Budapesti Műszaki és Gazdaságtudományi Egyetem
Informatikai Tudományok Doktori Iskola

Újfajta hálózatoptimalizálási és adattömörítési eljárások

Nagy Máté Imre
okleveles villamosmérnök

Tézisfüzet

Konzulens:

Rétvári Gábor , Ph.D. (MTA doktora)

Távközlési és Médiainformatikai tanszék,
Budapesti Műszaki és Gazdaságtudományi Egyetem

Budapest, Magyarország
2022

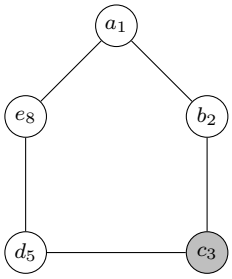
1. Bevezető

Manapság az okostelefonok elterjedésével mindenki napi szinten találkozik az olyan késleltetés-érzékeny szolgáltatásokkal mint telefónia, videómegosztás vagy instant csetelés [1]. Azonban bármely rendszer, mely a távoli végpontok közti kommunikáció kiszolgálását célozza, komoly kihívásokkal kell hogy szembenézzen. Egy jól ismert példa erre a videóközvetítés, ahol a felhasználói élményt gyakran elrontja a hirtelen leromló képminőség (limitált sávszélesség okán) vagy a rendszeresen szétesőő kapcsolat (hálózati eszközök hibái). Emiatt a hálózat teljes életciklusa alatt az operátorok két legfontosabb célkitűzése, hogy **(I) mérsékeljék a lehetséges hibák hatását** az általuk üzemeltetett infrastruktúrán, valamint hogy **(II) biztosítsák az eszközeik maximális hatásfokon történő működését**.

Tetszőleges hálózati hibára felkészülni azonban már önmagában is komplex feladat. Sajnos az Internet Protokoll (IP) egy nem megbízható, úgynevezett „best-effort" jellegű működésmódon alapul, mely igen komoly kihívások elé állítja az operátorokat ha tartani szeretnék szolgáltatásaik „ötkilences" (99.999%) elérhetőségét. Az elosztott, útválasztók által menedzselt csomagtovábbítási rendszer könnyedén inkonzisztens állapotba kerülhet, amely során a hálózati hibák hatására egymásnak ellentmondó bejegyzések kerülnek a továbbítási táblákba. Ez gyakran továbbítási hurkok kialakulásához vezethet, ahol a csomagok az eszközök interfészei között pattognak — mígnem azok összes számítási kapacitását fel nem emésztik.

Erre a jelenségre a megoldás, hogy mindenfajta forgalomkezelést szüneteltetünk egészen addig amíg az útválasztók fel nem oldják a helyzetet egy úgynevezett helyreállítási procedúra végrehajtásával [2, 3]. Amint ez befejeződött a forgalom újraindulhat — de persze csak ha túlélte egyáltalán a kiesés teljes időtartamát. Ez gyakran több száz milliszekundumig [4] is eltarthat, ami általában elfogadhatatlan méretű kiesés az érzékeny forgalom számára. Ezen probléma áthidalására vezette be a hálózati közösség az IPFRR keretrendszert [5], mely a hibákra való *előzetes felkészülés* mellett a *lokális szintű beavatkozást* preferálja. A kihívás itt olyan alternatív útvonalak megtalálásában rejlik, melyek hurokmentesek és ezáltal képesek a csomagok célbajuttatására a hálózat inkonzisztens állapota során. Az elmúlt időkben erre egy sajátos megoldás, a hurokmentes elkerülőutak módszere (Loop-Free Alternate, LFA) [6], emelkedett ki leginkább a javaslatok széles palettájából.

Tekintsük át az ötletet az 1a. ábrán. Az öt útválasztó egység súlyú élekkel van összekötve és a csomagok a lehetséges legrövidebb útvonalakat követik a végpontok között. A (c, b) link hibáját feltételezve a $c \to a$ forgalom megállni kényszerül a *c*-*b*-*a* útvonalon míg a helyreállítási procedúra be nem fejeződik, hacsak nem találunk egy alkalmas alternatív



(a) Példa topológia

cél	nh.	LFA
–	d	–
a	b	d
b	b	–
d	d	–
e	d	b

(b) Útválasztó tábla, c

1. ábra. Az LFA demonstrációja egy választott topológián.

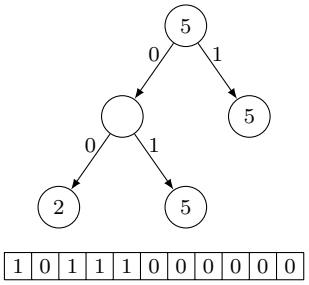
útvonalat amelyre c átirányíthatná a forgalmát. Szerencsére könnyű észre venni, hogy a c-b él hibájától függetlenül a szomszédos útválasztó, d, továbbra is képes csomagszállításra a irányába a d-e-a útvonalon. Ami a legfontosabb, hogy ez az útvonal garantáltan hurokmentes lesz mivel nem tartalmazza a csomagok forrását, c-t, ezáltal nem áll fenn a c és d közötti továbbítási pingpong veszélye. Sajnos azonban nem minden forrás-célpárhoz létezik LFA ahogyan azt a 1b. táblázat is szemlélteti. Ezért a Disszertáció első célkitűzése, hogy *különböző hálózatoptimalizálási problémák megfogalmazását követően olyan algoritmusokat mutasson be, melyek a leggyakoribb meghibásodási modelleket magukban foglalva tökéletes LFA védelmet biztosítanak a kommunikációs hálózatok számára.*

Azon törekvésünk, hogy növeljük a hálózati adatfeldolgozás hatékonyságát szintén nem könnyű feladat. Nem csak az adatok méretének növekedésében, de az azokhoz történő hozzáférési mintákban is jelentős változások történtek az elmúlt időszakban. Ezeket a változásokat nyilvánvalóan az adatábrázolási formák is követték, ami egy sor újszerű adatstruktúra kialakulásához vezetett a végső célkitűzést azonban sosem figyelmen kívül hagyva: *a lehető leggyorsabb hozzáférést biztosítani a lehető legkisebb tárolási méret árán.* Gyakorlati példa az ilyen együtthatásra a gyenge teljesítményű útválasztó, mely nagyban hozzájárul a végpontok közti késleltetés növekedéséhez valamint a jitter megjelenéséhez. Demonstrációs célból áttérünk a hálózati címek egy valósághűbb (id alapú) ábrázolási formájára, melyet a c útválasztó továbbítási táblájához vonatkozóan a 2a. ábrán ismertetünk. A prefix oszlop megmutatja hogyan lehet összevonni többféle szabályt a cél címek bináris formátumát, valamint a $*$ helyettesítő karaktert felhasználva. Ezen módszer segítségével a szabályok száma 4-re csökken (az 1b. táblázatban látható 5-höz képest), ugyanakkor továbbra is végig kell lépkedni az összes bejegyzésen, ha a keresett cél címéhez tartozó szabály a tábla utolsó sorában helyezkedik el.

Mivel ez nem túl hatékony módszer, a hálózati közösség a továbbítási tábla prefix-fa

cél	prefix	nh.	LFA
-	*	5	2
1,2	00*	2	5
5	0101	5	2
8	1*	5	2

(a) Továbbítási tábla



(b) Prefix fa

2. ábra. Útválasztó tábla tömörítésének demonstrációja.

alapú reprezentációja felé fordult, mely a 2b. ábrán látható. A fa kétségkívül egy jóval egyszerűbb és tetszetősebb rálátást nyújt a szabályokra de ugyanakkor felveti a kérdést, hogy hogyan kódoljuk binárisan? A példánkban a level-order kódolást választjuk, mely végigmegy a leveleken a legfelső rétegtől a legalsóig és 1-ket ír azon csomópontok esetén melyek értéket tartalmaznak, valamint 0-t máskülönben (az így keletkezett bitvektor a fa alatt látható). A kódolás a fa struktúrájára vonatkozik és a szimbólumokat egy tömbben tároljuk mellette. Ehhez összesen csak 11 bitre van szükségünk de *a kódolt formátumon történő lekérdezések elvégzéséhez elengedhetetlen néhány speciális művelet.* Mivel a struktúra elrendezését a bitvektorban lévő 0-k és 1-k pozíciója határozza meg, így képesnek kell lennünk az olyan lekérések támogatására minthogy „mi az i.-ik 0 pozíciója?" vagy „adjuk vissza az 1-ek számát az i.-ik pozícióig". A kutatók megállapították, hogy a bitvektor szabadon tömöríthető egészen addig míg a fenti műveletek gyorsan elvégezhetőek rajta. Mindazonáltal a méretcsökkentés és az emellett biztosítani kívánt gyors hozzáférés számos kompromisszumot igényel a megvalósítás során. Ennek következményeképp a Disszertáció utolsó célkitűzése egy *olyan újszerű adatstruktúra bemutatása, mely észrevehető sebesség növekedéssel jár a mai modern implementációkhoz képest mialatt entrópia-korlátos méretcsökkenést ér el a tárolási méretben.*

2. Kutatási Célok

A késleltetés-érzékeny adatfolyamok ultra gyors reakciót kívánnak hibák esetén. Az IGP hálózati protokollok elosztott sémája azonban meggátolja az instant hibajavítást, amiért az IPFRR lokális és proaktív elven működő keretrendszeréhez fordulunk. Az egyetlen IPFRR módszer, amely mostanáig képes volt a nagyobb gyártók figyelmének felkeltésére, a hurokmentes elkerülőutak módszere (LFA) azonban nem biztosít védelmet az összes lehetséges

hibaesetre. Másrészről, mivel az LFA választás opciója mára a legtöbb kereskedelemben is elérhető útválasztón elérhető [7, 8, 9], ezért adott a lehetőség egy olyan pusztán szoftveralapú frissítésre, mely önmagában teljes IPFRR védelmet biztosít a hálózat számára. Ehhez csak annyi a dolgunk, hogy kihasználjuk a hálózat topológiája valamint a védett pontpárok száma közti függőséget.

Első célunk, hogy a hálózathoz történő SRG-független (Shared Risk Group – közös meghibásodási csoport) kiegészítő élek hozzáadásával elérhető alternatívákat nyújtsunk az egyébként védtelen forrás-célpárok számára. Leginkább arra vagyunk kíváncsiak, *hogy a probléma egyáltalán megoldható-e tetszőleges topológiára, és ha igen, akkor hogyan találjuk meg a kiegészítő élek minimális halmazát, melyek hálózathoz adása teljes védelmet eredményez.* Emellett tudni szeretnénk, *hogy hogyan generáljuk le a kiegészítő élek felső korláttal maximált halmazát, mely a legnagyobb mértékben növeli a hálózat LFA lefedettségét.* Másodszor ugyanezt a célt tűzzük magunk elé, de ezúttal anélkül hogy hozzányúlnánk a fizikai topológiához. Kihasználjuk a virtualizációs technológiában rejlő lehetőségeket és egy védőhálót építünk a fizikai topológia fölé. Ennek megfelelően *különböző költség hozzárendelési stratégiákat azonosítunk és vetünk össze a virtuális éleken*, hogy minimális új virtuális eszköz hozzáadásával maximalizálni tudjuk a védett pontpárok számát.

A hálózati megbízhatóság mellett, az információ visszanyerési műveletek is nagyban befolyásolják az átfogó felhasználói élményt. Épp ezért az utolsó célkitűzésünk, hogy bemutassunk egy újfajta tömörítési sémát („RRR–Developed Data structure for big Data", R3D3), amely kihasználva a nagyobb blokkméret előnyeit jobb végrehajtási idővel válaszolja meg a lekérdezéseket mint az eddig ismert megoldások. Az ötlet abban rejlik, hogy feloldva a jelenlegi tömörítési módszerek legnagyobb kihívását, az indexet is tömörítjük nem csak a hasznos adatot. *Duplán opportunista adatstruktúrának* hívjuk, ami lehetővé teszi az `access`, `rank` és `select` lekérdezések $O(\log n)$ időben történő megválaszolását.

3. Kutatási Módszertan

Az alkalmazott kutatási módszertan azt a sémát követi, ahol az elméleti eredményeket analitikus módszerekkel nyerjük majd ezeket numerikus kiértékelés segítségével ellenőrizzük. Sok esetben a probléma komplexitása nemdeterminisztikus polinom idejű, amely szükségessé teszi egészértékű lineáris programok felírását az optimális megoldás eléréséhez. Mivel az LP megoldók általában gyenge futási idővel rendelkeznek, *közelítő algoritmusokra* teszünk javaslatot melyek közelítik az optimumot. A *szimulációs eszközök* halmaza, melyeket a kutatási eredmények alátámasztására fejlesztettünk, nyilvánosan elérhetőek a GitHub-on [10].

A legtöbb forrásfájl C++ programozási nyelven írodott és nagyban támaszkodik szabadon elérhető könyvtárakra úgymint LEMON [11], Gurobi [12] vagy a BOOST tesztelési keretrendszer [13].

Bemenetként mintahálózatok széles skáláját használjuk. Az egyszerűsített AS1221, AS1755, AS3257, AS3967 és AS6461 topológiák a Rocketfuel adathalmazból származnak [14]. Ezen gráfok valós szolgáltatói hálózatokat írnak le becsült élköltségekkel. Emellett használjuk az Abilene, Italy, NSF, Germany, AT&T és a bővített német gerinchálózatot (Germ_50) az SNDlib-ből [15]. Továbbá rendelkezésünkre áll a Topology-Zoo projekt [16] adathalmaza, mely esetében a hiányzó élköltségeket véletlenül generált értékekkel pótoltuk. Mindenhol eltávolítottuk a párhuzamos éleket, valamint az élköltségeket is szimmetrikussá tettük.

Az újfajta adattömörítő eljárásunk kiértékelésére (R3D3) különböző forrásokból szerzett szöveges adatokat a Pizza-Chile adathalmazból [17], az USC gén adatbázisból [18] valamint a Calgary Corpusból [19] vettük. Az IP útválasztó táblák a [20]-ból származnak. A forráskód szintén szabadon elérhető a GitHub-on [10].

4. Új Eredmények

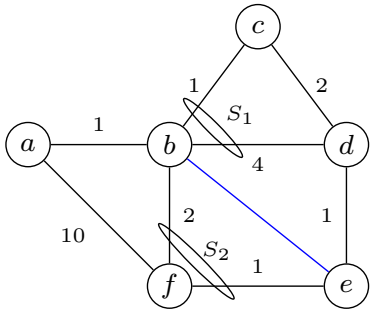
4.1. LFA Gráf-kiterjesztés Összefüggő Hibákra

1. Téziscsoport. *[J3, J4, C2, C3] Formálisan definiáltam az LFA gráf-kiterjesztési problémát az összefüggő hibák esetére (*$\text{minLFA}_{\text{SRG}}$*). Javaslatot tettem egy előfeldolgozó és számos közelítő algoritmusra, melyekről kiterjedt mérések segítségével megmutattam, hogy képesek 100%-os LFA él-védelmet elérni a vizsgált hálózatokban az eredeti élek átlagosan mindössze 30%-nak hozzáadásával. Hasonlóképp megmutattam, hogy a 100%-os pont-védő LFA védelem az eredeti élek átlagosan 60%-nak hozzáadásával érhető el.*

Ahogy a cím is sugallja az első célunk, hogy a hálózatot okosan kiegészítve *a minimális számú komplemens élhalmazzal az LFA lefedettség a legnagyobb mértékben javuljon.* A közösen meghibásodó élek esetét is figyelembe vesszük, mivel a hálózatok IP kontroll sík által érzékelt, logikai reprezentációja gyakran eltér a valós fizikai elrendezéstől.

4.1.1. Definíció

Legyen $S = \{(i,j) \in E\}$ egy SRG élhalmaz mely tartalmazza (i,j) élet. Az élek száma maximum $\deg(i) - 1$ lehet, de ahhoz hogy S egy lokális SRG legyen a következő két feltételnek kell teljesülnie: $(i,j) \in S$ és $(u,v) \in S$: $i = u$. Megjegyezzük, hogy az SRG-k a modellünkben, és általában is, aszimmetrikusak tehát $(i,j) \in S$ nem vonja magával hogy



3. ábra. Példahálózat egy LFA védettség növelése céljából hozzáadott éllel (kékkel jelölve).

$(j, i) \in S$. Így tehát minden irányított $(i, j) \in E$ élre létre lehet hozni az SRG-k unióját mely tartalmazza (i, j)-t:

$$S(i,j) = \bigcup_{S:(i,j)\in S} S \ .$$

1. Definíció. *Valamely forrás s és célpont d-re legyen e az alapértelmezett next-hop*[1]*. Ekkor s néhány t szomszédja egy* SRG-független él-védő LFA *[6] s-ből d-be ha*

i) $t \neq e$,
ii) $\mathrm{dist}(t, d) < \mathrm{dist}(t, s) + \mathrm{dist}(s, d)$, *és*
iii) $(s, t) \notin S(s, e)$.

Tekintsük át ezen irányelveket a 3. ábrán vázolt topológián. Az $a \to d$ forgalom az alapértelmezett legrövidebb, a-b-c-d, útvonalon halad és feltételezzük az (a, b) él meghibásodását. A hiba következtében a-nak megmarad egy elérhető szomszédja, f, de még kérdéses hogy ez LFA-e d irányába? A válasz igen, hiszen az f-e-d legrövidebb útvonal nem tartalmazza a forrás, a pontot. Másrészről a $b \to c$ pontpár védtelen a (b, c) él hibája esetén, mivel b elérhető szomszédai, a és f, b-n keresztül jutnak el a c pont irányába. Ekkor ha a (b, e) élet megfelelően magas élköltséggel beillesztjük a hálózatba ez megakadályozza, hogy a forgalom visszafolyjon b-be így e SRG független él-védő LFA-vá válik a $b \to c$ pontpárra.

Az él-védelemhez hasonlóan léteznek a next-hop kiesése ellen védő LFA-k is. Például az f csomópont egy él-védő LFA $a \to d$-re, mivel a legrövidebb útvonal f-ből nem keresztezi a forrást, a-t. Sőt mi több, az f csomópont maga a next-hop, b, kiesése ellen is védelmet nyújt hiszen az f-d legrövidebb útvonal sem megy keresztül b-n. Ezt a fajta feltételt *SRG-független pont-védő LFA*-nak hívjuk.

[1] A szakirodalom egy adott útvonalon soron következő útválasztót jelöli így.

2. Definíció. *Valamely forrás s, célpont d és alapértelmezett s-d next-hopra, e-re, az s pont t szomszédait SRG-független pont-védő LFA-nak hívjuk ha*

i) $t \neq e$,
ii) $\mathrm{dist}(t,d) < \mathrm{dist}(t,s) + \mathrm{dist}(s,d)$,
iii) $\mathrm{dist}(t,d) < \mathrm{dist}(t,e) + \mathrm{dist}(e,d)$, *és*
iv) $(s,t) \notin S(s,e)$.

Ami még hátravan, hogy megfelelő módon mérni tudjuk az adott hálózat LFA védelmének szintjét. Ehhez egyszerűen a védett versus az összes pontpár arányát vesszük:

$$\eta_{\mathrm{LP\,/\,NP}}(G) = \frac{\#\ (s,d) \text{ párok él/pont-védő LFA-val}}{\#\text{összes } (s,d) \text{ párok}}\ . \tag{1}$$

Az $\eta_{\mathrm{LP}}(G)$ és $\eta_{\mathrm{NP}}(G)$-vel jelöljük az él valamint a pont-védő *LFA lefedettségeket.* Ez az érték a topológia elrendezésétől, csakúgy mint az élköltségek kiosztásától illetve az SRG-k hálózatbeli sűrűségétől függ. A 3. ábrán bemutatott példánkon $\eta_{\mathrm{LP}}(G) = 0.73$ és $\eta_{\mathrm{NP}}(G) = 0.66$.

Alább definiáljuk az összefüggő hibatűrésű LFA gráf-kiterjesztési problémát. Itt a feladat egy súlyozott élköltségű gráf kiegészítése *minimális számú új éllel*, melyhez megfelelően választott élköltségek mellett az LFA lefedettség 100%-ká válik és közben a legrövidebb útvonalak sem módosulnak. Formálisan:

3. Definíció. Összefüggő hibatűrésű LFA gráf-kiterjesztési probléma ($\mathrm{minLFA_{SRG}}$): *Egy adott élsúllyal rendelkező, szimmetrikusan irányított $G(V,E)$ gráfra, az SRG-k $\mathcal{S} = \{S\}$ halmaza és l egész szám ismeretében létezik-e olyan szimmetrikus élhalmaz, $F \subseteq \overline{E}$, hogy $|F| \leq l$ valamint hozzá tartozó megfelelő élköltségek, amivel* (i) *az SRG-független él-védő LFA lefedettség $\eta_{\mathrm{LP}}(G(V, E \cup F)) = 1$ és* (ii) *a legrövidebb útvonalak $G(V,E)$-n egybeesnek a legrövidebb útvonalakkal $G(V, E \cup F)$-ben?*

A fenti definíció egyértelműen kiterjeszthető a pont-védő esetre is ha $\eta_{\mathrm{LP}}(G)$-t $\eta_{\mathrm{NP}}(G)$-vel helyettesítjük. Megjegyezzük, hogy az újonnan hozzáadott élek nem részei egyetlen létező SRG-nek sem és újakat sem alkotnak. Ez a megfontolás abból a tényből fakad, hogy az új élek feladata a hálózatban a legmagasabb szintű védelem biztosítása anélkül, hogy ezt bármely más hálózati eszköz befolyásolná. Ezt követően meghatározzuk a $\mathrm{minLFA_{SRG}}$ komplexitását.

1.1. Tézis. *A* $\mathrm{minLFA_{SRG}}$ *probléma NP-teljes.*

A tökéletes hibavédelmet biztosító élhalmaz megtalálása olykor túlságosan is ambíciózus cél lehet. Ennélfogva a probléma relaxált változatát is számításba vesszük, amit az összefüggő hibatűrésű LFA gráf-javítási problémának nevezünk el és amelynél célunk a legnagyobb javítás elérése egy korlátozott számosságú élhalmaz hozzáadásával.

4. Definíció. Összefüggő hibatűrésű LFA gráf-javítási probléma: *Egy adott élsúllyal rendelkező, szimmetrikusan irányított $G(V,E)$ gráfra, az SRG-k $\mathcal{S} = \{S\}$ halmaza és l, k egész számok ismeretében létezik-e olyan szimmetrikus élhalmaz, $F \subseteq \overline{E}$, és hozzá tartozó élköltségek hogy* (i) *legalább k forrás-célpár rendelkezzen SRG-független él-védő LFA-val $(G(V, E \cup F))$-ben és* (ii) *a legrövidebb útvonalak $G(V,E)$-n egybeesnek a legrövidebb útvonalakkal $G(V, E \cup F)$-ben?*

Az LFA gráf-javítási probléma is NP-teljes, mert különben $\text{minLFA}_{\text{SRG}}$ megoldható lenne LFA gráf-javítással a $k = |V^2| = n(n-1)$ beállítást felhasználva. Ezután, mielőtt a probléma megoldására ugranánk, megvizsgáljuk hogy egyáltalán megoldható-e az tetszőleges bemenetre.

4.1.2. Az előfeldolgozási probléma

Az LFA azon az ötleten alapszik, hogy olyan szomszédnak adjuk hiba esetén a csomagot aki nem juttatja vissza azt a forrásba. Sajnos ilyen szomszéd gyakran nem létezik. A $\text{minLFA}_{\text{SRG}}$ probléma megoldásaként komplemens éleket húzunk a védtelen pontpárok forrása és olyan pontok között, akik elterelőúttal rendelkeznek a célpont irányába. Azonban ha létezik a hálózatban egy csomópont, *melyen minden másik pontból induló legrövidebb útvonal keresztül vezet egy bizonyos célpont irányába*, akkor a $\text{minLFA}_{\text{SRG}}$ feladat megoldhatatlanná válik.

Ennek megfelelően, ahhoz hogy LFA védettséget tudjunk nyújtani s-ből d irányába, biztosítanunk kell hogy a d pont legalább kettő legrövidebb útvonalon keresztül elérhető legyen. Az algoritmus, mely minden célpontot két irányból is elérhetővé tesz, az *előfeldogozó* nevet kapta és vagy új élet illeszt a hálózatba vagy a meglévő élek költségét változtatja.

5. Definíció. LFA gráf-kiterjesztési előfeldolgozási probléma: *Egy adott egyszerű, élsúlyozott és szimmetrikusan irányított gráfhoz, $G(V,E)$, az S SRG halmazt figyelembe véve, tudunk-e olyan módosított $G'(V,E')$ gráfot és hozzá tartozó élsúlyokat találni, hogy a* $\text{minLFA}_{\text{SRG}}$ *probléma megoldható legyen G'-n? Létezik-e olyan E' minimális számú új és módosított élet tartalmazó élhalmaz, hogy a legrövidebb utak közti különbség G és G' között minimális legyen?*

Egy újfajta algoritmust mutatunk be, mely figyelemre méltó teljesítménynövekedést ér el az általunk ismert legjobb előfeldolgozási sémához képest [21]. Jelöljük a feldolgozásra váró pontokat $V_p \in V$-vel, és az éleket $E_p : \forall(i,j) \in E$ ahol $i,j \in V_p$. Ezáltal kapunk egy $G_p(V_p, E_p)$ gráfot, ami részgráfja $G(V,E)$-nek. A következő algoritmus $G_p(V_p, E_p)$-t kapja bemenetként és négy lépésben előállítja a $G'(V,E')$ gráfot:

Algorithm 1 Továbbfejlesztett előfeldolgozó algoritmus

1.) Minimális lefedő élek keresése $G_p(V_p, E_p)$-ben.
2.) V_p-ben a megmaradt pontok párosítása és köztes élek beillesztése
3.) Új él felvétele $u, v \in V_p$ közt, ahol u az utolsó előfeldolgozás nélkül maradt pont
4.) A fent behúzott új élek költségének meghatározása, úgy hogy csak $v \in V_p$-ben lévő legrövidebb utak változzanak.

1.2. Tézis. *Az 1. algoritmus polinom futási idővel rendelkezik és optimális eredményt szolgáltat.*

4.1.3. A $\text{minLFA}_{\text{SRG}}$ megoldása páros gráf modellel

1.3. Tézis. *A pont-védő* $\text{minLFA}_{\text{SRG}}$ *probléma páros-gráf alapú előállításához készítettem egy* $O(n^3)$ *futási idejű konstrukciót.*

A $\text{minLFA}_{\text{SRG}}$ probléma megoldásának első lépése egy megfelelő modell felépítése. Ahogy azt az 1.1. tézisben is említettük, a probléma visszavezethető a páros gráfokon történő minimális lefedő élhalmaz megtalálására [22]. Az ötlet tehát, hogy egy megfelelően megalkotott páros gráf modellen oldjuk meg a $\text{minLFA}_{\text{SRG}}$ problémát.

Legyen $(s_i, d_i) : i \in 1, \ldots, k$ a védtelen forrás-célpárok halmaza és legyen $\{(u_j, v_j) : j \in 1, \ldots, l\}$ a komplemens élek halmaza a gráfban, melyekből kihagyjuk a visszairányú éleket, i.e., a halmaz elemei (u_j, v_j) vagy (v_j, u_j), de nem mindkettő. Legyen $G'(A, B, F)$ egy irányítatlan páros gráf $A \cup B$ pont és F élhalmazzal, ahol minden egyes $a_i \in A$ pont egy védtelen $(s_i, d_i) : i \in 1, \ldots, k$ pontpárt, valamint $b_j \in B$ egy komplemens $(u_j, v_j) : j \in 1, \ldots, l$ élet jelöl. Valamely $a_i \in A$ és $b_j \in B$ pontokat akkor kötjük össze G'-ben, ha (u_j, v_j) vagy (v_j, u_j) éleket megfelelően nagy élköltséggel G-hez adva egy SRG-független él-védő LFA-t biztosítanak (s_i, d_i) számára. Ehhez annyit kell tennünk, hogy az 1. definícióban leírt feltételt megvizsgáljuk minden (s_i, d_i)-re az (u_j, v_j) és (v_j, u_j) élekkel kiegészített gráfban.

A $G'(A, B, F)$ páros gráf $O(n^2)$ ponttal és $O(n^4)$ éllel rendelkezik, amely így $O(n^2(n^2 \log n + nm))$ időben építhető fel hiszen minden pontpárra és minden $O(n^2)$ komplemens élre legrövidebb utat kell számolnunk. A művelet továbbá kiegészül azzal, hogy az (u_j, v_j) él G-hez

történő hozzáadása esetén a megfelelő b_j pontot, és szomszédait A-ban szintén el kell távolítani G'-ből. Mivel a G-ben lévő legrövidebb útvonalak érintetlenek maradnak, ezért a keletkezett páros-gráf modell az LFA gráf-kiterjesztési problémának egy érvényes megfelelője lesz. Az egyetlen hátralevő lépés a minimális lefedési probléma megoldása ($minSC$) G' felett.

4.1.4. Algoritmusok

A 1.1. tézisben megállapítottuk hogy a $\text{minLFA}_{\text{SRG}}$ probléma NP-teljes. Ez a fajta komplexitás lehetetlenné teszi nagy hálózatokra az ésszerű időkereteken belül történő optimális eredmény meghatározását. Ugyanakkor megfigyeljük, hogy a minimális lefedő halmaz megtalálása páros gráfokon megegyezik a minimális lefedő ponthalmaz megtalálásának problémájával hipergráfokon [23], amelyre az irodalomban számos hatékony heurisztika áll rendelkezésre.

A Lovász-Johnson-Chvatal (LJC, [24]) algoritmus a legnagyobb fokszámú $v \in B$ pontot adja a lefedéshez minden lépésben. Az SBT-re a [25] cikkben tettek javaslatot és az LJC-vel ellentétben az így kapott halmaznak nem lehet másik lefedő részhalmaza (inclusion-wise minimal).

A Reverse SBT algoritmus [23], ahogy a neve is utal rá, egy fordított SBT-t futtat ami minden iterációban a legnagyobb fokszámú pontot jelenti. Következésképp a pszeudokód ugyanaz mint az 3. algoritmusban azzal az apró különbséggel, hogy a 3. sornál $v \leftarrow \text{argmax}_{b \in B} \deg(b)$-t írunk. A Modified SBT algoritmus [23] egy kisebb optimalizációs lépést tartalmaz az SBT-vel szemben. Megjegyezzük, hogy az SBT, RSBT és MSBT algoritmusok olyan lefedő halmazokat generálnak, melyek részhalmazai nem lefedőek.

4.1.5. Numerikus Kiértékelés

1.4. Tézis. *[J3, C2, C3] Az irodalomból vett heurisztikus algoritmusokat javasoltam a legkisebb lefogó ponthalmaz megtalálására. Átfogó szimulációk eredményeként megmutattam, hogy a vizsgált kis és közepes hálózatok esetén, a tökéletes él-védő LFA lefedettség eléréséhez átlagosan 20–40% új él hozzáadása, valamint pont-védő esetben 50–70% új él hozzáadása szükséges és ez az eltérés az optimális élszámhoz képest maximum 5-15%. Megállapítottam, hogy a vizsgált algoritmusok közül a Lovász-Johnson Chvátal (LJC) és MSBT működik a leghatékonyabban.*

Arra voltunk kíváncsiak, hogy mennyi új él szükséges a teljes LFA védelem eléréséhez a

Algorithm 2 LJC

$B^c \leftarrow \emptyset$
while $A \neq \emptyset$ **do**
    $v \leftarrow \text{argmax}_{b \in B} \deg(b)$
    $B^c \leftarrow B^c \cup \{v\}$
    $A \leftarrow A \setminus \text{neigh}(v)$
    $B \leftarrow B \setminus \{v\}$
end while

Algorithm 3 SBT

$B^c \leftarrow \emptyset$
while $A \neq \emptyset$ **do**
    $v \leftarrow \text{argmin}_{b \in B} \deg(b)$
    if $\exists n \in \text{neigh}(v)$ with $\deg(n) = 1$
        $B^c \leftarrow B^c \cup \{v\}$
        $A \leftarrow A \setminus \text{neigh}(v)$
    end if
    $B \leftarrow B \setminus \{v\}$
end while

Algorithm 4 RSBT

$B^c \leftarrow \emptyset$
while $A \neq \emptyset$ **do**
    $v \leftarrow \text{argmax}_{b \in B} \deg(b)$
    if $\exists n \in \text{neigh}(v)$ with $\deg(n) = 1$
        $B^c \leftarrow B^c \cup \{v\}$
        $A \leftarrow A \setminus \text{neigh}(v)$
    end if
    $B \leftarrow B \setminus \{v\}$
end while

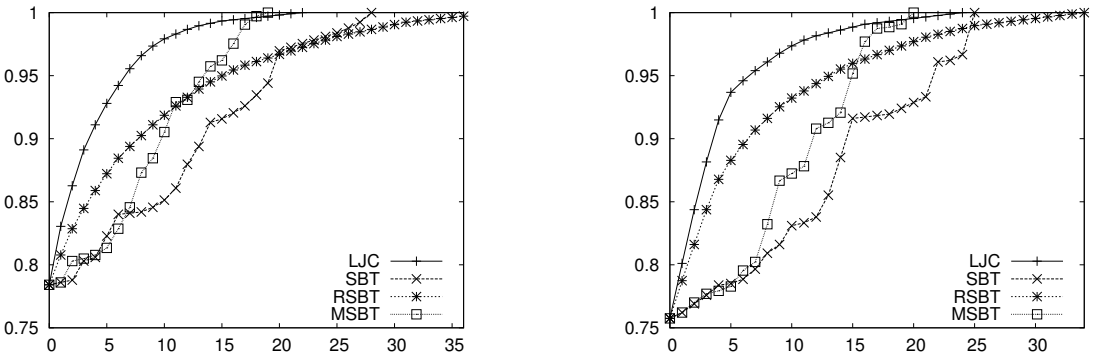
Algorithm 5 MSBT

$B^c \leftarrow \emptyset$
while $A \neq \emptyset$
    $v \leftarrow \text{argmin}_{b \in B} \deg(b)$
    if $\exists n \in \text{neigh}(v)$ with $\deg(n) = 1$
        $B^c \leftarrow B^c \cup \{v\}$
        $A \leftarrow A \setminus \text{neigh}(v)$
    else
        for each $a \in \text{neigh}(v)$ [2] with $\deg(a) = 2$
            $w \leftarrow u \in \text{neigh}(a) \setminus \{v\}$
            $B^c \leftarrow B^c \cup \{w\}$
            $A \leftarrow A \setminus \text{neigh}(w)$
        end for
    end if
    $B \leftarrow B \setminus \{v\}$
end while

4. ábra. Az LJC, SBT, RSBT és MSBT algoritmusok pszeudo-kódja a $G'(A, B, F)$ gráfon

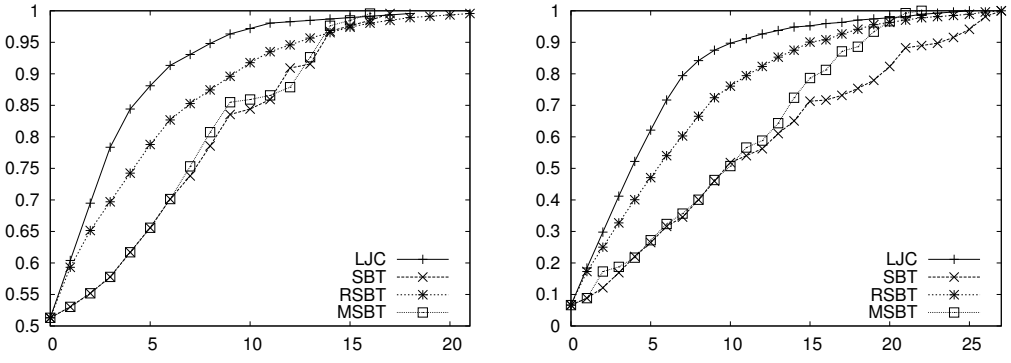
különböző algoritmusokkal mind az él- mind a pont-védő esetekben. A méréseinket különböző hibamodellek mellett végeztük. A topológiákat úgy választottuk ki, hogy a [21]-ben javasolt ILP még lefusson — legalább az SRG mentes esetben — és így össze tudjuk annak optimális eredményét hasonlítani a különböző heurisztikákkal. A mérések megkezdése előtt a továbbfejlesztett előfeldolgozó algoritmussal garantáltuk, hogy a probléma mindig megoldható legyen.

Az SRG mentes esetben minden heurisztika meglepően jól teljesít, az optimumot maximum 5-15%-kal túllépve, sőt néhány esetben azt is elérve. Az MSBT algoritmus az egyértelmű győztes mind él- és pont-védő esetben, és az SBT és LJC algoritmusok is meglehetősen jól működnek míg az RSBT nyújtja a legrosszabb eredményt.



(a) Italy (b) AS1239

5. ábra. LFA lefedettség a heurisztika egyes lépéseiben link-védő esetben az Italy és pont-védő esetben az AS1239 topológiákra.



(a) AT&T (b) Germany

6. ábra. LFA coverage in each iteration of different heuristics in the link-protecting case for AT&T with SRG density $\delta = 0.5$, and node-protecting case for Germany topology with $\delta = 0.9$

Ezután lokális-SRG-ket generáltunk egy SRG sűrűségi paraméter, $\delta \in [0,1]$, segítségével mely az összes lehetséges és kiválasztott SRG halmazok aránya. A $\delta = 0.1$ beállítással minden szomszédos élpárt 0.1 valószínűséggel adunk egy SRG-hez. Az algoritmusok hatékonysága érdemben nem változik, továbbra is az MSBT teljesít a legjobban.

4.2. Virtuális Pontok LFA Védelem Javítására

2. Téziscsoport. *Formálisan definiáltam a* RIOD *(Resilient IP Overlay Design) problémát, melynek célja egy olyan virtuális védőháló létrehozása, mely tökéletes LFA védettséget nyújt. Megmutattam, hogy a probléma NP-teljes és minden esetben megoldható. Bemutattam egy algoritmikus keretrendszert, mely paraméterezhető vagy a virtuális elemek számának vagy az algoritmus futási idejének minimalizálására. Numerikus szimulációk segítségével megmutattam, hogy a vizsgált topológiákon átlagosan fizikai útválasztónként egy virtuális pont hozzáadásával teljes LFA védettség elérhető.*

Az útválasztók virtualizációját az eszközökben elérhető erőforrások hatékonyabb felhasználására találták ki [26]. Ennek megfelelően a virtuális példányok megkülönböztethetetlenek a fizikai eszközöktől, mindegyik saját adat és kontroll síkkal rendelkezik, mely így lehetővé teszi virtuális útválasztók LFA-nak történő kijelölését egyébként védtelen pontpárok számára.

4.2.1. Definíció

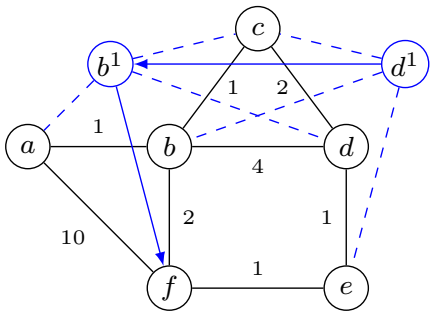
Először is kiegészítjük az LFA definícióját a virtuális rétegek esetére.

6. Definíció. *Valamely forrás s, célpont d és alapértelmezett s-d next-hopra, e-re, az n pont egy SRG-független él-védő $s \to d$* LFA *ha*

i) $n \in N_V(s)$ *és* $n \neq e$, *valamint*
ii) $\mathrm{dist}(n,d) < \mathrm{dist}(n,s) + \mathrm{dist}(s,d)$, *és*
iii) $(s,n) \notin S_{s,e}$ *(lokális SRG feltétel), és*
iv) $\mathrm{dist}(n,d) < \mathrm{dist}(n,s^i) + \mathrm{dist}(s^i,d)$ *minden* $i = 1,\ldots,k_s$.

A 7.-es ábrán mutatunk egy eshetőséget, mikor hiba hatására a forgalom LFA hurokba kerül. Azért hogy ezt a jelenséget teljesen kiküszöböljük, szükségünk van a *iv)* feltételre mely megtiltja a „kaszkád LFA-kat" a virtuális rétegben.

1. Javaslat. *A modellünkben egy csomag maximum csak egyszer irányítható LFA-hoz a forrás és célpont közti útvonalon.*



7. ábra. Mintahálózaton a lehetséges LFA hurok (a legrövidebb utat nyilak jelölik a virtuális rétegben): tegyük fel, hogy b küld csomagot f számára. Ilyenkor, a (b, f) él meghibásodása esetén a b csomópont átirányíthatja forgalmát a d^1 LFA irányába. Mindazonáltal, a forgalom soha nem érkezik meg f-hez, hiszen a $d^1 \to f$ elkerülő útvonal a $b{-}d^1{-}b^1{-}c{-}b$ LFA hurokba vezet. A probléma forrása, hogy b^1 is átkapcsol a saját LFA-jára, c-re, mikor észreveszi hogy az általa ismert útvonal f-be eltűnt.

Az iménti jelölésekkel immár megfogalmazhatjuk a *Resilient IP Overlay Design* (RIOD) problémát. A feladat egy olyan virtuális védőháló létrehozása, mely adott számú virtuális útválasztó segítségével maximalizálja az LFA lefedettséget.

7. Definíció. RIOD$(G_S, c, U, k, \eta_{\min})$*: adott* $G_S = (V_S, E_S)$ *gráf,* c *élköltségek,* $U \subseteq V_S$ *ponthalmaz és* k *pozitív egész szám esetén alkossunk olyan* $G_V = (V_V, E_V)$ *gráfot és* c_V *élköltségeket hogy:*

- $V_S \subseteq V_V$ *és virtuális pontokat csak* U *felett hozunk létre,*
- $E_S \subseteq E_V$ *és virtuális éleket csak a fizikailag is összekapcsolt útválasztók közé húzunk,*
- *a legrövidebb utak* V_S *pontpárjai közt nem változnak* (a fizikai réteg változatlan)*,*
- $|V_V \setminus V_S| \leq k$ (nem több mint k virtuális példány)*, és*
- $\eta(G_V, c_V) \geq \eta_{\min}$ (az LFA lefedettség legalább $\eta_{\min}$)*.*

4.2.2. A RIOD probléma megoldhatósága

Elsőként megmutatjuk hogy a teljes LFA lefedettség mindig elérhető:

2.1. Tézis. *[J2, C1] Kétszeresen összefüggő* G_S *gráfokra mindig létezik* G_V *és hozzá tartozó költségkiosztás,* c_V*, hogy a* RIOD(G_S, c, V_S, ∞) *probléma megoldható,* $\eta(G_V, c_V) = 1$*.*

Másrészről azt találjuk, hogy a komplexitás kezelhetetlen:

2.2. Tézis. *[J2, C1] A* $RIOD(G_S, c, U, k, \eta_{min})$ *probléma NP-teljes.*

Sajnos a mai, hatalmas méretű IP gerinchálózatoknak köszönhetően ez a probléma gyakran megoldhatatlan lineáris programokat eredményez. Következésképp, egy olyan heurisztikus algoritmus megalkotásával folytatjuk, mely egy egyszerű konfigurációs paraméter beállításával *vagy az optimális eredményt próbálja visszaadni, vagy garantáltan polinomiális időben szolgáltat egy közelítő eredményt.*

4.2.3. Heurisztikus Algoritmusok a RIOD Problémára

A heurisztikánk fő ötlete, hogy iteratív módon új virtuális pontokat adunk a hálózathoz egészen addig mígnem teljes LFA lefedettséget érünk el. Sajnos amikor csak egyetlen pontot adunk a hálózathoz minden lépésben, az algoritmus bizonyos esetekben elakadhat.

1. Tétel. *Léteznek olyan esetek, mikor az LFA lefedettség nem növelhető pusztán egyetlen virtuális pont hozzáadásával.*

2.3. Tézis. *[J2, C1] Definiáltam egy algoritmikus keretrendszert, mely tetszőlegesen választott k virtuális elemszámhoz, lehetővé teszi a* RIOD *probléma megoldását annak komplexitását „elcserélve" az elért LFA védettség mértékére. Mutattam hozzá egy konstrukciót, mely maximum $O(n^5)$ lépésben végez.*

Egyre növekvő számú virtuális pont hozzáadását javasoljuk, melyet *összefüggő l-halmaznak hívunk.*

8. Definíció. *Hívjunk egy G_S gráfhoz tartozó $U_l \in V_S$ ponthalmazt* összefüggő l-halmaznak *ha az így keletkezett U_l részgráfja G_S-nek összefüggő és $|U_l| = l$. Az U_l feletti virtuális pontokat U'_l-el jelöljük.*

Ezután definiáljuk a GLFAVirt(G_S, c_S, U_l, j) problémát, ahol a feladat a virtuális élek költségeinek megfelelő beállítása úgy, hogy az LFA lefedettség maximális legyen.

9. Definíció. GLFAVirt(G_S, c_S, U_l, j)*: Adott fizikai $G_S(V_S, E_S)$ gráfhoz, c_S élköltséghez, j pozitív egész számhoz és $U_l \subseteq V_S$ összefüggő l-halmazhoz ($l \geq 1$) létrehozott $G_V(V_V, E_V)$ virtuális topológián, ahol $V_V = V_S \cup U'_l$, $E_V \subseteq E_S \cup \{(v,u) : v \in U'_l, u \in N_S(v)\}$, létezik-e olyan c_V költség beállítás, hogy (i) az élköltségek és a legrövidebb utak G_S-ben nem változnak és (ii) # protected (s,d) párok $\geq j$, $s,d \in V_S \times V_S$?*

A heurisztikánk alapja, hogy egészen addíg míg nem növekszik az LFA lefedettség, egyre nagyobb méretű összefüggő halmazokból álló virtuális pontokat próbálgatunk a hálózatba.

Algorithm 6 Mohó algoritmus a RIOD$(G_S, c, U, \infty, \eta_{min})$ problémára

while $\eta_{min} > \eta(G_S, c)$ **do**
  for each $l = 1, \ldots, k$ **do**
    for each összefüggő l-halmaz $U_l \subseteq U$ **do**
      $(c_{U_l}, \eta_{U_l}) \leftarrow$ GLFAVirt(G_S, c, U_l) eredménye
    end for
    $(U_l', \eta') \leftarrow$ válasszuk $U_l \in U$ hogy maximalizálja η_{U_l}
    if $\eta' > \eta$ **then** hozzáadjuk U_l'-t G_V-hez és a költségeket beállítjuk $c_{U_l'}$-re
      break
    end if
  end for
end while

Ahhoz hogy csökkentsük az 6. algoritmus futási idejét, bevezetjük a *legrövidebb útvonal szeletek* fogalmát.

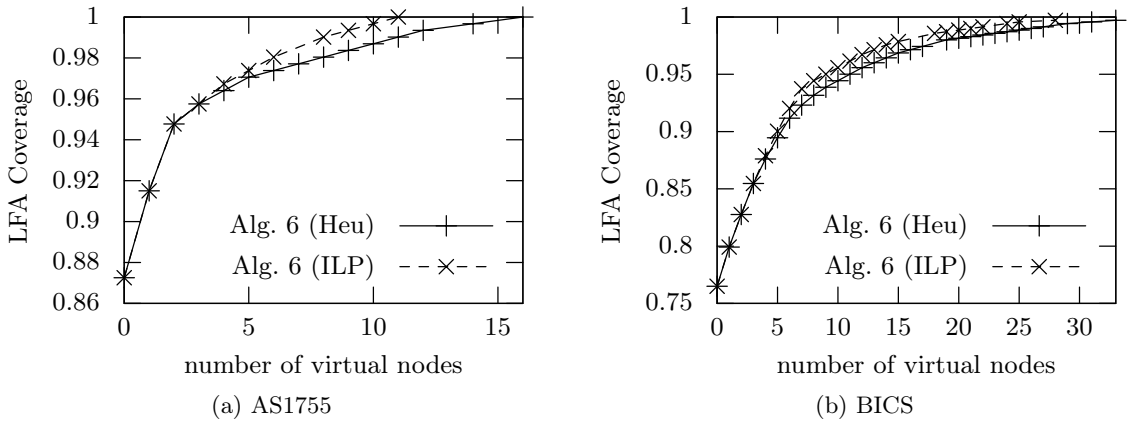
10. Definíció. *A G_S gráfon hívjuk az $U_l \subseteq V_S$ ponthalmazt az l* rangú legrövidebb útvonal szeletnek*, ha az U_l által kijelölt ponthalmaz része valamely G_S-ben futó legrövidebb útvonalnak és $|U_l| = l$.*

Emellett felgyorsítjuk az LFA lefedettség számolását azzal, hogy csak azon ponthalmazra végezzük el a számolást mely LFA védelmet nyerhet egy bizonyos virtuális pont létrehozásával. Ehhez nyomon követjük a *választható pontpárok* halmazát ($\mathcal{L}$) melynek elemei LFA-t nyerhetnek. Nyilvánvaló, hogy az u' virtuális pont csak akkor tud LFA-t szolgáltatni bármely forrásnak ha azzal szomszédos. Legyen $\mathcal{L}_{U_l} \subseteq \mathcal{L}$ azon választható pontpárok halmaza, melyek forrása szomszédos U_l-el, formálisan $\mathcal{L}_{U_l} \subseteq \mathcal{L} | (s, d) \in \mathcal{L}, s \in \text{neigh}(U_l)$. Más szóval az új virtuális U_l pontok $\mathcal{L}_{U_l}$ számára képesek LFA-t szolgáltatni, ezért $\frac{|\mathcal{L}_{U_l}|}{n(n-1)}$ a felső korlát $\eta(G)$ növekedésére ha U_l-t hozzáadjuk a hálózathoz.

Ennek megfelelően a 6. algoritmus elméleti komplexitása $O(n^5)$ a legrosszabb esetben, azonban gyakorlati tapasztalataink azt mutatják, hogy az összes pontpár közti legrövidebb útvonal megtalálása dominálja a futási időt, ami így inkább az $O(n^2)$ lépésszámhoz közelít.

4.2.4. Numerikus Kiértékelés

2.4. Tézis. *[J2, C1] Mérési eredményeimmel megmutattam, hogy a vizsgált topológiák esetén a javasolt ILP mindössze a fizikai pontok körülbelül 30%-nak hozzáadásával 95%-os LFA védettséget ér el. Ugyanakkor a tökéletes LFA védettség eléréséhez a fizikai pontok 70%-nak megfelelő virtuális pont hozzáadása szükséges. Másrészről, mikor a cél a futási idő csökkentése, megállapítottam hogy a vonatkozó 95%-os védettség eléréséhez a heurisztika átlagosan 36%-nyi virtuális pontot használ. Hasonlóképp a heurisztika a fizikai pontoknak körülbelül*



(a) AS1755 (b) BICS

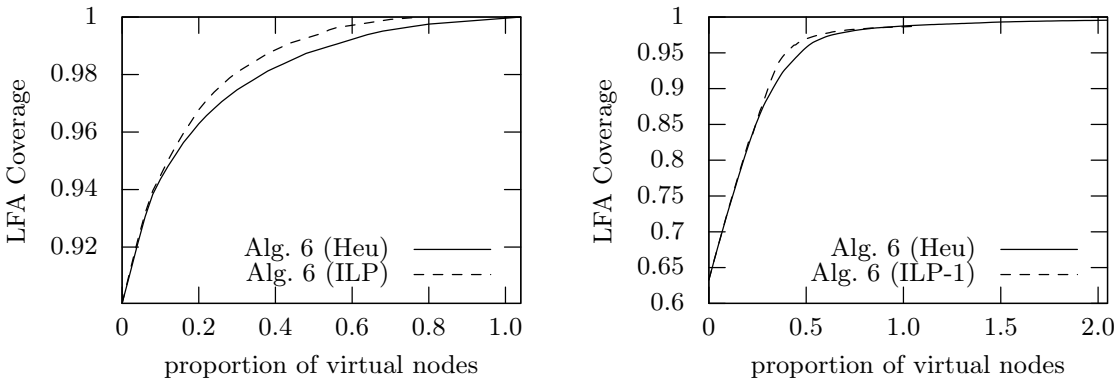
8. ábra. LFA lefedettség javulása kis és középméretű hálózatokon.

85%-át hozza létre a teljes védettség eléréséhez, ami 15%-os közelítési hibát jelent. Megmutattam, hogy a védőutak hossza átlagosan 30%-al hosszabb az eredeti legrövidebb útvonalakhoz képest.

Bemutattunk egy egészértékű lineáris programot (ILP) [Disszertáció, 3.3.4 bekezdés], amely összehasonlítási alapként szolgál a heurisztika kiértékelése során. Érdeklődésünk középpontjában a heurisztika és ILP által szolgáltatott eredmények közelsége áll, melyet az LFA lefedettség metrikával mérünk. Reményeink szerint az eltérés nem bántóan nagy és ezért a különbségért cserébe a futási idő bőven kompenzálni fog.

A heurisztika átlagosan 19%-kal lépi túl az ILP által hozzáadott virtuális pontok számát, azonban tízszer gyorsabban szolgáltat eredményt a vizsgált hálózatokra. Az LFA lefedettség javulása logaritmikus trendet mutat, tehát ha a cél pusztán egy bizonyos védelmi szint elérése akkor általában néhány virtuális pont elégnek bizonyul. Ezzel szemben a teljes védelem eléréséhez elterelő alagutakat kell létrehoznunk minden forrás-célpárra, ami jelentősen növeli a virtuális réteg méretét.

A logaritmikus javítási trend tisztán látható a 8-9-es ábrákon. Az első fázisban meredeken emelkedik az LFA védelem mértéke és a teljesítménykülönbség a heurisztika és ILP közt minimális. Szintén megfigyeljük, hogy az algoritmus a lépések nagyrészében egyetlen virtuális pontot ad a hálózathoz, azonban vannak esetek (lásd 8b. ábra), ahol mindkét metódus csak egy alagút kihúzásával jut túl egy kritikus ponton. Emellett mutatunk egy topológiát (Deltacom), ahol az ILP a $k = 3$ beállítással nem boldogul ésszerű időn belül, ami alátámasztja a heurisztikák szükségességét. Ebben a speciális esetben az ILP-re $k = 1$, a heurisztikára $k = 3$ beállítást alkalmazva kiderül, hogy a futás első fázisában 2-3%-os eltéres



(a) Germany_50 (b) Deltacom

9. ábra. LFA lefedettség javulása gerinchálózatokon.

van a két módszer között, majd ezután az ILP beragad, míg a heurisztika képes tovább javítani a hálózatot, lásd 9b. ábra. További eredmények megtalálhatóak a Disszertáció 3.3.5-ös mellékletében.

4.3. R3D3: Duplán Opportunista Adatstruktúra

Amellett, hogy a hálózatokat még inkább hibatűrővé tegyük, az őket működtető eszközök hatékonyságának növelése is céljaink közt szerepel. Hogy elérjük ezt a célt, egy újfajta tömör adatstruktúrára teszünk javaslatot, mely lehetővé teszi a gyors műveletvégzést közvetlenül a tömörített formátumon.

3. Téziscsoport. *Javasoltam egy új adatstruktúrát, mely ötvözi az RRR tárolási sémáját az Elias-Fano kódolással. Analitikus módszerekkel megmutattam, hogy a struktúra mind az adat mind az index méretében eléri az entrópia korlátot, valamint megadtam az* `access`*,* `rank` *és* `select` *műveletek komplexitását. A numerikus kiértékelés eredménye azt mutatja, hogy az adatstruktúra maximum* 25%*-al gyorsabb műveleteket eredményez, kismértékű (*15% *körüli) méretnövekedés árán.*

4.3.1. Definíciók

Tegyük fel, hogy n lehetséges esemény $p_1, p_2, \ldots, p_n$ előfordulási valószínűséggel következik be. Csak a valószínűségekkel vagyunk tisztában de semmit nem tudunk egy adott időpillanatban arról, hogy épp melyik esemény fog bekövetkezni. A kérdés, hogy mennyire vagyunk

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
0	0	0	0	1	0	1	0	0	0	0	0	1	0	0	0

10. ábra. Egy példa bitvektor.

bizonytalanok egy esemény bekövetkezésével kapcsolatban és hogy ezt hogyan tudnánk mérni? A választ *entrópiának* nevezzük és H-val jelöljük:

$$H = -\sum_{i=1}^{n} p_i \log(p_i) \tag{2}$$

Vegyük észre, hogy kétfajta lehetséges eseménynél (pl. egy bit 1 vagy 0) a 2. egyenlet így alakul:

$$H_0 = p \log(\frac{1}{p}) + (1-p) \log(\frac{1}{1-p}) \tag{3}$$

Egy adatstruktúrát *tömörnek* (succinct) nevezünk ha az képes az optimális, plusz egy elhanyagolható kis extra helyen tárolni az adatokat: $opt + o(opt)$. Az n hosszúságú, t bitmap tömör kódolása legrosszabb esetben $n + o(n)$ bitet foglal, ugyanakkor lehetővé teszi a `rank` és `select` műveletek elvégzését $O(1)$ időben. De miért fontosak ezek a műveletek egyáltalán?

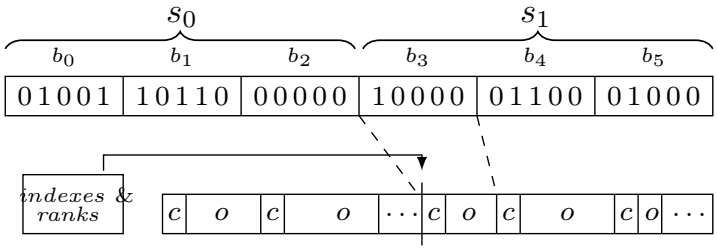
Jacobson [27] mutatott egy új kódolási sémát gráfok és fák tömörítésére, amely nemcsak minimális helyet foglal hanem lehetővé teszi azok bejárását konstans időben. Az ötlet szerint a fát egy bitvektorba lehet kódolni, ahol a lefele mozgást (gyerek) a bitvektoron elvégzett `rank` művelet, a felfele mozgást (szülő) pedig ugyanígy a `select` teszi lehetővé. Ezeket a műveleteket a t bitvektoron a következőképp definiáljuk:

- $\texttt{rank}_q(t, i)$: visszaadja a q szimbólum előfordulásainak számát i pozícióig, $t[1, i]$;
- $\texttt{select}_q(t, i)$: visszaadja a q szimbólum pozícióját a t bitvektoron.

Az 10. ábrán, $\texttt{rank}_1(t, 8) = 2$ ami megadja az 1-re állított bitek számát a 8. pozícióig és $\texttt{select}_1(t, 2) = 7$ ami megadja hogy a második 1-re állított bit a 7. pozícióban található.

RRR

Raman, Raman és Rao építve Brodnik, Munro [28] és Pagh [29] eredményeire kifejlesztett egy teljesen indexelhető szótárat (fully indexable dictionary, FID), ami kombinálja a hashing gyorsaságát a rendezett tömbök sokoldalúságával [30]. Ezt a tömörített bitvektort a szerzők után *RRR*-nek hívjuk, ami az n hosszúságú t bitvektort nH_0 biten képes tárolni hozzáadva



11. ábra. Az *RRR* kódolási séma vázlata.

az indexhez szükséges $O(\frac{n \log\log n}{\log n}) = o(n)$ bitet. Az *RRR* implementálja az `access`, `rank` és `select` műveletek konstans idejű lekérdezését.

11. Definíció. *Egy adott n hosszúságú t bitvektorra az* 1*-re állított bitek számát, i.e.* $\texttt{rank}_1(n)$*, populációs számnak hívjuk és* $\texttt{popcount}(t)$*-vel jelöljük.*

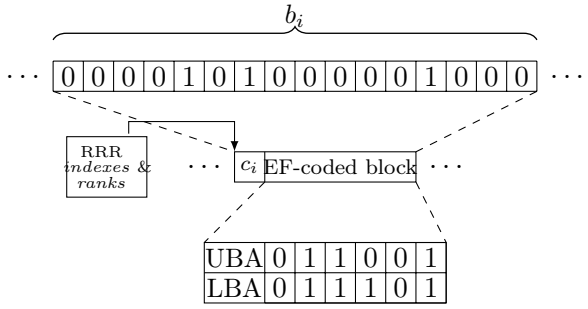
A struktúra $b_1, b_2, \ldots$ blokkokra osztja a bemenetként megadott bitvektort, ahol $b = \frac{\log n}{2}$ bit méretű (lásd az 11. ábrát illusztrációért). Minden egyes blokkot egy *class-offset* párossal kódolunk ahol $c_i = \texttt{popcount}(b_i)$, az adott blokkban található egyesek száma, és az offset pedig egy előregyártott tábla megfelelő sorára mutató index.

Claude és Navarro optimalizálási javaslatot tettek az RRR-re [31] és [32] cikkeikben, ahol jelentős méretcsökkenést értek el azzal hogy az RRR univerzális blokk kódoló tábláját lecserélték egy „on-the-fly" működő kombinatorikus dekódolóra [33]. Ezzel sikerült megszabadulniuk a relatív blokk indexektől és `rank` számlálóktól, viszont a blokkok megtalálása lineáris kereséssé degradálódott.

Az Elias–Fano kódolási séma

Az *Elias–Fano kódolási séma* a t bitvektort $nH_0 + O(n)$ méretű helyen tárolja és a $\texttt{select}_1$ lekérdezést $O(1)$ időben végzi, ugyanakkor nem támogatja a hasonló teljesítményű `rank` és `access` lekérdezéseket. Létezik olyan alternatív *EF* séma is [30, 34, 35], amely $nH_0 + O(m)$ biten kódol és $O(m)$ időben válaszolja meg az `access`, `rank`, és `select` kéréseket, ahol $m = \texttt{popcount}(t)$.

Az EF séma a t bitvektor helyett az egyesek pozíciójának karakterisztika vektorát kódolja, $\{x_1, x_2, \ldots, x_m\}$ ahol $m = \texttt{popcount}(t)$ és $x_i = \texttt{select}_1(t, i) : i \in \{1, \ldots, m\}$. Ehhez az MSB bucketing technikát használja, ahol minden egyes x_i pozíciót két részre bont, egyrészt az alacsonyabb rendű bitek sorfolytonos módon az úgynevezett Lower-bits Array, *LBA*-ba kerülnek, másrészt a magasabb rendű bitek pedig egységkódolással (unary encoding) az Upper-bits Array, *UBA*-ban kapnak helyet. Az UBA egységkódolása a következőképp fest: annyi darab 1-est írunk ahány bucket szükséges x_i ábrázolásához, majd ezt egy 0-val zárjuk.



12. ábra. Egy 16 bites blokk R3D3 kódolása és a hozzátartozó EF blokk-kód.

4.4. R3D3

Az R3D3 kombinálja az RRR [32] tárolási sémát az Elias–Fano kódolással. Az ötlet, hogy támaszkodunk az RRR rendkívül gyors hozzáférést biztosító indexelésére de az időigényes kombinatorikus blokk dekódolást lecseréljük az EF hatékonyabb sémájára.

Az R3D3 séma *duplikált indexelést* alkalmaz; először az RRR indexek alapján megkeresi a blokk kezdetét majd onnan az UBA segítségével indexeli az LBA-t és végezetül csak a szükséges LBA értékeket dekódolja.

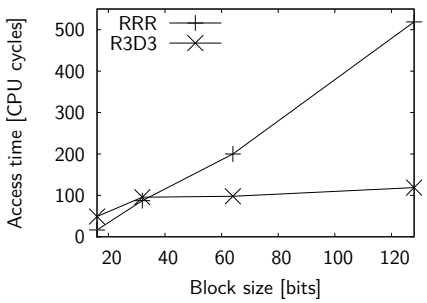
3.1. Tézis. *[J1] Megmutattam hogy az R3D3 egy tetszőleges, n hosszúságú, t bitvektort tetszőleges b blokkméretet választva*

$$nH_0 + np + \frac{n}{b}(2 + \log b) \quad \textit{biten kódolja.} \tag{4}$$

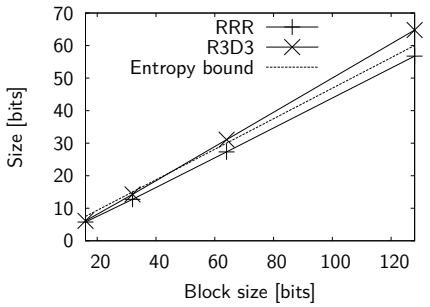
.

Az R3D3 végrehajtási ideje a következőképp alakul: az `access`(t, i) komplexitását a lineáris keresés dominálja, amely segít eljutni a megfelelő EF-kódolt blokk elejéig ($O(\log n)$ az RRR-hez hasonlóan), majd következik a blokk-dekódolás ami további $O(pb)$ lépést igényel. Hogy az RRR sebességével paritásban legyünk, választhatunk sokkal nagyobb blokkméretet is, hiszen esetünkben a blokk dekódolás sebessége $pb = O(\log n)$ és $p < 1$, míg az RRR esetében $b = O(\log n)$. A nagyobb blokkok kevesebb indexet eredményeznek, amely jelentős helymegtakarítást jelent. Ugyanez vonatkozik a `rank`(t, i)-re is. Ami a `select`(t, i)-t illeti, első körben az eltárolt szuperblokk és blokk rank értékeken végzett bináris kereséssel eljutunk a megfelelő blokkhoz ($O(\log n)$), majd ismét csak $O(pb)$ lépésben dekódoljuk azt. A lekérdezések teljes végrehajtási ideje $O(\log n) + O(pb) = O(\log n)$ ha $pb = O(\log n)$.

3.2. Tézis. *[J1] Javaslatot tettem egy blokkméret beállításra, $pb = O(\log n)$, mellyel az R3D3*



13. ábra. Átlagos access elérési idők random pozíciókra az RRR és EF blokk-kódolók esetén, a blokk-méret növelésének függvényében ($p = 0.1$).



14. ábra. RRR és EF kódolt blokkméretek, valamint az entrópia limit összehasonlítása random bitvektoron a blokkméret függvényében ($p = 0.1$).

képes az RRR indexének tömörítésére és az n hosszúságú t bitvektort

$$nH_0 + nH_0\left(\frac{1}{2} + O\left(\frac{\log\log n}{\log n}\right)\right) \tag{5}$$

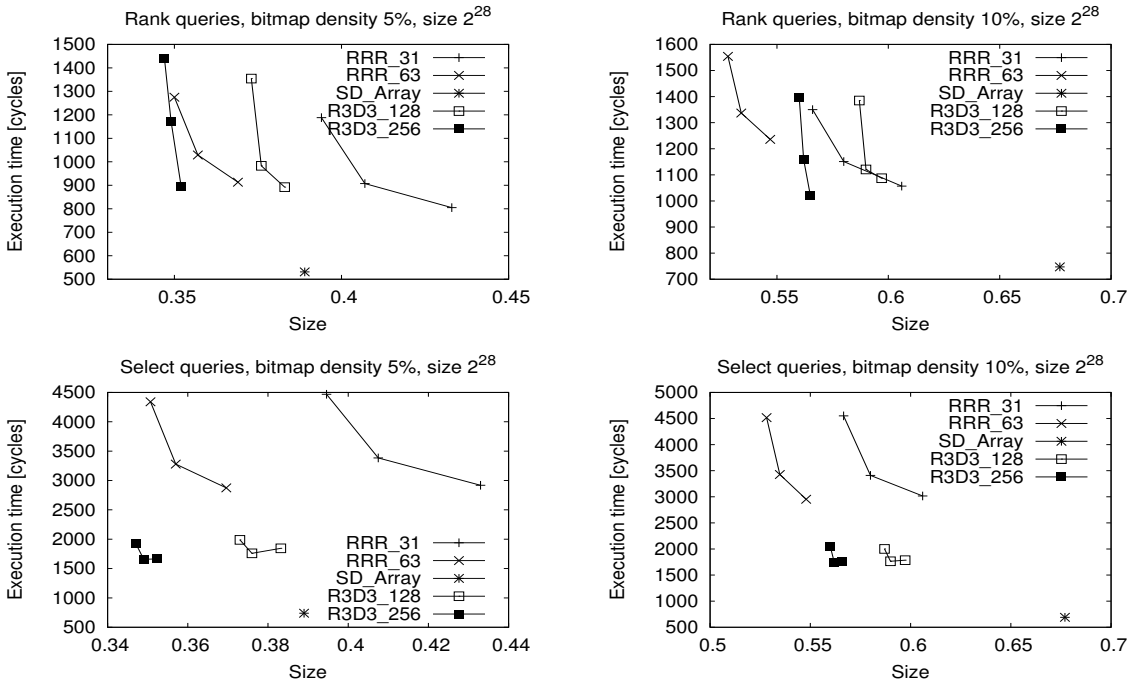
biten ábrázolja. Megmutattam, hogy az `access`*,* `rank` *és* `select` *műveletek $O(\log n)$ időben elvégezhetők.*

4.5. Kísérleti Eredmények

3.3. Tézis. *[J1]*

Kiterjedt numerikus kiértékelést végeztem szintetikus és valós adatok széles spektrumán, hogy összehasonlíthassam az R3D3 teljesítményét az általam ismert legjobb (succinct) tömörítési eljárásokkal. Jelentős különbséget találtam az R3D3 javára a blokk-dekódoló teljesítményében. A szintetikus adatokon végzett mérések $2-3\%$-os méretcsökkenést mutattak az RRR-hez képest, ami a `rank` *sebességét számottevően nem, azonban a* `select`*-ét háromszorosára gyorsítja. A nagyobb sűrűséggel rendelkező komplex adatstruktúrákon az R3D3 25%-os teljesítménynövekedést mutat, azonban ezt $10-15\%$-os (néhol 40%) méretnövekedés árán éri el.*

Blokk-kódolás. Az első kísérlet célja az EF blokk-kódoló összehasonlítása az RRR kombinatorikus dekódolójával szemben. Megfigyelhetjük az 13. ábrán, hogy az EF blokk-kódoló kevésbé érzékeny a választott blokk-méretre, míg az R3D3-nak csak háromszor több idő szükséges egy 128 bites blokk dekódolásához, addig az RRR-nél ez huszonötszörös szorzó. Mindezt az R3D3 némileg nagyobb blokkmérettel éri el (14. ábra).



15. ábra. Átlagos méret–futási idő összehasonlítás RRR és R3D3 random rank és select lekérések esetén, 2^{28} bit méretű bitvektorokon.

Random szintetikus bitvektorok. Ehhez a kísérlethez ismét random bitvektort használunk, de most teljes egészében vizsgáljuk az RRR-t és R3D3-at, csakúgy mint Navarro és társai [32]. Lemérjük a `rank` és `select` lekérések átlagos végrehajtási idejét és a teljes tömörített méret függvényében ábrázoljuk azt. Megfigyeléseink szerint alacsony kitöltöttség mellett az R3D3_256 nagyjából 2-3%-al kevesebb helyet foglal mint az RRR és hasonló teljesítményt ér el a `rank` lekérdezéseknél, a `select` esetében viszont háromszoros az R3D3 gyorsasága. Az SD_Array-hez képest 6-8%-os méretcsökkenést ér el. Nagyobb, 10% körüli kitöltöttségnél, szintén 15-20%-kal jobb eredményeket érünk az RRR-el összevetésben, efölötti kitöltöttség esetén azonban az R3D3 teljesítménye leromlik.

Internet továbbítási táblák. Méréseket végeztünk továbbítási táblákkal is, amely egy viszonylag új alkalmazási területe a komplex adatstruktúráknak [36]. A táblák egy valós operátori hálózatból valók és a 1. táblázatban mutatjuk a hozzájuk kapcsolódó eredményeket. Ismét egy szerény méretnövekedést (néhol 40%) láthatunk az R3D3-mal ami viszont majdnem kétszer gyorsabb kereséseket eredményez mint az RRR. Tapasztalataink szerint az R3D3 előnye már 128 és 256 bites blokkméretek esetén is megmutatkozik.

1. táblázat. RRR és R3D3 által kódolt továbbítási táblák: tábla neve, prefixek száma, entrópia korlát [36] szerint; tömörített méret és random lekérdezések átlagos végrehajtási ideje. Méretek Kbyte-ban (KiB) az idő CPU órajelben.

			RRR_63		R3D3_128		R3D3_256	
Név	#Prefix	Entrópia	Méret	Keresés	Méret	Keresés	Méret	Keresés
`hbone-szeged`	453,685	70.1	63.1	21340	86.9	12670	93	11800
`access_d`	403,245	149.1	83.5	18670	115.4	11000	124	10900
`access_v`	2,970	1.08	4.6	20960	6.2	14800	6.5	11540
`mobile`	4,391	1.32	3.4	15400	3.7	10960	3.8	11580
`hbone-vh1`	453,741	222.6	160.8	20340	222	13850	238	13690

5. Az Új Eredmények Alkalmazhatósága

A Disszertációban elméleti és kísérleti eredményeket mutattunk be a hálózati megbízhatóság és adattömörítési területekhez kapcsolódóan. A Disszertáció első felében rámutattunk, hogy a kereskedelemben is elérhető IPFRR séma, az LFA, jellegéből adódóan nem képes teljes védelmet nyújtani a hálózatban. Hogy ezt a korlátot áthidaljuk, bemutattunk kettő hálózatkiegészítésen alapuló stratégiát (Chapter 2-3).

Az élkiegészítésen alapuló eredményeinket az *Ericsson Research* felhasználta egy belső szoftver megalkotásához, mely képes adott hálózatok LFA analízisére és optimalizálására szolgáltatói hálózatokban. Eredményeink tudományos jelentésekben kerültek hivatkozásra, úgymint IEEE [37] és FIA [38] kiadványok. Munkánk következményeképp Tapolcai mutatott egy redundáns fákon alapuló konstrukciót, mely teljes LFA lefedettséget ér el meghatározott számú virtuális pont hozzáadásával [39].

Végül, de nem utolsó sorban újfajta tömörítési eljárásokkal foglalkoztunk és bemutattuk az R3D3 adatstruktúrát. Lehetséges felhasználási területeket mutattunk a 4.3-as bekezdésben, úgymint *szöveg indexelés*, *géntérkép tömörítés*, *adatbányászat*, *adatbázisok* és *továbbítási táblák tömörítése*.

Publikációk

Folyóiratcikkek

[J1] **M. Nagy**, J. Tapolcai and G. Rétvári. „R3D3: A Doubly Opportunistic Data Structure for Compressing and Indexing Massive Data". *Infocommunications Journal*, pp. 58-66., 2019. (4/2 = 2)

[J2] **M. Nagy**, J. Tapolcai and G. Rétvári. „Node Virtualization for IP Level Resilience". *IEEE/ACM Transaction on Networking*, 2018. (6/2 = 3)

[J3] **M. Nagy**, J. Tapolcai and G. Rétvári. „Optimization Methods for Improving IP-level Fast Protection for Local Shared Risk Groups with Loop-Free Alternates". *Telecommunication Systems 56*, pp. 103-119., 2014. (6/2 = 3)

[J4] L. Csikor, **M. Nagy** and G. Rétvári. „Network Optimization Techniques for Improving Fast IP-level Resilience with Loop-Free Alternates". *Infocommunications Journal*, pp. 2-10., 2012. (4/2 = 2)

Konferenciacikkek

[C1] **M. Nagy**, J. Tapolcai and G. Rétvári. „On the Design of Resilient IP Overlays". In *Proc., DRCN 2014*, Ghent, Belgium, 1-3. April 2014. (3/2 = 1.5)

[C2] **M. Nagy** and G. Rétvári. „IP hálózatok védelmének optimalitása többszörös hibák esetére". In *Proc., Mesterpróba*, Budapest, Hungary, May 2012. (1/1 = 1)

[C3] **M. Nagy** and G. Rétvári. „An evaluation of approximate network optimization methods for improving IP-level fast protection with Loop-Free Alternates". In *Proc., RNDM 2011*, Budapest, Hungary, September 2011. (3/1 = 3)

Egyéb publikációk

[C4] M. Szalay and **M. Nagy** and D. Géhberger and Z. Kiss and P. Mátray and F. Németh and G. Pongrácz and G. Rétvári and L. Toka. „Industrial-scale Stateless Network Functions". In *Proc., IEEE Cloud*, Milan, Italy, 2019. ($3/8 = 0.37$)

Szabadalmak

[P1] **M. Nagy** and D. Fiedler and D. Géhberger and P. Mátray and G. Németh and B. Pinczel „Fast session restoration for latency sensitive middleboxes". International Application No. PCT/IB2019/060031, TELEFONAKTIEBOLAGET LM ERICSSON, 2018. ($2/6 = 0.33$)

[P2] **M. Nagy** and D. Fiedler and D. Géhberger and P. Mátray and G. Németh and B. Pinczel and A. Császár „N+1 redundancy for virtualized services with low latency failover". International Application No. PCT/IB2019/060037 , TELEFONAKTIEBOLAGET LM ERICSSON, 2018. ($2/7 = 0.28$)

Teljes publikációs pontszám: 16.48

Irodalomjegyzék

[1] L. Humphreys, T. von Pape, and V. Karnowski, „Evolving Mobile Media: Uses and Conceptualizations of the Mobile Internet," *Journal of Computer-Mediated Communication*, 2013.

[2] J. Moy, „Ospf version 2," RFC 2328, Apr 1998.

[3] D. Oran, „Osi is-is intra-domain routing protocol," RFC 1142, Febr 1990.

[4] G. Iannaccone, C.-N. Chuah, R. Mortier, S. Bhattacharyya, and C. Diot, „Analysis of link failures in an ip backbone," in *ACM SIGCOMM Internet Measurement Workshop*, 2002, pp. 237–242.

[5] M. Shand and S. Bryant, „IP Fast Reroute framework," RFC 5714, Jan 2010.

[6] A. Atlas and A. Zinin, „Basic specification for IP fast reroute: Loop-Free Alternates," RFC 5286, 2008.

[7] Cisco Systems, „Cisco IOS XR Routing Configuration Guide for the Cisco CRS Router, Release 4.2," 2012.

[8] Hewlett-Packard, „HP 6600 Router Series: QuickSpecs," 2008, available online: http://h18000.www1.hp.com/products/quickspecs/13811_na/13811_na.PDF.

[9] Juniper Networks, „JUNOS 12.3 Routing protocols configuration guide," 2012.

[10] M. Nagy, „Github homepage," https://nmate.github.io, 2019.

[11] „LEMON – Library for Efficient Modeling and Optimization in Networks," http://lemon.cs.elte.hu/, 2014.

[12] „GUROBI – Linear Programming Solver," http://www.gurobi.com, 2018.

[13] „BOOST – C++ Libraries," http://www.boost.org, 2018.

[14] R. Mahajan, N. Spring, D. Wetherall, and T. Anderson, „Inferring link weights using end-to-end measurements," in *ACM IMC*, 2002, pp. 231–236.

[15] SNDlib, „Survivable fixed telecommunication network design library," http://sndlib.zib.de.

[16] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, „The Internet Topology Zoo," http://www.topology-zoo.org.

[17] P. Ferragina, G. Navarro, „Pizza & Chili Corpus," http://pizzachili.dcc.uchile.cl/.

[18] UCSC Genome Bioinformatics, „The UCSC Genome Browser," http://hgdownload.cse.ucsc.edu/downloads.html.

[19] I. Witten, T. Bell, and J. Cleary, „The Calgary Corpus," 1987, http://corpus.canterbury.ac.nz/descriptions/#calgary.

[20] G. Rétvári and A. Kör osi and J. Tapolcai, „The Internet Routing Entry Monitor," http://lendulet.tmit.bme.hu/fib_comp/.

[21] G. Rétvári, J. Tapolcai, G. Enyedi, and A. Császár, „IP Fast ReRoute: Loop Free Alternates revisited," in *INFOCOM 2011*, 2011, pp. 2948–2956.

[22] M. Garey, , and D. Johnson, *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., 1990.

[23] B. Mazbic-Kulma and K. Sep, „Some approximation algorithms for minimum vertex cover in a hypergraph," in *Computer Recognition Systems 2*, ser. Advances in Soft Computing, M. Kurzynski, E. Puchala, M. Wozniak, and A. Zolnierek, Eds. Springer Berlin / Heidelberg, 2007, vol. 45, pp. 250–257.

[24] L. Lovász, „On the ratio of optimal integral and fractional covers," *Discrete Mathematics*, vol. 13, no. 4, pp. 383–390, 1975.

[25] P. Kulaga, P. Sapiecha, and K. Sej, „Approximation Algorithm for the Argument Reduction Problem," in *Computer recognition systems: proceedings of the 4th International Conference on Computer Recognition Systems, CORES'05*. Springer Verlag, 2005, p. 243.

[26] Cisco, „Cisco catalyst 4500 series switch software configuration guide, 15.0," 2016.

[27] G. Jacobson, „Space-efficient static trees and graphs," in *Proceedings of the 30th Annual Symposium on Foundations of Computer Science*, ser. FOCS '89, 1989, pp. 549–554.

[28] A. Brodnik and I. Munro, „Membership in constant time and almost-minimum space," *Journal on Computing*, vol. 28, no. 5, pp. 1627–1640, 1999.

[29] R. Pagh, „Low redundancy in static dictionaries with constant query time," *Journal on Computing*, vol. 31, no. 2, pp. 353–363, 2001.

[30] V. R. R. Raman and S. R. Satti, „Succinct indexable dictionaries with applications to encoding k-ary trees, prefix sums and multisets." *ACM Transactions on Algorithms*, vol. 3(4), 2007.

[31] F. Claude and G. Navarro, „Practical rank/select queries over arbitrary sequences," in *Proc. 15th International Symposium on String Processing and Information Retrieval (SPIRE)*, ser. LNCS 5280. Springer, 2008, pp. 176–187.

[32] G. Navarro and E. Providel, *Fast, Small, Simple Rank/Select on Bitmaps.* Springer Berlin Heidelberg, 2012, pp. 295–306.

[33] D. E. Knuth, *The Art of Computer Programming: Combinatorial Algorithms*, ser. Series in Computer Science. Addison-Wesley, 2011.

[34] D. Okanohara and K. Sadakane, „Practical entropy-compressed rank/select dictionary," in *Proceedings of the Meeting on Algorithm Engineering & Expermiments*, 2007, pp. 60–70.

[35] S. Gog, „Compact and succinct data structures: From theory to practice," available online: http://es.csiro.au/ir-and-friends/20131111/anu_gog_seminar.pdf, 2015.

[36] G. Rétvári, J. Tapolcai, A. Kőrösi, A. Majdán, and Z. Heszberger, „Compressing IP forwarding tables: towards entropy bounds and beyond," in *ACM SIGCOMM*, 2013, paper http://lendulet.tmit.bme.hu/~retvari/publications/sigcomm_2013_tech_rep.pdf, pp. 111–122.

[37] M. Chiesa, A. Kamisiński, J. Rak, G. Rétvári, and S. Schmid, „Fast recovery mechanisms in the data plane," in *IEEE Communications Surveys and Tutorials*, 2020.

[38] L. Csikor, G. Rétvári, and J. Tapolcai, „High availability in the Future Internet," in *The Future Internet*, ser. Lecture Notes in Computer Science, A. Galis and A. Gavras, Eds. Springer Berlin Heidelberg, 2013, vol. 7858, pp. 64–76.

[39] M. Nagy, J. Tapolcai, and G. Rétvári, „Node virtualization for IP level resilience," *IEEE/ACM Transactions on Networking*, pp. 1–14, 2018.