# VÉLETLENÍTETT ALGORITMUSOK

## 5.ea.

1. Valószínűségszámítási módszer

### 1.1. Áttekintés

Két visszatérő ötlet van a valószínűségszámítási módszerben. Egyik az, hogy bármely véletlen változó felvesz legalább egy olyan értéket, ami nem kisebb, mint a várható érték, és legalább egy olyan értéket, ami nem nagyobb tőle. A másik ötlet az, hogy ha egy halmazból egy véletlenszerűen kiválasztott elem pozitív valószínűséggel kielégít egy tulajdonságot, kell lennie a halmazban egy olyan elemnek, ami kielégíti ezt a tulajdonságot. Például ha egy dobozból 1/3 valószínűséggel választunk piros golyót, akkor biztosan van benne legalább egy piros. Bár ezek egyszerű megállapítások, mégis meglepő erővel bírnak, ezekre látunk példákat a továbbiakban.

### 1.2. Játékfa kiértékelés

A már korábban vizsgált feladat esetén bármely algoritmus ami minden esetben helyes választ ad, egyúttal egy igazolást állít ki a helyességéről: minden példányra ki tud választani egy levél-halmazt, amelyek együttes értéke garantálja, hogy a kiválasztott érték a helyes válasz. Korábban láttuk, hogy a kiértékelések számának várható értéke $n^{0.793}$, ami azt jelenti, hogy bármely fánál van egy olyan $n^{0.793}$ elemű levélhalmaz, hogy elemeinek értéke meghatározza a gyökér értékét. (létezik algoritmus, ami még ha véletlenített is $\Rightarrow$ mégis biztos tudást ad)
A valószínűségszámítási módszer 2 szintből áll, először tervezünk egy gondolatkísérletet, amiben a véletlen szerepet játszik, majd a második részben elemezzük a véletlenszerű kísérletet, és adott egyedi esetre vonunk le következtetést.

### 1.3. Maximális vágás feladat

Ebben a feladatban adott egy G(V,E) irányítatlan gráf n csúccsal és m éllel, és szeretnénk a csúcsokat két halmazra, (A és B) bontani úgy, hogy a két halmazt összekötő élek számát maximalizáljuk. A probléma NP-nehéz.

Állítás: $\exists A, B : A \cup B$ =V(G) ,amire $|(u,v) \in E,\ u \in A,\ v \in B| \geq \frac{m}{2}$

Bizonyítás: Készítsük el a két halmazt úgy, hogy minden csúcsot ½ valószínűséggel teszünk A-ba vagy B-be. Ekkor annak valószínűsége, hogy egy (u,v) él két végpontja más halmazban van ½ . Minden élre $\xi_i$ =1, ha benne van az i. él a vágásban (indikátor változó), és $\xi_i$ =0, ha nincs benne. A vágás mérete így a ($\xi_1 + ... + \xi_m$) érték lesz. Ennek várható értékére vagyunk kíváncsiak. A várható érték linearitása miatt

$E(\xi_1 + ... + \xi_m) = E(\xi_1) + ... + E(\xi_m) = P(\xi_1 = 1) + ... + P(\xi_m = 1) =$ ½ +…+ ½ = $\frac{m}{2}$

A valószínűségszámítási módszer fenti típusú gondolatkísérlete és egy véletlenített algoritmus között az a fő különbség, hogy az előbbiben csak azt akarjuk megmutatni, hogy valami nem 0 valószínűséggel létezik, míg utóbbiban a hatékonyság fontos szempont, nem tudunk elfogadni egy kis valószínűségű sikerességet. Például ha az előző állítással kapcsolatban csak azt tudtuk volna megmutatni, hogy $2^{-n}$ valószínűséggel sikerül m/2 nagyságú vágást találni, akkor képtelenek lennénk egy hatékony véletlenített algoritmust készíteni a maximális vágás megtalálására. Azonban mi azt mutattuk meg, hogy a vágás méretének várható érétke m/2, így ez a véletlenszerű felosztás tekinthető egy hatékony véletlenített algoritmusnak is. Az egyik kérdés, amit felteszünk az ilyen típusú példákban, hogy ha a valószínűségszámítási módszerrel bizonyítottuk valami létezését (bekövetkezését nem 0 valószínűséggel), akkor meg is tudjuk-e találni hatékonyan azt? A válasz esetenként más és más, van amikor találunk olyan determinisztikus polinomidejű algoritmust, ami megtalálja ezt a már garantáltan létező objektumot, van hogy egy véletlenített polinom idejű algoritmusunk van, ami magas valószínűséggel jut sikerre, de van amikor nem ismerünk ilyen algoritmusokat.

## 2. MAX-SAT feladat, approximációs algoritmus

### 2.1. MAX-SAT

Az ismert kielégíthetőségi feladatnak ebben a változatában a kérdés az, hogy maximum hány klóz elégíthető ki az adott KNF-ből. Feltételezhetjük azt, hogy egy klóz sem tartalmaz egyszerre egy literált és annak komplementerét (az ilyeneket minden kiértékelés kielégítené). Egy optimalizációt végzünk el, és ahelyett, hogy azt keresnénk, hogy melyik kiértékelés elégíti ki az összes klózt, maximalizálni szeretnénk a kielégített klózok számát.

Erről a MAX-SAT néven ismert problémáról ismerjük, hogy NP-nehéz, de a következő valószínűségszámítási módszer alapján megmutatjuk, hogy bármely m klózhalmazra van a változóknak olyan kiértékelése, ami legalább m/2 klózt kielégít. (Egyébként ez a lehető legjobb általános garancia, hiszen tartalmazhatja a halmaz két klóza pl. x-et és $\overline{x}$ -t, amikor is nem lehetséges a felénél többet kielégíteni)

Tétel: $\forall$ m klózból álló halmazra $\exists$ kiértékelés, amely legalább a klózok felét kielégíti.

Biz.: Feltételezzük, hogy a változóknak egymástól függetlenül ½ - ½ valószínűséggel igaz, vagy hamis az értékük. Legyen $\forall$ $1 \le i \le m$ -re $\xi_i$ indikátor változó értéke 1, ha az i. klózt kielégíti a kiértékelés, és 0, ha nem. Ekkor annak a valószínűsége, hogy egy klózt, ami mondjuk k db literált tartalmaz, nem elégít ki ez a véletlenszerű kiértékelés:

$\Pr(\xi=0) = \frac{1}{2^k}$ (hiszen ehhez az kell, hogy egyik literál sem legyen igaz érétkű, így k db független eseményről van szó). Így annak a valószínűsége, hogy az adott klózt kielégíti a kiértékelés: $\Pr(\xi=1) = 1 - \Pr(\xi=0) = 1 - \frac{1}{2^k} \ge \frac{1}{2}$ . A a kielégített klózok várható értéke a várható érték linearitását figyelembe véve így : $E(\xi_1 + ... + \xi_m) = E(\xi_1) + \ldots + E(\xi_m) \ge \frac{m}{2}$

### 2.2. $\alpha$–Approximációs algoritmusok

Az előző példánál maradva, mivel az NP-nehéz, keresünk egy olyan módszert, amivel közelíteni tudjuk a kielégített klózok számának maximumát. Erre szolgálnak az $\alpha$-approximációs algoritmusok. Egy adott $I$ példányra legyen $m(I)$ ez a maximum, $m_A(I)$ pedig egy A algoritmussal elérhető maximum. Ekkor definiálhatunk egy $\alpha$ értéket, ami az algoritmus hatékonyságát jellemzi: $\alpha = \inf \frac{m_A(I)}{m(I)}$ (ez alsó korlát az összes $I$ példányra). Ekkor az algoritmust $\alpha$-approximációs algoritmusnak nevezzük. Véletlenített algoritmusokra a fenti képlet $\alpha = \inf \frac{E(m_A(I))}{m(I)}$ formájú, ezek az $\alpha$-approximációs véletlen algoritmusok, ebben az esetben $m_A(I)$ értéke véletlen változó lesz. (Érdemes megjegyezni, hogy ellentétben a SAT problémával, itt lehetnek klózok, amiket kielégítetlenül hagyunk, sőt ez elkerülhetetlen, ezért a cél csak a lehetséges maximum megközelítése, nem az összes m klóz kielégítése.) A továbbiakban megadunk ehhez egy ¾-approximációs algoritmust.

Az ötlet az, hogy egy egészértékű lineáris programozási feladatként tekintsük a problémát, majd majd az ún. randomized rounding technikát használjuk. Minden $C_j$ klózhoz hozzárendelünk egy $z_j$ indikátor változót, ami a lineáris programozási feladatban jelöli, hogy a j. klóz ki van-e elégítve, valamint minden $x_i$ változóhoz is hozzárendelünk egy $y_i$ indikátor változót, ami az adott változó értékét jelzi (1, ha az TRUE és 0, ha FALSE), azaz $C_j \rightarrow z_j \in \{0,1\}$ , j=1..n és $x_i \rightarrow y_i \in \{0,1\}$ , i=1..n . Jelölje $C_j^+$ a negálatlan változók halmazát, $C_j^-$ pedig a negált változók halmazát a $C_j$ klózban. Ekkor a következő lineáris programozási feladatot fogalmazhatjuk meg:

célfüggvény: max $\sum_j z_j$ (a kielégített klózok száma)

feltételek: $y_i, z_j \in \{0,1\}$ $\forall$ i-re és j-re (1.)

$$\sum_{i \in C_j^+} y_i + \sum_{i \in C_j^-} (1 - y_i) \geq z_j \quad \forall j\text{-re} \quad \text{(2.)}$$

Az egészértékű programozási feladatot relaxálva megengedjük, hogy $y_i$ és $z_j$ valós értékeket vegyen fel, azaz $y_i$ , $z_j \in [0,1]$. Jelöljük a lineáris programozási feladat megoldásait $\hat{y}_i$ -vel és $\hat{z}_i$ -vel. Ekkor $\sum_j \hat{z}_j$ egy felső becslés lesz a kielégített klózok számára. Mivel $\hat{y}_i$ , $\hat{z}_i \in [0,1]$, tekintsük ezeket valószínűségeknek, és az algoritmusban minden $x_i$ változót $\hat{y}_i$ valószínűséggel állítsunk igazra a randomized rounding technikánk során.

<u>Tétel</u>: A fenti módszerrel készített kiértékelés során a kielégített klózok számának várható értéke legalább 1 – 1/e -szerese a maximálisan kielégíthető klózok számának.

<u>Biz.</u>: Mivel csak egy $C_j$ klózt vizsgálnánk, feltehetjük, hogy minden változó negálatlan (ha nem, átírható ilyenre ebben a klózban) Ekkor a klóz $x_1 \vee x_2 \vee ... \vee x_k$ formába írható. A lineáris programozási feladatbeli (2.) kényszer miatt $\hat{y}_1 + \hat{y}_2 + ... + \hat{y}_k \geq \hat{z}_j$ , ahol $\hat{y}_i$ annak a valószínűsége a fentiek szerint, hogy az $x_i$ igazra lesz állítva. Mivel ez az egyes változóknál függetlenül történik, ezért az, hogy mind hamis legyen $\prod_{i=1}^{k} (1 - \hat{y}_i)$ valószínűséggel történik meg. Tehát az az esemény, hogy van közte igaz $1 - \prod_{i=1}^{k} (1 - \hat{y}_i)$ valószínűséggel következik be. A $\beta_k$ jelölést bevezetve

$$1 - \prod_{i=1}^{k} (1 - \hat{y}_i) \geq [1 - (1 - \frac{1}{k})^k] \hat{z}_j = \beta_k \hat{z}_j \geq (1 - \frac{1}{e}) \hat{z}_j$$

(Felhasználtuk, hogy a képletben szereplő szorzat akkor minimális, ha minden $\hat{y}_i$ tagja = $\hat{z}_i / k$ és az így kapott $1-(1-z/k)^k$ egy konkáv függvény, úgyhogy nem lesz a két végpontban felvett (ami 0 és 1

mivel $0 \le z \le 1$) értékeknél kisebb, így lineáris függvényt is alkalmazhatunk helyette.)

A várható értékekre: $E(\sum_j z_j) = \sum_j E(z_j) \ge (1 - \frac{1}{e}) \sum \hat{z}_j$

Az eddig megismert két véletlenített algoritmus (az egyik, ami ½ valószínűséggel állít be minden változót igaz értékre, a másik az előbb ismertetett lineáris programozási módszer) használhatósága adott k értékekre a következőképpen alakul k függvényében:

| k | $1-2^{-k}$ | $\beta_k$ |
|---|---|---|
| 1 | 0.5 | 1.0 |
| 2 | 0.75 | 0.75 |
| 3 | 0.875 | 0.704 |
| 4 | 0.938 | 0.684 |
| 5 | 0.969 | 0.672 |

Látható, hogy a két algoritmus esetén $1-2^{-k}$ és a $\beta_k$ átlaga mindig $\ge 0.75$.

Jelölje $n_1$ az első és $n_2$ a második algoritmus várható eredményét.

Tétel: $\max\{n1; n2\} \ge \frac{3}{4} \sum_j \hat{z}_j$

Biz.: Elég megmutatni, hogy $(n_1+n_2)/2$ -re igaz, mert ez az átlag monoton csökken k növekedésével. Jelölje $S_k$ azt a klózhalmazt, amiben a k literált tartalmazó klózok az elemei. Ekkor $n_1$ -re és $n_2$ -re

$n_1 = \sum_k \sum_{C_j \in S^k} (1 - \frac{1}{2^k}) \ge \sum_k \sum_{C_j \in S^k} \hat{z}_j$ . Hasonlóan $n_2 \ge \sum_k \sum_{C_j \in S^k} \beta_k \hat{z}_j$ . Az átlagukra így

$\frac{n_1 + n_2}{2} \ge \sum_k \sum_{C_j \in S^k} \frac{(1 - \frac{1}{2^k}) + \beta_k}{2} \hat{z}_j$ . Mivel $\frac{(1 - \frac{1}{2^k}) + \beta_k}{2} = \frac{(1 - \frac{1}{2^k}) + [1 - (1 - \frac{1}{k})^k]}{2} \ge \frac{3}{4}$ ,

a tételt bebizonyítottuk. Elmondhatjuk tehát, hogy bármely példány esetén valamelyik algoritmus egy ¾ – approximációs algoritmus, ezért mindkettőt futtatva, majd a jobb megoldást választva a klózok legalább ¾ -ét ki tudjuk elégíteni.

# 3. Lovász Lokális Lemma

## 3.1. Bevezetés, fogalmak

Teljes függetlenség: Pr(AB)=Pr(A)Pr(B)

S={$A_1$,...,$A_n$} , T⊆S esetén $\Pr(\cap_{A_i \in T} A_i) = \prod_{A_i \in T} Pr(A_i)$

Példa: Egy tetraéderrel dobunk, amin az 1,2,3,4 számok vannak.

Legyen az A esemény: a dobott érték ≤2 ⇒ Pr(A)= ½

B esemény: a dobott érték páros ⇒ Pr(B)= ½

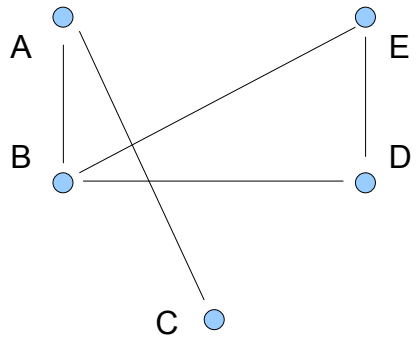
C esemény: a dobott érték prím ⇒ Pr(C)= ½

Ezek az események páronként függetlenek, de a teljes rendszer nem, azaz Pr(ABC) ≠ 1/8.

Az A teljesen független S-től: ∀ T⊆S -re $\Pr(A \mid \cap_{A_i \in T} A_i) = P(A)$

Függőségi gráf: $A_1$, …, $A_n$ események a csúcsok. Teljesülnie kell, hogy az $A_i$ esemény legyen teljesen független az események azon halmazától, amelyek vele nem szomszédosak (azaz nem köti össze él).

Legyen egy KNF, amit gráffal úgy ábrázolunk, hogy ha van közös változó (negált vagy negálatlan mindegy), akkor húzok élt az egyes klózok közé, pl.:




Itt C értéke a {B,D,E} -től független (nincs közös változó)

## 3.2. Lovász Lokális Lemma

Legyen G(V,E) az $A_1$,...,$A_n$ események függőségi gráfja egy valószínűségi mezőben. Tfh. ∀ eseményhez ∃ $x_i$∈[0,1], amire igaz, hogy ∀ 1 ≤ i ≤ m -re igaz az, hogy ha

$Pr(A_i) \le x_i \prod_{(i,j) \in E} (1 - x_j)$ , akkor $Pr(\cap_{i=1}^{m} A_i) \le \prod_{i=1}^{m} (1 - x_i)$ .

Következmény: Ha $A_1$ ,..., $A_n$ események, és van egy függőségi gráf, valamint ∀ i-re $Pr(A_i) \leq p$. Azon feltételek mellett, hogy egyrészt ∀ esemény soktól teljesen független (legfeljebb d kivétellel az összes többitől), másrészt $ep(d+1) \leq 1$ („ep" = felső becslés) igaz a következő: $Pr(\cap_{i=1}^{m} \overline{A_i}) > 0$

### 3.3. k-SAT probléma

Alkalmazzuk most a fenti következményt a k-SAT probléma egy példányára. (Amiben az adott KNF-ben minden klóz k db literált tartalmaz) Nézzünk egy olyan változatot, ahol ∀ változó legfeljebb $2^{k/50}$ klózban szerepel. Állításunk az, hogy a fenti k-KNF mindig kielégíthető. Tekintsük azt a kiértékelést, amiben ½ valószínűséggel kapnak igaz vagy hamis értékeket az egyes változók. Jelölje $A_i$ azt az eseményt, hogy a $C_i$ klóz nincs kielégítve ($1 \leq i \leq m$ ). Mivel mindegyik klóz k db literált tartalmaz $Pr(A_i)=2^{-k}$. Az $A_i$ esemény független minden $A_j$ eseménytől, kivéve azoktól, ahol ahol a $C_i$ és $C_j$ klózban legalább egy változó közös. $d=k*2^{k/50}$ értékkel jelöljük, hogy legfeljebb hány szomszédja van egy csúcsnak a gráfban. Alkalmazzuk tehát a Lovász Lokális Lemma következményét $d=k*2^{k/50}$ értékkel, és következtetünk, hogy pozitív valószínűséggel a véletlen kiértékelés kielégíti az összes m db klózt. Ennélfogva biztosan létezik kielégítő kiértékelés a fenti feltételeket teljesítő k-SAT példányokra. A Lovász Lokális Lemma következménye azt mutatja, hogy egy véletlen kiértékelés pozitív valószínűséggel kielégítő, de ez a valószínűség egészen kicsi is lehet. Ezért előfordulhat, hogy többször kell próbálkoznunk, mire találunk egy megfelelőt, ami az összes m db klózt kielégíti.

### 3.4. Algoritmus a k-SAT problémára

A következőkben leírunk egy Las Vegas típusú algoritmust, ami polinom időben fut m függvényében (de nem polinom idejű k függvényében) és alkalmas egy kielégítő kiértékelés megtalálására. Az algoritmus futása során bármely időpontban lehetnek változók, amik már fixálva vannak 0 vagy 1 értékkel, míg lehetnek olyanok amik még nem kaptak értéket. Kezdetben még egyik változónak sincs értéke. A k értékét konstansnak tekintjük. Az algoritmus két fázisból áll. Az első fázis fixálja néhány változó értékét, és a maradékot elhalasztja a második fázis részére.

1.fázis

Szekvenciálisan haladva a változókon, ½ valószínűséggel igaz vagy hamis értékre rögzítjük őket. Közben ún. veszélyes klózok keletkeznek, amik két kritériuma a következő :

1. k/2 literál már ki van értékelve a $C_i$ klózból
2. $C_i$ még nincs kielégítve

A veszélyessé vált klózokban található többi literál (a másik fele) rögzítését elhalasztjuk, átugorva a szekvenciális értékadásban. A fázis végén azt mondjuk egy klózról, hogy túlélő, ha nincs kielégítve az első fázisban rögzített változók által. A második fázisban csak azokkal a változókkal foglalkozunk, amik az első fázisban még nem kaptak értéket, valamint a túlélő klózokkal. Egy $C_i$ klóz akkor lehet túlélő a fentiek alapján, ha

1. Veszélyessé vált
2. Szomszédja veszélyessé vált (ugyanis ekkor van közös változójuk, ami a szomszéd miatt még nem kapott értéket)

Mindegyik túlélő klóznak legalább k/2 változója van, ami még nem kapott értéket, és [1,d+1] db klóz veszélyessé válásának eredményeképp válhatott veszélyessé.

2.fázis

A túlélő klózok által feszített G részgráfot nézzük. A valószínűsége, hogy egy bármelyik klóz az 1.fázis során veszélyessé vált, legfeljebb $2^{-k/2}$, hiszen pontosan k/2 rendelkezik a literáljai közül rögzített értékkel, és ezek közül egy sem elégíti ki a klózt. Így annak valószínűsége, hogy a klóz túlélő legfeljebb $(d+1)2^{-k/2}$ . (ezzel az értékkel felülről becsültük, mert $P(A_1 \cup ... \cup A_n) \leq P(A_1)+...+ P(A_n)$) Meg kell még jegyezni, hogy a G részgráfban két összekapcsolt túlélő klóz nem tartalmazhat közös elhalasztott változót. Ezért az ilyen változók egyértelműen kijelölnek összefüggő komponenseket a részgráfban. Bármelyik ilyen komponensre igaz, hogy a még nem rögzített változók száma a komponens klózaiban O(log m), így polinom m időben fel tudjuk sorolni a $2^{O(\log m)}$ igaz kiértékelést ezekre a változókra, amíg találunk egyet, ami kielégíti a komponens összes klózát. Ezt ismételjük a második fázisban, függetlenül mindegyik összekapcsolt komponensre, addig, míg a nem rögzített változók is értéket kapnak, így kielégítve az összes túlélő klózt. A módszerrel tehát végeredményben polinom m időben kielégítő kiértékelést találunk.

Készítette: Oláh Péter