Eötvös Loránd Tudományegyetem
Természettudományi Kara

Sztochasztikus dinamikai rendszerek és alkalmazásaik

Diplomamunka

Készítette: Hegyes István, fizikus

Témavezető: Dr. Lőrincz András

Eötvös Loránd Tudományegyetem, Információs Rendszerek Tanszék

Budapest, 2002

Kivonat

A world wide web megalakulása óta exponenciálisan növekszik. Mérete mára meghaladta a kétmilliárd dokumentumot. Ebben a hatalmas adathalmazban egyre nehezebb megtalálni a minket érdeklő információkat. A hagyományos keresők nem tudnak lépést tartani a növekedéssel. Új utak keresésére van szükség.

Dolgozatomban az információ felkutatásának egy új eszközével host-barangoló architektúra kialakításának lehetőségével foglalkozom. Bemutatom a közelmúlt irodalmi eredményeit, mind a barangolókra, mind pedig a világháló szerkezetére vonatkozóan, majd bemutatom az általunk kialakított rendszer viselkedését. Ehhez két népszerű mesterséges intelligencia módszert, a szózsák alapú szöveg-osztályozást és a megerősítéses tanulást, alkalmazom. Ezek után bemutatom az általam készített szimulációs rendszert, melyet kísérleteim elvégzésére felhasználtam, különös tekintettel az ún. weblogok kialakítására. Megmutatom, hogy ezt alkalmazva, hogyan képesek a barangolók felosztani a rendelkezésükre álló területet. Tesztjeimet kimerítően értelmezem. Felhasználva a tapasztalatokat ismertetem a rendszer alkalmazásának lehetőségeit a való világban.

TARTALOMJEGYZÉK

1 Bevezetés

Napjainkra kialakult a világ leghatalmasabb elosztott információs rendszere, az Internet, mely lehetővé tette információ hatékony megosztását világszerte. Az Internet számos tulajdonsága jól modellezhető, vizsgálható és megérthető véletlen gráfok segítségével. A world wide web az Internet hálózati kiszolgálóin megtalálható, böngészőprogramok segítségével megjeleníthető, hiperhivatkozásokat tartalmazó oldalak összessége. 1995-ben a world wide web mindössze 50 millió oldalt tartalmazott, melyek többnyire gyakorlatilag is a kiszolgáló meghatározott tárterületein található statikus fájlok voltak, melyek irányított hivatkozásokkal rendelkeztek egymásra. A gyors technológiai fejlődés ezt a képet részben megváltoztatta. Egyrészt a web mérete 1995 óta minden évben megduplázódott, 2000 elejére elérve az 1 milliárd oldalt, napjainkra pedig már meghaladta a 2 milliárdot is. Másrészt a web dinamikus tartalma szintén növekszik, olyan időfüggő oldalak képében, mint a különböző híroldalak, pénzügyi, tőzsdei oldalak, szórakoztatóipar új produkcióinak oldalai.

Kialakult egy személyes segítő koncepció, mely egy olyan intelligens internetes barangolót jelent, mely a világháló linkjeit követve segíti használóját az őt érdeklő téma-specifikus információk megtalálásában. Annak érdekében, hogy a barangoló alkalmazkodhasson mind a felhasználó igényeihez, mind pedig a világháló lokális viszonyaihoz, megerősítéses tanuláson alapuló algoritmusok alkalmazhatóak. Nagy problémája azonban ennek az elképzelésnek, hogy amennyiben minden felhasználó

használ ilyen személyes segítőket, az könnyen a hálózat túlterheléséhez vezethet. Ennek legfőbb oka az, hogy a barangolók egymástól függetlenül működnek, nem befolyásolják egymás viselkedését, nem képesek az együttműködésre, az információk megosztására. Ennek elkerülése érdekében távlati célunk olyan host-barangoló architektúra létrehozása, mely hatékonyan képes feltérképezni az Internet behatárolt területeit, annak dinamikáját, változékonyságát is figyelembe véve. A barangolók a világháló linkjeit követik, és megerősítéses tanulás segítségével igyekeznek meghatározni, hogy melyik linkeket választva juthatnak releváns oldalakhoz. Az általuk bejárt területről kialakítanak egy kompakt reprezentációt, melyet képesek megosztani más barangolókkal, így működnek együtt.

Egy ilyen rendszer legalapvetőbb tulajdonságai modelleztem, és vizsgáltam működését a világháló behatárolt területein. A rendszert MATLAB környezetben valósítottam meg, míg a világháló megfelelő modelljének kialakítását mintavételezéssel, valódi barangoló szoftver felhasználásával oldottam meg. Idő hiányában egyelőre csupán a megfigyelhető jelenségek kis részét tudtam megvizsgálni.

2 A világháló statisztikai tulajdonságainak vizsgálata

2.1 A világháló gráf modellje

A world wide web modellezhető irányított gráfként melynek csomópontjai a különféle honlapok. A gráfban (*i*,*j*) irányított él jelenik meg, ha létezik hiperhivatkozás az *i* oldalon, mely a *j* oldalra mutat. Egy másik gráf definiálható, ha a domain szinten

tekintünk a világhálóra. Itt akkor van kapcsolat *a* és *b* domainek között, ha létezik legalább egy oldal az *a* domainen mely a *b* domain bármely oldalára mutat. *A* web akkor került az érdeklődés középpontjába, amikor kiderült, hogy a fent definiált gráfok kapcsolatainak eloszlása széles tartományokon keresztül hatványfüggvény alakú, ami skála-invariáns szerkezetet sejtet. Mivel a web irányított gráf, eltérő eloszlások mérhetőek az adott oldalon található, valamint az adott oldalra mutató kapcsolatok figyelembe véve, melyeket jelöljünk $P_{ki}(k)$, valamint $P_{be}(k)$ módon, ahol *k* jelenti a kapcsolatok számát. A web jelentős részére kiterjedő mérések alapján mindkét eloszlás hatványfüggvény alakú nagy *k* értékekre. $P_{ki}(k) = k^{-\gamma_{ki}}$, $P_{be}(k) = k^{-\gamma_{be}}$. Albert és Barabási az *nd.edu* tartomány szerkezetét vizsgálták meg, mely mintegy 325.792 oldalt tartalmazott. Méréseik szerint a letöltött gráf valóban hatványfüggvény-szerű eloszlást mutatott, melyre $\gamma_{ki} = 2.45$ és $\gamma_{be} = 2.1$. Ennél nagyobb, 200 millió oldalt tartalmazó Alta Vista barangolásból származó mintán végzett méréseket Border, és ettől némileg eltérő eredményeket kapott, $\gamma_{ki} = 2.72$ és $\gamma_{be} = 2.1$ adódott.

A világháló további érdekes statisztikai tulajdonságokkal rendelkezik Az egyik ilyen tulajdonság a hasonló méretű véletlen gráfokra nem jellemző magas klasztereződési hányados. Amennyiben az *i*. csúcs k_i éllel rendelkezik és a vele szomszédos csúcsok által kijelölt algráf $\acute{E}_i$ élt tartalmaz, az *i*. csúcshoz tartozó klasztereződési hányados értéke $C_i = \frac{2 * \acute{E}_i}{(k_i - 1) * k_i}$ melyet a teljes gráfra átlagolva kapjuk a gráf klasztereződési hányadosát. Irányított gráfok esetében a klasztereződési hányados nem értelmezhető a

hagyományos módon, ezért ebben az esetben a gráfot úgy tekintik, mintha nem lenne irányított, és így mérik ezt az arányszámot. Mint láttuk, véletlen gráfok esetén a hányados értéke igen kicsi, és a gráf méretének növekedésével tart 0-hoz. A web esetén azonban ez a szám több nagyságrenddel meghaladja a hasonló méretű véletlen gráfra jellemző mértéket, akár oldal szinten, akár domain szinten tekintjük. Yook (2000) mérései szerint $C_{domain} = 0.22 - 0.3$ míg hasonló méretű véletlen gráf esetén ez az érték $C_{rand} = 0.001$ lenne. A vizsgálatok eredményei részletesebben megismerhetőek [15]-ből.

Nagyon lényeges tulajdonsága a világhálónak az úgynevezett „kis világ" jellege. Ez a fogalom a 60-as években az USA-folytatott szociológiai vizsgálatok eredményeként került a köztudatba. Stanly Milgram vizsgálta a társadalmi kapcsolatok hálózatát, és azt tapasztalta, hogy az Egyesült Államok 250 millió lakosa közül véletlenszerűen párokat kiválasztva, azok átlagosan 5-6 személy közbeiktatásával képesek kapcsolatba lépni egymással.

A „kis világ" tulajdonság kvantitatív méréséhez definiálnunk kell a web-gráf átmérőjét, amire több definíció közül választhatunk. A hagyományos definíció szerint minden ($N * (N-1)$ számú) csomópont-pár távolságát meg kellene mérni, majd a kapott értékek maximuma lenne az átmérő:

$$diam = \max\{dist(i, j) | (i, j) \in V \times V, i \neq j\}$$

ahol V a gráf csomópontjainak halmaza. Ezzel a definícióval az a probléma, hogy egyrészt legtöbbször nem értelmezhető, mert találhatunk olyan csomópontokat, melyek között nem létezik irányított út, másrészt az útvonalhosszak eloszlásának maximuma nem

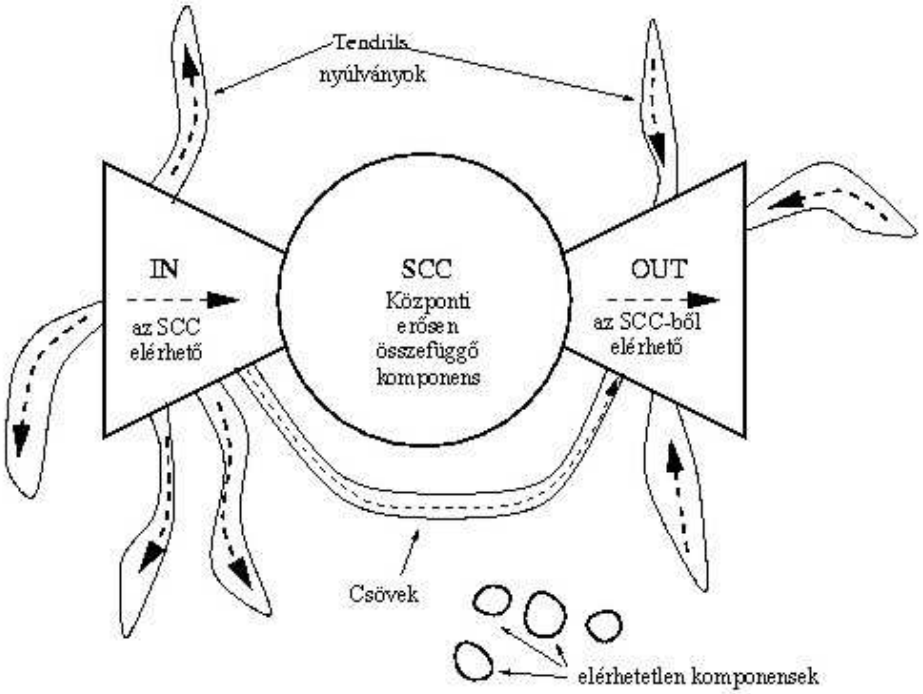
feltétlenül jellemzi jól az eloszlást. Éppen ezért Albert és Barabási a mért távolságok átlagát tekintette átmérőnek. Sajnos azonban ezzel is megmarad az a probléma, hogy ha az összekötetlen csomópontok távolságát ∞-nek tekintjük, akkor az átlag szintén végtelen lesz.

Ennél is pontosabb definíciókat javasol Border. Gráfelméletben irányított gráfok esetén definiált fogalom a gráf *erős komponense*. Egy erős komponenst alkot olyan csomópontok halmaza, melyen melyek közül bármelyik pontból létezik irányított útvonal bármelyik másikba. Egy irányított gráfnak lehet egy vagy több erős komponense. Erős komponensen belül már mindig jól definiáltak és véges gráf esetén véges eredményt adnak a fenti átmérő definíciók. Másik fontos fogalom az irányított gráfok elméletben a gráf *gyenge komponense*. Ha az irányított gráfot irányítatlanként vizsgáljuk, kapcsolatait megtartva, definiálhatjuk az irányítatlan gráf komponenseit, mint olyan csúcsok halmazát, melyben bármely két csúcs között létezik út.

Ha a világháló hivatkozásait irányítatlanként tekintve vizsgáljuk a kialakuló gráf tulajdonságait, Border 200 millió oldalra kiterjedő kísérletei alapján a kapott gráf több mint 90 százaléka egyetlen hatalmas komponenshez tartozott. Ha azonban figyelembe vesszük a gráf irányított jellegét szerkezete ennél sokkal gazdagabb. A vizsgálatba bevont 203 millió oldal természetes módon négy, körülbelül azonos méretű osztályra bomlott. A háló központja egy viszonylag kicsi 56 millió oldalt tartalmazó erős komponens (SCC) volt, mely meghatározó szerepet játszik a gráf bejárhatósága szempontjából. Ehhez kapcsolódtak olyan csúcspontok, melyekről nem érhető el az SCC, fordított irányba viszont létezik útvonal (OUT). A világhálón ezek az oldalak többnyire vállalati siteok

oldalai, melyek elérhetőek ugyan a világhálóról, viszont nem tartalmaznak visszafelé mutató hivatkozásokat. Vannak olyan, többnyire új, oldalak, melyekről elérhető az SCC viszont ők nem érhetőek el az SCC-ről kiindulva (IN). Vannak továbbá olyan oldalak, melyek nem érhetőek el az SCC-ből, és róluk sem érhető el az SCC (TENDRILS). Az utóbbi három osztály mérete körülbelül 44 millió oldal volt a kísérleti mintában.

Ezek után az SCC átmérője (maximális távolságon alapuló definíciót alkalmazva) méréseik szerint 28 volt, míg ugyanez a szám a világháló egészére legalább 500. Megfigyelték azt is, hogy véletlenszerűen kiválasztva egy rendezett oldalpárt (a, b), csupán 24% az esélye annak, hogy az irányított hivatkozásokat követve létezik útvonal *a*-ból *b*-be. Ha azonban létezik ilyen útvonal, az átlagos távolság körülbelül 16. A világháló struktúrája tapasztalataink szerint a fenti az 1. ábrával jellemezhető. A fent említett vizsgálati módszerek és a kísérletek eredményei részletes kifejtésre kerül [20]-ban.



1. ábra

Border több mint 200 millió dokumentumra kiterjedő mérései alapján a világháló, mint irányított gráf szerkezete ezzel az úgynevezett "csokornyakkendő modellel" jellemezhető. A legnagyobb erősen összefüggő komponens meghatározó szerepet tölt be a gráf bejárhatóságában

Mivel barangolóin csak az irányított éleket követve tudnak haladni a gráfon, ez az a modell, mely számunkra releváns volt.

2.2 Skála-invariáns hálózatok dinamikus modelljei

Felvetődött a kérdés, milyen jelenségek hatására alakultak ki az Internethez hasonló skála-invariáns hálózatok, és mivel magyarázhatóak az exponensekben megfigyelhető eltérések. A megfigyelhető hálózatok két nagyon fontos tulajdonságát kell

megemlítenünk. Az egyik ilyen tulajdonság, hogy a csúcsok száma nem állandó, hanem folyamatosan nő. A világháló dokumentumainak száma naponta több millióval nő, illetve néhány százezer oldal naponta eltűnik. Másrészt a valódi hálózatoknak jellemző tulajdonsága a *preferential attachement*, az, hogy egy csúcs mely csúcsokhoz kapcsolódik, jelentős mértékben függ az illető oldal éleinek számától. A világhálón megjelenő új oldal nagy valószínűséggel kapcsolódik olyan oldalakhoz, melyekre már sokan mutatnak. Ezt figyelembe véve tegyük fel, hogy egy megjelenő új csúcs a következő valószínűséggel kapcsolódik az $i.$ csúcshoz:

$$\Pr(i) = \frac{k_i}{\sum_{j=1}^{N} k_j}$$

Ekkor kis számú csúcsból kiindulva folyamatosan adunk gráfhoz új csúcsokat, ezzel az eloszlással kötve őket m darab csúcshoz (m lesz a kapcsolatok átlagos száma). A kialakuló gráf kapcsolatainak eloszlása hatványfüggvény lesz $\gamma = 3$ kitevővel n értékétől függetlenül. Drogovtsev, Mendes és Samukhin ezt a modellt kiegészítette úgy, hogy bevezetett egy kezdeti vonzóképességet, ami miatt bármely csúcs véges eséllyel rendelkezik arra, hogy a megjelenő csúcs hozzá kapcsolódjon. Ekkor:

$$\Pr(i) \propto A + k_i$$

A modell magyarázatot nyújtott arra jelenségre, hogy a valóságban megfigyelhető hálózatok eloszlásaiban megfigyelhető exponensek értéke nem 3, hanem legtöbbször 2 és 3 közötti érték. A modell szerint $\gamma = 2 + \frac{A}{m}$ az eddigi jelöléseket alkalmazva.

A gráf „kis világ” tulajdonsága azt jelenti, hogy átmérője fejlődése során méretével logaritmikusan változott, azaz $l(G) \propto \log(N)$. Az empirikus kísérletek szerint a fenti algoritmusokkal kialakított hálózatokra valóban jellemző a fent leírt egyenlet teljesülése, azaz jól illeszthető a mérési eredményekre az $l = A * \log(N - B) + C$ egyenlet. Számos más modell létezik, melyek magyarázzák a világhálóhoz hasonló tulajdonságokkal rendelkező gráfok kialakulását, a problémával részletesen foglalkozik [12] és [15].

3 Intelligens hálózati barangolók (crawlerek)

Elméleti Háttér

Ebben a fejezetben a dolgozatomban ismertetett módszerek megértéséhez feltétlenül szükséges elméleti alapokról lesz szó.

3.1 Szózsák alapú szövegkategorizáció

Elsőként meg kell határoznunk, hogy a barangolóink hogyan reprezentálják a világhálón megtalálható szöveget tartalmazó HTML dokumentumokat. A reprezentációt végző függvény elkészítése után milyen módszerekkel osztályozhatjuk dokumentumokat, és hogyan dönthetünk azok relevanciáját illetően. Végeredményül kialakul majd a világháló egy modellje, egy olyan állapottér, melyben barangolóink keresést végeznek.

3.1.1 Szövegreprezentáció

Ahhoz, hogy a hagyományos kategorizáló, osztályozó eljárásokat dokumentumokra tudjuk alkalmazni, a dokumentum szövegét valamilyen transzformációval vektor alakra kell hoznunk. A legegyszerűbb eljárás az lenne, ha a szövegben előforduló összes szóhoz egy pozitív egész számot rendelnénk, és e számok vektoraként értelmeznénk a szöveget. Ennek a megközelítésnek a nehézségei nyilvánvalóak: minden dokumentumot változó hosszúságú vektorral reprezentálnánk, a különböző szavak száma is nagyon sok lehet. A megoldás, hogy a szövegekből kiválasztunk n db jellemző szót, ezek alkotják az ún. szózsákot (*bag-of-words*). Ezek után minden dokumentumot leírhatunk egy n dimenziós vektorral, ahol a vektor *i*.-ik eleme azt mondja meg, hogy a dokumentum hányszor tartalmazza a szózsák *i*.-ik elemét (vagy bináris esetben, hogy tartalmazza-e vagy sem). Ez az egyszerű reprezentáció, bár semmit nem mond el a szavak sorrendjéről, kombinálva a szokásos kategorizáló eljárásokkal meglepően jó eredményeket ér el a szokásos szövegkategorizáló feladatokon. Látható, hogy a reprezentáció egyetlen nehéz feladata, eldönteni mi kerüljön a szózsákba. Erre a problémára, amit *feature* kiválasztásnak neveznek, több megoldás is létezik. Én a két legelterjedtebbet ismertetem: ezek a TF-IDF (*term frequency times inverse document frequency*) és a kölcsönös információn alapuló módszerek.

Amikor kiválasztjuk, hogy a tanító példák szavai közül melyek kerüljenek be a szózsákba, három fontos szempontra kell figyelnünk:

- Nagyon gyakori szavak eltávolítása

- Nagyon ritka szavak eltávolítása
- Azoknak a szavaknak a kiválasztása, melyek magas *kölcsönös információval* rendelkeznek a célosztályokra vonatkozóan

Az első két szempont nyilvánvaló, a harmadik viszont magyarázatra szorul. A kölcsönös információ azt méri, hogy mennyben csökkenti egy véletlen változó ismerete egy másik véletlen változó eloszlásának entrópiáját.

$$I(T,\omega) = E(T) - E(T|\omega) = -\sum_{c\in C} \Pr(T(d)=c) * \log(\Pr(T(d)=c)) -$$
$$-\sum_{c\in C} \Pr(T(d)=c,\omega=0) * \log(\Pr(T(d)=c|\omega=0)) -$$
$$-\sum_{c\in C} \Pr(T(d)=c,\omega=1) * \log(\Pr(T(d)=c|\omega=1)) +$$

ahol $\Pr(T(d)=c)$ annak a valószínűsége, hogy a d dokumentuma c osztályba tartozik, $\Pr(T(d)=c,\omega=1)$ pedig annak a valószínűsége, hogy a d dokumentum c osztályba tartozik, és megtalálható benn az ω kifejezés. Hasonlóan értelmezett $\Pr(T(d)=c|\omega=1)$, azzal a különbséggel, hogy itt feltételes valószínűséget számítunk. Ezután az n legnagyobb $I(T,\omega)$ értékkel rendelkező kifejezést kell kiválasztani a szózsák kialakítása során. A módszert részletesebben bemutatja [11] és [21]. Miután a feature kiválasztás megvan, készen állunk rá, hogy a különféle osztályozó eljárásokat alkalmazzuk.

3.1.2 Szövegkategorizáció

A szövegkategorizáció során egy szöveges dokumentumot kell előre rögzített számú osztály egyikébe besorolni. Ehhez rendelkezésünkre áll osztályok egy halmaza C, valamint tanító példák halmaza D, és ismert egy függvény $T: D \to C$, mely

hozzárendeli a tanító halmaz elemeihez a hozzájuk tartozó osztályt. Feladatunk egy olyan $H(d)$ függvény keresése, mely az összes dokumentum terén értelmezett és a lehető legjobban megközelíti a T függvényt abban az értelemben, hogy ismeretlen d dokumentumok esetében maximalizálja annak valószínűségét, hogy a $H(d) \in C$ lesz az osztály, melybe d tartozik. A további fejezetekben bemutatok három elterjedt szövegosztályozó eljárást, további lehetséges módszereket mutat be [10].

3.1.3 TFIDF alapú osztályozók

A TFIDF (*term frequency inverse document frequency*) módszer egy régi, de meglepően jól működő eljárás dokumentumok osztályzására. A fentebb említett szózsák kiválasztása után minden a szózsákban szereplő ω kifejezésre kiszámítjuk az úgynevezett inverz dokumentum gyakoriságot: $IDF(\omega) = \log\left(\frac{|D|}{DF(\omega)}\right)$ ahol $DF(\omega)$ azoknak a dokumentumoknak a száma a tanító halmazban, melyekben szerepel az ω kifejezés, míg $|D|$ az összes dokumentum száma. $TF(\omega, d)$ kifejezés értéke pedig az a szám, ahányszor ω kifejezés szerepel a d dokumentumban. Ezután minden dokumentumhoz kiszámítunk egy vektort, melynek elemei: $d_i = TF(\omega_i, d) * IDF(\omega_i)$. Minden C osztályhoz kiszámítunk egy prototípus vektort, mely: $\vec{c} = \sum_{d \in C} \vec{d}$, majd pedig egy ismeretlen d' dokumentum esetén az osztályozó által választott osztály az lesz, melynek prototípus vektora legkisebb szöget zárja be $\vec{d}'$ vektorral:

$$H(d') = \arg\max_{c \in C} \frac{\vec{d}' * \vec{c}}{\left\|\vec{d}'\right\| * \left\|\vec{c}\right\|}$$

3.1.4 Naïve Bayes módszerek

A Naïve Bayes osztályozók kialakítása során egy olyan feltételezésre kell támaszkodnunk, mely bár a valóságban nem teljesül, a gyakorlatban mégis igen jól működő módszerhez vezet. A feltételezés: a dokumentumok úgy keletkeznek, hogy szavakat generálunk valószínűsége eloszlások alapján. Tegyük fel, hogy van $|C|$ számú különböző valószínűségi eloszlásunk, minden kategóriához egy. Minden osztályozandó dokumentum úgy keletkezett, hogy valamelyik eloszlásból minden szavát független mintavételezéssel nyertük. Ez a modell konzisztens a fentebb leírt szózsák alapú szövegreprezentációval.

Ezek utána a tanítóhalmazt felhasználva könnyen konstruálható olyan Naïve Bayes osztályozó, amely megbecsüli az egyes osztályok valószínűségét, ha adottak a dokumentum *feature* vektorai, azaz olyan vektorok, melyek tartalmazzák, hogy az egyes kifejezések hányszor fordultak elő az adott dokumentumban. A valószínűségek becsléséhez Bayes tételét használjuk, mely szerint annak valószínűsége, hogy egy dokumentum a c. kategóriába tartozik, feltételezve, hogy a *feature* vektora $\vec{x}$:

$$\Pr(C = c|\vec{x}) = \frac{\Pr(\vec{x}|C = c) * \Pr(C = c)}{\sum_{c \in C} \Pr(\vec{x}|C = c) * \Pr(C = c)}$$

ahol $\Pr(\vec{x}|C=c)$ annak a valószínűsége, hogy egy dokumentum $\vec{x}$ szózsák alakú reprezentációval rendelkezik, feltéve, hogy a c osztályba tartozik. Ez a valószínűség nem könnyen becsülhetőek, ha nem teszünk fel bizonyos egyszerűsítő feltételeket. A Naïve Bayes esetben, az $\vec{x}$ *feature* vektor elemeiről tesszük fel, hogy azok kölcsönösen függetlenek, ha adott a kategória. Ez egyszerűsíti a számításokat:

$$\Pr(\vec{x}|C=c)=\prod_k \Pr(x_k|C=c)$$

Ezek után a valószínűségeket a tanító halmazon mért tapasztalati arányokkal helyettesítjük be, majd pedig az osztályozandó dokumentum $\vec{x}$ vektorára kiszámítva az egyenlet jobb oldalát, megkaphatjuk az adott osztályba tartozás valószínűségét.

3.1.5 Valószínűségi TFIDF módszer

Mi ennek a két módszernek a kombinációját alkalmaztuk barangolóinkban. Az eljárás egyrészt valószínűségi alapokon nyugszik, másrészt gyakori szóeloszlások és osztályozási problémák mellett ekvivalens a TFIDF eljárással. A koncepció lényege, hogy meg kell különböztetnünk a szöveget, annak reprezentációjától. Először egy Θ függvény leképezi a szöveget annak reprezentációjára, majd az osztályozó a reprezentáció alapján osztályozza a dokumentumot. A döntés itt is valószínűségi alapokon nyugszik:

$$H_{\Pr TFIDF}(d')=\arg\max_{c\in C}\Pr(c|d',\Theta)$$

Az egyenletből látszik, hogy a valószínűségek explicit függenek a reprezentációtól. A fenti egyenlet jobb oldalán álló kifejezést két részre bonthatjuk:

$$\Pr(c|d',\Theta) = \sum_{x} \Pr(c|x) * \Pr(x|d',\Theta)$$

ez azt jelenti, hogy a keresett valószínűség két valószínűség szorzataként áll elő. $\Pr(c|x)$ annak a valószínűsége, hogy x reprezentációval rendelkező dokumentum a c osztályba tartozik, míg $\Pr(x|d',\Theta)$ azt a valószínűséget jelöli, hogy a Θ leképezés a d' dokumentumhoz x reprezentációt rendeli. Legyen most egy reprezentáció egyetlen ω szó előfordulása egy szövegben. Annak valószínűsége, hogy éppen ω kifejezést figyeljük meg, mint d' reprezentációját:

$$\Pr(\omega|d',\Theta) = \frac{TF(\omega,d')}{\sum_{\omega \in F} TF(\omega,d')}$$

a korábbi jelöléseket használva, F a kiválasztott *feature* halmaz. $\Pr(c|\omega)$ a Bayes tétel segítségével ismét felbontható, $\Pr(\omega|c)$ pedig a tapasztalati előfordulási aránnyal becsülhető:

$$\Pr(\omega|c) = \frac{TF(\omega,c)}{\sum_{\omega \in F} TF(\omega,c)}$$

behelyettesítve az eredeti képletbe, megkaphatjuk az osztályozóra vonatkozó döntési szabályt:

$$H_{\Pr TFIDF}(d') = \arg\max_{c \in C} \sum_{\omega \in F} \frac{\Pr(\omega|c) * \Pr(c)}{\sum_{c' \in C} \Pr(\omega|c') * \Pr(c')} * \frac{TF(\omega,d')}{\sum_{\omega \in F} TF(\omega,d')}$$

Az algoritmus pontosabb elemzése meghaladná a dolgozat kereteit, de megtalálható például [18]-ban.

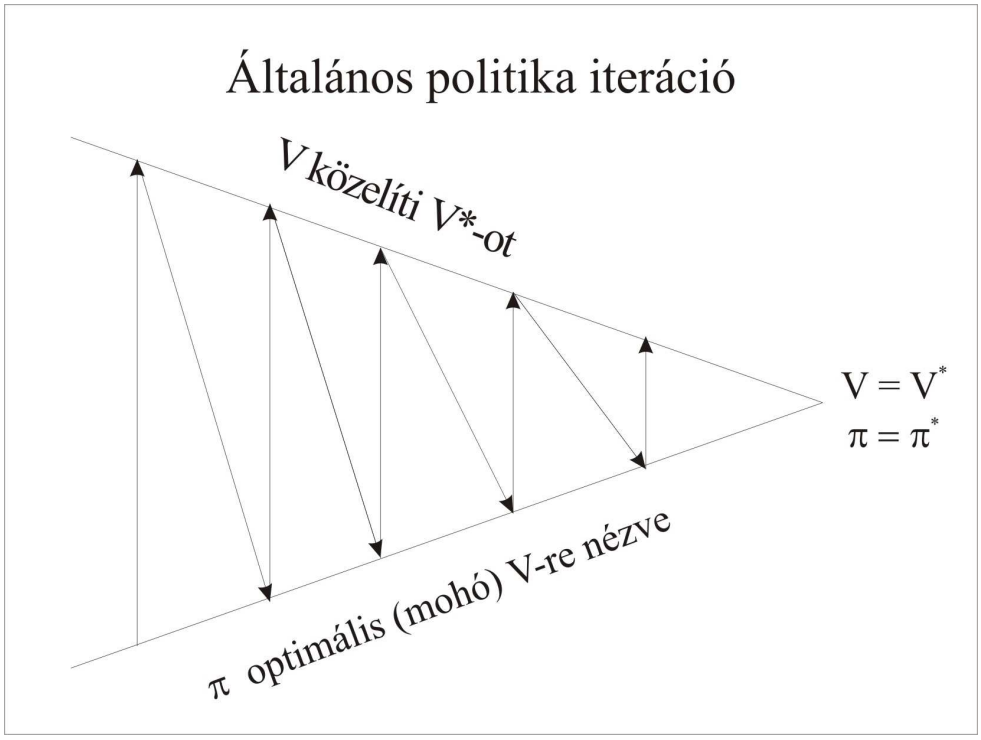
3.2 Megerősítéses tanulás

3.2.1 A megerősítéses tanulás alapfogalmai

A *megerősítés tanulás* (*reinforcement learning, röviden RL*) optimális döntéseket próbál tanulni jutalmazás, vagy büntetés segítségével. Különbözik a felügyelt (*supervised*) tanulástól, mivel a tanító sose mondja meg a helyes akciót egy adott esetre, csupán annyit közöl, hogy a rendszer által választott akció mennyire volt jó vagy rossz. Ezt az ún. *reward* (*jutalom*) segítségével teszi. Egy megerősítéses tanulási feladatot definiálhatunk az állapothalmazzal, $s \in S$, az akciók halmazával, $a \in A$, az állapot-akció átmenet függvénnyel, $T : S \times A \to S$, és a jutalomfüggvénnyel $R : S \times A \to \Re$ Minden lépésben, a tanuló (az ún. ügynök) választ egy akciót, megfigyeli az eredményül kapott jutalmat, és a rendszer átmegy az új állapotba. A megerősítéses tanulás célja, hogy megtanuljunk egy *politikát* (*policy*), egy leképezést az állapotokról az akciókra, $\pi : S \to A$, ami maximalizálja a hosszabb távon nyerhető jutalmat. A hosszú távú jutalom leggyakoribb formalizálása a jutalmak diszkontált összege. A diszkont faktor, $0 \le \gamma \le 1$, az információt fejezi ki, a korábban megkapott jutalmakat értékesebbé téve, mint a később bekövetkezőket.

$$V^{\pi}(s) = \sum_{t=0}^{\infty} \gamma^t r_t$$

ahol r_t a jutalom, amit *t* lépéssel az *s* állapotból történő indulás után kapunk, ha a π politikát követjük. Az optimális politika (π^*) az, amelyik maximalizálja a $V^\pi(s)$értéket minden $s \in S$ állapotra.



2. ábra

Az általános politika - iterálás (GPI) sematikus megjelenítése. A valóságban persze a geometria sokkal bonyolultabb, de általánosan teljesülő feltételek mellett megvalósul a konvergencia

Az optimális politika tanulására az *általános politika - iterálás* (*generalized policy iteration*, GPI) sémáját használhatjuk. Politikaiterálás alatt két együttműködő, szimultán eljárást értünk, az egyik az értékfüggvényt teszi konzisztenssé a jelenlegi politikával (politika kiértékelés), a másik a politikát teszi mohóvá az aktuális értékfüggvénnyel szemben (politika-javítás). A politika iterálásnál, ez a két eljárás váltakozik, mindkettő

befejeződik mielőtt a másik elkezdődne, de ez valójában nem szükséges. Olyan eljárások is vannak, amikben a kiértékelésnek csak egy lépését végezzük el mielőtt a politikát javítanánk, vagy csak egy állapot értékét igazítjuk az egyik eljárásban mielőtt áttérnénk a másikra, de a két eljárás lehet átfedő is. Amíg mindkét eljárás változtatja az összes állapot értékét, addig a végeredmény általában hasonló: konvergencia az optimális értékfüggvényhez (V^*) és az optimális politikához (π^*). Az általános politika iterálás elnevezést a politika kiértékelés és a politika javítás eljárások együttműködésére használjuk, függetlenül a két eljárás váltakozásainak pontos részleteitől. Majdnem minden megerősítéses tanulási eljárás leírható a GPI sémában.

Könnyű látni, hogyha a kiértékelő és a javító eljárások stabilizálódnak, azaz többé nem produkálnak változást, akkor az értékfüggvénynek és a politikának optimálisnak kell lennie. Az értékfüggvény csak akkor lesz stabil, ha konzisztens a politikával, a politika pedig csak akkor stabil, ha mohó a saját értékfüggvényére nézve. Az algoritmus konvergenciájára vonatkozó bizonyítások megtalálhatóak [4] [13]-ban.

3.2.2 Megerősítéses tanuláson alapuló algoritmusok

A kiértékelő és a javító eljárások a GPI-ben egyszerre együttműködők és versenyzők. Versenyeznek abban az értelemben, hogy ellentétes irányba mozognak, hiszen a politika változtatása elrontja az értékelőfüggvényt, és fordítva. Hosszútávon viszont mindkettő az egyetlen közös megoldáshoz tart: az optimális értékfüggvényhez és az optimális politikához.

A valós helyzet geometriája bonyolultabb, de a 2. ábra megmutatja mi is történik valójában. A GPI két eljárása közül a politika javítása triviális. Esetünkben a mohóság azt jelenti, hogy az adott állapotból elérhető akciók közül azt választjuk, amelyik által elérhető állapot $V(s)$ értéke maximális. A gyakorlatban nem mohóvá, hanem ε-mohóvá szokás tenni a politikát, azaz néha, ε valószínűséggel nem mohón dönt, hanem véletlenszerűen. Ilyenkor az ügynök felfedez, olyan állapotokba kerülhet, amikbe a mohó választás használatával sose került volna, pedig lehet, hogy nagyon jók. A mohó választást kizsákmányolásnak is szokták nevezni, hiszen ilyenkor a rendszer a jelenlegi tudása alapján próbálja a maximális jutalmat begyűjteni, míg a felfedező választásoknál, megengedve, hogy rossz helyre kerülhet, megpróbálja környezetét felderíteni. Az ε-t az idővel csökkenteni szokás, hiszen az idő múlásával a környezet egyre nagyobb részét deríti fel az ügynök, ezért egyre jobb lesz az értékelő-függvénye, azt felhasználva egyre több jutalmat gyűjthet. A politika kiértékelés bonyolultabb feladat. Több eljárás alkalmazható rá, pl. dinamikus programozás, Monte Carlo módszerek, TD (*temporal difference*) módszerek. Én ez utóbbit használtam. A levezetések túlmennek jelen dolgozat keretein, de megtalálhatóak például [13] és [3]-ban. Látható, hogy ebben az esetben mindig csak az aktuális állapotot megelőző állapot értékét változtatjuk. A TD(λ) [2] esetben viszont a többi olyan állapotot is változtatjuk, amelyiknek köze volt a jelenlegi állapotba kerülésünkhöz (azaz rajta volt azon az útvonalon, amin idáig jutottunk). Hogy ezt meg tudjuk valósítani, minden állapothoz ún. *eligibility trace*-t (útvonal értéket) rendelünk. Az útvonal értéket egy s állapotra a t időpillanatban $e_t(s) \in \Re^+$ Minden

lépésben az útvonal értékeket $\gamma\lambda$ szorosára csökkentjük, és annak az állapotnak az útvonal értékét, amit éppen látogatunk, eggyel növeljük:

$$e_t(s) = \begin{cases} \lambda * \gamma * e_{t-1}(s) & ha\ s \neq s_t \\ \lambda * \gamma * e_{t-1}(s) + 1 & ha\ s = s_t \end{cases}$$

Minden lépésben kiszámoljuk az ún. TD hibát, ami az állapot értékének becslésénél:

$$\delta_t = r_{t+1} + \gamma * V_t(s_{t+1}) - V_t(s_t)$$

Ezek után az értékfüggvényt a következő módon változtatjuk:

$$V(s) \leftarrow V(s) + \alpha * e_t(s) * \delta_t \quad \forall\ s \in S\ állapotra$$

3.2.3 Függvény-approximátorok megerősítéses tanulásban

Az imént vázolt megközelítés hátránya, hogy csak akkor alkalmazható egyenesen, ha az állapotok diszkrétek és számuk véges. Folytonos esetben, vagy ha az állapotok száma túl nagy, nem tudjuk minden *s* állapothoz tárolni a $V(s)$ értéket. A mi problémánk is ez mindkét csoportba beletartozik, mert dokumentumainkat egy viszonylag magas (50) dimenziós vektorral reprezentáljuk az RL részére, melynek elemei valós értékeket vehetnek fel. Ilyen esetekben a $V(s)$-t parametrikus formában szokás felírni. Ilyenkor $V_t(s)$ egy w_t paraméter vektortól függ, és V_t változtatása w_t változtatását jelenti. Például V_t lehet egy neuronháló által számított függvény, ahol a neuronháló súlyai vannak a w_t vektorban. A súlyok állítgatásával függvények széles tartománya implementálható.

Általánosságban elmondható, hogy a paraméter vektor dimenziója kevesebb, mint az állapotok száma, tehát nem csak az aktuális állapot értéke változik, ha a súlyokat változtatjuk, hanem több más állapoté is. Tehát ilyen értelemben a változásból általánosítunk. Ha a tanulás kiértékelő fázisában az adott állapotból és a hozzá keletkező értékből álló párokat hagyományos tanító példaként fogjuk fel, akkor függvény-approximátorok széles körét alkalmazhatjuk az értékfüggvény becslésére, neuronhálókat, a döntési fákat stb. Nem minden módszer alkalmazható ugyanakkor. Az alkalmazandó függvény-approximátornak képesnek kell online tanulnia, együttműködve a környezetével. Ezen kívül a megerősítéses tanulás olyan függvény-approximátort feltételez, ami képes kezelni az időben változó, nem állandó, célfüggvényeket (lásd GPI). A leggyakrabban alkalmazott tanulómódszerek, amiket a megerősítéses tanulásnál széles körben alkalmaznak, a meredekség csökkentő eljárások. Ezeknek az eljárásoknak az ismertetése és levezetése szintén meghaladja a dolgozat kereteit. Az ismertetett módszerek összefoglalva a 3. ábrán látható algoritmussal írhatók le. A folytonos állapottérben működő megerősítéses tanulási algoritmusokkal foglalkozik részletesen [8] és [9], itt megismerhetők az egyéb választható algoritmusok, valamin azok konvergenciájának feltételei. A megerősítéses tanulás lehetőségeivel és korlátaival foglalkozik ezen kívül [6] és [7].

1. Inicializáljuk w-t és $e(s) = 0 \quad \forall s \in S$
2. **Repeat** (minden epizódra)
   a) $s \leftarrow$ az epizód kezdő állapota
   b) **Repeat** (az epizód minden lépésere)
      I. $s' \leftarrow$ a π politika által választott akció kimenetele
      II. $r \leftarrow$ az s' állapot jutalma
      III. $\delta \leftarrow r + \gamma * V(s') - V(s)$
      IV. $\vec{e} \leftarrow \lambda * \gamma * \vec{e} + \vec{f}(s)$
      V. $\vec{w} \leftarrow \vec{w} + \alpha * \delta * \vec{e}$
      VI. $s \leftarrow s'$
   c) **Until** s befejező állapot

3. ábra

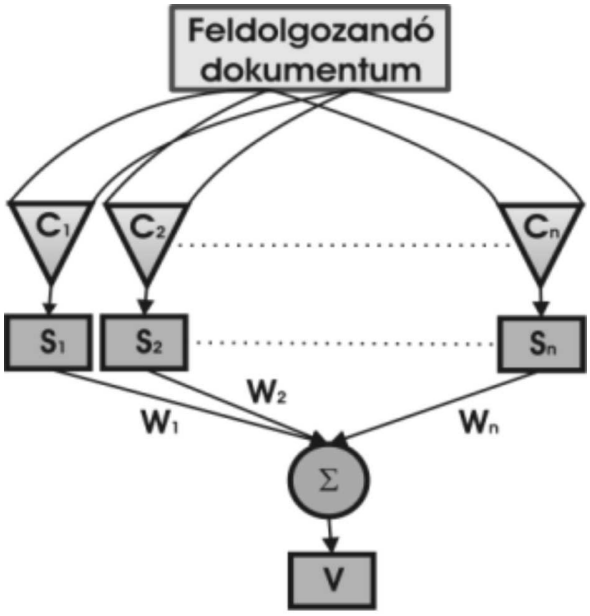
Barangolóinkban működő megerősítéses tanulás algoritmus. A TD algoritmus egy implementációja lineáris függvény-approximátorokat használva, legnagyobb meredekség módszerrel változtatva a függvény paramétereit

3.3 A barangolók működése

A megerősítéses tanulás és a szövegkategorizáció képezik az alapját a barangolók működésének. Egy általános gráfbejáró algoritmus úgy működik, hogy a gráf élei mentén sorra bejárja annak csúcsait, melyeket különböző színekkel színez. Egy csúcs „kiterjesztésén” az azzal szomszédos csúcsok megvizsgálását értjük. A még nem ismert csúcsok színe fehér, azoknak az oldalaknak a színe, melyeket már kiterjesztett az algoritmus, fekete, szürkére színezzük azokat a csúcsokat, melyeket már elért az algoritmus, de még nem terjesztett ki, tehát még van olyan szomszédja, melyet nem vizsgált meg. A szürke színű csúcsok halmaza folyamatosan változik, ez alkotja az ún.

crawl frontier-t. A bejáró algoritmusok abban különböznek, hogy milyen módon választják ki a crawl frontier-hez tartozó csúcsokat kiterjesztésre. Eszerint beszélhetünk például mélységi, vagy szélességi keresésről és ezt figyelembe véve írom le az általunk megvalósított algoritmust.

A barangoló szoftver három alapvető rétegből épült fel. A legalsó réteg biztosítja a kapcsolatot a környezettel, a világháló dokumentumaiból álló gráffal. Ez végzi a dokumentumok letöltését és szövegének kiemelését. A második réteg felelős a kinyert szöveges tartalmon osztályozásáért, melyet korábban már részletesen bemutattam. 50 osztályozót használtunk, melyeket a *geocities* (yahoo) letöltött oldalainak általános témáin tanítottunk be. Az osztályozók eredményei alkották az állapotteret a megerősítéses tanulással foglalkozó fejezetben bemutatott algoritmust alkalmazó értékelő számára, mely lineáris függvény-approximátoron keresztül súlyokat tanult. A jutalmazásról úgy gondoskodtam, hogy az ismert „call for papers” keresési problémával foglalkoztam. Jutalmat akkor kapott a rendszer, ha olyan dokumentumot talált, melyben szerepelt a „call for papers” kifejezés. A barangoló sematikus ábrája 4. Ábrán látható. A barangolók koncepciója egyébként megismerhető [1] valamint [19]-ből.



4. ábra

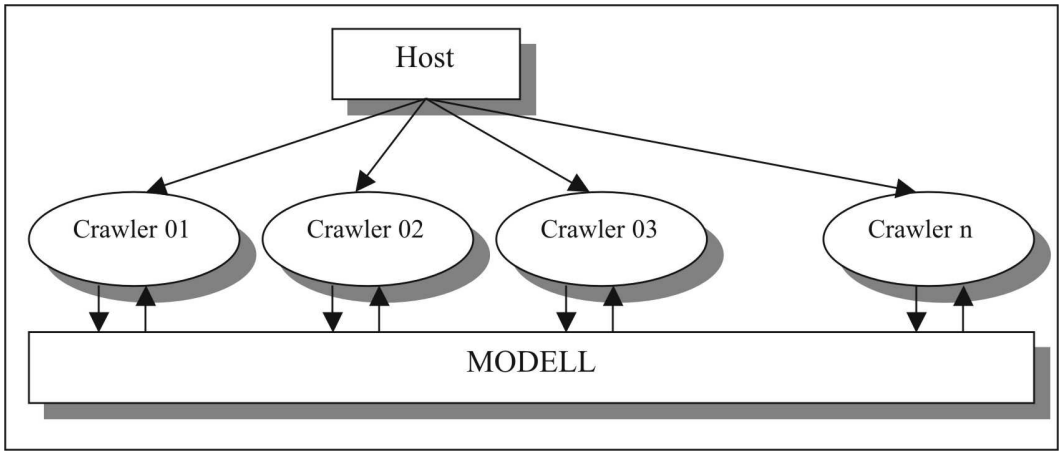
A barangolók működési elvének sematikus ábrája. A barangoló a kiválasztott oldal szöveges tartalmát egy szózsák alapú reprezentációra képezi le, majd pedig 50 darab, különböző témákra betanított osztályozó eredményeit figyelembe véve dönt arról, hogy melyik legyen a következő kiterjesztendő oldal. A *V* érték kiszámításához megerősítéses tanulást és lineáris függvény-approximátort használ *w* súlyokkal.

4 Versengő barangolók szimulációja

A személyes segítő barangoló koncepció [1] [14] [19] kapcsán felmerülő legnagyobb probléma a következő: Amennyiben mindenki ilyen vagy hasonló intelligens segítőket futtat számítógépén, az igen gyorsan túlterheli a hálózati erőforrásokat. Ennek legfőbb oka az, hogy a barangolók egymástól függetlenül működnek, nem befolyásolják egymás viselkedését, nem képesek az együttműködésre, az információk megosztására. Ennek elkerülése érdekében távlati célunk olyan host-barangoló architektúra létrehozása, mely

hatékonyan képes feltérképezni az Internet behatárolt területeit, annak dinamikáját, változékonyságát is figyelembe véve. A barangolók a világháló linkjeit követik, és megerősítéses tanulás segítségével igyekeznek meghatározni, hogy melyik linkeket választva juthatnak releváns oldalakhoz. Az általuk bejárt területről rendelkeznek egy kompakt reprezentációval, melyet képesek megosztani más barangolókkal. A koncepció részletesen megtalálható [14] és [5]-ben.

A host ezzel szemben számon tartja a megszerzett információkat, és jutalmat számít a barangolók számára. Ez a rendszer meglehetősen összetett, ezért nem a világhálón vizsgáltuk működését, hanem modelleztük azt. A MATLAB matematikai modellező rendszer segítségével létrehoztam a kialakítandó rendszer modelljét, ezen vizsgáltam annak viselkedését.



5. Ábra

A megvalósított rendszer sematikus rajza. Egy host rendszer képes barangolók indítására, melyek száma meghatározható (eddigi kísérleteink során ez a szám 2 volt). Szintén a host dönt egy MAXQ algoritmus segítségével arról, hogy mely barangoló juthat erőforrásokhoz, azaz melyik végezheti a keresést.

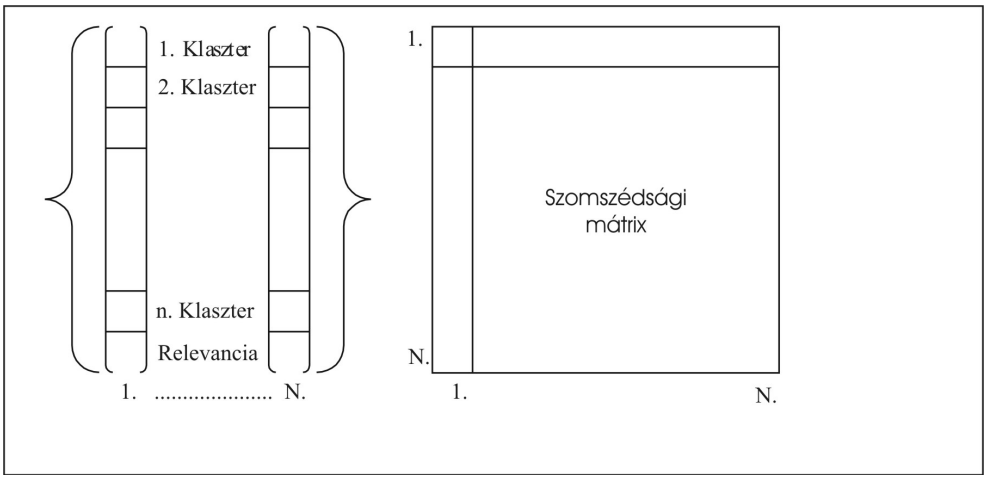
A kialakított rendszer szerkezetét az 5. ábra szemlélteti. A rendszer alapját a világháló egy modellje képezi, mely tartalmaz egyrészt a gráf szerkezetére vonatkozó információkat, másrészt pedig az egyes oldalak tartalmára jellemző állapotvektorokat. A modellek kialakítására több módszer is kínálkozik. Egyrészt lehetséges gráfok generálása valamilyen algoritmussal. Albert és Barabási számos lehetőséget említ, melyekkel a világhálóhoz hasonló gráfok kialakítása lehetséges. Problémát jelent azonban, hogy a tartalmi információk nem tisztázott módon összefügghetnek a kialakuló gráf szerkezetével, és ezen információk generálására nincsenek elfogadott eljárások.

4.1 A modell tulajdonságai

A gráfok elméleti alapokon nyugvó kialakítása helyett inkább a mintavételt választottam. Ehhez rendelkezésemre álltak HTML értelmező, valamint szövegfeldolgozó eszközök JAVA nyelven elkészített implementációi. Ezekre támaszkodva elkészítettem egy olyan programot, mely, és képes volt a világháló meghatározott algráfjainak letöltésére, elemzésére, majd a megszerzett információ tárolására. Ugyanakkor előnyös lett volna, ha a gráf statisztikai tulajdonságai a lehető legjobban megközelítik az egész világhálóra érvényes korábban bemutatott „nyakkendő modell” szerkezetét.

Az algráf kiválasztásának módja alapvetően a barangolók bemutatása során leírt módszernek megfelelő volt. Alaposabb vizsgálatok után azonban kiderült, hogy az alkalmazott algoritmus nagyban meghatározta a letöltött gráf tulajdonságait. A vizsgálatok során 5-10.000 csúcsot tartalmazó gráfokat töltöttem le a világhálóról

breadth first, *RL* valamint egy *korlátozott RL* algoritmussal. Utóbbi lénygében azt jelentette, hogy a barangoló tevékenységét meghatározott számú domainen belül végezhette. Ez egy egyszerű módszer arra, hogy a világháló behatárolt területére korlátozzuk vizsgálatainkat. Az elmentett információt azután a MATLAB rendszerben vizsgáltam tovább. A kialakuló webmodell elemeit a következő ábrán szemléltetem:



6. ábra

A világháló egy letöltött gráfjának modellünk szempontjából érdekes tulajdonságai. A világháló minden dokumentuma a szomszédsági mátrix egy-egy csúcspontja, míg a dokumentumok tartalmát osztályozók értékeinek vektorával reprezentáljuk. A relevanciára vonatkozó információ szintén része a modellnek.

A modell a korábban bemutatott PrTFIDF algoritmus alapján kialakított osztályozók eredményeit tartalmazta minden csomóponthoz hozzárendelve. Tartalmaz még egy relevanciára vonatkozó információt tartalmazó mezőt, szintén minden csomóponthoz, valamint a letöltött gráf szomszédági mátrixát. Van tehát egy gráfunk, melynek csúcsai sokdimenziós vektorok, melyek az adott csúcsponthoz tartozó dokumentum szöveges

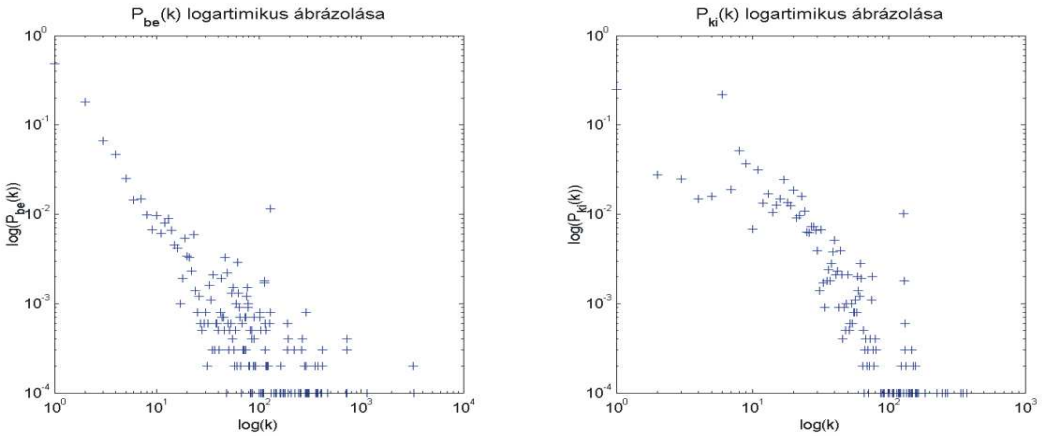
tartalmát reprezentálják. Ezek a vektorok a csúcsokhoz tartozó állapotvektorok, melyekkel már a megerősítéses tanulással foglalkozó fejezetben is megismerkedhettünk.

4.2 A gráfok statisztikai tulajdonságainak vizsgálata

Az azonos kezdőpontból különböző gráf-bejáró algoritmussal letöltött szomszédsági mátrixokat ezután az Internet gráf modelljével foglalkozó fejezetben leírt szempontok szerint alapos vizsgálatnak vetettem alá. Kiszámítottam és ábrázoltam a kapcsolatok számának eloszlásait, a klaszterezettségi hányadost, valamint a világháló nyakkendő modelljének [20] megfelelően megkerestem a gráf erős komponenseit. Kiszámítottam és ábrázoltam ezek eloszlását. Megkerestem a legnagyobb erős komponenst a gráfokban, és ezt tekintve a központi összekötő komponensnek, meghatároztam a többi komponens szerepét a gráfban. Ez az elemzés lehetővé tette, hogy pontos képet alakíthassak ki a gráf bejárhatósága szempontjából. Ennek hiányában nehezen lett volna eldönthető, hogy a barangolók a gráf szerkezetéből adódó okok miatt járnak be adott útvonalakat, vagy tanult viselkedésük eredménye a hatékonyabb működés. Az alábbi ábrákon és táblázatokon láthatóak két általam kiválasztott modell statisztikus szempontú elemzésének eredményei.

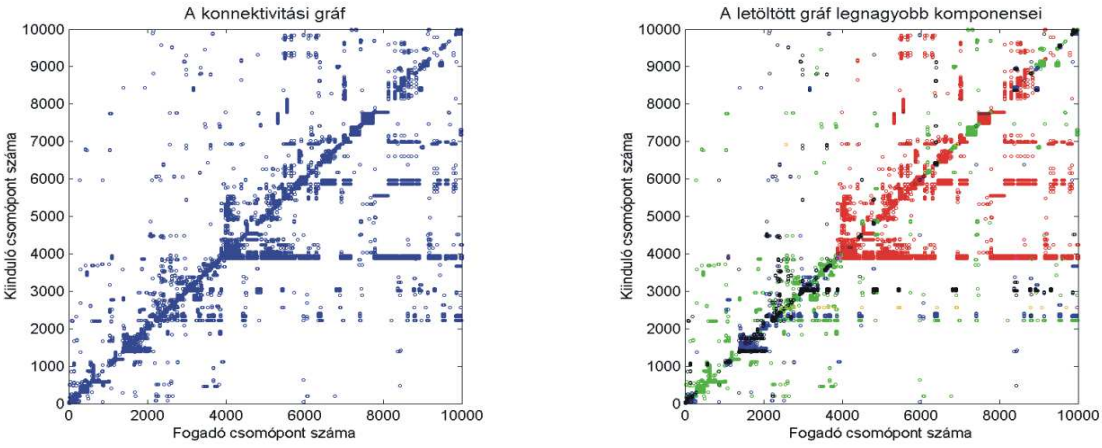
Látható, hogy a világháló skála-invariáns jellege, valamint a mintavétel módja miatt a viszonylag kis méretű letöltött modellek tulajdonságai nagyon hasonlítanak a Border általa elvégzett kísérletekben kapott eredményekhez. A mélységi és szélességi kereséssel végzett mintavétel nem bizonyult használhatónak, mert az így letöltött gráfok

gyakorlatilag fa struktúrával rendelkeztek, így nem voltak bejárhatóak barangolóink számára.



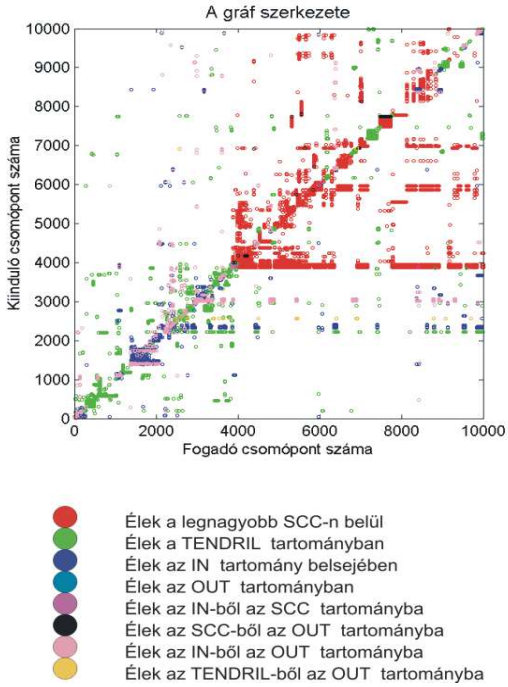
7. Ábra

Az ábra egy 10000 csúcsot tartalmazó, tanuló barangoló általa letöltött webgráf éleinek eloszlását ábrázolja. A tengelyeken az egy csúcshoz tartozó élek számának logaritmusa, valamint az előfordulási gyakoriság logaritmusa található. Jól megfigyelhető a hatványfüggvény jelleg mind az adott csúcsból induló, mind pedig az adott csúcsra mutató élek eloszlásában.



8. ábra

Az előző ábrán elemzett gráf szomszédsági mátrixának grafikus megjelenítése. Jól láthatóak az ábrán a mintavétel során használt barangoló által bejárt siteok mintázatai. Ezek a mintázatok teszik lehetővé jelentős méretű erős komponensek kialakulását a világhálón. Ezt a feltételezést megerősíti a jobb oldalon látható ábra, amelyen a legnagyobb öt komponenst eltérő színekkel jelenítettem meg. Az ábrán megfigyelhető, a barangoló útvonala a főátló mentén.



9. ábra

A gráf szerkezetének elemzése a „nyakkendő modell” [20] alapján. Jól látható a piros színnel jelölt legnagyobb erős komponens, valamint az abból kivezető rózsaszínnek jelölt élek. Futtatásaink során csak az összefüggő komponenseken futottak a barangolóink.

A mintavétel körülményeire és az elemzés eredményeire vonatkozó információkat röviden az 1. táblázatban foglaltam össze.

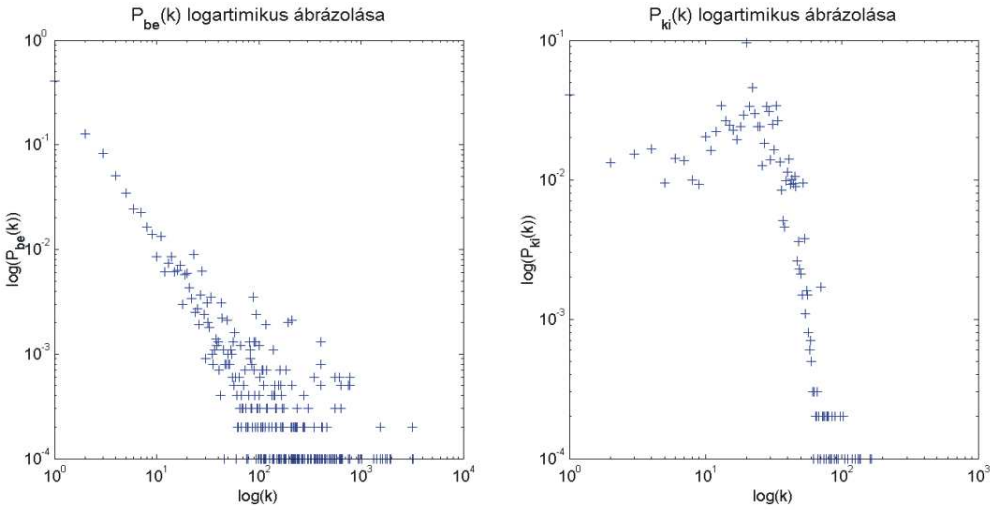
1. táblázat

A fent elemzett modell pontos tulajdonságai, valamint az elemzés eredményeként kapott információk. A klaszterezettség átlagos értéke feltűnően nagy. A mintavétel során egyedi jellegzetesség az OUT komponens kis mérete, ennek oka - ami az ábrán is megfigyelhető - hogy az utolsóként vizsgált site volt a legnagyobb

A MODELL TULAJDONSÁGAI						
Kiinduló URL	Keresett kifejezés	C	IN	SCC	OUT	TENDRILS
http://www.english.upenn.edu/CFP/	„call for papers”	0.65944	1568	4325	17	4072

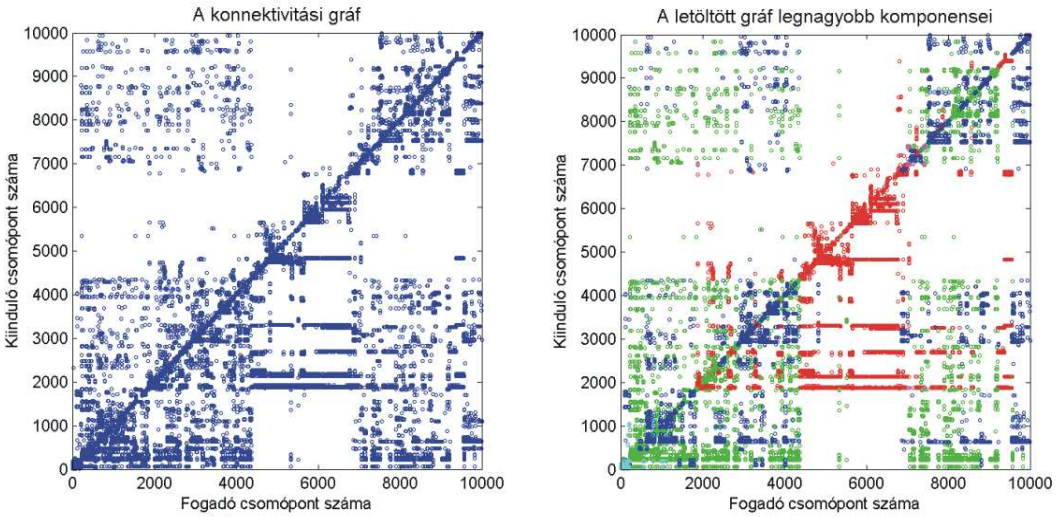
Az elemzésekből kiderült, hogy az RL módszerrel kiválasztott gráf skála - invariáns módon tükrözi az egész világháló statisztikus tulajdonságait. Az egy csúcshoz tartozó élek eloszlása megközelítően hatványfüggvény alakú a nagy élszámok tartományában, akár a befutó, akár a kifelé mutató élek megoszlását vizsgáljuk. A klaszterezettség igen magas értéket mutat, és a gráfban nagy méretű erős komponenseket találhatunk, melyek alkalmasak az általunk tervezett kísérletek elvégzésére. A komponensek mérete szintén hatványfüggvény-szerű eloszlást mutat. A világháló nyakkendő modelljét szintén jól tükrözi az általunk vizsgált kis méretű minta.

A korlátozott RL algoritmussal vett minták szintén számunkra érdekes statisztikai tulajdonságokkal rendelkeztek:



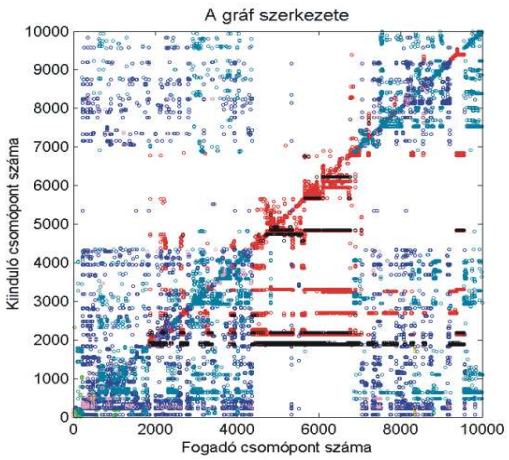
10. ábra

Az ábrán egy az előzővel azonos méretű, korlátozott tanuló barangoló általa letöltött webgráf éleinek eloszlása hasonlít a 7. ábrán megfigyelhető eloszláshoz. Jelentősen eltér viszont az adott oldalról induló kapcsolatok eloszlásában. Ennek oka, hogy a site egyedi statisztikái határozzák meg a végeredményt.



11. ábra

A gráf szomszédsági mátrixának grafikus megjelenítése. Szintén megfigyelhetőek rajta a siteokra jellemző mintázatok. A legnagyobb komponenseket eltérő színekkel megjelenítve látható hogy a barangolónak nem volt lehetősége új siteok vizsgálatára, ezért a barangoló útja mentén egyenletesebben oszlanak el a vizsgált siteok.



12. ábra

A gráf szerkezetének elemzése a „nyakkendő modell” alapján. Jól látható a piros színnel jelölt legnagyobb erős komponens. Site-specifikus tulajdonsági miatt végül nem használtuk fel a futtatások során ezeket a modelleket.

A mintavétel körülményeit, valamint az elemzés legfontosabb információit a 2. táblázatban foglaltam össze. Ebben az esetben a korlátozott mozgástér miatt a kiindulási pontnak nagy befolyása van a kialakuló modell tulajdonságaira. Végül futtatásaink során nem használtuk ezeket a gráfokat, mert a világháló sitejainak lokális szerkezete tükröződik a gráf tulajdonságaiban.

2. táblázat

Erre a gráfra is jellemző a magas klaszterezettség. Itt a talált komponensek megoszlása sokkal egyenletesebb. A TENDRIL komponens kis mérete annak köszönhető, hogy a barangoló a számára engedélyezett 50 site-on megtalálta az integritást biztosító nagy konnektivitású oldalakat.

A MODELL TULAJDONSÁGAI						
Kiinduló URL	Keresett kifejezés	C	IN	SCC	OUT	TENDRILS
http://www.isqed.org/	„call for papers”	0.53082	2832	3962	2859	347

Kedvező statisztikai tulajdonságai miatt végül az RL algoritmussal letöltött gráfokat választottam ki a rendszer vizsgálatához.

4.3 Dinamikus webmodellek

A korábbi fejezetekben bemutatott gráf modellezte rendszerünk szempontjából a világháló egy bejárható, behatárolt területét. A kialakuló gráf fejlődésének szimulációjára számos lehetőség kínálkozik, melyek megismerhetőek [15] [12] -ből. Alapvető célunk új információ keresése volt a világhálón, tehát változó gráfra volt szükségünk. Ennek érdekében bármely eljárás alkalmazható dinamikus webmodellekkel foglalkozó fejezetben leírt módszerek közül. A mi modellünk azonban a gráf szomszédsági

viszonyain kívül a dokumentumok tartalmára vonatkozó további információkat tartalmaz, ami megsokszorozza a dinamika kialakítására vonatkozó lehetőségek számát.

1. A legegyszerűbb, általunk is megvalósított megoldás valamilyen zajszerű állapotváltozás bevezetése a gráf csomópontjain. Ez érintheti azok tartalmát, relevanciáját, illetve szomszédsági viszonyait. Ekkor rendszerünk viselkedését zajos környezetben vizsgáljuk.
2. A csúcsok állapotvektorait tetszőleges mintákat követve is változathatjuk
3. Kölcsönhatásokat definiálhatunk a gráfon az egyes csúcspontok állapotvektorai között, ami magában foglalhatja akár a szomszédsági viszonyok megváltoztatását is.

Az előző fejezetben bemutatott és elemzett modelleket megváltoztattam oly módon, hogy a barangolók csak a modell legnagyobb erős komponensén dolgozhattak. Így bármely barangoló a gráf bármely csúcsából bármely másikba eljuthatott. Emellett a relevanciára vonatkozó információkat is megváltoztattam. Kiszámítottam az eredetileg is releváns csúcspontok állapotvektorainak átlagát ($\vec{R}$). A csúcspontok egy előre meghatározott arányán alakítottam ki relevanciát oly módon, hogy annak valószínűsége, hogy k csúcs releváns lesz:

$$P(k) = \frac{\cos\left(\vec{s}(k), \vec{R}\right)}{\sum_{i=1}^{N} \cos\left(\vec{s}(i), \vec{R}\right)}$$

tehát annak valószínűsége, hogy egy csúcspont relevanciát nyert arányos volt annak állapotvektora és a releváns oldalak átlagos állapotvektora által bezárt szög

koszinuszával. A futtatások során a releváns oldalak paraméterben rögzített aránya frissült – a barangolók és a host részére ismét jutalmat jelentett – miután a barangolók a modell szintén paraméterben rögzített arányát megvizsgálták. Ezzel a módszerrel modelleztük az újdonság megjelenését a rendszerben.

A modell felett kialakítottam egy versengő barangolókból álló populációt, melyet egy host irányított. Célunk egyrészt az volt, hogy megvizsgáljuk, mennyire hatékonyan képes egy ilyen rendszer az információk begyűjtésére a gráf egy behatárolt területéről, másrészt pedig meg akartuk vizsgálni, hogy a megerősítéses tanulást alkalmazva a barangolók képesek-e spontán felosztani a rendelkezésükre álló területet. A hatékonyság növelése érdekében a host egy folyamatosan figyelemmel kíséri barangolói eredményeit, majd *MAXQ* algoritmus alkalmazásával választ közülük. A *k*. barangoló a következő valószínűséggel kapta meg a jogot, hogy a következő időszeletben futhasson:

$$p_k^t = \exp\left(\frac{H_k^t}{T_t}\right) \quad ahol \quad H_k^t = \sum_{i=1..t} \gamma^{t-i} * \frac{rel_i}{all_i}$$

ahol rel_i a barangoló által az *i*. körben begyűjtött releváns oldalak száma, míg all_i az ugyanabban a körben letöltött oldalak száma. γ az úgynevezett diszkont faktor, mely meghatározza, hogy a host mennyivel kevesebbre értékelje a múltbeli hatékonyságot a jelenleginél. Ez a választási eljárás biztosíthatta, hogy a hatékonyabb barangolók több erőforráshoz jussanak, ezáltal végeredményben javulhasson az egész rendszer hatékonysága. A host feladata ezen kívül a barangolókban működő RL algoritmus számára szükséges jutalom kiszámítása is. Egy releváns csúcspont megtalálásáért csak az a barangoló kaphatott jutalmat, amely elsőként találta azt meg. Hasonló rendszert

alakított ki Menczer, melyet leír például [16]-ban. Ebből a forrásból megismerhetjük a barangoló architektúrák minősítésének problémáit, valamint legelterjedtebb módszereit.

Mivel szimulációink során elsősorban arra voltunk kíváncsiak, hogy a barangolók képesek-e saját szegmensek kijelölésére a gráfon, a host nem alkalmazott valódi szelekciót. A fenti képletben szereplő hőmérséklet paraméter elegendően nagy volt ahhoz, hogy a barangolók közelítőleg azonos valószínűséggel juthassanak erőforrásokhoz.

4.4 Barangolóink speciális tulajdonságai

Az általunk elkészített barangolók kis mértékben különböznek a gyakorlatban működő és az elméletben leírt barangoló architektúráktól. A gyakorlatban ugyanis a barangolók végtelen memóriával rendelkeznek. Céljuk a világháló egy szegmensének feltérképezése, és az ott található információ indexelése kulcsszavak alapján. Az egyszer már felkeresett csúcspontokat nem látogatják meg újra. Mi ezzel szemben a világhálón bolyongó ügynököket akartunk vizsgálni, akiknek végesek, sőt erősen korlátozottak az erőforrásai. Ennek megfelelően a barangolók véges méretű memóriával rendelkeznek, így nem várható el hogy rövid időn belül bejárják az egész gráfot. A memória mérete egy paraméterben volt megválasztható. Ennek értéke szabta meg, hogy a barangoló hány meglátogatott csúcspontra és azok környezetére, mint kiterjeszthető csúcsokra emlékezett.

4.5 A véges memória következményei

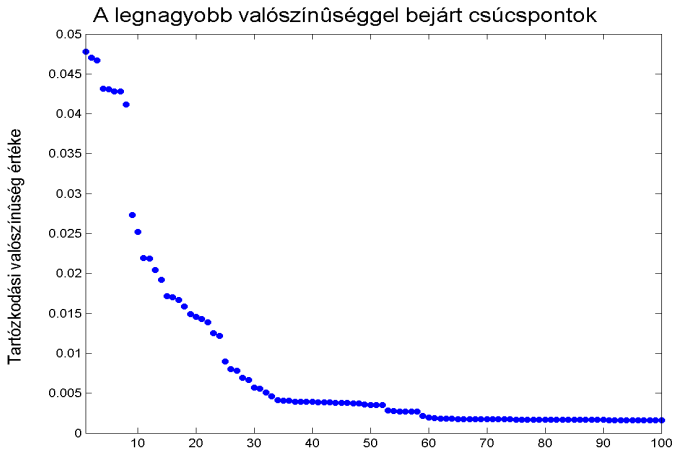
A memória mérete határozza meg, hogy a barangoló milyen valószínűséggel mely csúcson található adott időpontban. A kialakuló valószínűség eloszlást a Markov láncok elméleti modelljét alkalmazva kaphatjuk. Ezt a módszert alkalmazza többek között a Google PageRank algoritmusa, vagy Kleinberg Hub-Authority modellje.

A modell alapja, hogy egy diszkrét állapottérben az állapotok között normált átmeneti valószínűségi mátrixot definiálunk $\left(P, \quad p_{i,j} = \Pr(s_i \to s_j)\right)$. Célunk az, hogy kiszámítsuk, hogy egy folyamat során, melyben a fenti valószínűséggel történnek állapotváltozások, az állapotok egy kezdeti normált eloszlásából (x) milyen stacionárius eloszlás alakul ki. Az eloszlás időfüggését a következő egyenlet írja le:

$$\vec{x}_{t+1} = \vec{x}_t * P$$

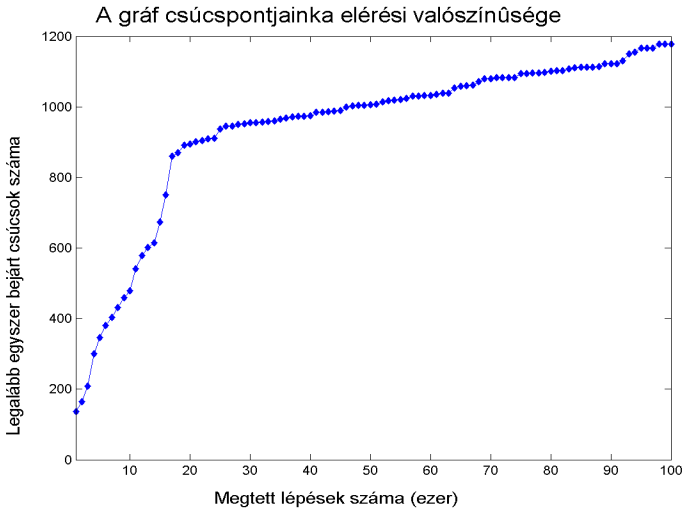
A kialakuló stacionárius eloszlás esetén tehát teljesülnie kell az $\vec{x} = \vec{x} * P$ sajátérték egyenletnek. Amennyiben az átmeneti valószínűségek mátrixa ergodikus, azaz bármelyik állapot elérhető bármelyik másikból nem nulla valószínűséggel, és minden időpontban minden állapotban nem nulla valószínűséggel tartózkodhat a rendszer, akkor P a következő tulajdonságokkal rendelkezik: Pontosan egy olyan sajátvektora van, melynek minden eleme azonos előjelű tehát valószínűségi eloszlásként értelmezhető. Ez felel meg a stacioner eloszlásnak és az ehhez tartozó sajátérték 1. A stacioner eloszlás ismeretében már megbecsülhető, hogy adott idő alatt az állapottér mekkora hányadát járja be a rendszer.

A barangolókra értelmezve a fenti modellt a legegyszerűbb esetben memóriájuk értéke 0, azaz a barangoló csak a jelenlegi állapotát érzékeli, és véletlenszerűen bolyong a gráfon. Ebben az esetben a szomszédsági mátix oszlopait normálva egyszerűen megkaphatjuk az átmeneti valószínűségek mátrixát és kiszámíthatjuk a stacioner eloszlást. Az eljárás nagyon hasonlít a PageRank algoritmusra. Az Internethez hasonló hatványfüggvény alakú kapcsolati eloszlást mutató gráfokon azonban a nagy konnektivitású állapotokban nagy valószínűséggel található meg a rendszer, míg az alacsony konnektivitású távoli állapotok elérési valószínűsége gyakorlatilag 0, így a barangoló a gráf kis területére korlátozódik. A jelenség pontos vizsgálata céljából elkészítettem az egyik gyakran felhasznált webmodellünk numerikus elemzését abból a szempontból, hogy hogyan alakulnak a stacioner megtalálási valószínűségek az egyes csúcspontokon, és ennek milyen hatása van a gráf bejárhatóságára véletlen bolyongást feltételezve.



13. ábra

A stacioner tartózkodási valószínűségek alakulása a gráfon végzett véletlen bolyongást feltételezve csökkenő sorrendbe állítva az értékeket. A 4325-ből csak a legnagyobb 100 értéket ábrázoltam, hogy az ábrán jobban megfigyelhető legyen a nagy valószínűséggel elérhető csúcspontok aránya.



14. ábra

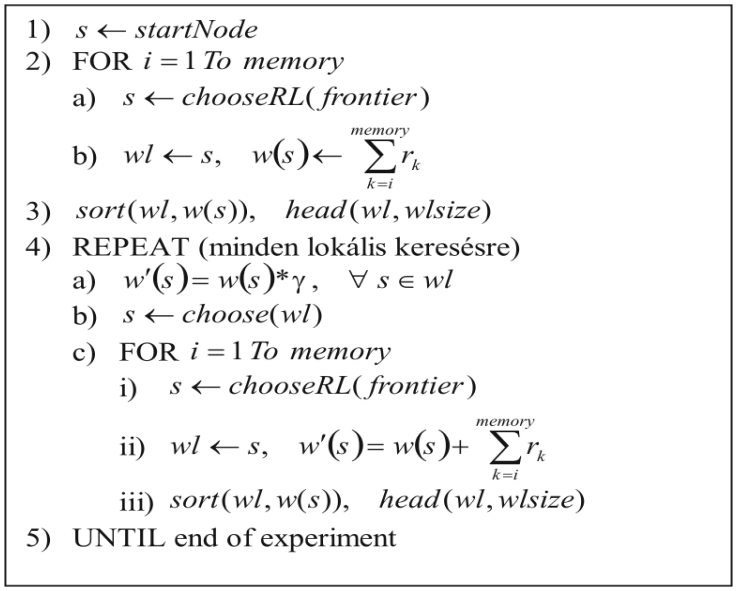
A gráfon a stacioner tartózkodási valószínűséget feltételezve a $P = 0.5$ valószínűséggel legalább egyszer elért csúcspontok számának alakulása a megtett lépések számának függvényében véletlen bolyongást feltételezve. Vegyük figyelembe, hogy a gráf mérete 4325 csúcspont!

Az eredményekből jól látható, hogy a világhálóra jellemző hatványfüggvény-szerű kapcsolati eloszlást mutató gráfok véletlenszerűen bolyongó ügynök számára gyakorlatilag nagyon nehezen bejárhatóak. A stacioner tartózkodási valószínűségek egyenetlenül oszlanak meg, ennek következtében egy bolyongó ügynök a csúcspontok nagy hányadát nem elhanyagolható valószínűséggel csupán több százezer lépésben éreti el egy csupán néhány ezer csúcsot tartalmazó gráfon. A világhálóhoz hasonló gráfokon megvalósuló keresés problémáival foglalkozik [17].

4.6 Weblogok kialakítása

A szimulációkhoz implementáltam a barangolókkal foglalkozó architektúrát a MATLAB matematikai szimulációs rendszerben és ezeket a barangolókat a fent bemutatott modellen vizsgáltam.

Vizsgálatink során tehát a barangolók meghatározott frissítési rátával változó környezettel szembesülnek, mely ráadásul egy „kis világ” tulajdonsággal rendelkező gráfon alakul ki. Az egész rendszer hatékonyságát nagyban növeli, ha a párhuzamosan működő barangolók képesek felosztani egymás között a rendelkezésükre álló bejárható gráfot. Ahhoz, hogy ez a viselkedés kialakulhasson szükséges, hogy a barangolók értékeljék a gráf csúcsait abból a szempontból, hogy mennyire fontos kiindulási pontot jelentenek számukra a jutalom begyűjtése során. Ezt az értékelést a barangolók weblogok kialakításával és tanulásával valósítják meg. A weblogok kialakításának pontos algoritmusa a 15. ábrán látható.

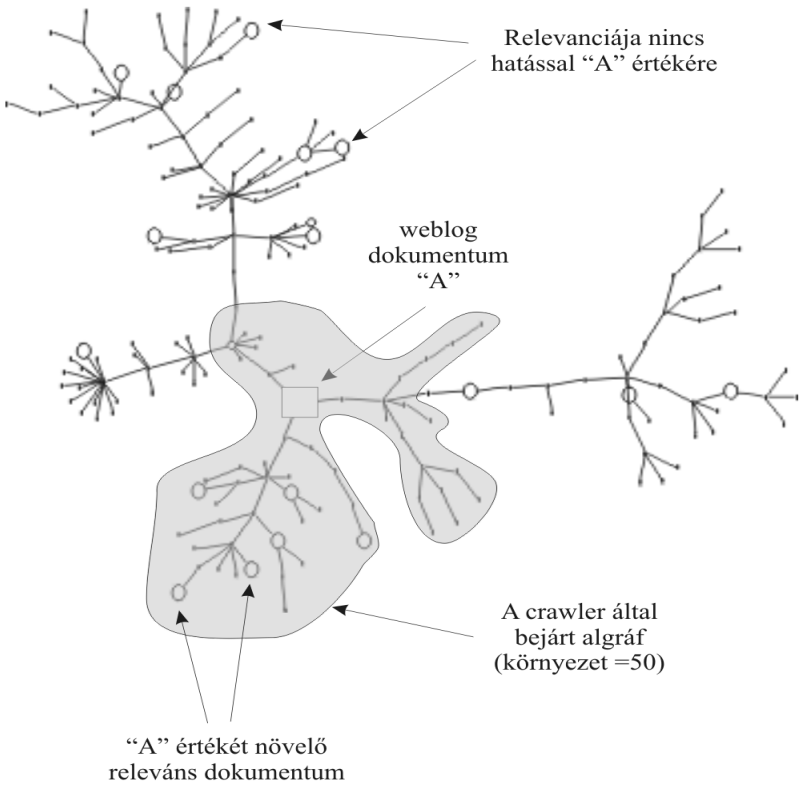


15. ábra

A weblogok kialakításának algoritmusa. Az algoritmusnak három lényeges paramétere van. Az első a *memory* ami annak a környezetnek a méretét határozza meg, melyet egy csúcspont értékelése során vizsgál a barangoló. A másik nagyon lényeges paraméter a *wlsize*, ez határozza meg a weblog méretét. A harmadik a γ, diszkont faktor, mely azt mutatja, mekkora a hatása a múltbeli eredményeknek a weblog értékekre.

Miután inicializáltuk a barangoló paramétereit, elindítjuk, és a *memory* paraméterben meghatározott lépésen keresztül engedjük futni. Eközben minden kiválasztott oldalt elhelyezünk a weblogban. A ciklus végén kiszámítjuk a weblogba került csúcspontok értékét, mint a belőlük induló kutatás kumulált jutalmát. Ezután értékeik szerint sorba rendezzük weblog dokumentumait, és csak a legmagasabb értékkel rendelkező, *wlsize*, darab csúcsot tartunk meg. Ezután bekerülve a fő ciklusba, egy γ paraméterrel diszkontáljuk a meglevő $w(s)$ értékeket, majd a weblogból újabb kiindulási csúcsot választunk, és megismételjük a korábban leírt lépéseket.

A 14. ábrán szemléltetem a barangolók működését egy gráfon. Egy adott, a weblogban megtalálható csúcspontból kiindulva a barangoló megtesz 50 lépést, azaz kiválaszt 50 csúcspontot kiterjesztésre. Ebben az esetben a webloghoz tartozó elem környezetének nagysága tehát 50 volt. Eddigi kísérleteink során a barangoló memóriájának mérete mindig pontosan megegyezett az itt ábrázolt környezet nagyságával, azonban annál kisebbre is beállítható. Az ábrán szürkével jelölt megvizsgált környezeten belül elhelyezkedő releváns csúcspontok megtalálása miatt kapott jutalmak hozzájárulnak a központi csúcspont $w(A)$ értékéhez. A ciklus során kiválasztott bármely csúcspont bekerülhet a weblogba, amennyiben értékes kiindulópontot jelent. Ha több barangoló közeli csúcspontokat választ weblogjaiba, párhuzamosan vizsgálják az adott területet, így végül valamelyikük weblogjából a szóban forgó csúcspont eltűnik. Ez a mechanizmus biztosítja, hogy a barangolók képesek hatékonyan felosztani az általuk vizsgált gráfot. A weblogokra vonatkozó három lényeges paraméter tehát magának a weblognak a mérete, valamint annak a környezetnek a mérete, ami egy elemhez tartozik, és a γ diszkont faktor. Ezek közül mi az első két paraméter mentén vizsgáltuk a rendszer viselkedését.



16. ábra

A weblogok kialakítása során egy „A” dokumentum értékét a következő határozta meg: „A”-ról kiindulva a barangoló egy paraméterben rögzített lépés megtételét követően hány releváns csúcsot talált. A weblogba bármely, a ciklus során kiválasztott dokumentum hasonló szempontok alapján kerülhetett be.

4.7 Kísérleti eredmények

Úgy készítettem el a szimulációs szoftvert, hogy a barangolók és a host működésével kapcsolatos minden információt tároljon. Rögzítettem minden barangoló útvonalát, az egyes barangolók által felfedezett releváns csúcspontokat, a host választásai a rendelkezésére álló barangolók között, valamint az egyes barangolók hatékonyságát jelző mutatók alakulását. Bevezettem egy általános idő paramétert, aminek mértékegysége a barangolók lépésein alapult, annak többszöröse volt. A barangolók egy paraméterben meghatározott lépésszámon keresztül kaptak lehetőséget a hosttól a kutatásra, majd a host dönthetett, hogy melyik barangolója végezze a következő ciklust. Egy ilyen ciklus képezte az idő mértékegységét rendszerünk szempontjából. A szoftver szintén tárolta, hogy melyik csúcspont melyik időpontban frissült.

Az így kialakuló nagy méretű adathalmazból ábrákat készítettünk, melyek jellemzik a rendszer viselkedését az általunk vizsgált szempontokból. Ezek a szempontok:

1. Új információ gyors felfedezése a gráfon.
2. Szegmentáció kialakulása, a rendelkezésre álló terület hatékony felosztása
3. Eredményes keresés egy adott témában a gráf által kijelölt állapotterében

A vizsgálataink során elkészített nagy számú ábratípusból csak azokat mutatom itt be, melyek valóban jól szemléltetik a fenti funkciók megvalósulásának alakulását a kísérletek során. A rendszer nagyon sok paraméterrel rendelkezett, melyek párhuzamos változtatása sok futtatást igényelt volna. Ezért a paraméterek nagy részét rögzítettük olyan értékekre, melyek nagy valószínűséggel nem befolyásolják érdemben vizsgálatinkat.

A kísérletek során rögzítettük a gráf frissülésére vonatkozó mindkét paramétert: a frissítésre akkor került sor, amikor a barangolók megvizsgálták a gráf 2%-át, és ekkor a releváns dokumentumok 1,5%-át frissítettük. Ezek a számok alapvetően meghatározták a barangolók eredményességi rátáját. A futtatások során a releváns csúcspontok arányát a gráfban kezdetben 10%-ra állítottuk be, a host MAXQ algoritmusában szereplő hőmérséklet paraméter pedig ennek kétszerese volt. Ez biztosította, hogy a barangolók megközelítőleg azonos eséllyel kaptak szerepet.

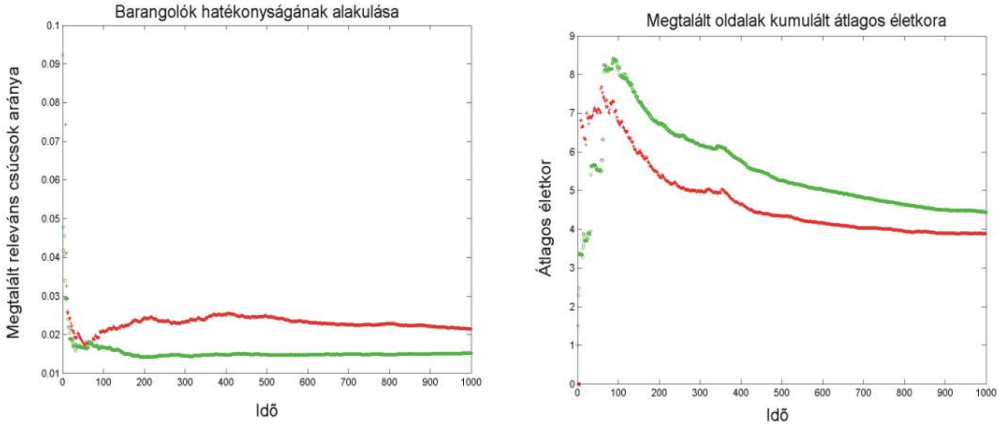
Szintén rögzítettem a barangolókban működő a 3. ábrán részletesen bemutatott RL algoritmus paramétereit. A barangolók ezen a paraméterek ilyen beállítása mellett kísérleteink szerint stabilan és hatékonyan működnek:

$$\alpha = \frac{1}{2 * \dim(state)}, \quad \gamma = 0.9, \quad \lambda = 1$$

A futtatások során mindig 2 barangolót engedélyeztünk a host számára, rögzítettük még a weblogokhoz felhasznált algoritmusban a γ paraméter értékét. Ennek érdekében futtatásokat végeztünk. Kísérleteink alapján ezt a paramétert $\gamma = 0.8$ értékre beállítva a várt viselkedést kaphatjuk.

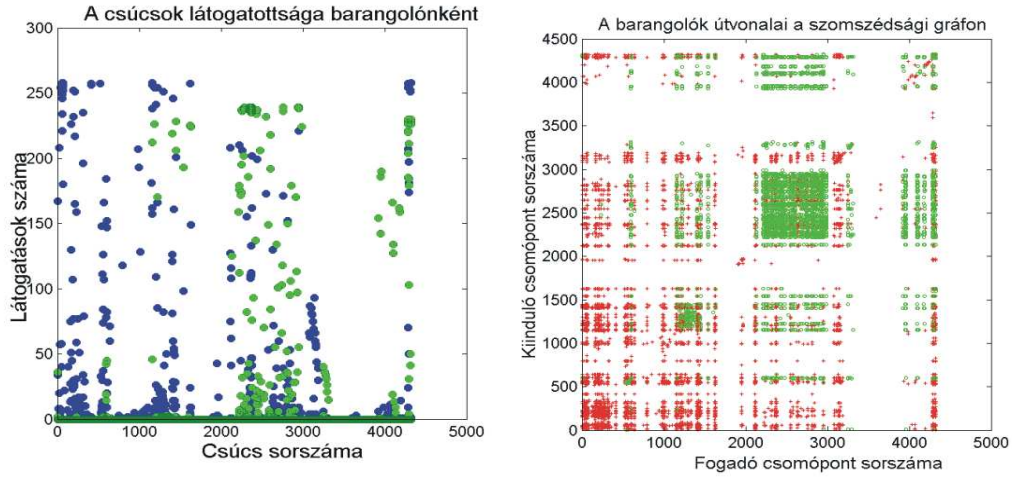
A felsorolt paraméterek rögzítése után vizsgáltuk a barangolók teljesítményét a weblog méretét, valamint a memória és környezet méretét változtatva. Amint eddigi vizsgálataink során is láttuk, a memória méretének meghatározó szerepe van abból a szempontból, hogy a barangolók a gráf mekkora területét képesek megvizsgálni. Amikor a memória nagyságát 10-re csökkentettük, az a barangolók teljesítményének romlását eredményezte. Ekkor a gráf hatványfüggvény alakú kapcsolati eloszlása miatt a

barangoló csak a nagy konnektivitású csúcspontok kis környezetét képes bejárni, ezért a megjelenő új információ kis hányadát képes csupán begyűjteni (22. ábra). Mivel a weblog mérete ugyanekkor 100 volt, a megtalált új információ kora szintén sokkal magasabb, mint a többi kísérletben (23. ábra). A másik szélső esetben, amikor a weblog mérete csupán 10 volt, a vizsgált környezet viszont 100 csúcspontra terjedt ki, a barangolók hatékonysága jelentősen megnőtt, mivel a gráf jóval nagyobb hányadát képesek voltak megvizsgálni (21. ábra). Szegmentáció viszont nem alakult ki, mivel a nagy környezet miatt a barangolók képtelenek voltak elkerülni egymást. A weblog dokumentumai között viszont átfedés alakult ki, mert a kevés számú nagy konnektivitású csúcspont nehezen kerül ki a weblogokból (22. ábra). Ezzel szemben a nagy méretű weblog még a nagy környezet mellett is segíti a szegmentáció kialakulását (18. ábra), és így ebben az esetben alakult ki a legmagasabb hatékonyság, míg a releváns dokumentumok átlagos kora is viszonylag alacsony maradt (17. ábra). A legteljesebb szegmentációt viszont mindkét paraméter közepes értéke mellett sikerült elérnünk (20. ábra), viszonylag magas hatékonyság mellett (19. ábra).



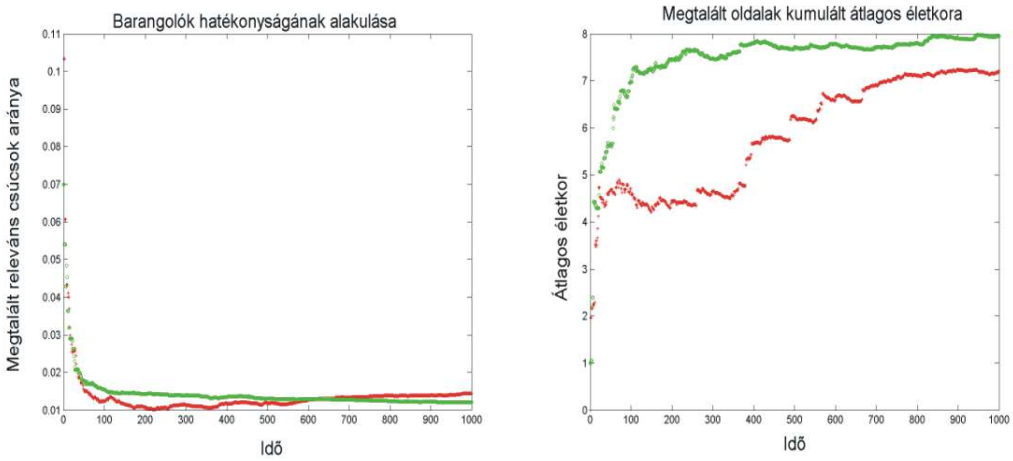
17. ábra

A két barangoló hatékonyságát jellemző mutatók alakulása abban az esetben, amikor a környezet kiterjedtségét meghatározó *memory*, és a *welblog* mérete egyaránt 100 volt. A megtalált releváns oldalak arányát és azok életkorát a frissülés rátáját figyelembe véve egyaránt a pirossal jelölt barangoló volt a hatékonyabb



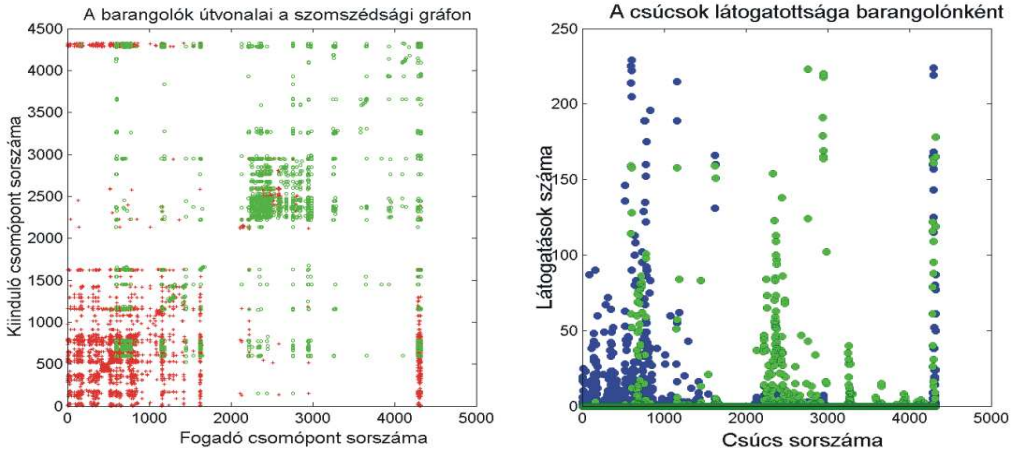
18. ábra

A két barangoló által választott linkek, valamint a gráf csúcspontjainak látogatottsága abban az esetben, amikor a környezet kiterjedését meghatározó *memory*, és a *welblog* mérete egyaránt 100 volt. Jól látható, hogy a barangolók hatékonyan fel tudták osztani egymás között a rendelkezésre álló területet.



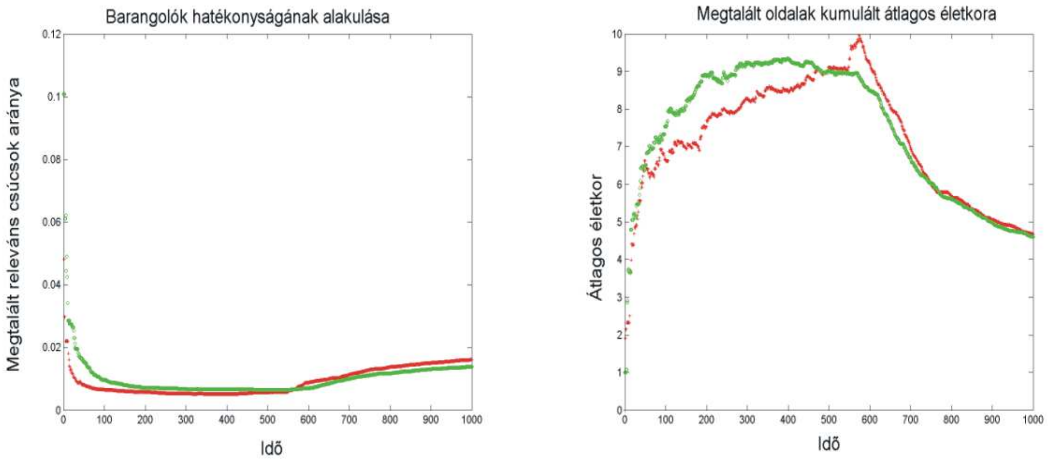
19. ábra

Amikor a *memory*, és a *welblog* mérete egyaránt 50 volt a szegmentáció szintén megfigyelhető. A pirossal jelölt barangoló time=400 időpontban ismeretlen területet talált, így javítani tudta teljesítményét.



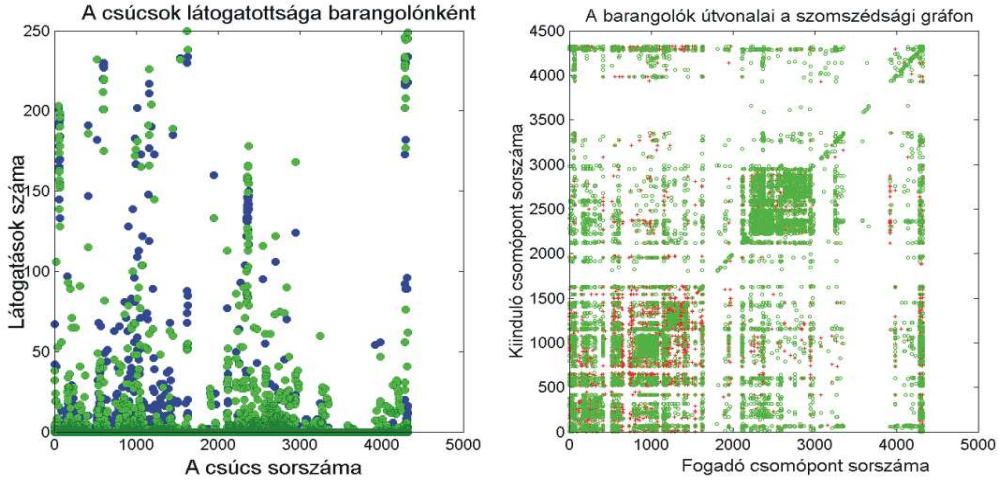
20. ábra

Amikor a környezet kiterjedését meghatározó *memory*, és a *welblog* mérete egyaránt 50 volt, megfigyelhető, hogy mindkét barangoló rendelkezett olyan csúcsponttal weblogjában, amelyik a másik barangoló általa vizsgált területhez tartozott. Amennyiben a weblog mérete kisebb, és γ értéke szintén kisebb, ezek a csúcspontok gyorsabban eltűnnek.



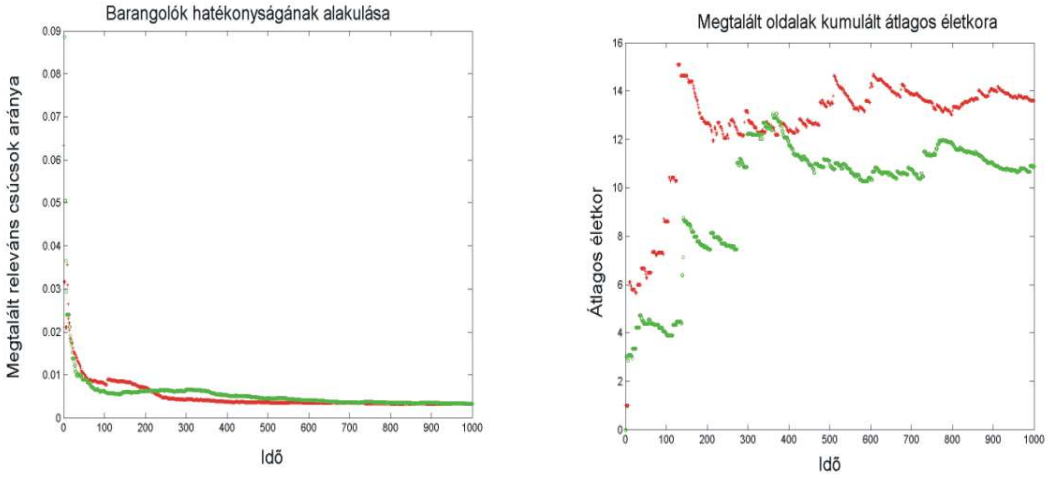
21. ábra

Amikor a *memory* mérete 100, míg a *welblog* mérete egyaránt csupán 10 volt a szegmentáció nem alakult ki úgy, ahogyan vártuk. A time=400 időpont környékén stabilizálódtak a weblogok ezért ekkor elkezdett csökkenni a begyűjtött releváns dokumentumok életkora. A hatékonyság sokáig csupán fele az ideális értéknek, majd a stabilizációt követően nőni kezd.



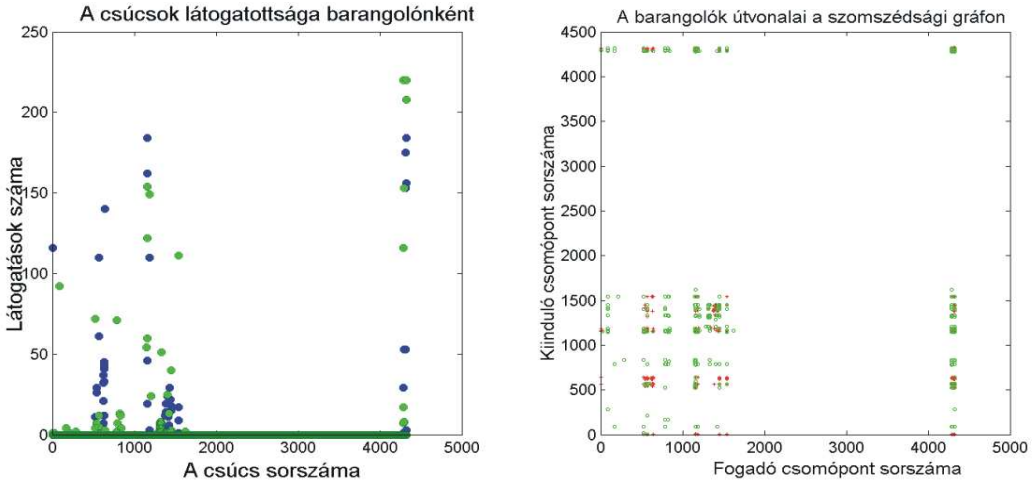
22. ábra

A *memory*=100 és *welblog*=10 paraméterek mellett megfigyelhető, hogy szegmentáció már nem alakult ki teljes mértékben, azaz a barangolók a gráf ugyanazon területeit vizsgálták párhuzamosan. Ezt igazolja látogatottság alakulása és a választott élek megjelenítése egyaránt.



23. ábra

A *memory*=10 és *welblog*=100 paraméterek mellett viszont az előzővel ellentétes tendenciák figyelhetőek meg. A barangolók hatékonysága messze elmarad a korábban vizsgált esetekben megfigyelttől. A megtalált releváns csúcspontok kora viszont többszörösen meghaladja az eddigieket.



24. ábra

A *memory*=10 és *welblog*=100 paraméterek mellett a barangolók a gráf elenyészően kis hányadát tudták bejárni a kísérlet időtartalma alatt. Ennek oka a gráf kapcsolatainak eloszlásában keresendő. A barangolók mozgása a nagy konnenktivitású csúcspontokra korlátozódik.

5 Köszönetnyilvánítás

Végezetül szeretném megköszönni Dr. Lőrincz Andrásnak, hogy ötleteivel, javaslataival nagymértékben segítette munkámat, és az ELTE Neurális Információfeldolgozási Csoport tagjainak, hogy az eredményes munkához a légkört megteremtették, hozzászólásaikkal, kérdéseikkel konstruktív módon befolyásolták a témáról alkotott elképzeléseimet.

Irodalomjegyzék

[1] I. Kókai, A. Lőrincz. Learning to search on the world wide web

[2] P. Dayan. The convergence of $TD(\lambda)$ for general λ, *Machine Learning* 1992. vol. 8, 341-362. oldal

[3] R. Sutton. Learning to predict by the method of temporal differences, *Machine Learning* 1988. vol. 3, 9-44. oldal

[4] S. Singh and T. Jaakkola and M.~L. Littman and Cs. Szepesvári. Convergence results for single-step on-policy reinforcement-learning algorithms, *Machine Learning*, 2000 vol. 38, 287-303. oldal

[5] Cs. Szepesvári és A. Lőrincz. Behavior of an adaptive self-organizing autonomous agent working with cues and competing concepts, *Adaptive Behavior*, 1994 vol. 2, 131-160. oldal

[6] P. Dayan. Improving generalization for temporal difference learning, *Neural Computation*,1993 vol. 5, 613-624. oldal

[7] T. Thrun and A. Schwartz. Finding structure in reinforcement learning, *Morgan Kaufmann*, 1995 vol. 7, Advances in Neural Information Processing Systems

[8] K. Doya. Temporal difference learning in continuous time and space *Advances in Neural Information Processing Systems* 1996 vol. 8

[9] K. Doya. Reinforcement learning in continuous time and space, *Neural Computation* 1999 vol. 12, 243-269. oldal

[10] T. Hofmann. Learning the Similarity of Documents: An Information-Geometric Approach to Document Retrieval and Categorization *Neural Information Processing Systems*, 2000 vol. 2, 914-920. oldal

[11] A.K. McCallum. BoW: A toolkit for statistical language modelling,text retrieval, classification and clustering 1996.

[12] A.-L. Barabási. The physics of the Web, 2001. július

[13] R. Sutton és A.G. Barto. Reinforcement Learning: An Introduction *MIT Press, Cambridge*, 1998

[14] A. Lőrincz, I. Kókai és A. Meretei. New generation of the world wide web: anticipating the birth of the `hostess' race *International Journal of Foundations of Computer Science*, 2001.

[15] R. Albert. Statistical mechanics of complex networks *Phd Thesis* 2001.

[16] F. Menzer, G. Pant és P. Srinivasan. Evaluating topic driven web crawlers 2001

[17] A. Lada. Search in power low networks, 2001. március

[18] T. Joachim. A probabilistic analysis of the Rocchio algorithm with TFIDF for text categorization, *Proc. 14th International Conference on Machine Learning*1996

[19] S. Chakrabarti, M. van der Berg és B. Dom. Focused crawling: a new approach to topic-specific Web resource discovery, *8th International World Wide Web Conference* 1999

[20] A. Border. Graph structure of the web 2001.

[21] S. Chakrabarti, Data Mining for Hypertext: a Tutorial Survey, *Explorations: Newsletter of the Special Interest Group (SIG) on Knowledge Discovery & Data Mining*, vol. 1. 2000