Megbízható számolás

Gács Péter

2005. március 11.

Kivonat

Ez a jegyzet egy haladó algoritmus-tankönyv számára készült egy fejezetnek. A témák (A „megbízható” szó elhagyásával belőlük):

- Bevezetés
- Logikai hálózatok
- Információ-tároló hálózatok

Tartalomjegyzék

1. Valószínűségszámítás **5**
1.1. Terminológia 5
1.2. A nagy számok törvénye (nagy eltérésekkel) 6
1.3. Gyakorlatok 9

2. Logikai hálózatok **10**
2.1. Boole-függvények és kifejezések 10
2.2. Logikai hálózatok 12
2.3. Gyors összeadás logikai hálózattal 14
2.4. Gyakorlatok 17

3. Költséges hibatűrés logikai hálózatokban **18**
3.1. Gyakorlatok 22

4. A részeredmények védelme **23**
4.1. Kábelek 24
4.2. Sűrítők 25
4.3. A biztonság terjesztése 28

4.4. Végjáték 30
4.5. Sűrítők konstrukciója 32
4.6. Gyakorlatok 34

5. A megbízható információtárolás problémája **36**
5.1. Ütemezett hálózatok 36
5.2. Információtárolás 38
5.3. Hibajavító kódok 39
5.4. Frissítők 45
5.5. Gyakorlatok 53

6. Feladatok **54**

7. Történeti megjegyzések **57**

Egy tervezett számítás lefuttatásakor előre nem várható hatásoknak lesz kitéve. Néhány példa:

(1) Elveszett, vagy megváltozott adatok a végrehajtás során.

(2) Véletlen, fizikai hibák a gépben.

(3) Váratlan kölcsönhatások a rendszer különböző, egyidőben működő részei között, vagy elveszett hálózati összeköttetések.

(4) Hibák a programban.

(5) Rosszindulatú támadások.

Egyelőre nem ismertek algoritmusok, amelyek a programhibák problémáját megoldanák. A szoftver-mérnöki szakma a programok struktúrájának és előállítási folyamatának tanulmányozásával és javításával közelíti meg ezt a kérdést.

Rosszindulatú támadásokkal az informatikai biztonság szakmája foglalkozik. A javasolt megoldásoknak gyakran része a kriptográfia.

A (3) típusú problémák nagyon fontosak: az osztott számítások tudományát ezek vizsgálatára hozták létre.

Az adattárolási hibák problémája hasonló a megbízható kommunikáció problémájához, melyet az információelmélet tanulmányoz: felfoghatjuk úgy, mint a jelenből jövőbe való kommunikációt. Mindkét esetben a zaj ellen *hibajavító kódok* segítségével védekezhetünk.

Ebben a fejezetben néhány példát tárgyalunk, főleg a (2) problémafajtából. Itt különbséget kell tenni állandó és átmeneti hibák között. Egy hiba *állandó*, ha a számítóberendezés egy része fizikai kárt szenved, és hosszú időre hibás marad, amígcsak külső beavatkozás nem történik (szerelő jön). A hiba *átmeneti*, ha csak egyetlen lépésben történik: a berendezésnek az a része, ahol történt, nem károsul, és a következő lépésekben megint helyesen működik. Például, ha a memória egy bitje 0-ról 1-re változik véletlenül, de a következő beírási művelet megint tud 0-t írni az érintett helyre, akkor átmeneti hiba történt. Ha a bit 1-re változott, és a gép nem tudja megint 0-ra változtatni, akkor ez állandó hiba.

Ezek közül a problémák közül egyesek, különösen az átmeneti hibák problémái, egyidősek a gépi számolással. Bármely fizikai számítási hiba részletei függenek a használt számítógéptől (és persze a kivitelezendő számítástól). De miután elvonatkoztatunk egy sereg zavaró részlettől, tiszta, de még mindig nehéz feladatokat adó elméleti megfogalmazásokra juthatunk, melyekre szép megoldások is léteznek. Érdekes kapcsolatokra bukkanunk más tudományágakkal, mint például a statisztikus fizika és a biológia.

A számítógépipar az utóbbi öt évtizedben elképesztően sikeresen tette a számítógép-alkatrészeket kisebbé, gyorsabbá, és ugyanakkor megbízhatóbbá. A sajtóban naponta olvasható számítógépes rémtörténetek közül feltűnően hiányzik az, ahol a processzor egy 1-est írt 0 helyett, csak úgy szeszélyből. (Ilyen vitathatatlanul történik, de túl ritkán ahhoz, hogy látható működési rendellenesség azonosítható forrásává váljon.) Más oldalról viszont vannak olyan, az átmeneti hibák javítására vonatkozó eredmények, melyeknek általánossága több helyzetben is alkalmazhatóvá teszi őket. Még ha az egyes fizikai processzorok nagyon megbízhatóak is (hiba talán egyszer történik minden 10^{20} ciklusban), amikor egy egész hálózat működését tekintjük számolásnak, akkor a megbízhatatlan hálózati kapcsolatok vagy akár kártevő szándékú résztvevők által okozott problémák sokban hasonlítanak a megbízhatatlan processzorok által okozottakra.

A számítások megbízhatóvá tételének kulcsgondolata a *redundancia*, melyet a következő két módszerként fogalmazhatunk meg:

(i) Tároljuk információnkat olyan formában, hogy egyetlen kis részének elvesztése se végzetes: visszaállítható a megmaradó adatokból. Például, tároljunk mindent több példányban.

(ii) Hajtsuk végre a számítást többször, hogy az esetleges hibás eredményt a többi verzió „leszavazza".

Fejezetünk csak ezeket a módszereket fogja használni, de vannak más figyelemre méltó gondolatok, melyeket nincs itt alkalmunk továbbkövetni. Például, a (ii) módszer különösen költségesnek látszik; jó lenne a sok ismétlést elkerülni. A következő ötletek ezt a dilemmát veszik célba.

(A) Hajtsuk végre számításainkat közvetlenül az információ redundáns formáján: akkor talán elkerülhető a legtöbb ismétlés.

(B) Osszuk be a számítást „szakaszokra" úgy, hogy a továbbiakban is felhasználandó részeredmények olcsón ellenőrizhetők legyenek, minden „mérföldkőnél" a szakaszok között. Csak ha az ellenőrzés hibát talál, akkor ismételjük meg az utolsó szakaszt.

1. Valószínűségszámítás

Fejezetünk nem kíván túl magas felkészültséget valószínűségszámításból, de bizonyos tények ismételten felhasználásra kerülnek: ezeket itt átvesszük. Akinek az itt közölt információnál többre van szüksége, az azt bármely haladó valószínűségszámítási tankönyvben megtalálja.

1.1. Terminológia

Egy *valószínűségi teret* egy $(\Omega, \mathcal{A}, \mathbf{P})$ hármas ír le, ahol Ω az *elemi események* halmaza, az $\mathcal{A}$ az Ω bizonyos részhalmazainak osztálya, melyeket *eseményeknek* nevezünk, és $\mathbf{P} : \mathcal{A} \to [0,1]$ egy függvény. Ha $E \in \mathcal{A}$, akkor a $\mathbf{P}(E)$ értéket az E esemény *valószínűségének* nevezzük. Megköveteljük, hogy $\Omega \in \mathcal{A}$, és hogy ha $E \in \mathcal{A}$, akkor $\Omega \setminus E \in \mathcal{A}$. Továbbá, ha a halmazoknak egy (esetleg végtelen) sorozata $\mathcal{A}$-ban van, akkor az uniójuk is. Azt is megköveteljük, hogy $\mathbf{P}(\Omega) = 1$ és hogy ha $E_1, E_2, \ldots \in \mathcal{A}$ diszjunktak, akkor

$$\mathbf{P}\Big(\bigcup_i E_i\Big) = \sum_i \mathbf{P}(E_i).$$

Ha $\mathbf{P}(F) > 0$, akkor az E esemény F-re vonatkoztatott *feltételes valószínűségét*

$$\mathbf{P}(E \mid F) = \mathbf{P}(E \cap F)/\mathbf{P}(F)$$

definiálja. Az $E_1, \ldots, E_n$ események *függetlenek*, ha minden $1 \leqslant i_1 < \cdots < i_k \leqslant n$ sorozatra

$$\mathbf{P}(E_{i_1} \cap \cdots \cap E_{i_k}) = \mathbf{P}(E_{i_1}) \cdots \mathbf{P}(E_{i_k}).$$

1.1. Példa. *Legyen $\Omega = \{1, \ldots, n\}$, ahol $\mathcal{A}$ az Ω összes részhalmazából áll, és $\mathbf{P}(E) = |E|/n$.*

Általánosabban, egy diszkrét valószínűségi tér meg van adva, ha adott egy megszámlálható $\Omega = \{\omega_1, \omega_2, \ldots\}$ halmaz, és egy $p_1, p_2, \ldots$ sorozat, melyre $p_i \geqslant 0$, $\sum_i p_i = 1$. Az események $\mathcal{A}$ halmaza az Ω összes részhalmazainak halmaza, és egy $E \subset \Omega$ esemény valószínűsége így van definiálva: $\mathbf{P}(E) = \sum_{\omega_i \in E} p_i$. ♢

Egy *valószínűségi változó* egy valamilyen Ω valószínűségi tér feletti f függvény valós szám értékekkel, továbbá azzal a tulajdonsággal, hogy minden $\{\, \omega : f(\omega) < c \,\}$ formájú halmaz esemény, tehát $\mathcal{A}$-ban van. Valószínűségi változókat gyakran X, Y, Z nagybetűkkel jelölünk, esetleg indexekkel, és az ω független változót elhagyjuk az $X(\omega)$ teljes formából. Az $\{\, \omega : X(\omega) < c \,\}$ eseményt így is írjuk:

$[\,X < c\,]$. Ezt a jelölést analóg módon bonyolultabb eseményekre is ki fogjuk terjeszteni. Egy X valószínűségi változó *eloszlása* az $F(c) = \mathbf{P}[\,X < c\,]$ függvény. Sokszor csak változóink eloszlását adjuk meg, és nem is említjük a hozzájuk tartozó valószínűségi teret, ha az összefüggésből világos, hogy így-vagy-úgy megadható. Beszélhetünk két vagy több valószínűségi változó *közös eloszlásáról*, de csak akkor, ha feltesszük, hogy mint függvények, közös valószínűségi téren definiálhatók. Az $X_1, \dots, X_n$ közös eloszlással rendelkező valószínűségi változókat *függetleneknek* nevezzük, ha az összes ilyen típusú esemény n-es független: $[\,X_1 < c_1\,]$, $\dots$, $[\,X_n < c_n\,]$.

Ha egy X valószínűségi változó az $x_1, x_2, \dots$ értékeket $p_1, p_2, \dots$ valószínűséggel veszi fel, akkor a *várható értékét* a

$$\mathbf{E}X = p_1 x_1 + p_2 x_2 + \cdots$$

képlet definiálja. Könnyű látni, hogy a várható érték lineárisan függ a valószínűségi változótól:

$$\mathbf{E}(\alpha X + \beta Y) = \alpha \mathbf{E}X + \beta \mathbf{E}Y,$$

még akkor is, ha X, Y nem független. Ha az X, Y változók függetlenek, akkor a várható értékeket össze is lehet szorozni:

$$\mathbf{E}XY = \mathbf{E}X \cdot \mathbf{E}Y. \tag{1.1}$$

Van egy fontos, egyszerű egyenlőtlenség, a *Markov egyenlőtlenség*, mely azt mondja, hogy egy tetszőleges nemnegatív X valószínűségi változóra és bármely $\lambda > 0$ értékre

$$\mathbf{P}[\,X \geqslant \lambda\,] \leqslant \mathbf{E}X/\lambda. \tag{1.2}$$

1.2. A nagy számok törvénye (nagy eltérésekkel)

Az itt megadott egyenlőtlenségek a későbbiekben hasznosak lesznek:

$$\frac{x}{1+x} \leqslant \ln(1+x) \leqslant x, \quad \text{ha } x > -1. \tag{1.3}$$

Itt a jólismert $\ln(1+x) \leqslant x$ felső korlát abból következik, hogy az $\ln(1+x)$ függvény görbéje az $x = 0$ pontban húzott tangens alatt fekszik. Az alsó korlátot az $\frac{1}{1+x} = 1 - \frac{x}{1+x}$ azonosságból és a következőkből kapjuk:

$$-\ln(1+x) = \ln\frac{1}{1+x} = \ln\left(1 - \frac{x}{1+x}\right) \leqslant -\frac{x}{1+x}.$$

Legyenek $X_1, \dots, X_n$ független, egyforma eloszlású valószínűségi változók, melyekre

$$\mathbf{P}[\, X_i = 1 \,] = p, \quad \mathbf{P}[\, X_i = 0 \,] = 1 - p.$$

Legyen

$$S_n = X_1 + \cdots + X_n.$$

Meg akarjuk becsülni a $\mathbf{P}[\, S_n \geqslant fn \,]$ valószínűséget minden $0 < f < 1$ konstansra. A „nagy számok törvénye" azt állítja, hogy ha $f > p$, akkor ez a valószínűség gyorsan 0-hoz tart $n \to \infty$ esetén, míg ha $f < p$, akkor gyorsan 1-hez tart. Legyen

$$D(f,p) = f \ln \frac{f}{p} + (1-f) \ln \frac{1-f}{1-p} \tag{1.4}$$

$$> f \ln \frac{f}{p} - f = f \ln \frac{f}{ep}\,, \tag{1.5}$$

ahol az egyenlőtlenség (hasznos, ha f kicsi és $ep < f$) következik $1 > 1 - p > 1 - f$-ből és $\ln(1-f) \geqslant -\frac{f}{1-f}$-ből (lásd (1.3)-t). A logaritmus konkáv tulajdonsága segítségével megmutatható, hogy $D(f,p)$ mindig nemnegatív, és csak akkor 0, ha $f = p$ (lásd az 1.1. gyakorlatot).

1. Tétel (Nagy eltérések pénzfeldobásokra). *Ha $f > p$, akkor*

$$\mathbf{P}[\, S_n \geqslant fn \,] \leqslant e^{-nD(f,p)}.$$

Eszerint a tétel szerint ha $f > p$, akkor $\mathbf{P}[\, S_n > fn \,]$ exponenciális sebességgel tart 0-hoz. Az (1.5) egyenlőtlenséggel tovább egyszerűsítve a

$$\mathbf{P}[\, S_n \geqslant fn \,] \leqslant e^{-nf \ln \frac{f}{ep}} = \left(\frac{ep}{f}\right)^{nf} \tag{1.6}$$

formát kapjuk, amely hasznos akkor, ha f kicsi, és $ep < f$.

Bizonyítás. Egy később megválasztandó $\alpha > 1$ valós számra legyen Y_n az a valószínűségi változó, amely α, ha $X_n = 1$ és 1 ha $X_n = 0$, és legyen $P_n = Y_1 \cdots Y_n = \alpha^{S_n}$: akkor

$$\mathbf{P}[\, S_n \geqslant fn \,] = \mathbf{P}[\, P_n \geqslant \alpha^{fn} \,].$$

A Markov egyenlőtlenség (lásd (1.2)-t) alkalmazásával

$$\mathbf{P}[\, P_n \geqslant \alpha^{fn} \,] \leqslant \mathbf{E} P_n / \alpha^{fn} = (\mathbf{E} Y_1 / \alpha^f)^n,$$

ahol $\mathbf{E}Y_1 = p\alpha + (1-p)$. Válasszunk így: $\alpha = \frac{f(1-p)}{p(1-f)}$, ez > 1, ha $p < f$. Ekkor $\mathbf{E}Y_1 = \frac{1-p}{1-f}$, és így

$$\mathbf{E}Y_1/\alpha^f = \frac{p^f(1-p)^{1-f}}{f^f(1-f)^{1-f}} = e^{-D(f,p)}.$$

□

Ez a tétel binomiális együtthatókra is jól használható becsléseket ad. Legyen

$$h(f) = -f\ln f - (1-f)\ln(1-f).$$

Ezt néha az $(f, 1-f)$ valószínűségeloszlás *entrópiájának* nevezzük (a 2 alapú logaritmus helyett e alapú logaritmusban mérve). Az (1.3) egyenlőtlenségből a

$$-f\ln f \leqslant h(f) \leqslant f\ln\frac{e}{f} \tag{1.7}$$

becslést kapjuk, ami hasznos kis f-re.

1.2. Következmény. *Az $f \leqslant 1/2$ esetben*

$$\sum_{i\leqslant fn}\binom{n}{i} \leqslant e^{nh(f)} \leqslant \left(\frac{e}{f}\right)^{fn}. \tag{1.8}$$

Ha például $f = k/n$, ahol $k \leqslant n/2$, akkor

$$\binom{n}{k} = \binom{n}{fn} \leqslant \left(\frac{e}{f}\right)^{fn} = \left(\frac{ne}{k}\right)^k. \tag{1.9}$$

Bizonyítás. Az 1. tétel azt adja az $f > p = 1/2$ esetre, hogy

$$2^{-n}\sum_{i\geqslant fn}\binom{n}{i} = \mathbf{P}[\,S_n \geqslant fn\,] \leqslant e^{-nD(f,p)} = 2^{-n}e^{nh(f)},$$

$$\sum_{i\geqslant fn}\binom{n}{i} \leqslant e^{nh(f)}.$$

A $g = 1-f$ helyettesítéssel, észrevéve az $\binom{n}{f} = \binom{n}{g}$, $h(f) = h(g)$ szimmetriákat és (1.7)-t kapjuk az (1.8) eredményt. □

1.3. Megjegyzés. *Az* (1.6) *egyenlőtlenség a $\mathbf{P}[\,S_n \geqslant fn\,] \leqslant \binom{n}{fn}p^{fn}$ triviális becslésből is következik, ha azt* (1.9)*-el kombináljuk.* ♢

1.3. Gyakorlatok

1.1. Gyakorlat. Bizonyítsuk be a főszöveg állítását, hogy $D(f, p)$ mindig nemnegatív, és csak akkor 0, ha $f = p$.

1.2. Gyakorlat. Az $f = p + \delta$ értékkel, vezessük le az 1. tételből a következő hasznos korlátot:

$$\mathbf{P}[\, S_n \geqslant fn \,] \leqslant e^{-2\delta^2 n}.$$

[Útmutatás: legyen $F(x) = D(x, p)$, és használjuk a Taylor-formulát: $F(p + \delta) = F(p) + F'(p)\delta + F''(p + \delta')\delta^2/2$, ahol $0 \leqslant \delta' \leqslant \delta$.]

2. Logikai hálózatok

Olyan számítási modellben, mely hibákat is figyelembe vesz, természetes az a feltevés, hogy hibák *mindenütt* megjelenhetnek. A számítógép legismerősebb formája—melyben a processzor és tár elválik egymástól—ilyen körülmények között rendkívül sebezhetőnek tűnik: amíg a processzor nem „néz oda", a zaj kijavíthatatlan kárt okozhat a tárban. Dolgozzunk ezért inkább olyan modellekkel, melyek *párhuzamosak*: a rendszer minden része dolgoz fel információt, nem csak egyes kitüntetett helyei. Ekkor a hibajavítást a rendszer minden részébe be lehet építeni. A legjobban ismert párhuzamos számítási modellre: a logikai hálózatokra szorítkozunk.

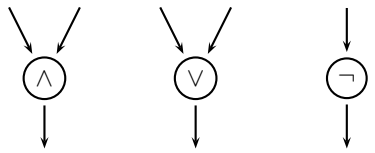
2.1. Boole-függvények és kifejezések

Vessünk egy pillantást a számítógép belsejébe (vagy inkább az integrált áramkör belsejébe, egy mikroszkóppal). A rengeteg irreleváns fizikai részlettől megzavarodva, inkább a hálózattervező rajzai felé fordulunk; mégpedig a tervezés olyan szakaszában, amikor ezek az áramkör legkisebb elemeit számítási funkcióik megjelölésével együtt mutatják. Olyan vonalak hálózatát fogjuk látni, melyek két állapotot vehetnek fel (elektromos potenciáljuk szerint): „magas" vagy „alacsony", „igaz" vagy „hamis", vagy, ahogy mi fogjuk jelölni, 1 vagy 0. A pontok, melyeket ezek a vonalak összekötnek, az ismerős *logikai összetevők*: a számítás legalacsonyabb szintjén a tipikus számítógép *biteket* dolgoz fel. Egész számok, lebegőpontos számok, betűk mind megadhatók bitfüzérekkel, és a szokásos elemi aritmetikai műveletek is összeállíthatók bit-műveletek összekapcsolásával.

2.1. Definíció. Egy *Boole-vektorfüggvény* egy $f : \{0,1\}^n \to \{0,1\}^m$ leképezés. Többnyire az $m = 1$ esettel fogunk foglalkozni, és ekkor *Boole-függvényről* fogunk beszélni. ♢

Az $f(x_1, \ldots, x_n)$ kifejezés változóit néha *logikai változóknak*, *Boole-változóknak*, vagy *biteknek* nevezzük.

2.2. Példa. *Egy irányítatlan, N pontú G gráfra érdekelhet minket az a kérdés, van-e Hamilton köre (a G összes csúcsainak egy olyan $(u_1, \ldots, u_n)$ felsorolása, hogy minden $i < n$-re (u_i, u_{i+1}) egy él, és (u_n, u_1) is egy él). Ezt a kérdést egy f Boole-függvény a következőképpen írja le. A gráfot $\binom{N}{2}$ logikai változóval adhatjuk meg: x_{ij} $(1 \leqslant i < j \leqslant N)$: $x_{ij} = 1$, ha él fut az i és j csúcsok között. Az $f(x_{12}, x_{13}, \ldots, x_{N-1,N})$ érték 1, ha G-ben van egy Hamilton kör, és 0 egyébként.* ♢



1. ábra. ÉS, VAGY és NEM kapu

2.3. Példa. *(Boole-vektorfüggvény) Legyen $n = m = 2k$, legyen bemenetünk két k-bit hosszúságú bitfüzérrel felírt egész szám: u és v: $x = (u_1, \ldots, u_k, v_1, \ldots, v_k)$. A kimenet az $y = u \cdot v$ szorzat, (bináris formában írva): ha $u = 5 = (101)_2$, $v = 6 = (110)_2$, akkor $y = u \cdot v = 30 = (11110)_2$.* ♢

Csak négy egyváltozós Boole-függvény van: 0 és 1 konstansok, az azonosság, és a *tagadás*: $x \to \neg x = 1 - x$. A következő további kétváltozós Boole-függvényeket említjük meg: a *konjukció* (logikai ÉS) művelete:

$$x \wedge y = \begin{cases} 1, & \text{ha } x = y = 1, \\ 0 & \text{különben,} \end{cases}$$

ami azonos a szorzás műveletével. A *diszjunkció*, azaz VAGY művelete:

$$x \vee y = \begin{cases} 0, & \text{ha } x = y = 0, \\ 1 & \text{különben.} \end{cases}$$

Könnyű látni, hogy $x \vee y = \neg(\neg x \wedge \neg y)$: más szóval az $x \vee y$ diszjunkció a $\neg, \wedge$ függvényekből kifejezhető az *összetétel* műveletének segítségével. A következő kétváltozós logikai függvényeket ugyancsak gyakran használják:

$$\begin{aligned} x \to y &= \neg x \vee y && \text{(implikáció),} \\ x \leftrightarrow y &= (x \to y) \wedge (y \to x) && \text{(ekvivalencia),} \\ x \oplus y &= x + y \bmod 2 = \neg(x \leftrightarrow y) && \text{(bináris összeadás).} \end{aligned}$$

Véges sok Boole-függvény elégséges ahhoz, hogy az *összes többit* kifejezzük: tehát tetszőlegesen bonyolult Boole-függvényeket ki lehet „számolni" „elemi" műveletekkel. Bizonyos értelemben ez történik minden számítógépben.

2.4. Definíció. A Boole-függvények egy Q halmaza *teljes bázis*, ha minden Boole-függvény kifejezhető a Q elemeiből képezett összetétellel. ♢

2.5. Állítás. *Az $\{\wedge, \vee, \neg\}$ halmaz teljes bázis. Más szóval, minden Boole-függvény megadható egy logikai kifejezéssel, mely csak ezt a két műveletet használja.*

A bizonyítás a kijelentéslogika bármely tankönyvében megtalálható. Mivel $\vee$ kifejezhető $\{\wedge, \neg\}$ segítségével, ez utóbbi halmaz is teljes bázis (és $\{\vee, \neg\}$ is).

Mostantól egy *logikai kifejezés* (képlet) alatt olyan kifejezést értünk, amely valamilyen adott teljes bázis elemeiből van felépitve. Ha a teljes bázist nem említjük, akkor $\{\wedge, \vee, \neg\}$-ra gondolunk.

Általában egy és ugyanaz a Boole-függvény többféleképpen is megadható logikai kifejezésekkel. Az adott kifejezésből már könnyű a függvény kiszámítása. Csakhogy a legtöbb Boole-függvényt csak nagyon nagy logikai kifejezésekkel lehet leírni (lásd a 2.4 gyakorlatot).

2.2. Logikai hálózatok

Egy logikai kifejezés néha azért nagy, mert felírása nem ad lehetőséget a részeredmények újrafelhasználására. (Például az

$$((x \vee y \vee z) \wedge u) \vee (\neg(x \vee y \vee z) \wedge v)$$

kifejezésben az $x \vee y \vee z$ rész kétszer jelenik meg.) Ezt a hiányt pótolja a következő, általánosabb formalizmus.

Egy logikai hálózat lényegében egy ciklusmentes irányított gráf, melynek minden csúcsa egy (valamely teljes bázisból vett) Boole-függvényt számol ki a bemenő élein érkező biteken, és az eredményt kiküldi a kimenő élein (lásd a 2. ábrát). Lássuk a szabatos definíciót.

2.6. Definíció. Legyen Q a Boole-függvények egy teljes bázisa. Egy N számra legyen $V = \{1, \ldots, N\}$ a *csúcsok* egy halmaza. Egy *logikai hálózat* a Q bázis fölött a következőkkel van megadva:

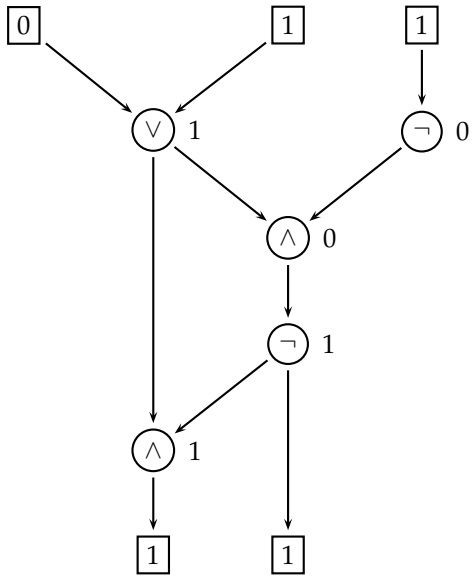
$$\mathcal{N} = (V, \{\, k_v : v \in V \,\}, \{\, \arg_j(v) : v \in V; j = 1, \ldots, k_v \,\}, \{\, b_v : v \in V \,\}). \qquad (2.1)$$

Minden v csúcshoz egy k_v természetes szám mutatja *bemeneteinek* számát. A *forrásokat*, azaz olyan v csúcsokat, melyekre $k_v = 0$, *bemenet-csúcsoknak* nevezzük: jelöljük őket, növekedő sorrendben, így:

$$\mathrm{be}_i \quad (i = 1, \ldots, n).$$

Minden nem-bemeneti v csúcshoz tartozik egy Boole-függvény

$$b_v(y_1, \ldots, y_{k_v})$$



2. ábra. Egy értékadás (értékek a csúcsokon, konfiguráció) a kapukon át terjed tovább. Ez a „számolás”.

a Q teljes bázisból: ezt a v csúcs *kapujának* nevezzük. A változók száma a csúcsba bejövő élek számával egyenlő. A gráf *nyelőit*, a kimenő élek nélküli csúcsokat, *kimeneti csúcsoknak* nevezzük. Őket így jelölhetjük:

$$\mathrm{ki}_i \quad (i = 1, \ldots, m).$$

(A mi logikai hálózatainknak többnyire csak egy kimeneti csúcsa lesz.) Minden nem-bemeneti v csúcshoz és minden $j = 1, \ldots, k_v$ számhoz egy $\arg_j(v) \in V$ csúcs tartozik (a csúcs, mely a v csúcs kapujába a j-edik változó értékét küldi). A hálózat egy $G = (V, E)$ gráfot definiál, melynek élhalmaza

$$E = \{\, (\arg_j(v), v) : v \in V,\ j = 1, \ldots, k_v \,\}.$$

Megköveteljük, hogy $\arg_j(v) < v$ teljesüljön minden j, v-re (a csúcsokat az $1, \ldots, N$ természetes számokkal azonosítottuk): ebből következik, hogy a G gráfban nincsenek irányított ciklusok. A hálózat

$$|\mathcal{N}|$$

nagysága a csúcsok száma. Egy v csúcs *mélysége* a bemeneti csúcsokból v-be vezető leghosszabb irányított út hossza. A hálózat mélysége a legmélyebb kimeneti csúcs mélysége. ♢

2.7. Definíció. Egy logikai hálózat egy *bemeneti értékadása*, vagy *bemeneti konfigurációja* egy $\boldsymbol{x} = (x_1, \ldots, x_n)$ Boole-vektor, amely az x_i értéket adja a be_i csúcsnak:

$$\text{ért}_{\boldsymbol{x}}(v) = y_v(\boldsymbol{x}) = x_i,$$

ha $v = \mathrm{be}_i$, $i = 1, \ldots, n$. Az $y_v(\boldsymbol{x})$ függvény egyértelműen kiterjeszthető a hálózat összes többi csúcsára egy $v \mapsto y_v(\boldsymbol{x})$ *konfigurációvá*, a következőképpen. Ha a b_v kapunak k változója van, akkor

$$y_v = b_v(y_{\arg_1(v)}, \ldots, y_{\arg_k(v)}). \tag{2.2}$$

Például, ha $b_v(x, y) = x \wedge y$, és $u_j = \arg_j(v)$ $(j = 1, 2)$ a v csúcshoz tartozó bemenő csúcsok, akkor $y_v = y_{u_1} \wedge y_{u_2}$. Ezt a folyamatot, melyben a fenti egyenletet követve fokozatosan kiterjesztjük a konfigurációt, a hálózat *számolásának* is nevezzük. Az $y_{\mathrm{ki}_i}(\boldsymbol{x})$ $(i = 1, \ldots, m)$ értékvektor a számolás *eredménye*. Azt mondjuk, hogy a logikai hálózat *kiszámítja* az

$$\boldsymbol{x} \mapsto (y_{\mathrm{ki}_1}(\boldsymbol{x}), \ldots, y_{\mathrm{ki}_m}(\boldsymbol{x})).$$

vektorfüggvényt. Az értékadási eljárást *szakaszokban* is lehet végezni: a t-edik szakaszban minden t mélységű csúcs értéket kap.

Az élekhez is rendelünk értékeket: egy élhez rendelt érték ugyanaz, mint amit a kiindulópontjában találunk. ♢

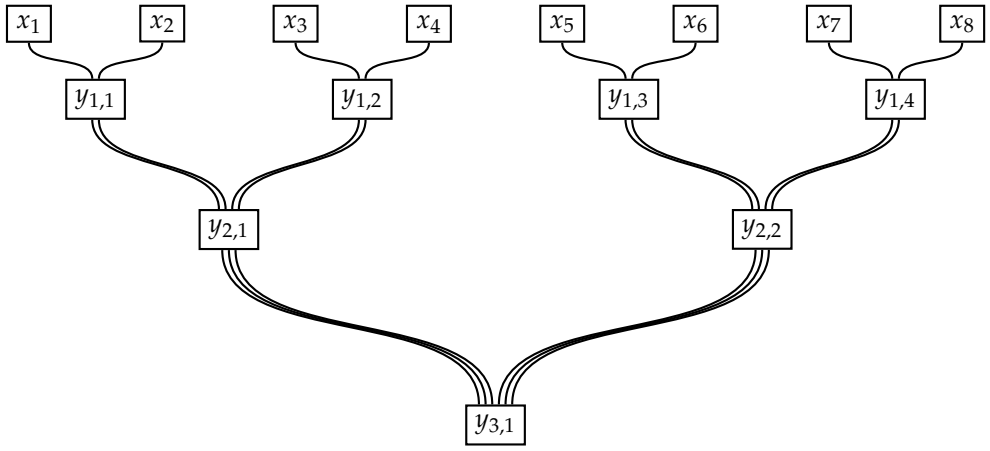
2.3. Gyors összeadás logikai hálózattal

Egy logikai hálózat mélysége annak a legrövidebb időnek tekinthető, amennyi alatt ez a hálózat a bemenetvektorból a kimenetvektort ki tudja számolni. Logikai hálózatok egy alkalmazásaként, dolgozzunk ki egy hálózatot, mely bemenő bitjeinek összegét nagyon gyorsan számolja ki. Az eredményre e fejezetben később szükség lesz hibajavító célokra.

2.8. Definíció. Egy logikai hálózat *közel-többséget* számol ki, ha kimenete egy y bit, a következő tulajdonsággal: ha a bemeneti biteknek legalább $3/4$-e b-vel egyenlő, akkor $y = b$. ♢

Hálózatunk mélysége nyilván $\Omega(\log n)$, mert a kimenetből utaknak kell vezetni a bemenetek többségébe. A többség kiszámítása érdekében a bemeneti bitek összeadásának feladatát is megoldjuk.

2. Tétel.



3. ábra. Naív párhuzamos összeadás

(a) *Egy teljes bázis felett, mely az összes 3-változós Boole-függvényeket tartalmazza, minden n-re létezik egy n bemenetnagyságú és $\leqslant 3\log(n+1)$ mélységű logikai hálózat, melynek kimenetvektora a bemenő bitek összegét adja meg, binárisan ábrázolva.*

(b) *Ugyanezen teljes bázis felett, minden n-re létezik egy n bemenetnagyságú és $\leqslant 2\log(n+1)$ mélységű logikai hálózat, mely közel-többséget számol ki.*

Bizonyítás. Először az (a) részt bizonyítjuk. Az egyszerűség kedvéért tegyük fel, hogy $n = 2^k - 1$: ha n nem ilyen formájú, akkor néhány ál-bemenetet vezethetünk be. A naív megközelítés a 3. ábra szerint járna el: először legyen $y_{1,1} = x_1 + x_2$, $y_{1,2} = x_3 + x_4$, $\ldots$, $y_{1,2^{k-1}} = x_{2^k-1} + x_{2^k}$, majd számoljuk ki a következőket: $y_{2,1} = y_{1,1} + y_{1,2}$, $y_{2,2} = y_{1,3} + y_{1,4}$, és így tovább. Ekkor a $y_{k,1} = x_1 + \cdots + x_{2^k}$ értéket k szakaszban megkapjuk.

Némi nehézséget okoz, hogy $y_{i,j}$ szám, nem bit, ezért egy *bitvektor* adja meg, azaz a hálózat csúcsainak egy csoportja, nem csak egyetlen csúcs. Csakhogy az általános

$$y_{i+1,j} = y_{i,2j-1} + y_{i,2j}$$

összeadási művelet, naív módon végrehajtva a tipikus esetben több, mint konstans mennyiségű „lépésbe" kerül: az $y_{i,j}$ számok hosszúsága $i+1$, ezért az összeadás további i-vel növelheti a mélységet, az $1 + 2 + \cdots + k = \Omega(k^2)$ értéket adva.

A következő észrevétellel csökkenthető a mélység. Legyen a, b, c három szám bináris jelölésben: például $a = \sum_{i=0}^{k} a_i 2^i$. Egyszerű párhuzamos képletek

segítségével e három szám összegét két másik szám összegével fejezhetjük ki: $a + b + c = d + e$, ahol d, e is binárisan ábrázolt számok:

$$\begin{aligned} d_i &= a_i + b_i + c_i \bmod 2, \\ e_{i+1} &= \lfloor (a_i + b_i + c_i)/2 \rfloor. \end{aligned} \tag{2.3}$$

Miután mindkét képlet egyetlen 3-változós kapuval kiszámolható, 3 számot 2-vel lehet helyettesíteni egyetlen párhuzamos számolási lépésben (az összeg megtartásával). Két ilyen lépés 4 számot 2-re csökkent. Tehát $2(k-1)$ lépésben 2^k tag összegét 2 tag összegére redukálhatjuk. E két szám szokásos módon való összeadása a mélységet még k-val növeli: azaz 2^k bitet $3k-2$ lépésben tudunk összeadni.

A (b) rész bizonyításához építsük meg az (a) rész hálózatát, de az utolsó összeadás nélkül: a kimenet két k-bit szám, melynek összege érdekel minket. Ezen számok legmagasabb helyiértékű nemnulla bitje valamely $< k$ helyen található. Ha az összeg több, mint 2^{k-1}, akkor e számok egyikében nem-0 bit van a $(k-1)$ vagy $(k-2)$ helyeken. Ez megfelelő 3-bemenetű kapuk két további alkalmazásával felismerhető. □

2.4. Gyakorlatok

2.1. Gyakorlat. Mutassuk meg, hogy $\{\oplus, \wedge\}$ teljes bázis.

2.2. Gyakorlat. Mutassuk meg, hogy az $x \,\mathrm{NOR}\, y = \neg(x \vee y)$ függvény önmagában is teljes bázist alkot.

2.3. Gyakorlat. Rögzítsük az $\{\wedge, \neg\}$ teljes bázist. Bizonyítsuk a 2.5 állítást (vagy keressük meg egy tankönyben). Adjunk segítségével felső korlátot, általános n-változós f Boole-függvény esetén, a következőkre:

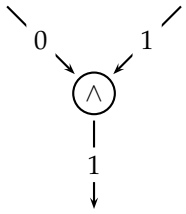
(a) az f-et leíró legkisebb logikai kifejezés nagysága;

(b) az f-et kiszámító legkisebb logikai hálózat nagysága;

(c) egy f-et kiszámító logikai hálózat legkisebb lehetséges mélysége.

2.4. Gyakorlat. Mutassuk meg, hogy minden n-re létezik egy n-változós f Boole-függvény, melyre minden, az $\{\wedge, \neg\}$ teljes bázisban őt kiszámoló logikai hálózat $\Omega(2^n/n)$ csúcsot tartalmaz. [Útmutatás: egy $c > 0$ konstansra, becsüljük felülről a legfeljebb $c2^n/n$ csúcsú logikai hálózatok számát, és hasonlítsuk ezt össze az n-változós Boole-függvények számával.]

2.5. Gyakorlat. Tekintsük azt az $\mathcal{M}_3^r$ hálózatot 3^r bemenettel, melynek egyetlen kimenő bitjét r-szer iterált 3-bemenetű többségi kapukkal kapjuk a bemenetekből. Mutassuk meg, hogy létezik egy olyan $\boldsymbol{x}$ bemenetvektor, amely csak $n^{1/\log 3}$ helyen 1, de amellyel $\mathcal{M}_3^r$ kimenete 1 lesz. Tehát a bemeneteknek elenyésző kisebbségét is el lehet ügyesen úgy rendezni, hogy a hálózat kimenetét meghatározza.



4. ábra. Hiba egy kapunál.

3. Költséges hibatűrés logikai hálózatokban

Vegyünk egy $\mathcal{N}$ logikai hálózatot, a 2.6. definíció szerint. Ha zaj is felléphet, akkor az

$$y_v = \text{ért}_{\boldsymbol{x}}(v)$$

értékeket nem határozza meg már a (2.2) képlet. Helyükre az Y_v valószínűségi változók lépnek. Nevezzük az $(Y_v : v \in V)$ véletlen vektort *véletlen konfigurációnak*.

3.1. Definíció. A v csúcsnál legyen

$$Z_v = b_v(Y_{\text{arg}_1(v)}, \ldots, Y_{\text{arg}_k(v)}) \oplus Y_v, \tag{3.1}$$

azaz $Z_v = 1$, ha Y_v nem azonos a zaj nélküli b_v kapu által az $Y_{\text{arg}_j(v)}$ bemenetekből kiszámolt értékkel. (Lásd a 4. ábrát.) A csúcsok halmazát, ahol Z_v nem nulla, a *tévedések* halmazának nevezzük (ezzel elkerüljük a többféleképpen érthető „hiba" szót).

Nevezzük az $\text{ért}_{\boldsymbol{x}}(v) \oplus Y_v$ különbséget a v csúcs *eltérésének*. ♢

Szorítsuk meg, milyen fajta zajt engedünk meg. Minden tévedés legfeljebb ε valószínűséggel léphet fel. Két megadott helyen egyszerre legfeljebb ε^2 valószínűséggel léphet fel tévedés, és így tobább.

3.2. Definíció. Adott $\varepsilon > 0$-ra azt mondjuk, hogy az $(Y_v : v \in V)$ véletlen konfiguráció *ε-megengedett*, ha

(a) $Y_{\text{be}(i)} = x_i$ minden $i = 1, \ldots, n$-re.

(b) A nem bemeneti csúcsok minden C halmazára

$$\mathbf{P}[\, Z_v = 1 \text{ minden } v \in C\text{-re} \,] \leqslant \varepsilon^{|C|}. \tag{3.2}$$

♢

Más szóval, egy ε-megengedett véletlen konfigurációban, k különböző megadott kapunál legfeljebb ε^k valószínűséggel léphet fel egyszerre tévedés. Így követeljük meg, hogy ne csak kicsi legyen a tévedések valószínűsége, de ráadásul a tévedések ne „esküdhessenek össze". A megengedettségi feltétel teljesül, ha a tévedések egymástól függetlenül, $\leqslant \varepsilon$ valószínűséggel következnek be.

Célunk olyan hálózatot készíteni, amely nagy valószínűséggel helyesen működik, a mindig jelenlévő zaj ellenére: más szóval, a hibák *nem halmozódnak*. Ezt a fogalmat formalizáljuk a következőkben.

3.3. Definíció. Egy $\mathcal{N}$ hálózatot, melynek kimenő csúcsa w, akkor nevezünk (ε, δ)-*ellenállónak*, ha minden $\boldsymbol{x}$ bemenetvektorhoz, minden ε-megengedett $\boldsymbol{Y}$ konfigurációban $\mathbf{P}[\, Y_w \neq \text{ért}_{\boldsymbol{x}}(w)\,] \leqslant \delta$. ♢

Járjuk körül egy kicsit ezt a fogalmat. Nincs (ε, δ)-ellenálló hálózat, ha $\delta < \varepsilon$, mert még az utolsó kapu is ε valószínűséggel tévedhet. Engedjük meg ezért, egy kicsit nagylelkűen, a $\delta > 2\varepsilon$ lehetőséget. Nyilván minden $\mathcal{N}$ hálózathoz és minden $\delta > 0$ értékhez lehet olyan kicsi ε-t választani, amely biztosítja, hogy (ε, δ)-ellenálló legyen $\mathcal{N}$. De hát nem ezt akarjuk elérni: remélhetőleg nem lesz szükség egyre megbízhatóbb kapukra, ahányszor csak nagyobb hálózatokat akarunk építeni. Egy olyan

$$F(N, \delta)$$

függvényt keresünk tehát, és egy olyan $\varepsilon_0 > 0$ tévedés-korlátot, hogy minden $\varepsilon < \varepsilon_0$ és $\delta \geqslant 2\varepsilon$ esetén, minden N nagyságú $\mathcal{N}$ Boole-hálózatra, legyen egy $F(N, \delta)$ nagyságú (ε, δ)-ellenálló $\mathcal{N}'$ hálózat, amely ugyanazt a függvényt számolja ki, mint $\mathcal{N}$. Ha ezt elérjük, akkor elmondhatjuk, hogy megakadályoztuk a hibák halmozódását. Persze azt akarjuk, hogy $F(N, \delta)$ viszonylag kicsi legyen, és ε_0 nagy (nagyobb zajt megengedve). Az $F(N, \delta)/N$ függvényt *redundanciának* nevezhetjük: ezzel szorzóval kell növelni a hálózat nagyságát, hogy ellenállóvá tegyük. Érdemes megyjegyezni, hogy a probléma még akár $\delta = 1/3$ esetén sem triviális. Ha a hibák halmozódása előtt nincs akadály, akkor fokozatosan minden információ elvész a kívánt kimenő értékről, és semmilyen $\delta < 1/2$ nem garantálható.

Hogy javítsuk ki a hibákat? Egy egyszerű gondolat: számoljunk ki „mindent" 3-szor, aztán folytassuk a többségi szavazás eredményével.

3.4. Definíció. Egy d páratlan természetes számra, a d-bemenetű többségi kapu az a Boole-függvény, amelynek kimenő értéke egyenlő a bemenő értékek többségével. ♢

A d-bemenetű többségi érték kiszámítható, $O(d)$ ÉS és VAGY kapu segítségével.

Miért várható, hogy a többségi szavazás segít? A következő, *informális diszkusszió* segít megérteni az előnyöket és buktatókat. Tegyük fel egy pillanatra, hogy az egész számítás kimenete egyetlen bit. Ha bármelyik, függetlenül kiszámolt eredmény tévedési valószínűsége δ, akkor annak a valószínűsége, hogy legalább 2 téves közülük, $3\delta^2$-el korlátozható. Mivel maga a többségi szavazás is hibázhat $\leqslant \varepsilon$ valószínűséggel, a kudarc teljes valószínűségét $3\delta^2 + \varepsilon$ korlátozza. Tehát a hiba δ valószínűsége csökken, a $3\delta^2 + \varepsilon < \delta$ feltétel mellett.

Úgy látszik tehát, hogy ha δ kicsi, akkor ismétlés és többségi szavazás tovább csökkentheti. Persze, ha a hibavalószínűség növekedését meg akarjuk akadályozni, a többségi szavazást újra és újra végre kell hajtani. Tegyük fel például, hogy számításunk t egymást követő szakaszból áll. Az i-edik szakasz után korlátunk a hibás kimenet valószínűségére δ_i. Minden szakasz után többségi szavazást akarunk végrehajtani. Hajtsuk végre az i szakaszt háromszor. A hiba valószínűségére most a

$$\delta_{i+1} = \delta_i + 3\delta^2 + \varepsilon \tag{3.3}$$

korlátot kapjuk. Tehát a különböző szakaszok hibavalószínűségei halmozódnak, és még a $3\delta^2 + \varepsilon < \delta$ egyenlőtlenség esetén is csak a $\delta_t < (t-1)\delta$ korlátot kapjuk. Ez a stratégia tehát nem működik tetszőlegesen nagy számításokra.

Egy vad ötlet a halmozódás elkerülésére: ismételjünk meg *mindent* háromszor, ami az i-edik szakasz előtt történt, ne csak magát az i-edik szakaszt! Ekkor az egyre növekvő (3.3) korlátot a

$$\delta_{i+1} = 3(\delta_i + \delta)^2 + \varepsilon$$

korlát helyettesíti. Most, ha $\delta_i < \delta$ és $12\delta^2 + \varepsilon < \delta$, akkor megintcsak $\delta_{i+1} < \delta$, tehát a hibák nem halmozódnak. De irdatlan árat fizettünk: mire a számítás $(i+1)$-edik szakaszába értünk, a hibatűrő verzió nagysága 3-szorosa annak, ami az i-edik szakaszig volt. Ha t szakaszt akarunk ilyen módon hibatűrővé tenni, ez egy 3^t szorzóba kerül. Így a fent bevezetett $F(N, \delta)$ függvény N-ben exponenciálissá válhat.

Az alábbi tétel egy lehetséges szabatos verziója az itt elhangzott megfontolásoknak.

3. Tétel. *Legyen R egy teljes, véges bázisa a Boole-függvényeknek. Ha $2\varepsilon \leqslant \delta \leqslant 0.01$, akkor minden függvényt ki lehet számolni egy (ε, δ)-ellenálló hálózattal R fölött.*

Bizonyítás. Az egyszerűség kedvéért az eredményt csak olyan teljes bázisban bizonyítjuk, amely tartalmazza a háromváltozós többségi szavazást, és nem tartalmaz háromnál több változós függvényt. Azt is föltesszük, hogy a hibák függetlenek egymástól.

Legyen $\mathcal{N}$ egy t mélységű zaj nélküli hálózat, mely az f függvényt számolja ki. Bebizonyítjuk, f-et ki lehet számolni egy $2t$ mélységű (ε, δ)-ellenálló $\mathcal{N}'$ hálózattal. A bizonyítás t szerinti indukció. Az ε-ra és δ-ra vonatkozó elégséges feltételek menet közben fognak kiderülni.

Az állítás természetesen igaz $t = 1$-re, tegyük fel tehát, hogy $t > 1$. Legyen g az $\mathcal{N}$ hálózat kimeneti kapuja, akkor $f(\boldsymbol{x}) = g(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), f_3(\boldsymbol{x}))$. Az f_i függvényeket $\leqslant t-1$ mélységű $\mathcal{N}_i$ részhálózatok számolják ki. Az induktív feltevés szerint az f_i függvényeket ki lehet számolni (ε, δ)-ellenálló, $\leqslant 2t-2$ mélységű $\mathcal{N}_i'$ hálózatokkal. Legyen $\mathcal{M}$ az új hálózat, amely az $\mathcal{N}_i'$ hálózatok példányait tartalmazza (a megfelelő beviteli csúcsok összeragasztásával), és egy új csúcsot, amelyben a g kapu bemenetként kapja az $\mathcal{N}_i'$ hálózatok kimeneteit, és kiszámolja $f(\boldsymbol{x})$-et. Ekkor $\mathcal{M}$ hibavalószínűsége legfeljebb $3\delta + \varepsilon < 4\delta$, ha $\varepsilon < \delta$, mert mindhárom hálózat δ valószínűséggel hibázhat, és a g kaput tartalmazó csúcs $\leqslant \varepsilon$ valószínűséggel tévedhet.

Végül állítsuk össze az $\mathcal{N}'$ hálózatot az $\mathcal{M}$ hálózat három egyforma példányából (összeragasztott bemenetekkel), és mégegy csúcsból, amely a három kimenet között többségi szavazással dönt. Az $\mathcal{N}'$ hálózat hibavalószínűsége legfeljebb $3(4\delta)^2 + \varepsilon = 48\delta^2 + \varepsilon$. Valóban, a hibát vagy a többségi kapu tévedése okozza, vagy legalább ketten hibáztak az $\mathcal{M}$ hálózat három független példánya közül. Tehát a

$$48\delta^2 + \varepsilon \leqslant \delta \tag{3.4}$$

feltétel mellett az $\mathcal{N}'$ hálózat (ε, δ)-ellenálló. A feltétel teljesül, ha $2\varepsilon \leqslant \delta \leqslant 0.01$. □

A bizonyításban konstruált $\mathcal{N}'$ hálózat legalább 3^t-szer nagyobb, mint $\mathcal{N}$, a redundancia tehát irdatlanul nagy. Szerencsére, sokkal gazdaságosabb megoldásokat is fogunk látni. De vannak érdekes, kis mélységű hálózatok, amelyekre a 3^t szorzó nem extravagáns.

4. Tétel. *Álljon teljes bázisunk az összes 3-változós Boole-függvényből. Ekkor minden elég kicsi $\varepsilon > 0$-ra, ha $2\varepsilon \leqslant \delta \leqslant 0.01$, akkor minden n-re van egy n-bemenetű, (ε, δ)-ellenálló logikai hálózat, melynek mélysége $\leqslant 4\log(n+1)$ és nagysága $(n+1)^7$, és mely a bemenetek közelítő többségét számolja ki (a 2.8. definíció szerint).*

Bizonyítás. Alkalmazzuk a 3. tételt arra a hálózatra, melyet a 2. tétel (a) részéből kapunk. Ez egy új, $4\log(n+1)$-mély (ε, δ)-ellenálló hálózatot ad, mely közelítő többséget számol ki. Bármilyen ilyen hálózat, mely 3-bemenetű kapukból áll, legfeljebb $3^{4\log(n+1)} = (n+1)^{4\log 3} < (n+1)^7$ nagyságú. □

3.1. Gyakorlatok

3.1. Gyakorlat. A 2.5. gyakorlat azt sugallja, hogy az $\mathcal{M}_3^r$ iterált többségi szavazás manipulálható. Bizonyos körülmények között azonban nagyon jól működik. Legyen az $\mathcal{M}_3^r$ bemenete a független Boole-értékű valószínűségi változók $\boldsymbol{X} = (X_1, \ldots, X_n)$ vektora, melyre $\mathbf{P}[\, X_i = 1\,] = p < 1/6$. Legyen Z a hálózat (véletlen) kimenet-bitje. Feltéve, hogy többségi kapuink egymástól függetlenül $\leqslant \varepsilon \leqslant p/2$ valószínűséggel tévedhetnek, bizonyítsuk be az alábbi becslést:

$$\mathbf{P}[\, Z = 1\,] \leqslant \max\{10\varepsilon,\ 0.3(p/0.3)^{2^k}\}.$$

[Útmutatás: Legyen $g(p) = \varepsilon + 3p^2$, $g_0(p) = p$, $g_{i+1}(p) = g(g_i(p))$, és bizonyítsuk a következőt: $\mathbf{P}[\, Z = 1\,] \leqslant g_r(p)$.]

3.2. Gyakorlat. Azt mondjuk, hogy az $\mathcal{N}$ hálózat az $f(x_1, \ldots, x_n)$ függvényt (ε, δ)-*bemenet-biztos* módon számolja ki, ha a következő teljesül. Bármilyen $\boldsymbol{x} = (x_1, \ldots, x_n)$ bemenő vektorra, bármilyen ezt „perturbáló" független $\boldsymbol{X} = (X_1, \ldots, X_n)$ Boole-változó sorozatra, amely a $\mathbf{P}[\, X_i \neq x_i\,] \leqslant \varepsilon$ követelménynek tesz eleget, az $\boldsymbol{X}$ bemenetű $\mathcal{N}$ hálózat Y kimenete keveset változik: $\mathbf{P}[\, Y = f(\boldsymbol{x})\,] \geqslant 1 - \delta$. Mutassuk meg, hogy ha létezik olyan logikai hálózat, amely az $x_1 \oplus \cdots \oplus x_n$ függvényt $(\varepsilon, 1/4)$-bemenet-biztosan kiszámítja, akkor $\varepsilon \leqslant 1/n$.

4. A részeredmények védelme

Ebben a fejezetben olyan hibatűrési módszereket ismerünk meg, amelyeknek viselkedése a rendszernagyság növelésekor kedvezőbb. Meg fogjuk mutatni a következőt:

5. Tétel. *Léteznek olyan R_0, ε_0 konstansok, hogy az*

$$F(n, \delta) = N \log(n/\delta)$$

definícióval, minden $\varepsilon < \varepsilon_0$, $\delta \geqslant 3\varepsilon$ esetére, minden N nagyságú determinisztikus számításhoz van egy (ε, δ)-ellenálló, $R_0 F(N, \delta)$ nagyságú számítás, amely ugyanazt az eredményt adja.

Bevezetünk egy fogalmat, amely egyszerűsíti hálózataink hiba-elemzését, függetlenítve azt a bemeneti $\boldsymbol{x}$ vektortól.

4.1. Definíció. Egy $\mathcal{N}$ logikai hálózat egy v csúcsában nevezzünk egy többségi kaput *javító többségi kapunak*, ha $\mathcal{N}$ minden $\boldsymbol{x}$ bemeneti vektorára, a v csúcs minden bemeneti élén ugyanaz az érték érkezik. Tekintsük az $\mathcal{N}$ hálózat egy számítását: ez a számítás egyes csúcsokat és vezetékeket *megfertőz*. A következő szabályok szerint terjed a fertőzés:

– A bemeneti csúcsok nem fertőzöttek.

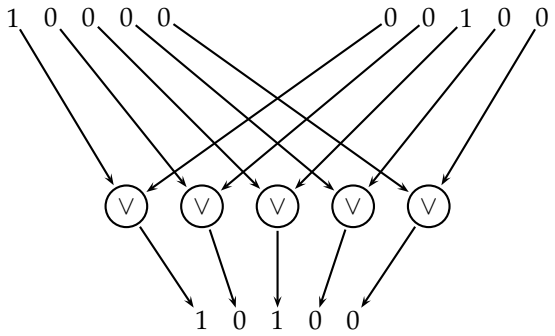
– Ha egy csúcs fertőzött, akkor minden kimeneti vezetéke fertőzött.

– Egy javító többségi kaput tartalmazó csúcs akkor fertőzött, ha vagy téved, vagy bemeneteinek többsége fertőzött.

– Minden más csúcs akkor fertőzött, ha vagy téved, vagy valamelyik bemenete fertőzött.

$\diamondsuit$

Nyilván, ha minden ε-megengedett véletlen konfigurációnál a hálózat kimenete $\leqslant \delta$ valószínűséggel fertőzött, akkor a hálózat (ε, δ)-ellenálló.



5. ábra. Végrehajtó szerv.

4.1. Kábelek

Egyelőre még csak a jelen könyvrész bevezetésének (ii) ötletét használtuk: a számítási lépések ismétlését. Próbáljuk az (i) ötletet is használni logikai hálózatokban (az információt redundáns formában tartani).

Hogy kaputól kapuig védjük az áramló információt, a zaj nélküli hálózat minden vezetékét egy k vezetéket tartalmazó „kábellel" helyettesítjük (ahol k-t alkalmasan fogjuk választani). Egy kábel minden vezetékének ugyanazt a bit információt kellene szállítani, és azt reméljük, hogy többségük ezt a bitet viszi majd, még ha egyes vezetékek tévednek is.

4.2. Definíció. Egy $\mathcal{N}'$ logikai hálózatban, az élek egy bizonyos halmazát *kábelnek* hívhatjuk, ha a hálózat minden zajmentes számolásában, mindegyik él ugyanazt a Boole értéket viszi. A halmaz elemszámát a kábel *vastagságának* nevezzük. Rögzítsünk egy megfelelő konstans ϑ küszöböt. Tekintsük az $\mathcal{N}'$ hálózat egy zajos verziójának bármilyen lehetséges számolását, és ebben egy k vastagságú kábelt. Ezt a kábelt *ϑ-biztonságosnak* nevezzük, ha legfeljebb ϑk él fertőzött benne. $\diamondsuit$

Vegyünk egy $\mathcal{N}$ hálózatot, amelyet ellenállóvá akarunk tenni. Amint az $\mathcal{N}$ vezetékeit az $\mathcal{N}'$ (egyenként k vezetéket tartalmazó) kábeleivel helyettesítünk, egy-egy v csúcsnál minden 2-bemenetelű zajnélküli kaput egy úgynevezett *végrehajtó szerv* nevű, k kapuból álló egységgel helyettesítjük. Ez, minden $i = 1, \ldots, k$-ra, az első és a második kábel i-edik vezetékét a végrehajtó szerv i-edik csúcsába vezeti. Mindezek a csúcsok ugyanazt a b_v típusú kaput tartalmazzák. Az ezekből a csúcsokból előbukkanó vezetékek adják a végrehajtó szerv *kimeneti kábelét*.

A kimeneti kábelben túl magasra nőhet a fertőzött vezetékek száma: valóban, ha az x vezetékben ϑk fertőzött vezeték volt, és az y vezetékben ugyancsak, akkor a $g(x, y)$ vezetékben már akár $2\vartheta k$ fertőzött vezeték is lehet (nem is számolva

6. ábra. Felújító szerv.

a végrehajtó szerv tévedései által hozzáadott új fertőzéseket). A konstrukció döntő része az, hogy a végrehajtó szervhez még egy úgynevezett *felújító szervet* is illesztünk: ennek az egységnek az a feladata, hogy a kábel fertőzöttségét csökkentse.

4.2. Sűrítők

Hogy készítsünk felújító szervet? Szem előtt tartva, hogy ennek a szervnek is zajban kell működni, építhetnénk (megfelelő δ'-re) egy speciális (ε, δ')-ellenálló hálózatot, amely k bemenetének közel-többségét számolja ki, k független példányban. A 4. tétel egy $k(k+1)^7$ nagyságú hálózatot szolgáltat erre.

Szerencsére jobb megoldás is van, legalábbis aszimptotikusan. *Nagyon egyszerű* felújító szervet fogunk keresni, olyat, amelynek a saját zaját már könnyű lesz elemezni. Mi egyszerűbb, mint egy olyan hálózat, melyben csak *egy lépés* van a bemenetek és kimenetek között? Rögzítsünk egy páratlan d egész számot (például, $d = 3$). Szervünk minden kapuja egy d-bemenetű többségi kapu lesz.

4.3. Definíció. Nevezzünk *multigráfnak* minden olyan gráfot, amelyben minden pontpár között több él is futhat, nem csak 0 vagy 1. Egy páros multigráfot, melynek k bemenete és k kimenete van, nevezzünk *d-félregulárisnak*, ha minden kimeneti pont d fokú. Egy ilyen gráfot (d, α, γ, k)-*sűrítőnek* nevezünk, ha a következő tulajdonsággal bír: a bemenetek minden legfeljebb αk pontot tartalmazó E halmazához, legfeljebb $\leqslant \gamma\alpha k$ olyan kimenet van, amely az E legalább $d/2$ elemével van összekötve (multiplicitást is számolva). ♢

A sűrítő tulajdonság általában a $\gamma < 1$ esetben érdekes. Például egy

$(5, 0.1, 0.5, k)$-sűrítőben a kimenetek foka 5, és a többségi szavazás ezekben a pontokban minden olyan hibahalmazt, amely legfeljebb a bemenetek 10%-át foglalja el, a kimenetek 5%-ára csökkent.

Egy megfelelő paraméterekkel bíró sűrítő működhetne felújító szervként: csökkentve a kisebbségben lévő eltéréseket, a kábel biztonságát helyreállíthatja. De vannak-e sűrítők?

6. Tétel. *Minden* $0 < \gamma < 1$*, és páratlan egész* d*-re, ha*

$$1 < \gamma(d-1)/2, \tag{4.1}$$

akkor létezik egy $\alpha > 0$ *korlát, mellyel minden egész* $k > 0$*-ra vannak* (d, α, γ, k)*-sűrítők.*

Amint látjuk, a $d = 3$ esetre a tétel nem garantál sűrítőt $\gamma < 1$ értékkel.

Bizonyítás. Nem adunk explicit konstrukciót a keresett multigráfra, csak megmutatjuk, hogy létezik. Választunk egy véletlen d-félreguláris multigráfot (minden ilyen multigráfot ugyanolyan valószínűséggel), és megmutatjuk, hogy ez pozitív valószínűséggel (d, α, γ, k)-sűrítő lesz. Ezt a bizonyítási módszert *valószínűségi módszernek* nevezik. Legyen

$$s = \lfloor d/2 \rfloor.$$

Konstrukciónk kicsit általánosabb lesz, $k' \neq k$ számú kimenetet is megengedve. Képezzünk egy véletlen páros multigráfot k bemenettel és k' kimenettel, a következőképpen: minden kimenethez d élt húzunk véletlen bemeneti csúcsokból, amelyeket függetlenül és egyenletes eloszlással választunk az összes bemeneti csúcsok közül.

Legyen A egy αk nagyságú bemenethalmaz, legyen v egy kimeneti csúcs, és legyen E_v az esemény, hogy v-be legalább $s+1$ él vezet A-ból. Ekkor

$$\mathbf{P}(E_v) \leqslant \binom{d}{s+1} \alpha^{s+1} = \binom{d}{s} \alpha^{s+1}.$$

Jelöljük a jobboldalt p-vel. Átlagban (várható értékben), az E_v esemény pk' különbözű kimenetben következik be. Egy A bemenethalmazhoz legyen F_A az az esemény, hogy a v kimenetek száma, melyekben az E_v esemény bekövetkezik, több, mint $\gamma\alpha k'$. Az (1.6) egyenlőtlenség alapján

$$\mathbf{P}(F_A) \leqslant \left(\frac{ep}{\gamma\alpha}\right)^{k'\gamma\alpha}.$$

Az összes lehetséges legfeljebb αk elemű A bemenethalmazok M számára az (1.7) egyenlőtlenség a következő becslést adja:

$$M \leqslant \sum_{i \leqslant \alpha k} \binom{k}{i} \leqslant \left(\frac{e}{\alpha}\right)^{\alpha k}.$$

Annak valószínűsége, hogy véletlen gráfunk nem sűrítő, legfeljebb akkora, mint annak valószínűsége, hogy az F_A esemény legalább egy A bemenethalmazra bekövetkezik. Ezt most így becsülhetjük:

$$M \cdot \mathbf{P}(F_A) \leqslant e^{-\alpha D k'},$$

ahol

$$D = -(\gamma s - k/k') \ln \alpha - \gamma\big(\ln \tbinom{d}{s} - \ln \gamma + 1\big) - k/k'.$$

Az α konstans csökkentésével ebben a kifejezésben az első tag dominál. Együtthatója pozitív, a (4.1) feltétel miatt. Tehát $D > 0$, ha

$$\alpha < \exp\left(-\frac{\gamma\big(\ln \tbinom{d}{s} - \ln \gamma + 1\big) + k/k'}{\gamma s - k/k'}\right).$$

□

4.4. Példa. *A $\gamma = 0.4$, $d = 7$, választással $\alpha = 10^{-7}$ megfelel.* ♢

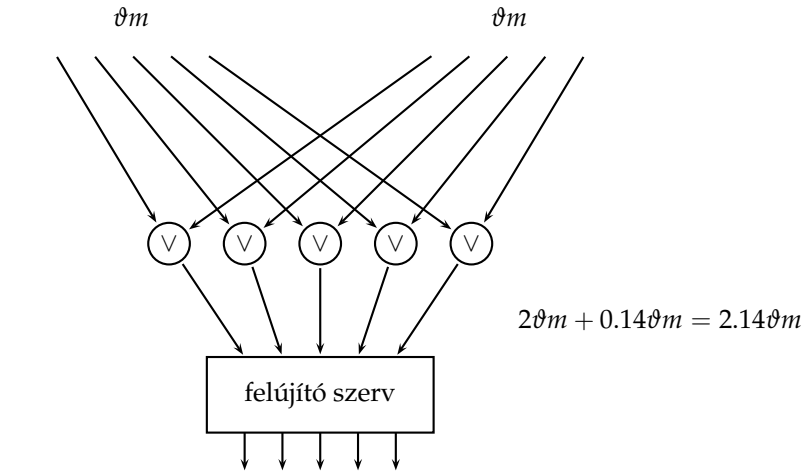
Egy (d, α, γ, k)-sűrítőből úgy csinálunk egy $\mathcal{R}$ felújító szervet, hogy a kimenő csúcsaiba d-bemenetű többségi kapukat teszünk. Ha a kapuk néha tévednek, akkor $\mathcal{R}$ kimenete véletlen. Tegyük fel, hogy $\mathcal{R}$-nek legfeljebb αk bemenete fertőzött. Ekkor csak úgy lehet $(\gamma + \rho)\alpha k$ kimenet fertőzött, ha $\alpha\rho k$ többségi kapu hibázik. Legyen

$$p_{\mathcal{R}}$$

ennek az eseménynek a valószínűsége. Feltéve, hogy a kapuk $\mathcal{R}$-ben egymástól függetlenül $\leqslant \varepsilon$ valószínűséggel hibáznak, az (1.6) egyenlőtlenségől

$$p_{\mathcal{R}} \leqslant \left(\frac{e\varepsilon}{\alpha\rho}\right)^{\alpha\rho k} \tag{4.2}$$

következik.



7. ábra. Végrehajtó szervet felújító szerv követ.

4.5. Példa. *Válasszuk a következő értékeket:* $\gamma = 0.4$, $d = 7$, $\alpha = 10^{-7}$, *akárcsak a 4.4 példában, továbbá* $\rho = 0.14$ *(ez teljesíteni fogja a későbbiekben szükséges* (4.3) *egyenlőtlenséget). Ekkor az* $\varepsilon = 10^{-9}$ *tévedéskorláttal a* $p_{\mathcal{R}} \leqslant e^{-10^{-8}k}$ *korlátot kapjuk.*

A vonzóan kicsi $d = 7$ *fokszám sajnos kiábrándítóan gyenge korlátot ad csak annak valószínűségére, hogy a sűrítő kudarcot vall. Ez a korlát ugyan exponenciális sebességgel csökken a k kábelvastagság függvényében, de csak szélsőségesen nagy k esetén lesz tényleg kicsi.* ♢

4.6. Példa. *Megint* $\gamma = 0.4$*-et választva, de hozzá* $d = 41$*-et (tehát minden sűrítő kapu 41 vezeték többségét veszi 7 helyett), valamivel reálisabb eredményeket kapunk. Ez a választás lehetővé teszi az* $\alpha = 0.15$ *értéket. Legyen megint* $\rho = 0.14$, $\varepsilon = 10^{-9}$, *akkor* $p_{\mathcal{R}} \leqslant e^{-0.32k}$ *következik.*

Ezek a számok kevésbé ijesztőek, de még mindig közel száz vezeték kell ahhoz, hogy a felújítási hiba valószínűsége kicsi legyen. És bár a gyakorlatban a számítógép-elemek sokkal kisebb, mint 10^{-9} *gyakorisággal tévednek, az a kérdés is érdekelhet minket, mi a legnagyobb megtűrhető* ε*.* ♢

4.3. A biztonság terjesztése

Sűrítők segítségével olyan logikai hálózatot építhetünk, melynek minden kábele nagy valószínűséggel biztonságos.

4.7. Definíció. Egy adott $\mathcal{N}$ logikai hálózathoz, melynek (egyszerűség kedvéért) csak egyetlen bit a kimenete, egy k kábelnagysághoz és egy $\mathcal{R}$ logikai hálózathoz, melynek k bemenete és k kimenete van, legyen

$$\mathcal{N}' = \mathrm{Cab}(\mathcal{N}, \mathcal{R})$$

a logikai hálózat, melyet a következőképpen kapunk. A bemenetek ugyanazok, mint $\mathcal{N}$-nek. Az $\mathcal{N}$ minden vezetékét egy k vastagságú kábellel helyettesítjük, és $\mathcal{N}$ minden kapuját helyettesítjük egy végrehajtó szervvel, amit egy olyan felújító szerv követ, mely az $\mathcal{R}$ hálózat másolata. Az új hálózatnak k kimenete van: ezek az $\mathcal{N}'$ utolsó felújító szervének kimeneteivel azonosak. ♢

Zaj nélküli számításokban, minden bemenő Boole-vektorhoz, $\mathcal{N}'$ kimenete ugyanaz, mint $\mathcal{N}$-é, de k azonos példányban.

4.8. Lemma. *Léteznek olyan $d, \varepsilon_0, \vartheta, \rho > 0$ konstansok, és minden k kábelvastagságra létezik egy $2k$ nagyságú $\mathcal{R}$ hálózat $\leqslant d$ bemenetszámú kapukkal, a következő tulajdonsággal. Minden $\mathcal{N}$ logikai hálózathoz, melynek kapunagysága 2 és nagysága N, minden $\varepsilon < \varepsilon_0$-ra, az $\mathcal{N}' = \mathrm{Cab}(\mathcal{N}, \mathcal{R})$ hálózat minden ε-megengedett konfigurációjára, annak a valószínűsége, hogy $\mathcal{N}'$-nek nem minden kábele ϑ-biztonságos, kisebb, mint $2N(\frac{e\varepsilon}{\vartheta\rho})^{\vartheta\rho k}$.*

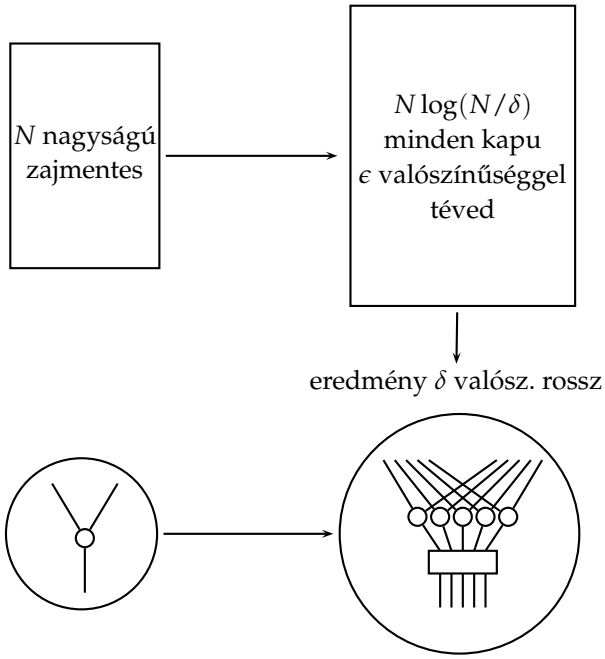
Bizonyítás. Tudjuk, hogy található olyan d, α és $\gamma < 1/2$, melyre minden k-hoz létezik egy (d, α, γ, k)-sűrítő. Válasszuk ρ-t úgy, hogy a következő egyenlőtlenség teljesüljön:

$$\gamma(2+\rho) + \rho \leqslant 1, \tag{4.3}$$

és legyen

$$\vartheta = \alpha/(2+\rho). \tag{4.4}$$

Készítsünk egy $\mathcal{R}$ felújító szervet egy (d, α, γ, k)-sűrítőből. Tekintsük az $\mathcal{N}$ hálózat egy v kapuját, és az $\mathcal{N}' = \mathrm{Cab}(\mathcal{N}, \mathcal{R})$ hálózat megfelelő végrehajtó és felújító szervét. Becsüljük meg annak az E_v eseménynek a valószínűségét, hogy ennek a kombinált szervnek bemenő kábelei ϑ-biztonságosak, de kimenő kábele nem. Tegyük fel, hogy a két bemenő kábel biztonságos: akkor a végrehajtó szervnek legfeljebb $2\vartheta k$ kimenete fertőzött a bemenő kábelek miatt: új fertőzés ezenkívük még új tévedések miatt is megjelenhet. Legyen E_{v1} az esemény, hogy a végrehajtó szerv legalább további $\rho\vartheta k$ kimenetet fertőz meg. Ekkor $\mathbf{P}(E_{v1}) \leqslant (\frac{e\varepsilon}{\rho\vartheta})^{\rho\vartheta k}$, a (4.2) becslést használva. A végrehajtó szerv kimenetei a felújító szerv bemenetei. Ha ezekből legfeljebb $(2+\rho)\vartheta k = \alpha k$ fertőzött, akkor, amennyiben a felújító szerv tökéletesen működik, a fertőzött vezetékek mennyisége $\gamma(2+\rho)\vartheta k$-ra csökkenne. Legyen E_{v2} az esemény, hogy a felújító szerv legalább további $\rho\vartheta k$ vezetéket fertőz



8. ábra. Megbízható hálózat, egy zaj nélküli hálózatból.

meg. Ekkor ugyanazt a becslést használva, $\mathbf{P}(E_{v2}) \leqslant (\frac{e\varepsilon}{\rho\vartheta})^{\rho\vartheta k}$. Ha se E_{v1}, se E_{v2} nem következik be, akkor legfeljebb $\gamma(2+\rho)\vartheta k + \rho\vartheta k \leqslant \vartheta k$ fertőzött vezeték bukkan elő a felújító szervből (lásd (4.3)-t), tehát a kimenő kábel biztonságos. Tehát $E_v \subset E_{v1} \cup E_{v2}$, és így $\mathbf{P}(E_v) \leqslant 2(\frac{e\varepsilon}{\rho\vartheta})^{\rho\vartheta k}$.

Legyenek $V = \{1, \ldots, N\}$ az $\mathcal{N}$ hálózat csúcsai. Mivel az egész $\mathcal{N}'$ hálózat bemenő kábelei biztonságosak, azt az eseményt, hogy található egy nem biztonságos kábel, az $E_1 \cup E_2 \cup \cdots \cup E_N$ esemény tartalmazza: tehát valószínűsége legfeljebb $2N(\frac{e\varepsilon}{\rho\vartheta})^{\rho\vartheta k}$. □

4.4. Végjáték

A 5. tétel bizonyítása. Csak arra az esetre bizonyítjuk a tételt, amikor a számítás egy egyetlen bit kimenetű logikai hálózat. Az általánosítás több bitre egyszerű. A 4.8. lemma olyan $\mathcal{N}'$ hálózatot ad, melynek kimeneti kábele biztonságos, kivéve egy legfeljebb $2N(\frac{e\varepsilon}{\rho\vartheta})^{\rho\vartheta k}$ valószínűségű eseményt. Válasszuk k-t úgy, hogy ez $\leqslant \delta/3$ legyen:

$$k \geqslant \frac{\log(6N/\delta)}{\rho\vartheta \log \frac{\rho\vartheta}{e\varepsilon_0}}. \tag{4.5}$$

Már csak az van hátra, hogy ehhez a kimeneti kábelhez egy kis hálózatot illesszünk, mely a többségi értéket megbízhatóan előhozza belőle. Ez megtehető a 4. tétel segítségével, amely egy $(k+1)^7$ nagyságú úgynevezett „kóda" hálózatot ad $\mathcal{N}'$-hez. Nevezzük a kapott hálózatot $\mathcal{N}''$-nek.

Annak valószínűsége, hogy a kimeneti kábel nem biztonságos, $< \delta/3$. Annak valószínűsége, hogy a kimeneti kábel biztonságos, de a „kóda" hálózat téved, kisebb, mint 2ε. Tehát annak valószínűsége, hogy $\mathcal{N}''$ hibázik, legfeljebb $2\varepsilon + \delta/3 \leqslant \delta$, használva a $\delta \geqslant 3\varepsilon$ feltételt.

Becsüljük meg $\mathcal{N}''$ nagyságát. A (4.5) egyenlőtlenség szerint választhatunk egy $k = O(\log(N/\delta))$ kábelvastagságot. Mivel $|\mathcal{N}'| \leqslant 2kN$,

$$|\mathcal{N}''| \leqslant 2kN + (k+1)^7 = O(N \log(N/\delta)).$$

□

4.9. Példa. *Vegyük a 4.6 példa konstansait, és ϑ-t* (4.4)*-ból: akkor $\varepsilon_0 = 10^{-9}$, $d = 41$, $\gamma = 0.4$, $\rho = 0.14$, $\alpha = 0.15$, $\vartheta = 0.07$, tehát*

$$\frac{1}{\rho\vartheta \ln \frac{\rho\vartheta}{e\varepsilon_0}} \approx 6.75.$$

Ha most k-t a lehető legkisebbre választjuk, akkor $k \approx 6.75 \ln(N/\delta)$. Ha $\delta = 10^{-8}$, $N = 10^{12}$, akkor ez a $k = 323$ kábelvastagságot engedi meg. Ráadásul ehhez az igazán kellemetlen kábelvastagsághoz, a „kóda" hálózat nagysága $(k+1)^7 \approx 4 \cdot 10^{17}$, ami az $\mathcal{N}''$ hálózat egészét dominálja (noha $N \to \infty$ esetén aszimptotikusan elhanyagolható). ♢

Amint a 4.9 példa mutatja, a redundanciának a fenti bizonyításból kiszámolható ára a gyakorlatban elfogadhatatlan. Az $O(\log(N/\delta))$ faktor jól hangzik, mert csak logaritmikus a számítás nagyságához képest, és egy elég nagy többségi kaput választva (41 bemenet), a 6.75 szorzó az $O(\cdot)$-ban ugyancsak nem mutat rosszul; de mégse várnánk, hogy a megbízhatóság ára ilyen nagy legyen.

Mennyire javítható ez a redundancia optimalizálással, vagy más módszerekkel? A 6 gyakorlat azt mutatja, hogy egy valamivel szigorúbb hibamodellben (a hibák függetlenek és azonos valószínűségűek), több véletlenítéssel, valamivel jobb konstansok érhetők el. A 4.1, 4.2 és 4.6 gyakorlatok a „kóda" hálózatot javítgaták. De ezeknek a javításoknak egyike se hozza a redundanciát elfogadható szintre. Még ha eltekintünk is a véletlen választással járó kellemetlenségektől (ezen valamennyire lehet segíteni), maguk a koncentrátorok nagyok és ügyetlenek. A baj valószínűleg azzal van, hogy kiinduló modellnek logikai hálózatokat választottunk. Egy általános logikai hálózatot nem lehet természetes módon nem-konstants nagyságú részegységekre bontani, és így a megbízhatósági problémát modulárisan kezelni.

4.5. Sűrítők konstrukciója

Ez az alfejezet az előzőknél vázlatosabb, és némi lineáris algebrai tudást feltételez.

Megmutattuk, hogy sűrítők léteznek. Milyen költséges egy (d, α, γ, k)-sűrítőt találni, mondjuk a $d = 41$, $\alpha = 0.15$, $\gamma = 0.4$ paraméterekkel, mint a 4.6 példában? Determinisztikus algoritmust használva, végigkereshetnénk a körülbelül d^k páros d-félreguláris gráfot. Ezek mindegyikében végigpróbálhatnánk az összes $\leqslant \alpha k$ nagyságú bemeneti halmazt: mint tudjuk, ezek száma $\leqslant (e/\alpha)^{\alpha k} < 2^k$. Minden részhalmaz ellenőrzésének költsége $O(k)$, tehát a műveletek teljes száma $O(k(2d)^k)$. Bár ez a szám exponenciális k-ban, emlékezzünk rá, hogy hibajavító konstrukciónkban $k = O(\log(N/\delta))$, ahol N a zajmentes hálózat nagysága: a sűrítőkeresés teljes műveletszáma tehát N-ben polinomiális.

A 6. tétel bizonyítása mutatja, hogy egy véletlenül választott d-félreguláris páros gráf nagy valószínűséggel sűrítő. Van tehát egy gyorsabb, randomizált algoritmus sűrítő generálására. Válassz egy véletlen páros gráfot, ellenőrizd, sűrítő-e: ha nem, kezdd előről. Átlagban konstans sok ismétlés után megállhatunk. Ez az algoritmus gyorsabb, de még mindig exponenciális k-ban, mert minden ellenőrzés $\Omega(k(e/\alpha)^{\alpha k})$ műveletbe kerül.

Van-e explicit konstrukció sűrítőre, k-ban exponenciális keresés elkerülésével? A válasz igenlő. De ebben a fejezetben csak azt mutatjuk meg, hogy a sűrítő tulajdonság egy bizonyos lineáris algebrai tulajdonságból következik, amit polinomiális időben ellenőrizni lehet. Ismeretesek explicit módon megadott gráfok, melyek ezzel a tulajdonsággal rendelkeznek. Leginkább ezeket nem sűrítő, hanem *tágító* tulajdonságuk miatt keresik (lásd a 4.3 gyakorlatot).

Ha $\boldsymbol{v}, \boldsymbol{w}$ vektorok, akkor legyen $(\boldsymbol{v}, \boldsymbol{w})$ a skaláris szorzatuk. Egy $2k$ csúcsú d-félreguláris páros multigráf egy $\boldsymbol{M} = (m_{ij})$, *incidencia mátrixszal* definiálható, melyben m_{ij} azon élek száma, melyek a j bemenetet az i kimenethez kötik. Legyen $\boldsymbol{e}$ a csupa egyes $(1, 1, \ldots, 1)^T$ vektor. Ekkor $\boldsymbol{Me} = d\boldsymbol{e}$, tehát $\boldsymbol{e}$ *sajátvektora* az $\boldsymbol{M}$ mátrixnak, a d *sajátértékkel*. Sőt, d az $\boldsymbol{M}$ legnagyobb sajátértéke. Valóban, a $|\boldsymbol{x}|_1 = \sum_i |x_i|$ jelöléssel, minden $\boldsymbol{x} = (x_1, \ldots, x_k)$ sorvektorra $|\boldsymbol{xM}|_1 \leqslant |\boldsymbol{x}|_1$.

7. Tétel. *Legyen G az $\boldsymbol{M}$ mátrix által definiált multigráf. Minden $\gamma > 0$ és*

$$\mu < d\sqrt{\gamma}/2 \tag{4.6}$$

értékre létezik olyan $\alpha > 0$, hogy ha az $\boldsymbol{M}^T\boldsymbol{M}$ mátrix második legnagyobb sajátértéke μ^2, akkor G egy (d, α, γ, k)-sűrítő.

Bizonyítás. Az $\boldsymbol{M}^T\boldsymbol{M}$ mátrix legnagyobb sajátértéke d^2. Mivel szimmetrikus, van ortogonális egység hosszúságú $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_k$ sajátvektorokból álló bázisa, a

$$\lambda_1^2 \geqslant \cdots \geqslant \lambda_k^2$$

sajátértékekkel, ahol $\lambda_1 = d$, $\boldsymbol{e}_1 = \boldsymbol{e}/\sqrt{k}$.

Emlékezzünk, hogy az $\{\boldsymbol{e}_i\}$ ortonormális bázisban minden $\boldsymbol{f}$ vektort az $\boldsymbol{f} = \sum_i (\boldsymbol{f}, \boldsymbol{e}_i)\boldsymbol{e}_i$ módon fejezhetünk ki. Tetszőleges $\boldsymbol{f}$ vektorra, az $|\boldsymbol{M}\boldsymbol{f}|^2$ értéket a következőképpen becsülhetjük.

$$\begin{aligned}|\boldsymbol{M}\boldsymbol{f}|^2 = (\boldsymbol{M}\boldsymbol{f}, \boldsymbol{M}\boldsymbol{f}) = (\boldsymbol{f}, \boldsymbol{M}^T\boldsymbol{M}\boldsymbol{f}) &= \sum_i \lambda_i^2 (\boldsymbol{f}, \boldsymbol{e}_i)^2 \\ &\leqslant d^2(\boldsymbol{f}, \boldsymbol{e}_1)^2 + \mu^2 \sum_{i>1} (\boldsymbol{f}, \boldsymbol{e}_i)^2 \leqslant d^2(\boldsymbol{f}, \boldsymbol{e}_1)^2 + \mu^2(\boldsymbol{f}, \boldsymbol{f}) \\ &= d^2(\boldsymbol{f}, \boldsymbol{e})^2/k + \mu^2(\boldsymbol{f}, \boldsymbol{f}).\end{aligned}$$

Legyen most $A \subset \{1, \dots, k\}$ egy αk nagyságú halmaz, és legyen $\boldsymbol{f} = (f_1, \dots, f_k)^T$, ahol $f_j = 1$, ha $j \in A$, és 0 különben. Ekkor a $\boldsymbol{M}\boldsymbol{f}$ vektor i-edik koordinátája azon élek d_i számát adja, melyek az A halmazból az i csúcsba érkeznek. Továbbá, $(\boldsymbol{f}, \boldsymbol{e}) = (\boldsymbol{f}, \boldsymbol{f}) = |A|$, az A elemszáma. Azt kapjuk, hogy

$$\sum_i d_i^2 = |\boldsymbol{M}\boldsymbol{f}|^2 \leqslant d^2(\boldsymbol{f}, \boldsymbol{e})^2/k + \mu^2(\boldsymbol{f}, \boldsymbol{f}) = d^2\alpha^2 k + \mu^2 \alpha k,$$

$$k^{-1} \sum_i (d_i/d)^2 \leqslant \alpha^2 + (\mu/d)^2 \alpha.$$

Tegyük fel, hogy $c\alpha k$ olyan i csúcs van, melyre $d_i > d/2$, akkor ebből

$$c\alpha \leqslant 4(\mu/d)^2 \alpha + 4\alpha^2$$

következik. Ilyen módon, mivel (4.6) miatt $4(\mu/d)^2 < \gamma$, ha α elég kicsi, akkor $\mathcal{M}$ egy (d, α, γ, k)-sűrítő. □

A (4.6) feltétel enyhíthető. Valójában elegendő olyan gráfokat keresni, nagy k-ra, melyekben $\mu/d < c < 1$, ahol d, c konstansok. Ehhez definiáljuk két $2k$ elemű páros multigráf szorzatát a megfelelő mátrixok szorzatával.

Tegyük fel, $\boldsymbol{M}$ szimmetrikus: akkor második legnagyobb sajátértéke μ, és $\boldsymbol{M}^r$ két legnagyobb sajátértékének aránya $(\mu/d)^r$. Tehát elég nagy r-re a $\boldsymbol{M}^r$ mátrix ki fogja elégíteni a (4.6) feltételt. Sajnos a hatványozás a d fokszámot megnöveli, valószínűleg méginkább eltávolítva minket a gyakorlati megvalósíthatóságtól.

Azt találtuk, hogy létezik konstrukció egy kívánt paraméterekkel rendelkező sűrítőre, ha csak találunk tetszőleges nagy $2k$ nagyságú multigráfokat, szimmetrikus $\boldsymbol{M}_k$ mátrixszal, melyekben a két legnagyobb sajátérték aránya egy k-tól független $c < 1$ konstans alatt van. Ilyen multigráfokra különböző konstrukciók léteznek (a történeti visszatekintésben adunk néhány utalást ezekre). A sajátérték-arány becslése egyik esetben se nagyon egyszerű.

4.6. Gyakorlatok

4.1. Gyakorlat. Az 5. tétel bizonyítása egy $(k+1)^7$ nagyságú „kóda" hálózatot használ. Miután bebizonyítottuk, természetesen ezt a tételt a befejező többségi érték kiszámítására is használhatjuk: ez a „kóda" hálózat nagyságát $O(k \log k)$-ra csökkentené. Próbáljuk ki ezt a fent használt numerikus példákon, hogy lássuk, lényeges javuláshoz vezet-e.

4.2. Gyakorlat. A 6. tétel bizonyítása olyan k-bemenetű, sűrítő-tulajdonságú páros gráfokat is nyújt, melyeknek csak $k' < 0.8k$ kimenete van. Az 5. tétel bizonyításában szereplő „kóda" hálózat talán csökkenthető több ilyen sűrítő egymás után kapcsolásával. Próbáljuk ezt ki, annak szem előtt tartásával, hogy k csökkenésekor a (4.2) egyenlőtlenség „exponenciális" hibabecslése gyengül.

4.3. Gyakorlat. Egy d-félreguláris páros multigráfot, melyben a k bemenet halmaza A és a k kimenet halmaza B akkor nevezünk (d, α, λ, k)-*tágítónak (expandernek)*, ha a következő tulajdonsággal bír: minden $E \subset A$ halmazra, ha $|E| \leqslant \alpha k$, akkor B-nek legalább $\lambda \alpha k$ eleme van E-vel összekötve. Bizonyítsuk a következő tételt, mely a 6. tétel analógja: Minden $\lambda < d$ esetén létezik egy olyan α, hogy minden $k > 0$-ra van (d, α, λ, k)-tágító. [Útmutatás: A 6. tétel bizonyításához hasonlóan mutassuk meg, hogy egy véletlen d-félreguláris multigráf nagy valószínűséggel tágító.]

4.4. Gyakorlat. Egy zajos logikai hálózatban legyen $F_v = 1$, ha a v csúcs kapuja téved, és 0 különben. Továbbá legyen $T_v = 1$, ha v fertőzött, és 0 különben. Tegyük fel, hogy az F_v valószínűségi változók eloszlása nem függ a bemeneti Boole-vektortól. Mutassuk meg, hogy akkor a T_v valószínűségi változók együttes eloszlása is független a bemenetvektortól.

4.5. Gyakorlat. Ez a gyakorlat a 3.1. gyakorlat eredményét terjeszti ki véletlen bemenetvektorokra: megmutatja, hogy ha egy véletlen bemenetvektorban csak kevés hiba van, akkor a 2.5. gyakorlat $\mathcal{M}_3^r$ iterált többségi szavazása még mindig működhet rajta, amennyiben a bemeneti vezetékeket véletlenül átrendezzük. Legyen $k = 3^r$, és legyen $\boldsymbol{j} = (j_1, \ldots, j_k)$ egy $j_i \in \{1, \ldots, k\}$ egész számokból álló vektor. A $C(\boldsymbol{j})$ logikai hálózatot a következőképpen definiáljuk. Ez a hálózat veszi az $\boldsymbol{x} = (x_1, \ldots, x_k)$ bemenetvektort, kiszámolja az $\boldsymbol{y} = (y_1, \ldots, y_k)$ vektort, ahol $y_i = x_{j_i}$ (más szóval, egyszerűen egy vezetéket visz a j_i bemenetcsúcstól a „közbülső" i csúcshoz), majd beadja $\boldsymbol{y}$-t az $\mathcal{M}_3^r$ hálózatba.

Jelöljük $C(\boldsymbol{j})$ (esetleg véletlen) kimenetbitjét Z-vel. Minden rögzített $\boldsymbol{x}$ bemenetvektorhoz, feltéve, hogy többségi kapuink egymástól függetlenül $\leqslant \varepsilon \leqslant \alpha/2$ valószínűséggel tévednek, legyen $q(\boldsymbol{j}, \boldsymbol{x}) := \mathbf{P}[\, Z = 1 \,]$. Tegyük fel, hogy a bemenet

a (nem feltétlenül független) Boole valószínűségi változók egy $\boldsymbol{X} = (X_1, \dots, X_k)$ vektora, melyre $p(\boldsymbol{x}) := \mathbf{P}[\,\boldsymbol{X} = \boldsymbol{x}\,]$. Az $|\boldsymbol{X}| = \sum_i X_i$ jelöléssel, tegyük fel, hogy $\mathbf{P}[\,|\boldsymbol{X}| > \alpha k\,] \leqslant \rho < 1$. Bizonyítsuk, hogy a $\boldsymbol{j}$ vektornak létezik olyan választása, melyre

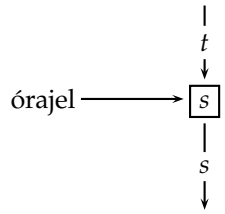
$$\sum_{\boldsymbol{x}} p(\boldsymbol{x}) q(\boldsymbol{j}, \boldsymbol{x}) \leqslant \rho + \max\{10\varepsilon,\ 0.3(\alpha/0.3)^{2^k}\}.$$

Ez a választás függhet az $\boldsymbol{X}$ véletlen vektor eloszlásától. [Útmutatás: Válasszuk a $\boldsymbol{j}$ vektort (és ezzel a $C(\boldsymbol{j})$ hálózatot) véletlenül, azaz mint egy $\boldsymbol{J} = (J_1, \dots, J_k)$ véletlen vektort, ahol a J_i valószínűségi változók függetlenek, és egyenletes eloszlásúak $\{1, \dots, k\}$ felett, és legyen $s(\boldsymbol{j}) := \mathbf{P}[\,\boldsymbol{J} = \boldsymbol{j}\,]$. Bizonyítsuk a következőt:

$$\sum_{\boldsymbol{j}} s(\boldsymbol{j}) \sum_{\boldsymbol{x}} p(\boldsymbol{x}) q(\boldsymbol{j}, \boldsymbol{x}) \leqslant \rho + \max\{10\varepsilon,\ 0.3(\alpha/0.3)^{2^k}\}.$$

Ehhez, cseréljük fel a $\boldsymbol{x}$ és $\boldsymbol{j}$ szerinti átlagolást, majd vegyük észre, hogy $\sum_{\boldsymbol{j}} s(\boldsymbol{j}) q(\boldsymbol{j}, \boldsymbol{x})$ a $Z = 1$ valószínűsége, ha a J_i „drótokat" véletlenül, „menetközben" választjuk a hálózat számolása során.]

4.6. Gyakorlat. A 4.4. gyakorlat jelölésével tegyük fel, mint ott, hogy az F_v valószínűségi változók eloszlása nem függ a bemeneti Boole-vektortól. Vegyük a 4.7. definícióban bevezetett $\mathrm{Cab}(\mathcal{N}, \mathcal{R})$ logikai hálózatot, és definiáljuk a $\boldsymbol{T} = (T_1, \dots, T_k)$ véletlen Boole-vektort, melyben $T_i = 1$, ha az i-edik kimeneti csúcs fertőzött. Alkalmazzuk a 4.5. gyakorlatot annak megmutatására, hogy létezik egy $C(\boldsymbol{j})$ hálózat, melyet a kimenő csúcsokhoz illeszthetünk, és amely a „kóda" hálózat szerepét játszhatja az 5. tétel bizonyításában. A $C(\boldsymbol{j})$ nagysága csak lineáris k-ban, nem pedig $(k+1)^7$, mint abban a bizonyításban. De a hibák eloszlásáról kicsit többet tettünk fel, ezenkívül a $\boldsymbol{j}$ „drótozás" függ a $\mathrm{Cab}(\mathcal{N}, \mathcal{R})$ hálózattól.



9. ábra. Bit-tároló elem

5. A megbízható információtárolás problémája

5.1. Ütemezett hálózatok

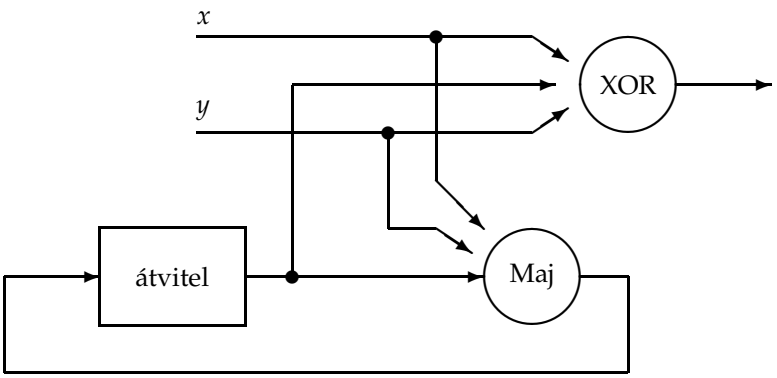
A közönséges számolásoknak egy eleme feltűnően hianyzik a fent leírt logikai hálózat modellből: az *ismétlések*. Egyes műveleteket megismételni csak akkor lehet, ha a számoló egységek munkáját időzítjük, és az egymást követő lépések között a részeredményeket *tároljuk*. Pillantsunk megint a hálózattervező rajzaira: olyan egységeket is látni fogunk, mint a 9. ábrán. Ezeknek egy bemenő éle van, és nincs logikai művelet hozzájuk rendelve; *bit-tárolóknak* fogjuk őket hívni. A bit-tárolót egy központi órajel-generátor vezéreli (ez nem látható az ábrán). Minden órajelre a bemenő élen érkező logikai érték átugrik a kimenő élekre, és a „tárolóban marad". A 10. ábra mutatja bit-tárolók lehetséges felhasználását egy hálózatban.

5.1. Definíció. Egy *ütemezett hálózatot* a Q teljes bázis felett formálisan a logikai hálózatokhoz hasonlóan adunk meg (lásd (2.1)). A hálózat ugyancsak hasonlóan definiál egy $G = (V, E)$ gráfot is. Emlékezzünk, hogy a csúcsokat az $1, \ldots, N$ természetes számokkal azonosítottuk. Minden v nem-bemeneti csúcshoz vagy egy b_v kaput rendelünk, mint azelőtt, vagy egy *bit-tárolót*: ez esetben $k_v = 1$ (csak egy „argumentum" van). Nem kívánjuk, hogy a gráf aciklikus legyen, de megkívánjuk, hogy minden irányított ciklus (ha van ilyen) legalább egy bit-tárolón haladjon át.

A hálózat működését a $t = 0, 1, 2, \ldots$-el jelzett *óraciklusok* sorozatára bonthatjuk fel. A t-edik óraciklus bemenet-vektorát jelöljük $\boldsymbol{x}^t = (x_1^t, \ldots, x_n^t)$-vel, a bit-tárolók állapotát $\boldsymbol{s}^t = (s_1^t, \ldots, s_k^t)$-vel, és a kimenet-vektort $\boldsymbol{y}^t = (y_1^t, \ldots, y_m^t)$-vel.

A hálózatnak az a része, mely a bemenetektől a bit-tárolókhoz megy, két Boole-vektorfüggvényt definiál: $\lambda : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^m$ és $\tau : \{0,1\}^k \times \{0,1\}^n \to \{0,1\}^k$. Az ütemezett hálózat működését a következő egyenletek írják le (lásd a 11. ábrát, mely nem mutatja a bemeneteket és kimeneteket).

$$\boldsymbol{y}^t = \lambda(\boldsymbol{s}^t, \boldsymbol{x}^t), \quad \boldsymbol{s}^{t+1} = \tau(\boldsymbol{s}^t, \boldsymbol{x}^t). \tag{5.1}$$



10. ábra. Egy hálózat része, mely két bináris szám, x és y, összegét számítja ki. Az x és y számjegyeit a legkisebb helyiértéktől kezdve tápláljuk be a bemeneteken. Az összeg számjegyei a kimeneti élen bukkannak elő. Egy bit-tároló őrzi az átviteli számjegyet.

♢

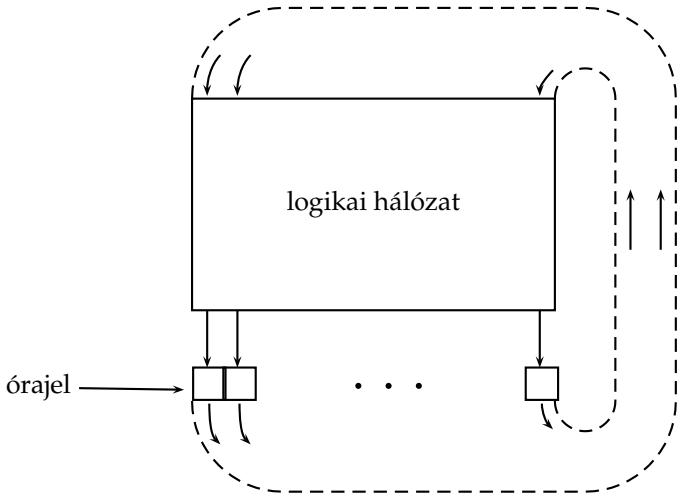
Gyakran, a hálózat számolása folyamán nincsenek kimenetek és bemenetek, ezért az (5.1) egyenleteket így egyszerűsíthetjük:

$$\boldsymbol{s}^{t+1} = \tau(\boldsymbol{s}^t). \tag{5.2}$$

Hogyan használjunk egy ilyen egyenlettel leírt ütemezett hálózatot számolásra? Valamilyen kezdeti értékeket írunk a bit-tárolókba, majd a logikai értékek hozzárendelését továbbvisszük a logikai kapukat használva, az adott óracikluson belül. Ezután küldünk egy órajelet a memóriába (a bit-tárolóknak), ez erre új értékeket ír kimeneti éleire (melyek azonosak a hálózat bemeneti éleivel). Ezután kiszámoljuk az új hozzárendelést, és így tovább.

Hogyan számolhatunk ki egy $f(x)$ *függvényt* egy ilyen hálózat segítségével? Itt egy lehetséges konvenció. Beírjuk az x bemenetet (csak az első lépésben), aztán járatjuk a hálózatot, amíg csak egy extra kimenő élen nem jelzi, hogy a többi kimenő él tartalmazza a várt $f(x)$ eredményt.

5.2. Példa. *Ez a példa a fentitől eltérő konvenciót használ: a hálózat minden lépésben új bemenő biteket kap, és az eredményt folyamatosan szolgáltatja. A 10. ábra bináris összeadójában legyen u^t és v^t a két bemenő bit a t-edik ciklusban, legyen c^t az átvitel, és w^t*



11. ábra. Egy „számítógép” memóriából (bit-tárolókból) áll, és egy azt vezérlő logikai hálózatból. Egy *számítás nagyságát* a számítógép-nagyság és a lépésszám szorzatával definiálhatjuk.

a kimenet ugyanabban a ciklusban. Az (5.1) *egyenletek most a következő formát kapják:*

$$w^t = u^t \oplus v^t \oplus c^t, \quad c^{t+1} = \mathrm{Maj}(u^t, v^t, c^t),$$

ahol Maj *a többségi szavazás művelete.* ♢

5.2. Információtárolás

Az ütemezett hálózat érdekes párhuzamos számítógép modell, de adjunk most csak egy olyan feladatot neki, amely triviális a zajtalan esetben: információtárolást. Bizonyos mennyiségű információt szeretnénk tárolni olyan módon, hogy egy idő után elő tudjuk venni, annak ellenére, hogy a hálózatban tévedések fordulnak elő. Ebben az esetben a fent bevezetett τ átmenetfüggvény nem lehet egyszerűen az identitás: *hibajavító* műveleteket kell végrehajtania. Természetesen adódik, hogy az előző alfejezetben tárgyalt *felújító szerveket* használjuk. Tegyük fel, hogy k memóriacellát (bit-tárolót) szentelünk egy bit információ tárolására. Nevezzük ezt a k-ast *biztonságosnak*, ha a helyes értéktől eltérő memóriacellák száma valamilyen ϑk küszöb alatt van.

Legyen a hálózat maradék része egy (d, α, γ, k)-sűrítőre épített felújító szerv, melyben $\alpha = 0.9\vartheta$. Tegyük fel, hogy a bemenő kábel biztonságos. Ekkor annak

valószínűsége, hogy az átmenet után a kimenő kábel (és így az új állapot) nem biztonságos, $O(e^{-ck})$ lesz valami c konstanssal. Tegyük fel, hogy a hálózatot t lépésen át működtetjük. Ekkor annak valószínűsége, hogy az állapot nem biztonságos ezen lépések valamelyikében, $O(te^{-ck})$. Ez kicsi, amennyiben t lényegesen kisebb, mint e^{ck}. Ha m bit információt tárolunk, akkor annak valószínűsége, hogy ezen bitek bármelyike valamelyik lépésben elveszti biztonságát, $O(mte^{-cm})$.

Ha szigorúvá akarjuk tenni tárgyalásunkat, az ütemezett hálózatokra is hibamodellt kell bevezetni. Mivel csak olyan egyszerű τ átmenetfüggvényeket fogunk vizsgálni, melyekben csak egyetlen számolási lépés történik a t és $t+1$ időpontok között (akárcsak a fenti többségi szavazásban), modellünk is egyszerű lesz.

5.3. Definíció. Tekintsünk egy ütemezett hálózatot, melyet az (5.2) egyenlet ad meg: állapotát ekkor minden $t = 0, 1, 2, \ldots$ időpontban az $\boldsymbol{s}^t = (s_1^t, \ldots, s_n^t)$ bitvektor írja le. Legyen $\boldsymbol{Y}^t = (Y_1^t, \ldots, Y_n^t)$ a véletlen bitvektorok sorozata $t = 0, 1, 2, \ldots$-re. A (3.1) egyenlethez hasonlóan legyen

$$Z_{i,t} = \tau(\boldsymbol{Y}^{t-1}) \oplus Y_i^t. \tag{5.3}$$

Ekkor $Z_{i,t} = 1$ azt jelenti, hogy az (i,t) téridő pontban *tévedés* történik. Az $\{\boldsymbol{Y}^t\}$ sorozatot *ε-megengedettnek* nevezzük, ha a (3.2) egyenlőtlenség áll a $t = 0$ után bekövetkező téridő pontok minden véges C halmazára. ♢

Az imént említett felújító-szerves konstrukcióval m bit információt T lépésen át lehet tartani, ha

$$O(m \log(mT)) \tag{5.4}$$

memóriacellát használunk. Pontosabban, az $\boldsymbol{Y}^T$ kábel nagy valószínűséggel biztonságos lesz minden megengedett $\boldsymbol{Y}^t$ $(t = 0, \ldots, T)$ evolúcióban. Lehet ezen javítani?

A megbízható információtárolás feladata rokon az *információ-továbbítás* feladatával: egy *feladó* egy $\boldsymbol{x}$ *üzenetet* át akar juttatni egy *zajos csatornán* egy *címzettnek*. Csak éppen itt feladó és címzett ugyanaz a személy, és a zajos csatorna csak az idő folyása. Most ezért először bevezetjük a megbízható információátvitel néhány alapfogalmát, azután pedig alkalmazzuk ezeket egy adattároló rendszer készítésére, mely gazdaságosabb, mint a most látott naív ismétléses megoldás.

5.3. Hibajavító kódok

5.3.1. Hibafelismerés

Információvédelem céljára az ismétléses módszernél hatékonyabban is használhatunk redundanciát. Még azt is megpróbálhatjuk, hogy az üzenethez

csak egyetlen további bitet teszünk hozzá. Legyen $\boldsymbol{x} = (x_1, \ldots, x_6)$, $(x_i \in \{0,1\})$ a védeni kívánt szó. Képezzük az

$$x_7 = x_1 \oplus \cdots \oplus x_6$$

hibajavító bitet. Például, $\boldsymbol{x} = 110010$, $\boldsymbol{x}' = 1100101$. Ha $\boldsymbol{x}' = (x_1, \ldots, x_7)$ *kódszavunkat* zajnak tesszük ki, egy új szóvá, $\boldsymbol{y}$-ná változik át. Ha $\boldsymbol{y}$ az $\boldsymbol{x}'$-től csak egyetlen megváltoztatott bitben különbözik, akkor ezt *felismerjük*, mert szavunk megszegi az

$$y_1 \oplus \cdots \oplus y_7 = 0$$

hibaellenőrző összefüggést. A hibát nem tudjuk kijavítani, mert nem tudjuk, melyik bit romlott el.

5.3.2. Egyetlen hiba javítása

Ha *javítani* is akarunk hibákat, akkor több hibaellenőrző bitet kell csatolni az üzenethez. Próbálkozhatunk két további bit hozzáadásával:

$$\begin{aligned} x_8 &= x_1 \oplus x_3 \oplus x_5, \\ x_9 &= x_1 \oplus x_2 \oplus x_5 \oplus x_6. \end{aligned}$$

Ekkor a romlatlan $\boldsymbol{y}$ szónak a következő hibaellenőrző összefüggéseket kell teljesíteni:

$$\begin{aligned} y_1 \oplus \cdots \oplus y_7 &= 0, \\ y_1 \oplus y_3 \oplus y_5 \oplus y_8 &= 0, \\ y_1 \oplus y_2 \oplus y_5 \oplus y_6 \oplus y_9 &= 0, \end{aligned}$$

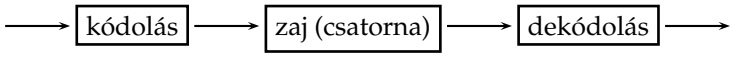
vagy mátrix jelöléssel $\boldsymbol{Hy} \bmod 2 = 0$, ahol

$$\boldsymbol{H} = \begin{pmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 \end{pmatrix} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_9).$$

Észrevehetjük, hogy $\boldsymbol{h}_1 = \boldsymbol{h}_5$. A $\boldsymbol{H}$ mátrixot *hibaellenőrző mátrixnak*, vagy *párosság-ellenőrző mátrixnak* nevezik. A hibaellenőrző összefüggéseket más módon a következő formában írhatjuk:

$$y_1\boldsymbol{h}_1 \oplus \cdots \oplus y_5\boldsymbol{h}_5 \oplus \cdots \oplus y_9\boldsymbol{h}_9 = 0.$$

Most, még ha $\boldsymbol{y}$ csak egyetlen helyen van is elrontva, sajnos még mindig nem javítható: mivel $\boldsymbol{h}_1 = \boldsymbol{h}_5$, a hiba lehet az 1. vagy az 5. helyen, nem tudnánk ezt a két



12. ábra. Adatátvitel zajos csatornán

esetet megkülönböztetni. Ha viszont $\boldsymbol{H}$ hibaellenőrző mátrixunkat úgy választjuk, hogy a $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots$ oszlopvektorok *mind különbözők* (persze nullától is), akkor ha csak egy hiba van, az javítható. Valóban, ha a hiba a 3. helyen van, akkor

$$\boldsymbol{H}\boldsymbol{y} \bmod 2 = \boldsymbol{h}_3.$$

Mivel a $\boldsymbol{h}_1, \boldsymbol{h}_2, \ldots$ vektorok mind különbözők, ha a $\boldsymbol{h}_3$ vektort látjuk, következtethetjük, hogy az y_3 bit romlott el. Ezt a kódot a *Hamming-kódnak* nevezik. Például, a következő hibaellenőrző mátrix definiálja a 7 nagyságú Hamming-kódot:

$$\boldsymbol{H} = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{pmatrix} = (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_7). \tag{5.5}$$

Általában, ha s hibaellenőrző bitünk van akkor kódunk nagysága $2^s - 1$ lehet, így az információtárolás („üzenet") számára fennmaradó bitek, az *információ-bitek* száma $m = 2^s - s - 1$. Tehát ahhoz, hogy m információbitet egyetlen hibától megvédjen, a Hamming-kód $\approx \log m$ további, hibajavító bitet igényel. Ez sokkal jobb, mint minden bitet 3-szor ismételni.

5.3.3. Kódok

Foglaljuk össze a hibajavító felállást általánosabb formában. Zaj elleni védelem céljából a feladó *kódolja* az $\boldsymbol{x}$ *üzenetet* a ϕ_* *kódoló függvény* segítségével egy $\phi_*(\boldsymbol{x})$ hosszabb sorozatba, amelyről az egyszerűség kedvéért feltesszük, hogy bináris. Ezt a *kódszót* a zaj egy $\boldsymbol{y}$ sorozattá változtatja. A címzett megkapja $\boldsymbol{y}$-t és a ϕ^* *dekódoló függvényt* alkalazza rá.

5.4. Definíció. A $\phi_* : \{0,1\}^m \to \{0,1\}^n$ és $\phi^* : \{0,1\}^n \to \{0,1\}^m$ függvényből álló párt *kódnak* nevezzük, ha minden $\boldsymbol{x} \in \{0,1\}^m$-re $\phi^*(\phi_*(\boldsymbol{x})) = \boldsymbol{x}$ teljesül. Az $\boldsymbol{x} \in \{0,1\}^m$ szavakat *üzeneteknek* nevezzük, az $\boldsymbol{y} = \phi_*(\boldsymbol{x}) \in \{0,1\}^n$ formájú szavakat *kódszavaknak*. (Néha a kódszavak halmazát egymagában is kódnak nevezik.) Minden $\boldsymbol{x}$ üzenethez, a $C_{\boldsymbol{x}} = \{\, \boldsymbol{y} : \phi^*(\boldsymbol{y}) = \boldsymbol{x} \,\}$ szóhalmazt az $\boldsymbol{x}$ *dekódoló halmazának*

nevezzük. (Természetesen a dekódoló halmazok diszjunktak.) Az

$$R = m/n$$

számot a kód *sebességének (rátájának)* nevezzük.

Azt mondjuk, hogy kódunk *t hibát kijavít*, ha minden $\boldsymbol{x} \in \{0,1\}^m$ üzenetre, ha az $\boldsymbol{y} \in \{0,1\}^n$ megkapott szó a $\phi_*(\boldsymbol{x})$ kódszótól legfeljebb t helyen különbözik, akkor $\phi^*(\boldsymbol{y}) = \boldsymbol{x}$. ♢

Ha a sebesség R, akkor az n-bit kódszavak Rn bit hasznos információt hordoznak. A dekódoló halmazok nyelvén, a kód t hibát javít, ha minden dekódoló $C_{\boldsymbol{x}}$ halmaz minden olyan szót tartalmaz, mely a $\phi_*(\boldsymbol{x})$ kódszótól legfeljebb t jelben különbözik (ezeknek a szavaknak a halmaza egyfajta t sugarú „gömb”).

A Hamming-kód egy hibát javít ki, és sebessége közel 1. A hibajavító kódokkal kapcsolatos egyik fontos kérdés az, mennyire kell a sebességet csökkenteni, ha több hibát akarunk kijavítani.

A kód-jelölések használatával most megfogalmazhatjuk ennek a fejezetnek az információtárolásra vonatkozó fő eredményét.

8. Tétel (Információtárolás hálózatban). *Léteznek $\varepsilon, b, R > 0$ konstansok, a következő tulajdonsággal. Minden m-re, minden $n \geqslant m/R$-re létezik egy (ϕ_*, ϕ^*) kód m üzenethosszal és n kódszóhosszal, és egy $O(n)$ nagyságú $\mathcal{N}$ ütemezett logikai hálózat n bemenettel és n kimenettel, és a következő képességgel: Tegyük fel, hogy a 0-dik időpontban, a hálózat memóriacellái egy tetszőleges $\boldsymbol{Y}_0 = \phi_*(\boldsymbol{x})$ kódszót tartalmaznak. Tegyük fel továbbá, hogy a hálózat $\boldsymbol{Y}_1, \boldsymbol{Y}_2, \ldots, \boldsymbol{Y}_t$ által leírt működése ε-megengedett. Ekkor $\mathbf{P}[\,\phi^*(\boldsymbol{Y}_t) \neq \boldsymbol{x}\,] < te^{-bn}$.*

A tétel azt mutatja, hogy lehetséges m bit információt t lépésen át tárolni, egy

$$O(\max(\log t, m))$$

nagyságú ütemezett hálózatban. Amíg a tárolási t idő az exponenciális e^{cm} korlát alatt marad egy bizonyos c konstansra, addig ez a hálózatnagyság csak konstansszor nagyobb, mint a tárolt információ m mennyisége. (Amikor külön felújító szervet használtunk minden bithez, akkor további $\log m$ szorzóra volt szükség: lásd (5.4).) A tétel hallgat arról, milyen nehéz kiszámolni a $\phi_*(\boldsymbol{x})$ kódszót az induláskor, és milyen nehéz a $\phi^*(\boldsymbol{Y}_t)$ dekódolás a végén. Sőt, ezt a két műveletet is kívánatos lenne zajtűrő módon kivitelezni. Mindkét feladat megoldható, de a jelen fejezetben magára az információtárolásra koncentrálunk.

5.3.4. Lineáris algebra

Miután többet is fogunk bitmátrixokkal foglalkozni, kényelmes bevezetni az

$$\mathbb{F}_2 = (\{0,1\}, +, \cdot)$$

algebrai struktúrát, ami egy kételemű *test*. Az összeadást és szorzást $\mathbb{F}_2$-ben modulo 2 definiáljuk (persze ez a szorzást illetően nem változás). Ugyancsak érdemes a bináris sorozatok $\{0,1\}^n$ halmazát az n-dimenziós vektortér $\mathbb{F}_2^n$ struktúrájával felruházni. Az elemi lineáris algebra legtöbb tétele és algoritmusa tetszőleges test felett is érvényes: többek között, definiálhatjuk egy mátrix sor-rangját, mint a lineárisan független sorok maximális számát, és az oszlop-rangot hasonlóan. Ekkor tétel az, hogy a sor-rang egyenlő az oszlop-ranggal. Mostantól, biteken és bitvektorokon végzett algebrai műveletek esetében $+$ jelet írunk $\oplus$ helyett, amikor ez nem okozhat félreértést. Helymegtakarítás céljából, oszlopvektorokat néha vízszintesen fogunk írni: azaz

$$\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix} = (x_1, \ldots, x_n)^T,$$

ahol $\boldsymbol{A}^T$ jelöli az $\boldsymbol{A}$ mátrix transzponáltját. Az $\mathbb{F}_2^r$ vektortér feletti identitás mátrixot

$$\boldsymbol{I}_r$$

jelöli.

5.3.5. Lineáris kódok

Általánosítsuk a Hamming-kód gondolatát.

5.5. Definíció. Egy (ϕ_*, ϕ^*) kód m üzenethosszal és n kódszóhosszal *lineáris*, ha az üzenet- és kódvektorokat az $\mathbb{F}_2$ test feletti vektoroknak tekintve, a kódoló függvényt a

$$\phi_*(\boldsymbol{x}) = \boldsymbol{G}\boldsymbol{x}$$

képlet adja meg, ahol $\boldsymbol{G}$ egy $m \times n$ mátrix, amit a kód *generáló mátrixának* neveznek. Az m szám a kód *információ-bitjeinek* száma, a

$$k = n - m$$

szám pedig a *hibaellenőrző* biteké. ♢

5.6. Példa. *A* $\boldsymbol{H}$ *mátrixot* (5.5)*-ben írhatjuk így:* $\boldsymbol{H} = (\boldsymbol{K}, \boldsymbol{I}_3)$*, ahol*

$$\boldsymbol{K} = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{pmatrix}.$$

Ekkor a hibaellenőrző összefüggés így írható:

$$\boldsymbol{y} = \begin{pmatrix} \boldsymbol{I}_4 \\ -\boldsymbol{K} \end{pmatrix} \begin{pmatrix} y_1 \\ \vdots \\ y_4. \end{pmatrix}.$$

Amint látszik, az $y_1, \ldots, y_4$ *biteket tekinthetjük a kód üzenet-bitjeinek, vagy „információ-bitjeinek", és ezzel a Hamming-kód lineáris kóddá válik, az* $(\boldsymbol{I}_4, -\boldsymbol{K})^T$ *generáló mátrixszal. (Persze,* $-\boldsymbol{K} = \boldsymbol{K}$ *az* $\mathbb{F}_2$ *test felett.)* ♢

A következő állítás rutin lineáris algebrai módszerekkel bizonyítható, és általánosítja a hibaellenőrző mátrix és generáló mátrix közötti kapcsolatot, amit az 5.6 példában láttunk.

5.7. Állítás. *Legyen* $k, m > 0$*, és* $n = m + k$*.*

(a) *Minden, az* $\mathbb{F}_2$ *test feletti* $n \times m$*,* m *rangú* $\boldsymbol{G}$ *mátrixhoz létezik egy* $k \times n$*,* k *rangú* $\boldsymbol{H}$ *mátrix, melyre*

$$\{\, \boldsymbol{G}\boldsymbol{x} : \boldsymbol{x} \in \mathbb{F}_2^m \,\} = \{\, \boldsymbol{y} \in \mathbb{F}_2^n : \boldsymbol{H}\boldsymbol{y} = 0 \,\}. \tag{5.6}$$

(b) *Minden, a* $\mathbb{F}_2$ *test feletti* $k \times n$*,* k *rangú* $\boldsymbol{H}$ *mátrixhoz létezik egy* $n \times m$*,* m *rangú* $\boldsymbol{G}$ *mátrix, mely az* (5.6) *egyenlőséget teljesíti.*

5.8. Definíció. Jelölje $|\boldsymbol{x}|$ egy $\boldsymbol{x}$ vektor nemzéró elemeinek számát: ezt az $\boldsymbol{x}$ *súlyának* is fogjuk hívni. ♢

A következőkben kényelmes lesz kódjainkat inkább a $\boldsymbol{H}$ hibaellenőrző mátrixból kiindulva megadni. Ha a mátrix rangja k, akkor a kód sebessége

$$R = 1 - k/n.$$

Az oszlopok bármely lineárisan független S részhalmazát rögzíthetjük, és az $i \in S$ indexeket nevezhetjük *hibaellenőrző biteknek*; az $i \notin S$ indexek lesznek ekkor az *információ-bitek*. (Az 5.6 példában $S = \{5, 6, 7\}$.) De fontos operációkat lehet végrehajtani a kódon anélkül, hogy bitjeit szétválasszuk információ-bitekre és hibaellenőrző bitekre.

5.4. Frissítők

Egyetlen hiba kijavítása nem volt túl nehéz; sokkal nehezebb hasonló elrendezést találni 2 hiba kijavítására. Pedig n bit tárolásánál általában εn (azaz sokkal több, mint 2) bitünk romlik el minden lépésben. Vannak leleményes és meglehetősen hatékony kódok (n-től független) pozitív sebességgel, melyek ennyi hibát is kijavítanak. De az információ-tárolóban a hibajavító mechanizmus maga is zajban fog működni, ezért valami különösen egyszerűt keresünk. Szerencsére nem muszáj minden hibát kijavítani: elég, ha alaposan csökkentjük a számukat, akárcsak a felújító szervben, amit megbízható logikai hálózatokhoz használtunk korábban.

Az egyszerűség kedvéért hálózatunk kapuiként olyan Boole-függvényeket is megengedünk, melyeknek változó-száma nagy (bár konstans). Cserébe viszont logikai hálónk mélysége csak 1 lesz, a 4. fejezet felújító szervéhez hasonlóan. Minden kapu kimenete egy memóriacella (bit-tároló) bemenete. Az egyszerűség kedvéért a kaput és a memóriacellát azonosítjuk, és *sejtnek* hívjuk. Minden órajelre, a sejt leolvassa bemeneteit a többi sejttől, egy Boole függvényt alkalmaz rájuk, és tárolja az eredményt (a következő órajelig). Csakhogy most az egyes sejtek által kiszámolt Boole-függvény kissé bonyolultabb lesz, mint a korábbi egyszerű többségi szavazás a bemenő értékek között.

Pontosabban, felújító műveleteinket egy bizonyos $k \times n$ méretű $\boldsymbol{H} = (h_{ij})$ párosság-ellenőrző mátrix segítségével definiáljuk. Legyen $\boldsymbol{x} = (x_1, \ldots, x_n)^T$ egy bitvektor. A $j = 1, \ldots, n$ értékekre, legyen V_j (a „vertikális" szóból) azon i indexek halmaza, melyekre $h_{ij} = 1$. Az $i = 1, \ldots, k$ értékekre, legyen H_i (a „horizontális" szóból) azon j indexek halmaza, melyekre $h_{ij} = 1$. Ekkor a $\boldsymbol{Hx} = 0$ feltételt úgy is kifejezhetjük, hogy minden i-re, $\sum_{j \in H_i} x_j \equiv 0 \pmod 2$. A H_i halmazokat *párosság-ellenőrző halmazoknak* nevezzük. Mostantól kezdve az i indexeket *vizsgálatoknak* fogjuk nevezni, a j indexeket *helyeknek*.

5.9. Definíció. Egy $\boldsymbol{H}$ lineáris kód *alacsony-sűrűségű párosság-ellenőrző kód* a $K, N > 0$ korlátokkal, ha a következő feltételek teljesülnek:

(a) Minden j-re $|V_j| \leqslant K$;

(b) Minden i-re $|H_i| \leqslant N$.

Más szavakkal, minden sor súlya legfeljebb N, és minden oszlop súlya legfeljebb K. ♢

Konstrukcióinkban a K, N korlátokat konstansnak tartjuk, míg a kódszavak n hossza növekszik. Tekintsük a helyzetet, amikor $\boldsymbol{x}$ egy kódszó, melyet néhány hiba

elrontott. Ha az x_j bit helyességét akarjuk ellenőrizni, megvizsgálhatjuk az

$$s_i = \sum_{j \in H_i} x_j$$

összegeket, az összes $i \in V_j$-re. Ha mindezek értéke 0, akkor nem gyanakodnánk arra, hogy x_j hibás. Ha ezen összegeknek csak egyike különbözik 0-tól, akkor tudni fogjuk, hogy hibák vannak $\boldsymbol{x}$-ben, de még mindig gondolhatjuk, hogy a hiba nem az x_j bitben van. De ha az összegeknek jelentős hányada nem 0, akkor gyaníthatjuk, hogy x_j a bűnös, és javasolhatjuk megváltoztatását. Ez a gondolat sugallja a következő definíciót.

5.10. Definíció. A K, N korlátokkal rendelkező $\boldsymbol{H}$ alacsony-sűrűségű párosság-ellenőrző kódhoz egy *frissítő* műveletet rendelünk, melyet az összes j helyen egyidőben kell végrehajtani:

> Állapítsuk meg, hogy az összes s_i összegek között ($i \in V_j$-re) több, mint $\lfloor K/2 \rfloor$ különbözik-e nullától. Ha igen, billentsük át az x_j bitet.

Jelöljük $\boldsymbol{x^H}$-val az $\boldsymbol{x}$-ből e művelettel kapott vektort. Az $0 < \alpha, \gamma < 1$ paraméterekre, nevezzük $\boldsymbol{H}$-t egy $(\alpha, \gamma, K, N, k, n)$-*frissítőnek*, ha minden n hosszúságú $\boldsymbol{x}$ vektorra, melynek súlya $|\boldsymbol{x}| \leqslant \alpha n$, az eredményvektor súlya így csökken: $|\boldsymbol{x^H}| \leqslant \gamma\alpha n$. ♢

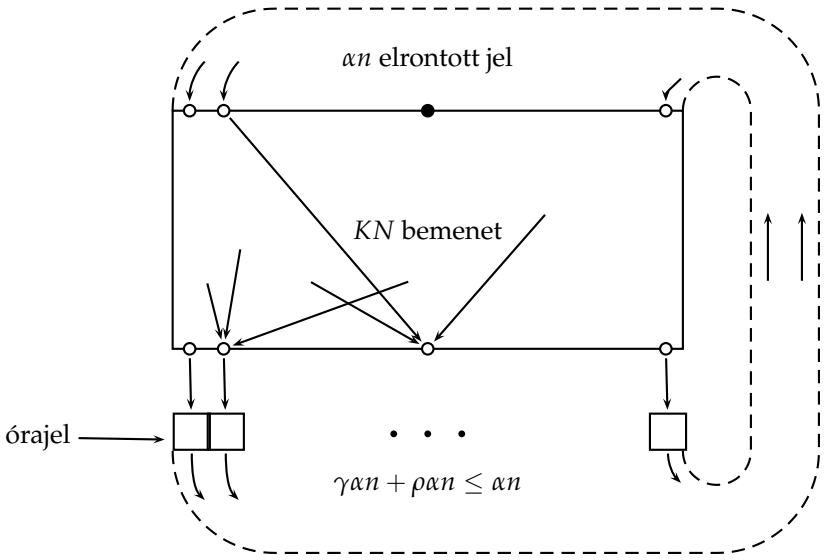
Könnyű észrevenni a sűrítőkhöz való hasonlóságot. A következő lemma a frissítők alkalmazását mutatja, és példát ad a lineáris kódok használatának előnyeire.

5.11. Lemma. *Egy $(\alpha, \gamma, K, N, k, n)$-frissítő $\boldsymbol{H}$ mátrixhoz legyen $\boldsymbol{x}$ egy n-vektor és $\boldsymbol{y}$ egy n hosszúságú kódszó, melyekre $|\boldsymbol{x} - \boldsymbol{y}| \leqslant \alpha n$. Ekkor $|\boldsymbol{x}^H - \boldsymbol{y}| \leqslant \gamma\alpha n$.*

Bizonyítás. Mivel $\boldsymbol{y}$ kódszó, tehát $\boldsymbol{Hy} = 0$, amiből $\boldsymbol{H}(\boldsymbol{x} - \boldsymbol{y}) = \boldsymbol{Hx}$ következik. Tehát a hibajavítás ugyanazokat a biteket billenti át $\boldsymbol{x} - \boldsymbol{y}$-ben, mint $\boldsymbol{x}$-ben: $(\boldsymbol{x} - \boldsymbol{y})^{\boldsymbol{H}} - (\boldsymbol{x} - \boldsymbol{y}) = \boldsymbol{x^H} - \boldsymbol{x}$, azaz $\boldsymbol{x^H} - \boldsymbol{y} = (\boldsymbol{x} - \boldsymbol{y})^{\boldsymbol{H}}$. Tehát ha $|\boldsymbol{x} - \boldsymbol{y}| \leqslant \alpha n$, akkor $|\boldsymbol{x}^H - \boldsymbol{y}| = |(\boldsymbol{x} - \boldsymbol{y})^{\boldsymbol{H}}| \leqslant \gamma\alpha n$. □

9. Tétel. *Minden $K \geqslant 11$ korláthoz léteznek α, γ, N és $R > 0$ paraméterek, melyekkel, minden elég nagy n kódhosszhoz van egy $(\alpha, \gamma, K, N, k, n)$-frissítő, legalább $n - k \geqslant Rn$ információ-bittel.*

Alkalmazzuk ezt a tételt, mielőtt bizonyítjuk:



13. ábra. Frissítő használata

A 8. tétel bizonyítása. A 9. tétel egy információtároló berendezést ad. Valóban, az $\boldsymbol{x} \to \boldsymbol{x^H}$ műveletet megvalósíthatjuk úgy, hogy az $\boldsymbol{x}$ vektor minden j bitjéhez egyetlen g_j kaput használunk, melynek legfeljebb KN bemenete van. Most ha az $|\boldsymbol{x} - \boldsymbol{y}| \leqslant \alpha n$ egyenlőtlenség teljesül valamilyen $\boldsymbol{y}$ kódszóra, az $|\boldsymbol{x^H} - \boldsymbol{y}| \leqslant \gamma\alpha n$ egyenlőtlenség következik. Persze minden kapu tévedhet $\leqslant \varepsilon$ valószínűséggel, és új eltéréseket vezethet be, valami $\boldsymbol{x}'$ vektort adva $\boldsymbol{x^H}$ helyett. Legyen $e\varepsilon < \rho < 1 - \gamma$. Ekkor akárcsak korábban, annak valószínűségét, hogy több, mint $\rho\alpha n$ tévedés történik, az exponenciálisan csökkenő $(e\varepsilon/\rho)^{\rho n}$ kifejezés korlátozza. Kevesebb, mint ρn új eltéréssel még mindig $|\boldsymbol{x}' - \boldsymbol{y}| < (\gamma + \rho)\alpha n < \alpha n$. □

5.12. Definíció. Definiáljuk a tétel $\boldsymbol{H}$ frissítőjét.

Először, megfelelő k', n' pozitív egész számokat választunk. Egy véletlen $k' \times n'$ nagyságú nemnegatív egész $\boldsymbol{H}' = (h'_{ij})$ mátrixot definiálunk a következőképpen. Válasszunk Kn' véletlen $I_{js} \in \{1, \ldots, k'\}$ egész számot $s = 1, \ldots, K$-re, egymástól függetlenül, és legyen

$$h'_{ij} = \bigvee_s [I_{js} = i].$$

Tehát minden i-re K „golyót" dobálunk véletlenül a $(j,1), \ldots, (j,k')$ „urnák" egyikébe, és $h'_{ij} = 1$ mutatja, került-e a (j,i) urnába golyó. Legyen

$$V_j = \{\, i : h'_{ij} > 0 \,\}, \quad H_i = \{\, j : h'_{ij} > 0 \,\}.$$

(Ez nem fog félreértést okozni, noha korábban V_j, H_i a $\boldsymbol{H}$ mátrixhoz volt rendelve hasonló módon.) A $\boldsymbol{H}'$ mátrix $\boldsymbol{H} = (h_{ij})$ részmátrixát a következőképpen definiáljuk. Legyen $\mathcal{R} = \{r_1, \ldots, r_k\}$ a $\boldsymbol{H}'$ azon i sorainak halmaza, melyekre $|H_i| \leqslant N$, és legyen $\mathcal{C} = \{c_1, \ldots, c_n\}$ azon j oszlopok halmaza, melyekre $V_j \subset \mathcal{R}$. Ekkor

$$h_{ij} = h'_{r_i c_j}.$$

Tehát a $\boldsymbol{H}$ mátrixot úgy kapjuk, hogy csak azokat az i sorokat (vizsgálatokat) tartjuk meg $\boldsymbol{H}'$-ból, melyekre $H_i \leqslant N$, és csak azokat a j oszlopokat (helyeket), amelyeket ezek a sorelhagyások nem befolyásolnak. ♢

5.13. Lemma. *Minden elég nagy n'-re, ha $k' = \lceil 0.7n' \rceil$, akkor $\geqslant 0.9$ valószínűséggel a $\boldsymbol{H}$ részmátrix k, n dimenziói eleget tesznek a $k'/2 \leqslant k \leqslant 0.8n$ feltételnek.*

A lemma bizonyítását az 5.2 feladat vázolja. Ennek a lemmának a segítségével fogunk egy $R \geqslant 0.2$ sebességű frissítőt készíteni.

Egy n'-dimenziós $\boldsymbol{z}$ vektorhoz definiáljuk az n-dimenziós $\mathrm{Le}(\boldsymbol{z})$ vektort: $(\mathrm{Le}(\boldsymbol{z}))_i = z_{c_i}$, és egy n-dimenziós $\boldsymbol{x}$ vektorhoz, legyen $\mathrm{Fel}(\boldsymbol{x})$ a következő n'-dimenziós vektor: $(\mathrm{Fel}(\boldsymbol{x}))_{c_j} = x_j$, és $(\mathrm{Fel}(\boldsymbol{x}))_s = 0$, ha $s \notin \mathcal{C}$. Dolgozhatunk a $\boldsymbol{H}'$ mátrixszal, még akkor is, ha végül a $\boldsymbol{H}$ részmátrix hibajavító tulajdonságai érdekelnek minket. A következő, könnyen ellenőrizhető egyenlőség teszi ezt lehetővé:

$$\boldsymbol{x}^{\boldsymbol{H}} = \mathrm{Le}((\mathrm{Fel}(\boldsymbol{x}))^{\boldsymbol{H}'}). \tag{5.7}$$

Egy n'-dimenziós $\boldsymbol{x}$ vektorra legyen

$$E_{\boldsymbol{x}} = \{\, j : x_j = 1 \,\}.$$

Célunk megmutatni, hogy nagy valószínűséggel olyan $\boldsymbol{H}'$ mátrixot kapunk a véletlen választás után, hogy az $|\boldsymbol{x}^{\boldsymbol{H}'}| \leqslant \gamma\alpha n$ összefüggés minden $\boldsymbol{x}$ vektorra igaz lesz, melyre $E_{\boldsymbol{x}} \subset \mathcal{C}$ és $|E_{\boldsymbol{x}}| \leqslant \alpha n$.

5.14. Definíció. Definiáljuk a

$$T = \lfloor K/2 \rfloor$$

küszöböt.

– Egy $j \in \mathcal{C} \setminus E_{\boldsymbol{x}}$ hely *rossz*, ha az $i \in V_j$ vizsgálatok közül több, mint T olyan, hogy $H_i \cap E_{\boldsymbol{x}} \neq \emptyset$. Legyen $\mathrm{Rossz}(\boldsymbol{x})$ a rossz helyek halmaza.

– Egy $j \in E_{\boldsymbol{x}}$ hely *jó*, ha több, mint T olyan $i \in V_j$ vizsgálat van, melyre $H_i \cap E_{\boldsymbol{x}} = \{j\}$. Legyen $\mathrm{Jó}(\boldsymbol{x})$ a jó helyek halmaza.

♢

Szemléletesen, egy $j \in E_{\boldsymbol{x}}$ „hiba" hely akkor jó, ha $\boldsymbol{H}'$ biztosan kijavítja, és egy $j \notin E_{\boldsymbol{x}}$ „nem-hiba" hely akkor rossz, ha $\boldsymbol{H}'$ esetleg „kijavítja". Könnyű látni, hogy ha a j hely jó, akkor $x_j^{\boldsymbol{H}'} = 0$, és ha egy $j \notin E_{\boldsymbol{x}}$ hely nem rossz, akkor $x_j^{\boldsymbol{H}'} = 0$.

A következőkben feltesszük, hogy $|\boldsymbol{x}| \leqslant \alpha n$ egy megfelelően kicsi α konstansra. Először felülről becsüljük a rossz helyek számát, megmutatva, hogy $|\mathrm{Rossz}(\boldsymbol{x})| \leqslant c_1 \alpha n$ egy alkalmasan kicsi c_1 konstansra. Ezután alulról becsüljük a jó helyek számát, megmutatva, hogy $|\boldsymbol{x}| < c_2 \alpha n$ vagy $|\mathrm{Jó}(\boldsymbol{x})| \geqslant (1 - c_2)\alpha n$ egy alkalmasan kicsi c_2 konstansra, ahol $c_1 + c_2 < 1$. Ebből a két eredményből az következik, hogy $|\boldsymbol{x}|^{\boldsymbol{H}'} \leqslant (c_1 + c_2)\alpha n$, ezért $\gamma = c_1 + c_2$ választható a frissítőhöz.

5.15. Lemma. *Legyen*

$$c_1 > 1/T. \tag{5.8}$$

Ekkor létezik egy $\alpha > 0$ konstans, melyre $\geqslant 0.9$ valószínűséggel a $\boldsymbol{H}'$ mátrix választásában, minden $\boldsymbol{x} \in \mathbb{F}_2^{n'}$-re, ha $|\boldsymbol{x}| \leqslant \alpha n$, akkor $|\mathrm{Rossz}(\boldsymbol{x})| \leqslant c_1 \alpha n$.

Bizonyítás. Rögzítsünk egy j helyet és egy $\boldsymbol{x}$ értéket. Minden $s = 1, \ldots, K$ értékre, annak valószínűsége, hogy H_{I_s} metszi az $E_{\boldsymbol{x}}$ halmazt, nem nagyobb, mint annak valószínűsége, hogy a véletlen I_s szám benne van a $\bigcup_{p \in E_{\boldsymbol{x}}} V_p$ halmazban, azaz legfeljebb

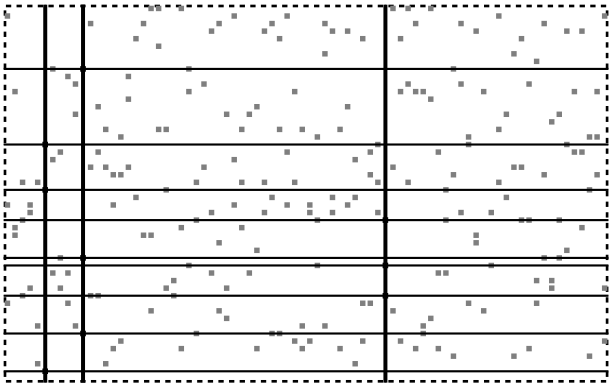
$$K\alpha n/k' \leqslant K\alpha n'/k' = K\alpha n'/\lceil 0.7n' \rceil \leqslant K\alpha/0.8,$$

ha n' nagy. Annak valószínűsége, hogy j rossz legfeljebb akkora, mint annak valószínűsége, hogy több, mint T ilyen s érték van:

$$p_{\boldsymbol{x}} := \mathbf{P}[\, j \in \mathrm{Rossz}(\boldsymbol{x})\,] \leqslant \binom{K+1}{T+1} (K\alpha/0.8)^{T+1}.$$

Jelöljük a jobboldalt $K_1 \alpha^{T+1}$-el. Minden a $\mathcal{C} \setminus E_{\boldsymbol{x}}$ halmazba eső j helyre, azok az események, hogy j rossz, függetlenek. Legyen $\beta = |\boldsymbol{x}|/n$, akkor $\mathcal{C} \setminus E_{\boldsymbol{x}} = (1 - \beta)n$. Becsüljük meg annak a valószínűségét, hogy legalább $c_1 \alpha n$ hely rossz a $\mathcal{C} \setminus E_{\boldsymbol{x}}$ halmazban. Vezessük be a $c_1 \alpha n = f(1 - \beta)n$ jelölést (így $f = c_1 \alpha/(1 - \beta)$). Az (1.6) egyenlőtlenség ekkor ezt adja:

$$\begin{aligned}
\mathbf{P}[\, |\mathrm{Rossz}(\boldsymbol{x})| \geqslant c_1 \alpha n\,] = \mathbf{P}[\, |\mathrm{Rossz}(\boldsymbol{x})| \geqslant f(1-\beta)n\,] &\leqslant \left(\frac{e p_{\boldsymbol{x}}}{f}\right)^{f(1-\beta)n} \\
&= \left(\frac{e p_{\boldsymbol{x}} (1-\beta)}{c_1 \alpha}\right)^{c_1 \alpha n} = \left(\frac{e K_1 \alpha^T (1-\beta)}{c_1}\right)^{c_1 \alpha n} \\
&\leqslant \left((e K_1/c_1)^{c_1} \alpha^{c_1 T}\right)^{\alpha n}.
\end{aligned}$$



14. ábra. A pöttyök 1-esek a $\boldsymbol{H}$ mátrixban, a függőleges vonalak mutatják a hibákat, a vízszintes vonalak pedig az eleven vizsgálatokat.

Legfeljebb $\sum_{i\leqslant\alpha n}\binom{n}{i} \leqslant (e/\alpha)^{\alpha n}$ lehetséges választás van $\boldsymbol{x}$-re, ha $|\boldsymbol{x}| \leqslant \alpha n$ (itt az (1.8) egyenlőtlenséget használtuk). Ezért

$$\mathbf{P}[\,\exists \boldsymbol{x}(|\boldsymbol{x}| \leqslant \alpha n \wedge |\mathrm{Rossz}(\boldsymbol{x})| \geqslant c_1\alpha n)\,] \leqslant U(\alpha)^{\alpha n},$$

ahol

$$U(\alpha) = (e/\alpha)(eK_1/c_1)^{c_1}\alpha^{c_1 T} = e(eK_1/c_1)^{c_1}\alpha^{c_1 T-1}.$$

Ha α elég kicsi, akkor az (5.8) feltételből $U(\alpha) < 1$ következik. □

A jó helyek számának alsó korlátját előkészítve, nevezzük a

$$W_{\boldsymbol{x}} = \bigcup_{j\in E_{\boldsymbol{x}}} V_j$$

halmaz elemeit *eleven vizsgálatoknak*: ezek a vizsgálatok találkoznak valóban hibával. A következő lemma azt mutatja, hogy ha az eleven vizsgálatok száma a maximumhoz közel van, akkor sok a jó hely.

5.16. Lemma. *Legyen* $0 < d$. *Ha* $|W_{\boldsymbol{x}}| > K|\boldsymbol{x}|(1-d)$, *akkor* $|\mathrm{Jó}(\boldsymbol{x})| > (1-4d)|\boldsymbol{x}|$.

Általánosabban, $T \leqslant K/2$*-re, legyen* $U_1,\dots,U_p$ *egy halmazcsalád, ahol* $|U_j| \leqslant K$, *és legyen* $U_j' = U_j \setminus \bigcup_{s\neq j} U_s$. *Ha* $|\bigcup_j U_j| > (1-d)Kp$, *akkor* $|\{\, j : |U_j'| > T \,\}| > (1-4d)p$.

Bizonyítás. A speciális állítást az $E_{\boldsymbol{x}} = \{s_1, \dots, s_p\}$, $U_j = V_{s_j}$ helyettesítéssel kapjuk az általánosabból. Legyen

$$U = \bigcup_j U_j, \quad U' = \bigcup_j U'_j, \quad J = \{\, j : |U'_j| > T \,\}.$$

Tegyük fel, hogy $|\{\, j : |U'_j| > T \,\}| \leqslant (1-4d)p$, megmutatjuk, hogy akkor $|U| \leqslant Kp(1-d)$, az eredeti feltevéssel ellentétben. Az $|U| \leqslant |U'| + (Kp - |U'|)/2$ egyenlőtlenség abból következik, hogy az U unióhalmaz $U \setminus U'$ részében minden elem legalább kétszer van fedve. Továbbá, $c = 1 - 4d$ jelöléssel:

$$\begin{aligned}
|U'| &\leqslant |J|K + (p - |J|)T = pT + |J|(K - T) \leqslant p(T + c(K - T)),\\
|U| &\leqslant |U'| + (Kp - |U'|)/2 = Kp/2 + |U'|/2\\
&\leqslant Kp/2 + (pT + c(K - T))/2 = (p/2)(K(1 + c) + T(1 - c))\\
&\leqslant (p/2)(K(1 + c) + (K/2)(1 - c)) = Kp(3 + c)/4 = Kp(1 - d).
\end{aligned}$$

□

A következő lemma az 5.16. lemmával együtt már maga után vonja, hogy sok jó hely van.

5.17. Lemma. *Létezik olyan $\alpha > 0$ konstans, hogy minden elég nagy n-re, $\geqslant 0.9$ valószínűséggel a $\boldsymbol{H}'$ véletlen mátrix választásánál, minden olyan $\boldsymbol{x}$ vektorra, ha $4d\alpha n \leqslant |\boldsymbol{x}| \leqslant \alpha n$, akkor a*

$$2.8d^2K > 1 \tag{5.9}$$

egyenlőtlenségből $|W_{\boldsymbol{x}}| > (1-d)K|\boldsymbol{x}|$ következik.

Bizonyítás. Rögzítsünk egy $\boldsymbol{x}$ vektort, és bármilyen $\mathcal{W} \subset \{1, \dots, k'\}$ halmazt, melyre $|\mathcal{W}| \leqslant (1-d)K|\boldsymbol{x}|$. A $W_{\boldsymbol{x}} \subset \mathcal{W}$ esemény akkor következik be, ha $I_{js} \in \mathcal{W}$ minden $j \in E_{\boldsymbol{x}}$-re, és $s = 1, \dots, K$-ra. Ennek valószínűsége legfeljebb $((1-d)K|\boldsymbol{x}|/k')^{K|\boldsymbol{x}|}$, ezért

$$\mathbf{P}[\,|W_{\boldsymbol{x}}| < (1-d)K|\boldsymbol{x}|\,] \leqslant \binom{k'}{(1-d)K|\boldsymbol{x}|} ((1-d)K|\boldsymbol{x}|/k')^{K|\boldsymbol{x}|}.$$

Jelöljük a jobboldalt q_1-el. A $\beta = |\boldsymbol{x}|/k'$ jelöléssel, az (1.7) egyenlőtlenségből:

$$\binom{k'}{(1-d)K|\boldsymbol{x}|} \leqslant ((1-d)K\beta/e)^{-(1-d)K\beta k'}.$$

Feltettük, hogy $4d\alpha \leqslant \beta \leqslant \alpha$ és $k' = \lceil 0.7n' \rceil \geqslant 0.7n$, ezért

$$\begin{aligned} q_1 &\leqslant e^{(1-d)K\beta k'}((1-d)K\beta)^{dK\beta k'} \leqslant e^{K\beta k'}(K\alpha)^{dK\beta k'} \\ &= \left(e^{1/d}K\alpha\right)^{dK\beta k'} \leqslant \left(e^{1/d}K\alpha\right)^{4d^2K\alpha k'} \leqslant \left(e^{1/d}K\alpha\right)^{2.8d^2K\alpha n}, \end{aligned}$$

ahol feltettük még, jogosan, hogy α elég kicsi az $e^{1/d}K\alpha \leqslant 1$ teljesüléséhez is. Mint az 5.15. lemma bizonyításában, legfeljebb $(e/\alpha)^{\alpha n}$ lehetőség van $\boldsymbol{x}$ számára, ezért

$$\begin{aligned} &\mathbf{P}\left[\exists \boldsymbol{x}\big(4d\alpha n \leqslant |\boldsymbol{x}| \leqslant \alpha n \wedge |W_{\boldsymbol{x}}| \leqslant (1-d)K|\boldsymbol{x}|\big)\right] \\ &\qquad \leqslant (e/\alpha)^{\alpha n}\left(e^{1/d}K\alpha\right)^{2.8d^2K\alpha n} = \left(e^{0.7cK+1}K^{2.8d^2K}\alpha^{2.8d^2K-1}\right)^{\alpha n}. \end{aligned}$$

Mivel feltettük az (5.9)-t, a zárójeles kifejezés kisebb lesz, mint 1, ha α elég kicsi. □

A 9. Tétel bizonyítása. Megfelelő c_1, d számokat fogunk választani az (5.8) és (5.9) feltételek teljesüléséhez, továbbá legyen $c_2 = 4d$, $\gamma = c_1 + c_2$. Az 5.15. lemma szerint $\mathbf{P}[\,|\text{Rossz}(\boldsymbol{x})| \leqslant c_1\alpha n\,] > 0.9$ a $\boldsymbol{H}'$ mátrix véletlen választásakor. Tegyük fel először, hogy $|\boldsymbol{x}| \leqslant c_2\alpha n$, akkor

$$|\boldsymbol{x}^{\boldsymbol{H}'}| \leqslant (c_1 + c_2)\alpha n. \tag{5.10}$$

Most pedig tegyük fel, hogy $c_2\alpha n \leqslant |\boldsymbol{x}| \leqslant \alpha n$. Ekkor az 5.17. lemma szerint $\mathbf{P}[\,|W_{\boldsymbol{x}}| \geqslant (1-d)K|\boldsymbol{x}|\,] > 0.9$ a $\boldsymbol{H}'$ mátrix véletlen választásakor. Az 5.16. lemmából $\mathbf{P}[\,|\text{Jó}(\boldsymbol{x})| \geqslant (1-c_2)|\boldsymbol{x}|\,] > 0.9$ következik. Tehát a frissítő művelet az $\boldsymbol{x}$ vektornak legfeljebb $c_2|\boldsymbol{x}| \leqslant c_2\alpha n$ bitjét hagyja javítatlanul, azaz (5.10) teljesül, legalább 0.8 valószínűséggel.

Az van még hátra, hogy a c_1 és d paramétereket a (5.8) és (5.9) feltételek teljesítésével válasszuk. Például, $K = 21$-el, a $c_1 = 0.15$, $d = 0.14$ választás $\gamma = c_1 + c_2 = 0.71$-et ad. Könnyű látni, hogy még akkor is, ha $K = 11$, lehet a c_1, d párt úgy választani, hogy $c_1 + c_2 < 1$ legyen. □

A 4. fejezet végén felhozott összes kifogás méginkább érvényes a megbízható információtárolás itt bemutatott eredményeire. Az α, ε-ra kapott korlátok nagyon kicsik, és ennek megfelelően az az n tárnagyság, melyre ez az elrendezés először értelmet kap, nagyon nagy. (Lásd az 5.4 gyakorlatot.)

5.5. Gyakorlatok

5.1. Gyakorlat. Bizonyítsuk az 5.7. állítást.

5.2. Gyakorlat. A $\boldsymbol{H}$ mátrix konstrukciójában, a $k' = 0.6n'$ választással, legyen $\mathcal{C}_0$ a kihagyott oszlopok halmaza. Bizonyítsuk be, hogy minden $i \in \{1, \dots, k'\}$, $j \in \{1, \dots, n'\}$-re

$$\mathbf{P}[\, j \in H_i,\ |H_i| > N \,] \leqslant (K/k') \binom{n'-1}{N} (K/k')^N \leqslant (K/k') \frac{(K/(1-r'))^N}{N!},$$

$$\mathbf{E}|\mathcal{C}_0|/n' \leqslant K \frac{(K/(1-r'))^N}{N!} \to 0,$$

ha $N \to \infty$. Mutassuk meg ebből, hogy $\mathbf{P}[\, k'/2 \leqslant k \leqslant 0.8n \,] \to 1$, ha $N \to \infty$.

5.3. Gyakorlat. Bizonyítsuk az (5.7) egyenlőséget.

5.4. Gyakorlat. A $K = 21$ esetre, számoljunk ki felső korlátokat N, α, ε-ra és alsó korlátot n-re, melyek biztosítják, hogy létezik $(\alpha, \gamma, K, N, k, n)$-frissítő a $k \leqslant 0.8n$ tulajdonsággal. Ekkor létezik tehát információ-tároló hálózat n kódszó-hosszúsággal, és $R \geqslant 0.2$ sebességgel.

6. Feladatok

1. Feladat. (Kritikus érték) Vegyük a 2.5 gyakorlat $\mathcal{M}_k$ hálózatát, feltételezve, hogy minden kapu $\leqslant \varepsilon$ valószínűséggel egymástól függetlenül téved. Tegyük fel, hogy a bemenetvektor csupa 0, és legyen $p_k(\varepsilon)$ annak valószínűsége, hogy a kimenet 1. Mutassuk meg, hogy létezik egy $\varepsilon_0 < 1/2$ érték azzal a tulajdonsággal, hogy minden $\varepsilon < \varepsilon_0$ esetén $\lim_{k\to\infty} p_k(\varepsilon) = 0$, és $\varepsilon_0 < \varepsilon \leqslant 1/2$ esetén $\lim_{k\to\infty} p_k(\varepsilon) = 1/2$. Becsüljük meg a konvergencia sebességét is mindkét esetben.

2. Feladat. (Reguláris sűrítő) Egy sűrítőt úgy definiáltunk, mint egy d-félreguláris páros multigráfot. Nevezzünk egy sűrítőt *regulárisnak*, ha d-reguláris multigráf (a bemenő csúcsok foka is d). Bizonyítsuk a 6. tétel megfelelőjét: minden $\gamma < 1$-re létezik egy egész $d > 1$ és egy $\alpha > 0$ melyekkel minden pozitív egész k-ra létezik reguláris (d, α, γ, k)-sűrítő. [Útmutatás: Válasszunk egy véletlen d-reguláris páros multigráfot a következő eljárással: (1. Helyettesítsünk minden csúcsot egy d csúcsból álló csoporttal. 2. Válasszunk egy teljes párosítást az új bemenő és kimenő csúcsok között. 3. Olvasszuk újra egy csúcsba össze mindegyik d csúcsból álló csoportot.) Bizonyítsuk, hogy e választás után kicsi annak a valószínűsége, hogy a kapott d-reguláris multigráf nem sűrítő. Ehhez, fejezzük ki ezt a valószínűséget faktoriálisokkal, és becsüljük azokat a Stirling formulával.]

3. Feladat. (Kétirányú tágító) Emlékezzünk a tágítók definíciójára a 4.3 gyakorlatban. Nevezzünk egy (d, α, λ, k)-tágítót *regulárisnak*, ha d-reguláris multigráf (a bemeneti csúcsok foka d). Ezt a multigráfot *kétirányú tágítónak* nevezzük, ha mindkét irányba tágító: A-ból B-be, és B-ből A-ba. Bizonyítsunk egy tételt, mely a 2 feladathoz hasonló: minden $\lambda < d$-re létezik egy $\alpha > 0$, mellyel minden pozitív egész k-ra létezik kétirányú (d, α, λ, k)-tágító.

4. Feladat. (Sűrítő csak hármas szavazásból) A 6. tétel bizonyítása nem garantál (d, α, γ, k)-sűrítőt, ha $\gamma < 1/2$, $d < 7$. Ha csak 3-bemenetű többségi kapukat akarunk használni, akkor a következő konstrukció kínálkozik. Először egy 3-félreguláris páros G multigráfot készítünk $u_1, \ldots, u_k$ bemenetekkel és $v_1, \ldots, v_{3k}$, kimenetekkel, egy 3-bemenetű többségi kapuval minden v_i-ben. Azután a $w_1, \ldots, w_k$ új csúcsokat vezetjük be, minden w_j-ben egy 3-bemenetű többségi kapuval. A w_1 kapu a v_1, v_2, v_3 csúcsok többségét számolja ki, a w_2 kapu a v_4, v_5, v_6 többségét, és így tovább. Számítsuk ki, hogy a G gráf véletlen választása után a hálózat az $(u_1, \ldots, u_k)$ bemenetekkel és $(w_1, \ldots, w_k)$ kimenetekkel frissítő szervként tud-e működni. Ezután próbálkozzunk három lépéssel a kettő helyett (amikor G-nek $9k$ kimenete van), és állapítsuk meg, mit nyerünk.

5. Feladat. (Felújító szerv NOR kapukból) A többségi kapu nem az egyetlen kapu, mely a többséget erősíteni tudja. Emlékezzünk a 2.2 gyakorlatban bevezetett NOR kapura, és képezzük a következőt: $\mathrm{NOR}_2(x_1, x_2, x_3, x_4) = (x_1 \,\mathrm{NOR}\, x_2)\,\mathrm{NOR}\,(x_3 \,\mathrm{NOR}\, x_4)$. Mutassuk meg, hogy egy, a 4 feladathoz hasonló konstrukcióban a 3-bemenetű többségi kapu helyett NOR_2 is használható.

6. Feladat. (Több véletlenség, kisebb sűrítők) A 4.4 gyakorlat jelölésével tegyük fel, hogy mint ott, az F_v valószínűségi változók eloszlása nem függ az egész hálózat bemenő vektorától. Járjunk el a 4.6 gyakorlathoz hasonlóan minden felújító szerv építésében. Az eredeti hálózat minden kapujához egy új felújító szervet válasszunk: a választás függ az elkészült hálózatnak ezt a kaput megelőző részétől. Mutassuk meg, hogy ebben az esetben hibabecsléseink lényegesen megjavulnak. A javulás abból jön, hogy, mint a 4.6 gyakorlatban, most nem kell a hibavalószínűséget megszorozni a fertőzött drótok összes lehetséges $\leqslant \alpha k$ nagyságú halmazainak számával. Mivel ezen véletlen halmaz eloszlása ismert, átlagolhatunk felette.

7. Feladat. (Közel-sorbarendezés tágítókkal) Ebben a feladatban megmutatjuk, hogy tágítókat „közel-sorbarendezésre" lehet alkalmazni. Legyen G egy reguláris kétirányú (d, α, λ, k)-tágító, melynek két k-nagyságú része A és B. König tétele szerint minden d-reguláris páros multigráf (élhalmaza) d teljes párosítás (élhalmazainak) diszjunkt uniója: legyenek ezek $M_1, \ldots, M_d$. Egy ilyen tágítóhoz egy d mélységű logikai hálózatot rendelünk a következőképpen. A csúcsokat az $i = 0, 1, \ldots, d$ szintekbe csoportosítjuk. Az i szinten két diszjunkt k nagyságú csúcshalmaz, A_i, B_i helyezkedik el, az a_{ij}, b_{ij} $(j = 1, \ldots, k)$ csúcsokkal. Az a_{ij}, b_{ij} csúcsokban található érték x_{ij} illetve y_{ij} lesz. Jelöljük az i szint $2k$-elemü vektorát így: $z_i = (x_{i1}, \ldots, y_{ik})$. Ha (p, q) az M_i párosítás egy éle, akkor egy $\wedge$ kaput teszünk a_{ip}-be és egy $\vee$ kaput $b_{iq} - ba$:

$$x_{ip} = x_{(i-1)p} \wedge y_{(i-1)q}, \quad y_{iq} = x_{(i-1)p} \vee y_{(i-1)q}.$$

Ez a hálózat a 0-kat A_i-be és az 1-eseket B_i-be igyekszik küldeni d lépésben. Általánosabban, a z_i vektorokban található értékek tetszőleges számok is lehetnek. Ekkor az $x \wedge y$ jelentése továbbra is $\min(x, y)$, és az $x \vee y$ jelentése $\max(x, y)$, és ezzel minden z_i vektor a z_0 vektor permutációja. Legyen $\beta = (1 + \lambda)\alpha$. Bizonyítsuk be, hogy z_d a következő értelemben β-*rendezett*: minden m-re, a z_d-nek m legkisebb számából legalább βm van a bal félben, és legnagyobb számaiból legalább βm van a jobb félben.

8. Feladat. (Felújító szerv közel-sorbarendezőkből) Építsünk felújító szervet tágítók felhasználásával, a következőképpen. Először is, az A bemeneti kábel minden vezetékét ágaztassuk ketté, hogy az A'_0, B'_0 halmazokat kapjuk. Most illesszük

ezekhez a 7 feladat β-sorbarendezőjét: a kimenet az A'_d, B'_d halmazokra oszlik. Most ágaztassuk a B'_d halmaz vezetékeit az A''_0, B''_0 halmazokba. Illesszünk oda megint egy β-sorbarendezőt, ez az A''_d, B''_d kimenetekhez vezet. Tartsuk meg csak a $B = A''_d$ halmazt a kimenő kábelnek. Bizonyítsuk be, hogy az A-ból B-be vezető logikai vektor-hálózat felújító szerv.

9. Feladat. (Hierarchikus hibajavítás) Hierarchikus hálózatot építünk, amelyet egy bit információ megbízható tárolására is, és felújító szervnek is lehet használni. Egy d páratlan számra és egy $r \geqslant 1$ egész számra, egy d^r-bemenetű, d^r-kimenetű $(r+1)d^r$ nagyságú R^r_d hálózatot építünk rekurzívan. Ha $r = 1$, akkor az $x(1), \ldots, x(d)$ bemenetekből a $z(1), \ldots, z(d)$ kimenetek mindegyikét ugyanúgy számoljuk ki: $z(j) = \mathrm{Maj}(x(1), \ldots, x(d))$. Tegyük fel, hogy R^r_d már megépült, építsük meg R^{r+1}_d-t, az $x(1), \ldots, x(d^{r+1})$ bemenetekkel és $z(1), \ldots, z(d^{r+1})$ kimenetekkel a következőképpen. Legyenek $C_1, \ldots, C_d$ az R^r_d hálózat másolatpéldányai. Az $i = 1, \ldots, d$ esetekben vegyük a C_i hálózatot, az $x((i-1)d^r + j)$ bemenetekkel $(j = 1, \ldots, d^r)$ és y_{ij} kimenetekkel $(j = 1, \ldots, d^r)$. Ezután legyenek $D_1, \ldots, D_{d^r}$ az R^1_d példányai. A $j = 1, \ldots, d^r$, esetekben vegyük a D_j hálózatot az y_{ij} bemenetekkel $(i = 1, \ldots, d)$, és a $z((j-1)d^r + i)$ kimenetekkel $(i = 1, \ldots, d)$. Ezzel R^{r+1}_d elkészült.

Egy h küszöbértékre, definiáljuk egy $\boldsymbol{x} = (x_1, \ldots, x_{d^r})$ Boole-vektor $S_h(\boldsymbol{x})$-szel jelölt *h-erejét*, rekurzíve r-ben. Ha $r = 0$, akkor $S_h(x) = x$. Ha $r > 0$ akkor $i = 1, \ldots, d$-re legyen $\boldsymbol{x}_i$ az $\boldsymbol{x}$ i-edik d^{r-1} hosszúságú szakasza. Legyen $S_h(\boldsymbol{x}) = 1$, ha az $S_h(\boldsymbol{x}_i) = 1$ eset az i-nek több, mint h értékére áll fenn; különben $S_h(\boldsymbol{x}) = 0$.

Mutassuk meg, hogy léteznek $0 < \varepsilon, \rho < 1$ a következő tulajdonsággal. Legyen $d = 11$, $h = 3$. Tegyük fel, hogy az R^r_d hálózat zajos, de ε-megengedett, és a véletlen $\boldsymbol{Z}$ kimenetvektort adja. Ekkor amennyiben az $\boldsymbol{x}$ bemenetvektorra $S_h(\boldsymbol{x}) = 0$ áll, ebből $\mathbf{P}[\, S_h(\boldsymbol{Z}) = 1\,] < \rho^{2^r}$ következik. Tehát az R^r_d hálózat egy bit információ hosszantartó tárolására használható. Mutassuk meg, hogy a d és h paramétereknek más értéket választva ezt az eredményt tovább erősíthetjük, és felújító szervet kapunk. (Az erősítés azért kell, mert a végrehajtó szerv két felújító szerv kimenetét olvasztja össze.)

10. Feladat. (A hierarchia ára) Mutassuk meg, hogy a 9 feladatban kapott felújító szerv olyan megbízható hálózatot ad, melyben a redundancia $(\log N)^{\log 3} \log\log N$: rosszabb, mint a korábban kapott $\log N$.

7. Történeti megjegyzések

A nagy eltérések tételét (1. tétel), vagy hasonló tételeket, sokszor Chernoffnak vagy Bernsteinnek tulajdonítják. Az 1.2 gyakorlat az egyik gyakran használt változatot mutatja be.

A megbízhatatlan elemekkel végrehajtható megbízható számolások problémáját a [22] cikkben vizsgálta Neumann János a logikai hálózatok modelljén. Az ottani eredmény teljes bizonyítása először R. L. Dobrusin és Sz. I. Ortyukov [4] cikkében jelent meg (melyben a felújító szerv nem ugyanaz, amit Neumann János javasolt). Fejezetünk előadása N. Pippenger [13] cikkének egyes részeire épül.

Dobrusin és Ortyukov alsó korlátja a [3] cikkben (melynek hibáit a [15], [16] és [6] cikk javítja) azt mutatja, hogy a $\log n$ redundancia általában elkerülhetetlen egy olyan megbízható számolásban, melynek bonyolultsága n. De ez az alsó korlát csak annak szükségességét mutatja, hogy a bemenetet redundánsan kódolt formába kell tenni (különben az első lépésben kritikus információ veszhet el). Amint [13] mutatja, több fontos függvényosztály esetén lineáris redundancia is elegendő.

Természetesnek látszik a kezdeti kódolás költségét különválasztani: akkor talán a számolás többi része sokkal kevesebb redundanciával is végrehajtható. D. Spielman [19] cikke ebben az irányban fontos lépést tett, (lényegében) az ütemezett logikai hálózatok modelljében. Spielman egy w elemi alkotórészből álló hálózaton t ideig futó számítást vesz, és megbízhatóvá teszi, csak $(\log w)^c$-szer több processzort használva, és $(\log w)^c$-szer tovább futtatva. A hibavalószínűség $t\exp(-w^{1/4})$. Ez kicsi mindaddig, amíg t nem sokkal nagyobb, mint $\exp(w^{1/4})$. Tehát a redundanciát a *térszükséglet* logaritmusának egy hatványával lehet korlátozni; az időszükséglet nem is jelenik meg nyíltan. Egy (aciklikus) logikai hálózatban a tér- és idő-bonyolultság nincs külön definiálva: a hálózat elemszáma azzal a mennyiséggel analóg, amit más modellekben az idő- és a tér-bonyolultság összeszorzásával kapunk.

Az információtárolási eredményekhez A. V. Kuznyecov [10] cikkét használtuk (a cikk főtétele, a frissítők létezéséről, M. Pinszker eredménye). Az alacsony-sűrűségű párosság-ellenőrző kódokat R. G. Gallager vezette be a [8] könyvben, és információ-tárolási használatukat először M. G. Taylor javasolta [20] cikkében. Ilyen kódoknak új, konstruktív változatait fejlesztették ki M. Sipser és D. Spielman a [18] cikkben, szupergyors kódolással és dekódolással.

A logikai hálózatoknál sokkal szabályosabb párhuzamos számolási modellben, a sejtautomatákban is lehetséges megbízható számolás. Ezeket az eredményeket itt nem volt alkalmunk bemutatni: lásd például a [7] és [5] cikkeket.

A 2.4 gyakorlat C. Shannontól származik: lásd [17]. A legrosszabb

függvényekre az aszimptotikusan legjobb hálózatnagyságot O. B. Lupanov találta meg a [11] cikkben. A 3.1 gyakorlat a [4] cikken alapul, a 3.2 gyakorlat pedig az [3] cikken (és javításain).

A 4.3 gyakorlatban bevezetett tágítókat az elméleti számítógéptudományban kiterjedten használják: bevezetésnek, lásd a [12] könyvet. A könyv kis sajátérték-arányú gráfok konstrukciójára is ad utalásokat. A 4.5 gyakorlat és a 6 feladat a [4] cikken alapul.

Az 1 feladatnál bonyolultabb kérdéseket tárgyal a [14] cikk. A 2 feladat módszerét véletlen d-reguláris multigráfok generálására például a [2] cikk elemzi. Sokkal nehezebb egyszerű reguláris gráfokat (nem multigráfokat) generálni egyenletes eloszlással: lást például a [9] cikket.

A 7 feladat a $\log n$ mélységű sorbarendező hálózatok indító ötletén alapul (lásd [1]). A 9 feladathoz a [21] cikk gondolatai vezettek.

Irodalom

[1] Miklós Ajtai, János Komlós, és Endre Szemerédi, *Sorting in c* log *n parallel steps*, Combinatorica **3** (1983), 1–19.

[2] E. Bender és R. Canfield, *The asymptotic number of labeled graphs with given degree sequences*, Combinatorial Theory Series A. **24** (1978), 296–307.

[3] R. L. Dobrushin és S. I. Ortyukov, *Lower bound for the redundancy of self-correcting arrangements of unreliable functional elements*, Problems of Information Transmission **13** (1977), 59–65.

[4] _____, *Upper bound on the redundancy of self-correcting arrangements of unreliable elements*, Problems of Information Transmission **13** (1977), no. 3, 201–208.

[5] Peter Gács, *Reliable cellular automata with self-organization*, Journal of Statistical Physics **103** (2001), no. 1/2, 45–267, See also www.arXiv.org/abs/math.PR/0003117 and the proceedings of the 1997 Symposium on the Theory of Computing.

[6] Peter Gács és Anna Gál, *Lower bounds for the complexity of reliable boolean circuits with noisy gates*, IEEE Transactions on Information Theory **40** (1994), 579–583.

[7] _____ és John Reif, *A simple three-dimensional real-time reliable cellular array*, Journal of Computer and System Sciences **36** (1988), no. 2, 125–147.

[8] R. G. Gallager, *Low-density parity-check codes*, MIT Press, Cambridge, Mass., 1963.

[9] J. H. Kim és V. H. Vu, *Generating random regular graphs*, Proc. STOC'03, 2003, pp. 213–222.

[10] A. V. Kuznetsov, *Information storage in a memory assembled from unreliable components*, Problems of Information Transmission **9** (1973), no. 3, 254–264, Translated from Russian.

[11] O. B. Lupanov, *On a method of circuit synthesis*, Izvestia VUZ (Radiofizika) **1** (1958), 120–140, In Russian.

[12] R. Motwani and P. Raghavan, *Randomized algorithms*, Cambridge University Press, Cambridge, UK, 1995.

[13] Nicholas Pippenger, *On networks of noisy gates*, Proc. of the 26-th IEEE FOCS Symposium, 1985, pp. 30–38.

[14] Nicholas Pippenger, *Analysis of error correction by majority voting*, Randomness in Computation (Silvio Micali, ed.), Advances in Computing Research (a scientific annual), vol. 5, JAI Press, Greenwich, Conn., 1989, pp. 171–198.

[15] _____, G. D. Stamoulis és J. N. Tsitsiklis, *On a lower bound for the redundancy of reliable networks with noisy gates*, IEEE Trans. Inform. Theory **37** (1991), no. 3, 639–643.

[16] Rüdiger Reischuk és B. Schmeltz, *Reliable computation with noisy circuits and decision trees—a general n* log *n lower bound*, Proc. of the 32-nd IEEE FOCS Symposium, 1991, pp. 602–611.

[17] Claude Shannon, *The synthesis of two-terminal switching circuits*, Bell System Technical Journal **28** (1949), 59–98.

[18] Daniel A. Spielman, *Linear-time encodable and decodable error-correcting codes*, Proc. of the 27th ACM STOC Symposium, 1995, pp. 388–397.

[19] _____, *Highly fault-tolerant parallel computation*, Proc. of the 37th IEEE FOCS Symposium, 1996, pp. 154–163.

[20] M.G. Taylor, *Reliable information storage in memories designed from unreliable components*, Bell System Tech. J. **47** (1968), no. 10, 2299–2337.

[21] Boris S. Tsirel'son, *Reliable information storage in a system of locally interacting unreliable elements*, Interacting Markov Processes in Biology (V. I. Kryukov R. L. Dobrushin and A. L. Toom, eds.), Scientific Centre of Biological Research, Pushchino, 1977, In Russian. Translation by Springer., pp. 24–38.

[22] John von Neumann, *Probabilistic logics and the synthesis of reliable organisms from unreliable components*, Automata Studies (C. Shannon and McCarthy, eds.), Princeton University Press, Princeton, NJ., 1956.