Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Távközlési és Médiainformatikai Tanszék

Sebők Tamás

FÖLDRAJZI ÚTVONALVÁLASZTÁS HEURISZTIKUS OPTIMALIZÁLÁSA

KONZULENS

Dr. Gulyás András,
Csernai Márton

BUDAPEST, 2013

Tartalomjegyzék

Összefoglaló **5**

Abstract **6**

1 Bevezetés **7**

2 Elméleti háttér **8**

2.1 A földrajzi útvonalválasztás 8

2.1.1 Definíció 8

2.1.2 Metrikus terek 8

2.1.3 Mohó továbbítás 9

2.1.4 Irány alapú továbbítás 11

2.1.5 Vetített távolság alapú módszerek 13

2.2 Alkalmazási területek 15

2.2.1 Ad-hoc hálózatok 15

2.2.2 Szenzor hálózatok 17

2.2.3 Internet hiperbolikus térben 21

2.2.4 Szociális hálózatok 24

3 A tervezett algoritmus **28**

3.1 A heurisztikák 28

3.1.1 Definíció 28

3.1.2 Használatuk okai 28

3.1.3 Sorrend alapú heurisztikák 28

3.1.4 Irány alapú heurisztikák 31

3.2 A szimulációs környezet 36

3.2.1 A fejlesztői környezet 36

3.2.2 A vizsgált topológiák 38

3.2.3 A vizsgált tulajdonságok 40

3.3 A szimuláció eredménye 43

3.3.1 Az inicializálásokra fordított idő 43

3.3.2 Az inicializáló keresések hatása a TO útvonalválasztásra 44

3.3.3 A szomszédlistával elérhető nyereség irány alapú továbbításnál 45

3.3.4 A továbbító algoritmusok összehasonlítása 46

3.3.5 A kombinált továbbítás változatainak összehasonlítása 53

3.3.6 Az irány közelebb továbbításnál (ST) vizsgálandó szomszédok száma..... 58
4 Összefoglalás, továbblépési lehetőségek..... 60
4.1 Összefoglaló..... 60
4.2 Továbblépési lehetőségek..... 61
Köszönetnyilvánítás..... 62
Irodalomjegyzék..... 63

HALLGATÓI NYILATKOZAT

Alulírott **Sebők Tamás**, szigorló hallgató kijelentem, hogy ezt a diplomatervet meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2013. 12. 13.

...…….

Sebők Tamás

Összefoglaló

A földrajzi útvonalválasztás gondolata először az 1980-as években merült fel rádiós hálózatokkal kapcsolatban. Jelenleg is főleg vezeték nélküli hálózatokban használják, használata az ad-hoc és szenzor hálózatok esetén a legelterjedtebb. A földrajzi útvonalválasztás azonban más területeken is használható, kutatások vannak például az alkalmazására közösségi, szociális hálók esetében is, sőt, vizsgálták a használatát Internet topológián is, ezzel új lendületet adva a földrajzi útvonalválasztás kutatásának is.

A földrajzi útvonalválasztásnál használt mohó továbbítás egy egyszerű ötletre épít: a hálózatban lokális döntések segítségével próbáljunk egy globálisan is jó útvonalat találni. A mohó útvonalválasztást nem csak az egyszerűsége (csak lokális információk kellenek a döntéshez) teszi népszerűvé, hanem az elosztott működése is. Nincs szükség a teljes hálózat ismeretére az útvonalválasztáshoz, nem kell az útvonalat előre meghatározni, az útvonal a továbbítás közben, lokális döntések következtében alakul ki.

Azonban a mohó továbbítással kapcsolatban vannak megoldandó problémák. A legnagyobb probléma a számításigénye, ugyanis a mohó algoritmus mindig a lokális optimumot keresi. Ehhez meg kell vizsgálni minden szomszédot, ezért meghatározása sok szomszéddal rendelkező csomópontok esetén költséges lehet. A dolgozatban azt mutatom meg, hogyha a továbbításnál lemondunk a lokális optimum megkereséséről, és heurisztika segítségével próbálunk egy lokálisan jó (de nem feltétlenül a legjobb) döntést hozni, akkor közel olyan jó megoldáshoz jutunk lényegesen kevesebb számítással, ezáltal sokkal gyorsabban.

Abstract

The idea of using geographic routing in radio networks appeared in the 80's. Nowadays geographic routing is widely used in ad-hoc and sensor networks. However, geographic routing can be used in other types of networks. Researches show that geographic routing is efficient in social networks, and it can also be used on a metric Internet topology.

Greedy forwarding is the basic forwarding method of geographic routing. It is based on a simple idea: we can try to reach a generally good path of the network by making locally optimal decisions. Simplicity is not the only positive feature of greedy forwarding. Greedy forwarding works in a completely distributed fashion, which is a requirement in large networks. No information is needed about the whole network, and there is no need to predefine the path. Instead, it is built up step by step, based on local decisions.

Although it is an effective concept, greedy forwarding has some problems to be solved. Namely, its largest problem is its computational complexity. This complexity comes from making locally optimal decisions. To make an optimal decision, every node has to examine all of its neighbors, which is expensive to do if the point has many neighbors. In this paper I show that by giving up finding the optimal decision, and making a locally good (but perhaps not the best) decision with heuristics, we can find a solution almost as good as greedy forwarding with much less computation. And less computation means faster forwarding.

1 Bevezetés

A földrajzi útvonalválasztás egy régóta ismert, és számos területen alkalmazott útvonalválasztási módszer. Népszerűsége főleg annak köszönhető, hogy könnyen implementálható, és mivel elosztottan működik, a hálózat mérete se szab határt a használhatóságának.

Bár a hálózat mérete nem okoz problémát, de ez a számításigényről nem mondható el. Ugyanis a földrajzi útvonalválasztásnál használt mohó továbbítás minden lépésben a célhoz legközelebb eső pont felé továbbít. Ehhez azonban minden lehetőséget, azaz a pont minden szomszédját megvizsgálja, hogy vajon az a pont lenne-e a legjobb választás a továbbításhoz. Ez rengeteg számítással jár, és különösen megterheli azokat a csomópontokat, melyek sok szomszéddal rendelkeznek.

A feladatom a földrajzi útvonalválasztás heurisztikus optimalizálása, tehát heurisztika segítségével próbálom az előbb említett műveletigénnyel kapcsolatos problémát orvosolni. A heurisztika a továbbítással szemben támasztott követelmények gyengítéséért cserébe kisebb műveletigényű, ezáltal gyorsabb algoritmust ad, így segíti a földrajzi útvonalválasztás egyszerűsítését.

A dolgozat elején ismertetem a földrajzi útvonalválasztás elméletét, megmutatom, hogy milyen hálózatok esetén, és hogyan lehet használni. Ezt követően megadom, hogy milyen követelményeket akarok kielégíteni az új heurisztikával, majd nekilátok a heurisztika megtervezésének. Több heurisztikákat mutatok be, megadva hogy az adott heurisztika használata milyen előnyökkel és hátrányokkal jár.

A heurisztikák működésének vizsgálatához létrehozok egy szimulációs környezetet, ahol a hagyományos mohó továbbítás mellett más algoritmusokkal is össze tudom vetni a tervezett heurisztikákat. A kapott eredmények ismeretében összehasonlítom a tervezett heurisztikákat a korábban már ismertetett algoritmusokkal, és kiértékelem őket. A dolgozat végén összefoglalom az elvégzett munkát, összegzem a kapott eredményeket, és írok a továbblépési lehetőségekről.

2 Elméleti háttér

2.1 A földrajzi útvonalválasztás

2.1.1 Definíció

A földrajzi útvonalválasztás onnan kapta a nevét, hogy az útvonalválasztás a földrajzi helyzeten alapul. Ennek megfelelően minden csomópontot a helyzetével (koordináták) azonosítunk, és a csomópontok a saját helyzetükön felül ismerik a szomszédjaik helyzetét is. Ha egy csomagot akarunk továbbítani, akkor ismernünk kell a cél címét is. Azonban a csomópontok ezen felül semmilyen egyéb információt nem tartanak fenn a hálózatról, nem kell ismerniük a hálózati topológiát, tehát nincs szükség a hálózat felderítésére és ilyen információk tárolására és megosztására. Ez jelentősen lecsökkenti a hálózat fenntartásához szükséges kommunikáció mennyiségét a hálózatokban, és az eszközök is kevesebb erőforrást igényelnek egy földrajzi útvonalválasztást használó hálózatban.

2.1.2 Metrikus terek

A földrajzi útvonalválasztásnál nincs megkötés arra, hogy a csomópontok helyzetét konkrétan hogyan írjuk le, tetszőleges metrikus térben használható. Metrikus térnek egy olyan halmazt nevezünk, amelyen értelmezve van egy távolságfüggvény (más néven metrika). A távolságfüggvény a halmazban lévő bármely elempárhoz egy nemnegatív valós számot rendel, amit az elemek távolságának nevezünk.

A távolságfüggvénytől az alábbi tulajdonságok teljesülését követeljük meg (x, y, z a halmaz tetszőleges elemei, d(x,y) a távolságfüggvény):

1. Két pont távolsága pontosan akkor 0, ha a két pont egybeesik.
   $d(x, y) = 0 \Leftrightarrow x = y$

2. A távolságfüggvény szimmetrikus.
   $d(x, y) = d(y, x)$

3. A távolságfüggvényre teljesülnie kell a háromszög egyenlőtlenségnek, azaz két pont távolsága nem lehet kisebb, ha egy köztes ponton

keresztüli távolságot nézzük.

$$d(x,z) \le d(x,y) + d(y,z)$$

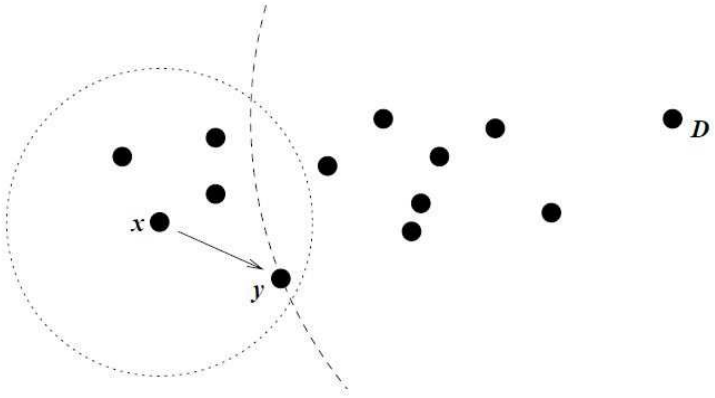
A legismertebb metrikus tér az Euklideszi-tér, ahol az Euklideszi távolságfüggvény szerint határozzuk meg a pontok távolságát. Az Euklideszi távolságfüggvény n dimenziós Euklideszi térben $d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$ alakban írható fel, ennek az általánosítása a Minkowski távolságfüggvény ($d(x,y) = \sqrt[p]{\sum_{i=1}^{n} |x_i - y_i|^p}$). A Minkowski távolságfüggvény p=1 esetben a Manhattan-távolságot adja, ami úgy fogható fel, mintha egy egységoldalú négyzethálónak a vonalain vett távolságot néznénk. A hiperbolikus és gömbi térben is megvannak a megfelelő távolságfüggvények, melyek a tér megfelelő függvényeit használják a távolság meghatározásához.

Vannak azonban más távolságfüggvények is, melyek tetszőleges halmazon teljesítik a három axiómát. Például a $d(x,y) = \begin{cases} 0, ha\ x = y \\ 1, ha\ x \neq y \end{cases}$ függvény, ami minden nem egybe eső pont távolságát 1-nek veszi, vagy a $d(x,y) = \max_{1 \le i \le n} \left\{ |x_i - y_i|^2 \right\}$ függvény. Ezekkel a függvényekkel csak azt szeretném megmutatni, hogy a metrika használata nem jelent megszorítást, azonban a metrikák eltérő tulajdonságai miatt fontos a megfelelő metrika kiválasztása.

2.1.3 Mohó továbbítás

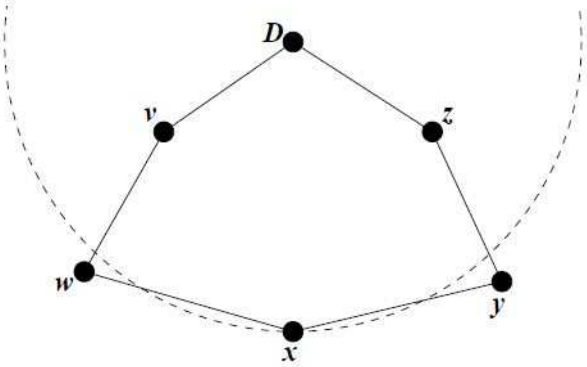
A földrajzi útvonalválasztás a routing során a mohó algoritmust használja a továbbításra. A mohó továbbítás teljesen elosztott módon működik, az útvonal minden pontjában lokális információk alapján történik a döntés, hogy merre történjen a továbbítás. A mohó továbbítás lényege, hogy az adott pontban elérhető összes alternatívát megvizsgálva annak a csomópontnak történik a továbbítás, ami az adott pontban a legjobb választás. A hagyományos mohó továbbításnál a csomópont kiszámítja az összes szomszédjának a céltól vett távolságát (a megfelelő metrikus térben), és annak a szomszédjának továbbítja a csomagot, amelyik a célhoz legközelebb esik.

A mohó továbbítást menetét szemlélteti a 2.1. ábra, ahol x pont felől akarunk D csomópontnak továbbítani egy csomagot. A továbbításhoz x pont az y szomszédját használja, mivel az van a legközelebb a cél csomóponthoz, azaz D-hez.



2.1. ábra A mohó továbbítás [1]

Ahhoz, hogy a mohó továbbítás ne kerüljön végtelen ciklusba, leállási feltételt kell megadnunk. A leállási feltétel lehet például az, hogy csak addig történjen továbbítás, amíg a célhoz közelebb kerülünk. Ilyenkor azonban a mohó továbbítás nem minden esetben sikeres, ezt mutatja a 2.2. ábra. Az ábrán látszik, hogy x mindkét szomszédja (y és w) is távolabb van a céltól (D-től) mint x, ezért a továbbítás leáll, hiába lenne út w-n vagy y-on keresztül is D-be.



2.2. ábra A mohó továbbítás nem minden esetben sikeres [1]

A mohó továbbítás sikeressé tehető például az ú.n. „face” routing segítségével, így garantálni lehet a sikeres továbbítást, amennyiben létezik útvonal a forrás és a cél csomópont között. A face routing technológia azt az esetet tudja orvosolni, amikor a mohó algoritmus lokális optimumba kerül, és segítségével a routing ilyenkor folytatható [2].

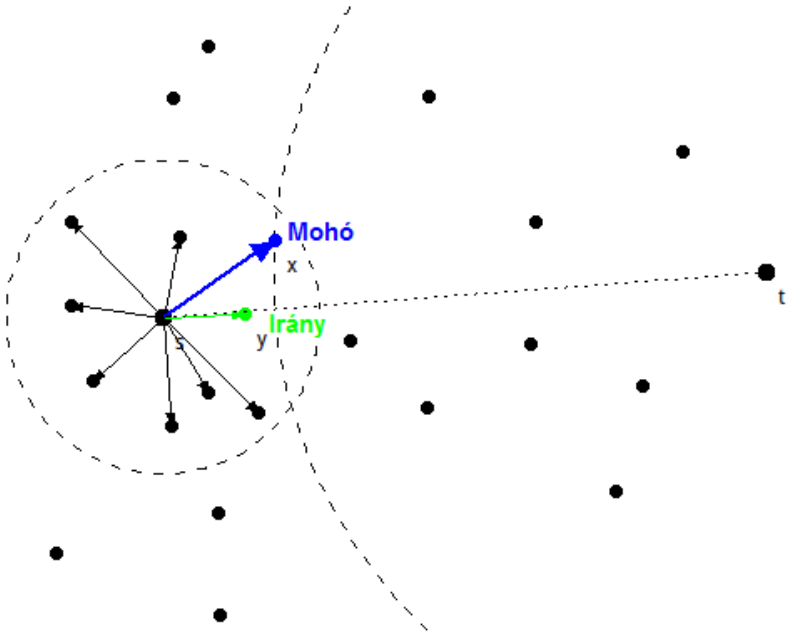
Az alábbiakban látható a mohó algoritmus pszeudokódja. A pszeudokódban megadott algoritmus akkor áll le, ha nem tud közelebb kerülni a célhoz. A szimulációkban ennek a pszeudokódnak az implementációját használom.

```
Adott G(V,E) gráf; egy forrás és egy cél csomópont
begin
    aktuális = forrás
    előző = Ø
    while(aktuális != cél és előző != aktuális)
          akt_táv = távolság(aktuális, cél)
          legközelebbi = aktuális
          for szomszed sz in szomszédjai(aktuális)
                szomsz_táv = távolság(sz, cél)
                if (szomsz_táv < akt_táv)
                      akt_táv = szomsz_táv
                      legközelebbi = sz
                end if
          end for
          előző = aktuális
          aktuális = legközelebbi
    end while
end
```

1. pszeudokód Mohó továbbítás

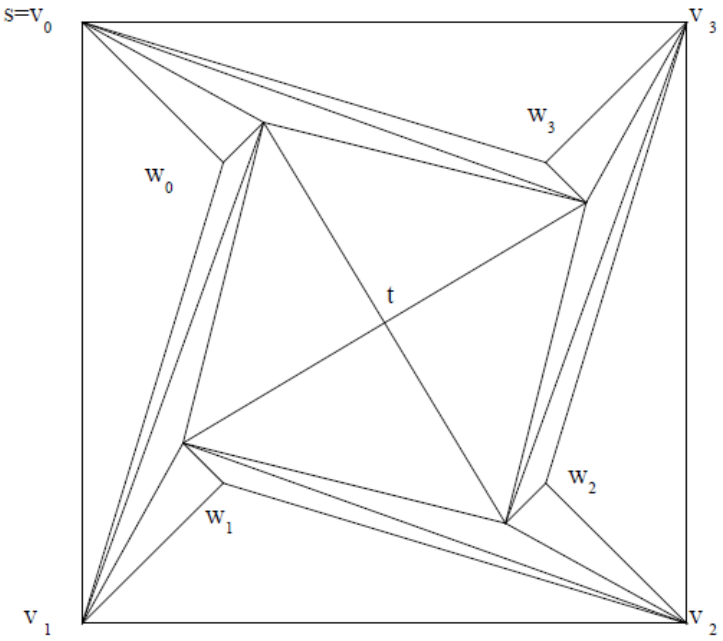
2.1.4 Irány alapú továbbítás

A földrajzi útvonalválasztásnál a mohó továbbítás mellett más módszerek is rendelkezésre állnak a továbbításra. A mohó továbbításhoz hasonlóan működik az irány alapú továbbítás, csak itt az algoritmus nem a legközelebbi, hanem a céllal legjobban egyező irányba eső szomszéd felé továbbít. Az irány alapú továbbítás és a mohó továbbítás összehasonlítása látható a 2.3-as ábrán. A mohó továbbítás (a definíciójából eredően) minden továbbítás során legalább olyan közel visz a célhoz, mint az irány alapú továbbítás.



2.3. ábra Az irány alapú továbbítás

Azonban a továbbítás során végtelen ciklusba kerülhetünk, ezt mutatja a 2.4. ábra. Akármelyik v_i pontból indulunk ki, a legjobban egyező irányú szomszédnak továbbítás során soha nem fogunk eljutni t pontba, ugyanis v_i-ből w_i esik a legközelebb t irányához, ahonnan azonban v_{i+1}-be vezet az algoritmus, mivel annak az iránya hasonlít a legjobban t irányához. Pedig az ábrán látszik, hogy 2 lépés alatt bármelyik v_i-ből eljuthatnánk t-be. A végtelen ciklus elkerülésének érdekében be kell vezetni egy lépésszám (hopcount) határt, aminek elérésekor az algoritmus leáll.



2.4. ábra Az irány alapú továbbítás végtelen ciklusba kerülhet [3]

Az algoritmus pszeudo kódja az alábbiakban látható. Fontos eltérés a mohó továbbításhoz képest, hogy itt ismernünk kell a hálózat tulajdonságait, és ezen tulajdonságok alapján kell beállítani a lépésszám korlátot. Túl alacsony korlát esetén csökkenhet a sikeresség, mivel a távolabbi csúcsok közti hosszabb útvonalakat kiszűri a korlát. Túl magas korlát esetén viszont a hatékonyság romolhat olyan végtelen ciklusba került továbbításokkal, amik sose lesznek sikeresek.

```
Adott G(V,E) gráf; egy forrás és egy cél csomópont; és egy lépészám_korlát
begin
    aktuális = forrás
    lépésszám = 0
    while(aktuális != cél és lépésszám < lépészám_korlát)
          továbbítás_iránya = irány(aktuális, cél)
          különbség = ∞
          for szomszéd sz in szomszédjai(aktuális)
                szomszéd_iránya = irány(sz, cél)
                if (eltérés(továbbítás_iránya, szomszéd_iránya) < különbség)
                      továbbítás_iránya = szomszéd_iránya
                      következő = sz
                end if
          end for
          aktuális = következő
    end while
end
```

2. pszeudokód Irány alapú továbbítás

Az algoritmus sikerességét lehet javítani a face routing technológia segítségével (lásd Compass Routing II, [3]), akárcsak a mohó továbbítást. Ezen algoritmusra a dolgozatomban több néven is hivatkozok. Az irány alapú továbbítás mellett a legjobban egyező irányú szomszédnak továbbítás is erre az algoritmusra utal, továbbá a BMD rövidítéssel (Best Matching Direction) is erre az algoritmusra hivatkozok.

2.1.5 Vetített távolság alapú módszerek

Ezek a módszerek a hagyományos mohó továbbításhoz hasonlóan távolságvizsgálat segítségével döntenek arról, hogy melyik szomszédjuk felé továbbítsanak. Azonban a hagyományos mohó továbbítással ellentétben ezek az algoritmusok nem a csomópont szomszédjainak céltól vett távolságát vizsgálják, hanem a csomópont és a cél közti egyenesre vetített távolságot, tehát hogy ezen az elképzelt

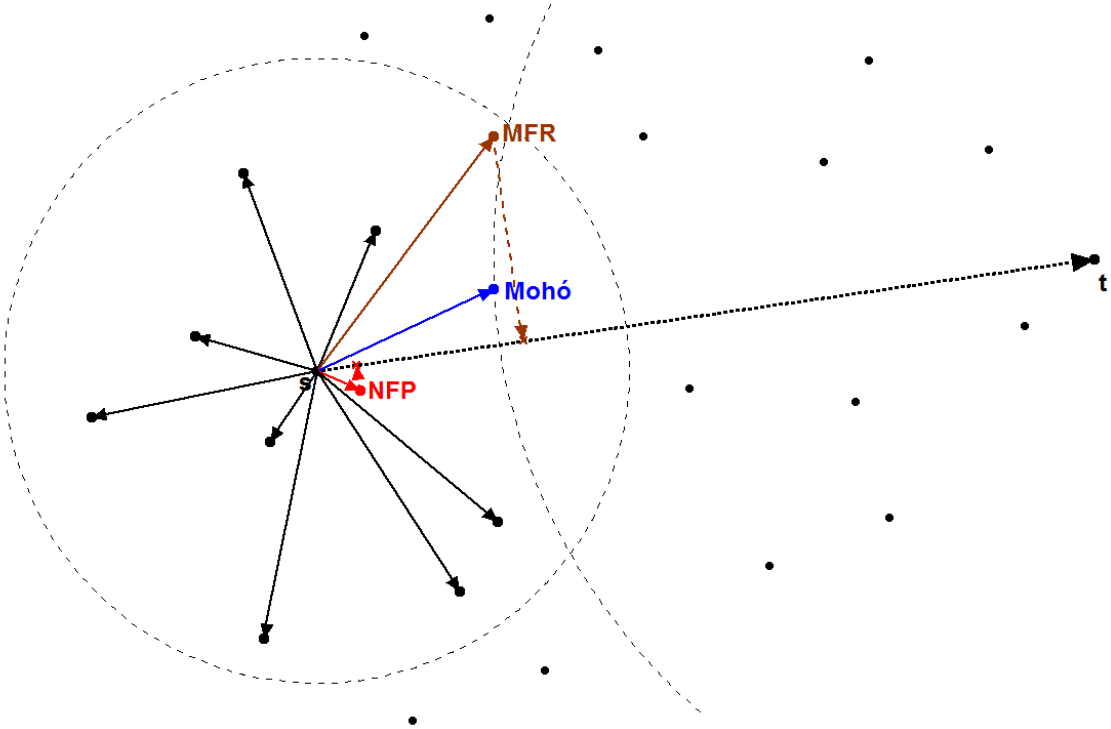
egyenesen mennyire közelítenék meg a célt, ha az adott szomszédnak továbbítanának [4].

2.1.5.1 Legkisebb vetített távú (NFP)

Ennél a továbbítási módszernél annak a szomszédnak történik a továbbítás, aki a kiindulási ponthoz (s) legközelebb van a kiinduló (s) és a cél (t) csomópont közti egyenesre vetítve. Az algoritmust Hou és Li használta először [5], a rövidítés az angol „Nearest Forwarding Progress” kifejezésből jön. A 2.5-ös ábrán pirossal jelöltem azt a csomópontot, amit az algoritmus az ábrán jelölt szituációban választana.

2.1.5.2 Legnagyobb vetített távolságú (MFR)

Ez a továbbítási módszer tekinthető az NFP ellentettjének, ugyanis itt az s és t közti egyenesre vetítve legmesszebbi csomópontnak történik a továbbítás [6]. Mint a 2.5. ábra is mutatja, előfordulhat, hogy a mohó továbbítás által választott szomszéd vetített távolsága kisebb, mint az MFR algoritmus által választott szomszédé, annak ellenére, hogy a mohó algoritmus közelebb visz a célhoz, mint az MFR algoritmus. Az MFR rövidítés a „Most Forward within Radius” kifejezésből ered [6].



2.5. ábra Vetített távolság alapú továbbítások

2.2 Alkalmazási területek

A földrajzi útvonalválasztás gondolata először az 1980-as években merült fel rádiós hálózatokkal és elkülönülő hálózatok összekapcsolásával [7] kapcsolatban. Jelenleg is főleg vezeték nélküli hálózatokban használják, a legelterjedtebb megoldás a face routingot is használó GPSR [1]. Van olyan megoldás is [8], ahol nem szükséges a földrajzi helyzet ismerete, ez főleg ad-hoc és szenzor hálózatok számára lett kidolgozva. A földrajzi útvonalválasztás azonban más területeken is használható, kutatások vannak az alkalmazására közösségi, szociális hálók esetében is, sőt, vizsgálták a használatát Internet topológián is [9]. Ebben a fejezetben ezen hálózatokat mutatom be.

2.2.1 Ad-hoc hálózatok

Az ad-hoc hálózatok decentralizált hálózatok, ahol a hálózat topológiája nem előre tervezett és rögzített, hanem dinamikusan változik (egy új csomópont csatlakozik a környezetében lévő csomópontokhoz). Így adatok továbbításnál a továbbításban részt vevő csomópontok dinamikusan változnak, nincs egy előre rögzített hierarchia. Az ad-hoc hálózatok jellemzően egyenrangú, vezeték nélküli eszközökből épülnek fel.

Az ad-hoc hálózatok elődjének a DARPA (Defense Advanced Research Projects Agency – Fejlett Védelmi Kutatási Projektek Ügynöksége, az Egyesült Államok Védelmi Minisztériumának kutatásokért felelős részlege) által létrehozott PRNet (Packet Radio Network) és SURAN (Survivable Radio Network) hálózatok tekinthetőek [10]. Ezek a katonai célra fejlesztett hálózatok tekinthetőek az első csomagkapcsolt hálózatoknak, így nem csak az ad-hoc, hanem minden csomagkapcsolt hálózat ősei.

A vezeték nélküli ad-hoc hálózatok egy speciális típusa a mobil ad-hoc hálózat (Mobile Ad-hoc NETwork, MANET). A MANET alkalmazásokra példa a gépjárművek közti kommunikációs hálózat, amit az ütközések elkerülésére hoztak létre [11], vagy a mobil eszközök közti Internet megosztására létrehozott hálózatok. De számos más területen lehet alkalmazni őket, például [10]:

- taktikai műveleteknél (katonai kommunikációs hálózat kiépítése ellenséges területen)
- mentőakcióknál (kommunikációs hálózat kiépítése ott is, ahol egyéb vezeték nélküli hálózatok nem elérhetőek)

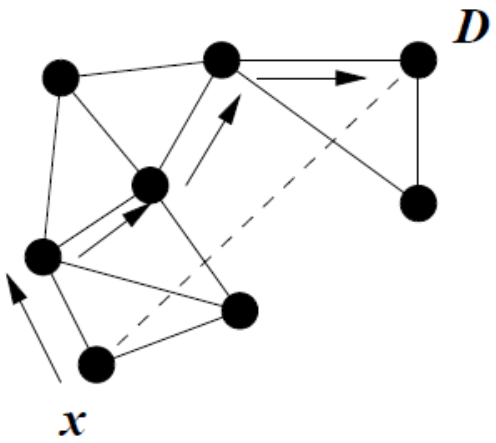
- vészhelyzetek esetén (ha meglévő kommunikációs hálózat sérült vagy nem használható)
- bűnüldözés esetén (ideiglenes rendőrségi kommunikációs hálózat)
- üzleti célokra (kiállítások, konferenciák kommunikációs hálózataként)
- oktatási célokra (virtuális osztálytermek)
- szenzor hálózatok (lásd következő fejezet)

A MANET-ek peer-to-peer hálózatok, azaz a kommunikáció a végpontok közt közvetlen módon, szerver beiktatása nélkül történik, emiatt teljesen elosztottan működő útvonalválasztó algoritmusokra van szükség. Számos ilyen algoritmus ismert, a MANET hálózatokban jellemzően a source routing és a földrajzi útvonalválasztó algoritmusok terjedtek el. Előbbiek esetén az útvonal meghatározás és frissítés előre megtörténik lekérdezések segítségével, amit az eszköz rögzít a csomag fejlécében. Az ad-hoc hálózatokban leggyakrabban használt source routing protokoll a DSR (Dynamic Source Routing Protocol [12] [13]).

A földrajzi útvonalválasztó algoritmusoknál a source routing algoritmusokkal ellentétben nincs szükség az útvonalak felderítésére, nyilvántartására, a nyilvántartás karbantartására, így jobban használhatóak nagyobb, dinamikusan változó hálózatok esetén [14]. A földrajzi útvonalválasztásra használhatók az előző fejezetben említett továbbító algoritmusok (mohó továbbítás, irány alapú /compass/ továbbítás, NFP, MFR algoritmusok), azonban ezeknél a sikeresség nem minden esetben garantált. Sikeres továbbítást garantáló algoritmusok a GPSR [1] és a face routing [2], ezek a lokális optimumokból tudják kisegíteni a mohó továbbítást síkba rajzolható (nincsenek egymást keresztező élek) topológiák esetén. Nem síkba rajzolható gráfok esetén is van megoldás, amivel a gráf egy részgráfját síkba rajzolhatóvá tehetjük [15]. Ezek a javító algoritmusok akkor lépnek életbe, amikor a mohó továbbítás elbukik (nincs közelebbi szomszéd), és addig történik a továbbítás a segítségükkel, ameddig nem talál egy, az elbukás helyénél közelebbi csomópontot a cél felé.

A GPSR [1] javító algoritmusát mutatja a 2.6-os ábra. Tegyük fel, hogy x csomópontban az algoritmus javító módban van (csak demonstrációs feltevés, az ábrán van közelebbi szomszéd, így működne a mohó továbbítás), és D csomópont felé kell továbbítani. Belső területek az élekkel körbezárt területek a gráfban, a külső terület, ami

nincs éllel körülzárva. Javító módba váltás után az algoritmus a területeket (face-eket) váltva (a terület szélét jelentő élen) továbbít folyamatosan egyre közelebbi területeken D csúcs felé. A következő területet mindig jobb kéz szabály szerint választjuk ki (az előző területtől jobbra esőt választjuk). Az ábrán látszik, hogy az egymás után következő területeket mind elvágja az $\overline{xD}$ egyenes. Amennyiben a javító algoritmus egy olyan pontba jut, ami közelebb van D-hez az elakadási helyénél (ahol javító módba váltott az algoritmus), az algoritmus visszavált a normál mohó továbbításra.



2.6. ábra A GPSR algoritmus [1]

Az ad-hoc hálózat eszközeiben helymeghatározási rendszer (pl. GPS) hiánya esetén is működhet földrajzi útvonalválasztó algoritmus. Egyfelől nem kötelező, hogy a rendszerünk metrikus terének bármi köze legyen a valós földrajzi helyzethez, másrészt létezik olyan algoritmus, ami egy relatív koordináta rendszerbe helyezi el a csomópontokat [16].

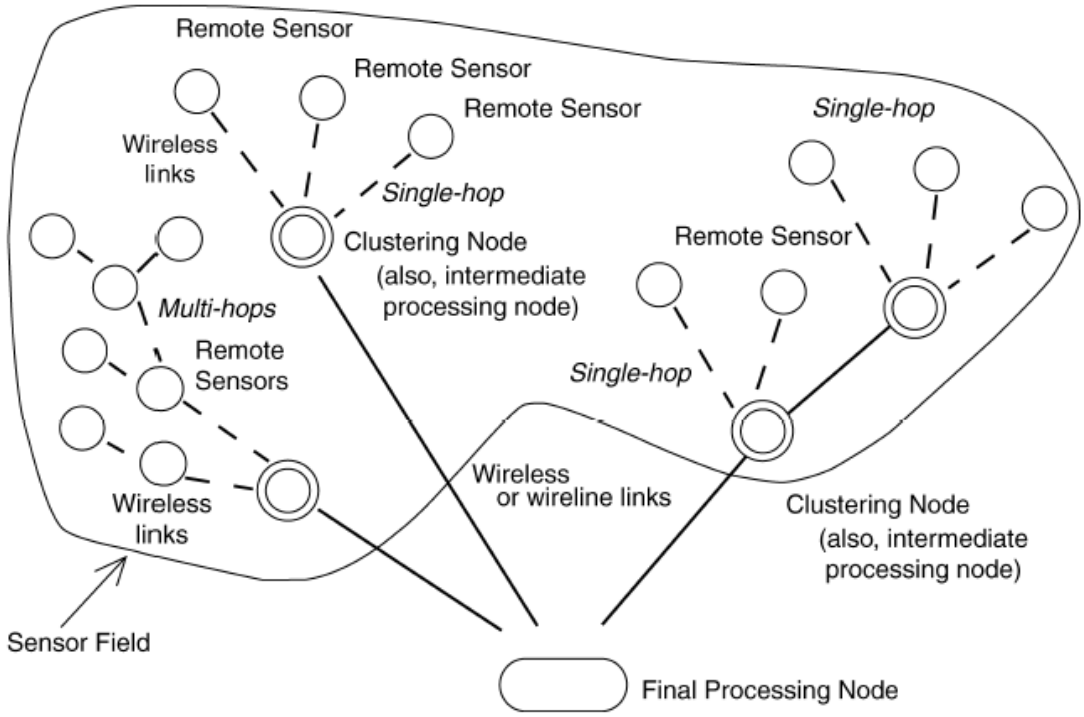
2.2.2 Szenzor hálózatok

Bár a szenzorhálózatok egy friss és aktív kutatási téma, mára számos, a témával foglalkozó könyv és cikk jelent meg, melyek részletesen tárgyalják ezen hálózatok működését, kitérve a speciális esetekre is. Dolgozatomban egy rövid, általános leírást adok ezekről a hálózatokról, melynek az alapjait a WSN hálózatok alapjait ismertető könyvek [17][18] és cikk [19] adja.

A vezeték nélküli szenzor hálózatok (Wireless Sensor Networks, WSNs) a térben elhelyezett szenzorokból épülnek fel. Ezen szenzorok feladata a környezetük monitorozása, az adatok feldolgozása és továbbítása. A szenzorhálózatok mérete rendkívül változó lehet, a pár szenzorostól pár ezer szenzoros hálózatokig. Kezdetben

főleg katonai célokra használták, azonban mára számos más alkalmazása is elterjedt. Elterjedéséhez a mikroelektronika és a vezeték nélküli hálózatok fejlődése vezetett, mivel elérhetővé váltak az olcsó, alacsony energiafogyasztású, multifunkciós szenzorok.

A szenzorok hálózaton belüli helyét nem tervezik előre, így lehetőség van megközelíthetetlen, vagy katasztrófa sújtotta területeken is WSN hálózatokat létrehozni. Azonban gondoskodni kell az algoritmusokban a hálózat önszerveződő képességéről, és megfelelő hálózati protokollokat kell használni, amelyek támogatják a véletlenszerűen létrejövő hálózatokon való továbbítást. Mivel a WSN hálózatok jellemzően egy adott területre leszórással jönnek létre, így természetesnek adódik a földrajzi útvonalválasztás használata. A szenzorhálózatok hierarchikus felépítését mutatja a 2.7. ábra. Az ábra a szenzorok jellemző szerepeit mutatja, a szenzorhálózatokban lévő szenzorok jellemzően azonos eszközök.



2.7. ábra A szenzorhálózatok felépítése [17]

A szenzorhálózatok ugyan az ad-hoc hálózatok közé tartoznak, azonban azoktól sok mindenben eltérnek. Jellemzően több eszközt tartalmaznak egy gyakrabban változó topológiában, az eszközök sűrűbben vannak telepítve, egyszerűbbek (limitált számítási és memória kapacitás), és jobban ki vannak téve meghibásodásoknak.

Szenzorok segítségével környezetünk különböző paramétereit tudjuk mérni, például a hőmérsékletet, páratartalmat, megvilágítást, nyomást, hangerőt, egy anyag meglétét és összetevőit, mozgást, objektumok jelenlétét és méretét. A sok paraméter sok alkalmazást is jelent, ezek az alábbi típusúak lehetnek [19]:

- Katonai (saját és ellenséges csapatok monitorozása szenzorok segítségével, hadszíntér megfigyelése, ellenálló erők felderítése, célzás segítése, károk felbecslése, biológiai és nukleáris támadások észlelése)
- Környezet megfigyelése (erdőtüzek detektálása, természeti katasztrófák /földcsuszamlás, földrengés, árvíz/ észlelése és előrejelzése, meteorológiai megfigyelések, víz és levegőminőség figyelése, vegyszerek észlelése, állatok mozgása)
- Egészségügyi (beteg fiziológiai állapotának monitorozása, betegek és orvosok helyének követése kórházban)
- Otthoni (automatizálás, okos otthon)
- Ipari (gép állapotának monitorozása, gyártással kapcsolatos adatok összegyűjtése és naplózása)

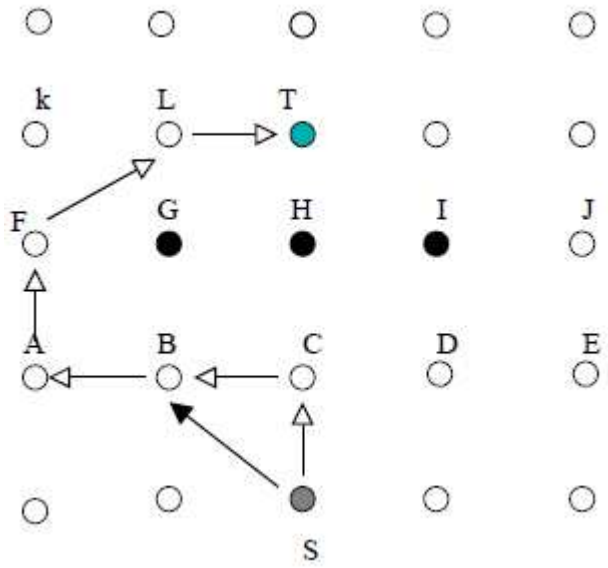
Mivel a szenzorhálózatoknál jelentős probléma az eszközök energiaellátása (jellemzően beépített akkumulátorral üzemelnek), ezért több cikk is született, amiben energiahatékonyság szempontjából is vizsgálták az útvonalválasztó algoritmusokat [20][21]. Vizsgálták kifejezetten az energiafelhasználás okait és csökkentésének lehetőségeit [22] is. Ezen cikkekből kiderül, hogy a kommunikációhoz több energia szükséges, mint a számításokhoz, és a rádió kikapcsolásával (alvó módba helyezésével) sok energia spórolható meg [22].

Ez a felismerés vezetett a GEAR protokollhoz [23] is. A GEAR kifejezetten szenzor hálózatok számára kidolgozott, a hagyományos földrajzi útvonalválasztáson és mohó továbbításon alapuló protokoll, amely figyelembe veszi a szenzorhálózatok sajátosságait, és az elsődleges cél a tervezésekor az energia hatékony felhasználása volt.

A GEAR algoritmus a mohó továbbítással ellentétben nem a legközelebbi szomszédot keresi, hanem a közelebbi csomópontok közül választ egyet (egy minimális költségűt), ha létezik ilyen szomszédja. Ha nem létezik ilyen szomszéd (tehát az adott irányban lyuk van), akkor a GEAR egy speciális tanulási folyamattal megtanulja, hogy

az adott lyukat hogy tudja elkerülni. Ehhez a protokoll költségeket definiál. Amennyiben egy pontból egy másikba nincs még megtanult költség, becsült költséget használ az algoritmus. A becsült költség számításánál figyelembe veszi a távolságot és a csomópontban felhasznált energiamennyiséget (a két tényezőt α illetve 1-α súllyal veszi, így α=1 beállításával lehetséges például csak a távolság alapján is dönteni). Később ezt a becsült költséget bírálja felül egy megtanult költséggel.

A 2.8-as ábrán egy ilyen lyuk észlelése és kerülő út kialakítása látható. Tegyük fel, hogy a G, H, I szenzorok lemerültek vagy meghibásodtak. S-ből akarunk T-be továbbítani, és az algoritmus első lépésében S továbbította a csomagot C szenzornak (csomópontnak). Ekkor C szenzor észleli, hogy T irányába nincs közelebbi szomszédja. Tegyük fel, hogy a lyuk még korábban nem került észlelésre, és így az algoritmus csak becsült költségekkel dolgozik. Ha a becsült költségben csak a távolságot vesszük figyelembe B és D becsült költsége ugyanannyi lesz. Válasszunk a két egyenlő költségű közül egy szabály alapján (pl a kisebb sorszámút, azaz B-t). Ekkor a megtanult költség C szenzor és T szenzor közt létrejön, és a becsült költségnél a B-ből C-be továbbítás költségével lesz nagyobb. A következő S-ből T-be továbbításkor már S-ből B felé továbbít az algoritmus, nem C felé, mivel arrafelé kisebb a költség.



2.8. ábra A GEAR algoritmus lyuk tanulási folyamata [23]

Szenzor hálózatok topológiájának generálásához használható a GENSEN [24] algoritmus, ahol a szenzorhálózat jellemzőinek megadása után előállíthatjuk a szenzorhálózat topológiáját. Ezt a topológiát fel lehet használni további vizsgálatokhoz.

2.2.3 Internet hiperbolikus térben

Az Internet, vagy más néven világháló a legnagyobb számítógépes hálózat a Földön, amit minden nap több mint egy milliárd ember használ munkára, kommunikációra, információk keresésére és megosztására, hírek olvasására. Az Internet autonóm rendszerek (AS, Autonomous System) összekapcsolásából épül fel. Egy AS az Internet egy olyan különálló részének tekinthető, amit egy szervezet birtokol és tart fenn (adminisztrál). Az AS-ek mérete nagyon különböző lehet, pár számítógéppel rendelkező kis cégektől multinacionális vállalatokig számos szervezet üzemeltet AS-t. Nincs központi Internet hatóság, ami meghatározná, hogy melyik AS-nek melyik AS-hez kell csatlakozni, az AS párok hoznak független döntéseket arról, hogy kapcsolódnak egymáshoz. Emiatt az Internet egy tisztán önszerveződő hálózatnak tekinthető [9].

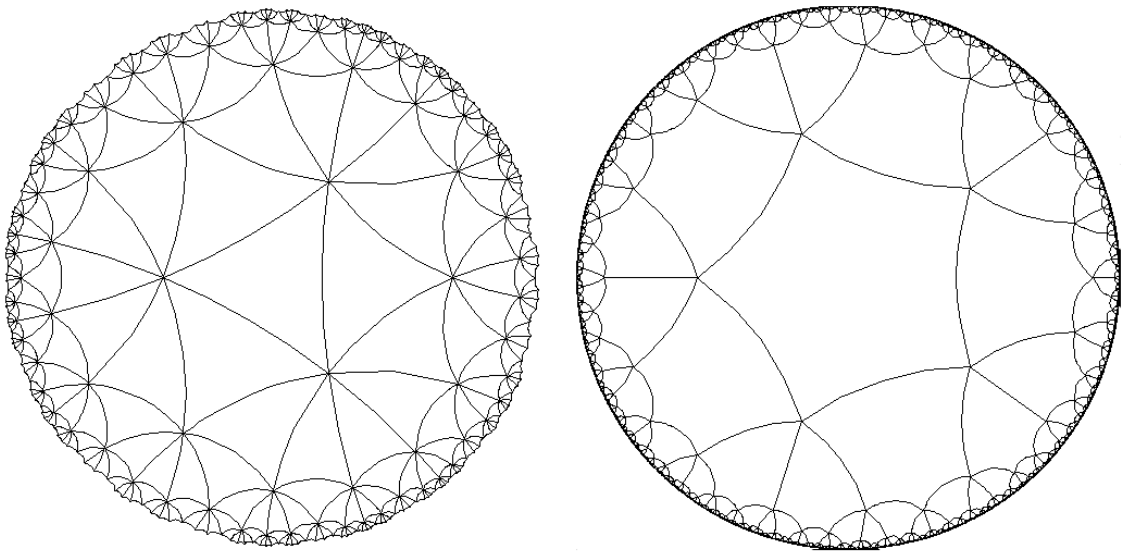
Az Internet rohamtempóban nő, évente körülbelül 2400 új AS csatlakozik rá, jelenleg több mint 30000 AS hálózatból áll össze az Internet [25]. Kérdés, hogy a rohamosan növekvő Internet fenn tudja-e tartani eredeti funkcióját (csomagok küldése számítógépek közt), nem teszi-e tönkre ez a rohamos növekedés? A kérdés jogos, ugyanis jelenleg az Interneten csomagok továbbításához az összes AS-nek fel kell deríteni és nyilván kell tartani minden lehetséges cél felé az útvonalat. Ez egy statikus hálózaton is nagyon bonyolult és memóriaigényes folyamat, de a dinamikusan változó Internet hálózaton még ennél is nehezebb a feladat, ami jelentős kommunikációs többletköltséget (overheadet) jelent és routing tábla növekedést okoz. Ez egy valós és sürgősen megoldandó probléma, ezt több szakember is jelezte már [26].

A földrajzi útvonalválasztás és a mohó továbbítás pontosan ezt a problémát oldja meg. Nincs szükség a hálózat globális ismeretére (ezáltal ezen ismeret folyamatos frissítésére se), pusztán a környezetünkben lévő csomópontokat kell ismerni. A feladat tehát adott: meg kéne alkotni az Internet metrikus térbe ágyazását. Ezt szerencsére elvégezték már [9], a következőkben erről a beágyazástól írok.

Ahhoz, hogy az Internetet a metrikus térbe lehessen ágyazni, először egy geometriai modellt kell alkotni. A legegyszerűbb modellként vehetünk egy körlapot, amin egyenletesen elosztjuk a csomópontokat. A következő kérdés, hogy mely pontokat kössük össze egymással. Ha a csomópontok közti távolsággal fordított arányban van az összeköttetés valószínűsége, akkor jól közelítjük a valós hálózatokat, ahol

hasonlóképpen történik az összekötés. Azonban ilyen összekötés esetén a csomópontok fokszáma Gauss-eloszlást követ, az Internet pedig egy skálafüggetlen hálózat, fokszámeloszlása hatványeloszlást követ [27]. Emiatt a modellben a csomópont fokszámokat a skálafüggetlen eloszlás szerintinek kell venni, ezután az összekötéseket pedig a távolsággal fordított arányosságú valószínűséggel kell behúzni.

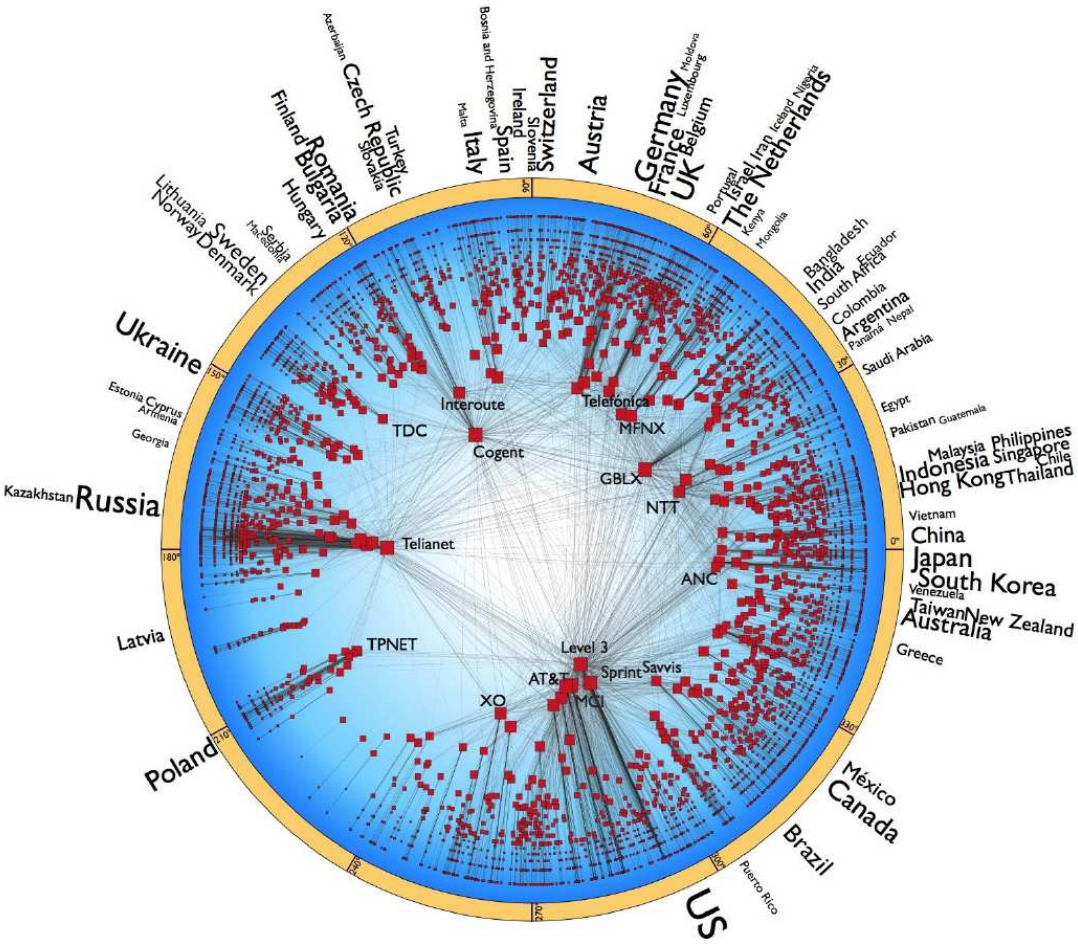
A mohó továbbítás szempontjából a leghatékonyabbnak az Internet hiperbolikus térbe ágyazása bizonyult [9]. A hiperbolikus geometria egy nemeuklideszi geometria, euklideszi térben csak modellek segítségével (például Poincaré disk modell [28]) tudjuk ábrázolni a hiperbolikus teret. A hiperbolikus geometria legfontosabb tulajdonsága, hogy a tér exponenciálisan bővül, tehát egy kétdimenziós r sugarú hiperbolikus körlap területe (ezt jelölöm $A(r)$-rel) r növelése esetén exponenciálisan nő, azaz $A(r) \sim e^r$. Így a hiperbolikus körlemezre egyenletesen szórva a csomópontokat a nagyobb sugár értékekhez exponenciálisan több csomópont tartozik. Az euklideszi térben ábrázolva a hiperbolikus körlapot (a Poincaré modell miatt) ez úgy jelenik meg, hogy a körlap szélein sűrűbben vannak a csomópontok. A Poincaré modell e furcsaságát ábrázolja a 2.9. ábra, ahol szabályos síkidomokkal van leparkettázva a hiperbolikus tér (a baloldalin háromszögekkel, a jobb oldalin ötszögekkel), a síkidomok mérete a hiperbolikus térben ugyanakkora (és a síkidomok oldalai is egyenesek a hiperbolikus térben).



2.9. ábra Hiperbolikus parkettázások [29]

A hiperbolikus körlemezre leszórt x_{ij} (hiperbolikus térbeli) távolságra lévő i és j pontokat az alábbi valószínűséggel köti össze ezután a modell: $p(x_{ij}) = \left(1 + e^{\frac{x_{ij} - R}{2T}}\right)^{-1}$, ahol R a körlemez sugara, $T \geq 0$ pedig a klaszterezettség befolyásolásához használt paraméter [9].

A hiperbolikus térbe ágyazás, tehát az AS-ek hiperbolikus térbeli koordinátáinak meghatározása ezután statisztikai következtetések segítségével történik. A beágyazó algoritmus minden lépésben növeli a hasonlóságot az előbb említett modell és az Internet AS szintű topológiája (ezt egy projekt keretében mérésekkel határozták meg [30]) között. A 2.10. ábrán látható a beágyazás eredménye, a későbbiekben ezt a hiperbolikus térbe ágyazott topológiát is használtam a méréseimhez. Érdekes, hogy a földrajzilag közel lévő AS-ek a hiperbolikus térbe ágyazva is közel esnek egymáshoz, bizonyítva a metrikus térbe ágyazás és földrajzi útvonalválasztás helyességét [9].



2.10. ábra Az Internet hiperbolikus térképe [9]

2.2.4 Szociális hálózatok

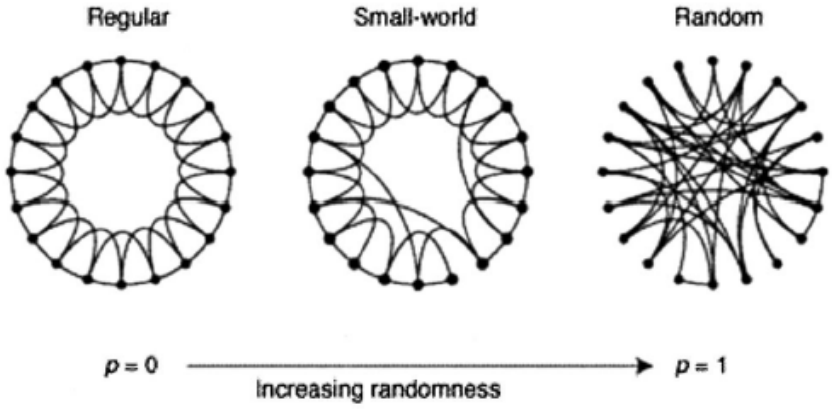
A internetes közösségi oldalak vizsgálata egy friss kutatási téma, azonban az emberi kapcsolatrendszer már régóta foglalkoztatja a kutatókat. Az egyik első és jelentős kísérlet Stanley Milgram szociálpszichológus nevéhez köthető. Ő azt vizsgálta, hogy két véletlenszerűen kiválasztott, az Egyesült Államokban élő ember között „mekkora távolság van”, azaz hány ismeretségi kapcsolaton keresztül lennének összeköthetőek [31].

A kísérletben résztvevőknek egy, az ország másik felén élő, általuk nem ismert címzetteknek kellett egy levelet továbbítani úgy, hogy csak általuk ismert embernek továbbíthatták a levelet (akiről úgy gondolták, hogy közelebb visz a cél személyhez), megkérve őket, hogy ők is hasonlóképpen tegyenek. A kísérlet eredménye szinte mindenkit meglepett: átlagosan 6 lépés elég az ismeretségi hálón, hogy egy tetszőleges embertől egy másik tetszőleges emberig eljussunk [32]. Ez volt a szociális hálók tanulmányozásához vezető első lépcső. A kísérletet később megismételték elektronikus levelek (e-mail) segítségével [33], és ott is hasonló eredmény született.

Sok idő (és hálózati modell) kellett, mire a kutatók rájöttek, hogyan is néznek ki valójában a szociális hálók [34]. A reguláris gráfok képezték az első hálózati modellek alapjait. Ezek a szabályos gráfok matematikailag jól megfoghatóak, minden pont fokszáma azonos, a pontok ugyanúgy viselkednek, ez jelentősen megkönnyíti ezen hálózatok vizsgálatát. A baj csak az, hogy a valóságban előforduló hálózatok távol állnak a modelltől. Ennek a modellnek gyakorlatilag az ellentétét képezi az Erdős-Rényi modell [35] és a véletlen gráfok elmélete. Az Erdős-Rényi modell két egymáshoz közel álló modellt jelenthet. A G(n, N) modell azt mondja, hogy vegyünk az összes lehetséges n csúcsú, N élű gráfok közül egyet véletlenszerűen, a G(n, p) modell esetén egy n csúcsú üres gráfból kiindulva minden lehetséges élet egymástól függetlenül, p valószínűséggel húzunk be. Az Erdős-Rényi modell már magyarázza a hálózaton belüli rövid útvonalakat (6 lépés távolság), azonban szociális hálózatok modellezésére nem alkalmas. Ez érthető, mivel az emberi kapcsolatok ritkán ennyire véletlenszerűek, a szociális hálókra jellemzőek a csoportok (klaszterek), amiken belül a tagok ismerik egymást.

Watts és Strogatz ezt a két modellt ötvözte egy közös modellben [36], észrevéve, hogy a „kis világ”, ahol vannak csoportok és távoli kapcsolatok is, e két modell közt

található (lásd 2.11. ábra). A modell egy szabályos gráfból indul ki (egy gyűrűből, ahol a csomópontok K szomszédjukkal vannak összekötve, mindkét irányba a K/2 legközelebbi csomóponttal), és aztán p valószínűséggel minden élet megváltoztat (azaz a párhuzamos és hurok éleket kerülve megváltoztatja az élek egyik végpontját). Ez a modell már sok mindenben hasonlít a szociális hálókra, de ez se írja le tökéletesen őket, ugyanis nem magyarázza a szociális hálókban lévő nagy fokszámú csomópontok meglétét (a sok ismerőssel rendelkező embereket), a modell fokszámeloszlása nem hasonlít a szociális hálózatokéra.



2.11. ábra A kis világok a véletlen gráfok és a szabályos gráfok közt [36]

Nem sokkal Watts és Strogatz modellje után következett a skálafüggetlen hálózatok felfedezése és a felismerés, hogy a szociális hálózatok is a skálafüggetlen hálózatok közé tartoznak [27]. A skálafüggetlen modell két lényeges dologban tért el a Watts-Strogatz modelltől, a hálózat létrejötte dinamikus, és a csomópontok közti kapcsolódás nem véletlenszerű, hanem a népszerűségi elv alapján történik. Ez a gyakorlatban úgy történik meg, hogy egy újonnan csatlakozó csomópont nagyobb valószínűséggel csatlakozik a nagyobb fokszámú csomópontokhoz, így a hálózat fokszámeloszlása hatványfüggvény szerinti lesz.

Az online közösségi hálózatokon végzett mérések igazolták, hogy ez a modell már pontosan leírja ezeket a hálózatokat, azaz az online közösségi hálózatokra jellemző a kis világ tulajdonság, a klaszterezettség, és a skálafüggetlenség (azaz a hatványfüggvény szerinti fokszámeloszlás) [37]. A modell és a mögötte álló gráfok megértése szükséges a jelenlegi szociális hálók kiértékeléséhez, újak tervezéséhez, valamint a szociális hálók Internetre gyakorolt hatásának megértéséhez. A modell megértése hozzásegíthet olyan algoritmusok létrehozásában, aminek segítségével a

szociális hálóban lévő megbízható és befolyásos felhasználók detektálhatóak [37], ahogy a világháló tanulmányozása vezetett a megbízható források megtalálását végző algoritmushoz [38]. A szociális hálók ismerete segíthet olyan algoritmusok fejlesztésében, amik segítik az email spam elkerülését [39], vagy javítják az Interneten való keresést [40].

Kleinberg mutatta meg, hogy a kis világ tulajdonsággal rendelkező hálózatokban mikor működik hatékonyan a mohó algoritmus, azaz mikor lehet hatékonyan megtalálni a hálózatban a rövid útvonalakat pusztán lokális információk alapján. Ahhoz, hogy egy kétdimenziós kis világ hálózatban hatékonyan működjön a mohó algoritmus, a klaszterezettségi kitevőnek (α) kettőnek kell lennie, azaz egy csomópontnak a közvetlen szomszédjai mellett egy távoli, tőle r távolságban lévő csomóponthoz $p = r^{-\alpha} = r^{-2}$ valószínűséggel kell csatlakoznia [41]. Ettől eltérő kisvilág tulajdonságú hálózatokon a mohó algoritmus nem tudja megtalálni a rövid utakat, mert ezek megtalálásához a hálózat globális ismerete szükséges.

Kleinberg azonban vizsgálataihoz egy négyzetrácson alapuló kisvilág hálózatot használt, ahol a csomópontok helye és a távolságok könnyen meghatározhatóak. A valós szociális hálózatokban azonban ez nem ilyen egyszerű. Több vizsgálat is foglalkozott a szociális hálókon való keresésekkel, azaz hogy az emberek hogy tudnak pusztán lokális információk alapján munkát, barátokat, szerelmet, vagy csak hasonló érdeklődésű embereket találni, és ezt hogy tudják algoritmusok segíteni az online szociális hálókon [42][43]. A rövid utak megtalálását jellemzően hierarchia vagy földrajzi elhelyezkedés segíti, például egy céges hálózatban a beosztotti hierarchia [43].

A társadalomban nem létezik azonban egy ilyen hierarchia. Több ilyen hierarchia van, egy-egy ilyen hierarchiában elfoglalt helyét a felhasználó identitásának nevezzük. Ezek a hierarchiák több szintűek, a hierarchiában lefelé haladva egyre kisebb, egyre specializáltabb csoportok találhatóak, mígnem eljutunk egy adott emberig. Ezek a hierarchiák segítik a szociális hálón a rövid útvonalak megtalálását [42].

A kérdés ezután már csak az, hogyan ágyazzuk metrikus térbe a szociális hálókat. A válasz hasonló az Internet beágyazásánál látottakhoz. Ismert a szociális hálókat leíró modell (tetszőleges méretben előállítható, már metrikus térbe ágyazva), a feladat a modell és a valós hálózat összeforrasztása. Erre a feladatra már ki lett dolgozva egy Markov-lánc Monte Carlo (MCMC) algoritmus [44], amivel előállíthatunk egy

metrikus térbe ágyazott hálózatot, pusztán a hálózatot leíró gráf megadásával. Az így kapott hálózaton már jól működik a hagyományos mohó továbbítás.

3 A tervezett algoritmus

3.1 A heurisztikák

3.1.1 Definíció

A heurisztika lassan, vagy pontosan nem megoldható problémák megoldására használt technika, mely bizonyos engedmények (optimalitás, teljesség, pontosság terén) mellett szolgáltat közelítő megoldásokat.

A földrajzi útvonalválasztásnál azért használok heurisztikát, hogy gyorsabban, kevesebb művelet segítségével, ezáltal erőforrás-takarékosabban találjak egy, az optimális útvonalhoz hasonló útvonalat. Ez különösen fontos nagyméretű, egyszerűbb eszközöket tartalmazó hálózatoknál. Ilyenek például a szenzorhálózatok, ahol az eszköz energia fogyasztását döntően határozza meg az adatok továbbítására használt energia. A heurisztika által talált útvonal nem feltétlenül optimális, így szükséges annak vizsgálata, hogy mennyire tér el az optimális útvonaltól.

3.1.2 Használatuk okai

A földrajzi útvonalválasztás egyik legnagyobb előnye, hogy nem eltárolt routing táblák alapján továbbít, ezáltal nincs szükség routing táblák nyilvántartására, megosztására és frissítésére. Cserébe minden továbbításkor meg kell hozni egy döntést a cél címe alapján, hogy merre történjen a továbbítás. Ezért fontos, hogy ennek a döntésnek a meghozása minél egyszerűbben és gyorsabban megtörténjen.

Mohó továbbításkor a döntés meghozása annál tovább tart, minél több opciót kell az algoritmusnak figyelembe vennie. Az opció továbbítás esetén a szomszédokat jelenti, azaz minél több szomszéddal rendelkezik egy csomópont, annál több továbbítási lehetőséget kell vizsgálnia. Ez azért különösen nagy probléma, mert jellemzően minél több szomszéddal rendelkezik egy adott csomópont a hálózaton, annál nagyobb forgalom megy át rajta, tehát annál többször kell futtatnia a mohó továbbító algoritmust.

3.1.3 Sorrend alapú heurisztikák

Amikor elkezdtem azon gondolkodni, hogyan lehetne a mohó továbbítást gyorsítani, az első gondolatom az volt, hogy lehetőleg minél kevesebb szomszéd

vizsgálatára legyen szükség a továbbításkor. Ez rögtön két kérdést is felvet: Ha nem vizsgálom az összes szomszédot, akkor mi alapján fogadok el egy szomszédot a továbbítás szempontjából jónak? Milyen sorrendbe vizsgálom a szomszédokat?

Ezen heurisztikáknál enyhítek az optimalitás feltételén, tehát továbbításkor nem követelem meg, hogy az optimális, a célhoz legközelebbi szomszédnak történjen a továbbítás, megelégszem azzal is, ha a továbbítással a célhoz közelebb jutunk. Ez a feltétel hasonlít a GEAR algoritmus [23] feltételéhez. Tehát a továbbítás feltétele sorrend alapú heurisztikáknál, hogy közelebb kerüljünk a cél csomóponthoz. Fontos megjegyezni, hogy a mohó továbbítás leállási feltétele megmaradt, tehát csak addig történik továbbítás, amíg a célhoz közelebb jutunk.

A sorrend alapú heurisztikák működésének lényege, hogy előre meghatározott sorrendben ellenőrzik az adott csomópont szomszédjait, és amint találnak egy olyan szomszédot, ami kielégíti a továbbítás feltételét (közelebb visz a célhoz), megtörténik a továbbítás. Minden csomóponthoz tartozik egy sorrend, ami a csomópont minden szomszédját pontosan egyszer tartalmazza, így minden szomszédot legfeljebb egyszer vizsgál továbbításkor a sorrend alapú heurisztika. A sorrend alapú heurisztikák működését mutatja az alábbi pszeudokód. A szomszéd_sorrend függvény adja vissza a csomópont szomszédjainak rendezett listáját.

```
Adott G(V,E) gráf; egy forrás és egy cél csomópont
begin
    aktuális = forrás
    előző = Ø
    while(aktuális != cél és előző != aktuális)
          akt_táv = távolság(aktuális, cél)
          lista = szomszéd_sorrend(aktuális)
          legközelebbi = aktuális
          while(lista != {Ø} és legközelebbi==aktuális)
                n = következő_elem(lista)
                szomsz_táv = távolság(n, cél)
                if (szomsz_táv < akt_táv)
                      legközelebbi = n
                end if
          end while
          előző = aktuális
          aktuális = legközelebbi
    end while
end
```

3. pszeudokód Sorrend alapú továbbítás

Mivel a gyengébb továbbítási feltételt legalább annyi, de jellemzően több csomópont is kielégíti, mint a mohó továbbításnál, és a mohó továbbításnál minden szomszéd ellenőrzése megtörténik, itt pedig csak addig vizsgáljuk a szomszédokat, amíg egy közelebbit nem találunk, ezért legfeljebb annyi, de jellemzően sokkal kevesebb szomszéd vizsgálata is elég. Az, hogy pontosan mennyi szomszéd vizsgálata szükséges egy csomópontban, az erősen függ a felállított sorrendtől.

A sorrend alapú heurisztikák eltérő útvonalakat adhatnak, és emiatt a továbbítás sikeressége is eltérő lehet a hagyományos mohó továbbításhoz képest, azonban ha egy csomópontban a hagyományos továbbítás sikeres, akkor a sorrend alapú heurisztika alapján történő továbbítás is sikeres lesz.

A továbbiakban megadok két sorrend alapú heurisztikát, amit implementáltam, azonban más sorrend alapú heurisztikák (például fokszám alapú heurisztika) is elképzelhetőek. A két megadott heurisztika a véletlen sorrend (RO, Random Order), és a forgalom alapú sorrend (TO, Traffic Order).

3.1.3.1 Véletlen sorrend (RO)

Véletlen (random) sorrend alapú heurisztikánál egy adott csomópontban a csomag továbbításakor a csomópont szomszédjait véletlen sorrendben vizsgálom. Nagy előnye az egyszerűsége, ugyanis a többi sorrend alapú heurisztikával ellentétben nem igényel bonyolult, előzetes sorrend felállítást, azonban a véletlen sorrendnél sokkal jobb sorrendek is vannak, ahol hamarabb találni egy megfelelő szomszédot.

Fontos megjegyezni, hogy nem véletlenszerűen sorsolja az adott csomópont szomszédjait vizsgálatra, hanem egy listán megy végig, ami véletlen sorrendben tartalmazza a szomszédokat. Bár nem tűnik lényegesnek az eltérés, mégis nagyon lényeges, mivel így lehet garantálni, hogy az algoritmus minden szomszédot maximum egyszer vizsgál, és egy adott csomópontban legfeljebb annyi vizsgálat történik, mint a hagyományos mohó továbbításnál.

3.1.3.2 Forgalom alapú sorrend (TO)

A forgalom alapú sorrendnél arra a gondolatra építek, hogyha felmérjük az éleken átmenő forgalmat, és a csomópont szomszédjait az oda vezető él forgalma alapján sorrendezzük (nagyobb forgalmú előrébb szerepel), akkor jobb eredményt kapunk a véletlen sorrendnél. Így a hálózaton előzetesen hagyományos mohó

továbbítást használva kereséseket (csomagtovábbításokat) végzek, melynek során rögzítem az éleken átmenő forgalmat. Fontos, hogy az előzetes keresés során ugyanolyan, véletlen kereséseket hajtok végre, mint amit majd a szimulációnál is tesztelni fogok, hogy jobban közelítsem a szimulációban is előforduló forgalmat. A forgalom felmérése után minden csomópontnál létrehozom a szomszédok sorrendjét tartalmazó listát.

Az előzetes, inicializáló keresések száma fontos paramétere a forgalom alapú sorrendnek, így érdemes vizsgálni, hogy ezt a paramétert változtatva mi történik.

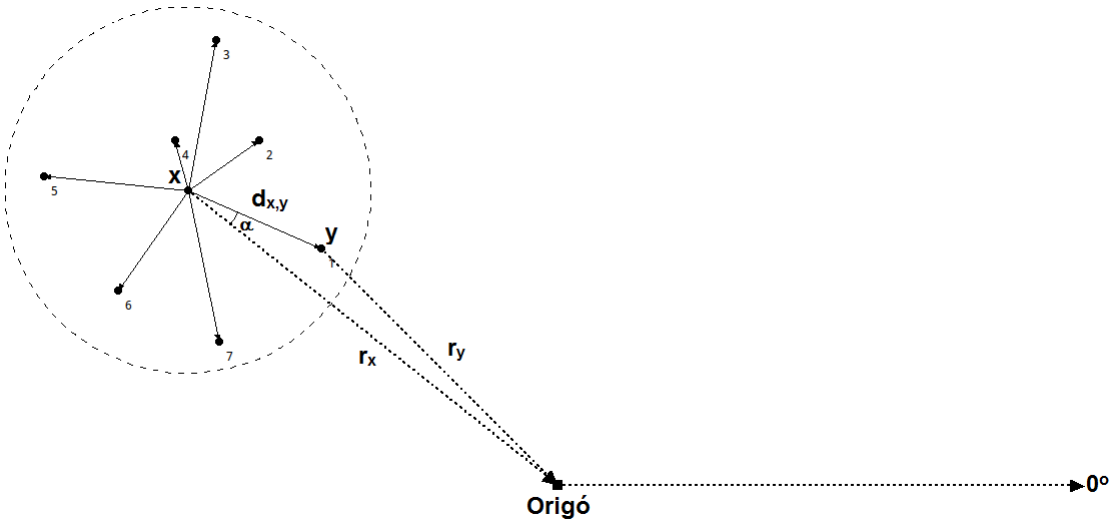
3.1.4 Irány alapú heurisztikák

3.1.4.1 Előzetes számítások

Az irány alapú heurisztika bizonyos szempontból hasonlít a sorrend alapú heurisztikákra, azonban egy teljesen eltérő ötletre épül. A hasonlóság alapja, hogy az irány alapú heurisztikánál is minden csomópontnak van egy rendezett listája a szomszédjairól.

Már a forgalom alapú heurisztika kitalálásakor is láttam, hogy nehéz egy kellően jó sorrend kialakítása. Ugyanis elvégezhetünk sok mohó keresést a hálózatban, amivel meg tudjuk állapítani, hogy általában merre kell továbbítani (és aztán ezt felhasználhatjuk vizsgálati sorrendként), azonban ez nem veszi figyelembe a konkrét esetet, hogy a keresés honnan jött, és merre tart.

Arra jutottam, hogy legyen egy olyan lista, ami irány szerint rendezve tartalmazza a csomópont szomszédjait. Ennek a listának a létrehozását szemlélteti a 3.1. ábra. Kell egy referencia irány (az origó iránya), és ehhez az irányhoz képest nézzük meg a csomópont szomszédjaiba vezető élek irányát, tehát hogy az él a referencia iránnyal milyen szöget (az ábrán α–val jelölve) zár be.



3.1. ábra Az irány szerint rendezett szomszédlista létrehozása

A csomópontok irány szerint rendezett szomszédlistájának előállítását mutatja az alábbi pszeudokód is:

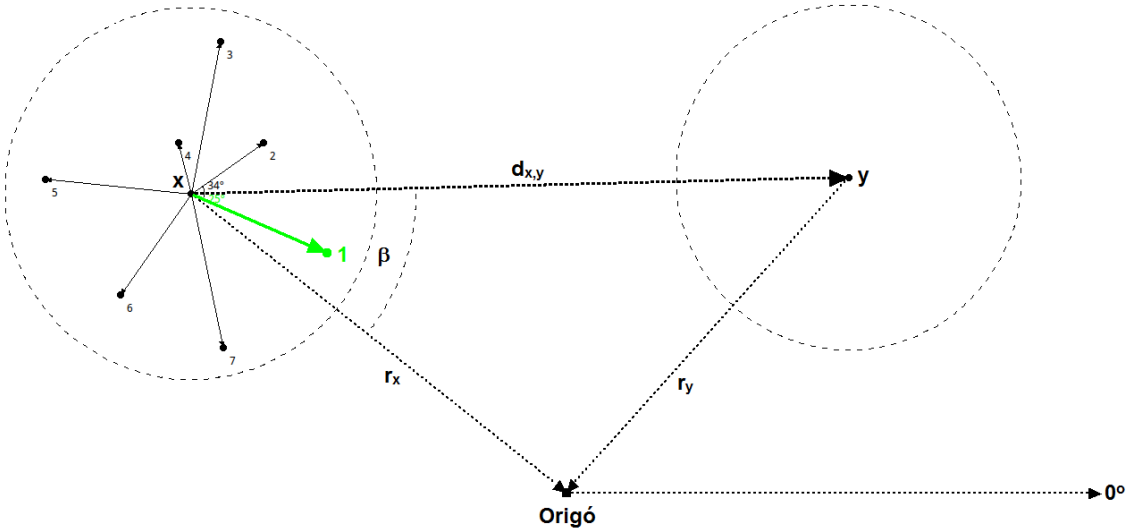
```
Adott G(V,E) gráf
begin
    for all v in V
          szomszéd_lista={}
          for szomszéd sz in szomszédjai(v)
                irány = irány_számítás(origo, v, sz)
                irány = irány_korrekció(v, sz, irány)
                beszúrás(szomszéd_lista, sz, irány)
          end for
    end for
end
```

4. pszeudokód Irány alapján rendezett szomszédlista létrehozása

A pszeudokódban az irány_számítás függvény végzi el a szögszámítást az origó, az aktuális csomópont, és a szomszédos csomópont közt. Ez a szögszámítás a koszinusz tételre vezethető vissza (ismert a háromszög három oldala, és az egyik csúcsánál lévő szögre vagyunk kíváncsiak), azonban a különböző metrikus terekben különbözőképpen kell számolni. Miután a kapott szög 0 és π közötti, a szomszédok listája pedig 0 és 2π közti, így korrigálni kell a csúcs és a szomszédjának helyzete alapján, ezt végzi az irány_korrekció függvény. A beszúrás függvény ezután beszúrja a csúcs szomszédjainak listájába a megfelelő helyre (ezt az angle paraméter határozza meg) az

adott szomszédot. Fontos, hogy olyan struktúrát kell létrehozni, amiben később az irány alapján tudunk keresni, és megtudni, hogy az adott irányba milyen szomszéd található.

Az irány alapú heurisztikánál ezután arra használjuk fel a rendezett listát, hogy továbbításnál gyorsan ki tudjunk választani egy adott irányba eső szomszédot. Ugyanis az irány alapú továbbítás a mohó továbbítással ellentétben meggyorsítható úgy, hogy a szomszédokat előre rendezzük egy referencia irányhoz képest, így elég a cél és a referencia irány közti különbséget kiszámolni és megkeresni az adott irányhoz legközelebb eső szomszédot a rendezett listában. A továbbítást szemlélteti a 3.2. ábra.



3.2. ábra Továbbítás az irány alapján rendezett szomszédlista segítségével

3.1.4.2 Tartományba eső szomszéd választása (RID)

Az irány alapú továbbításnál tapasztalt végtelen ciklusba kerülési problémát javítja a tartományba eső szomszéd választása módszer. Ez a módszer arra épül, hogy megnézi, hogy milyen szögtartományba eső szomszédok választásakor kerülünk közelebb a célhoz (így a továbbítás feltétele a sorrend alapú heurisztikáknál használtra hasonlít), majd e szomszédok közül egy véletlenszerűen választottnak továbbít. A következőkben ezt a módszert mutatom be részletesebben.

Adott, hogy egy csomópont mekkora sugarú körben ismeri a szomszédjait, erre az értékre a továbbiakban r_{conn}-ként hivatkozok. Azt is ki lehet számolni, hogy a csomópont (s) milyen messze van a céljától (t), ezt a továbbiakban d_{st}-nek nevezem (erre alapul a hagyományos mohó algoritmus). Az s csomópont körüli r_{conn} sugarú kör és a cél csomópont (t) körüli d_{st} sugarú kör által átfedésben lévő terület az a terület,

melyen az adott csomópont cél felé vezető szomszédjai lehetnek, a két kör két metszéspontja pedig megadja a tartomány két határpontját (mindenképpen két metszéspont van, mert a d_{st} sugarú kör átmegy az r_{conn} sugarú kör középpontján).

Az r_{conn} sugarú körlap két határpont közti körcikkelye megad egy tartományt, amely ugyan kisebb, mint az a terület, amin belül a cél felé közelebb eső szomszédjai vannak, ám ezen körcikkelybe eső szomszédok megtalálása rendkívül egyszerű, és így is egy kellően nagy tartományt kapunk, amit felhasználhatunk a továbbításhoz.

A cikkelyen lévő szomszédok kinyeréséhez az irány alapján rendezett szomszéd listát tudjuk használni, ehhez csak két szög előzetes kiszámolására van szükségünk. A határpont, s, t által meghatározott szöget jelölöm α-val, a t, s, origó által meghatározott szöget pedig β-val. A szomszédlistában a (β-α) és (β+α) közt lévő szögekhez tartozó szomszédok felé továbbítás esetén biztosan közelebb kerülünk a célhoz. Természetesen a listát úgy kell elképzelni, hogy az eleje és a vége össze van kapcsolva.

A továbbítás feltétele, hogy találjunk egy szomszédot a listán ebben a tartományban. Mivel ez a továbbítás is bizonyos esetekben végtelen ciklusba kerülhet (például ha egyáltalán nincs is út a célhoz), ezért amennyiben a továbbítások száma eléri egy limitet, az algoritmus leáll.

3.1.4.3 Kombinált továbbítás (CD)

Kombinált továbbítás esetén kombinálom a különböző továbbító algoritmusok előnyeit. Az útválasztás ugyanúgy indul, mint a legjobban egyező irányú szomszéd választásánál (BMD), azonban ha elakad, akkor más módszerrel folytatódik a továbbítás. Ennek előnye, hogy az esetek többségében nincs elakadás, és ilyenkor a BMD nagyon hatékonyan működik, azonban bizonyos esetekben a BMD nem tud sikeresen továbbítani, pedig ilyenkor más továbbító algoritmusok továbbra is sikeresen működnének.

Először is azt kell meghatározni, hogy mit nevezünk elakadásnak, és azt hogy veszi észre az algoritmus. A BMD-nél a végtelen továbbítás elkerülésére vezettem be a lépésszám korlátot, ezt itt is átveszem azzal a módosítással, hogy a lépésszám korlát elérésekor áttérek a másik továbbító algoritmusra. Ezen felül vizsgálom a RID-nél ismertetett tartományba esést is, és ha nincs a tartományba eső szomszéd, akkor szintén átváltok a másik továbbító algoritmusra. Fontos mindkét vizsgálat, ugyanis gyakran a lépésszám korlát elérése előtt kiderül, hogy a cél irányába nem lehetséges továbbítani

(nincs arrafelé szomszéd), ilyenkor számos felesleges továbbítást meg lehet spórolni. Azonban az is előfordulhat, hogy nem derül ki, hogy az adott tartományba nem lehet továbbítani, és csak a lépésszám elérésekor áll le az algoritmus. Például ha két csomópont is r_{conn} távolságon belül van a célhoz, azonban egyikük se ismeri a célt, akkor ők egymásnak oda-vissza továbbítanak, mivel úgy látják, hogy a másik belesik a cél tartományába.

Az elakadás észlelésekor tetszőleges továbbítási módra lehet váltani, érdemes figyelembe venni a használt topológiát, és megnézni, hogy a BMD milyen eredményeket ad az adott topológián, és ennek ismeretében kiválasztani az elakadás után használt továbbító algoritmust.

3.1.4.4 Irány közelebb (ST)

Ennél a heurisztikánál a legfontosabb módosítás az irány alapú továbbításhoz (BMD) képest, hogy minden továbbítás előtt megvizsgálja a csomópont, hogy az adott szomszédnak továbbítás közelebb visz-e a célhoz. Amennyiben nem, eltér az irány alapú továbbítástól, és próbál egy jobb döntést hozni. Ez esetben a szomszédok rendezett listájában a szomszédtól jobbra illetve balra eső szomszédok közül kiválasztja azt, amelyik irányban közelebb esik a cél irányához, és megvizsgálja, hogy az közelebb visz-e. Egészen addig lépked ebben a listában, amíg nem talál egy közelebbi szomszédot, vagy körbe nem ér. Ha nincs olyan szomszéd, aki közelebb vinne, az algoritmus leáll.

Nagy előnye ennek a heurisztikának, hogy a fenti leírt működés segítségével el lehet kerülni az irány alapú továbbításnál látott végtelen ciklusba kerülést, nem szükséges megadni lépésszám korlátot, és egy olyan hálózaton, ahol a mohó keresés mindig sikeres, ez a heurisztika is mindig sikeres lesz. Továbbá a kombinált továbbításhoz (CD) és az irány alapú továbbításhoz (BMD) képest itt nincs szükség se a lépésszám nyilvántartására (és a korlát elérésének figyelésére), se átállásra, azaz más továbbító algoritmusra váltásra hiba esetén. A heurisztika használatához viszont olyan adatstruktúrában kell tárolni a szomszédlistát, ami mindkét irányba járható (például kétirányú láncolt lista, tömb).

3.2 A szimulációs környezet

A továbbító algoritmusok működésének vizsgálatához egy szimulációs környezetet építettem, ahol hatékonyan lehet vizsgálni és összehasonlítani a továbbító algoritmusok működését.

3.2.1 A fejlesztői környezet

A szimulációkat egy tanszéki, Linux operációs rendszert futtató gépen végeztem. A forráskódokat C++ programozási nyelven írtam, a fejlesztéshez az igraph csomagot használtam. Az igraph egy hálózatkutatásra kifejlesztett, több programozási nyelven is elérhető szoftvercsomag, aminek segítségével hatékonyan lehet hálózatokat modellezni és vizsgálni [45]. Én az igraph C nyelven írt változatát használtam, a felhasznált függvények részletes leírása a dokumentációban [46] megtalálható. A statisztikai számítások végzéséhez és a szükséges ábrák generálásához több programot is igénybe vettem, például az R programozási nyelvet és fejlesztői környezetet (amihez szintén létezik igraph kiegészítés), és a Microsoft Excel programot.

3.2.1.1 A szimulációhoz épített keretrendszer

A különböző továbbítások vizsgálata egy keretrendszer segítségével történik. A program futtatásakor ez a keretrendszer indul el. A program értelmezi a kapott bemeneti paramétereket, majd beolvassa a hálózatot. Ezután elvégzi a heurisztikák által megkövetelt inicializálásokat, végrehajtja a forgalom alapú heurisztikához (TO) szükséges kereséseket, minden csomóponthoz létrehozza a szomszédok forgalom illetve irány szerint rendezett listáját. Ezután jön a program törzse, ahol a kereséseket végrehajtja a program. Véletlen pontpárokat sorsol a hálózatban, és ellenőrzi, hogy a sorsolt pontpárok között van-e út, és ha igen, megnézi, hogy milyen hosszú a legrövidebb útvonal a két pont közt. Amennyiben a pontok között van út, végrehajtja az útvonalkeresést a bemutatott továbbító algoritmusokkal. Ezután értékeli a kapott eredményeket.

3.2.1.2 A továbbító algoritmusok implementálása

A továbbító algoritmusok az útvonalkereső (útvonalválasztó) függvényekben vannak implementálva, amiket a keretrendszer hív a kisorsolt pontpárokkal. Az útvonalkereső függvények egy egységes struktúrában tartják nyilván az útvonalkeresés vizsgált paramétereit, amit a keresés után visszaadnak a keretrendszernek. A továbbító

algoritmusokat a korábbiakban leírtaknak megfelelően implementáltam, továbbá hibakeresés céljából hibakereső módban a program a továbbítás minden lépését kiírja a program kimenetére, hogy ellenőrizni lehessen az algoritmusok és implementálásuk helyességét.

3.2.1.3 A távolságszámító függvények

Az euklideszi, gömbi és hiperbolikus térben használt távolságszámító függvények vannak implementálva, így az ezekbe a terekbe ágyazott hálózatokon lehetséges futtatni a programot. A program indulásakor meg kell adni, hogy melyik távolságszámító függvény alapján történjen a távolságszámítás.

A távolságszámító függvények:

- Euklideszi tér:
  - Ha a pontok x, y koordinátákkal rendelkeznek:
    $a = (x_1, y_1),\ b = (x_2, y_2)$
    $d_{euclidean}(a,b) = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$
    Mivel a gyökfüggvény a pozitív számokon (négyzetszámok összege sosem negatív) monoton nő, ezért ha a cél csak a legközelebbi szomszéd megtalálása, akkor elhagyható a gyökvonás (kisebb szám négyzetgyöke kisebb)
  - Polár koordináta rendszer (a pontok sugárral (r) és szöggel (γ) vannak azonosítva):
    $a = (r_1, \gamma_1),\ \mathrm{b} = (\mathrm{r}_2, \gamma_2)$
    $x_1 = r_1 * \cos(\gamma_1),\ \mathrm{y}_1 = r_1 * \sin(\gamma_1)$
    $x_2 = r_2 * \cos(\gamma_2),\ \mathrm{y}_2 = r_2 * \sin(\gamma_2)$
    A koordináták átalakítása után a távolságszámítás a $d_{euclidean}$ képlet alapján történik.
- Hiperbolikus tér:
  - Poincaré disk modell [28] (a pontok x, y koordinátákkal rendelkeznek):

$$a = (x_1, y_1),\ b = (x_2, y_2)$$

$$d_{poincaré}(a,b) = \cosh^{-1}\left(1 + \left(2 * \frac{d_{euclidean}(a,b)}{(1 - d_{euclidean}(0,a)) * (1 - d_{euclidean}(0,b))}\right)\right)$$

- Hiperboloid modell [28] (a pontok sugárral (r) és szöggel (γ) vannak azonosítva):

$$a = (r_1, \gamma_1),\ \mathrm{b} = (\mathrm{r}_2, \gamma_2)$$

$$d_{hyperboloid}(a,b) = \cosh^{-1}(\cosh(r_1) * \cosh(r_2) - \sinh(r_1) * \sinh(r_2) * \cos(\gamma_1 - \gamma_2))$$

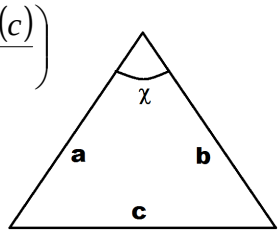
- Gömbi tér:
  - Ha a pontok sugárral (r) és szöggel (γ) vannak azonosítva, a gömb sugara pedig ρ ($\rho = 1$ alapértelmezésben):

$$a = (r_1, \gamma_1),\ \mathrm{b} = (\mathrm{r}_2, \gamma_2)$$

$$d_{spherical}(a,b) = \rho * \cos^{-1}\left(\cos\left(\frac{r_1}{\rho}\right) * \cos\left(\frac{r_2}{\rho}\right) + \sin\left(\frac{r_1}{\rho}\right) * \sin\left(\frac{r_2}{\rho}\right) * \cos(\gamma_1 - \gamma_2)\right)$$

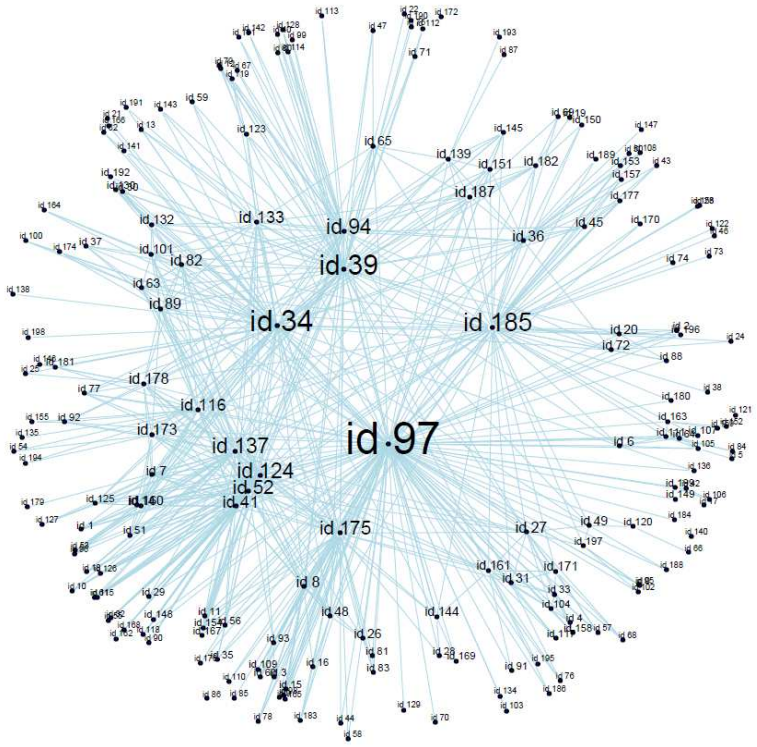
3.2.1.4 Szögszámító függvények

- Euklideszi tér: $\chi = \cos^{-1}\left(\frac{a^2 + b^2 - c^2}{2ab}\right)$
- Hiperbolikus tér: $\chi = \cos^{-1}\left(\frac{\cosh(a) * \cosh(b) - \cosh(c)}{\sinh(a) * \sinh(b)}\right)$
- Gömbi tér: $\chi = \cos^{-1}\left(\frac{\cos(c) - \cos(a) * \cos(b)}{\sin(a) * \sin(b)}\right)$



3.2.2 A vizsgált topológiák

Korábbi vizsgálataim során azt az eredményt kaptam, hogy a hiperbolikus terek használata előnyös a földrajzi útvonalválasztás és a topológiák építése szempontjából is, sok valós hálózat mögött is hasonló természetű terek húzódnak. E miatt a továbbító algoritmusok működését egy kisméretű (200 csomópontból álló), hiperbolikus térbe ágyazott hálózaton teszteltem.



3.3. ábra A továbbító algoritmusok teszteléséhez használt topológia

Egy ilyen méretű hálózaton még jól lehet követni a továbbító algoritmusok egyes lépésit, de már kellően nagyméretű ahhoz is, hogy a továbbító algoritmusok közti eltéréséket fel lehessen deríteni. Ezt a hálózatot mutatja a 3.3. ábra, a hálózatra a továbbiakban hyperbolic próbaként hivatkozok.

A továbbító algoritmusok működését kipróbáltam egy nagyobb (több mint 23000 csomópontot tartalmazó) hiperbolikus térbe ágyazott Internet topológiával is. Ez a topolólógia egy tudományos cikk [9] eredményeként állt elő, ahol azt vizsgálták, hogy az Internetet alkotó AS-ek (Autonomous System, azaz autonóm rendszer) hogyan kapcsolódnak egymáshoz, majd a kapott hálózatot egy speciális eljárással hiperbolikus térbe ágyazták. Ez a hálózat is nagyon jó alapot nyújt a továbbító algoritmusok vizsgálatához, erre a topológiára a továbbiakban hyperbolic AS topológiaként hivatkozok.

Továbbá kipróbáltam az algoritmust speciálisan generált hálózatokon is. Egy saját TOP_GEN algoritmussal paramétereiben valós hálózatokhoz hasonló, greedy routolható (ahol a mohó keresés minden esetben sikeres) topológiákat hoztam létre. Az alábbi pszeudokód mutatja a generáláshoz használt algoritmust:

```
Adott G(V,E) gráf, ahol E={}
for all v in V
    fed_távok={}
    for all u in V\v
          fed_távok=fed_távok ∪ min_k(d(k,v)|d(k,u)<d(v,u))
    end for
    max_táv=max(fed_távok)
    for all u in V\v
          if d(u,v)<max_táv
                E=E ∪ (u,v)
          end if
    end for
end for
```

5. pszeudokód Topológiagenerálás

Ezen algoritmus segítségével tetszőleges méretű hálózatok generálhatóak, teszteléshez 200 csomópontú hálózatokat, mérésekhez 10000 csomópontból álló euklideszi, hiperbolikus, és gömbi térbe elhelyezett hálózatokat generáltam. Ezen hálózatokon a mohó továbbítás minden esetben sikeres, így kiválóan alkalmasak a különböző heurisztikák kipróbálására és összehasonlítására. A fenti algoritmussal generált topológiákra a továbbiakban generált hálózat néven hivatkozok.

3.2.3 A vizsgált tulajdonságok

Ahhoz, hogy a továbbító algoritmusokat össze tudjuk vetni egymással, meg kell határozni, hogy az útvonalválasztás mely tulajdonságait vizsgáljuk. Ezeket a tulajdonságokat definiálom a következőkben.

Sikeres útvonalkeresések aránya

A szimulációnál véletlenszerűen sorsolt pontpárok között végzek útvonalkereséseket. A sikeres útvonalkeresések arányát úgy kapjuk meg, hogy a sikeres útvonalkeresések (amikor talált az adott továbbító algoritmus utat a két pont között) számát elosztjuk az összes útvonalkeresés számával (sikeresség = sikeres keresések száma / keresések száma). Az útvonalkeresések végrehajtása előtt a program ellenőrzi, hogy az adott két pont között létezik-e útvonal, és ha nincs útvonal, akkor nem hajt végre útvonalkeresést, így a sikeresség arányát nem rontják azok az esetek, amikor nem is létezik út a két pont között.

Átlagos úthossz

Az útvonalkeresések során kapott útvonalhosszak átlaga. Az útvonal hossza alatt a megtett lépések számát értem (nem pedig a metrikus térben megtett út hosszát). A továbbító algoritmusok összevethetőségének érdekében csak azok az útvonalkeresések számítanak bele az átlagba, amikor az összes vizsgált továbbító algoritmus sikeres útvonalkeresést eredményezett.

Elvégzett műveletek száma

Ahhoz, hogy össze lehessen vetni a továbbító algoritmusok komplexitását, bevezettem egy absztrakt művelet fogalmat (Operation, „O"). Mivel a ténylegesen elvégzett műveletek száma architektúra és implementációfüggő, emiatt nehezen meghatározható, így magasabb szintű műveleteket definiáltam. Ezek a hagyományos mohó továbbításnál a következőek:

Vizsgálat, hogy a cél a szomszédjaim közt van-e ($O_{sz}(x)$)

Ez a művelet akkor hajtódik végre, ha a beérkező csomag (vagy keresés) címzettje nem az aktuális csomópont. Ha az aktuális csomópont közvetlen szomszédja a cél csomópont, akkor ez az egyetlen művelet történik, és az aktuális csomópont továbbítja a csomagot a cél csomópontnak (ami a közvetlen szomszédja).

Távolságszámítás ($O_{tsz}(x,y)$)

Két csomópont (x, y) közti távolság számítása. Ehhez szükség van a két csomópont koordinátáira, valamint arra, hogy milyen metrikát és távolságszámítást használunk (ezt nevezem modellnek, a modell a szimuláció előtt rögzítésre kerül). Távolságszámításra akkor kerül sor, hogyha a csomag címzettje nem az aktuális csomópont, és nem is valamelyik szomszédja. Ilyenkor az aktuális csomópont először a saját, majd később a szomszédjai célhoz képesti távolságát számítja ki.

Távolságok összehasonlítása ($O_{th}(t_1,t_2)$)

Két távolság (t_1, t_2) összehasonlítása. Ha az aktuális csomóponthoz képest a szomszédos csomópontok mind távolabb vannak a céltól, akkor mohó továbbítás sikertelen. Ha van az aktuális csomópontnál közelebbi szomszéd, akkor a legközelebbi felé történik a továbbítás.

Egy csomag mohó továbbítása során az útvonal egy x pontjában elvégzett műveletek száma:

$$M(x)=\begin{cases} O_{sz}(x), \text{ha a cél x szomszédja} \\ O_{sz}(x)+O_{tsz}(x,cél)+\sum_{i=1}^{N}\left(O_{tsz}\left(n_x(i),cél\right)+O_{th}\left(t_{n_x(i)},t_{\min}\right)\right), \text{egyébként} \end{cases}$$

N: x szomszédjainak száma;

$n_x(i)$: x i-edik szomszédja;

t_{min}: az eddigi nézett csomópontok közül a legközelebbi távolsága (kezdetben x távolsága)

A sorrend alapú heurisztikáknál is ugyanezek a műveletek hajtódnak végre, az eltérés a műveletek számában adódik. Ugyanis a sorrend alapú heurisztikáknál a szomszédokat a rendezett lista alapján veszi sorba, és nem veszi minden esetben az összes (N darab) szomszédot, csak addig megy, ameddig egy közelebbit nem talál.

Az irány alapú továbbítás műveletszámának meghatározásához azonban új műveletek definiálása szükséges, ezek a műveletek a következőek:

Szög számítása ($O_{szög}(x,y)$)

Szögszámítást az irány és a tartomány meghatározásakor használok. A szögszámítás egy koszinusz tételre vezethető vissza, egy háromszög három oldalának ismeretében könnyen meghatározható egy adott csúcsnál lévő szöge a szögfüggvények segítségével. A háromszöget a két csúcs (x,y), és az origó adja.

Szomszéd választása ($O_{val}(d)$)

A csomópont szomszédjainak irány szerint rendezett listájából egy szomszéd választása. Ennél a műveletnél előjön a csomópontok közti különbség, mivel egy rendezett listából egy elem kivételéhez szükséges lépések száma az elemek számával logaritmikusan arányos, tehát ennek a műveletnek a konkrét lépésszáma függ attól, hogy az adott csomópontnak hány szomszédja van.

Az útvonalkeresések által igényelt processzoridő (futási idő)

Ugyan az útvonalkeresések által használt processzoridő mérése erősen függ az útvonalkeresések, és az onnan hívott függvények implementálásától, azonban ez az adat áll a legközelebb a valósághoz, ez mutatja meg legjobban a továbbító algoritmusok

számításigénye közti különbséget (nem olyan absztrakt, mint az előbb definiált műveletek). A szimulációban így minden továbbító algoritmussal történő útvonalkeresés futtatása előtt és után lekérdezem a processzortól az aktuális időt, és amennyiben az összes továbbító algoritmussal sikeres útvonalkeresés történt (erre az összehasonlíthatóság miatt van szükség), az eltelt processzoridőt rögzítem. A processzor idő (clock tickben, azaz órajelben mérve) és a ténylegesen eltelt idő közti kapcsolatot a „CLOCKS_PER_SEC” nevű konstans (de különböző gépeken különböző értékű) változó teremti meg, a processzoridőt leosztva ezzel a konstanssal megkapjuk a futási idejét másodpercben.

3.3 A szimuláció eredménye

3.3.1 Az inicializálásokra fordított idő

Ugyan csak egyszer kell elvégezni, de nem elhanyagolható az az időráfordítás, amit az inicializálásra szánunk. Az 3.1. táblázat azt mutatja, hogy a szimuláció során mennyi időbe telt a különböző továbbításokhoz használt inicializálások végrehajtása (a teljes idő, az összes csomópontra). A hagyományos mohó továbbításnál, a véletlen heurisztikánál, és a szomszédlista nélküli irány alapú továbbításnál nincs szükség ilyen inicializálásra.

A legjobban egyező irányú szomszéd választása esetén (BMD) csak az irány alapú sorrend létrehozására van szükség, a tartományba eső szomszéd (RID) és a kombinált továbbítás (CD) választása esetén szükség van a tartomány meghatározása miatt az összeköttetési sugár számítására is, így ezen idők összegét kell az inicializálásra szánni. Az irány közelebb (ST) heurisztika két irányban bejárható szomszédlistát követel meg, azonban egy ilyen lista létrehozása elhanyagolható plusz költséggel rendelkezik a normál szomszédlista készítéséhez képest.

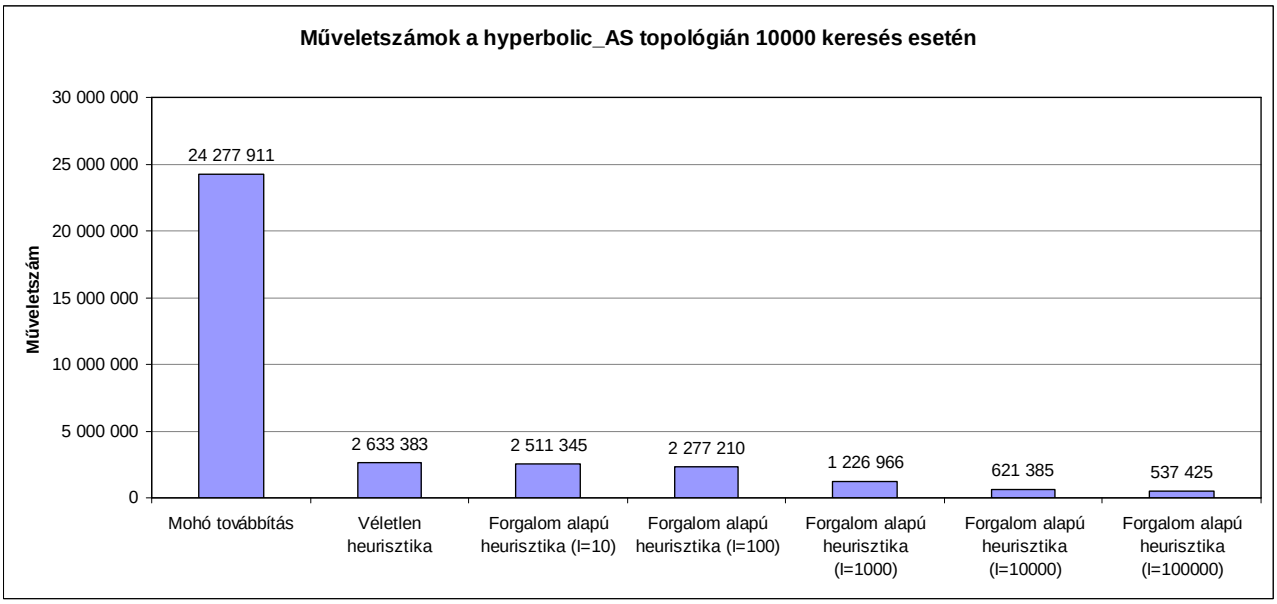
A forgalom alapú heurisztikánál látható, hogy 100 inicializáló keresés esetén nagyjából ugyanannyi az inicializálásra fordított idő, mint a RID vagy a CD esetén. A 3.4. ábra alapján azt kapjuk, hogy legalább 1000 inicializáló keresésre van szükség ahhoz, hogy a forgalom alapú heurisztika a véletlen heurisztikánál lényegesen kevesebb műveletet végezzen, az 3.1. táblázatból pedig az is kiderül, hogy ehhez nagyjából négyszer annyi időre van szüksége, mint a RID-nek vagy CD-nek.

Inicializálási idők Hyperbolic_AS gráfon	
Inicializálás	Szükséges idő (micro sec)
Irány alapú szomszédlista	134 901
Irány alapú szomszédlista (mindkét irányba bejárható)	135 790
Összeköttetési sugár számítás	45 161
Irány alapú szomszédlista + Összeköttetési sugár számítás	180 062
Forgalom alapú heurisztika (I=10)	103 628
Forgalom alapú heurisztika (I=100)	167 936
Forgalom alapú heurisztika (I=1000)	787 059
Forgalom alapú heurisztika (I=10000)	7 110 388
Forgalom alapú heurisztika (I=100000)	68 825 518

3.1. Táblázat A továbbító algoritmusok inicializálására fordított idők a Hyperbolic AS gráfon

3.3.2 Az inicializáló keresések hatása a TO útvonalválasztásra

A forgalom alapú sorrendet használó heurisztikánál rendkívül fontos, hogy hány inicializáló keresést hajtunk végre, mielőtt a heurisztikát élesben használnánk. A 3.4. ábrán látható, hogy a véletlen heurisztika nagyjából tizedére csökkenti a műveletek számát a mohó továbbításhoz képest valós hiperbolikus térbe ágyazott AS topológia esetében. Amennyiben az inicializáló keresések száma (ezt I-vel jelölöm) elhanyagolható (I=10), akkor a forgalom alapú heurisztika a véletlen heurisztikához hasonló eredményt ad. Amennyiben növeljük az inicializáló keresések számát, a forgalom alapú heurisztika annál jobb eredményt ad, a műveletszám akár a mohó továbbításnál tapasztalt 2%-ára csökkenthető, ez látható az 3.4. ábrán.

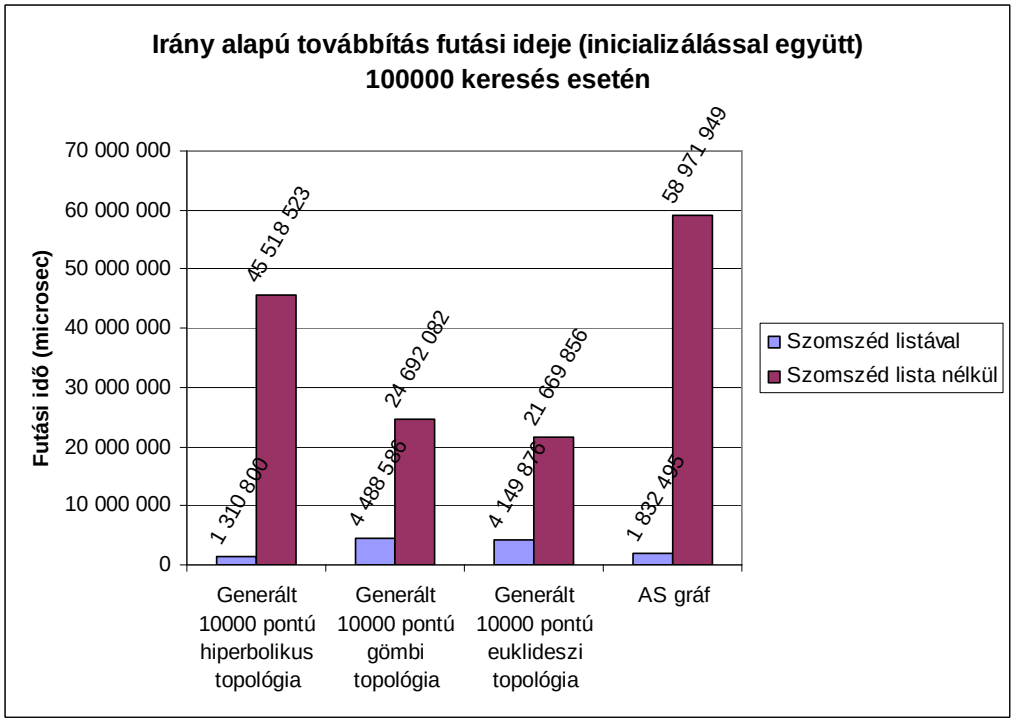


3.4. ábra Műveletszámok az AS hálózaton 10000 keresés esetén (sorrend alapú heurisztikák)

3.3.3 A szomszédlistával elérhető nyereség irány alapú továbbításnál

Az irány alapú továbbítás két implementációját is vizsgáltam. A két implementáció egyetlen dologban tért el egymástól: hogyan választom ki azt a szomszédomat, aki a legjobb irányba esik a cél felé. A hagyományos változatnál nincs semmilyen előre kiszámított és letárolt szomszédlista (ez esetben az inicializálással se kell foglalkozni). A listás változatnál a szomszédok irányát előre kiszámítjuk, és eltároljuk egy rendezett szomszédlistában. Továbbításkor ez esetben csak a cél csomópont irányát kell számítani, a szomszédok irányát már nem, az egy gyorsan kereshető listában el van tárolva. A két változat csak a számítási igényben (és ezáltal a futási időben) tér el egymástól, a kapott útvonalak megegyeznek, azaz se a sikerességbe, se az átlagos úthosszban nincs különbség a két változat közt.

A 3.5. ábrán látható a két változat futási idejének különbsége. A grafikonnál a futási időbe beleszámítottam az egyszeri, inicializálásra történő időráfordítást is, ami a listás algoritmus futtatásához szükséges. Minél nagyobb fokszámú csomópontok vannak egy hálózatban, a lista használatával annál több számítást (és időt) tudunk megspórolni a keresések során. A hiperbolikus térbe ágyazott topológiák nagyobb átlagos fokszámmal és rövidebb átlagos úthosszal rendelkeznek, emiatt jelentősebb a futási idő csökkenés a gömbi és az euklideszi térbe ágyazott topológiákhoz képest.



3.5. ábra Irány alapú továbbítás futási ideje (inicializálással együtt) 100000 keresés esetén

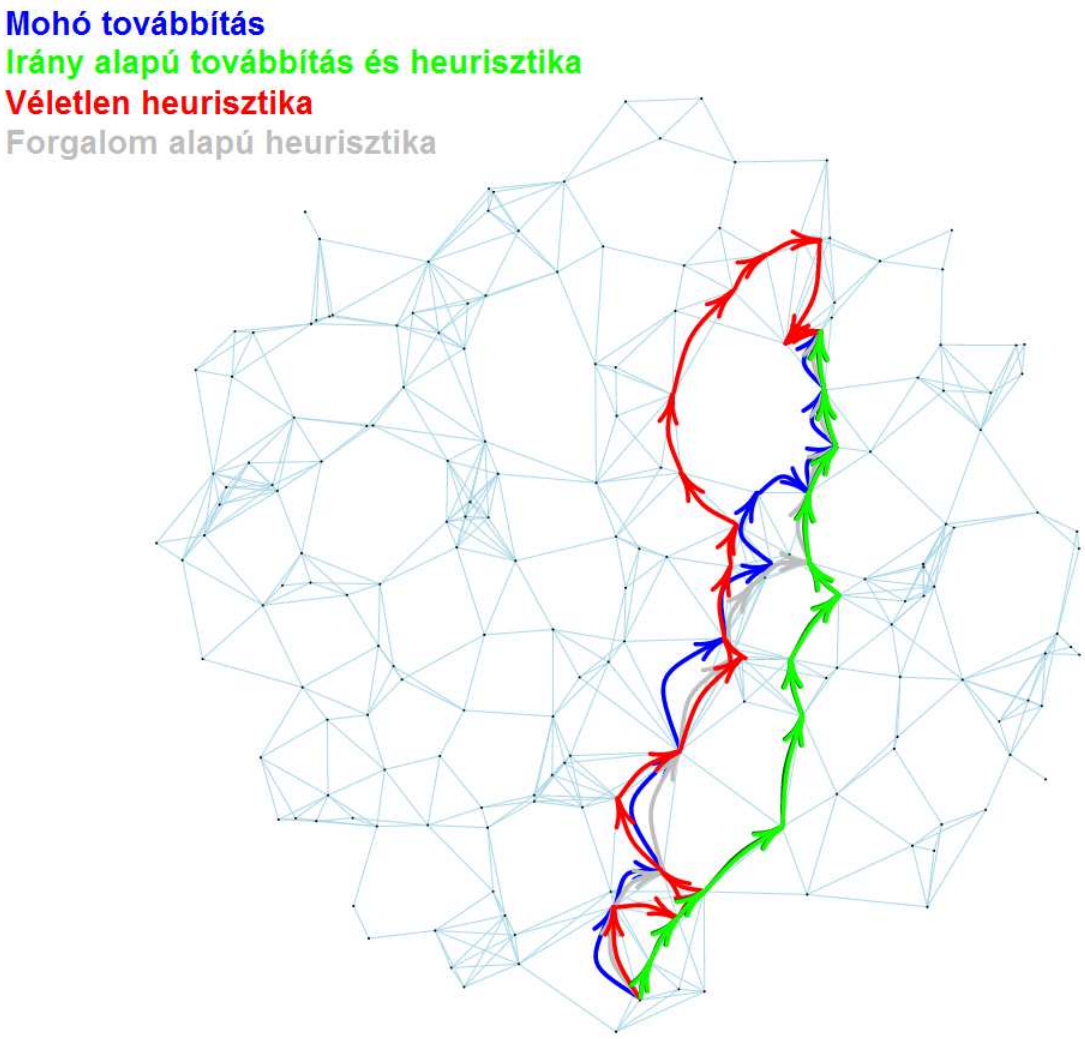
3.3.4 A továbbító algoritmusok összehasonlítása

A könnyebb követhetőség érdekében a 3.2. táblázatban összefoglaltam a szimulációkban szereplő továbbító algoritmusok rövidítéseit és a hivatkozásokban használt neveket.

Rövidítés	Feloldás, használt nevek
BMD	Best Matching Direction; Compass routing; Irány alapú továbbítás; Irány (listával); Irány (lista nélkül)
CD	Combined Direction; Kombinált továbbítás
CDG	Combined Direction Greedy; Mohó továbbítással kombinált továbbítás
CDRO	Combined Direction Random Order; Véletlen heurisztikával kombinált továbbítás
CDTO	Combined Direction Traffic Order; Forgalom alapú sorrend heurisztikával kombinált továbbítás
RID	Random In Direction; Tartományba eső szomszéd (véletlenszerű) választása
RO	Random Order; Véletlen sorrend alapú heurisztika
ST	Irány alapú heurisztika
TO	Traffic Order; Forgalom alapú heurisztikia; Forgalom alapú sorrend heurisztika

3.2. Táblázat A használt rövidítéseim és feloldásuk

A továbbító algoritmusok különbségeit jól szemlélteti a 3.6. ábra. A 3.6. ábrán egy generált, euklideszi térbe ágyazott, 200 csomópontú hálózatot láthatunk, valamint különböző továbbítási módszerek segítségével talált útvonalakat két rögzített pont között. Zölddel vannak jelölve az ábrán az irány alapú továbbítás különböző változatai (irány alapú továbbítás, CD, ST). Ezek közt az ábrán nincs eltérés, mivel a heurisztikák (CD, ST) a sikertelen irány alapú továbbítás javítására lettek kidolgozva, az ábrán viszont egy olyan keresés látható, ahol az irány alapú továbbítás is sikeres. Az ábrán és a későbbi összehasonlításokban sem szerepel a RID algoritmus, mert az az alacsony sikeressége miatt nehezen összevethető a többi algoritmussal. A RID alacsony sikerességét a túl nagy megengedett bolyongás (bármely tartományba eső szomszéd elfogadása továbbításkor) okozza. Jól látható az ábrán, hogy irány alapú továbbításkor az algoritmus a lehető legkisebb irány eltéréssel továbbít a cél felé, a mohó algoritmus (kék) pedig minden továbbításkor a lehető legnagyobb lépést teszi a cél felé. A véletlen heurisztika (piros) hosszabb, és a mohó továbbítástól jelentősebb mértékben eltérő útvonalat produkál, a forgalom alapú heurisztika (szürke) viszont az előzetes tanításnak (mohó inicializáló keresések) köszönhetően a mohó továbbításhoz egészen hasonló eredményt ad.



3.6. ábra Továbbítás különböző algoritmusokkal

A 3.3. táblázat mutatja a mért eredményeket négy különböző topológiára. A négy topológia három 10000 csomópontú, különböző (hiperbolikus, gömbi /szférikus/, euklideszi) térben TOP_GEN algoritmus segítségével generált, valamint az AS hálózatból áll. A táblázatban 100000 útvonalkeresésre vonatkozó eredmények találhatóak, mely eredmények a későbbi elemzéseim alapját adják. A szimulációban szereplő eredményeknél a kombinált továbbításnál (CD) az irány alapú továbbítást hagyományos mohó továbbítással kombináltam (CDG). A legrövidebb út oszlop azt mutatja, hogy a kereséseknél mennyi lett volna a minden esetben a létező legrövidebb útvonalat megtaláló algoritmusnál az átlagos úthossz. Ilyen (shortest path) algoritmust nem implementáltam (mivel rendkívül memória- és számításigényes), ennek kiszámítására az igraphot használtam.

Az eredményekbe az összevethetőség érdekében csak azon útvonal keresések eredményét vettem be, ahol minden továbbító algoritmus sikeresen tudott továbbítani. Ha ez nem így lenne, akkor egy, a távolabbi csomópontok közt is útvonalakat találó algoritmus rosszabb eredményeket érne el átlagos úthossz szempontjából egy olyan algoritmusnál, ami csak közeli pontok közt képes útvonalakat találni. Emiatt nem tartalmazza a (mérés és a) táblázat a RID algoritmus adatait, mert az csak nagyon kis részében volt sikeres, ami jelentősen torzítaná az összehasonlítást. A táblázatban található eredményeken megyek végig az alfejezetekben.

Generált 10000 pontú hiperbolikus topológia, 100000 keresés	Mohó	Sorrend alapú heurisztika		Kombinált	Irány alapú heurisztika	Irány alapú továbbítás (limit=100)		Legrövidebb út:
		RO	TO (l=1000)	CDG	ST	listával	lista nélkül	
Inicializálási idő (micro sec)	0	0	303 071	133 091	100 327	95 879	0	
Futási idő (micro sec)	23 576 901	4 932 692	5 051 987	948 254	2 157 281	1 214 921	45 518 523	
Műveletek száma	107 490 432	20 467 076	18 281 930	1 327 149	1 405 642	1 080 752	109 425 792	
Sikeres keresések aránya	1	1	1	1	1	0.9516		
Átlagos úthossz	3.42	6.94	6.33	3.59	3.56	3.59		3.36

Generált 10000 pontú gömbi topológia, 100000 keresés	Mohó	Sorrend alapú heurisztika		Kombinált	Irány alapú heurisztika	Irány alapú továbbítás (limit=100)		Legrövidebb út:
		RO	TO (l=1000)	CDG	ST	listával	lista nélkül	
Inicializálási idő (micro sec)	0	0	142 605	71 980	49 215	48 416	0	
Futási idő (micro sec)	12 046 318	7 413 095	5 780 737	5 366 145	6 478 766	4 440 170	24 692 082	
Műveletek száma	70 213 675	41 150 121	26 911 203	30 200 534	22 835 659	18 286 179	86 514 147	
Sikeres keresések aránya	1	1	1	1	1	0.9938		
Átlagos úthossz	40.53	63.59	49.32	45.87	46.75	46.75		36.15

Generált 10000 pontú euklideszi topológia, 100000 keresés	Mohó	Sorrend alapú heurisztika		Kombinált	Irány alapú heurisztika	Irány alapú továbbítás (limit=100)		Legrövidebb út:
		RO	TO (l=1000)	CDG	ST	listával	lista nélkül	
Inicializálási idő (micro sec)	0	0	140 193	67 455	44 453	43 694	0	
Futási idő (micro sec)	12 125 175	7 565 661	5 978 371	4 782 411	6 017 555	4 106 182	21 669 856	
Műveletek száma	71 239 323	41 921 689	28 049 369	30 658 299	23 120 570	18 513 759	87 651 965	
Sikeres keresések aránya	1	1	1	1	1	0.9979		
Átlagos úthossz	40.99	64.31	50.69	46.28	47.13	47.13		36.49

asgraph, 100000 keresés	Mohó	Sorrend alapú heurisztika		Kombinált	Irány alapú heurisztika	Irány alapú továbbítás (limit=100)		Legrövidebb út:
		RO	TO (l=1000)	CDG	ST	listával	lista nélkül	
Inicializálási idő (micro sec)	0	0	862 361	183 398	137 553	136 295	0	
Futási idő (micro sec)	44 190 764	6 116 092	4 035 018	1 243 534	2 905 885	1 696 200	58 971 949	
Műveletek száma	188 550 758	23 754 934	12 974 032	1 672 377	1 576 702	749 016	159 414 366	
Sikeres keresések aránya	0.8178	0.8933	0.8864	0.8714	0.8344	0.7893		
Átlagos úthossz	3.65	5.66	5.41	3.99	4.04	3.99		3.46

3.3. Táblázat A továbbító algoritmusok eredményei különböző topológiákon

3.3.4.1 Sikeres útvonalkeresések aránya

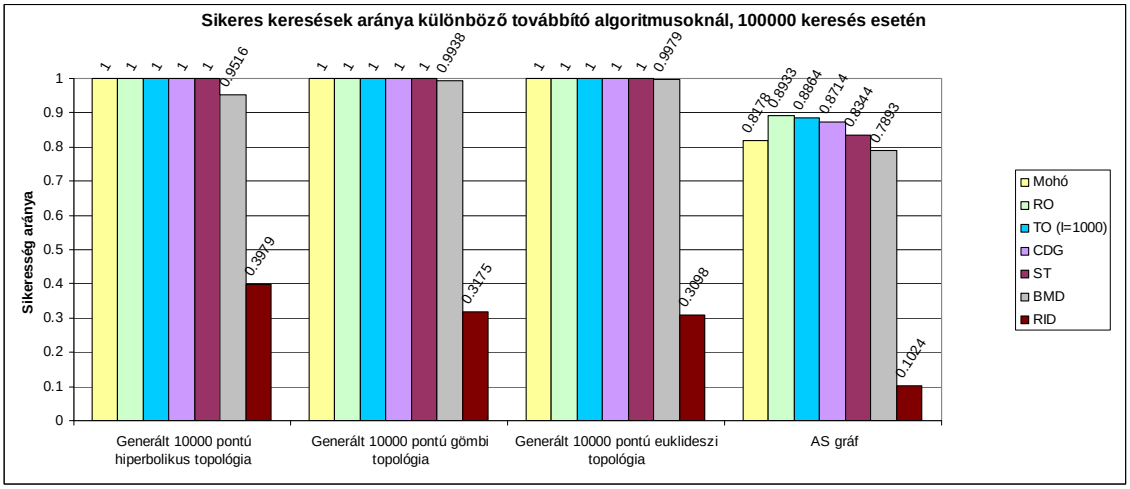
A generált hálózatokon az irány alapú továbbítást leszámítva a táblázatban szereplő továbbító algoritmusok minden esetben sikeresen működtek, az irány alapú továbbítás az esetek 0.1-5%-ban volt sikertelen. Ennek oka, hogy egy greedy routolható hálózaton nem feltétlen sikeres az irány alapú továbbítás minden esetben. Az irány alapú továbbítás (BMD) sikertelenségének oka főleg arra vezethető vissza, hogy a legjobban egyező irány választása esetén nincs mindig út a cél felé, és ilyenkor kettő vagy több csomópont egymásnak továbbít ciklikusan egészen addig, amíg a lépésszám limitet el nem éri a továbbítás.

A kombinált továbbítás és az irány alapú heurisztika azonban két jó megoldást mutat ennek a problémának a megoldására. A kombinált továbbításnál a hibadetektálásra megoldás a vizsgálat, hogy az irány alapján választott szomszéd a csomópont célhoz közelebb vivő tartományában van-e, valamint a lépésszám korlát figyelése, a javítás pedig úgy történik, hogy ilyenkor a továbbításnál átállok egy másik továbbító algoritmusra. Az irány alapú heurisztikánál egy ennél egyszerűbb (de számításigényesebb) vizsgálat detektálja és javítja a hibát. Itt azt vizsgálom, hogy az adott szomszéd kiválasztása után vajon közelebb kerülnék-e a célhoz, és ha nem, elkezdem sorba vizsgálni a szomszédjaimat, míg egy közelebbit nem találok. Ez egy egyszerűbb, kevesebb nyilvántartást igénylő módszer.

Ezeknél is egyszerűbb megoldást adna a BMD-nél látott végtelen ciklusba kerülésre a tartományba eső szomszéd (RID) módszer, azonban ez hozta sikeresség szempontjából a legrosszabb eredményt. Az alig 200 csomópontból álló próba topológián is az esetek 19%-ában elbukott. Ez két dologra vezethető vissza. Egyfelől arra, hogy a célhoz közelebb vivő tartomány bizonyos esetekben túl nagy, és a tartományba eső szomszédok közül egy véletlen választott szomszéd (ami nem feltétlen a célhoz legközelebbi, vagy legjobban irányba eső szomszéd) miatt a továbbító algoritmus ”bolyongásba kezd”, azaz túl hosszú utat jár be, ami miatt elérheti a lépésszám limitet. A másik ok pedig, hogy a vizsgált tartomány bizonyos esetekben kicsi is lehet, ugyanis a vizsgálat egyszerűsítése érdekében kisebb a vizsgált terület, mint amin ténylegesen elhelyezkedhetnek a célhoz közelebb vivő szomszédok, és emiatt előfordulhat, hogy ezen a kisebb területen nincs szomszéd, míg a tényleges tartományban volna.

A 23748 csomópontból álló AS topológián az előbb bemutatottak úgy módosulnak, hogy itt azok a továbbítások se sikeresek minden esetben, amik a generált topológiákon mindig sikeresek voltak. Ez a miatt van, hogy a mohó algoritmus ezen a topológián lokális optimumokba tud szorulni. Érdekes megfigyelni, hogy a heurisztikák és a kombinált algoritmusok is egy kicsit jobb eredményt érnek el, mint a hagyományos greedy, ez valószínűleg annak köszönhető, hogy a lazább továbbítási feltétel (CD és ST esetén pedig az alapvetőn irány alapú továbbítás miatt), ezek az algoritmusok jobban elkerülik a lokális minimumokat. A RID ezen a topológián még a próba topológiához képest is sokkal rosszabb eredményt ér el (mindössze 0.1 volt a sikeresség, azaz az esetek 90%-ban nem talált útvonalat), ez valószínűleg a topológia méretének

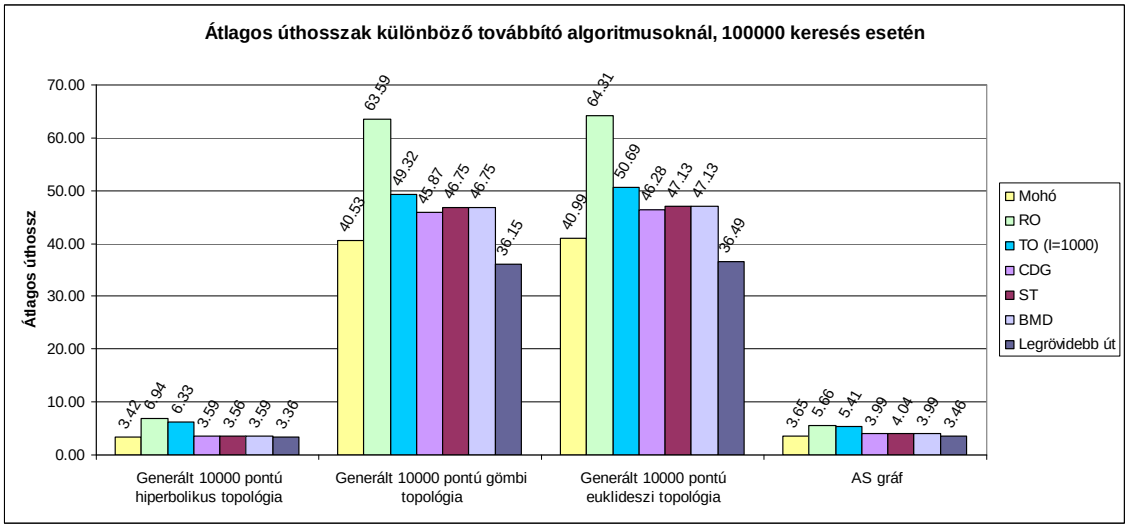
növekedése, és ez által a lehetséges bolyongás lehetőségének növekedése miatt van. Mivel az összehasonlíthatóság érdekében csak a minden továbbításnál sikeres keresések szerepelnek a táblázatban, ezért a vizsgálatoknál az alacsony sikeresség miatt a RID vizsgálatát mellőztem. A sikerességre kapott eredményeket grafikus formában a 3.7. ábra mutatja.



3.7. ábra A sikeres keresések aránya

3.3.4.2 Átlagos úthossz

Az átlagos úthosszra azt kapjuk, hogy a hagyományos mohó továbbítás úthossza közelíti meg legjobban a létező legrövidebb (shortest) út hosszát, ez főleg a szigorúbb továbbítási feltételnek köszönhető. Hasonlóan jó eredményt adnak az irány alapú továbbításon alapuló algoritmusok (BMD, ST, CD, de a RID nem), azaz az irány alapú továbbítással és a rá épülő heurisztikákkal lehet a mohó továbbításnál kapott útvonalakhoz hasonlóan rövid utat találni. Ezek között azért nem látható lényeges eltérés az adatoknál, mert a táblázat csak azon keresésekre vonatkozó adatokat mutat, ahol minden továbbító algoritmus sikeres volt, és sikeres irány alapú továbbítás esetén csak ritka esetekben (ha irány alapú továbbítást hibásnak véli és felülbírálja CD vagy ST, de később az irány alapú továbbítás mégis sikeres lesz) van eltérés a két algoritmus által adott útvonal között.



3.8. ábra Átlagos úthosszak

A sorrend alapú heurisztikák ezeknél (BMD, ST, CD) rosszabb eredményt adnak, közülük a forgalom alapú teljesít jobban. A RID átlagos úthossza is rossz, különösebben a nagyobb AS topológián, ennek oka is a már korábban említett bolyongásra vezethető vissza. Az átlagos úthosszra vonatkozó eredmények a 3.8. ábrán láthatóak (A RID az alacsony sikeressége miatt az összehasonlításból kimaradt).

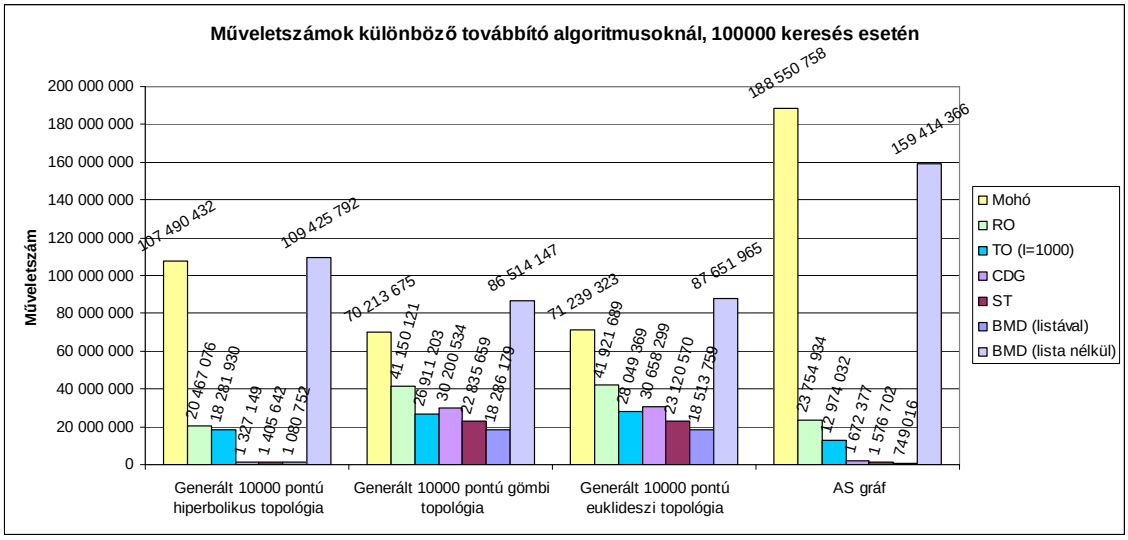
3.3.4.3 Műveletek száma

Az útvonalkeresés során elvégzett műveletek számát a 3.9. ábra mutatja. Az ábrára rátekintve rögtön látszik, hogy a heurisztikák mennyire le tudják csökkenteni a szükséges műveletek számát. A különbség különösen az AS topológiánál jelentős, ez az AS topológiánál lévő magas fokszámú csomópontokra vezethető vissza. A mohó továbbítás és az irány alapú továbbítás (BMD) közt nincs jelentős különbség műveletigény szempontjából, ha a BMD-nél nem használunk rendezett szomszédlistát. Azonban a BMD műveletigénye drasztikusan csökkenthető a szomszédlista segítségével, így ez válik a leghatékonyabb algoritmussá.

Az irány alapú heurisztika (ST) és a kombinált továbbítás (CD) feladata a BMD sikerességének a növelése, ami nyilván bonyolultabb algoritmust (és ezáltal több műveletet) követel meg, ezt figyelembe véve érthető, hogy kicsit magasabb a műveletszámuk a BMD-hez képest. Nem szabad elfelejteni, hogy az ábra csak olyan kereséseket mutat, ahol minden továbbító algoritmus sikeres volt, így a BMD-nek kedvez, hisz nem mutatja a sikertelen továbbításoknál végzett műveletek számát,

amikor látszódna a bonyolultabb továbbító algoritmusok előnye (például a csomagtovábbítás nem kerülhet ciklusba, felesleges továbbításokat generálva).

Az irány alapú heurisztikánál (ST) és a kombinált továbbításnál (CD) ugyan rosszabbak, azonban a mohó és a lista nélküli BMD-hez képest sokkal jobb eredményt érnek el a sorrend alapú heurisztikák. A véletlen heurisztikánál (RO) jobb eredményt ér el a forgalom alapú heurisztika, fontos azonban itt is megemlítenem, hogy az inicializáló keresések száma milyen jelentős hatással van a forgalom alapú heurisztikára, és növelésével jelentősen csökkenthető a műveletek száma.

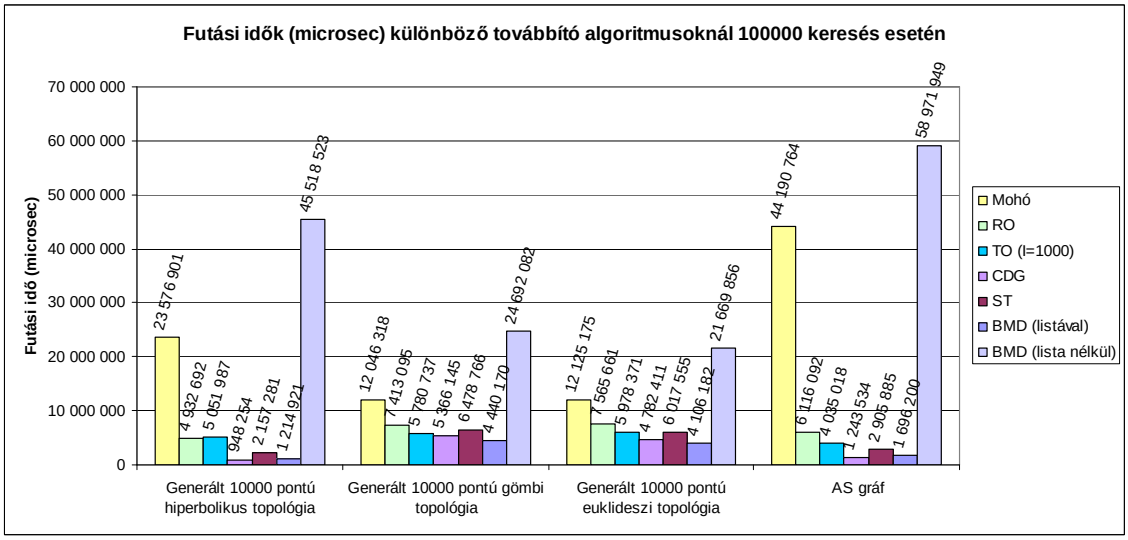


3.9. ábra Műveletszámok

3.3.4.4 Futási idő

A tényleges futási idők talán még a műveletszámnál is jobban megmutatják a továbbító algoritmusok közti különbségeket, mivel ezek mért adatok, nem absztrakt műveletszámok. A 3.9. ábrát a 3.10. ábrával összehasonlítva azonban jól látszik, hogy a grafikonok alakja igen hasonló, tehát nem voltak tévesek a műveletszám segítségével adott becslések.

Csak úgy, mint a műveletszámoknál, itt is a legjobb eredményt a legjobban egyező irányba továbbítás (BMD) és a kombinált továbbítás (CD) adja, ezután jön az irány alapú heurisztika (ST), majd a sorrend alapú heurisztikák, először a forgalom alapú, majd a véletlen heurisztika. A hagyományos algoritmusok a sor végén vannak jelentős lemaradással, először a mohó továbbítás, végül az irány alapú továbbítás lista nélkül.



3.10. ábra Futási idők

3.3.5 A kombinált továbbítás változatainak összehasonlítása

Az előző fejezetben csak a kombinált továbbítás (CD) egyik verzióját vizsgáltam, amelyikben az irány alapú továbbítás hibája esetén a hagyományos mohó továbbításra áll át az algoritmus. Ebben a fejezetben azt is megmutatom, hogy milyen eredményeket kapunk, ha ilyenkor nem a mohó továbbítást, hanem a forgalom vagy a véletlen sorrend alapú továbbítást választjuk. A továbbiakban ezeket hívom a kombinált továbbítás aleseteinek (változatainak), amikre a következő módon hivatkozok:

- CDG: mohó továbbítással kombinált továbbítás
- CDRO: véletlen heurisztikával kombinált továbbítás
- CDTO: forgalom alapú sorrend heurisztikával kombinált továbbítás

Mivel a kombinált továbbítás olyan esetekre lett kidolgozva, ahol az irány alapú továbbítás (BMD) elbukik, emiatt a következő vizsgálatoknál nem foglalkoztam az irány alapú továbbítások közül a BMD és a RID algoritmusokkal ezek alacsonyabb sikerességi rátája miatt. Így jobban előjönnek a kombinált továbbítási módszerek közti különbségek. Azonban a hagyományos mohó, a sorrend alapú heurisztikákat (RO, TO), és az irány közelebb (ST) heurisztikát továbbra is vizsgálom, hogy össze lehessen hasonlítani a kombinált továbbítással. A vizsgálatokat ugyanazokon a topológiákon végeztem, mint amin az előző fejezetben bemutatott méréseket is végeztem. A kapott eredményeket a 3.4. táblázatban foglaltam össze.

Generált 10000 pontú hiperbolikus topológia, 100000 keresés	Mohó	Sorrend alapú heurisztika		Kombinált			Irány alapú heurisztika	Legrövidebb út
		RO	TO (l=1000)	CDG	CDRO	CDTO (l=1000)	ST	
Inicializálási idő (micro sec)	0	0	294 940	135 040	135 040	429 980	101 558	
Futási idő (micro sec)	27 767 316	5 532 336	6 180 470	1 849 447	1 698 359	1 671 306	2 854 207	
Müveletek száma	114 699 708	22 045 048	21 287 156	3 542 285	3 152 933	3 047 009	1 643 315	
Sikeres keresések aránya	1	1	1	1	1	1	1	
Átlagos úthossz	3.44	6.98	6.47	5.92	5.95	5.95	3.64	3.38

Generált 10000 pontú gömbi topológia, 100000 keresés	Mohó	Sorrend alapú heurisztika		Kombinált			Irány alapú heurisztika	Legrövidebb út
		RO	TO (l=1000)	CDG	CDRO	CDTO (l=1000)	ST	
Inicializálási idő (micro sec)	0	0	133 277	73 779	73 779	207 056	50 300	
Futási idő (micro sec)	13 011 187	7 689 551	6 432 311	5 649 613	4 738 373	4 514 936	7 126 442	
Müveletek száma	71 034 472	41 620 924	27 548 646	30 802 051	26 285 579	24 324 077	23 110 616	
Sikeres keresések aránya	1	1	1	1	1	1	1	
Átlagos úthossz	40.76	63.93	49.98	46.10	49.70	47.78	47.01	36.33

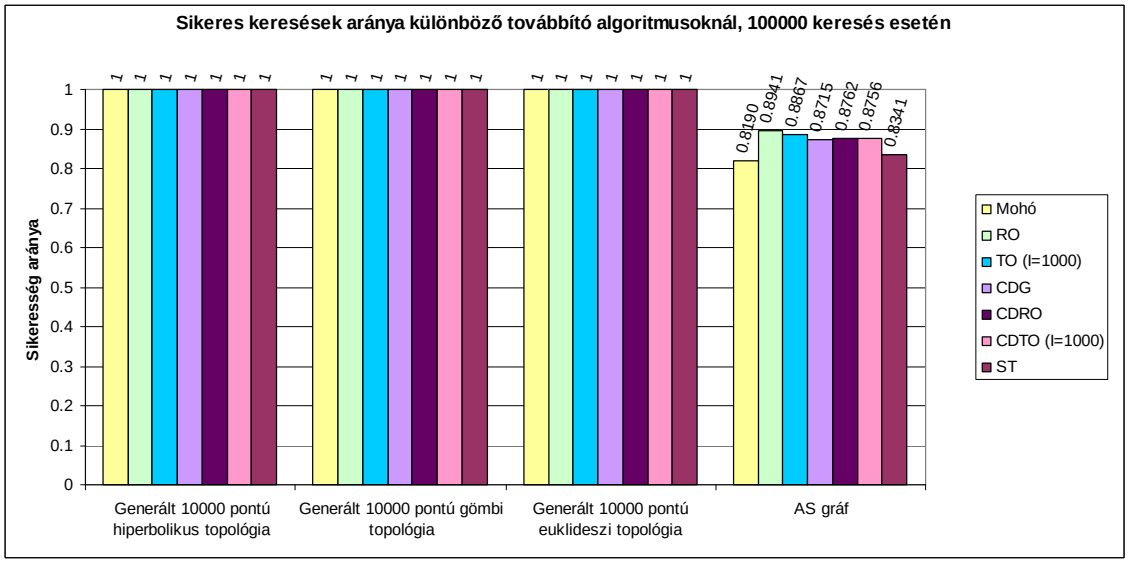
Generált 10000 pontú euklideszi topológia, 100000 keresés	Mohó	Sorrend alapú heurisztika		Kombinált			Irány alapú heurisztika	Legrövidebb út
		RO	TO (l=1000)	CDG	CDRO	CDTO (l=1000)	ST	
Inicializálási idő (micro sec)	0	0	140 259	68 485	68 485	208 744	45 694	
Futási idő (micro sec)	12 837 729	7 843 756	6 145 185	4 856 933	3 953 390	3 656 067	6 365 212	
Müveletek száma	71 550 352	42 109 288	28 015 748	30 806 929	26 429 605	24 427 525	23 218 863	
Sikeres keresések aránya	1	1	1	1	1	1	1	
Átlagos úthossz	41.09	64.47	50.43	46.37	49.97	47.95	47.23	36.55

asgraph, 100000 keresés	Mohó	Sorrend alapú heurisztika		Kombinált			Irány alapú heurisztika	Legrövidebb út
		RO	TO (l=1000)	CDG	CDRO	CDTO (l=1000)	ST	
Inicializálási idő (micro sec)	0	0	821 837	187 288	187 288	1 009 125	137 553	
Futási idő (micro sec)	55 283 312	8 144 617	5 419 857	12 487 119	5 055 721	4 374 280	4 489 923	
Müveletek száma	250 080 493	32 253 381	16 603 607	44 212 760	10 180 848	6 981 844	2 826 786	
Sikeres keresések aránya	0.8190	0.8941	0.8867	0.8715	0.8762	0.8756	0.8341	
Átlagos úthossz	3.75	5.80	5.52	10.74	11.06	11.00	4.23	3.52

3.4. Táblázat A kombinált továbbítások eredményei különböző topológiákon

3.3.5.1 Sikeres útvonalkeresések aránya

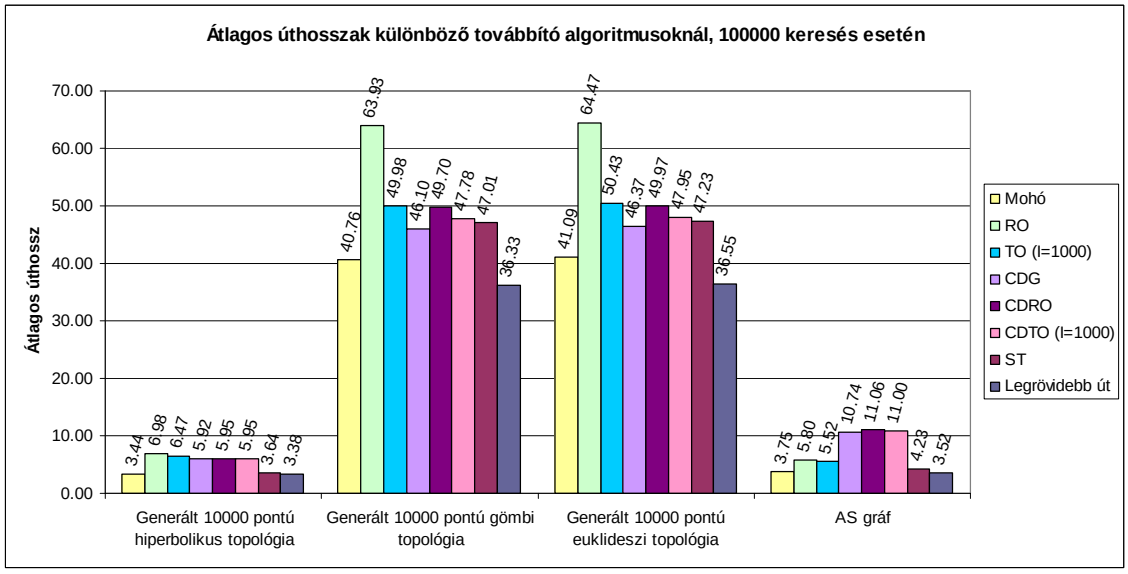
A méréseim során azt kaptam, hogy a generált topológiákon minden vizsgált továbbító algoritmus minden esetben sikeres volt. Az AS topológián viszont érdekesebb eredményeket kaptam, mind a heurisztikák (RO, TO, ST), mind a kombinált továbbító algoritmusok (CDG, CDRO, CDTO) jobb eredményt értek el, mint a hagyományos mohó továbbítás. Az ST heurisztika sikeressége 1,5 százalékkal volt jobb a mohónál, a kombinált továbbítások közt a sikeresség szempontjából nincs jelentős különbség, mindegyik nagyjából 5%-al jobb eredményt ért el, mint a hagyományos mohó továbbítás. A sorrend alapú heurisztikák közt sincs jelentős különbség, ezek még a kombinált továbbításnál is 2-3%-al jobb eredményt értek el (lásd 3.11. ábra). A magasabb sikeresség a nagyobb bolyongás (magasabb átlagos úthossz) egyfajta pozitív hozadéka, hisz ezen algoritmusok jobban elkerülik a lokális optimumokat, ahol a hagyományos mohó továbbítás elakad.



3.11. ábra A sikeres keresések aránya a sorrend alapú heurisztikáknál és kombinált továbbításnál

3.3.5.2 Átlagos úthossz

Átlagos úthosszak szempontjából jelentősen eltérnek a továbbító algoritmusok a különböző topológiákon. Két jellemző csoportra lehet osztani a vizsgált topológiákat átlagos úthossz szempontjából: a hiperbolikus térbe ágyazottakra (generált 10000 csomópontú topológia, AS gráf), és az euklideszi valamint gömbi térbe ágyazottakra (lásd 3.12. ábra).



3.12. ábra Átlagos úthosszak a sorrend alapú heurisztikáknál és kombinált továbbításnál

A hiperbolikus csoportban nagyon rövid átlagos úthosszakat láthatunk, a legalacsonyabb értékkel a mohó továbbítás rendelkezik, amit nagyon jól közelít az irány

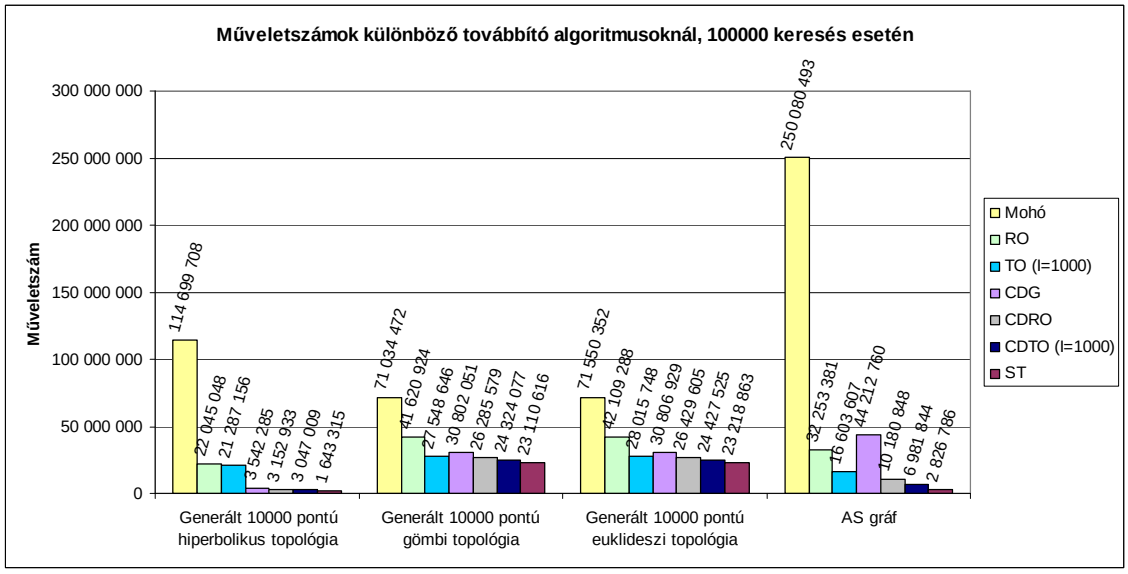
közelebb (ST) heurisztika. A generált topológián, ahol az irány alapú továbbítás viszonylag jól működik, a kombinált továbbítások is alacsony útvonalnyúlással működnek, azonban az AS gráfon jelentős útvonalnyúlással rendelkeznek és lényegesebben rosszabb eredményt adnak, mint az ST heurisztika, sőt a sorrend alapú heurisztikák (RO, TO) is jobb eredményt érnek el.

Az euklideszi és gömbi térbe ágyazott hálózatokon az ST heurisztika előnye már nem egyértelmű a kombinált továbbításhoz képest, a CDG meg is tudja előzni alacsonyabb útvonalnyúlásával. A kombinált továbbítások ezeken a topológiákon a sorrend alapú heurisztikáknál is jobb eredményt adnak.

A kombinált továbbítások közt mindig a CDG, CDTO, CDRO sorrend adódik, ami nem meglepő, tekintve, hogy a mohó, forgalom alapú heurisztika, véletlen heurisztika is ilyen sorrendben követik egymást útvonalnyúlás szempontjából.

3.3.5.3 Műveletek száma

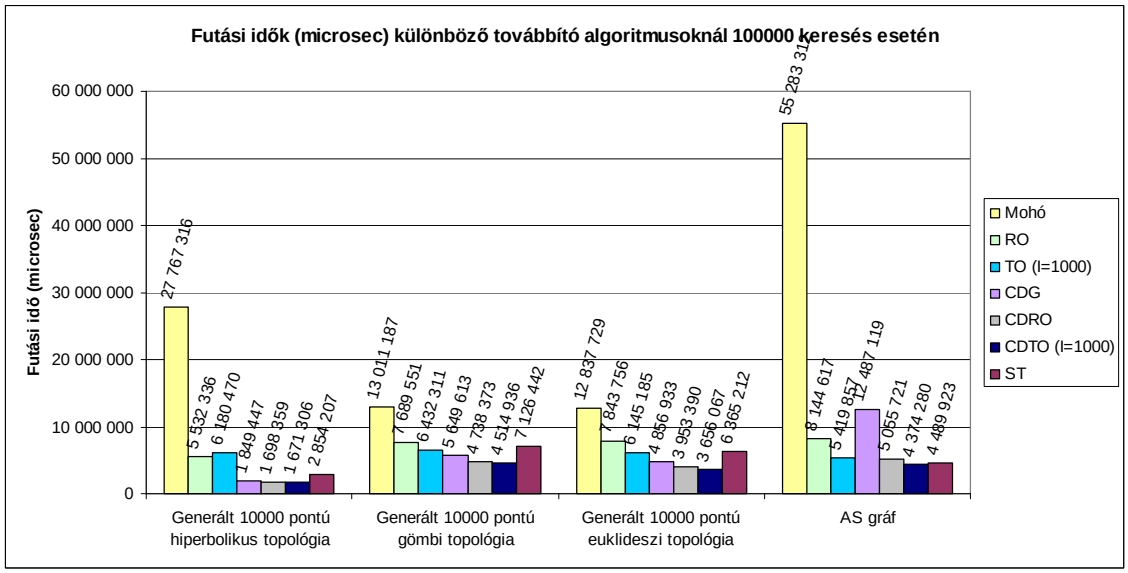
Műveletszám szempontjából két dolgot fontos észrevenni. Az első, hogy a kombinált továbbítás amiatt, hogy irány alapú továbbítással indít, mindig sokkal alacsonyabb műveletszámot ad, mint önmagában az az algoritmus, amire sikertelensége esetén átvált. Például a mohónál (Greedy) jobb a CDG, az RO-nál jobb a CDRO, a TO-nál pedig a CDTO. A másik, hogy akár kombinált továbbításban használjuk, akár nem, mindig a forgalom alapú sorrend a legjobb műveletszám szempontból, ezt követi a véletlen heurisztika, majd jelentősen lemaradva a hagyományos mohó továbbítás. Azonban műveletszám szempontjából minden esetben az ST heurisztika végez az élen, megelőzve a kombinált továbbításokat. Az AS topológia esetében ezek az eltérések sokkal jelentősebbek, és jobban mutatják a kombinált továbbítás előnyeit (lásd 3.13. ábra).



3.13. ábra Műveletszámok a sorrend alapú heurisztikáknál és kombinált továbbításnál

3.3.5.4 Futási idő

Itt is hasonló eredményeket kapunk mint a műveletek számánál, tehát a vizsgált absztrakt műveletszámok viszonylag jól közelítik a tényleges futási időt. Tehát ha a konkrét számoktól eltekintünk, a 3.13. és a 3.14. ábra erősen hasonlít egymásra, így itt is ugyanaz mondható el, mint a műveletek számánál. Ez alól az ST heurisztika a kivétel, ami bár nem rendelkezik rossz futási időkkel, de eredménye nem olyan kiugró, mint a műveletszámok grafikonon tűnt, a kombinált továbbítások általában futási idő szempontjából jobb eredményt érnek el.

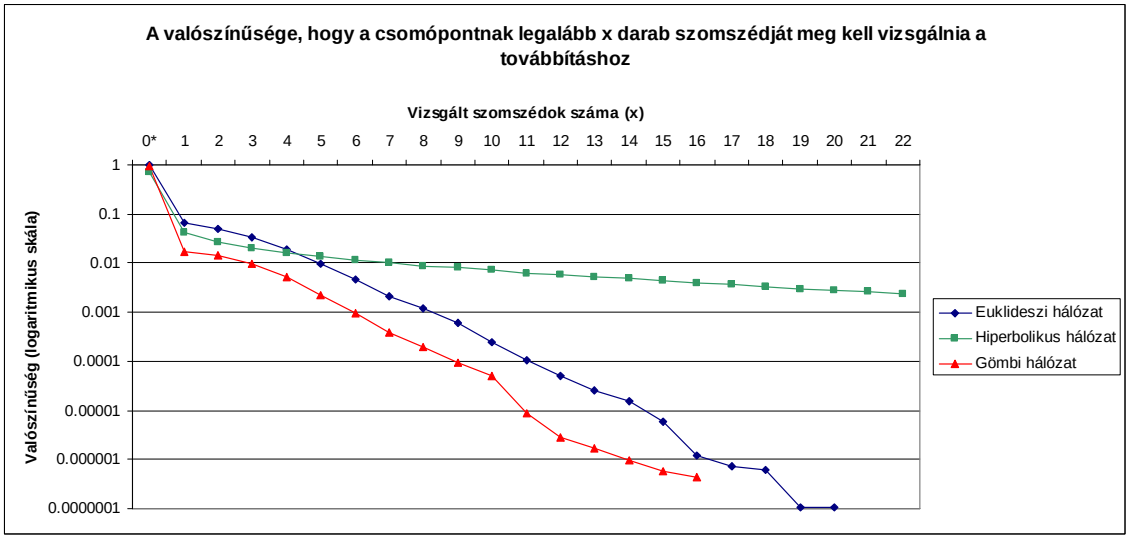


3.14. ábra Futási idők a sorrend alapú heurisztikáknál és kombinált továbbításnál

3.3.6 Az irány közelebb továbbításnál (ST) vizsgálandó szomszédok száma

Azt is vizsgáltam, hogy az ST heurisztikánál mekkora a valószínűsége, hogy a legjobban egyező irányú szomszéd (amit a BMD kiválaszt, és rögtön továbbít is felé) nem visz közelebb a célhoz, és ezért az ST algoritmusnak a csomópontban más szomszédot kell választania a továbbításhoz. A 3.15. ábra logaritmikus skálán mutatja, hogy egy csomópontnak milyen valószínűséggel kell megvizsgálni több (legalább x+1 darab) szomszédját. Három topológián vizsgáltam ezt a paramétert, mindhárom 5000 csomópontú, a TOP_GEN algoritmussal generált topológia, három különböző metrikus térbe ágyazva.

Nulla szomszéd vizsgálatára akkor van szükség, ha a cél az adott csomópontnak közvetlen szomszédja, ugyanis ilyenkor nem kell vizsgálni, hogy a továbbítással közelebb kerülök-e célhoz. Ezek alapján az x tengelyen a „0” helyen felvett érték jelöli azt a valószínűséget, hogy a címzett nem az adott csomópont közvetlen szomszédja, tehát meg kell keresnie a cél irányával legjobban egyező irányú szomszédját a csomópontnak, és ellenőriznie kell, hogy az közelebb van-e nála. Az „1” helyen felvett érték pedig azt jelöli, hogy mekkora valószínűséggel nem közvetlen szomszédja a cél a csomópontnak, és a legjobban egyező irányú szomszéd sincs közelebb a célhoz, ezért legalább még egy szomszédját kell vizsgálnia.



3.15. ábra Az irány közelebb (ST) továbbításnál szükséges szomszédvizsgálatok

Jól látható az ábrán, hogy a szükséges szomszédvizsgálatok száma logaritmikus lecsengésű, azaz nagyon alacsony valószínűséggel kell a csomópontnak sok szomszédjára kiszámolni a céltól vett távolságot, emiatt tud jóval hatékonyabban működni, mint a hagyományos mohó továbbítás.

4 Összefoglalás, továbblépési lehetőségek

4.1 Összefoglaló

A diplomámban azzal foglalkoztam, hogy hogyan lehetne egy megbízható és alacsony erőforrás igényű földrajzi útvonalválasztó algoritmust kitalálni. Először a földrajzi útvonalválasztásnál elterjedten használt továbbító algoritmusokat és ezek alkalmazási területeit vizsgáltam, hogy ebből kiindulva létre tudjak hozni egy heurisztikus módszert.

A célkitűzésem a heurisztika tervezésekor az volt, hogy olyan továbbító algoritmust hozzak létre, ami minőségi paramétereiben az elterjedt mohó továbbításhoz hasonló, számításigénye azonban töredéke annak. A legfontosabb minőségi paramétereknek a sikerességet (azaz a sikeres keresések arányát), valamint az átlagos úthosszt vettem, és ezen szempontok alapján akartam a mohó továbbításhoz hasonló heurisztikát tervezni. A tervezéskor a legfontosabb észrevételem az volt, hogy az irány alapú továbbítás hatékonyan meggyorsítható egy csomópontonként létrehozott, irány alapján rendezett szomszédlista segítségével. Ezután a feladatom főleg az irány alapú továbbítás hibáinak orvoslásából, az erre kidolgozott algoritmusok tervezéséből állt. Az elsőre kidolgozott RID algoritmus nem bizonyult elég hatékonynak, ám remek kiindulási pontot adott a kombinált algoritmusokhoz, ugyanis a kombinált továbbítás leállási feltételét a RID algoritmus adja. A kombinált továbbítások ugyan remek eredményeket adtak futási idő szempontjából, azonban egy olyan algoritmust akartam tervezni, ami tetszőleges hálózaton működőképes, nem kell például a megfelelő lépésszám korlát beállításával (és a tárolásával, továbbításával) foglalkozni, valamint az algoritmus is egyszerűbb lenne, ha nem kéne több továbbítási eljárást kombinálni benne. Ezen igények alapján született az irány közelebb (ST) heurisztika.

A futási idővel kapcsolatos eredmények meggyőzőek, az irány alapú továbbítás legnagyobb előnye, hogy az esetek többségében nagyon hatékonyan működik, hatékonyabban, mint a többi algoritmus. A hagyományos irány alapú továbbítás (BMD) legnagyobb hibája, hogy nem tudja, meddig működik helyesen a továbbítás, ezt azonban a kombinált (CD) és az irány közelebb (ST) algoritmus is detektálja. Ekkor a CD algoritmus esetén tetszőleges más algoritmussal lehet folytatni az útvonalválasztást,

az ST algoritmus pedig beépítve tartalmazza a javítást ilyen esetekre. A CD és ST algoritmusok közti választáskor az alapján kell döntenünk, hogy egy egyszerűbb, beállítást nem igénylő, kevesebb plusz információt továbbító algoritmust használjunk (ST), vagy esetleg egy ennél akár jobb eredményeket hozó algoritmust (CD), ami viszont az említett hátrányokkal rendelkezik.

4.2 Továbblépési lehetőségek

A kombinált továbbítás alapján további hibrid algoritmusokat lehetne fejleszteni, amivel a sikerességet is lehetne növelni, vagy akár garantálni is. Gondolok itt például a GPSR [1] és a face routing [2] technológiákra, érdemes lenne valahogy ezt is belevenni egy hibrid algoritmusba. Ugyanez igaz az irány alapú heurisztikára is, mivel ez az algoritmus a kombinált továbbításhoz hasonlóan akkor lesz sikertelen, ha nincs greedy next hop (azaz a célhoz közelebbi szomszéd), és pont ezen esetekre lettek kidolgozva az említett technológiák.

Az útvonalválasztó algoritmusokat számos egyéb szempontból is lehetne vizsgálni, például lehetne vizsgálni az algoritmusok hibatűrését is. A vizsgálatokat érdemes lenne kiterjeszteni más topológiákra is, esetleg vizsgálni, hogy a topológia milyen szerepet játszik a sikerességben, és mit kell megkövetelni a topológiától, hogy garantált legyen a sikeresség. Távlati lehetőség akár egy konkrét implementáció elkészítése is.

Köszönetnyilvánítás

Szeretném megköszönni konzulenseim, Dr. Gulyás András, Csernai Márton és Kőrösi Attila segítségét, mely nélkül munkám nem jöhetett volna létre. A közös munka már alapképzésen elkezdődött, ők segítettek a téma kiválasztásában, és ötleteikkel a megoldás kialakulásához is jelentősen hozzájárultak. A félév során mindig azonnal segítettek, ha elakadtam, és a dolgozatom átnézésében, a hibák javításában is jelentős szerepet játszottak.

Külön szeretnék köszönetet mondani Csernai Mártonnak. Ő segített a kezdeti program létrehozásában is, sokat segített a kódban lévő hibák keresésében, biztosította számomra a gépét tesztek végrehajtásához.

Szeretnék továbbá köszönetet mondani mindenkinek, aki segítette a félév során a munkámat. Gondolok itt az évfolyamtársaimra, akik számos tanácsot adtak a dolgozatom elkészítéséhez, és bíztattak a munkám készítése során. Szeretném megköszönni továbbá szüleimnek, akik kezdetek óta támogattak, és segítették a pályám alakulását.

Irodalomjegyzék

[1] B. Karp and H. Kung: *Greedy Perimeter Stateless Routing*. In Proceedings of the Sixth Annual ACM/IEEE International Conference on Mobile Computing and Networking (Mobicom 2000), 2000
http://www.eecs.harvard.edu/~htk/publication/2000-mobi-karp-kung.pdf

[2] Prosenjit Bose, Pat Morin, Ivan Stojmenović, Jorge Urrutia: *Routing with Guaranteed Delivery in Ad Hoc Wireless Networks*, Wireless Networks, Volume 7, Issue 6, pp 609-616, 2001 November
http://cg.scs.carleton.ca/~morin/publications/online/unitgraphs-wn.pdf

[3] Evangelos Kranakis, Harvinder Singh, Jorge Urrutia: *Compass Routing on Geometric Networks*, IN PROC. 11 Th Canadian Conference on Computational Geometry, 1999
http://www.matem.unam.mx/~urrutia/online_papers/comprout.pdf

[4] Tommaso Melodia, Dario Pompili, Ian F. Akyildiz: *Optimal local topology knowledge for energy efficient geographical routing in sensor networks*, INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies, 2004
http://www.eng.buffalo.edu/wnesl/people/papers/infocom2004.pdf

[5] Ting-Chaou Hou, V.O.K. Li: *Transmission Range Control in Multihop Packet Radio Networks*, IEEE Transactions on Communications, Volume 34, Issue 1, pp. 38-44, 1986 január
http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1096436&url=http%3A%2F%2Fieeexplore.ieee.org%2Fiel5%2F26%2F24000%2F01096436

[6] Hideaki Takagi, Leonard Kleinrock: *Optimal Transmission Ranges for Randomly Distributed Packet Radio Terminals*, IEEE Transactions on Communications, Volume 32, Issue 3, pp. 246-257, 1984 március
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.64.9747&rep=rep1&type=pdf

[7] Finn, Gregory G. : *Routing and Addressing Problems in Large Metropolitan-Scale Internetworks*, University of Southern California, 1987 március
http://www.isi.edu/div7/people/finn.home/routing_and_addressing_problems_in_large_metropolitan-scale_internetworks.BW.pdf

[8] A. Rao, S. Ratnasamy, C. Papadimitriou, S. Shenker, I. Stoica. *Geographic Routing without Location Information*. In Proceedings of the 9th annual international conference on Mobile computing and networking (Mobicom 2003), 2003.
http://www.cs.ucla.edu/classes/fall03/cs218/paper/p96-rao.pdf

[9] M. Boguñá, F. Papadopoulos, and D. Krioukov: *Sustaining the Internet with Hyperbolic Mapping*, Nature Communications, v.1, 62, 2010

http://www.nature.com/ncomms/journal/v1/n6/full/ncomms1063.html#/abstract
http://arxiv.org/pdf/1009.0267.pdf

[10] Zygmunt J. Haas, Jing Deng, Ben Liang, Panagiotis Papadimitratos, S. Sajama: *Wireless Ad Hoc Networks*, 2002
http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.13.8133

[11] Reza Azimi, Gaurav Bhatia, Ragunathan (Raj) Rajkumar, Priyantha Mudalige: *Vehicular Networks for Collision Avoidance at Intersections*, SAE 2011 World Congress & Exhibition, 2011
http://users.ece.cmu.edu/~sazimi/SAE2011.pdf

[12] David B. Johnson, David A. Maltz: *DSR: Dynamic Source Routing in Ad Hoc Wireless Networks*, Mobile Computing pp 153-181, 1996
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.129.426&rep=rep1&type=pdf

[13] David B. Johnson, David A. Maltz, Josh Broch: *DSR: The Dynamic Source Routing Protocol for Multi-Hop Wireless Ad Hoc Networks*, Ad Hoc Networking pp 139-172, 2001
http://www.monarch.cs.rice.edu/monarch-papers/dsr-chapter00.pdf

[14] M Mauve, A Widmer, H Hartenstein: *A survey on position-based routing in mobile ad hoc networks*, IEEE Network, Volume 15, Issue 6, 2001 december
http://erdos.csie.ncnu.edu.tw/~ccyang/WirelessNetwork/Papers/MANET/AdHocUnicast-2.pdf

[15] Godfried T. Toussaint: *The relative neighbourhood graph of a finite planar set*, Pattern Recognition, Volume 12, Issue 4, pp 261–268, 1980
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.54.6448&rep=rep1&type=pdf

[16] Srdjan Capkun, Maher Hamdi, Jean-Pierre Hubaux: *GPS free positioning in mobile ad hoc networks*, Proceedings of the 34th Annual Hawaii International Conference on System Sciences, 2001 január
http://pi4.informatik.uni-mannheim.de/pi4.data/content/courses/2005-ss/sensornetze/Capkun.pdf

[17] Kazem Sohraby, Daniel Minoli, Taieb Znati: *Wireless sensor networks: technology, protocols, and applications*, John Wiley and Sons Inc., 2007
http://www.tfb.edu.mk/amarkoski/WSN/Kniga-w02.pdf

[18] Waltenegus Dargie, Christian Poellabauer: *Fundamentals of wireless sensor networks: theory and practice*, John Wiley and Sons, 2010

[19] I.F. Akyildiz, W. Su, Y. Sankarasubramaniam, E. Cayirci: *Wireless sensor networks: a survey*, Computer Networks Volume 38, Issue 4, 2002 március
http://www.ece.gatech.edu/research/labs/bwn/surveys/sensornets.pdf

[20] Jamal N. Al-Karaki, Ahmed E. Kamal: *Routing techniques in wireless sensor networks: a survey*, IEEE Wireless Communications Volume 11 , Issue 6 pp 6-28, 2004 December

http://wiki-castelldefels.cimne.upc.edu/images/9/93/Routing_Techniques_in_Wireless_sensor_Networks._A_survey..pdf

[21] Kemal Akkaya, Mohamed Younis: *A survey on routing protocols for wireless sensor networks*, Ad Hoc Networks Volume 3, Issue 3, pp 325–349, 2005 május
http://home.deib.polimi.it/cesana/teaching/IoT/papers/routing/survey.pdf

[22] Giuseppe Anastasi, Marco Conti, Mario Di Francesco, Andrea Passarella: *Energy conservation in wireless sensor networks: A survey*, Ad Hoc Networks Volume 7, Issue 3, pp 537–568, 2009
http://cnd.iit.cnr.it/andrea/docs/aohoc09.pdf

[23] Yan Yu, Ramesh Govindan, Deborah Estrin: *Geographical and Energy Aware Routing: a recursive data dissemination protocol for wireless sensor networks*, UCLA Comp. Sci. Dept. tech. rep., 2001 augusztus 14
http://graphics.stanford.edu/courses/cs428-03-spring/Papers/readings/Networking/Estrin_geo-routing01.pdf

[24] Tiago Camilo, Jorge Sá Silva, André Rodrigues, Fernando Boavida: *GENSEN: A Topology Generator For Real Wireless Sensor Networks Deployment*, Software Technologies for Embedded and Ubiquitous Systems, pp 436-445, 2007
https://cs.wmich.edu/gupta/teaching/cs5950/fall2011/gensen%20tool%20for%20sensornet%20deployment%20by%20camilo-et-al%2010.1.1.106.7106.pdf

[25] Amogh Dhamdhere, Constantine Dovrolis: *Twelve Years in the Evolution of the Internet Ecosystem*, IEEE/ACM Transactions on Networking, Volume 19, Issue 5, pp 1420 - 1433, 2011 október
http://www.cc.gatech.edu/~dovrolis/Papers/amogh-ton-rewiring.pdf

[26] D. Meyer, L. Zhang, K. Fall: *RFC4984 (Report from the IAB Workshop on Routing and Addressing*), 2007 szeptember
http://tools.ietf.org/html/rfc4984

[27] Réka Albert, Albert-László Barabási: *Statistical Mechanics of Complex Networks*, Reviews of modern physics, Volume 74, 2002 január
http://www.stanford.edu/class/msande337/notes/200201-30_RevModernPhys-StatisticalMech.pdf

[28] Bővebb leírás található a Poincaré Disk Modellről és a Hyperboloid modellről, és az ezekben használt távolságfüggvényről a Wikipédia angol nyelvű oldalain:
http://en.wikipedia.org/wiki/Poincar%C3%A9_disk_model
http://en.wikipedia.org/wiki/Hyperboloid_model

[29] Hiperbolikus parkettázások tulajdonságainak magyarázata és további parkettázások az alábbi oldalon találhatóak:
http://aleph0.clarku.edu/~djoyce/poincare/poincare.html

[30] K. Claffy, Y. Hyun, K. Keys, M. Fomenkov, D. Krioukov: *Internet Mapping: from Art to Science*, CATCH '09 Proceedings of the 2009 Cybersecurity Applications & Technology Conference for Homeland Security pp 205-211, 2009

http://www.caida.org/publications/papers/2009/internet_mapping/internet_mapping.pdf

[31] Stanley Milgram: *The Small-World Problem*, Psychology Today, 1967
http://measure.igpp.ucla.edu/GK12-SEE-LA/Lesson_Files_09/Tina_Wey/TW_social_networks_Milgram_1967_small_world_problem.pdf

[32] Jeffrey Travers, Stanley Milgram: *An Experimental Study of the Small World Problem*, Sociometry, Vol. 32, No. 4., pp. 425-443, 1969 december
http://www.cis.upenn.edu/~mkearns/teaching/NetworkedLife/travers_milgram.pdf

[33] Peter Sheridan Dodds, Roby Muhamad, Duncan J. Watts: *An Experimental Study of Search in Global Social Networks*, Science, Vol. 301, No. 5634 pp 827-829, 2003 Augusztus 8
http://www.cis.upenn.edu/~mkearns/teaching/NetworkedLife/columbia.pdf

[34] Barabási Albert-László: *Behálózva*, Helikon Kiadó, 2008

[35] Erdős Pál, Rényi Alfréd: *On Random Graphs. I.*, Publicationes Mathematicae 6, pp 290–297, 1959
http://www.renyi.hu/~p_erdos/1959-11.pdf

[36] Duncan J. Watts, Steven H. Strogatz: *Collective dynamics of small-world networks*, Nature 393, pp 440-442, 1998 június 4
http://labs.yahoo.com/files/w_s_NATURE_0.pdf

[37] Alan Mislove, Massimiliano Marcon, Krishna P. Gummadi, Peter Druschel, Bobby Bhattacharjee: *Measurement and analysis of online social networks*, In Proceedings of the 7th ACM SIGCOMM conference on Internet measurement, pp 29-42, 2007
http://conferences.sigcomm.org/imc/2007/papers/imc170.pdf

[38] Jon M. Kleinberg: *Authoritative Sources in a Hyperlinked Environment*, Journal of the ACM (JACM), Volume 46 Issue 5, pp 604-632, 1999 Szeptember
http://www.cs.cornell.edu/home/kleinber/auth.pdf

[39] Scott Garriss, Michael Kaminsky, Michael J. Freedman, Brad Karp, David Mazières , Haifeng Yu: *Re: Reliable email*, In Proc. NSDI, pp 297-310, 2006
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.141.5732&rep=rep1&type=pdf

[40] Alan Mislove, Krishna P. Gummadi, Peter Druschel: *Exploiting social networks for Internet search*, In Proceedings of the 5th Workshop on Hot Topics in Networks (HotNets-V), 2006
http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.107.2420&rep=rep1&type=pdf

[41] Jon M. Kleinberg: *Navigation in a small world*, Nature 406, 2000 augusztus 24
http://www.cs.cornell.edu/home/kleinber/nat00.pdf

[42] Duncan J. Watts, Peter Sheridan Dodds, M. E. J. Newman: *Identity and Search in Social Networks*, Science, Volume 296, no. 5571, pp 1302-1305, 2002 május 17
http://labs.yahoo.com/files/w_d_n_SCIENCE.pdf

[43] Lada Adamic, Eytan Adar: *How to search a social network*, Social Networks, Volume 27, Issue 3, pp 187–203, 2005 július
http://www.cond.org/socsearch.pdf

[44] Oskar Sandberg: *Distributed Routing in Small-World Networks*, Proceedings of the Eighth Workshop on Algorithm Engineering and Experiments (ALENEX), 2006
http://www.siam.org/proceedings/alenex/2006/alx06_014osandberg.pdf

[45] Gábor Csárdi, Tamás Nepusz: *The igraph software package for complex network research*. InterJournal Complex Systems, 1695, 2006.

[46] Gábor Csárdi, Tamás Nepusz: *igraph Reference Manual*, 2006
http://igraph.sourceforge.net/doc/igraph-docs.pdf