Mérési útmutató a Mobil Kommunikáció és Kvantumtechnológiák Laboratórium méréseihez

Location Area tervezés következő generációs mobil hálózatokban

Mérés helye:

Hálózati Rendszerek és Szolgáltatások Tanszék
Mobil Kommunikáció és Kvantumtechnológiák Laboratórium (MCL)
I.B.113.

Összeállította:
Simon Vilmos
Péterfalvi Gábor

A dokumentum utolsó módosítása:
2013. február 15.

1. Bevezetés

Az 1990-es években tanúi lehettünk a mobil telekommunikációs hálózatok hihetetlen gyors fejlődésének. A kezdetben analóg rendszereket (1G) hamar felváltották a második generációs (2G) hálózatok, mint a például a GSM vagy az IS-95, melyek a jó minőségű hangtovábbítás mellett már kis bitsebességű adattovábbítást is lehetővé tettek. A 2G technológia, élen az európai GSM megvalósítással, a 90-es években igazi sikertörténet volt, és az új évezred hajnalán élte fénykorát. Napjainkban folyik az áttérés a 2G-ről az Európában (már az országok többségében ki is épültek a rendszerek) UMTS, az USA-ban CDMA-2000, Japánban W-CDMA néven futó harmadik generációs mobil hálózatokra (3G), melyek új kódolási, mobilitási megoldásokkal, nagyobb bitsebességgel próbálják kielégíteni a XXI. századi társadalom megnövekedett igényeit.

A fejlődés azonban nem állhat meg. Annak ellenére, hogy a 3G hálózatok még csak pár országban váltották be a hozzájuk fűzött reményeket (Japánban már pár éve hódít a 3G több tíz millió előfizetővel) egyre inkább előtérbe kerülnek a következő, negyedik generációs hálózatok (4G) tervezési-megvalósítási kérdései. Ennek okai a sávszélesség, a mobilitás, a minőség további növelésének alapvető igénye, új ultra-szélessávú szolgáltatások bevezetése, valamint olyan fejlődési trendek, mint például a „mindenütt jelenlevőség” (ubiquity), vagy a globalitás, melyek a 3G fejlesztésénél még nem voltak tervezési szempontok.

A mobil telekommunikációs piac óriási fejlődése a 90-es években a mobil beszédszolgáltatás futótűzszerű terjedésének volt köszönhető világszerte. Egyre többen vásároltak mobil készüléket, és éltek a mobilitás nyújtotta lehetőségekkel. Ennek eredményeképpen például Japánban 2000-re a mobil előfizetések száma meghaladta a vezetékes előfizetésekét. Azonban, ha már egy ország minden lakosának van mobil készüléke, akkor nem várhatjuk az előfizetések, vagy a hálózati forgalom további növekedését csupán a beszéd alapú szolgáltatástól. Így a jövőben egyre inkább a multimédia továbbítása fogja adni a hálózati terhelés nagy részét. A multimédia ilyen arányú megjelenése a forgalomban, nagy mennyiségű adat továbbítását követeli meg a hálózat peremén, a hozzáférési hálózatban is.

2. A többlet jelzésforgalom csökkentése Location Area-ákkal

Azonban a multimédiás szolgáltatások megkövetelik a hálózattól bizonyos minőségi paraméterek szigorú betartását. A következő generációs (3G illetve 4G) mobil hálózatok egyik legfontosabb szolgáltatás minőségi paramétere (Quality of Sevice, továbbiakban QoS) a késleltetés illetve a késleltetés ingadozás. Mivel ezekben a mobil hálózatokban lecsökken a cellák mérete, ezért gyakrabban kerül sor a mobil terminálok cellaváltására (handover). Minden cellaváltás járulékos jelzések generálását jelenti (mivel értesíteni kell a felsőbb hierarchia szinteket és ügynököket, hogy a mobil másik cellában tartózkodik), így ez a megnövekedett jelzésforgalom kihatással lesz a késleltetés ingadozására is, ami komoly fennakadásokat okozhat időfüggő, valós idejű szolgáltatások esetén. A jelzés többlet forgalom az ún. regisztrációs folyamat eredménye, amikor a mobil terminál frissíti a tartózkodási helyének információját az otthoni ügynökénél.

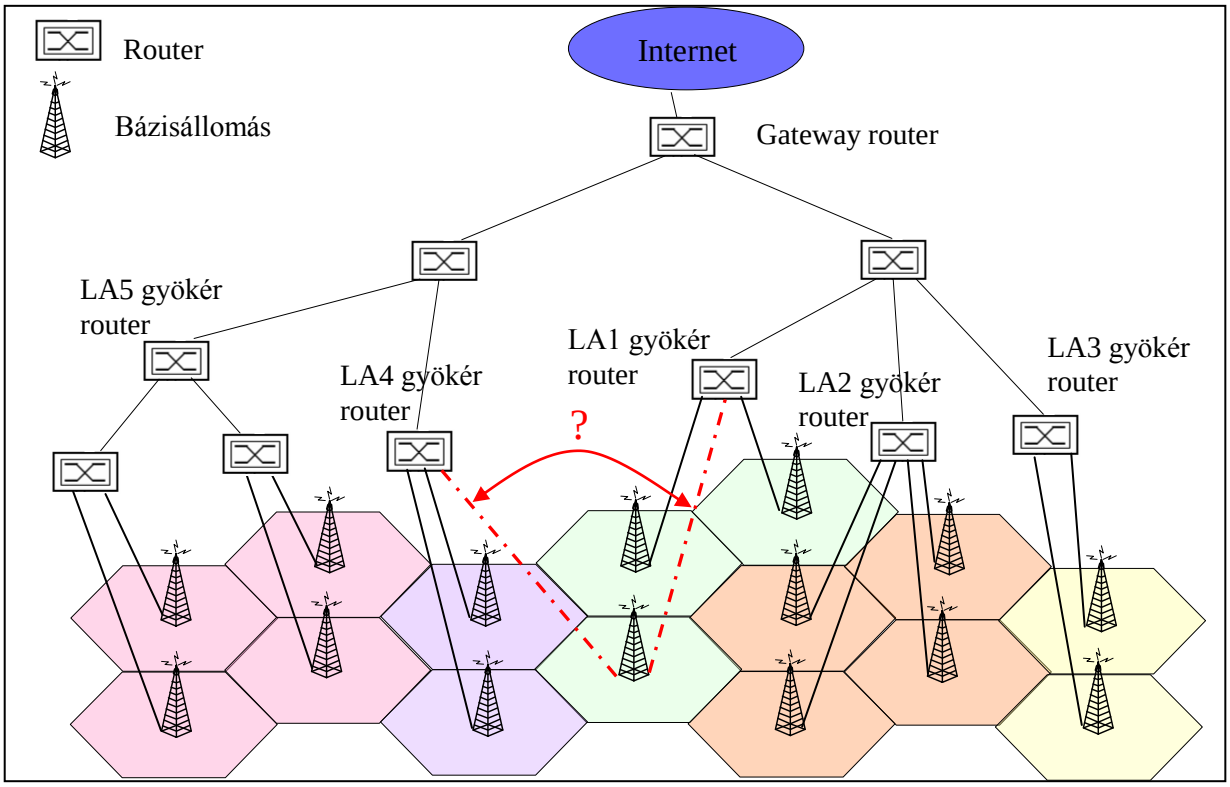
A jelzések csökkentése érdekében ún. Location Area-kat lehet kialakítani, ami azt jelenti hogy több cellát egy adminisztrációs egységbe vonunk össze és együtt kezeljük a jelzésforgalmukat. Ily módon az egységen belüli cella váltások rejtve maradnak a felső hierarchia szintek előtt. Ami azt jelenti, hogy csak akkor keletkezik többlet jelzés, ha az egységek (Location Area-k) határát lépi át a mobil terminál. Mivel ez sokkal ritkább, mint a cellahatár átlépés, így a jelzés üzenetek száma csökkeni fog.

Viszont felmerül a kérdés, hogy mekkora méretű legyen egy ilyen Location Area (továbbiakban LA). A méret csökkentésének és növelésének is megvan a maga előnye. Ugyanis ha több cellát vonunk össze egy LA-ba, akkor a LA-k közötti váltások száma csökkeni fog, így a hely információt hordozó (location update) üzenetek száma is kisebb lesz a szokásosnál. Viszont ha túl sok cella tartozik egy LA-ba, a hálózatba bejövő hívás sok paging üzenetet (a paging eljárás arra szolgál hogy megtaláljuk azt a cellát, amelyben az a mobil terminál tartózkodik, amelyiknek épp hívása érkezett) fog generálni, ugyanis minden cellába ilyen üzenetet kell kiküldeni hogy megtaláljuk vajon melyikben tartózkodik az adott mobil.

Ha kevesebb cellát vonunk össze egy ilyen egységbe, akkor nem kell annyi paging üzenetet kiküldeni (így kevésbé terheljük a csatornákat és a feldolgozási idő is lecsökken), de viszont ilyenkor meg növekszik az LA váltások száma (mivel kisebbek). A két egymásnak ellentmondó szemlélet között kell megtalálni a kompromisszumot.

Ezen két egymásnak ellentmondó szempont nagyon megnehezíti a LA struktúra kialakítását. Ráadásul a cellaváltás és a paging jelzésüzenetek költségeinek aránya a különböző technológiákat és protokollokat megvalósító hálózatokban változó. A valós, effektív költséget minimalizáló LA-struktúra formálásnál ezt az arányszámot is figyelembe kell venni. Ha például a cellaváltás költsége sokkal nagyobb, mint a paging üzenet költsége, akkor egy LA-nak viszonylag sok cellát kell tartalmaznia, hogy az adminisztrálandó cellaváltások száma lehetőleg kevés legyen.

Így már annak meghatározása is nehéz feladat, hogy átlagosan hány cellát vonjunk egy LA-ba, arról nem is beszélve, hogy egy adott LA kialakításánál pontosan mely cellákat vonjuk össze. Vagyis, csak akkor jelenthetjük ki biztosan egy struktúráról, hogy optimális a jelzésforgalom szempontjából, ha minden lehetséges LA felosztást megvizsgáltunk. Egy cella-rendszer LA felosztása megfelel egy a cellák halmazán vett lehetséges particionálásnak. Ezért az összes lehetséges LA struktúra száma, megegyezik a cellák halmazának az összes lehetséges partícionálásának a számával. Egy halmaz összes lehetséges particionálása alatt azt értjük, hogy hányféleképpen lehet az elemeit tetszőleges számú, nem üres, páronként diszjunkt halmazba rakni. Ezt a számot nevezzük Bell számnak. A Bell számokra fennáll a következő rekurzív összefüggés: $B_{n+1} = \sum_{k=0}^{n} \binom{n}{k} B_k$, valamint az n. Bell számra fennáll, hogy $B_n = \frac{1}{e} \sum_{k=0}^{\infty} \frac{k^n}{k!}$. Az első néhány Bell szám $n = 1$-től $n = 12$-ig: 1, 2, 5, 15, 52, 203, 877, 4140, 21147, 115975, 678570, 4213597. Ezekből az összefüggésekből is látható, hogy egy halmaz lehetséges partícióinak száma a halmaz eleminek exponenciális függvénye, ezt bizonyítja a Bell számok exponenciális generátor függvénye is mely $f(x) = e^{e^x - 1}$. Kijelenthetjük tehát, hogy az optimális LA struktúra megtalálása, már kis elemszámú halmazok esetében is NP nehéz feladat.. Szükség van tehát olyan algoritmusokra melyek ugyan nem képesek az optimális struktúra előállítására, azonban viszonylag gyorsan elő tudnak állítani olyan LA struktúrákat, melyek az optimális struktúrát kellően megközelítik (szuboptimális megoldások).



1. ábra: Egy LA struktúra felépítése

Ahogy az 1. ábrán látható, minden LA-nak van egy gyökér routera, ettől a routertól a routerfában lefelé haladva minden elérhető cella beletartozik az adott LA-ba. Természetesen a LA-k egymást nem metszhetik, hiszen ilyenkor nem lenne egyértelmű, hogy a terminál melyik LA-ban van, és, hogy mikor kell a cellaváltást adminisztrálni. Minél magasabb hierarchiaszintjén helyezkedik el egy LA gyökér routera a fa struktúrában, annál több cella tartozik az adott LA-ba. Alapvetően azonban az a jellemző, hogy a legalsó szint routerai lesznek a gyökér routerek, így egyszintes LA-k keletkeznek. A mérés folyamán ezzel az esettel foglalkozunk majd.

3. Költségfüggvények definiálása LA tervezéshez

Paging költségfüggvény

Amikor hívás érkezik a mobil hálózatba, a mobil kapcsoló központ minden hozzá tartozó bázisállomáshoz kiküldi a paging üzenetet, hogy megtalálja a mobil terminál tartózkodási helyét. Így az adott LA-ban minden cellába beérkeznek a paging üzenetek, annak ellenére, hogy csak egyikük fog felelni, ahol éppen a mobil pillanatnyilag van. Így definiálhatjuk a paging költségfüggvényt az l.-ik LA-ra, amivel leírható az a sávszélesség, ami a paging folyamat során lefoglalódik az adott időintervallumra:

$$C_{p_l} = \sum_{i=1}^{K} N \cdot \lambda_i \cdot B_p \qquad (1)$$

ahol

- N a cellák száma az adott j.-ik LA-ban

- λ_i a bejövő hívások száma az adott i. -ik mobil terminálon
- B_p a paging üzenet sávszélessége
- K a mobil terminálok száma az adott j. -ik LA-ban

Az (1) egyenlettel meghatározhatjuk a paging üzenetek költségét egy adott LA-ra, amit a bejövő hívás generál az adott időintervallumban. A teljes paging költség a rendszerünkben található összes LA-ra:

$$C_p = \sum_{l=1}^{M} C_{p_{ll}} = \sum_{l=1}^{M} \sum_{i=1}^{K} N \cdot \lambda_i \cdot B_p = \sum_{l=1}^{M} N \cdot B_p \cdot \sum_{i=1}^{K} \lambda_i \qquad (2)$$

ahol M a rendszerünkben lévő LA-ák száma.

Location update költségfüggvény

Ugyanígy definiálható a location update költségfüggvény is a hálózatunkra, amely a LA váltások számával függ össze (amikor a mobil terminál átlépi a saját cellacsoportjának a határát), ilyenkor járulékos helyinformáció frissítő jelzésforgalom keletkezik (értesítik az otthoni ügynöküket az új helyükről).

Így a location update költségfüggvény a következőképpen írható le:

$$C_{lu_l} = B_{lu} \cdot \sum_{j=1}^{B} q_j \qquad (3)$$

ahol

- B_{lu} a location update üzenet átvitelére szükséges sávszélesség
- q_j a j. -ik cellahatár átlépési intenzitása
- B a külső cellahatárok száma

A teljes location update költségfüggvény a rendszerünkben lévő összes LA-ra:

$$C_{lu} = \sum_{l=1}^{M} C_{lu_l} = \sum_{l=1}^{M} B_{lu} \cdot \sum_{j=1}^{B} q_j \qquad (4)$$

ahol M a rendszerünkben lévő LA-k száma.

A végső cél, hogy maximalizáljuk a LA-k közötti forgalmat, ugyanis így csökkenteni tudjuk a LA váltások számát, ezáltal az adminisztratív üzenetek számát. Ez úgy érhető el, ha azokat a cellákat rendeljük össze közös LA-ba, amelyek domináns mozgási irányok mentén helyezkednek el.

Viszont ahhoz, hogy ki tudjuk értékelni ezt a LA formáló algoritmust, minimalizálnunk kell az aggregált költségfüggvény várható értékét, olyan tetszőleges szorzótényezőkkel súlyozva, amelyek lehetővé teszik a két költségfüggvény fontosságának figyelembevételét:

$$\min E\{w_1 \cdot C_p + w_2 \cdot C_{lu}\} \qquad (5)$$

Így ezekkel a változtatható szorzótényezőkkel tudjuk módosítani a kifejezést, annak függvényében, hogy a paging vagy a location update költségfüggvénynek van nagyobb hatása a szolgáltatás minőségi paraméterekre.

Mivel a várható érték egy homogén lineáris operátor, az (5) kifejezés átalakítható:

$$\min\left(w_1 \cdot E\{C_p\} + w_2 \cdot E\{C_{lu}\}\right) \tag{6}$$

4. A mérés során használt LA tervező algoritmusok

Mint láttuk, a jelzésforgalom mértéke szempontjából globálisan optimális LA-struktúra keresése NP nehéz feladat. Közel optimális LA felosztások azonban találhatók különböző LA formáló algoritmusok segítségével. Ebben a fejezetben kerülnek bemutatatásra a mérés során használt LA formáló algoritmusok.

4.1 Alap algoritmus

Először is határozzuk meg a paging korlátot, ezzel megakadályozzuk, hogy a LA-ákban a paging költség túlságosan nagy legyen. Vegyük a rendszerbe érkező összes hívást, és szorozzuk meg egy tetszőleges természetes számmal (N), melyet az algoritmus futtatása elején kell megadni, majd osszuk el az összes cella számával. Az így kapott szám legyen PK (paging korlát), ez lesz az a felső korlát, amelynél több hívás egyetlen LA-ba sem érkezhet. PK tulajdonképpen az N darab cellába átlagosan beérkező hívások száma. Az N megadásával, pedig azt érjük el, hogy a LA-ákban átlagosan ennyi darab cellát fog összevonni az algoritmus. A forgalmasabb területeken az átlagnál kevesebb cellát von össze, míg a kevésbé forgalmas területeken többet, így próbálja minimalizálni a paging költséget.

Ezután vegyük azt a két cellát, melyek között a legnagyobb a cellaváltások száma, és ezt a kettőt vonjuk össze egy LA-ba. Az így kialakított LA-hoz vegyünk hozzá további cellákat. Egy lépésben mindig azt a cellát fogjuk a LA-hoz venni, amelyiknek a LA-ba irányuló, illetve a LA-ból induló cellaválásainak az összege a legnagyobb. Így a lehető legtöbb cellaváltást sikerül elfednünk a LA-struktúra cellaváltás költségének számításánál. Az új cellák LA-hoz vételét addig folytassuk, amíg el nem érjük a már kiszámolt PK értékünket a LA-ba érkező hívások számára. Ekkor kezdjünk egy új LA-t, a maradék cellák közötti legnagyobb cellaváltás szám megkeresésével. Ezt ismételjük addig, amíg sikerül létrehoznunk új LA-kat. Ezután még maradhatnak olyan cellák, melyeket még nem raktunk bele egyik LA-ba sem, mert a hívások száma túllépte volna a PK határt. Ezeket a cellákat végül vonjuk ahhoz a hozzájuk szomszédos LA-hoz, amelyik LA-nak a cellába irányuló, illetve a cellából induló cellaváltásainak a száma a legnagyobb.

4.2 A kiterjesztett alap algoritmus

Az alap algoritmusban a kapott LA struktúra költsége függött attól, hogy a futtatás elején mekkora N paramétert adtunk meg. A kiterjesztett alap algoritmus végigpróbálja, hogy a szóba jövő N értékek közül melyik érték adja a legkisebb költségű LA struktúrát, és végül ezzel az N értékkel futtatja az alap algoritmust. N értéke 1-től 100-ig terjedhet, ennél nagyobb N a jelenleg létező technológiák, és cellaméretek mellett nem jöhet szóba. Természetesen azt, hogy melyik N érték adja a legkisebb költséget, a cellaváltások, és a hívások aránya dönti el az adott területen. Ha például nagyon sok hívás jön átlagosan egy cellába, akkor nem érdemes túl sok cellát egy LA-ba vonni, mivel akkor nagyon nagy lesz a paging költség, ilyenkor

érdemes N-et kicsinek választani. Ezenkívül, mint láttuk az egy LA-ba vonandó cellák átlagos számát nagyban befolyásolja a $\frac{w_1}{w_2}$ arány is.

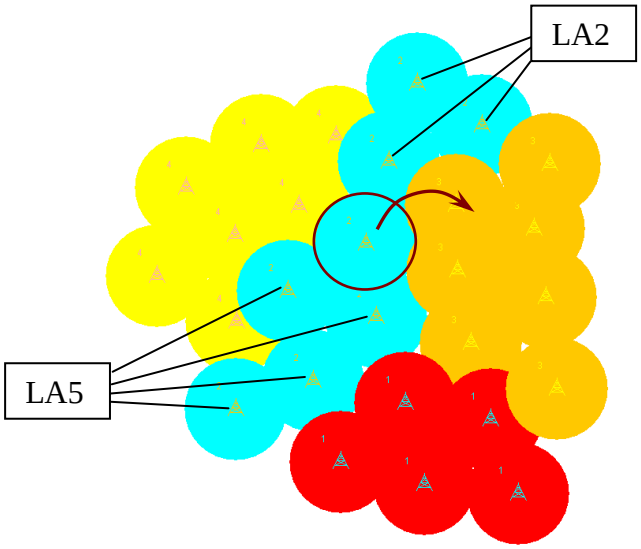
Az alap algoritmus ezzel a kiterjesztéssel optimalizálni tudja az átlagos cellaszámot a LA-kban, és ezzel egy hatékony LA formáló algoritmussá válik. Ráadásul, mivel az alap algoritmus futási ideje nagyon kicsi, ezért drasztikusan még akkor sem növekedik meg a kiterjesztett algoritmus futási ideje, ha viszonylag sok N értékre kell összehasonítania a keletkező struktúrák költségét.

4.3 Az újrarendező algoritmus

Az újrarendező algoritmus egy igen egyszerű szimulált lehűtéshez hasonlóan működő javító algoritmus. Az algoritmus egy már meglévő részben optimális LA-struktúrából indul ki, és azt addig finomítja, amíg az alkalmazott költségfüggvények már nem mutatnak változást a kialakított LA-struktúra összes költségében.

Működése:

Vegyünk két olyan cellát, melyek szomszédosak, de nem tartoznak egy LA-ba. Próbáljuk meg átrakni az egyik cellát a másik cella LA-jába. Ezután futtassuk a költségszámító függvényeinket, és ha a kialakított új LA-struktúra kisebb költségű, mint az eredeti, akkor véglegesítsük az új struktúrát. Előfordulhat azonban olyan eset, amikor az egyik LA-ból a másikba mozgatott cella eredeti LA-ja az áthelyezés miatt két, topológiailag különálló részre esik szét. (2. ábra)



2. ábra: LA-k szétesése az újrarendezés során

Ilyen LA-ákat tartalmazó struktúra elvileg megengedett, de a költséghatékonyság szempontjából mindenképpen elkerülendő. A szétesett LA két különálló része között nincs cellaváltás, így azzal hogy egy LA-ba tartoznak nem csökken a cellaváltás költsége. Ráadásul, ha bármelyik cellába hívás érkezik, a paging üzeneteket minden cellába ki kell küldenünk. A két különálló részt érdemes tehát két külön LA-ra bontani mivel így nagymértékben csökkenthetjük a paging költséget amellett, hogy a cellaváltás költsége nem változik. Ez a lépés azért is fontos, mert az algoritmus így képes a kiindulási struktúrában levő LA-k számát

növelni. Könnyen előfordulhat ugyanis, hogy a cella-rendszerben az optimális struktúra több LA-ból áll, mint az újrarendező algoritmus bemeneti struktúrája, ekkor pedig szükség lehet a LA-k számának szaporítására. A széteső LA-k felbontásával ilyen esetekben is jó esély van arra, hogy az újrarendező algoritmus egy optimálishoz közeli LA-struktúrát találjon.

A fenti lépéssorozatot folytassuk az összes lehetséges szomszédos cella-párra ugyanígy, ez lesz egy javítási kör. Mindaddig folytassuk a javítási körök futtatását, amíg egy teljes kör lefutása során sem tudunk javítani a struktúrán, hiszen ilyenkor biztosítva van, hogy a rákövetkező körben sem sikerül már javítanunk.

Előfordulhat olyan eset a globális optimum felé vezető úton, a LA struktúrák költségterében, hogy néhány lépésben rontani kellene a számított költségen. Az újrarendező algoritmus azonban minden lépésben csak javítani tud a fennálló állapoton, ezért a lokális minimumokat nem tudja elkerülni, így azokban megáll. Ezen a ponton tehát eltér a szimulált lehűtés elvétől, ahol bizonyos valószínűséggel visszalépés is történhet.

4.3 Halmaz algoritmus

Ez az algoritmus az újrarendező algoritmust használja fel LA-struktúrák kialakításához.

Működése:

A rendszer minden celláját rakjuk bele egy LA-ba, így minden cella egy külön LA-t képez. Ez a kiindulási állapot lesz az újrarendező algoritmus bemenete. Ezután futtassuk az újrarendező algoritmust.

Az algoritmus egyik hátránya, hogy már kevés cellából álló rendszerekre is a többi algoritmushoz viszonyítva sokáig fut. Az újrarendező algoritmus egy nagyon nagy költségű struktúrát kezd el javítani, így a javító lépések száma nagy lehet. A futásidő azonban még így is elenyésző az összes szóba jövő lehetőség végigpróbálásához képest, már kisszámú cella esetén is. A halmaz algoritmus természetesen örökli az újrarendező algoritmus gyengéit is, így ez is elakadhat egy lokális költségminimumban.

5. A LA tervező program felépítése

Ahhoz hogy az előzőekben bemutatott algoritmusok hatékonyságát meg tudjuk vizsgálni egy tesztprogram lett fejlesztve az MCL laborban. A program JAVA nyelven írodott, elsősorban azért, mert a grafikus megjelenítés egyszerűen megoldható volt vele, másrészt, mert a JAVA kód hordozható, melyre szükség van a különböző architektúrákon való futtatáshoz a tesztelés során. A program alapvetően két részből áll. Az egyik része tulajdonképpen egy szimulátor, mely megadott bázisállomás-elrendezésre és úthálózatra szimulálja a mobil terminálok mozgását és a hozzájuk beérkező hívásokat. Az ebből nyert adatokat használja fel a program másik része, mellyel az implementált LA formáló algoritmusokat lehet futtatni, és hatékonyságukat összehasonlítani.

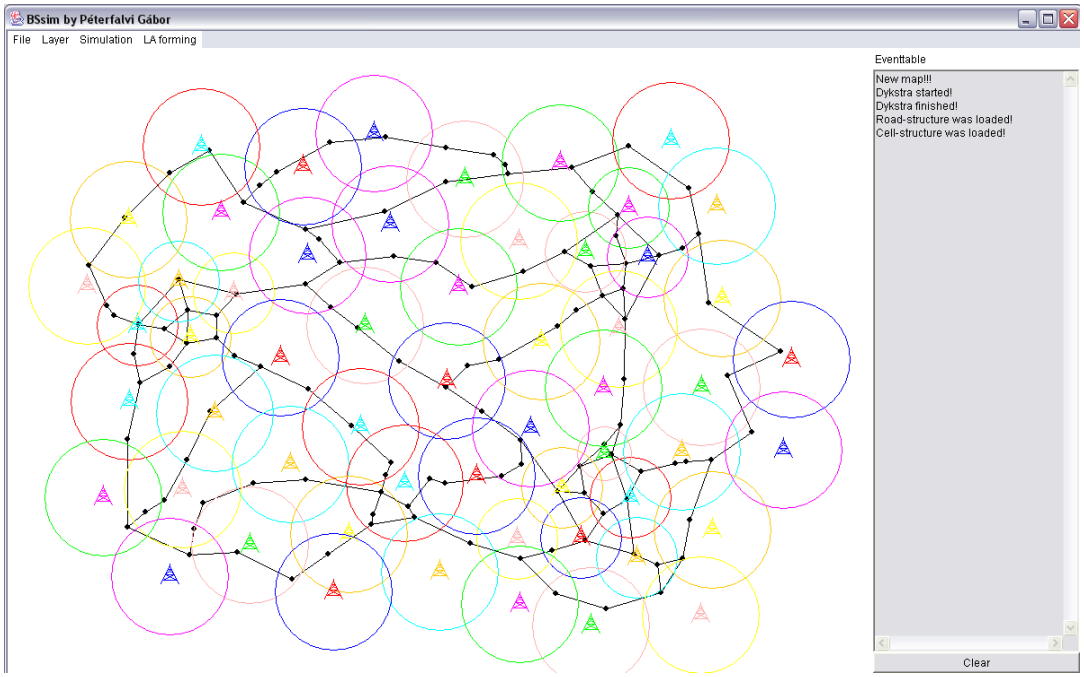
5.1 A mobilitási szimulátor

A program szimulátorával kialakíthatunk egy tetszőleges szimulációs környezetet, melyben a mobil terminálok mozognak. A szimulációs környezet úthálózatból és cella-struktúrából áll össze, melyeket egy térképre lehet lefektetni. Az úthálózat útvonalakból és ezek összekapcsolásaiból alakítató ki. Egy útvonal útpontokból és a szomszédos útpontok közötti összeköttetésekből épül fel, mely összeköttetéseket a program a legutoljára lerakott két útpont között automatikusan létrehozza. Az útpont-összekapcsolást kiválasztva egy jelzett

területen belül minden útpontot összeköthetünk. Az így kialakított útpont-hálózat tulajdonképpen egy gráf, melyben az összeköttetések, mint élek szerepelnek az útpontok között. Miután lefektettük a kívánt úthálózatot, futtatnunk kell a kialakított gráfra a legrövidebb út kereső algoritmust (Dijkstra), mely egy mátrixot ad vissza. Az úthálózat mátrix i. sorának j. oszlopa annak az útpontnak a sorszámát tartalmazza, amely az i. csomóponttól a j. csomópontig vezető legrövidebb úton az első meglátogatandó útpont. A mátrixban tehát pontról-pontra megkereshetjük tetszőleges két csomópont között a legrövidebb utat. Így a mobil terminálok a szimuláció során tudják, hogy a céljuk felé merre kell menniük. Ennek köszönhetően a szimuláció kellően valósághű lesz, hiszen a valóságban sem mozognak a mobil felhasználók céltalanul.

Az utak lefektetése után le kell fednünk a területet cellákkal, hogy a terminálok kommunikálni tudjanak. Három különböző méretű cellát rakhatunk le a térképre, melyek nagyjából a GSM, UMTS, és WLAN hotspot cellaméretek közötti különbségeket próbálják tükrözni. Érdemes a cellákat úgy elhelyezni, hogy az általuk lefedett területek metszék egymást, és így az állandóan mozgó terminálok folyamatosan elérhetőek legyenek.

Az elkészült cella-rendszert és úthálózatot el is menthetjük külön fájlba, így azok nem vesznek el a programból való kilépés során.



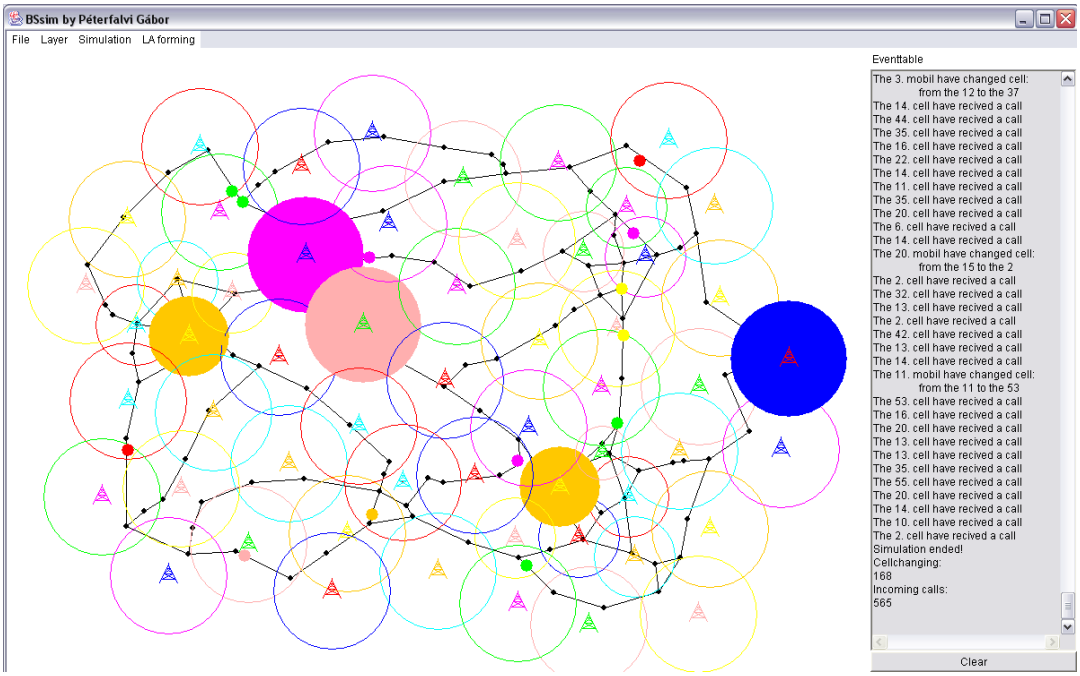
3. ábra: A lefektetett úthálózat és cella-rendszer

Miután elkészítettük a szimulációs környezetet, lerakhatunk tetszőleges számú mobil terminált is a térképre. A mobilokat csak útpontokra rakhatjuk le, és csak azután, miután futtattuk a legrövidebb útkereső algoritmust. A mobil terminálok mozgási sebessége háromféle lehet. A lassú mobiloknál kétszer, illetve háromszor gyorsabb mobilokat is lerakhatunk bármilyen eloszlásban.

A szimuláció során az egyes mobilokhoz érkező hívások között eltelt időtartamok exponenciális eloszlás szerint generálódnak. A mobilok exponenciális folyamatainak várható értékét, tehát azt, hogy átlagosan hány szimulációs lépésenként érkezzen hívás, a mobil lerakásakor meg kell adni.

A mobilok elhelyezése után elindíthatjuk a szimulációt. A szimuláció során a mobil terminálok sorsolnak egy távoli célt, és azt a legrövidebb úton közelítik meg. A terminálok a mozgás során megállapítják, hogy az éppen elhagyott csomópontból, melyik következő

csomóponton keresztül vezet a legrövidebb út a célhoz, és azon a ponton keresztül folytatják útjukat. Ha távoli céljukat elérték, várakoznak egy véletlenszerűen sorsolt egyenletes eloszlású időtartamot, majd újabb távoli célt sorsolnak. Mozgásuk során, ha egyik cellából átlépnek egy másikba, akkor azt a program feljegyzi egy cellaváltási átmenet-mátrixba; megjeleníti a térképen úgy, hogy a mobilt átszínezi az aktuális cella színére; és az eseménytáblára is kiírja a felhasználó számára. Amikor pedig egy mobilhoz hívás érkezik, az éppen aktuális cellában növeli a hívások számát; a térképen a cella felvillanásával jelzi; és ugyancsak megjeleníti az eseménytáblán. A szimulációt elindíthatjuk előre meg nem határozott időre, ebben az esetben nekünk kell leállítanunk; vagy előre megadható számú szimulációs lépésre. Abban az esetben, ha a szimulációt határozatlan időre futtatjuk, a szimuláció sebessége indítás után gyorsítható, illetve lassítható a nyomon követhetőség érdekében. Egy szimulációs lépésben a program minden mobil helyzetét egyszer frissíti. Az éppen mozgásban levő, legnagyobb sebességű mobilok esetén ez éppen egy lépésnek felel meg a célpont felé. Ha hosszabb szimulációt futtattunk, az átmenetmátrix és a hívások számának aránya az egyes cellákban az adott úthálózat struktúráját és a cellák elrendezését fogja tükrözni, mivel a mobilok véletlenszerű mozgása, hosszú távon kiátlagolódik. A szimuláció során így eltárolt adatok lesznek LA tervező algoritmusok bemeneti adatai. A program lehetőséget biztosít a cellaváltási mátrix és a cellák hívásszámainak elmentésére, így a szimuláció eredményeit az adott cella rendszerre és úthálózatra később egy következő futtatás alkalmával is felhasználhatjuk.



4. ábra A szimulátor működés közben

5.2 A LA tervező algoritmusokat megvalósító függvények

A program másik felének segítségével futtathatjuk és összehasonlíthatjuk a fentiekben tárgyalt, LA tervező algoritmusokat, melyek egy-egy függvényben lettek implementálva. Az egyes függvények az adott algoritmus által kialakított struktúrát, valamint annak költségértékeit adják vissza. Minden első hierarchiaszinten alkalmazható függvényre igaz, hogy a futása elején törli az előzőleg kialakított cella-struktúrát, így sorozatban, egymás után tetszőleges számú függvényt futtathatunk. A program lehetőséget biztosít arra is, hogy a felhasználó kijelöléssel, manuálisan maga alakítson ki tetszőleges cella-struktúrát tetszőleges

hierarchiaszinten. Az így intuitívan kialakított cella-struktúrák költségei alapjául szolgálhatnak az egyes algoritmusok eredményeinek összehasonlításában. A programban az algoritmusok által kialakított cella-struktúrákat elmenthetjük, így például a manuálisan kialakított struktúrák bármikor újra előállíthatók.

6. Ellenőrző kérdések

1. Miért van szükség Location Area-k kialakítására?
2. Milyen két egymásnak ellentmondó szempont nehezíti a Location Area optimalizálást?
3. Miért van szükség pagingre? Vezesse le a paging költségfüggvényt egy Location Area-ra!
4. Írja le a Location update költségfüggvényt egy Location Area-ra.
5. Mi a különbség az alap és a kiterjesztett alap algoritmus között?
6. Írja le az újrarendező algoritmus működését!
7. Hogyan épül fel a LA tervező program?

7. Mérési feladatok

A mérési feladatok elkezdése előtt a mérésvezető röviden bemutatja a LA tervező programot. A feladatokban 4 algoritmust kell használni: alap – basic, kiterjesztett alap – basic best, halmaz – distribution, újrarendező – regrouping (a kiinduló LA struktúrát kiterjesztett alap algoritmussal kell generálni).

1. Tervezzünk egy legfeljebb 12 cellából álló mobil rendszert, úthálózattal, mobil terminálokkal. Hasonlítsuk össze, hogy az implementált LA tervező algoritmusok mennyire jól közelítik meg az optimális felosztást, melynek a cellaváltás és paging költségeinek összege minimális az adott forgalmi körülmények és költségfüggvények mellett!
    a) Készítsen táblázatot és százalékos kimutatást az eredményekről (4 algoritmus, paging-cellaváltás-összes költség)!
    b) Változtatva a cellaváltási és paging költség közötti arányt (Set difference), hogyan változik az eredmény? Próbálja ki több értékre is! (eredmények összefoglalása táblázatba)
    c) Konklúzió?

2. Tervezzünk egy városi és egy rurális mobilitási környezetet! A városi környezet jellemzői a sűrű úthálózat, a kisméretű cellák, és a nagyszámú, lassan mozgó, nagy beérkező hívás intenzitású terminál. A rurális (városon kívüli) környezetre a kevés útpontból és útvonalból álló úthálózat, a nagyméretű cellák, és viszonylag kisszámú, nagysebességű mobil terminál a jellemző (Városi környezet: 30-40 cella, kb. 20 mobil terminál, rurális környezet: 10-30 cella, kb. 10 mobil terminál).
    a) Az úthálózat függvényében készítsen egy manuális LA felosztást (egy LA-ba tartozzanak azok a cellák amelyek azonos mozgási irányok mentén fekszenek) és hasonlítsa össze az implementált algoritmusok által kapott eredményekkel!Melyik a leghatékonyabb cellaváltási költség, paging költség illetve összes költség szempontjából? Készítsen táblázatot az eredményekről!
    b) Változtatva a cellaváltási és paging költség közötti arányt (Set difference), hogyan változik az eredmény? Próbálja ki több értékre is!(eredmények összefoglalása táblázatba)
    c) Miben különbözik az eredmények függvényében a városi és a rurális mobilitási környezet? Melyiknél melyik a leghatékonyabb algoritmus illetve költség arány?