, majd ciklusban kiértékeljük a fitnesszt, a feltétel szerint túlélőket választunk ki, mutáljuk őket valamilyen valószínűséggel



1.5. ábra. Spontán modularitás evolúciója Kashtan és munkatársai nyomán [1]: (A) fix feladat: a felső görbe a maximális fitness, az alsó az átlag. Viszonylag lassú megoldás (B) váltott feladat: a felső görbe a maximális fitness, az alsó az átlag. Viszonylag gyors megoldás. 20 generációnkénti pontábrázolás. (C) a balról levő háló fix feladatra optimalizált, nem moduláris. A jobbról levő hálók váltott feladatra optimalizáltak, modulárisak, és néhány átkötéssel, az egyikre optimalizált háló, a másikra lesz optimális. A genetikus algoritmus paraméterei: 1000 hálós populáció, 300 túlélő, 70%-os mutációs valószínűség, 11 kapu (csomópont), 4 bemenet, 1 kimenet. A váltott feladatnál 20 generációnkénti váltogatás.

2. fejezet

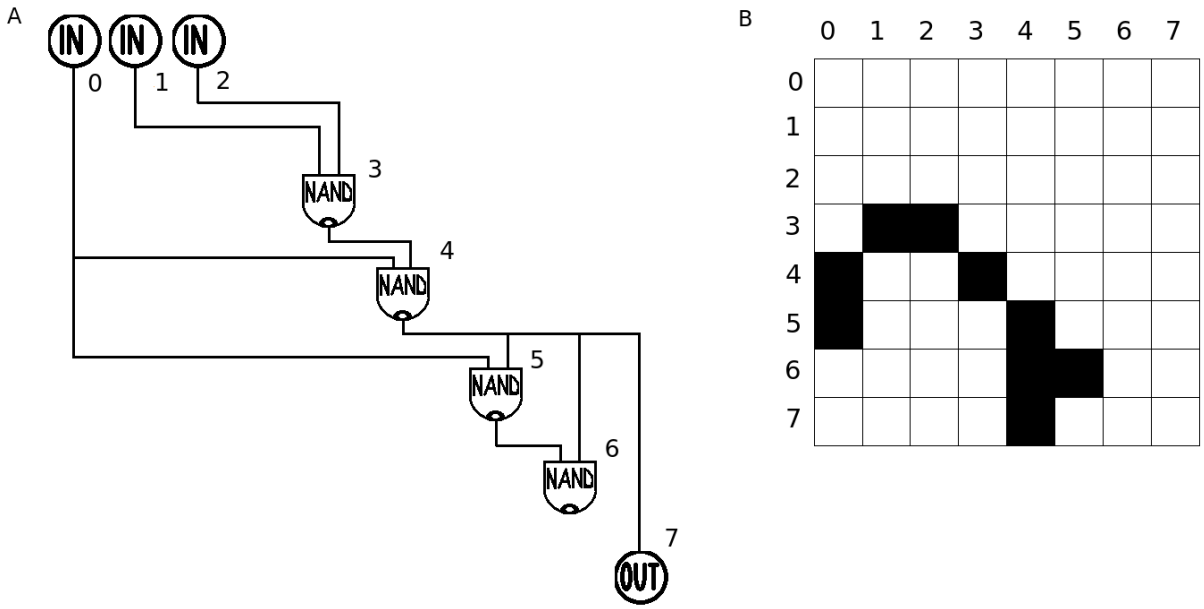
Kihasználtság és optimalizáció neurális hálókban

A kutatás következő részében megpróbáltuk reprodukálni a 1 fejezetben bemutatott eredményeit a [1] cikknek. Ennek első lépése a szimulációs kód megtervezése és megírása volt. A kód alapjául szolgáló algoritmus nagyvonalakban az 1.3.1 szakaszban volt tárgyalva. Ezen algoritmus számítógépes programba való ültetésére a C++ programozási nyelvet használtam. Ezt a programozási nyelvet azért választottam, mivel sok alapszintű műveletre van szűkség az algoritmusban, ezért a futás hatékonysága elsődleges szempont volt.

A szimuláció során a program néhány paraméter alapján dolgozik. Nézzük meg közülük a legfontosabbakat:

hálózatban felhasznált NAND kapuk száma: ezt az értéket széles intervallumon használjuk másmás szimulációk esetén: néhány kaputól kezdődően, egészen több ezerig, de igazság szerint csak a számítógép memóriája szab határt ennek a paraméternek A hálózatok száma: ezt jellemzően 1000-nek vettem a szimulációk során, de felsőhatárt itt is csak a memória adhat A túlélő hálók száma: ezt is jellemzően az előző paraméter függvényében 300-nak vettem Annak a valószínűsége, hogy egy hálót mutálunk egy generációban: ez is jellemző az előző két paraméter függvényében 0.7. A hálózat bemeneteinek száma: ez kisebb intervallumon mozog általában, mert a feladat bonyolultsága ($2^{N_{in}}$) exponenciálisan nő a bemenetek számával. A hálózat kimeneteinek száma: ez mozoghatna nagyobb intervallumon is, mert ennek növekedésével nem nő exponenciálisan a a feladat bonyolultsága, de jellemzően kisebb szokott lenni a bemenetek számánál sőt, sok esetben csak 1-et használok.

Meg kell említenem továbbá, hogy azokban a hálózatokban, mellyel a program dolgozik, kellett egy megszorítás: nem jöhet létre olyan kapcsolat, mely visszacsatolast okozhat, vagyis olyan kötések nem lehetnek, ahol egy logikai kapu bemenete direkt, vagy akár más kapukon keresztül indirekt módon a saját kimenetére csatlakozik, mivel ez határozatlansághoz vezet. Ennek érdekében a hálokban minden kapunak van egy plusz tulajdonsága, amelyet szintnek nevezek. Ez egy sorszám, hogy az adott kapu hányadik szinten helyezkedik el. A hálózat bemenetei az első szint, a kimenetei pedig az utolsó. A szintek száma változhat. Ez így megkönnyíti azt a megszorítást, hogy ne történjen visszacsatolás, mert akkor csak azt kell kikössük, hogy bármely kapu bemenete, csak a saját szintje alatti szinteken levő kapuk kimenetéhez csatlakozhat a hálózatban. A kezelhetőség és egyszerűség érdekében, minden szinten csak 1 kapu van, így az egész hálóban van egy topológiai sorrend: egymás utáni kapuk, egymás utáni szinten vannak, így minden kapu bemenete csak az előtte levő kapuk kimenetéhez csatlakozhat. Ez a megszorítás a szomszédsági mátrixban is megjelenik, mert a mátrix alsó háromszögében lesznek csak értékek, a felsőben nem, mert ott nincs kötés. Nézzük ehhez a 2.1 ábrát.



2.1. ábra. (A) NAND kapukból épített topológiailag rendezett (aciklikus) neurális háló (B) szomszédsági mátrix: az egyes sorokban a befestett négyzetek azt jelölik, hogy az adott kapu bemenetként értéket a befestett négyzetek oszlopaiban levő kapuktól kap, a kötési megszorítás miatt csak a mátrix alsó háromszögében lesznek kötések (befestett négyzetek)

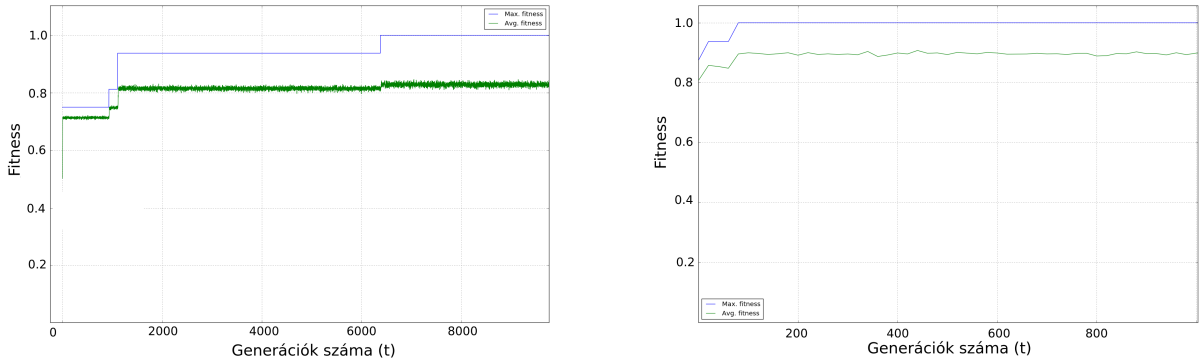
A 2.1 ábrán megfigyelhető a következő: a 6-es kapu kimenete nem vezet sehova, és a 5-ös kapu kimenete csak a 6-os ra megy. Tehát levonható az a következtetés, hogy az 5-ös és 6-os kapuk úgy-

mond nem hasznos kapuk, mivel nem befolyásolja a hálózat reakcióját a kimeneten, adott bemenetek esetén.

A fent leírtak alapján értelmezhetünk egy kihasználtságnak nevezett tulajdonságot, mely azt fejezi ki, hogy egy ilyen hálózatban mennyi a hasznos és az össz kapuk aránya. Ez egy 0 és 1 közti racionális szám lesz, a hasznos és az össz kapuk számának hányadosa.

Első lépésben a [1] Kashtan cikkjét alapul véve, mely néhány eredményét szerttük volna reprodukálni, szimulációkat végeztünk:

Ugyanazok a paraméterek mellett futtatott szimuláció: 1000 háló, 300 túlélő, hálónkénti 0.7 valószinűséggel mutálva, 11 kapuval, 4 bemenettel és 1 kimenettel. A szimulációból kapott eredmény a 2.2 ábrán látható.

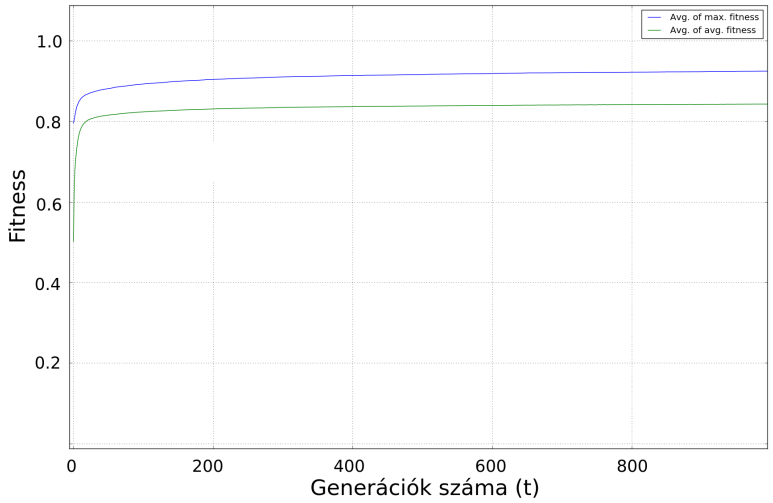


2.2. ábra. Saját algoritmussal szimulált fitness evolúció Kashtan [1] algoritmusával való összevetés végett. (Bal oldal) Fix feladat: A fitnessz populáció átlaga és maximuma. Zöld görbe a populáció átlag, kék görbe a populáció maximum. (Jobb oldal) Váltott feladat: A fitnessz populáció átlaga és maximuma. A genetikus algoritmus paraméterei: 1000 hálós populáció, 300 túlélő, 70%-os mutációs valószinűség, 11 kapu (csomópont), 4 bemenet, 1 kimenet. A váltott ebben afeladatnál 20 generációnkénti váltás.

Jól látható a 2.2 ábrán, hogy az átlagos fitness 0.5-ről indul. Ez minden futtatás és minden paraméterre így lesz ha sokaságátlagot nézünk, mert egy-egy kimeneten a hálóban csak két érték lehet, mert bináris számokkal dolgozunk, és ha az átlagot vesszük, az 0.5 arányban egyezni fog az elvárttal, mert véletlen hálókkal indítunk. Sikerült reprodukálni ezt, melyet a [1] cikk értelmében elvártunk.

A fent leírt paramétereket (1000 háló, 300 túlélő, hálónkénti 0.7 valószínűséggel mutálva, 11 kapuval, 4 bemenettel és 1 kimenettel) felhasználva 1000-szer végeztük el a szimulációt, és abból így egy sokaságátlagot kaphatunk a fitness értékének időbeli változására. Ez látható a 2.3 ábrán.

Jól kivehető a 2.2 és a 2.3 ábrákból, hogy visszakaptuk az [1] forrásban felvetett viselkedést,



2.3. ábra. A fitness sokaságátlagának evolúciója. Zöld görbe a populáció átlag, kék görbe a populáció maximum. A genetikus algoritmus paraméterei: fix feladat, 1000 hálós populáció, 300 túlélő, 70%-os mutációs valószinűség, 11 kapu (csomópont), 4 bemenet, 1 kimenet.

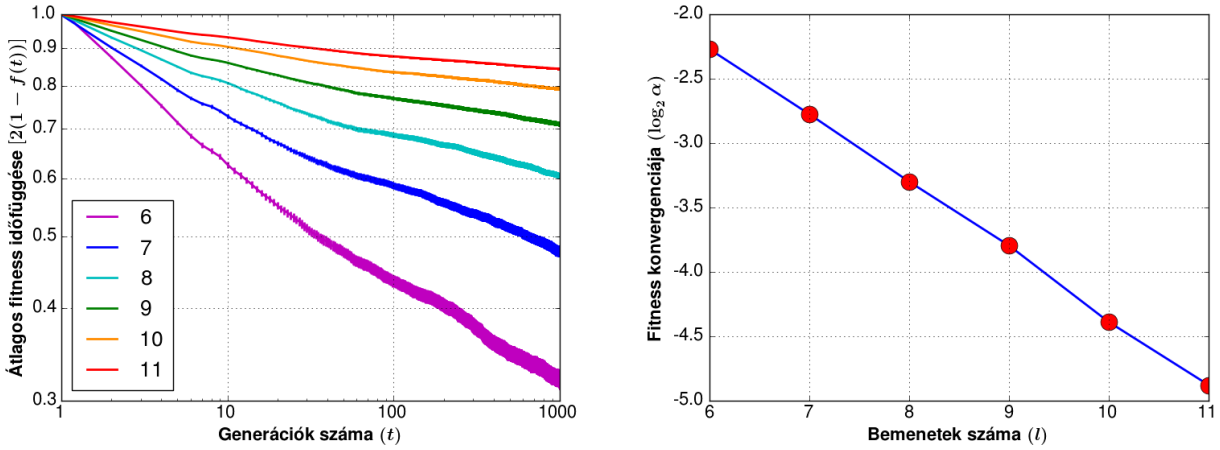
miszerint a fitness előre gyorsan, később lassan konvergál az egyhez

A következő, melyet megvizsgáltunk, hogy ha adott periódus mellett váltogatjuk a feladatot két feladat közt, akkor gyorsabban eljutunk a megoldáshoz. Látható a 2.2 ábra jobb oldalán, hogy a bal oldali ábrához képest gyorsabban eljutunk a megoldáshoz, és ha a fitness elérte az egyet, akkor idővel még a feladat váltásának perdiódusa alatt újra megtalálja a megoldást. A szimuláció paraméterei nem változtak: 1000 háló, 300 túlélő, hálónkénti 0.7 valószinűséggel mutálva, 11 kapuval, 4 bemenettel és 1 kimenettel, és 20 generációnként váltjuk a feladatot két feladat közt. Az eredményeket, melyet a [1] cikk értelmében elvártunk minőségileg sikerült reprodukálni. Mivel a cikkben egy konkrét futtatáss jelenik meg, ezért mi is csak egy futtatással tudjuk összehasonlítani, így számszerű eredmény nem hasonlítható össze.

A következőkben azt vizsgáltuk, hogy hogyan függ a fitness a feladat méretétől, vagyis a háló bemeneteinek a számától (a feladat mérete exponenciálisan nő a bemenetek számával).

A következőket végeztük el: minden szimulációban lerögzítettük a következő paramétereket: 1000 háló, 300 túlélő, hálónkénti 0.7 valószinűségü mutáció, 100 kapu, 1 kimenet, és 100-szoros szimuláció (sokaságátlag miatt). A bemeneteket változtattuk 6-tól 11-ig. A fitness bemenetek számától való függésére az 2.4 ábrán látható eredményeket kapuk.

A 2.4 ábra balról levő ábráján látható, hogy a feladat növekedésével lassabban nő a fitness. A loglog skálán közelíthetőleg egyenesek az egyes bemenetekre kapott fitnesszek. Mivel az első 6 gene-



2.4. ábra. Fitness bemenetek számától való függése: 1000 háló, 300 túlélő, hálónkénti 0.7 valószínűségü mutáció, 100 kapu, 1 kimenet, és 100-szoros szimulációra vett sokaságátlag. A bemeneteket változtattuk 6-tól 11-ig. A standard hibával van ábrázolva mindkét grafikon. A balról levő ábrán log-log skálán ábrázoltuk a $2(1 - fitnessz)$ időfüggését különböző bemeneti számra 6-11. Itt kicsitt nagyobb a szorás, viszont a standard hiba elég kicsi. A jobbról levő ábrán a bemenetek számának a függvényében van ábrázolva a fitness konvergenciája (2-es alapú logaritmusa a meredekségnek) az első 6 generációban. Itt kicsi a szorás és a standard hiba pedig 10^{-6} nagyságrendű

rációban a lineáris regresszió nagyon pontos (10^{-6} nagyságrendű az eltérés), és ennek a szakasznak a meghosszabbítása nagyjából rajta van a teljes görbén, ezért csak ezzel a szakasszal dolgoztunk a továbbiakban. A 2.4 ábra jobbról levő ábráján az első 6 generációval számolva láthatjuk, hogy a bementek számának növekedésével lineárisan csökken a fitness konvergenciája.

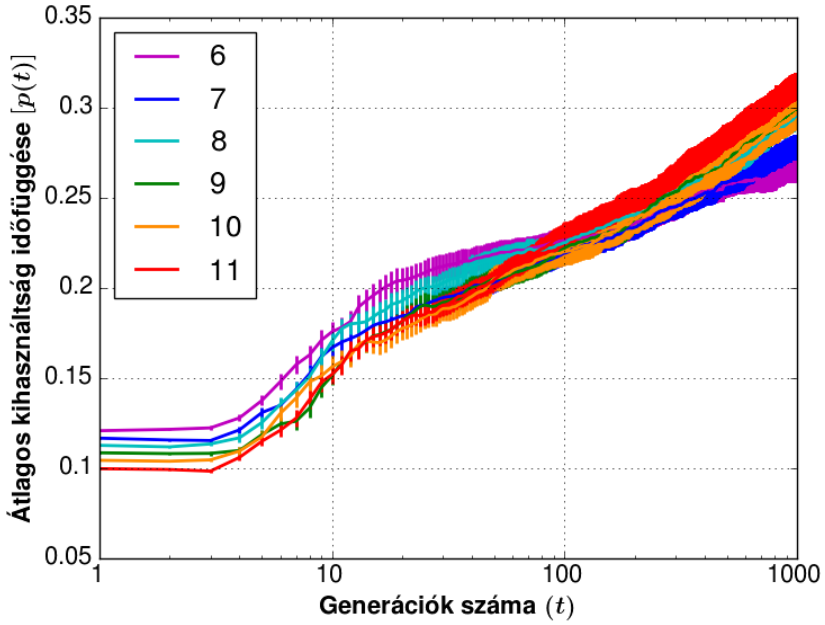
Levonahtjuk a következtetést, hogy a fitness viselkedése az időben hatványfüggvényszerű.

Vizsgáltuk a már említett kihasználtságot. A szimulációk paraméterei maradtak: 1000 háló, 300 túlélő, hálónkénti 0.7 valószínűségü mutáció, 100 kapu, 1 kimenet, és 100-szoros szimulációra vett sokaságátlag. A bemeneteket változtattuk 6-tól 11-ig. A kapott eredmények 2.5 ábrán láthatóak.

Látható a 2.5 ábrán, hogy a kihasználtság nem olyan egyszerű mint a fitness. Megfigyelhetjük, hogy egyértelműen nő a kihasználtság az időben (generációk). Továbbá azt tapasztaltuk, hogy kezdetben a könnyebb feladatot megoldó hálók kihasználtsága a nagyobb, viszont ez úgy 100 generációnál pont fordítottja lesz, a nehezebb feladatot megoldó hálók kihasználtsága lesz a nagyobb.

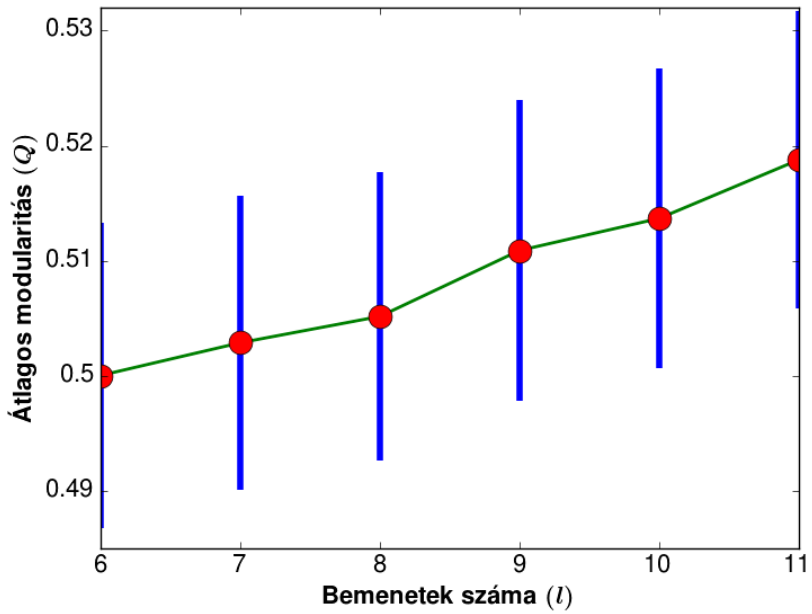
Megvizsgáltuk, hogy hogyan függ a hálózatok modularitása a feladat méretétől. Ehhez kombinált populáció- és sokaságátlagokat vizsgáltunk a végső generációkban a modularitásokra. Látható a 2.6 ábrán a kapott eredmények eredmények.

Bár jelentős szórás kíséri a modularitás feladatmérettől való függését, a szinte eltűnő standard



2.5. ábra. Kihasználtság időfüggése: log skálán ábrázolva az idő függvényében a kihasználtság. A szimulációk paraméterei: 1000 háló, 300 túlélő, hálónkénti 0.7 valószínűségü mutáció, 100 kapu, 1 kimenet, és 100-szoros szimulációra vett sokaságátlag. A bemeneteket változtattuk 6-tól 11-ig. A standard hiba is ábrázolva van, mely jelen esetben jelentős

hibára való tekintettel az összefüggés egyértelmű: nő a modularitás a feladat méretével



2.6. ábra. Modularitás feladatmérettől való függése: a feladat mérete, azaz a bemenetek számának függvényében a végső generációk hálózataiban kombinált populáció- és sokaságátlaga a modularitásra. Jelentős szórás kíséri a modularitás feladatmérettől való függését, de a szinte eltűnő standard hibára való tekintettel, az összefüggés egyértelművé válik

3. fejezet

Következtetés

Munkánk során Reprodukáltuk [1] Kashtan és munkatárasi eredményeit a témával kapcsolatban. Az általuk használt algoritmust álatlánosítottuk, és ezzel végzett szimulációknál tanulmányoztuk a fitness sokaságátlag viselkedését, a konvergencia feladat méretétől való függését. Kezdeti eredményeink azt mutatjak, hogy hatvanyfüggvény szerű a fitness időfüggése, és a konvergencia exponenciálisan függ a feladat méretétől.

A hálózatok kihasználtságának és a fitness kapcsolatára vonatkozó minőségi megallapításokra jutottunk. Ezek erősen korreláltak, aminek az lehet a magyarázata, hogy könnyebb jobb eredményt elérni több hálózati csomópont, azaz a rendszer fázisterének nagyobb felhasználása eseten.

A hálózatok modularitásának tanulmányozása alatt azt tapasztaltuk, hogy a modularitás nő, ahogyan növeljük a feladat méretét.

A szimulációs program általánosabb formában van megírva, mint azt a dolgozatban tárgyaltak szükségessé tették volna, így felhasználható lesz a jövőben a téma általánosított kutatására is. Különösképpen érdekes kérdés a hálózati elemek (kapuk) sajátosságainak hatása a rendszer evolúciójára. NAND kapukról általánosabb és változatosabb kapuhálózatra való áttérés kéne legyen az első további lépés.

Irodalomjegyzék

[1] N. Kashtan, U. Alon: *Spontaneous evolution of modularity and network motifs*, PNAS **102** no. 39 (2015)

[2] B. T. Gazdag: *Globális optimalizálás*, BME TTK Matematika Intézet, Typotex kiadó (2011)

[3] S. Boyd, L. Vandenberghe: *Convex optimization*, Cambridge University Press (2004)

[4] J. S. Lansing: *Complex adaptive systems*, annual review **32**:183-204 (2003)

[5] J. H. Holland: *Studying complex adaptive systems*, Journal of Systems Science and Complexity **19** Issue 1 (2006)

[6] D. C. Andrus: *Toward a complex adaptive intelligence community the wiki and the blog*, SSRN **49** no 3 (2005)

[7] J. M. Hofman, C. H. Wiggins: *Bayesian approach to network modularity*, Phys. Rev. Lett. **100**, 258701 (2008)

[8] A.E. Eiben, J.E. Smith: *Introduction to evolutionary computing*, citeulike 608199 (2007)

[9] L. M. Smitt: *Theory of genetic algorithms*, Elsevier Theoretical Computer Science **259** (2001)

[10] J. R. Koza: *Genetic programming: On the programming of computers by means of natural selection*, MIT press (1992)

[11] D. E. Goldberg: *Genetic algorithms in search, optimization, and machine learning*, Pearson Education (2006)

[12] M. Mitchell: *An Introduction to Genetic Algorithms*, MIT press (1996)

[13] J. J. Grefenstette: *Genetic Algorithms For Machine Learning*, Springer Science and Business Media **13** no 2-3 (1993)

[14] M. E. J. Newman: *Modularity and community structure in networks*, PNAS **103** no 23 (2006)

[15] G. Schlosser, G. Wagner: *Modularity in development and evolution*, The University Of Chicago Press (2004)

Köszönetnyilvánítás

A dolgozat zárásaként ezúton szeretnék köszönetet mondani témavezetőmnek, dr. Lázár Zsoltnak, a gyakori és rugalmas konzultációkért, és minden segítségért, amit tőle kaptam e kutatás alatt. E dolgozat ezek nélkül nem jöhetett volna létre.