Eötvös Loránd Tudományegyetem
Matematikai Intézet

Simon Anna Dorottya

Hatékony elosztott adattárolási rendszerek

BSc szakdolgozat

Témavezető: Bérczi-Kovács Erika

ELTE Operációkutatási Tanszék

2015
Budapest

Tartalomjegyzék

1. fejezet

Bevezetés

Adattárolás terén mindig fontos szempont az, hogy meghibásodás, például egy CD megkarcolódása esetén is visszanyerhető maradjon az adatunk. Ennek legegyszerűbb módja az adat többszöri eltárolása, de már számos kódolási technika létezik, melyek előnye többek között az, hogy kevesebb helyet foglal, és meghibásodás esetén mégis elérhető marad az adat.

Ha nagy mennyiségű adatot kell tárolnunk, előfordulhat, hogy egy lemez már nem elég, ilyenkor van szükség többlemezes tárolási módszerekre. Érdemes megemlíteni a RAID (Redundant Array of Independent Disks - független lemezek redundáns tömbje) technológiát, mely a leginkább elterjedt tárolási módszer, amiben az adatokat több lemezen tároljuk, melyeken nemcsak az adatok vannak, hanem néhány lemezen a hibajavító kód található.

Ha a lemezek, amelyeken az adatot tároljuk, külön tárolópontokba tömörülnek, amelyeket egy hálózat köt össze, a rendszert hálózati adattároló rendszernek nevezzük (Networked Distributed Storage System - továbbiakban NDSS). Egy ilyen rendszerben gyakran előfordul, hogy néhány tárolópont nem elérhető, akár a hálózat vagy tárolóeszközök meghibásodása, akár a P2P (peer-to-peer, azaz közvetlenül egymással kommunikáló) hálózatokban a felhasználó által működtetett tárolópontok ideiglenes vagy végleges megszüntetése miatt. Éppen ezért fontos a nagy adattároló rendszerekben, hogy bizonyos számú eszköz kimaradása esetén is elérhetők maradjanak az adatok. Ezt sokáig szintén csak többszöri eltárolással oldották meg, de a fentebb említett kódolási technikákkal és még egyéb módszerek segítségével jobb eredményeket is elérhetünk. Az egy helyen tárolt adatokhoz képest mások az optimális kódolási módszerek, mivel mások a céljaink.

Szakdolgozatom során azt mutatom be, hogy mivel előfordulhat tárolópontok megszűnése, az ezt követő javítás miatt milyen korlátok és összefüggések adhatók a fájl

méretére, a tárolópont kapacitására, és a tárolópontok közötti adatátvitel méretére. Speciális esetként külön vizsgálom több tárolópont egyidejű meghibásodását, azt az esetet, ha a javítás során az új tárolópont nem csak megadott számú tárolóponttal kommunikálhat, illetve azt, amikor a tárolópontok kapacitása nem egyforma. Ezen túl foglalkozom még rétegelt videófájlok eltárolásával oly módon, hogy az adott kapacitás mellett minél több felhasználó számára elérhető legyen.

2. fejezet

Kódelméleti alapfogalmak

Ahhoz, hogy a hálózati adattároló rendszerek működését megértsük, tisztában kell lennünk a kódelmélet alapfogalmaival. Ehhez a [7] könyvet vesszük alapul. Egy elkódolandó szót k hosszú sorvektorként ábrázolunk, elemei az A ábécé betűi, mely ábécé q elemből áll. A **kódolás** folyamata egy φ függvény, mely az elkódolandó szavak halmazáról (A^k) a $C \in A^n$ kódszavak halmazára képez. Ha ez a függvény injektív, akkor a kódolás által meghatározott kód **egyértelműen dekódolható**, különben veszteséges. A továbbiakban csak egyértelműen dekódolható kódokkal foglalkozunk.

1. Definíció. Két szó **Hamming-távolságán** azon helyek számát értjük, ahol a két szó különbözik. Egy kód Hamming-távolsága a kódszavak Hamming-távolságainak minimuma, ezt a továbbiakban $dist$-tel jelöljük.

2. Definíció. Egy kód **t-hibajelző**, ha bármely kódszót legfeljebb t helyen megváltoztatva a kapott vektor nem lehet kódszó. Tehát egy kód pontosan t-hibajelző, ha $t = dist - 1$. Egy kód **t-hibajavító**, ha bármely két különböző kódszó esetén mindkettőt t-t darab helyen megváltoztatva a kapott vektor nem lehet ugyanaz. Egy kód pontosan t-hibajavító, ha $t = \left\lfloor \frac{dist-1}{2} \right\rfloor$.

A továbbiakban **lineáris kódokat** fogunk használni, melynek ábécéje egy $\mathbb{F}_q$ véges test, és φ egy lineáris leképezés, amit tekinthetünk egy G $n \times k$-as, k rangú mátrixszal való szorzásnak is. Ezt a G mátrixot nevezzük a kód **generátormátrixának**. Ekkor a C az $\mathbb{F}_q^n$ k-dimenziós lineáris altere. Ezeket a kódokat (n, k) paraméterű kódoknak nevezzük.

3. Definíció. Egy szó **súlya** a 0-tól különböző jegyek száma. Egy kód súlya a nemnulla kódszavak súlyának minimuma, melynek jelölése a továbbiakban w.

Lineáris kódoknál a távolság és a súly megegyezik.

4. Tétel (Singleton-korlát)**.** *Adott $C \subseteq \mathbb{F}_q^n$ lineáris kódra $dist \leq n - k + 1$. Ha ez egyenlőséggel teljesül, akkor a kódot* ***MDS-kódnak****, azaz maximális távolságú szeparábilis kódnak nevezzük.*

Jelöljük m-mel azt a legnagyobb számot, amire G-nek van olyan m darab sora, hogy az általuk meghatározott részmátrix oszlopai összefüggők (kevesebb, mint k a rangjuk). Tudjuk, hogy $m \geq k - 1$, illetve hogy $n - m \geq dist$. Tehát ha a kód MDS-kód, akkor $n - (n - k + 1) \geq m \geq k - 1$, azaz $m = k - 1$, vagyis ha legalább k sort választunk ki, akkor már azok rangja k.

5. Tétel (Hamming-korlát)**.** *Adott $C \subseteq \mathbb{F}_q^n$ lineáris kódra*

$$|C| \leq \frac{q^n}{\sum_{k=0}^{\lfloor \frac{dist-1}{2} \rfloor} \binom{n}{k}(q-1)^k}$$

Ha ez egyenlőséggel teljesül, akkor a kódot ***perfekt kódnak*** *nevezzük.*

Legyen most az α az $\mathbb{F}_q$ véges test egy olyan nemnulla eleme, melynek multiplikatív rendje n és $0 < k < n$. Ekkor az α^i $(i = 1, \ldots, n)$ elemek páronként különböznek, és ezek a gyökei az $(x^n - 1) \in \mathbb{F}_q[x]$ polinomnak. Tehát $x^n - 1 = \prod_{i=0}^{n-1}(x - \alpha^i)$. Legyen $m = n - k$, és $g = \prod_{i=1}^{m}(x - \alpha^i)$, ez egy m-edfokú osztója $x^n - 1$-nek.

Legyen ekkor $C = \{ag : a \in \mathbb{F}_q[x], deg(a) < k\}$, ezt nevezzük az α által generált **Reed-Solomon-kódnak**, melynek generátorpolinoma a g. A kódszavak azon $c \in C$, melyre minden $i = 1, \ldots, m$-re $c(\alpha^i) = 0$. Ennek a kódnak a minimális távolsága $m + 1 = n - k + 1$, és egyenlőséggel teljesül rá a Singleton-korlát, azaz MDS-kód.

3. fejezet

Javítás

A következőkben azt vizsgáljuk, hogy ha az NDSS bizonyos tárolópontjai meghibásodnak, akkor hogyan érdemes a javításukat megoldani új tárolópontok létrehozásával.

6. Definíció. A **NDSS** egy fájltárolási rendszer, amely egy M részre feldarabolt fájl eltárolására n tárolópontot használ fel, melyekben egyenként α részt tárolunk, oly módon, hogy bármely k tárolópontból letöltött adatok segítségével vissza tudjuk állítani az eredeti fájlt.

Hogy az adatok eltárolása és visszaállítása pontosan milyen kódolás segítségével történhet, arról a fejezet 3.4. részében lesz szó.

Egy tárolópontot **aktívnak** nevezünk, ha elérhető, **inaktívnak**, ha meghibásodás vagy más ok miatt nem elérhető.

7. Definíció. A **javítás** t darab tárolópont egyidejű inaktívvá válása esetén a következőképpen történik: t darab új tárolópontot készítünk, egy új tárolópont $d \geq k$ segítő tárolóponttal lép kapcsolatba, majd tárolópontonként β adatot tölt le belőle, és β' adatot tölt le minden más új ponttól, úgy, hogy mindezen adatok letöltése után az új tárolópontokkal együtt az NDSS újra megfeleljen a definíciónak. A javítás során egy új tárolópont által a többitől letöltött adat mennyisége $\gamma = d\beta + (t-1)\beta'$, ezt a továbbiakban **adatforgalomnak** nevezzük.

A javítást illetően megkülönböztetünk funckionális és egzakt javítást.

8. Definíció. Az **egzakt javítás** után az NDSS új tárolópontjai megfeleltethetők a régieknek, és egy új tárolópontban pontosan ugyanazok az adatok lesznek, mint a régiben voltak. A **funkcionális javítás** során csak az NDSS definíció szerinti tulajdonságait állítjuk vissza, az új pontokban lévő adatok lehet, hogy nem lesznek pont ugyanazok, mint a régi pontokban lévők voltak.

A_1	A_3	$A_1 + A_3$	$A_2 + A_3$
A_2	A_4	$A_2 + A_4$	$A_1 + A_2 + A_4$

3.1. ábra. A meghibásodás előtti NDSS

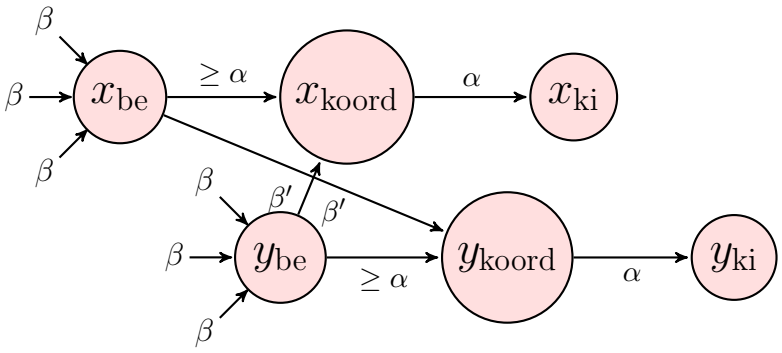
A két javítás közötti különbségek megértéséhez nézzük az alábbi egyszerű példát! Egy tárolópont 2 részt tárol el, és a négy részből álló $A = (A_1, A_2, A_3, A_4)$ fájlt akarjuk eltárolni 4 tárolóponton úgy, hogy bármely 2 tárolópontból visszaállítható legyen a fájl. A darabok a 3.1. táblázat szerint vannak elosztva.

Ekkor az első tárolópont meghibásodása esetén az új tárolópont a három régi tárolóponttól tölt le adatokat. Egzakt javítás esetén mindhárom tárolóponttól az első sorban szereplő részt tölti le, majd az $A_1 + A_3$-ból és az $A_2 + A_3$-ból kivonja A_3-at, így vissza tudja állítani az eredeti A_1-et és A_2-t. Funkcionális javítás esetén pedig például az A_4-et, $A_1 + A_3$-at és $A_2 + A_3$-at tölti le, és hozzáadja az A_4-et a másik kettőhöz. Így egy $(A_1 + A_3 + A_4, A_2 + A_3 + A_4)$-et tároló pontot hoz létre, de ez úgyanúgy helyreállítja az NDSS-t.

Hogy a javítás után helyreállt-e az NDSS definíció szerint, egy-egy **adatgyűjtővel** ellenőrizzük, amely k darab aktív tárolóponthoz kapcsolódik, és onnan minden adatot letölt, és ebből visszaállítja a fájlt. Mivel n darab aktív tárolópont van összesen, ezért $\binom{n}{k}$ darab adatgyűjtővel tudjuk leellenőrizni.

Javítások folyamatának vizsgálatához definiálunk egy élkapacitásos irányított gráfot, ahol a c élkapacitás jelöli az adott élen küldött adat mennyiségét. Ezt a c élkapacitást **adatátvitelnek** nevezzük. A gráfban van egy S forrás és egy T nyelő, amely az adatgyűjtőnek felel meg. Az eredeti n darab tárolópontot egy-egy x_{ki} csúccsal ábrázolunk, melybe egy darab α kapacitású él vezet az S forrásból.

Ha az x tárolópont új, vagyis a javítás során keletkezik, akkor három csúccsal ábrázoljuk a gráfban, egy x_{be}, egy x_{koord} és egy x_{ki} csúccsal. Az x_{be} csúcsba d darab β kapacitású él vezet azon régebbi tárolópontok x_{ki} csúcsaiból, melyektől x adatot kapott. Az x_{be}-ből x_{koord}-ba vezető élen a kapacitás az az adatmennyiség, amit a tárolópont átmenetileg tárol (ez legalább α), az x_{koord}-ból x_{ki}-be vezető élen pedig a kapacitás α. Egy javítás során a t új tárolópont közötti együttműködést az egyik tárolópont x_{be} csúcsából induló, a másik tárolópont x_{koord} csúcsába vezető β' kapacitású élekkel jelöljük (minden x_{koord} csúcsba $t-1$ ilyen él vezet). Ezt szemlélteti a 3.2. ábra. A T nyelőbe pedig az adatgyűjtő által meghatározott k aktív tárolópont x_{ki} csúcsából vezet végtelen kapacitású él.



3.2. ábra. A gráf részlete $t = 2$ és $d = 3$ esetén

9. Definíció. Az adott $(M, n, k, d, t, \alpha, \beta, \beta')$ paraméterekre létezik gráfoknak egy családja, amely a különbözőképpen alakuló NDSS-hálózatokat szemlélteti. Ezt a halmazt nevezzük $NDSS(M, n, k, d, t, \alpha, \beta, \beta')$-nek.

10. Definíció. Egy élkapacitásos digráfban, melynek S és T két csúcsa, **S$\overline{\text{T}}$-vágásnak** nevezünk egy $(U, \overline{U})$ halmazpárt, amelyre $S \in U$ és $T \notin U$. A **vágás értéke** az U-ból kilépő élek kapacitásának összege.

Egy NDSS javítását leíró gráfban minden $S\overline{T}$-vágás c-szerinti értékének legalább M nagyságúnak kell lennie, hiszen ekkora adatnak el kell jutnia a feltöltéstől az adatgyűjtőig.

Ezen megfigyelés segítségével a következőkben a fenti paraméterek között keresünk kapcsolatot, és megnézzük, hogy minimális tárolási hely (α) vagy minimális adatforgalom (γ) esetén legalább mekkorának kell lennie a többi paraméternek.

3.1. Javítás vizsgálata egy pontú hiba esetére

Ebben a részben a [3] cikket dolgozzuk fel. Az optimális választásról szóló tételek kimondásához először tekintsük azt az egyszerűbb esetet, amikor $t = 1$. A gráfok tehát $NDSS(M, n, k, d, 1, \alpha, \beta, 0)$ halmazbeliek.

Ez esetben a gráfban egy tárolópont x_{koord} kifoka és befoka is egy, ezért elhagyható. Így a tárolópont csak két csúcsból áll, az x_{be} és az x_{ki} csúcsokból, melyek között α a kapacitás.

11. Tétel (Dimakis, Godfrey, Wu, Wainwright, Ramchandran; [3])**.** *Egy adott $NDSS(M, n, k, d, 1, \alpha, \beta, 0)$-beli gráfban a c szerinti minimális $S\overline{T}$-vágás legalább*

$$\sum_{i=0}^{k-1} \min\{(d-i)\beta, \alpha\}$$

és ez az érték fel is vétetik.

Bizonyítás. Először lássuk be, hogy az érték felvétetik egy $NDSS(n,k,d,1,\alpha,\beta,0)$-beli gráf egy $S\overline{T}$-vágásán. Ehhez egy olyan esetet nézünk, ahol k darab javítás történt egymás után. Az ezt ábrázoló gráfot nevezzük G^*-nak. Az n darab eredeti tárolópontot számozzuk 1-től n-ig, a k darab új tárolópontot pedig $n+1$-től $n+k$-ig. Az $n+i$. tárolópont az $n+i-d,\ldots,n+i-1$ tárolópontokhoz kapcsolódik, az adatgyűjtő pedig a k új tárolóponthoz.

12. Lemma. *A G^* minimális $S\overline{T}$-vágásának értéke $\sum_{i=0}^{k-1}\min\{(d-i)\beta,\alpha\}$.*

Bizonyítás. Megadunk egy $(U,\overline{U})$-vágást, melynek minden éle az új tárolópontok x_{be} és x_{ki} csúcsai között lévő, vagy az x_{be} csúcsa és az oda belépő d darab x_{ki} csúcsból induló él.

A vágást az alábbi módon adjuk meg: $i=1\ldots k$-ra ha $\alpha\leq(d-i)\beta$, akkor az $n+i$. tárolópont x_{be} csúcsa U-ba kerül, ha nem, akkor $\overline{U}$-ba. Ekkor a vágás értéke pontosan $\sum_{i=0}^{k-1}\min\{(d-i)\beta,\alpha\}$.

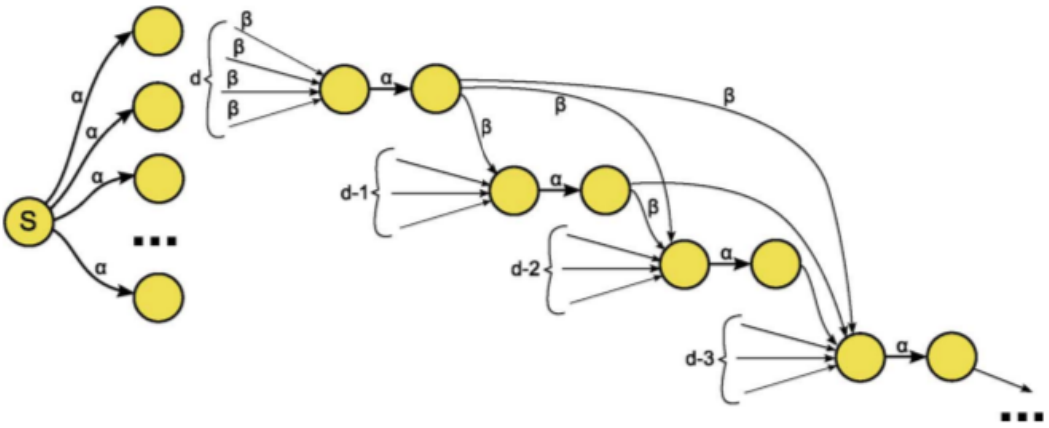
Most pedig nézzük meg, hogy ez a vágás minimális-e G^*-ban. Mivel a T-be vezető élek végtelen kapacitásúak, ezért az x_{ki} csúcsok biztosan $\overline{U}$-ban vannak. Az új tárolópontok x_{be} csúcsaiba menő, és azokból kiinduló élek kapacitását figyelembe véve úgy választottuk a pontokat, hogy az élek hozzájárulása a kapacitáshoz minimális legyen.

Tehát már csak az eredeti tárolópontok csúcsait kell megvizsgálnunk. Az eredeti tárolópontokat csak egy csúcs jelképezi, hiszen a feltöltőből α kapacitású élek indulnak. Vagyis ha beleveszünk az $\overline{U}$-ba egy eredeti tárolópont csúcsát, akkor egyrészt minden esetben nő α-val a vágás értéke, másrészt tekintsük a belőle adatot kinyerő tárolópontok x_{be} csúcsait! Ha U-ban van egy ilyen csúcs, akkor amiatt nem csökken a vágás értéke, viszont ha benne volt az $\overline{U}$-ban volt j darab ilyen tárolópont x_{be} csúcsa, akkor csökken $j\beta$-val a vágás értéke. Viszont mivel legalább j darab csúcs $\overline{U}$-ban volt, ezért tudjuk, hogy $(d-k+j)\beta\leq\alpha$, mivel így választottuk ki az $\overline{U}$ elemeit. Mivel $d\geq k$, ezért $j\beta\leq\alpha$, vagyis nem csökken a vágás értéke.

Mivel az eredeti tárolópontok csúcsai nem kapcsolódnak egymáshoz éllel, ezért több csúcs bevétele esetén sem csökken a vágás értéke. Tehát ez a vágás minimális. □

Ezzel bebizonyítottuk azt, hogy a minimum felvétetik, így már csak az egyenlőtlenség belátása van hátra.

13. Lemma. *Bármely $NDSS(M,n,k,d,1,\alpha,\beta,0)$-beli gráfban a minimális $S\overline{T}$-vágás értéke legalább $\sum_{i=0}^{k-1}\min\{(d-i)\beta,\alpha\}$.*



3.3. ábra. A G^* gráf. Az ábra forrása: [3]

Bizonyítás. Miután T-be végtelen kapacitású élek érkeznek, elég azokkal az $(U, \overline{U})$ vágásokkal foglalkoznunk, ahol ezek nincsenek benne, vagyis azon k tárolópont x_{ki} része, amikhez kapcsolódik a nyelő, az $\overline{U}$-ban van. Ha az adatgyűjtő eredeti tárolóponthoz is kapcsolódik, akkor az S-et tekintjük a tárolópont x_{be} csúcsának, amely mindig U-ban van.

Mivel az $NDSS(M, n, k, d, 1, \alpha, \beta, 0)$-beli gráfok irányított aciklikus gráfok, vehetjük a csúcsoknak egy topologikus sorrendjét. Tekintsük az első $\overline{U}$-beli x_{ki} csúcsot. Az x_{be} csúcs helyzetétől függően két eset lehetséges: ha U-ban van, akkor a vágáshoz az x_{be} x_{ki} él tartozik, ha $\overline{U}$-ban van, akkor pedig a d darab x_{be}-be vezető él tartozik a vágáshoz.

Nézzük most a második $\overline{U}$-beli x_{ki} csúcsot. Hasonlóan az előzőhöz vagy az x_{be} x_{ki} él, vagy a legalább $d-1$ darab x_{be}-be vezető él tartozik a vágáshoz. Azért $d-1$, mert az egyik él forrása már lehet a topologikus sorrend szerinti első $\overline{U}$-beli x_{ki} csúcs.

Mind a k darab $\overline{U}$-beli x_{ki} csúcsot megvizsgálva arra jutunk, hogy az i. csúcshoz tartozó, vágásban szereplő élek között van vagy egy α kapacitású él, vagy legalább $(d-i)$ darab β kapacitású él. Ezért ennek a vágásnak az értéke legalább

$$\sum_{i=0}^{k-1} \min\{(d-i)\beta, \alpha\}$$

Tehát bebizonyítottuk a lemmát. □

Ezzel a 11. tételt is beláttuk. □

Tehát az NDSS definíciója szerint szükséges, hogy ha egy javításnál az új pont bármilyen d ponthoz csatlakozhat, akkor a fenti szummánál kisebb vagy egyenlő

legyen M:

$$\sum_{i=0}^{k-1} \min\{(d-i)\beta, \alpha\} \geq M \tag{3.1}$$

3.2. Javítás vizsgálata több pontú hiba esetére

Ebben a részben a [8] összefoglaló 6. fejezetét dolgozzuk fel. Most pedig tekintsük azt az esetet, amikor nincs semmiféle megkötésünk a t-re. Ekkor az i. javításban résztvevő új tárolópontokat az i. csoportnak nevezzük.

14. Tétel (Kermarrec, Le Scouarnec, Straub; [6]). *Egy $NDSS(M, n, k, d, t, \alpha, \beta, \beta')$-beli gráfban, a c szerinti minimális $S\overline{T}$-vágás legalább*

$$\min_{\mathbf{u}\in P}\Big(\sum_{i=0}^{g-1} u_i \min\{(d-\sum_{j=0}^{i-1} u_j)\beta + (t-u_j)\beta', \alpha\}\Big),$$

ahol $P = \{\mathbf{u} : u_i \leq t, u_0 + \cdots + u_{g-1} = k\}$, illetve van olyan gráf, ahol ez az érték fel is vétetik.

Bizonyítás. A következőkben most az előzőekhez hasonlóan belátjuk, hogy a fenti érték felvétetik egy $NDSS(n, k, d, t, \alpha, \beta, \beta')$-beli G^* gráf egy $S\overline{T}$-vágásán. Feltesszük, hogy g darab javítás volt, mindegyik egyszerre legfeljebb t meghibásodott tárolóponttal, tehát $k/t \leq g \leq k$. A nyelőbe mindig k csúcsból megy él. Ebben a speciális esetben k darab tárolópont hibásodott meg összesen, és a nyelőbe ebből a k új tárolópontnak megfelelő x_{ki} csúcsból megy él. Ebben az esetben az u_{i-1} az i. csoportban lévő új tárolópontok számát jelöli, melyet egyelőre nem határozunk meg, hanem változóként kezelünk.

15. Lemma. *A G^* minimális $S\overline{T}$-vágásának c szerinti értéke*

$$\min_{\mathbf{u}\in P}\Big(\sum_{i=0}^{g-1} u_i \min\{(d-\sum_{j=0}^{i-1} u_j)\beta + (t-u_j)\beta', \alpha\}\Big).$$

Bizonyítás. Megadunk egy $(U, \overline{U})$-vágást. Minden olyan tárolópontnak, ami egy javítás során új volt, az x_{ki} része az $\overline{U}$-ban van.

Nézzük először a gráf azon részét, amely az első javításban résztvevő új tárolópontokat modellezi! Ekkor az u_0 darab tárolópont mindegyik x_{ki} része az $\overline{U}$-ban van, az x_{be}-k közül U-ban lévő csúcsok számát jelöljük m_0-lal, ekkor $u_0 - m_0$ darab van $\overline{U}$-ban. Az m_0 darab, részben U-ban lévő tárolópontoknak az x_{koord} része lehet U-ban

vagy $\overline{U}$-ban. Ha U-ban van, akkor a vágáshoz egy α kapacitású éllel járul hozzá, ha nem, akkor egy $\geq \alpha$ kapacitásúval. Így a vágás értéke akkor lesz kisebb, ha mindegyik U-ban van, vagyis ekkor $m_0\alpha$ a vágásérték az m_0 darab ponthoz csatlakozó éleken.

Mivel ez az első csoport, a maradék u_0-m_0 darab $\overline{U}$-beli x_{be} csúcsokba csak U-beli csúcsokból érkezhet él, vagyis $d\beta$ kapacitás pontonként. Ezen felül az x_{koord} csúcsokba $t-u_0+m_0$ darab U-beli x_{be} csúcsból vezet él, mivel egy új tárolópont $t-1$ másik újhoz kapcsolódik, de ebből u_0-m_0-1 már $\overline{U}$-ban van. Tehát $(u_0-m_0)(d\beta+(t-u_0+m_0)\beta')$ a vágás értéke az $u_0 - m_0$ darab pontba lépő vágásbeli éleken.

Legyen a vágás értéke az első csoportot tekintve c_0! Ekkor:

$$c_0 = m_0\alpha + (u_0 - m_0)(d\beta + (t - u_0 + m_0)\beta')$$

Mivel minimális vágást szeretnénk, ezért m_0-t olyannak választjuk, hogy c_0 a lehető legkisebb legyen. Mivel a jobb oldal konkáv a $[0, u_0]$ intervallumon, ezért a minimuma 0-ban vagy u_0-ban vétetik fel, a minimumhelytől függően tehát vagy minden első csoportbeli x_{be} csúcs U-ban vagy $\overline{U}$-ban van.

$$c_0 = \min\{u_0(d\beta + (t - u_0)\beta'), u_0\alpha\}$$

A második csoportnál hasonlóan járunk el, viszont az $u_1 - m_1$ darab $\overline{U}$-beli x_{be} csúcsba U-ból már csak $(u_1 - m_1)(d - u_0)$ darab β kapacitású él vezet, hiszen ezek már csatlakozhatnak az u_0 darab $\overline{U}$-beli pontra (amiknek az x_{ki} pontja mindenképpen $\overline{U}$-ban van). Azaz, ha a vágás értéke a második javítás után c_1, akkor:

$$c_1 = m_1\alpha + (u_1 - m_1)((d - u_0)\beta + (t - u_1 + m_1)\beta')$$

$$c_1 = \min\{u_1((d - u_0)\beta + (t - u_1)\beta'), u_1\alpha\}$$

Az $i+1$. csoportnál pedig az $u_i - m_i$ darab $\overline{U}$-beli x_{be} csúcs van. Ezekbe összesen $(u_i - m_i)(d - (u_0 + \ldots + u_{i-1})$ darab β kapacitású él vezet, mivel ennyi x_{ki} csúcs van $\overline{U}$-ban. A β' és α kapacitású élek hozzájárulása indexektől eltekintve teljesen megegyezik az első csoportéval. Így tehát ha a vágás értéke c_i, akkor:

$$c_i = m_i\alpha + (u_i - m_i)((d - \sum_{j=0}^{i-1} u_j)\beta + (t - u_i + m_i)\beta')$$

Ez a jobb oldal szintén konkáv $[0, u_i]$-n, tehát m_i-t 0-nak vagy u_i-nek választva:

$$c_i = \min\{u_i((d - \sum_{j=0}^{i-1} u_j)\beta + (t - u_i)\beta'), u_i\alpha\}$$

Miután a vágás értéke a c_i függvények összege, ezért a következő összeggel egyezik meg:

$$\sum_{i=0}^{g-1} u_i \min\{(d - \sum_{j=0}^{i-1} u_j)\beta + (t - u_j)\beta', \alpha\}$$

Ennek a kifejezésnek pedig tudjuk venni a minimumát $\mathbf{u}$ függvényében, vagyis ezzel pontosan meghatároztuk a G^* gráfot és a vágást is.

A 12. lemma bizonyításához hasonlóan belátható, hogy ez a vágás minimális. □

16. Lemma. *Bármely $NDSS(M, n, k, d, t, \alpha, \beta, \beta')$-beli gráfban a minimális $S\overline{T}$-vágás értéke legalább* $\min_{\mathbf{u}\in P}\left(\sum_{i=0}^{g-1} u_i \min\{(d - \sum_{j=0}^{i-1} u_j)\beta + (t - u_j)\beta', \alpha\}\right)$.

Bizonyítás. Mivel a T nyelőbe végtelen kapacitású élek érkeznek, ezért elég azokkal a vágásokkal foglalkoznunk, amelyekben a nyelőhöz kapcsolódó x_{ki} pontok $\overline{U}$-ban vannak. Mivel az $NDSS(M, n, k, d, t, \alpha, \beta, \beta')$-beli gráfok aciklikus gráfok, ezért vehetjük ezen csúcsok topologikus sorrendjét. Most csak azokkal a tárolópontokkal foglalkozunk, melyek csatlakoznak az adatgyűjtőhöz. Ezért most az i. csoport elemei azon tárolópontokból állnak, amelyek az adatgyűjtőhöz csatlakozó tárolópontokat sorba téve az i. javításban vettek részt, számukat jelöljük u_i-vel.

A 13. lemma bizonyításához hasonlóan ha az adatgyűjtő eredeti tárolópontokhoz is kapcsolódik, akkor azokat úgy tekintjük, hogy az x_{koord} csúcsuk az S, és az definíció szerint U-ban van, és úgy vesszük, hogy minden eredeti tárolópont egy olyan javításnak számít, amelyben $u_i = 1$.

Nézzük az i. csoport hozzájárulását a vágás értékéhez! Jelöljük m_i-vel azt a számot, ahány i. csoportbeli x_{be} csúcs van U-ban. Tekintsük először ezt az m_i darab tárolópontot. Ha egy ilyen tárolópont x_{koord} csúcsa U-ban van, akkor a vágáshoz való hozzájárulás értéke α, ha $\overline{U}$-ban van, akkor legalább α.

A maradék $u_i - m_i$ darab tárolóponthoz tartozó x_{koord} csúcsról feltehetjük $\overline{U}$-ban van. Ekkor ebben az esetben a vágáshoz való hozzájárulás legalább $d - \sum_{j=0}^{i-1} u_j$ darab β kapacitású él, hiszen ennyi él legalább U-beli csúcsokból indul, illetve $t - u_i + m_i$ darab β' kapacitású él, hiszen ennyi él lép át U-ból $\overline{U}$-ba az x_{be} és x_{koord} csúcsok között az i. csoportban.

Az i. csoport hozzájárulását alulról tudjuk tehát becsülni a következő kifejezéssel:

$$m_i\alpha + (u_i - m_i)((d - \sum_{j=0}^{i-1} u_j)\beta + (t - u_i + m_i)\beta')$$

Mivel a vágás az összes csoport általi hozzájárulás összegéből áll, ezért minden vágást alulról tudunk becsülni a következő összeggel:

$$\sum_{i=0}^{g-1} \Big(m_i\alpha + (u_i - m_i)((d - \sum_{j=0}^{i-1} u_j)\beta + (t - u_i + m_i)\beta') \Big)$$

Erről pedig már beláttuk, hogy a lemmában szereplő szumma alulról becsüli. □

Ezzel bizonyítottuk a tételt. □

A minimális vágás csak az $\mathbf{u}$ függvényében vett minimumnál lesz mindig nagyobb, de mivel az NDSS javítások során bármilyen $\mathbf{u}$ csoportokat használhat, ezért M-nek kisebbnek kell lennie az összegnél bármilyen $\mathbf{u} \in P$-re, azaz:

$$\sum_{i=0}^{g-1} u_i \min\{(d - \sum_{j=0}^{i-1} u_j)\beta + (t - u_j)\beta', \alpha\} \geq M, \tag{3.2}$$

ahol $P = \{\mathbf{u} : u_i \leq t, u_0 + \cdots + u_{g-1} = k\}$.

3.3. Paraméterek optimális választása

Most pedig nézzük meg adott (M, n, k, d, t) paraméterekre az α és a γ lehetséges választásait abban az esetben, amikor a 3.2 egyenlőtlenség egyenlőséggel teljesül bizonyos u_i-kre. Az ilyen paramétereket optimálisnak nevezzük. Két extrém esetet vizsgálunk meg, amikor α-t minimalizáljuk, illetve amikor γ-t. Ehhez meg kell néznünk azokat az eseteket, amikor a lehető legtöbb adatot küldenek az eredeti tárolópontok az újaknak, vagyis a 3.2-ben minimalizáljuk a $(t-u_j)\beta'$-t, illetve amikor a lehető legtöbb adatot küldenek egymásnak az új tárolópontok, azaz $(d-\sum_{j=0}^{i-1} u_j)\beta$-t minimalizáljuk.

Az első esetben $u_i = t$ minden i-re, és $g = k/t$, vagyis:

$$\sum_{i=0}^{k/t-1} t \min\{(d - it)\beta, \alpha\} = M \tag{3.3}$$

A második esetben pedig $u_i = 1$ minden i-re, és $g = k$, azaz:

$$\sum_{i=0}^{k-1} \min\{(d - i)\beta + (t - 1)\beta', \alpha\} = M \tag{3.4}$$

17. Definíció. Minimum storage regenerating point, azaz **MSR pont**: adott (M, n, k, d, t) paraméterekre α-t minimalizáljuk, majd a γ-t, azaz β-t majd β'-t minimalizáljuk.

18. Tétel (Kermarrec, Le Scouarnec, Straub; [6]). *Adott (M, n, k, d, t) paraméterekre az MSR pont:*

$$\alpha_{MSR} = \frac{M}{k}, \quad \beta_{MSR} = \beta'_{MSR} = \frac{M}{k}\frac{1}{d+t-k}, \quad \gamma_{MSR} = \frac{M}{k}\frac{d+t-1}{d+t-k}$$

Bizonyítás. Mivel az adatot vissza kell tudnunk nyerni bármely k pontból, ezért legalább $\frac{M}{k}$-nak kell lennie α-nak. A 3.3-ban az egyenlet bal oldalán k/t tagú az összeg, mindegyik tag legfeljebb tM/k, ha a minimum a második tagon vétetik fel, viszont legalább ennyinek is kell lennie, hiszen különben nem lenne egyenlőség. Tehát minden i-re:

$$t\min\{(d-it)\beta, \alpha\} = \frac{Mt}{k}$$

$$(d-it)\beta \geq \frac{M}{k}$$

$$\beta \geq \frac{M}{k(d-it)}$$

Ez pedig minden i-re akkor lesz igaz, ha a legnagyobb i-re is igaz, tehát $\beta \geq \frac{M}{k(d-k+t)}$. Nyilván ha ez egyenlőséggel teljesül, akkor lesz a legkisebb β.

A 3.4-ből hasonló elgondolás alapján következik, hogy:

$$\frac{M(d-i)}{k(d-k+t)} + (t-1)\beta' \geq \frac{M}{k}$$

$$\beta' \geq \frac{M(t-k+i)}{k(d-k+t)(t-1)}$$

Ennek teljesülnie kell minden $0 \geq i \geq k-1$-re. Ez akkor lesz igaz, ha $i = k-1$-re igaz, ekkor a fenti törtet tudjuk egyszerűsíteni $t-1$-gyel. β' akkor lesz a legkisebb, ha ez az egyenlőtlenség egyenlőséggel teljesül, azaz:

$$\beta' = \frac{M}{k(d-k+t)}$$

Mivel $\gamma = d\beta + (t-1)\beta'$, ezzel bizonyítottuk az állítást. □

19. Definíció. Minimum bandwidth regenerating point, azaz **MBR pont**: adott (M, n, k, d, t) paraméterekre γ-t, azaz β-t és β'-t minimalizáljuk, majd az α-t ennek függvényében a lehető legkisebbre vesszük.

20. Tétel (Kermarrec, Le Scouarnec, Straub; [6]). *Adott* (M, n, k, d, t) *paraméterekre az MBR pont:*

$$\alpha_{MBR} = \frac{M}{k}\frac{2d+t-1}{2d+t-k},\ \beta'_{MBR} = \frac{M}{k}\frac{1}{2d+t-k},\ \beta_{MBR} = 2\beta'_{MBR},\ \gamma_{MBR} = \alpha_{MBR}$$

Bizonyítás. Ahhoz, hogy γ-t minimalizáljuk, β-t és β'-t kell minimalizálnunk. Vagyis a minimum mindig az első helyen vétessen fel a 3.3-ban, azaz α legyen megfelelően nagy:

$$\sum_{i=0}^{k/t-1} t(d-it)\beta = M$$

$$\beta = \frac{M}{t(\sum_{i=0}^{k/t-1} t(d-it)d - t\sum_{i=0}^{k/t-1} t(d-it)i)}$$

$$\beta = \frac{2M}{k(2d-k+t)}$$

A 3.4-ből hasonlóan, ha α elég nagy:

$$\sum_{i=0}^{k-1} \frac{2M(d-i)}{k(2d-k+t)} + (t-1)\beta' = M$$

$$k(t-1)\beta' = M - \frac{M(2d-k+1)}{k(2d-k+t)}$$

$$\beta' = \frac{M}{k(2d-k+t)}$$

Mivel α nagyobb vagy egyenlő, mint $(d-i)\beta + (t-1)\beta'$ és mint $(d-it)\beta$ minden i-re, és ezek $i = 0$-ra a legnagyobbak, ezért $\alpha = d\beta + (t-1)\beta'$. Vegyük észre, hogy így α megegyezik γ-val. □

3.4. Javítás megvalósítása

Az előbbi állítások és tételek csak korlátokat adnak a legjobb elérhető kódoknak, de nem adnak megvalósítást. Az alábbiakban ezekre adunk példát.

3.4.1. MSR pont egy pontú hiba esetére

Egy pontú hiba esetén az MSR pont paraméterei adott (M, n, k, d)-re:

$$\alpha_{MSR} = \frac{M}{k},\quad \beta_{MSR} = \frac{M}{k}\frac{1}{d+1-k},\quad \gamma_{MSR} = \frac{M}{k}\frac{d}{d+1-k}$$

Legyen most $\beta_{MSR} = 1$, azaz ez lesz az egység, amennyit le tud tölteni egy új tárolópont egy másiktól. Ekkor $M = k(d+1-k)$, és $\alpha_{MSR} = d+1-k$. Ha $d = k+1$, akkor $M = 2k$, $\alpha_{MSR} = 2$, tehát egy tárolópont két egységnyi adatot tárol.

Ezekkel a paraméterekkel a következőképpen tudunk kódot megvalósítani:

Legyen $\mathbb{F}_q$ véges test. A fájl egy $M = 2k$ hosszú $\mathbb{F}_q^M$-beli vektor ($\mathbf{o} = (o_1, \ldots, o_M)$), melyet két k hosszú vektorra, $\mathbf{o}_1$-re, és $\mathbf{o}_2$-re tudunk felbontani. Az i. tárolópont $\alpha = 2$ részt tárol el a következőképpen:

$$x_i = (\mathbf{o}_1\mathbf{p}_i^\intercal, \mathbf{o}_2\mathbf{p}_i^\intercal + \mathbf{o}_1\mathbf{r}_i^\intercal) \in \mathbb{F}_q^2,$$

ahol $\mathbf{p}_i$ és $\mathbf{r}_i$ $\mathbb{F}_q^k$-beli vektorok határozzák meg a kódolást. A $\mathbf{p}_i^\intercal$ vektorok egy (n, k) paraméterű MDS-kód G $k \times n$-es generátormátrixának az oszlopai. Az R szintén $k \times n$-es méretű mátrix, oszlopai pedig az $\mathbf{r}_i^\intercal$ vektorok. Tehát az összes tárolópont elkódolását az alábbi szorzással kapjuk:

$$(\mathbf{o}_1, \mathbf{o}_2) \begin{bmatrix} G & R \\ \mathbf{0} & G \end{bmatrix}$$

Az i. tárolópont által eltárolt adatok az eredményül kapott $\mathbf{h} \in \mathbb{F}_q^{2n}$ vektor i. és $n+i$. elemei. Tehát a kódolás, amit használunk, egy ($2n$,$2k$) lineáris kód.

Már csak azt kell leellenőriznünk, hogy bármely k tárolópontból 2-2 egységnyi adatot letöltve visszaállítható az eredeti fájl, illetve hogy bármely tárolópont meghibásodása esetén visszaállítható az NDSS definíció szerint.

Fájl visszaállítása ♦ Az adatgyűjtő k tárolópontból tölt le összesen $2k$ egységnyi adatot. Mivel a G egy (n, k) paraméterű MDS-kód, ezért ebből már ki tudjuk számolni az $\mathbf{o}_1$-et. Ebből megkapjuk $\mathbf{o}_1\mathbf{r}_i^\intercal$-t, végül az $\mathbf{o}_2\mathbf{p}_i^\intercal$-ból az $\mathbf{o}_1$-hez hasonlóan ki tudjuk számolni az $\mathbf{o}_2$-t.

Tárolópont visszaállítása ♦ Ha a j. tárolópont meghibásodik, azaz elveszik az $(\mathbf{o}_1\mathbf{p}_i^\intercal, \mathbf{o}_2\mathbf{p}_i^\intercal + \mathbf{o}_1\mathbf{r}_i^\intercal)$ adat, akkor az új tárolópont $\beta = 1$ részt tölt le $d = k+1$ régi tárolóponttól. Feltehetjük, hogy az $1 \ldots k+1$ tárolópontoktól töltötte le a $z_i = a_i(\mathbf{o}_1\mathbf{p}_i^\intercal) + (\mathbf{o}_2\mathbf{p}_i^\intercal + \mathbf{o}_1\mathbf{r}_i^\intercal)$ részt, melyben az $a_i \in \mathbb{F}_q$-t később határozzuk meg.

Ebből az új tárolópont $(\sum_{i=1}^{k+1} b_i z_i, \sum_{i=1}^{k+1} c_i z_i)$-t tárol el, ahol b_i-t és c_i-t úgy adjuk meg, hogy $\sum_{i=1}^{k+1} b_i p_i^\intercal = \mathbf{0}$ és $\sum_{i=1}^{k+1} c_i p_i^\intercal = p_j^\intercal$. Ezeket meg tudjuk ilyennek választani, mert a G generátormátrix bármely, legalább k darab oszlopából alkotott mátrix rangja k.

$$\sum_{i=1}^{k+1} b_i z_i = \sum_{i=1}^{k+1} b_i \Big(a_i(\mathbf{o}_1 \mathbf{p}_i^\intercal) + (\mathbf{o}_2 \mathbf{p}_i^\intercal + \mathbf{o}_1 \mathbf{r}_i^\intercal) \Big) = \mathbf{o}_1 \sum_{i=1}^{k+1} b_i a_i \mathbf{p}_i^\intercal + \mathbf{o}_2 \sum_{i=1}^{k+1} b_i \mathbf{p}_i^\intercal + \mathbf{o}_1 \sum_{i=1}^{k+1} b_i \mathbf{r}_i^\intercal$$

Ez b_i megválasztása miatt:

$$\mathbf{o}_1 \sum_{i=1}^{k+1} b_i (a_i \mathbf{p}_i^\intercal + \mathbf{r}_i^\intercal)$$

Ekkor az a_i-ket úgy választjuk meg, hogy $\sum_{i=1}^{k+1} b_i (a_i \mathbf{p}_i^\intercal + \mathbf{r}_i^\intercal) = \mathbf{p}_j^\intercal$. Ezt megtehetjük, szintén a G generátormátrix bármely k darab oszlopának függetlensége miatt. Ezzel helyreállítottuk a meghibásodott tárolópont első részét.

A második rész helyreállítása hasonlóképpen történik:

$$\sum_{i=1}^{k+1} c_i z_i = \sum_{i=1}^{k+1} c_i \Big(a_i(\mathbf{o}_1 \mathbf{p}_i^\intercal) + (\mathbf{o}_2 \mathbf{p}_i^\intercal + \mathbf{o}_1 \mathbf{r}_i^\intercal) \Big) = \mathbf{o}_1 \sum_{i=1}^{k+1} c_i a_i \mathbf{p}_i^\intercal + \mathbf{o}_2 \sum_{i=1}^{k+1} c_i \mathbf{p}_i^\intercal + \mathbf{o}_1 \sum_{i=1}^{k+1} c_i \mathbf{r}_i^\intercal$$

Legyen az új $\mathbf{r}_j'^\intercal = \sum_{i=1}^{k+1} (c_i a_i \mathbf{p}_i^\intercal + c_i \mathbf{r}_i^\intercal)$! Ekkor c_i megválasztása miatt a második rész:

$$\mathbf{o}_1 \mathbf{r}_j'^\intercal + \mathbf{o}_2 \sum_{i=1}^{k+1} c_i \mathbf{p}_i^\intercal = \mathbf{o}_1 \mathbf{r}_j'^\intercal + \mathbf{o}_2 \mathbf{p}_j^\intercal$$

Mivel $\mathbf{r}_j'$ nem feltétlenül egyezik meg $\mathbf{r}_j$-vel, ezért ez funkcionális javítás.

3.4.2. MBR pont egy pontú hiba esetére

Egy pontú hiba esetén az MBR pont paraméterei adott (M, n, k, d)-re:

$$\alpha_{MBR} = \frac{M}{k} \frac{2d}{2d+1-k} = \gamma_{MBR}, \quad \beta_{MBR} = \frac{M}{k} \frac{2}{2d+1-k}$$

Legyen most $\beta_{MBR} = 1$! Ekkor $2M = kd + \frac{k-k^2}{2}$, így $\alpha_{MBR} = d$. Egy javítás során úgy lehet minél kisebb az egyik tárolóponttól letöltött részek mennyisége, ha minél több régi tárolóponttól tölt le részeket az új tárolópont, azaz $d = n - 1$.

Az előző példához hasonlóan $\mathbf{o} = (o_1, \ldots, o_M) \in \mathbb{F}_q^M$ a fájlunk. Mindegyik tárolópont $n-1$ kódolt részt tárol el. Az elkódoláshoz a tárolópontokat tekintsük úgy, mint egy n pontú teljes gráf pontjait. Az éleket számozzuk be 1-től $\frac{n(n-1)}{2}$-ig. Amelyik ponthoz csatlakozik a j. él, ott eltároljuk az $\mathbf{o}\mathbf{v}_j^\intercal$-t, ahol $\mathbf{v}_j \in \mathbb{F}_q^M$ és a $\mathbf{v}_j^\intercal$-okból képezett $M \times \frac{n(n-1)}{2}$ méretű V mátrix bármely $M \times M$-es részmátrixának rangja a lehető legnagyobb, azaz M.

Fájl visszaállítása ♦ Az adatgyűjtő bármely k tárolópontból összesen $k(n-1)$ részt tölt le, melyek közül $\frac{k(k-1)}{2}$ megegyezik, vagyis $k(n-1) - \frac{k(k-1)}{2}$ darab $\mathbf{ov}_j^\intercal$ alakú részből kell visszaállítanunk az $\mathbf{o}$-t, amely $M = k(n-1) - \frac{k^2-k}{2}$ hosszú. Ez megegyezik V egy négyzetes részmátrixának transzponáltjából képezett lineáris egyenletrendszerrel, és mivel ennek rangja M, ezért az egyenletrendszer megoldható.

Tárolópont visszaállítása ♦ Ha az i. tárolópont meghibásodik, akkor $n-1$ régi tárolóponttól tölt le egy-egy részt, ha az l. éllel vannak összekötve, akkor az $\mathbf{ov}_l^\intercal$-t, így visszaállítva az eredeti tárolópontot. Ez tehát egzakt javítás.

3.4.3. MSR pont több pontú hiba esetére

Vegyünk egy (n, k) paraméterű Reed-Solomon kódot az $\mathbb{F}_q$ test felett, ahol $q \geq n$. Tegyük fel, hogy az $M = tk$, ekkor az $\mathbf{o}$ fájlunk egy tk hosszú vektor. Ennek az elemeit átrendezhetjük egy $t \times k$-as O mátrixszá:

$$O = \begin{bmatrix} o_{11} & \dots & o_{1k} \\ \vdots & & \vdots \\ o_{t1} & \dots & o_{tk} \end{bmatrix}$$

Legyen $d = k$! Ekkor $\alpha_{MSR} = t$, $\beta_{MSR} = \beta'_{MSR} = 1$, $\gamma_{MSR} = k + t - 1$.

A Reed-Solomon kód G generátormátrixa $k \times n$-es méretű, az i. oszlopát jelölje $\mathbf{g}_i$. Az i. tárolópontban a t hosszú $O\mathbf{g}_i$ vektor elemei találhatók.

Fájl visszaállítása ♦ A Reed-Solomon kódolás miatt bármely k tárolópontból vissza tudjuk nyerni a fájlt.

Tárolópont visszaállítása ♦ Tárolópontok meghibásodása esetén az új t tárolópontot számozzuk 1-től t-ig. Az i. új tárolópont az $(\mathbf{o}_{i1}, \dots, \mathbf{o}_{ik})\mathbf{g}_{j_m}$ részt tölti le k tárolóponttól ($m = 1 \dots k$). Ebből már ki tudja számolni a $(\mathbf{o}_{i1}, \dots, \mathbf{o}_{ik})$ vektort. Így bármely j-re megadhatja a $(\mathbf{o}_{i1}, \dots, \mathbf{o}_{ik})\mathbf{g}_j$ vektort, és elküldheti a többi új tárolópontnak. Így a hiányzó részeket ki tudják pótolni az új tárolópontok egymás között. Így tehát ez is egzakt javítás.

4. fejezet

Alkalmazkodó javítás

Eddig a javítás során egy fix számú segítő pont segítségével állítottuk helyre az adatokat, viszont vannak olyan esetek, ahol érdemes lenne nem egy fix mennyiséget, hanem egy $D = \{d_1, d_2, \dots\}$ pozitív egész számokból álló halmazon belül szabadon választható d-t használni, ekkor a β_{d_j} jelöli a d_j darab segítő pont esetén használt sávszélességet. Ekkor β is egy halmaz. Bár ez a bővítés általánosabb az előző fejezetben lévő NDSS definíciójában megadottakkal, ez is egy hálózati adattároló rendszer. Egy ilyet megvalósító NDSS-t **alkalmazkodó NDSS**-nek nevezünk. Adott α-ra a 3.1 egyenlőtlenséget egyenlőséggel teljesítő, azaz minimális β_{d_j}-t $\beta^*_{d_j}(\alpha)$-val jelöljük.

Ebben a fejezetben $t = 1$, és az [1] cikket dolgozzuk fel.

21. Tétel (Aggarwal, Tian, Vaishampayan; [1]). *Egy $NDSS(M, n, k, D, 1, \alpha, \beta, 0)$-beli gráfban a c szerinti minimális vágás értéke legalább*

$$\sum_{i=0}^{k-1} \min(\alpha, \min_{d_j \in D}(d_j - i)\beta_{d_j}),$$

és ez az érték fel is vétetik.

Bizonyítás. Az előző fejezetben taglaltakhoz hasonlóan építjük fel az adatfolyam gráfját. Most az új tárolópontok bármely $d \in D$ számú régi aktív ponthoz kapcsolódhatnak. Legyen $e_i = \arg\min_{d_j \in D}(d_j - i)\beta_{d_j}$. Egy speciális esetet nézünk, ahol az eredeti n tárolópont számozva van 1-től n-ig, a k új pedig tovább $n+1$-től $n+k$-ig. Az $n+i$. tárolópont az $n+i-e_{i-1}, \dots, n+i-1$ tárolópontokhoz kapcsolódik. Az adatgyűjtő az utolsó k ponthoz kapcsolódik, és ha $\alpha \leq (e_i - 1)\beta_{e_i}$, akkor az x^{n+i}_{ki} pontot az $\overline{U}$-ba rakjuk, egyébként az U-ba. Az x^{n+1}_{be} mindig az U-ba kerül. Az előző tételekhez hasonlóan a vágás minimális vágás ebben a folyamban, és egyelő a tételben lévő szummával. Minden más, ebbe a családba tartozó gráfon lévő folyam esetén a minimális vágás nagyobb vagy egyenlő ennél. □

Az előző fejezethez hasonlóan szükséges, hogy a fenti összegnél kisebb vagy egyenlő legyen M, azaz:

$$\sum_{i=0}^{k-1} \min(\alpha, \min_{d_j \in D}(d_j - i)\beta_{d_j}) \geq M. \tag{4.1}$$

A továbbiakban a D halmaz l elemű, és $k \leq d_l < \ldots < d_1 < n$. A fenti tétel segítségével keresünk korlátokat α-hoz.

22. Tétel (Aggarwal, Tian, Vaishampayan; [1]). *Adott n, k, M, α és $|D| > 1$ esetén $(\alpha, \beta^*_{d_1}(\alpha), \beta^*_{d_2}(\alpha), \ldots, \beta^*_{d_l}(\alpha))$ akkor és csak akkor teljesíti a 4.1. egyenlőtlenséget, ha $k = 1$ vagy*

$$\alpha \leq \frac{(d_1 - k + 2)M}{(d_1 - k + 2)k - 1}.$$

Bizonyítás. Vegyük először a D első két elemét, azaz $\{d_1, d_2\} \subset D$-t, ahol $d_1 > d_2$. Mivel az $(\alpha, \beta^*_{d_1}(\alpha), \infty)$ teljesíti a 4.1 egyenlőtlenséget, így van olyan minimális $\beta_{d_2}(\alpha)$, amivel $(\alpha, \beta^*_{d_1}(\alpha), \beta_{d_2}(\alpha))$ teljesíti az egyenlőtlenséget. Ezt a minimális $\beta_{d_2}(\alpha)$-t jelöljük $\widetilde{\beta_{d_2}}(\alpha)$-val. Mivel egyrészt a $\beta^*_{d_1}(\alpha)$ egyenlőséggel teljesíti a 3.1. egyenlőlenséget, másrészt $\widetilde{\beta_{d_2}}(\alpha)$ is egyenlőséggel teljesíti a 4.1. egyenlőtlenséget, ezért a következő egyenleteket tudjuk:

$$\sum_{i=0}^{k-1} \min(\alpha, (d_1 - i)\beta^*_{d_1}(\alpha)) = M$$

$$\sum_{i=0}^{k-1} \min(\alpha, (d_1 - i)\beta^*_{d_1}(\alpha), (d_2 - i)\widetilde{\beta_{d_2}}(\alpha)) = M$$

Mivel a szumma belsejében lévő minimum a második kifejezésben legfeljebb akkora, mint az elsőben, ezért minden $0 \leq i \leq k-1$-re:

$$(d_2 - i)\widetilde{\beta_{d_2}}(\alpha) \geq \min(\alpha, (d_1 - i)\beta^*_{d_1}(\alpha))$$

Mivel $\widetilde{\beta_{d_2}}(\alpha)$ minimális, ezért tudjuk, hogy ez is egyenlőséggel teljesül valamely i-re, vagyis:

$$\widetilde{\beta_{d_2}}(\alpha) = \max_{i=0,\ldots,k-1} \min\left(\frac{\alpha}{d_2 - i}, \frac{d_1 - i}{d_2 - i}\beta^*_{d_1}(\alpha)\right)$$

Mivel $d_1 > d_2$, ezért mindkét kifejezés értéke nő, ha i nő. Tehát a maximum $i = k-1$-nél van.

$$\widetilde{\beta_{d_2}}(\alpha) = \min\left(\frac{\alpha}{d_2 - k + 1}, \frac{d_1 - k + 1}{d_2 - k + 1}\beta^*_{d_1}(\alpha)\right)$$

Megjegyezzük, hogy $\beta^*_{d_1}(\alpha)$ optimalitása miatt $\alpha \geq (d_1 - k + 1)\beta^*_{d_1}(\alpha)$, ezért

$$\widetilde{\beta_{d_2}}(\alpha) = \frac{d_1 - k + 1}{d_2 - k + 1}\beta^*_{d_1}(\alpha)$$

23. Lemma. *$\beta^*_{d_2}(\alpha) = \widetilde{\beta_{d_2}}(\alpha)$ pontosan akkor, ha $\widetilde{\beta_{d_2}}(\alpha)$ egyenlőséggel teljesíti a 3.1. egyenlőtlenséget $d = d_2$-re, azaz:*

$$\sum_{i=0}^{k-1} \min(\alpha, (d_2 - i)\frac{d_1 - k + 1}{d_2 - k + 1}\beta^*_{d_1}(\alpha)) = M.$$

A lemma a $\beta^*_{d_2}(\alpha)$ definíciójából következik. Most felső korlátot keresünk α-ra, úgy, hogy a lemmát feltesszük, azaz $\beta^*_{d_2}(\alpha) = \widetilde{\beta_{d_2}}(\alpha) = \frac{d_1-k+1}{d_2-k+1}\beta^*_{d_1}(\alpha)$, és teljesül a fenti egyenlőség.

Mivel $\beta^*_{d_1}(\alpha)$ egyenlőséggel teljesíti 3.1-t $d = d_1$-re, és $(d_2 - i)\frac{d_1-k+1}{d_2-k+1} \geq (d_1 - i)$ minden $0 \leq i \leq k-1$-re, ezért a két baloldalt álló kifejezésnek tagonként is meg kell egyezniük minden $0 \leq i \leq k-1$-re:

$$\min(\alpha, (d_2 - i)\frac{d_1 - k + 1}{d_2 - k + 1}\beta^*_{d_1}(\alpha)) = \min(\alpha, (d_1 - i)\beta^*_{d_1}(\alpha))$$

Mivel $i = k-1$ esetén a két kifejezés ugyanaz, ezért $k = 1$-re mindig teljesül a lemma. Tehát $k > 1$ esetén már csak a $0 \leq i \leq k-2$-re kell teljesülnie.

Mivel ezen indexek esetén $(d_2 - i)\frac{d_1-k+1}{d_2-k+1} > (d_1 - i)$, ezért csak akkor lehet egyenlőség, ha a minimum mindkét oldalon az első helyen vétetik fel. Mivel a $(d_1 - i)$ kifejezés értéke csökken, ahogy i nő, ezért ha azt akarjuk, hogy $\alpha \leq (d_1 - i)\beta^*_{d_1}(\alpha)$ legyen minden $0 \leq i \leq k-2$-re, akkor α-nak kisebbnek vagy egyenlőnek kell lennie a legkisebb jobboldali kifejezésnél is, vagyis a legnagyobb indexűnél is, tehát

$$\alpha \leq (d_1 - k + 2)\beta^*_{d_1}(\alpha). \tag{4.2}$$

Most megállapítjuk $\beta^*_{d_1}(\alpha)$ értékét. Tudjuk, hogy ha $\alpha = M/k$, akkor a

$$\sum_{i=0}^{k-1} \min\{(d_1 - i)\beta^*_{d_1}(\alpha), \alpha\} = M$$

egyenlőség bal oldalán lévő szumma minden tagja M/k értékű, és ekkor egy MSR pont az $(\alpha, \beta^*_{d_1}(\alpha))$, vagyis $\beta^*_{d_1}(\alpha) = \frac{M}{k(d_1-k+1)}$, ekkor $\alpha \leq \frac{M}{k}$, vagyis teljesül a tétel.

Mivel előzőleg már megállapítottk, hogy az $0 \leq i \leq k-2$ indexekre mindenképpen a második helyen vétetik fel a minimum, ezért csak a $k-1$. indexnél vétethetik fel a második helyen. Ezért ha $\alpha > M/k$, akkor:

$$\sum_{i=0}^{k-2} \alpha + (d_1 - k + 1)\beta^*_{d_1}(\alpha) = M$$

$$\beta^*_{d_1}(\alpha) = \frac{M - (k-1)\alpha}{d_1 - k + 1}$$

Így a 4.2 egyenlőtlenségbe behelyettesítve:

$$\alpha \leq \frac{(d_1-k+2)(M-(k-1)\alpha)}{(d_1-k+1)} = \frac{(d_1-k+2)M}{(d_1-k+1)} - \frac{(d_1-k+2)(k-1)}{(d_1-k+1)}\alpha$$

$$0 \leq \frac{(d_1-k+2)M}{d_1-k+1} - \frac{(d_1-k+2)(k-1)+(d_1-k+1)}{d_1-k+1}\alpha$$

$$\alpha \leq \frac{(d_1-k+2)M}{(d_1-k+2)(k-1)+(d_1-k+1)} = \frac{(d_1-k+2)M}{(d_1-k+2)k-1}$$

Ezzel a tételt a D első két elemére bizonyítottuk. Ha hozzáveszünk még egy d_i-t, annak bizonyítása, hogy mely esetben lesz a $\beta^*_{d_i}(\alpha)$ megfelelő, teljesen analóg módon történik. □

A fenti tétel szemléletesen azt jelenti, hogy ha $\alpha \leq \alpha_0$ és nem alkalmazkodó NDSS-ként optimális volt a rendszer, akkor alkalmazkodó javítást használva ugyanúgy optimális marad.

24. Következmény (Cadambe, Jafar, Maleki, Ramchandran, Suh; [2]). *Adott n, k és M értékekre az MSR pontok minden d értékre egyszerre is elérhetők, tehát az $(\frac{M}{k}, \beta^*_k(\frac{M}{k}), \beta^*_{k+1}(\frac{M}{k}), \ldots, \beta^*_{n-1}(\frac{M}{k}))$ teljesíti a 4.1. egyenlőtlenséget.*

5. fejezet

Heterogén adattároló rendszerek

Az eddigiekben minden tárolópont kapacitása egységes, azaz **homogén** volt. Viszont ez a gyakorlatban nem mindig kivitelezhető, ezért van szükség a **heterogén** adattároló rendszerek fogalmának bevezetésére, a [4] cikk alapján. Fontos, hogy a továbbiakban $t = 1$, azaz egy tárolópont meghibásodása esetén javítunk.

Ebben az esetben tehát ha a tárolópontokat $x_1, \ldots, x_n$-nel jelöljük, akkor az i. tárolópont kapacitása α_i. Vagyis az $\boldsymbol{\alpha}$ most nem egy szám, hanem egy n hosszú vektor. Feltesszük, hogy a tárolópontok a kapacitás szerinti növekvő sorrendbe vannak rendezve, azaz $\alpha_1 \leq \alpha_2 \leq \ldots \leq \alpha_n$. Mivel ekkor nem egyformák a tárolópontok kapacitásai, a tárolópontok közötti kommunikáció mérete sem lehet feltétlenül egységes. Tehát $\beta_{i,j,S}$-sel jelöljük azt, hogy mekkora adatot tölt le az x_j tárolópontot helyettesítő új pont az x_i ponttól, abban az esetben, ha az új pont az S halmazban lévő indexű pontoktól tölt le adatot ($|S| = d$). Speciális eset az, amikor ez a letöltött adatmennyiség nem függ a meghibásodott tárolóponttól és a segítő tárolópontok halmazától, csak attól a tárolóponttól, amitől letölti az adatot. Ezt β_i-vel jelöljük, vagyis $\beta_{i,j,S} = \beta_i$ minden j-re és S-re. Ekkor a $\boldsymbol{\beta}$ is egy n hosszú vektor (egyébként egy $nd\binom{n-1}{d}$ hosszú vektor). Ha minden $\beta_i = \beta$, akkor **szimmetrikus javításról** beszélünk.

Egy meghibásodás esetén a segítő tárolópontokat $\binom{n-1}{d}$-féleképpen választhatjuk ki. Tehát az x_j csúcs meghibásodása esetén az átlagos sávszélesség:

$$\gamma_j = \sum_{S: j \notin S, |S|=d} \sum_{i \in S} \beta_{i,j,S} / \binom{n-1}{d}.$$

Vagyis $\boldsymbol{\gamma}$ is egy n hosszú vektor.

Tehát minden tárolópontot egyszerre nézve az átlagos sávszélesség $\overline{\gamma} = \sum_{j=1}^{n} \gamma_j / n$, az átlagos kapacitás pedig $\overline{\alpha} = \sum_{j=1}^{n} \alpha_j / n$.

A következőkben ezekhez az adott α, β, γ paraméterekhez keressük az NDSS C kapacitását, vagyis azt az információmennyiséget, amit bármelyik k tárolópontból letöltött adatok után vissza tudunk nyerni. Ez a homogén rendszer esetében a 3.1. egyenlőtlenség alapján adott α és γ esetén

$$\sum_{i=0}^{k-1} \min\{(d-i)\gamma/d, \alpha\}, \tag{5.1}$$

melyet $C_{ho}(\alpha, \gamma)$-val jelölünk a továbbiakban.

25. Tétel (Ernvall, Rouayheb, Hollanti, Poor; [4])**.** *Egy heterogén NDSS C kapacitását felülről becsüli*

$$\sum_{i=0}^{k-1} \min\{(d-i)\overline{\gamma}/d, \overline{\alpha}\} = C_{ho}(\overline{\alpha}, \overline{\gamma}),$$

ahol $\overline{\alpha}$ az NDSS átlagos kapacitása, $\overline{\gamma}$ pedig az átlagos sávszélessége.

Bizonyítás. A bizonyításhoz a heterogén NDSS-ből homogén NDSS-t generálunk úgy, hogy kombinálunk NDSS-eket. Ezt a kombinálást a következő módon tehetjük meg: legyen $NDSS_1$ és $NDSS_2$ két NDSS ugyanannyi tárolóponttal, melynek tárolópontjait $x_1^1, \ldots, x_n^1$-nel és $x_1^2, \ldots, x_n^2$-nel jelöljük. Az új $NDSS_1 + NDSS_2 = NDSS_c$-vel jelölt rendszer tárolópontjai pedig $x_1^c, \ldots, x_n^c$ lesznek. Az x_i^c tárolópont kapacitása $\alpha_i^c = \alpha_i^1 + \alpha_i^2$, az új $\boldsymbol{\beta}^c$ vektor elemei pedig $\beta_{i,j,S}^c = \beta_{i,j,S}^1 + \beta_{i,j,S}^2$.

A heterogén NDSS-t jelöljük $NDSS$-sel! Vesszük ennek a permutációit úgy, hogy a tárolópontok sorrendjét változtatjuk meg. Tehát minden $\sigma : \{1, \ldots, n\} \mapsto \{1, \ldots, n\}$, $\sigma \in P_n$ permutációra $NDSS_\sigma$ azt a heterogén NDSS-t jelöli, melynek i. tárolópontja az eredeti NDSS $\sigma(i)$. tárolópontja, illetve $\sigma(S) = \{\sigma(i) : i \in S$. Ezekre az NDSS-ekre nyilván nem vonatkozik az α_i szerinti sorrend. Ezeket a permutált NDSS-eket kombináljuk: $\sum_{\sigma \in P_n} NDSS_\sigma = NDSS_h$. Ez az NDSS homogén, mivel a k. tárolópontjának a kapacitása

$$\sum_{\sigma \in P_n} \alpha_{\sigma(k)} = (n-1)! \sum_{i=1}^{n} \alpha_i = \alpha_h,$$

és a k. és l. tárolópont közötti adatforgalom adott S^* esetén:

$$\beta_{i,j,S^*}^h \sum_{\sigma \in P_n} \beta_{\sigma(k),\sigma(l),\sigma(S*)} = (n-d-1)!\,(d-1)! \sum_{j=1}^{n} \sum_{i=1, i \neq j}^{n} \sum_{|S|=d, i \in S, j \notin S} \beta_{i,j,S} = \beta_h,$$

melyeket $\overline{\alpha}$-val és $\overline{\gamma}$-val kifejezve:

$$\alpha_h = n!\,\overline{\alpha}$$

$$\beta_h = (n-d-1)!\,(d-1)! \sum_{j=1}^{n} \binom{n-1}{d} \overline{\gamma_j} = \frac{(n-1)!}{d} \sum_{j=1}^{n} \overline{\gamma_j} = n!\,\frac{\overline{\gamma}}{d}$$

Tehát az $NDSS_h$ kapacitása az 5.1. egyenlet alapján:

$$C_h = \sum_{i=0}^{k-1} \min\left\{(d-i)n!\,\overline{\gamma}/d, n!\,\overline{\alpha}\right\} = n! \sum_{i=0}^{k-1} \min\left\{(d-i)\overline{\gamma}/d, \overline{\alpha}\right\} = n!\,C_{ho}(\overline{\alpha}, \overline{\gamma})$$

Mivel az eredeti inhomogén NDSS kapacitása C volt, és $n!$ ilyen kapacitású NDSS-t kombináltunk, ezért az $NDSS_h$ kapacitása legalább $n!\,C$. Tehát:

$$n!\,C \leq C_h = n!\,C_{ho}(\overline{\alpha}, \overline{\gamma})$$

$$C \leq C_{ho}(\overline{\alpha}, \overline{\gamma})$$

Ezzel bebizonyítottuk a tételt. □

Az alábbi következmény azt mondja ki, hogy a 3. fejezetben megismert NDSS-nél a tárolópontok közötti kommunikáció homogén mivolta optimális.

26. Következmény. *Egy homogén NDSS-ben a szimmetrikus javítás maximalizálja C-t.*

Bizonyítás. Tekintsük a heterogén NDSS-ek azt a speciális esetét, amikor minden tárolópont kapacitása α, vagyis homogén, de a $\boldsymbol{\beta}$ nem feltétlenül egységes. Ekkor az előző tétel alapján a teljes kapacitást felülről becsüli $C_{ho}(\overline{\alpha}, \overline{\gamma})$, és mivel $\overline{\alpha} = \alpha$, ezért a következményt kaptuk. □

A következőkben csak azzal a speciális esettel foglalkozunk, amikor $\beta_{i,j,S} = \beta_i$ minden j-re és S-re. A β_i-ket nagyság szerinti növekvő sorrendbe helyezzük, azaz: $\beta_1^* \leq \beta_2^* \leq \ldots \leq \beta_n^*$.

27. Tétel (Ernvall, Rouayheb, Hollanti, Poor; [4])**.** *Egy heterogén NDSS C kapacitását egyrészt alulról becsüli a következő kifejezés:*

$$C_{\min} = \sum_{i=1}^{k} \min\left\{\alpha_i, \beta_1^* + \beta_2^* + \ldots + \beta_{d-i+1}^*\right\} = \min_{l=0,\ldots,k} \left\{\sum_{i=1}^{l} \alpha_i + \sum_{i=l+1}^{k} \sum_{j=1}^{d-i+1} \beta_j^*\right\}$$

Másrészt felülről becsüli a következő kifejezés:

$$C_{\max} = \sum_{i=1}^{k} \min\left\{\alpha_i, \beta_{i+1}^* + \beta_{i+2}^* + \ldots + \beta_{d+1}^*\right\} = \min_{l=0,\ldots,k} \left\{\sum_{i=1}^{l} \alpha_i + \sum_{i=l+1}^{k} \sum_{j=i+1}^{d+1} \beta_j^*\right\}$$

A két tétel közül egyik sem erősebb a másiknál, vannak olyan esetek, ahol a 25. tétel ad erősebb korlátot, illetve olyanok is, ahol a 27. tétel.

Vegyük észre, hogy ha az NDSS homogén, vagy minden $\beta_i = \beta$, esetleg minden $\alpha_i \leq \beta_1^*$, akkor $C_{\min} = C_{\max}$.

A tétel bizonyításához tekintsük először a következő állítást:

28. Állítás. *Egy heterogén NDSS C kapacitása:*

$$C = \min_{(f_1,\ldots,f_k);f_i\neq f_j \text{ ha } i\neq j} \sum_{i=1}^{k} \min(\alpha_{f_i}, \min\{\sum_{j\in S_i} \beta_j : |S_i| = d+1-i; S_i \cap (f_1,\ldots,f_i) = \emptyset\})$$

Bizonyítás. Az állítás a 11. tétel általánosítása, a bizonyítása hasonlóképp, élkapacitásos irányított gráfok segítségével történik. Itt is egy olyan speciális esetet nézünk, ahol k darab javítás történt, és a segítő tárolópontok közül a lehető legtöbb már új tárolópont. Az $(f_1, \ldots, f_k)$ indexhalmaz jelöli a k darab meghibásodott tárolópontot, az S_i pedig az új x_{f_i} tárolópont régi segítő pontjainak halmazát, amely $d + 1 - i$ elemű, mivel $i - 1$ darab segítő tárolópont az újak, azaz f_j indexűek közül kerül ki.

Annak bizonyítása, hogy ez minimális vágás, illetve hogy ennél kisebb vágás egyetlen másmilyen $NDSS(M, n, k, d, 1, \boldsymbol{\alpha}, \boldsymbol{\beta}, 0)$-beli gráfban sincs, hasonlóképpen történik, mint a 11. tételben. □

A pontos érték kiszámítása nagy rendszereknél lassú. Ezért adunk rá korlátokat a fenti tételben, melynek bizonyítása a következő:

Bizonyítás. A segédállításban szereplő képlet első minimumát adó $(f_1, \ldots, f_k)$ indexhalmazt jelöljük $(f'_1, \ldots, f'_k)$-vel. Ekkor:

$$C = \sum_{i=1}^{k} \min(\alpha_{f'_i}, \min\{\sum_{j\in S_i} \beta_j : |S_i| = d + 1 - i; S_i \cap (f'_1, \ldots, f'_i) = \emptyset\}),$$

amelyben a belső minimum alulról becsülhető a $d - i + 1$ legkisebb β_i összegével, vagyis:

$$C \geq \sum_{i=1}^{k} \min(\alpha_{f'_i}, \sum_{j=1}^{d-i+1} \beta_j^*)$$

Jelöljük l^*-gal azt a számot, ahány helyen a szummában a minimum az $\alpha_{f'_i}$-n vétetik fel. Ekkor ezeket az $\alpha_{f'_i}$-ket alulról tudjuk becsülni az l^* darab legkisebb α_i-vel. A maradék $k - l^*$ tagot pedig a legkisebb $\sum_{j=1}^{d-i+1} \beta_j^*$-okkal, amelyek mérete csökken, ha az i nő, tehát a $k - l^*$ legnagyobb indexű tagot kell venni.

$$C \geq \sum_{i=1}^{l^*} \alpha_i + \sum_{i=l^*+1}^{k} \sum_{j=1}^{d-i+1} \beta_j^* = \min_{l=0,\ldots,k} \{\sum_{i=1}^{l} \alpha_i + \sum_{i=l+1}^{k} \sum_{j=1}^{d-i+1} \beta_j^*\}$$

Ezzel megvan az alsó becslés. A felső becslés hasonlóképp adódik, viszont itt nem a minimumot adó $(f'_1, \ldots, f'_k)$ indexhalmazt vesszük, hanem az $(1, \ldots, k)$ indexhalmazt, ezzel a kifejezés legalább C.

$$C \leq \sum_{i=1}^{k} \min(\alpha_i, \min\{\sum_{j \in S_i} \beta_j : |S_i| = d + 1 - i; S_i \cap (1, \ldots, i) = \emptyset\})$$

A belső minimumban szereplő feltételek $d+1$ darab különböző indexre vonatkoznak, és mivel a β_i^*-k sorrendbe vannak rendezve, ezért tudjuk, hogy a minimumban csak a $d+1$ darab első index szerepelhet. Ezért úgy tudjuk felülről becsülni, hogy a $d-i+1$ darab legnagyobb, maximum $d+1$ indexű β_i-t vesszük.

$$C \leq \sum_{i=1}^{k} \min(\alpha_i, \sum_{j=i+1}^{d+1} \beta_j^*)$$

Legyen l^* azt a szám, ahány helyen a szummában a minimum az α_i-n vétetik fel. Mivel az α_i-k növekvő sorrendben vannak, a $C \sum \beta_j^*$-k pedig csökkenő sorrendben, ezért ez egyben azt is jelenti, hogy a minimum az első helyen az első l^* indexnél vétetik fel. Vagyis ekkor a következő átírással nem változtatunk az összegen:

$$\sum_{i=1}^{l^*} \alpha_i + \sum_{i=l^*+1}^{k} \sum_{j=i+1}^{d+1} \beta_j^* = \min_{l=0,\ldots,k} \{\sum_{i=1}^{l} \alpha_i + \sum_{i=l+1}^{k} \sum_{j=i+1}^{d+1} \beta_j^*\}$$

Ezzel bizonyítottuk a tételt. □

6. fejezet

Rétegelt videók tárolása

Az eddigiekben a meghibásodás esetén fellépő javításokra adtunk módszereket, most pedig azzal foglalkozunk, hogy nagy terheltség, vagyis minél több felhasználó esetén is elérhetőek maradjanak a fájlok. Ehhez más modellt érdemes használnunk, mint a javításoknál. Lényeges különbség, hogy nem hibásodnak meg a tárolópontok, hanem azt vizsgáljuk, hogy hány felhasználó igényeit tudja kiszolgálni a rendszer. Az [5] cikket dolgozzuk fel.

6.1. Modellezés

A fájlunk most egy videó, amit több, összesen L különféle felbontásban tudnak megtekinteni a felhasználók. Ahhoz, hogy ezt kivitelezhessük, a videó L rétegből áll, egy l_1 **alaprétegből**, és $L-1$ darab $l_2, \ldots, l_L$ **finomító rétegből**, ezt rétegelt videónak nevezzük. A videó i. méretű felbontásban való megjelenítéséhez az első i rétegre van szükségünk. Egy réteget egy w hosszú $\mathbb{F}_q^w$-beli vektorként modellezünk.

Egy felhasználó $p \in \{1, \ldots, L\}$ réteget kérhet, ezt **p-típusú kérésnek** nevezzük, az ilyen felhasználót pedig p-típusú felhasználónak. A p-típusú kérések beérkezését egy λ_p paraméterű Poisson-folyamattal modellezzük. Egy felhasználó kilépését a rendszerből $i_p\mu$ paraméterű Poisson-folyamattal modellezzük, ahol i_p a rendszerben lévő p-típusú felhasználók száma.

A beérkezett kéréseket egy szerver dolgozza fel, amely n tárolóponthoz kapcsolódik. Feltesszük, hogy a szerver úgy osztja ki az elérhető eltárolt rétegeket a felhasználóknak, hogy ha új felhasználók érkeznek, a rétegek esetleges újra kiosztása nem lassítja a régi felhasználók kiszolgálását.

Nevezzük **p-típusú tárolópontnak** azt a tárolópontot, amely az $l_1, \ldots, l_p$ rétegek közül tárol rétegeket, és az l_p biztosan szerepel a tárolt rétegek között.

Az i-típusú tárolópontokon elhelyezett - bármilyen típusú - rétegek számát jelöljük m_i-vel. Jelölje továbbá B azon rétegek számát, amennyit egyszerre egy tárolópontból lehet letölteni. Ekkor B végessége miatt az egyszerre rendszerben lévő felhasználók száma véges. Ha egy olyan p-típusú kérés érkezik be, amit már nem tud kiszolgálni a rendszer, elutasítja. Ha p-től függetlenül egyik kérést sem tudja fogadni, akkor **túlterhelt állapotba** kerül. Ennek a valószínűsége a **túlterheltségi valószínűség** (P_s), melyet minimalizálni szeretnénk.

A felhasználók egy halmaza **megengedett**, ha egyik kérést sem utasítja el a szerver.

Három különböző adattárolási módszert nézünk meg. Az elsőben vannak olyan tárolópontok, ahol csak az alapréteget tároljuk, majd olyanok, ahol csak az alapréteget és az első finomító réteget, és így tovább. Tehát nem használjuk ki, hogy egy videó rétegekből áll, hanem a különböző felbontású videókat teljesen különböző fájlként kezeljük. Ezt **klasszikus** módszernek nevezzük.

A másodikban rétegelt videókat tárolunk el, tehát a rétegeket külön tárolópontokon tároljuk, de a rétegeket nem kódoljuk, ezt nevezzük **kódolatlan többfelbontású** (URS) módszernek. A harmadikban pedig a rétegeket lineáris kóddal elkódolva tároljuk, ez a **kódolt többfelbontású** (CRS) módszer.

Ahhoz, hogy megértsük a három módszer közötti különbségeket, nézzünk egy példát $L = 2$-re! Ekkor az első módszer szerint m_1 réteg van az 1-típusú tárolópontokban, és $\left\lfloor \frac{m_2}{2} \right\rfloor$ darab l_1 és ugyanennyi l_2 a 2-típusú tárolópontokban.

A második szerint szintén m_1 darab l_1 van az 1-típusú tárolópontokban, de m_2 darab l_2 a 2-típusú tárolópontokban.

A harmadik módszerben pedig m_1 darab l_1 alapréteg van az 1-típusú tárolópontokban, m_2 darab réteg pedig l_1 és l_2 valamilyen lineáris kombinációja, melyeket a 2-típusú tárolópontok tárolnak. Ezek a lineáris kombinációk nem definíció szerinti rétegek, de ahhoz, hogy összehasonlíthassuk a kódolatlan módszerekkel, továbbra is rétegként hivatkozunk rájuk. Ebben a harmadik esetben l_p^k-val jelöljük az első p rétegből kódolt réteg k. verzióját, amit a $\sum_{i=1}^{p} \alpha_{i,k} l_i$ képlettel kapunk meg, ahol $\alpha_{i,k} \in \mathbb{F}_q$ egy véges testbeli együttható. Ezt a réteget egy p-típusú rétegnek vesszük.

6.2. Megengedettséghez szükséges feltételek

Ebben a részben a megengedett felhasználó-halmazhoz keresünk szükséges és elégséges, illetve szükséges feltételeket, melyek a következő rész numerikus eredményeinek eléréséhez szükségesek.

Jelölje $y_j^p \in \mathbb{N}$ a p-típusú rétegek számát, amit a j. felhasználó beolvas. Egy p-típusú felhasználó $(i \leq p)$-típusú rétegeket olvashat be. Összesen N felhasználó van a rendszerben, és $N = \sum_{p=1}^{L} i_p$. Legyen P egy függvény, amely egy felhasználó indexét leképezi a kérésének típusára, azaz $\{1, \ldots, N\}$-ről $\{1, \ldots, L\}$-re képez. T_j pedig jelölje azt a $p \times p$ méretű együtthatómátrixot, amelynek (a, b) eleme a j. (p-típusú) felhasználó által letöltött a. kódolt rétegben az l_b-hez tartozó α_b együttható.

29. Definíció. Ahhoz, hogy a felhasználók megengedettek legyenek, a következő egyenleteknek és egyenlőtlenségeknek kell teljesülniük az $y_j^p \in \mathbb{N}$ változókra:

$$\sum_{j=1}^{N} y_j^p \leq B m_p \qquad (\forall p \in \{1, \ldots, L\}) \tag{6.1}$$

Vagyis bármely p-típusú réteget tároló pont esetén van megfelelő méretű sávszélesség, hogy az összes felhasználó, aki ilyen réteget akar beolvasni, beolvashassa azt.

$$\sum_{z=1}^{P(j)} y_j^z = P(j) \qquad (\forall j \in \{1, \ldots, N\}) \tag{6.2}$$

Azaz minden p-típusú felhasználó pontosan p réteget olvas be.

$$\operatorname{rang}(T_j) = P(j) \qquad (\forall j \in \{1, \ldots, N\}) \tag{6.3}$$

Tehát minden felhasználó vissza tudja kódolni a p réteget, amit kért. Ezeket az egyenleteket és egyenlőtlenségeket nevezzük **pontos formulának**.

Viszont ezek a feltételek a sok változó és egyenlet miatt numerikus modellezéshez nehezen használhatók. Ezért vezette be Ferner, Wang és Médard [5] a következő közelítő formulát:

30. Definíció. A felhasználók halmaza megfelelő a **közelítő formula** szerint, ha az $i_p \geq 0$ változókra:

$$\sum_{p=1}^{L} p i_p \leq B \sum_{p=1}^{L} m_p \tag{6.4}$$

Azaz amennyi réteget az összes felhasználónak le kell töltenie összesen, kevesebb, mint amennyi elérhető.

$$i_p \leq B m_p \qquad (\forall p \in \{1, \ldots, L\}) \tag{6.5}$$

Vagyis a p-típusú felhasználók kevesebben vannak, mint amennyiszer le lehet tölteni az l_p réteget tároló pontról egy réteget.

31. Állítás. *Ha a felhasználók halmazára teljesül a pontos formula, akkor a közelítő formula is.*

Bizonyítás. Mivel $\sum_{p=1}^{L}\sum_{j=1}^{N} y_j^p$, azaz az összes felhasználó által beolvasott összes réteg száma egyenlő $\sum_{p=1}^{L} pi_p$-vel, ezért ha a 6.1-et összegezzük minden p-re, a 6.4-gyel egyezik meg.

Mivel minden p-típusú felhasználónak le kell töltenie legalább egy p-típusú réteget, ezért $i_p \leq \sum_{j=1}^{N} y_j^p$. Így ha a 6.1 teljesül, a 6.5 is. □

32. Tétel. *Ha $L > 2$, akkor van olyan felhasználó-halmaz, hogy a közelítő formula teljesül rá, de a pontos formula nem.*

Bizonyítás. Vegyük azt az esetet, amikor $B = 1$, azaz minden tárolópont egy réteget tud átadni. Legyen $m_1 = m_L = L$ és $m_2 = \cdots = m_{L-1} = 0$. Ha veszünk két L-típusú felhasználót, akik közül az egyik 1 darab 1-típusú kódolt réteget és $L-1$ darab L-típusú kódolt réteget kap, a másik pedig $L-1$ darab 1-típusú kódolt réteget és 1 darab L-típusú kódolt réteget kap. Látható, hogy a közelítő formulát teljesíti ez a leosztás, de a pontosat nem, hiszen a második felhasználó nem tudja visszakódolni a rétegeket. □

33. Tétel. *Ha $L = 2$, akkor nincs olyan felhasználó-halmaz, hogy a közelítő formula teljesül rá, de a pontos formula nem.*

Bizonyítás. A felhasználók száma szerinti teljes indukcióval bizonyítjuk, hogy ha a közelítő formula teljesül, akkor a pontos is. $N = 1$ esetén ez nyilván teljesül 1-típusú és 2-típusú felhasználó esetén is.

Adott N megengedett felhasználóra feltesszük, ha a közelítő formula teljesül, akkor a pontos is. Legyen egy új, $N+1$. felhasználó. Tegyük fel, hogy az új, $N+1$ elemű felhasználóhalmazra a közelítő formulák teljesülnek. Ha 1-típusú felhasználó, akkor a 6.5 miatt le tudja tölteni az alapréteget, $y_{N+1}^1 = 1$ és $y_{N+1}^2 = 0$, ezzel a pontos formulák teljesülnek.

Ha 2-típusú a felhasználó, akkor vagy egy alapréteget és egy kódolt réteget, vagy két kódolt réteget akar letölteni. Az első esetben a közelítő formulák miatt kell lennie elég rendelkezésre álló sávszélességnek, és mivel $y_{N+1}^1 = 1$ és $y_{N+1}^2 = 1$, ezért teljesülnek a pontos formulák.

Ha két kódolt réteget akar letölteni, akkor a 6.5 miatt van két szabad kódolt réteg letöltéséhez sávszélesség, így teljesülnek a pontos formulák. □

Tehát $L = 2$-re alkalmazhatjuk a közelítő formulát.

6.3. Numerikus eredmények

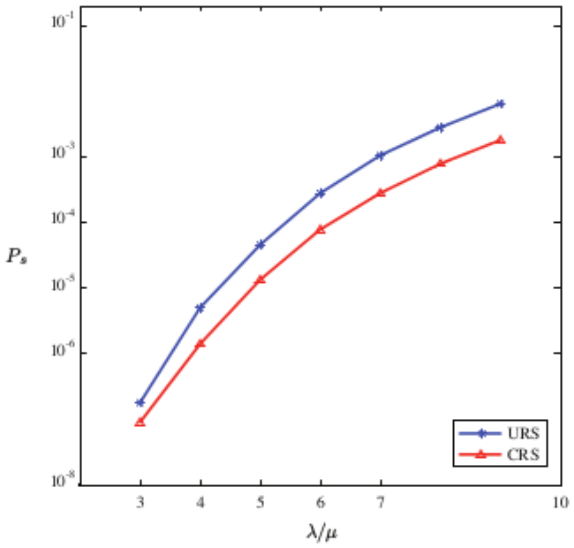
A következőkben numerikus számítások útján hasonlítjuk össze a három adattárolási módszert $L = 2$ esetén.

A klasszikus módszer túlterheltségi valószínűsége a sorbanállás-elmélet alapján:

$$P_S^{klasszikus} = \frac{(\lambda_1/\mu)^{m_1 B}/(m_1 B)!}{\sum_{i=0}^{m_1 B}(\lambda_1/\mu)^i/i!} \times \frac{2(\lambda_2/\mu)^{m_2 B}/(m_2 B)!}{\sum_{i=0}^{m_2 B}(2\lambda_2/\mu)^i/i!}$$

A kódolatlan és kódolt videórétegeket alkalmazó módszerek vizsgálatához a cikk írói Monte Carlo numerikus szimulációkat végeztek.

6.3.1. A szerver terhelésének hatása



6.1. ábra. A szerver terhelésének hatása P_s-re. Az ábra forrása: [5]

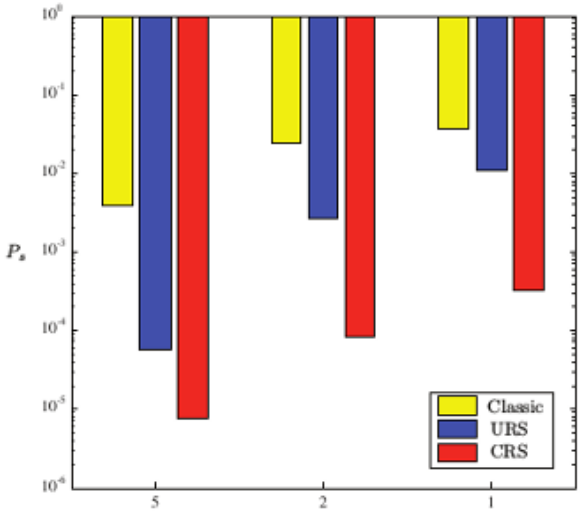
Legyen most $\lambda_1 = \lambda_2$, azaz ugyanannyian intéznek egyes és kettes típusú kérést. Jelöljük λ-val a $\lambda_1 + \lambda_2$-t.

Először azt nézzük meg, hogy a λ/μ függvényében (vagyis a beérkező felhasználók aránya a kilépő felhasználókhoz képest) hogyan változik a P_s túlterheltségi valószínűség. Ez a szerver terhelése.

Példaként tekintsük azt az esetet, amikor $B = 2$ és 12 tárolópont van, melyből $m_1 = 8$ tárolja csak az l_1 réteget, $m_2 = 4$ pedig az l_2 réteget az URS módszernél, illetve az l_2^k rétegeket a CRS módszernél. Azért így választjuk meg, mert 2-típusú kérések az URS módszernél megkétszerezik az l_1-et tároló pontok felé támasztott igények számát. A kapott eredményeket az 6.1. ábrán láthatók.

Csak kis mértékű és viszonylag konstans javulás figyelhető meg a CRS módszernél. Ezért a továbbiakban a λ/μ arányt lefixáljuk 6-ra.

6.3.2. A különböző kérések arányának hatása



6.2. ábra. A különböző kérések arányainak hatása P_s-re. Az ábra forrása: [5]

Most pedig a kérések arányában nézzük meg a P_s változását. Legyen $m = m_1 + m_2$ állandó, és az előbbi megkötésünk a λ_i-kre már nem érvényes. Ekkor m_2 és λ_1/λ_2 függvényében nézzük a változásokat. Feltesszük még az URS módszernél, hogy $m_2 \leq m_1$, hiszen ha több lenne, akkor nem lenne kihasználva legalább $m_2 - m_1$ darab tárolópont sávszélessége, hiszen minden l_2-t tároló ponthoz kell egy l_1-et tároló a 2-típusú kérésekhez. Nevezzük P_b^i-nek annak a valószínűségét, hogy a rendszer nem tud több i-típusú rendelést fogadni. Egyenlőként kezeljük a különböző típusú kéréseket, vagyis nem engedünk meg olyan eljárásmódokat, amelyek az egyik túlterheltségi valószínűségét úgy minimalizálja, hogy a másik minimalizálásával nem foglalkozunk.

Tehát adott λ_1/λ_2 esetén a $P_b^1 + cP_b^2$-t kell minimalizálnunk m_2 függvényében, ahol $c \in \mathbb{R}$. A numerikusan generált eredmények különböző λ_1/λ_2 arányokra a 6.2. ábrán láthatók. A c konstans változtatása nem okozott lényegi változást az eredményekben, így az ábrán a $c = 1$-re kapott eredmények láthatók.

Látható, hogy ha URS módszer helyett CRS-t használunk, nagyságrendes javulást érhetük el. Ezen felül, ahogy a λ_1/λ_2 arány csökken, úgy nő a túlterheltségi valószínűség. Ez várható volt, mivel az arány csökkenésével több a 2-típusú kérés, így több réteget kell letölteniük.

Irodalomjegyzék

[1] V. Aggarwal, C. Tian, V. A. Vaishampayan, and Y.-F. R. Chen. Distributed data storage systems with opportunistic repair. *arXiv preprint arXiv:1311.4096*, 2013. 19, 20

[2] V. R. Cadambe, S. A. Jafar, H. Maleki, K. Ramchandran, and C. Suh. Asymptotic interference alignment for optimal repair of mds codes in distributed storage. *Information Theory, IEEE Transactions on*, 59(5):2974–2987, 2013. 22

[3] A. G. Dimakis, P. B. Godfrey, Y. Wu, M. J. Wainwright, and K. Ramchandran. Network coding for distributed storage systems. *Information Theory, IEEE Transactions on*, 56(9):4539–4551, 2010. 7, 9

[4] T. Ernvall, S. El Rouayheb, C. Hollanti, and H. V. Poor. Capacity and security of heterogeneous distributed storage systems. *Selected Areas in Communications, IEEE Journal on*, 31(12):2701–2709, 2013. 23, 24, 25

[5] U. J. Ferner, T. Wang, and M. Médard. Network coded storage with multi-resolution codes. In *Signals, Systems and Computers, 2013 Asilomar Conference on*, pages 652–656. IEEE, 2013. 28, 30, 32, 33

[6] A.-M. Kermarrec, N. Le Scouarnec, and G. Straub. Repairing multiple failures with coordinated and adaptive regenerating codes. In *Network Coding (NetCod), 2011 International Symposium on*, pages 1–6. IEEE, 2011. 10, 14, 15

[7] E. Kiss. *Bevezetés az algebrába*. TypoTeX kiadó, 2007. 3

[8] F. Oggier and A. Datta. Coding techniques for repairability in networked distributed storage systems, 2012. 10