Szegedi Tudományegyetem

Informatikai Tanszékcsoport

Körfedés és alkalmazása a telekommunikációs hálózatokban

TDK dolgozat

Készítette:

Palatinus Endre

programtervező informatikus MSc szakos hallgató, 2. évfolyam

Témavezető:

Dr. Bánhelyi Balázs

egyetemi adjunktus

Szeged

2010

Tartalmi összefoglaló

A pakolási problémákat régóta nagy figyelem övezi, amelyeket még napjainkban is sokat vizsgálnak. Egyik népszerű feladat ebben a témakörben a körpakolás [1]. Ennek célja adott számú egybevágó, páronként diszjunkt körnek az egységnégyzetbe való legsűrűbb pakolásának a megadása.

A feladat duálisának tekinthető probléma a körfedés. Ez a probléma sokkal nehezebbnek bizonyul, amelyet a témában elért eredmények mennyisége is tükröz. Ebben az esetben adott számú egybevágó körrel az egységnégyzet legritkább lefedését keressük, ahol legritkább alatt azt értjük, hogy a lehető legkisebb sugarú körökkel fedjük le a négyzetet.

Az intervallum-aritmetikát használtuk fel annak eldöntésére, hogy a körök egy adott elrendezése teljesen lefedi-e az egységnégyzetet. Az előbbi kérdés eldöntésére egy branch-and-bound alapú eljárást fejlesztettünk ki. Az eljárásunk párhuzamosított változatát is megvizsgáltuk, amikor egyidejűleg több részproblémája is végrehajtható a branch-and-bound eljárásnak több CPU-mag felhasználásával.

Az eljárásunk hatékonyságát bemutatjuk egy telekommunikációs problémán. A fő rádióadó-tornyok pozícióinak ismeretében határozzuk meg Magyarország rádióadásokkal való optimális lefedését. Az egyes rádióadó-tornyok által kibocsájtott rádióhullámok kör alakú területeket fednek le, és ezen területek összegét szeretnénk minimalizálni, mely arányos a rádióadó-torony sugárzási energiájának nagyságával. Ennél a problémánál a körök sugarai különbözőek is lehetnek, és az optimális sugárméretek meghatározására egy branch-and-bound alapú megbízható optimalizáló eljárást alkalmaztunk.

1. fejezet

Bevezetés

Ebben a fejezetben ismertetjük a körfedési problémát, és bemutatunk egy jól skálázható, általános célú eljárást, amellyel többdimenziós intervallumok tulajdonságait vizsgálhatjuk. Ezután megmutatjuk, hogyan lehet ezt felhasználni annak eldöntésére, hogy a körök egy adott elrendezése teljesen lefedi-e az egységnégyzetet.

1.1. A körfedés probléma

A körfedés probléma a körpakolási feladat duálisának tekinthető. Ebben az esetben adott számú egybevágó körrel az egységnégyzet legritkább lefedését keressük, ahol legritkább alatt azt értjük, hogy a lehető legkisebb sugarú körökkel fedjük le a négyzetet. Optimális megoldásokat csak kisszámú kör esetén találtak (lásd [3]) és az optimalitásukat matematikailag bizonyították. A körök számának növekedésével az elhelyezkedési "mintáik" száma drasztikusan növekszik, amely nagyban megnehezíti az optimális megoldás megtalálását, az optimalitás matematikai módon való bizonyítását pedig még inkább.

Hasznos lehet számítógépes programokat írni a probléma megoldására, de a körök elhelyezkedési "mintáinak" többszörös szimmetriája, valamint a számítási bizonytalanságaink kezelése nagy nehézségeket okoznak. Mi a fedés eldöntésének kérdésevel foglalkoztunk annak érdekében, hogy minél gyorsabb eljárást dolgoz-

zunk ki ennek a részproblémának a megoldására.

1.2. A branch-and-bound–alapú párhuzamosított ellenőrző eljárás

Az általunk kifejlesztett algoritmus egy egyszerűsített B&B eljárás, amely adott korlátok mellett képes megbízhatóan bizonyítani egy többdimenziós intervallum tetszőleges tulajdonságát, felhasználva erre több CPU magot is a futásidő csökkentése érdekében [4]. A működéséhez szükséges egy olyan eljárás, amely egy adott intervallumról képes megbízhatóan megállapítani, hogy rendelkezik egy adott tulajdonsággal, azonban az ellenkezőjét nem kell megbízhatóan eldöntenie.

Bemenetként egy többdimenziós intervallumot vár, és a kimenete pedig igaz, ha az intervallum rendelkezik az adott tulajdonsággal, különben pedig visszatér egy részintervallummal, amely vagy kivételt képez, vagy az adott korlátok mellett nem dönthető el róla, hogy rendelkezik-e az adott tulajdonsággal.

Az eljárás futása közben létrehozzott részintervallumok minimális hosszát rögzítjük, és ezt ϵ-nal jelöljük. Erre hatákonysági okokból van szükség, mivel ez a paraméter nagyban befolyásolja az eljárás átlagos futásidejét. Ezzel korlátozzuk az eljárásunk bizonyító erejét is, azonban a legtöbb probléma esetében ennek a paraméternek az értéke megválasztható úgy, hogy elfogadható eredményt produkáljon az eljárás.

Egy másik hatékonyságbeli tényező az eljárás által egyidejűleg futtatható szálak maximális száma, amelyet $thread\_max$-szal jelölünk, és ennek felső korlátja a felhasznált számítógépben lévő CPU magok száma.

Az *algoritmus szerint* a következő esetekben nem rendelkezik a bemeneti intervallum a vizsgált tulajdonsággal:

- Egy részintervalluma nem rendelkezik az adott tulajdonsággal.
- Egy ϵ-nál kisebb méretű részintervallumáról nem tudjuk megbízhatóan bizonyítani, hogy rendelkezik az adott tulajdonsággal.

Tekintve, hogy a megbízható ellenőrző eljárásunk csak pozitív válasz esetén tér vissza megbízható eredménnyel, az algoritmusunk negatív válasz esetén mindig feloszt egy intervallumot az ϵ-os méretkorlátig, azonban az első ilyen találat esetén terminál.

Az algoritmus nem párhuzamosított változatának helyességigazolása megtalálható [5]-ben. A B&B eljárás független részproblémákat állít elő, amely lehetővé teszi a párhuzamosítását. Könnyen belátható, hogy ez a párhuzamosított algoritmus is megvizsgálja az összes részproblémát, és így szintén helyes matematikailag, valamint véges lépésben befejeződik a futása. A többszálú végrehajtás következtében a várakozásunk az volt, hogy a CPU-magok számával arányos sebességnövekedést tudunk majd elérni.

Az implementáció során néhány dologra különös figyelmet kellett fordítanunk: Minden szál felfüggesztett üzemmódban várakozik, amíg az általa indított gyerek szálak futása befejeződik, így az ilyen szülő szálak nem számítanak futó szálaknak. Az eljárás holtpont-mentes, mivel zárakat használtunk a szálak által közösen használt változók kezelésére. A számításaink bizonytalanságának kezelésére intervallum-aritmetikát használtunk, és a kódunkban ezt a C-XSC (lásd [6]) segítségével valósítottuk meg, amely egy széles körben elismert C++ osztálykönyvtár nagy pontosságú tudományos számítások elvégzésére.

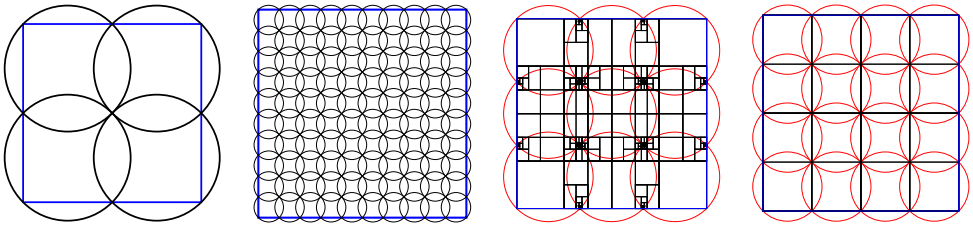
1.3. A körfedés probléma eldöntése

Itt a korábban bemutatott párhuzamosított ellenőrző eljárást alkalmaztuk a következők szerint: A bemeneti intervallum maga az egységnégyzet. Egy részintervallum vizsgált tulajdonsága, hogy vajon le van-e teljesen fedve bármelyik kör által. Ez azonban nem mindig dönthető el a véges pontosságú aritmetikánk miatt, így minden kört nyílt halmazoknak tekintünk a síkon. A megbízható ellenőrző eljárásunk pedig akkor tér vissza pozitív válasszal, ha a

$$sup(x - center) < inf(r^2)$$

egyenlőtlenség fennáll az intervallum minden x pontjára, ahol $center$ a kör középpontját, r pedig a kör sugarát jelöli.

Eljárásunk hatékonyságának tesztelésére néhány egyszerű tesztesetet használtunk fel: n^2 kör egy $n \times n$-es szabályos rácson elhelyezve úgy, hogy az átlós szomszédok érintsék egymást. Valójában két, egymáshoz nagyon közeli metszéspontjuk van, mivel a körök sugarát megnöveltük egy kis $\epsilon_r > \epsilon$ értékkel, mert különben a metszéspontjukról nem tudnánk eldönteni, hogy le van-e fedve, vagy sem. Néhány példát láthatunk erre az 1.1 ábrán.



(a) $n = 2$ (b) $n = 10$ (c) $n = 3$ (d) $n = 4$

1.1. ábra. Tesztesetek a körfedés problémához

Az eljárásunknak az előbbi teszteseteken való futás közbeni működésére az 1.1 ábrán láthatunk példákat. A kék négyzetek az egységnyégyzetet jelölik, a fekete négyzetek pedig a B&B eljárás által képezett részintervallumokat. Abban a speciális esetben, amikor n 2-nek a hatványa, akkor a legkisebb részintervallumok a körök beírt négyzetei. Mivel a körök teljesen lefedik a beírt négyzetüket, ezeket az intervallumokat már nem osztja tovább fel az eljárás. Ebből kifolyólag ezek a speciális tesztesetek nem alkalmasak a teljesítmény mérésére, mert az algoritmus futásideje ezeken nagyon kicsi lesz. Másrészről viszont, ha n nem 2-nek a hatványa, akkor a B&B-eljárás nem hoz létre olyan részintervallumokat, amelyek valamely kör beírt négyzetei lennének, mivel mindig két egyenlő részre osztja fel az intervallumokat, a legszélesebb oldaluk mentén. Ebben az esetben több részintervallum keletkezik a körök metszéspontjai körül, mivel ezeknek egy ϵ_r nagyságrendű környezetük van csak lefedve a körök által.

2. fejezet

A körfedés alkalmazása a telekommunikációs hálózatokban

Amint azt az előző fejezetben láthattuk, annak eldöntése, hogy az egységnégyzet le van-e fedve körök által vagy sem, egy viszonlag egyszerű feladat volt, és nincs különösebb ipari alkalmazhatósága. Azonban összetettebb alakzatok körökkel való optimális lefedése már nagyobb kihívást jelent, és hasznosabbnak bizonyulhat. Ebből kifolyólag megvizsgálunk egy telekommunikációval kapcsolatos feladatot, amelyben felhasználhatjuk a korábbi eredményeinket.

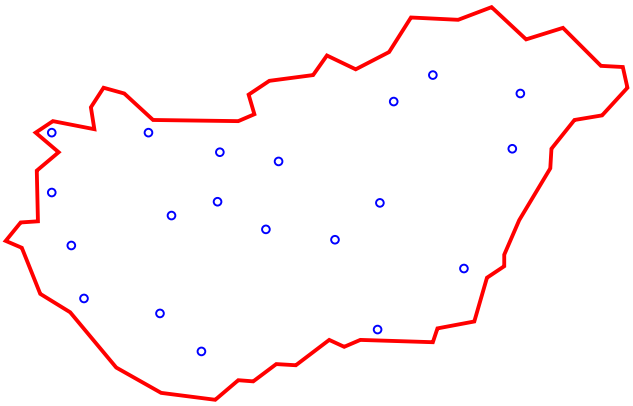
2.1. Magyarország rádióadásokkal való optimális lefedése

A feladatunk a fő rádióadó-tornyok pozícióinak ismeretében meghatározni Magyarország rádióadásokkal való optimális lefedését. Az rádióadó-tornyok elhelyezkedését a 2.1 ábrán láthatjuk. Az egyes rádióadó-tornyok által kibocsájtott rádióhullámok kör alakú területeket fednek le, amelyeknek a sugara a következő képlet alapján számítható ki:

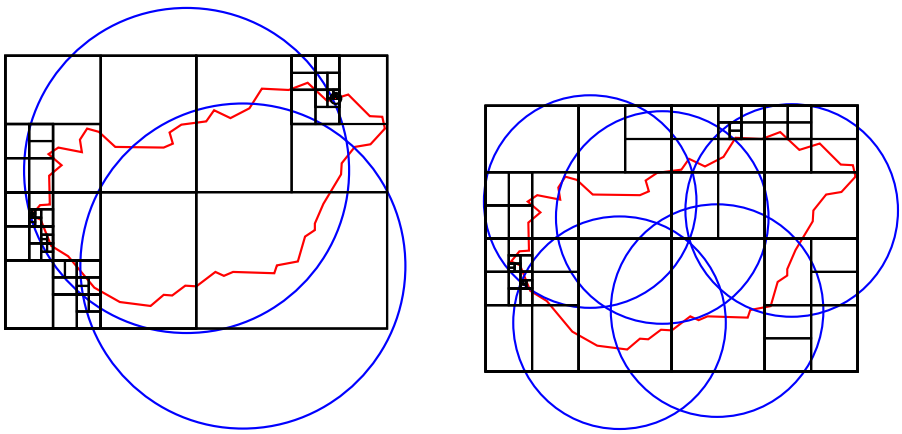
$$r = \frac{\lambda}{4\pi}\sqrt{\frac{P_{ado}}{P_{vevo}}}$$

ahol λ a rádióadás hullámhosszát jelöli, P_{ado} a rádióadó-toronyba betáplált teljesítményt, P_{vevo} pedig az a minimális teljesítmény, amivel a rádióhullámnak rendelkeznie kell ahhoz, hogy a vevő érzékelni tudja azt.

Ez a képlet a terepviszonyokat nem veszi figyelembe, mivel csak sík, számottevő akadályok nélküli terepen érvényes, így egy egyszerűsített modellnek tekinthető. Ennek legjobban egy C-osztályba tartozó rádióadó modellje feleltethető meg a Federal Communications Commission szabályozásai alapján, amiből az következik, hogy a vevők minimális érzékenysége $6.31E-07$ Watt.



2.1. ábra. A fő rádióadó-tornyok elhelyezkedése



(a) Nincs fedés (b) Teljesen lefedve

2.2. ábra. A B&B ellenőrző eljárás futása

A minél kisebb üzemeltetési költségek elérése érdekében a rádióadások által

lefedett területek összegét szeretnénk minimalizálni, mely arányos a rádióadó-tornyok sugárzási energiáinak összegével.

Ennél a feladatnál is alkalmazhatjuk a párhuzamosított ellenőrző eljárásunkat a következők szerint: A bemeneti intervallumunk egy tetszőleges intervallum, amely magába foglalja Magyarországot. Egy intervallum vizsgált tulajdonsága, hogy le van-e fedve bármely kör által, és hogy van-e közös pontja Magyarországgal. Ez utóbbi feltétel az egyetlen bővítés az előző megoldáshoz képest, amelyet a pont helyzetének vizsgálatára szolgáló közismert geometriai algoritmussal oldottunk meg, azonban itt a pontok szerepét felváltották az intervallumok. Az ellenőrző eljárásunk futását két teszteseten a 2.2 ábrán láthatjuk. Az első esetben két rádióadó-tornyunk van, amelyek az ország nagy részét lefedik. Az algoritmus által elsőként megtalált intervallum, amely az ország területén belül van, de nincs lefedve egyik kör által sem, a kép jobb felső sarkában látható kis körrel van megjelölve. A második esetben öt rádióadó-tornyunk van, amelyek teljesen lefedik az országot.

2.2. Az optimális sugárhosszak meghatározása

A feladat bemutatásakor kitértünk arra, hogy az adótornyok fogyasztása, azaz az általuk felhasznált energia egyenesen arányos az adótornyok által lefedett területek nagyságával. Azt is megemlítettük, hogy az adótornyok kör alakú területeket fednek le rádióadással. Ily módon átfogalmazhatjuk a feladatot a következőképpen: fedjük le Magyarországot adott középpontú körökkel úgy, hogy a körök által lefedett területek összege minimális legyen.

2.2.1. A feladat matematikai vizsgálata

2.1. Definíció. *Jelölje $r = (r_1, r_2, \ldots, r_n)$ az adótornyok által lefedett kör alakú területek sugárhosszait tartalmazó vektort. Ezt a továbbiakban konfigurációnak fogjuk nevezni.*

Ezt a jelölést felhasználva a célfüggvényünk a következő:

$$f\left((r_1, r_2, \ldots, r_n)\right) = \sum_{i=1}^{n} r_i^2 \to min \tag{2.1}$$

2.2. Definíció. *Egy r konfigurációt jónak nevezünk, ha az általa reprezentált körök lefedik Magyarországot. Ellenkező esetben a konfigurációt rossznak nevezzük.*

Most bebizonyítunk néhány egyszerű állítást a konfigurációkkal kapcsolatban.

2.3. Állítás. *Ha egy $r = (r_1, r_2, \ldots, r_n)$ konfiguráció jó, akkor bármely $r' = (r'_1, r'_2, \ldots, r'_n)$ konfiguráció is jó, ha $r'_i \geq r_i$ minden i-re.*

Bizonyítás. Mivel az r konfiguráció jó, így az általa reprezentált körök lefedik Magyarországot. Ha ezen körök közül bármelyiknek megnöveljük a sugarát, akkor a kapott körök szintén lefedik Magyarországot. Az utobbi gondolatot ismételve minden i-re látható, hogy r' is jó konfiguráció. □

2.4. Állítás. *Ha egy $r = (r_1, r_2, \ldots, r_n)$ konfiguráció rossz, akkor bármely $r' = (r'_1, r'_2, \ldots, r'_n)$ konfiguráció is rossz, ha $r'_i \leq r_i$ minden i-re.*

Bizonyítás. Hasonlóan az előző állításhoz. □

2.5. Definíció. *Jelölje $C = ([\underline{c_1}, \overline{c_1}], \ldots, [\underline{c_n}, \overline{c_n}])$ azon konfigurációk halmazát, amelyek i-edik komponense a $[\underline{c_i}, \overline{c_i}]$ intervallumba esik. A C halmaz két kitüntetett szerepű konfigurációja a következő: $\underline{C} = (\underline{c_1}, \ldots, \underline{c_n})$ és $\overline{C} = (\overline{c_1}, \ldots, \overline{c_n})$.*

2.6. Definíció. *A feladatunk optimális megoldása egy olyan **jó** konfiguráció, amelyre a 2.1 célfüggvény értéke minimális.*

A konfigurációk egyszerű tulajdonságai hasznosnak bizonyulnak az optimális megoldás keresése során, amelyet a következő állításokban részletezünk:

2.7. Tétel. *Ha egy C halmaz $\underline{C}$ és $\overline{C}$ kitüntetett konfiguráció rosszak, akkor az optimális megoldás nem lehet a $C' = ([0, \overline{c_1}], \ldots, [0, \overline{c_n}])$ halmazban.*

Bizonyítás. Mivel a $\overline{C}$ konfiguráció rossz, a 2.4 állításból következik hogy a C' halmaz összes konfigurációja is rossz. Az optimális megoldás azonban egy jó konfiguráció, így nem lehet eleme a C' halmaznak. □

2.8. Tétel. *Ha egy C halmaz $\underline{C}$ és $\overline{C}$ kitüntetett konfigurációi jók, akkor az optimális megoldás nem lehet a $C \setminus \underline{C}$ halmazban.*

Bizonyítás. A C halmazon a 2.1 célfüggvény a $\underline{C}$ pontban veszi fel a minimumát, amely felső korlátja az optimumnak, mivel $\underline{C}$ egy jó konfiguráció. Ebből következik, hogy az optimális megoldás nem lehet a megadott halmazban. □

Megjegyzendő, hogy nyitott körök esetén a $\underline{C}$ konfiguráció sem optimális, ugyanis létezik olyan kis $\epsilon > 0$, hogy $C_\epsilon = (\underline{c_1} - \epsilon, \ldots, \underline{c_n} - \epsilon)$ is jó konfiguráció és $f(C_\epsilon) < f(\underline{C})$.

2.9. Állítás. *Nem létezik konfigurációk olyan C halmaza, amelynek $\underline{C}$ konfigurációja jó és $\overline{C}$ konfigurációja pedig rossz.*

Bizonyítás. Mivel a $\underline{C}$ konfiguráció jó, a 2.3 állításból következik, hogy $\overline{C}$ is jó konfiguráció, ami ellentmondás. □

2.10. Következmény. *Csak olyan C halmaz, amelynek $\underline{C}$ konfigurációja rossz, és $\overline{C}$ konfigurációja jó, tartalmazhatja az optimális megoldást.*

2.2.2. Az optimalizáló eljárás

Most bemutatjuk az optimális sugárhosszak meghatározására szolgáló algoritmust, majd igazoljuk a helyességét.

A SUGÁRHOSSZAKAT MEGHATÁROZÓ OPTIMALIZÁLÓ ELJÁRÁS

- Q legyen egy üres, minimumos prioritási sor, amelyben a C_i-vel jelölt elemek prioritását a $\underline{C_i}$ konfiguráción felvett célfüggvényértékük határozza meg.
- Tegyük be a $C_0 = ([0, c_1], \ldots, [0, c_n])$ kezdeti konfigurációhalmazt a Q sorba, ahol a $\overline{C_0}$ egy olyan jó konfiguráció, amelyre a célfüggvény értéke kellően nagy.

- **Ciklus:**
  - Vegyünk ki egy C konfigurációhalmazt a Q sorból.
  - Osszuk fel két egyenlő részre a C halmazt, mint többdimenziós intervallumot, a legszélesebb oldala mentén a C_1 és C_2 halmazokra úgy, hogy $\underline{C} = \underline{C_1}$ és $\overline{C} = \overline{C_2}$ teljesüljönek.
  - Ha $\overline{C_1}$ jó konfiguráció, akkor tegyük be C_1-et a Q sorba.
  - Ha $\underline{C_2}$ rossz konfiguráció, akkor tegyük be C_2-t a Q sorba.
- **Amíg** C legszélesebb oldalának hossza $> \epsilon$
- Visszatérünk a $\overline{C}$ megoldással.

2.11. Tétel. *Az előző algoritmus megbízhatóan meghatározza a feladatunk optimális megoldásának egy ϵ-sugarú környezetét.*

Bizonyítás. Válasszuk meg a $\overline{C_0}$ konfigurációt úgy, hogy minden komponense nagyobb legyen, mint Magyarország két legtávolabbi pontjának a távolságának a fele. Ebből egyszerűen következik, hogy C_0 tartalmazza az optimális megoldást.

Az algoritmus ezután minden iterációban a C_0 halmaznak egy egyre finomodó partícionálását állítja elő, azaz minden lépésben egy meglévő partícióját bontja tovább. Az algoritmus által használt sorba ezek közül csak olyan C partíciók kerülnek be, amelyeknek a $\underline{C}$ konfigurációja rossz és $\overline{C}$ konfigurációja pedig jó. Könnyen belátható, hogy az aktuális partícionálás összes ilyen tulajdonságú partíciója mindig benne van a sorban. A 2.10 következmény alapján ezek közül az egyik tartalmazza az optimális megoldást.

Igaz továbbá az is, hogy a prioritási sorban lévő C_i halmazok $\overline{C}_i$ konfigurációiban felvett célfüggvényértékek minimuma felső korlátja az optimumnak. A prioritási sorunk tulajdonságából következik, hogy az aktuális iterációban a sorból kivett C halmaz $\underline{C}$ konfigurációjában felvett célfüggvényérték pedig alsó korlátja az optimumnak. Mivel az utolsó iterációban a sorból kivett C halmaznak, mint intervallumnak, a legszélesebb oldalának a hossza $\leq \epsilon$, az előzőek felhasználásával kapjuk, hogy $\overline{C}$ az optimális megoldásnak egy ϵ-sugarú környezetében van. □

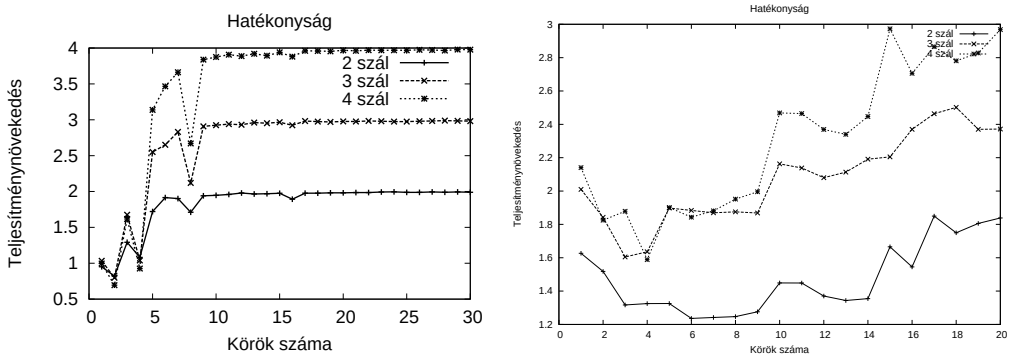
3. fejezet

Hatékonysági kérdések és eredmények

3.1. A körfedés eldöntésének hatékonysága

Az intervallumos ellenőrző eljárásunkat egy párhuzamosított környezetben vizsgáltuk meg, amikor a B&B–eljárás különböző lépései egyidejűleg több CPU magon is végrehajthatóak. Tesztjeinek egy 4-magos Sun számítógépen végeztük el, így az egyidejűleg futó szálak maximális száma 4 lehetett. A várakozásunk az volt, hogy több kör esetén az ellenőrző eljárás által létrehozott részintervallumok száma igen nagy lesz, és ennek megfelelően a többszálú végrehajtás javíthat a futásidőn.

A 3.1 ábrán látható teljesítmény-grafikonokról leolvasható, hogy a körfedés probléma eldöntése esetében 11×11 körtől kezdve az egyszálú végrehajtáshoz viszonyított teljesítménynövekedés arányos a CPU-magok számával. Magyarország lefedése esetében már nem látható ilyen egyértelmű kapcsolat, mivel az itt mért eredmények az egyszerű optimalizáló futásidején alapulnak. Több kör esetén azonban itt is megfigyelhető a teljesítménynövekedés. A körfedés esetében a futásidő a másodperc törtrésze volt, Magyarország lefedése esetében pedig a körök számától függően akár néhány perc is.



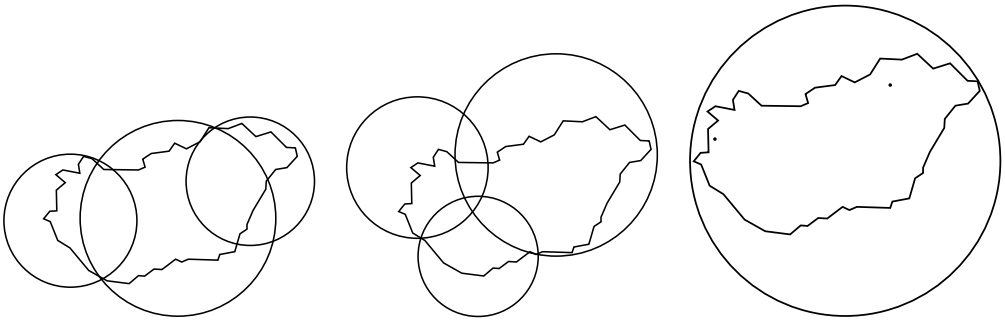
(a) Az egységnégyzet lefedése (b) Magyarország lefedése

3.1. ábra. Teljesítménynövekedés a CPU-magok számának függvényében.

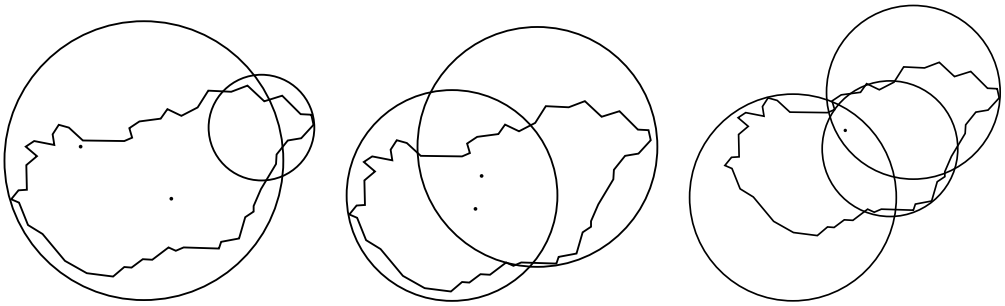
3.2. Magyarország optimális lefedése

A következőkben az optimális sugárhosszakat meghatározó eljárásunk által talált eredmények közül ismertetünk néhányat. Minden esetben az optimális megoldás 1 kilométer sugarú közelítését láthatjuk, ami elfogadhatónak mondható. Ezen eredmények meghatározásához a körök számától függően néhány órára is szükség volt.

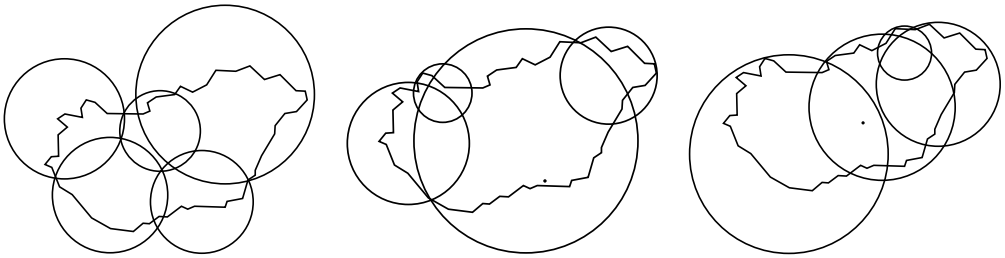
Az ábrákon látható, hogy a kiválasztott rádióadó-tornyok egymáshoz viszonyított elhelyezkedésétől függően az optimális megoldásban bizonyos tornyok 0-sugarú területet fednek le, azaz kedvezőbb, ha nem használjuk fel őket.



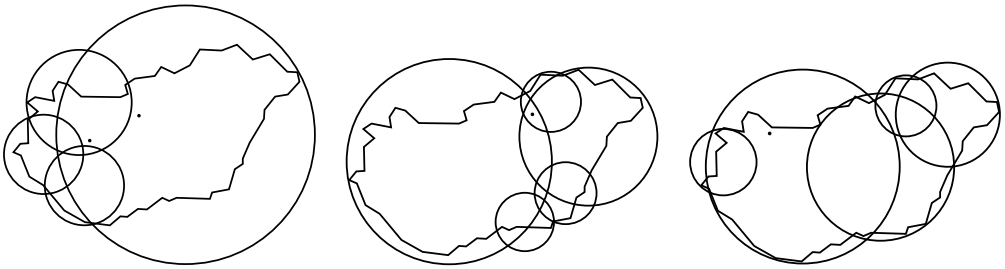
(a) n=3 (b) n=3 (c) n=3



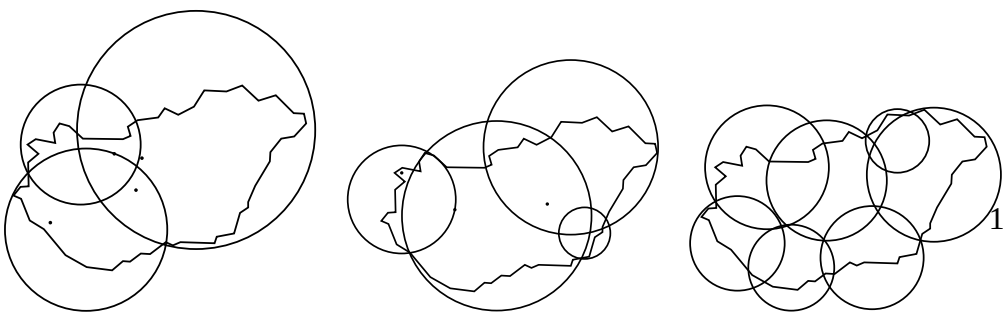
(d) n=4 (e) n=4 (f) n=4



(g) n=5 (h) n=5 (i) n=5



(j) n=6 (k) n=6 (l) n=6



(m) n=7 (n) n=7 (o) n=7

3.2. ábra. Optimális lefedettségi térképek

Irodalomjegyzék

[1] SZABÓ, P.G., M.CS. MARKÓT, T. CSENDES, E. SPECHT, L.G. CASADO, AND I. GARCÍA: New Approaches to Circle Packing in a Square – With Program Codes. *Springer, Berlin*, 2007.

[2] DAS, G.K., S. DAS, S.C. NANDY, AND B.S. SHINA: Efficient algorithm for placing a given number of base station to cover a convex region, *J. Parallel Distrib. Comput.* **66**(2006), 1353–1358.

[3] Erich Friedman: Circles Covering Squares. *http://www2.stetson.edu/ efriedma/circovsqu/*, (2005)

[4] CASADO, L.G., J. A. MARTINEZ, I. GARCIA, AND E.M.T. HENDRIX:, Branch-and-Bound interval global optimization on shared memory multiprocessors, *Optimization Methods Software*, **23**(2008), 689–701.

[5] Bánhelyi, B., Csendes, T. and Garay B. M.: A Verified Optimization Technique to Locate Chaotic Regions of Hénon Systems. *Journal of Global Optimization*, **35**(2006), 145–160.

[6] Hofschuster, Krämer, Wedner, Wiethoff, *C-XSC 2.0 - A C++ Class Library for Extended Scientific Computing*, Preprint BUGHW-WRSWT 2001/1, Universität Wuppertal, 2001.

[7] Palatinus, E., Bánhelyi, B.: Circle covering and its applications for telecommunication networks. *Proceedings of the 8th International Conference on Applied Informatics*, Eger (2010)