SZIMULÁCIÓS FELADAT

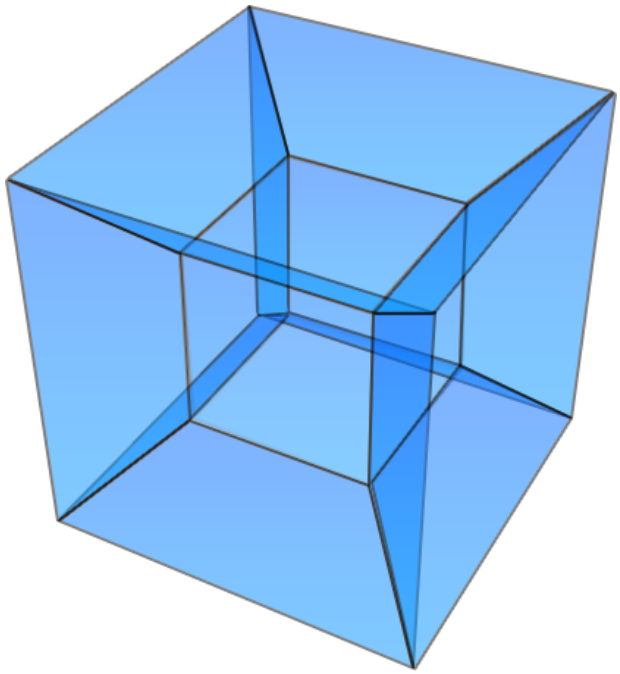
GÓCZA GERGELY

1. A probléma

1.1. Bevezetés Szimulációm elsősorban az ún. hiperkocka elrendezésű hálózatok viselkedésével foglalkozik. Azon belül is elsősorban a hálózaton belüli vezetőválasztás problémáját vizsgálja az előadáson ismertetett általános hálózatokra kitalált OptMaxTerjed nevű algoritmussal.

1.2. Szükséges ismeretek A hiperkocka elrendezésű hálózatok fő előnye a viszonylag nagy csúcsszám melletti kis átmérő. Egy d dimenziós hiperkocka 2^d darab csúcsot tartalmaz, ám emellett átmérője mindössze d. Elképzeléséhez szemléletes módszer lehet, ha az adott d dimenziós kockát két darab $d-1$ dimenziósból készítjük el, méghozzá úgy, hogy az egyik kisebb kocka csúcsait a másik megfelelő csúcsaival összekötjük egy irányítatlan éllel.

A kocka számítógépen történő ábrázolásának kézenfekvő módja az, hogy minden csúcsot egy d bites számmal azonosítunk. Két csúcs között pontosan akkor fut él, ha az őket címkéző d bites számok egyetlen bitben térnek el.



1.1. ábra. 4 dimenziós hiperkocka

1.3. Szimulációm célja Elsődleges célom annak megvizsgálása, hogy hogyan hat a szükséges üzenetszámra a kocka dimenziószámának növekedése, illetve mennyire befolyásoló tényező a csúcsok indexeinek permutációja.

2. A szimuláció menete

2.1. Előkészítés Az algoritmus szimulációja előtt elő kell állítani a bemenetül szolgáló hiperkockát. Ehhez a kocka csúcsaihoz egyedi azonosítót kell rendelni. Az általánosság megsértése nélkül feltehetjük, hogy a csúcsok azonosítói a $\left[0,\ 2^d - 1\right]$ intervallumba esnek. Ekkor az előkészítés a kérdéses intervallum egy permutációjának előállítása.

Ezen felül kezdetben minden csúcs a saját azonosítóját ismeri maximálisként, illetve az algoritmus által előírt „aktív" állapotban van.

2.2. Egy menet futtatása Egy menet d darab lépésből áll.

Minden lépés során az aktív csúcsok elküldik a szomszédaiknak az általuk ismert legnagyobb azonosítot. Azon csúcsok amelyek ezáltal új információhoz jutnak aktívak lesznek a következő lépésben, amelyek nem, azok passzívak.

2.3. Eredmény kiértékelése Minden menet után az adott kockában futtatott szimuláció ütemei alatt elküldött üzenetszámokból álló adatsort kapok eredményül. Az adatsorok által kifeszített görbék alakjából, illetve a köztük lévő különbségekből igyekszem következtetést levonni az eredeti célkitűzéssel kapcsolatban.

3. A megvalósított program

Az alábbiakban a programkód részletes ismertetése helyett az alapelgondolásokat és a program használatát fejtem ki.

3.1. Adatszerkezet A szimuláció során nem kezeltem a csúcsokat külön. Ehelyett csak a rájuk vonatkozó fontosabb adatokat tároltam el, melyek:

- az ismert legnagyobb azonosító
- az adott lépés során megismert új legnagyobb azonosító
- aktivitásjelző

Ezeket egy-egy 2^d méretű tömbben tárolom, melyet az algoritmus közvetlenül módosít.

3.2. Bemenő paraméterek A program bemenetként 2 paramétert vár. Ezek

(1) a szimulálandó hiperkocka dimenziója
(2) a futtatandó menetek száma

3.3. Kimenet formája Futás közben az alapértelmezett kimeneten az egyes menetek üzenetszámai jelennek meg ütemekre bontva, végükön egy összesített üzenetszámról informáló sorral. Példa egy 5 dimenziós menetre:

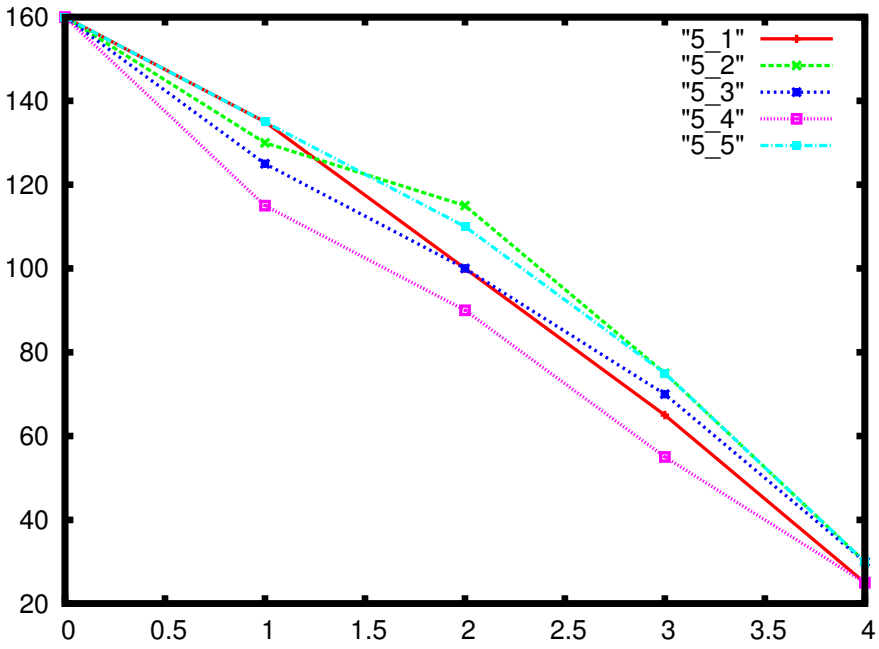
- 160 üzenet az 1. ütemben
- 110 üzenet az 2. ütemben
- 70 üzenet az 3. ütemben
- 55 üzenet az 4. ütemben
- 25 üzenet az 5. ütemben
- Szimuláció vége. Összes küldött üzenetek száma: 420

Emellett a program minden menethez készít egy kimeneti fájlt is, melyben sorokra bontva az egyes ütemek üzeneteinek száma jelenik meg. Az egyes fájlok neve D_n alakú, ahol D jelöli az adott szimulálandó hiperkocka dimenzióját, n pedig az adott menet sorszámát.

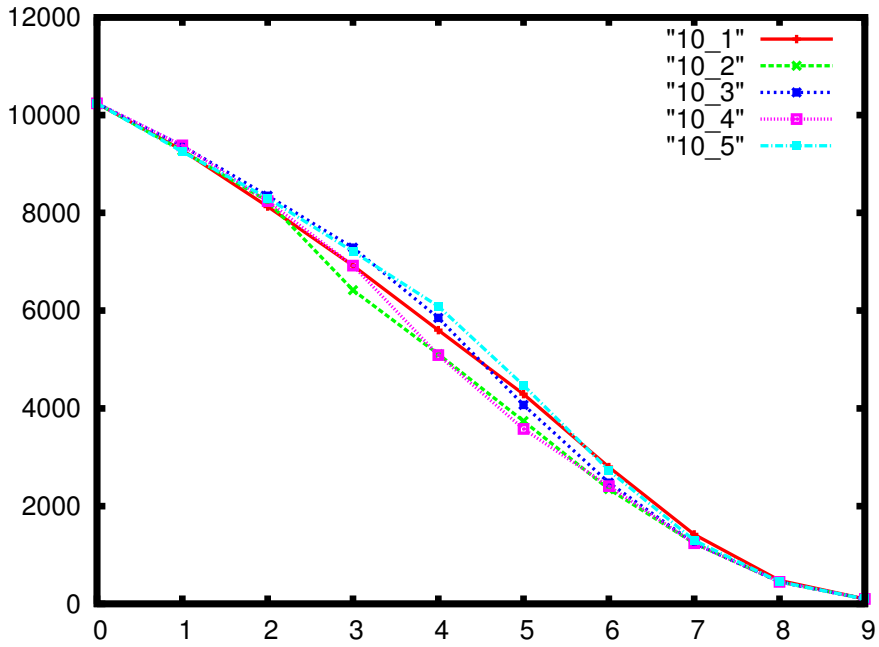
3.4. Grafikon készítése Az eredmények szemléltetése és megértése sokkal könnyebb adatsorok esetében, ha őket egy grafikonon ábrázoljuk. Jelen esetben a vonaldiagram jellegű grafikonok segítik legjobban az adatok megjelenítését. Elkészítésükhöz a GNUPLOT nevű programot használtam.

4. Első eredmények

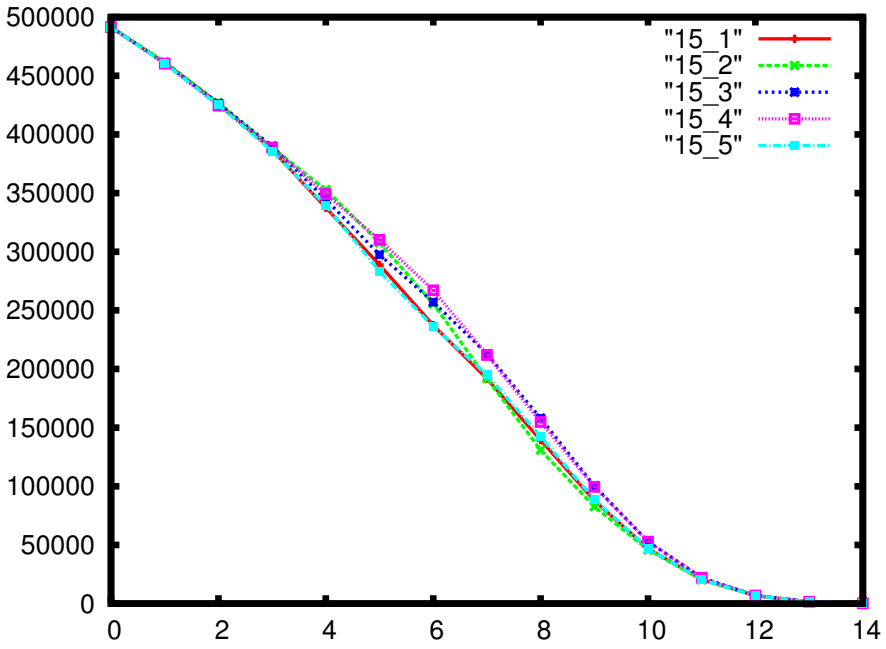
4.1. Kapott értékek A szimulációt 5-5 menetben futtattam 5,10,15,20 és 25 dimenziós kockákra. A grafikonok X tengelye az ütem sorszámát, Y tengelye az üzenetek számát jelöli. Az eredmények:



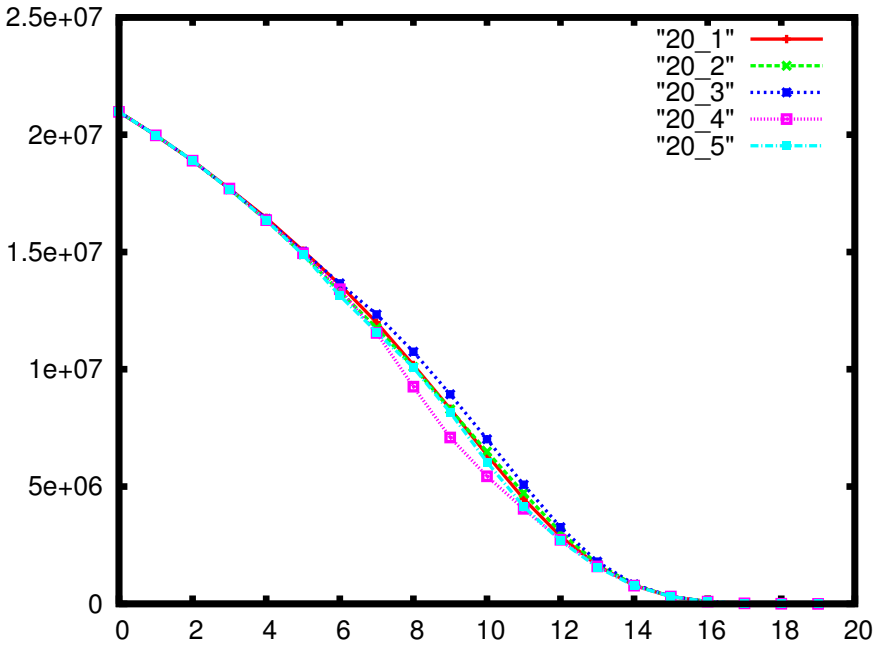
4.1. ábra. Öt dimenziós próba



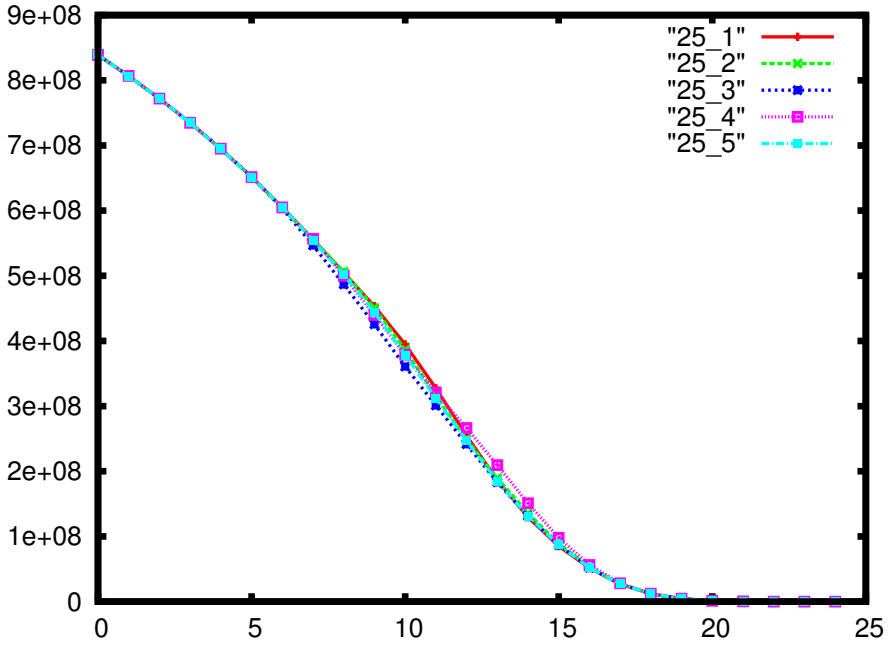
4.2. ábra. Tíz dimenziós próba



4.3. ábra. Tizenöt dimenziós próba



4.4. ábra. Húsz dimenziós próba



4.5. ábra. Huszonöt dimenziós próba

4.2. Következtetés Az egyes grafikonokról jól látható, hogy a kocka dimenziószámának növelése nem hozott lényegi változást a görbék alakjában, azaz az algoritmus ugyanúgy viselkedik függetlenül attól, hogy mekkora hálózaton futtatjuk.

A másik vizsgált kérdés, vagyis, hogy a hiperkocka csúcsainak permutálása miként változtatja meg az eredményt hasonló eredményt hozott: az egyes azonosítók helyzete nem befolyásolja számottevően az algoritmus üzenetszámát.

Kijelenthető tehát, hogy a hiperkocka a vizsgált tulajdonságok tekintetében egy stabil hálózati topológia.

5. A probléma nehezítése

5.1. Új alapötlet Mivel az eredeti probléma láthatóan nem vezetett érdekesebb eredményre, ezért folytattam a vizsgálódást. Az eddigi eredményekből kiindulva valószínűsíthető, hogy a hálózat toleráns lehet a hibákra, így a probléma nehezítéseként az eddig stabil üzenetátvitelt vonom meg a folyamatoktól. Az új probléma alapköve, hogy az üzenetváltás nem megbízható, az elküldött adat, lehet, hogy sosem érkezik meg.

Sejtésem, hogy a hiperkocka jellegéből fakadóan jól tűri a hibákat, ezért az átviteli hibát jellemző paramétert célszerű úgy megvalósítani, hogy annak növelése egyre finomodó skálán rontsa az átviteli esélyeket. Az én megoldásomban ez a paraméter azt jelzi, hogy várhatóan hány próbálkozásra sikerül elküldeni egy üzenetet.

5.2. Kipróbálandó algoritmusok Az OptMaxTerjed algoritmus jellegéből fakadóan igyekszik kizárni az üzenetekben rejlő redundanciát, ezért hibatűrése nem jó. Ebből kifolyólag az eredeti MaxTerjed algoritmussal is elvégzem a szimulációt.

A két tanult algoritmus mellet egy módosított MaxTerjed algoritmust is kipróbálok az eredmények színesítése érdekében.

5.3. Az új algoritmus Az új algoritmust a MaxTerjed algoritmus módosításaként készítem el. A különbség mindössze annyi, hogy az i. lépésben minden csúcs csak egyetlen szomszédjának küldi el az általa ismert maximális azonosítót, mégpedig annak, akinek az azonosítója csak az i. bitpozícióban tér el a sajátjától.

Belátható, hogy az algoritmus hibátlan üzenetátvitel esetén megoldja a vezetőválasztás problémáját, ám terjedelmi okokból ennek részletezésétől eltekintek.

5.4. Vizsgált értékek Az új probléma esetében célszerű megvizsgálni, hogy hány csúcs tudta meg a legnagyobb azonosítót az algoritmus időkerete alatt. Mivel az üzenetküldés sikeressége a véletlenen múlik, fontos, hogy sok menetet vizsgáljunk ugyanolyan paraméterekkel és az eredmények átlagával dolgozzunk a későbbiekben.

Vizsgálatom során különböző dimenziószámú hiperkockák, különböző hibafaktorok melletti összehasonlítását tűzöm ki célul.

6. A módosított program

6.1. Fő modosítások A nehezített probléma több változtatást is megkövetelt a programban. Ezek közül a legfontosabb a két új algoritmus és a nem megbízható kommunikáció implementálása. Ezen kívül figyelmet érdemel az új kimeneti formátum, ami jobban idomul a szimulációs jelleghez és segíti a grafikonok elkészítését. Végül a szimuláció kényelmesebb lebonyolítása érdekében a program különböző futtatásait egy shell-szkriptre bíztam.

6.2. Bemenő paraméterek A korábbiakhoz képest a szimuláció több paramétert kapott. Ezek:

(1) a szimulálandó hiperkocka dimenziója
(2) a futtatandó menetek száma
(3) a hibafaktor, melynek reciproka adja meg egy üzenet átviteli esélyét.
    Pl: 5-ös hibafaktornál egy üzenet átvitelének az esélye $\frac{1}{5}$, azaz várhatóan minden ötödik próbálkozás jár csak sikerrel.
(4) a használandó algoritmus száma
    - 0 - MaxTerjed
    - 1 - OptMaxTerjed
    - 2 - saját algoritmusom

6.3. Kimenet Mivel jó minőségű szimulációhoz rengeteg menetre van szükség, ami a képernyőn követhetetlen mennyiségű információ megjelenítését eredményezné, ezért a módosított program kizárólag kimeneti fájlokat készít.

Minden szimuláció alkalmával egy lefuttatott menet végeredménye a menet végén a tényleges vezető azonosítóját ismerő csúcsok számának aránya az összes csúcshoz képest. A szimuláció kimenete az összes menet végén kapott arányok átlaga, így nagy számú menettel minimalizálhatók az esetleges extrém végeredmények hatásai.

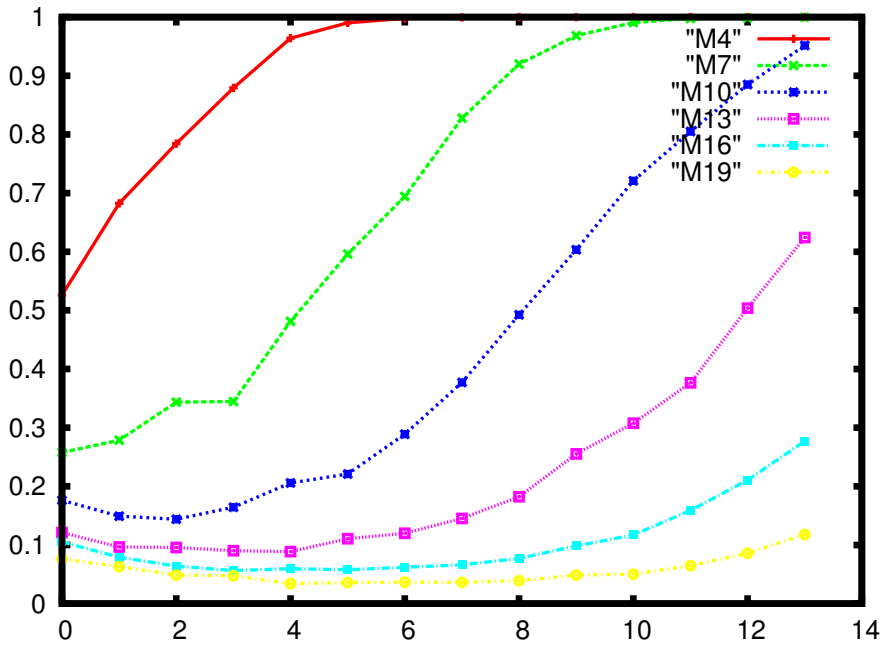
A futás végén egy kimeneti fájlt készít a programDv formájú névvel, ami a D dimenziós szimuláció végén a vezezető azonosítóját ismerő csúcsok arányát tartamazza.

6.4. Futtató szkript A kimeneti adatok kézzel történő értékelése gyakorlatilag lehetetlen, ezért szükséges ember számára is olvasható formába önteni őket. Mivel több paraméter szerinti változást is figyelemmel szeretnénk kísérni ezért sok szimulációra lesz szükség, melyek egyenként történő futtatása és az eredmények egy grafikonon történő ábrázolása rendkívül időigényes munka. Ennek megkönnyítésére készítettem egy shell-szkriptet, mely kész, grafikonon ábrázolható adatsorokat állít elő a kívánt paraméterek függvényében. A szkript bemenő paraméterei:

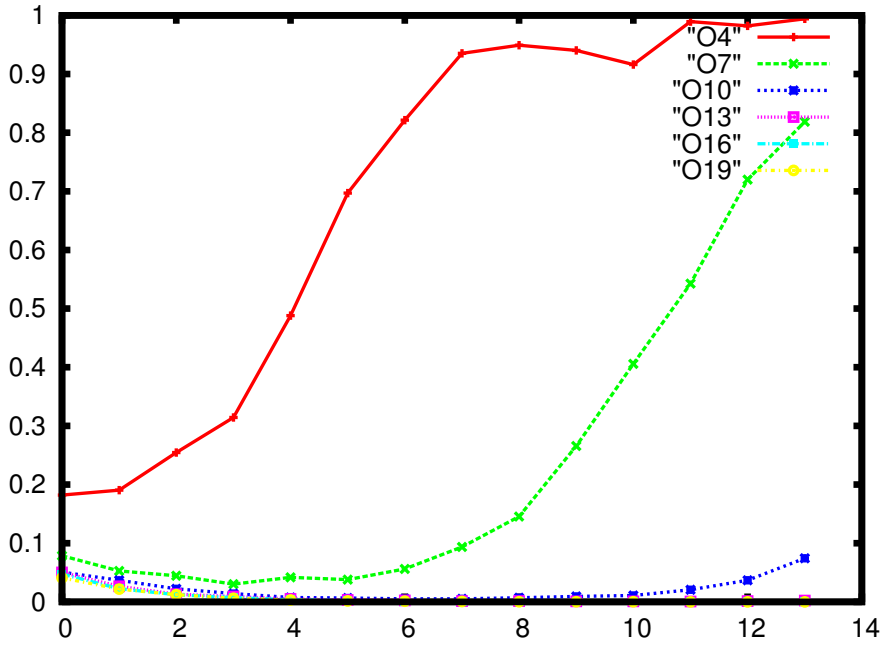
(1) menetszám: hány menetes szimulációkra lesz szükségünk
(2) kezdő dimenzió: melyik a legkisebb hiperkocka, amit látni szeretnénk a grafikonokon
(3) záró dimenzió: melyik a legnagyobb hiperkocka, amit látni szeretnénk a grafikonokon
(4) kezdő hibafaktor: melyik a legkisebb hibafaktor, amivel el akarjuk végezni a számításokat
(5) záró hibafaktor: melyik a legnagyobb hibafaktor, amivel el akarjuk végezni a számításokat
(6) lépésköz: két egymást követő szimuláció hibafaktora közti különbség
(7) algoritmus száma: a fenti számozás szerint választ algoritmust a szimulációhoz

7. Eredmények

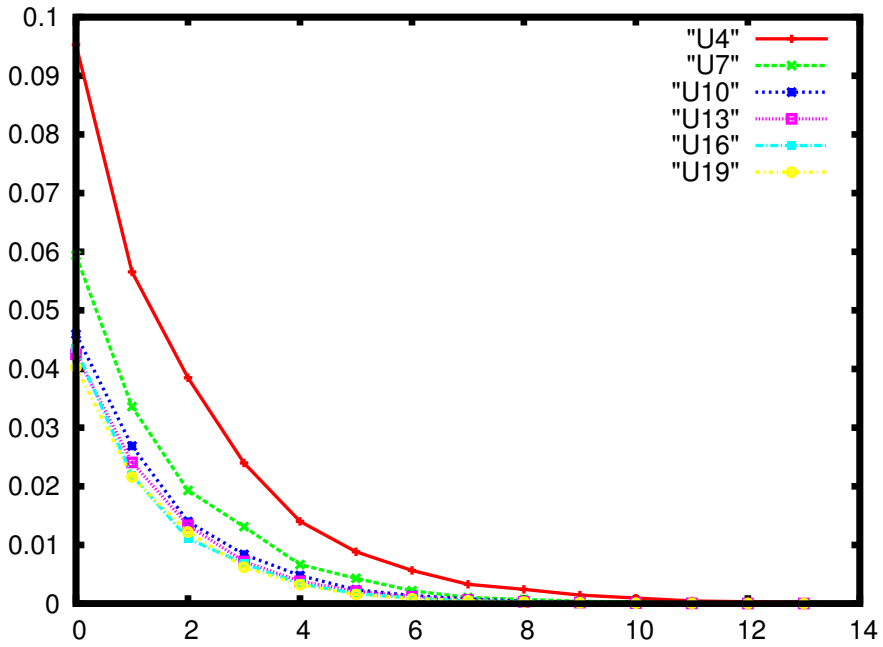
7.1. Kapott értékek A szimulációt 1-15 dimenziós kockákra végeztem el 100-100 példányban 4,7,10,13,16 és 19-es hibafaktor mellet mindhárom bejelentett algoritmusra. A grafikonok X tengelye a dimenziószámot, Y tengelye a jó információval rendelkező csúcsok arányát mutatja. Az összesített eredmények:



7.1. ábra. MaxTerjed algoritmus



7.2. ábra. OptMaxTerjed algoritmus



7.3. ábra. Saját algoritmus

7.2. Következtetés A MaxTerjed algoritmus mellett jól látható, hogy a hiperkocka dimenziószámának növekedése mellett megjelenő fokozott redundancia az információáramlásban fokozatosan kiküszöböli a hibás átvitel miatt elveszett üzenetek okozta adatvesztést.

A várakozásoknak megfelelően az OptMaxTerjed algoritmus lényegesen rosszabbul reagált a módosításokra. Ennek oka természetesen az, hogy a küldők nem tudnak arról, hogy az elküldött üzenetük célbaért e, így jóhiszeműen inaktív állapotba kerülhetnek anélkül, hogy az információ tovább terjedt volna a hálózatban. Megállapítható viszont, hogy továbbra is jelentős redundancia van az elküldött üzenetekben, hiszen látahatóan lehetséges a csúcsok nagy részének a jólinformáltsága.

Az általam adott algoritmus jó példája annak, hogy miért létszükség a nem megbízható kommunikációs folyamatokban a redundancia. Mivel minden összekapcsolt csúcspár között csak egyetlen kísérlet történik az üzenetváltásra, ezért annak sikertelensége esetén nagyban csökken annak az esélye, hogy a szimuláció végére jó eredményt kapunk az összes csúcsra.

Például, egy 2 dimenziós kocka esetén 2 útvonal van a két sarok között, melyek hossza egyenként 2. E két út és a bennük található két-két él független egymástól, így $2 \cdot \left(\frac{1}{hibafaktor}\right)^2$ esélyünk van arra, hogy a vezetővel átellenes csúcs megtudja a vezető azonosítóját.

Hosszabb utak esetén tovább csökken a célbaérés esélye, így a kocka növekedésével a korábbi eredményekkel ellentétben romlik a jól informált csúcsok aránya.

8. Lehetséges további vizsgálatok

- Adott dimenzióhoz mekkora a maximális hibafaktor, amely mellett bizonyos valószínűséggel minden csúcs helyes információt tud a szimuláció végére?
- Adott dimenzióhoz és adott hibafaktor mellett hány ütem szükséges, hogy a csúcsok mindegyike bizonyos valószínűséggel helyes értéket tudjon?
- Milyen esetekben állhat elő, hogy az OptMaxTerjed algoritmus futása közben minden csúcs inaktívvá válik mielőtt a szimuláció befejeződne.
- Miben változtatna az eredményen, ha a csúcsok értesülnének a sikertelen üzenetküldésről és újra próbálkoznának a következő ütemben?
- Mit tárna fel, ha ütemekre lebontva vizsgálnánk a problémát?

- Milyen kapcsolatban van a célbért üzenetek száma a jól informált csúcsok számával?
- Milyen görbét adna a jólinformált csúcsok aránya, ha súlyoznánk a dimenziószám és a hibafaktor hányadosával?