



MISKOLCI EGYETEM  
GÉPÉSZMÉRNÖKI ÉS INFORMATIKAI KAR



## TUDOMÁNYOS DIÁKKÖRI DOLGOZAT

# A polyhipergeometrikus eloszlás értékeinek megadása

**Sülyi Ákos**

Programtervező Informatikus BSc, III. évfolyam

**Rjabcsenkó Eugén**

Programtervező Informatikus BSc, III. évfolyam

**Dr. Fegyverneki Sándor**

egyetemi docens, tanszékvezető

Alkalmazott Matematikai Tanszék

**Dr. Karácsony Zsolt**

egyetemi docens

Alkalmazott Matematikai Tanszék

Miskolc, 2012



# Tartalomjegyzék

|                                                                       |    |
|-----------------------------------------------------------------------|----|
| 1. Bevezetés .....                                                    | 1  |
| 2. A probléma megfogalmazása .....                                    | 2  |
| 3. A probléma megoldása .....                                         | 4  |
| 3.1 Particionálás és ismétléses permutáció együttes alkalmazása ..... | 4  |
| 3.2 Hálóval történő kódolás, a háló szintjeinek felsorolása .....     | 6  |
| 4. Alkalmazási lehetőségek .....                                      | 17 |
| 5. Összefoglalás .....                                                | 19 |
| 6. Nyitott problémák .....                                            | 20 |
| Irodalomjegyzék .....                                                 | 21 |

## 1. fejezet

# Bevezetés

Dolgozatunk megrásához, és a megoldáshoz vezető úthoz egy televíziós vetélkedő adta az ötletet. A játékban a nyereményt különböző összegekre bontották, melyek között vannak azonosak, pontosabban mindegyik részösszeg kétszer szerepel. A játékos ezeket körökre bontot kérdé-felelek játékban szerezheti meg véletlenszerűen. Kijelölt számú kör után a játékosnak döntenie kell, hogy tovább játszik-e kockáztatva, hogy kiesik a játékból és az addig megszerzett nyereményét elveszíti, vagy abbahagyja megtartva azt. Ezt a döntési helyzetet valószínűség-számítás szempontjából vizsgálva a nyeremény várhatóértékét kell kiszámítanunk ahhoz, hogy ítéletet tudjunk adni, hogy a nyeremény milyen mértékben tekinthető soknak. Könnyen belátható, hogy a nyeremény polihipergeometrikus eloszlásból származó valószínűségi változóként adható meg. Kézenfekvően adódik a kérdés, hogy ezen értékek megadása, hogyan algoritmizálható. Az alkalmazás során a probléma megoldásokban általában az eloszlás közelítésével ez áthidalható. Kellően nagyszámú sokaságból való húzásoknál, mikor a húzások kimenetelét elhanyagolható mértékben befolyásolja egy elem eltávolítása, jól alkalmazható a polinomiális eloszlással történő közelítés.

Dolgozatunkban sikerült olyan felsorolását megadnunk az eloszlás eseményeinek, ami egyszerűen és gyorsan előállítható.

## 2. fejezet

### A probléma megfogalmazása

Tekintsük a klasszikus urna modellt, azaz mikor egy üvegben  $n$ -féle cukorka van, ezek száma legyen  $k_1, k_2, \dots, k_n$ , továbbá az összes cukorka száma pedig  $\sum_{i=1}^n k_i = N$ . *Visszatevés nélkül* kihúzunk  $h$  darabot, akkor annak a valószínűsége, hogy amit kihúztunk azok között  $q_1, q_2, \dots, q_n$ ,  $\left(\sum_{i=1}^n q_i = h\right)$  darab van az egyes cukorkákból, a polihipergeometrikus eloszlás adja meg a következő formában:

$$\mathbf{P}(\xi_1 = q_1, \xi_2 = q_2, \dots, \xi_n = q_n) = \frac{\binom{k_1}{q_1} \binom{k_2}{q_2} \dots \binom{k_n}{q_n}}{\binom{N}{h}}. \quad (2.1)$$

A polihipergeometrikus eloszlás speciális esetének tekinthető a hipergeometrikus eloszlás  $n = 2$  esetén. Ennek számos alkalmazási területe van. Egyik a selejtek számának becslése. Ekkor ismert számosságú sokaságból veszünk mintát és a mintában lévő selejtek száma alapján becsüljük az összes selejt számát. Egy másik megközelítés amikor az alapsokaság méretére szeretnénk becslést adni. Például az ökológiai mérések egyik célja lehet egy véges populáció méretének megállapítása. Ennek érdekében az egyedek egy részét oly módon jelölik, hogy az semmilyen módon ne befolyásolja a jelölt egyedek későbbi visszafogását. Ezután megvárva a „tökéletes” keveredéshez szükséges időt, az újabb befogás alakalmával egyedül a véletlen határozza meg, hogy jelölt vagy jelöletlen egyedet fogunk, így a jelöltek számának ismeretében a hipergeometrikus eloszlás segítségével becslést tudunk adni a teljes populáció méretére.

Természetesen számos szerencsejáték kimenetelének valószínűségeit is hipergeometrikus eloszlással adhatjuk meg. A Texas-Hold'em Pokerban az asztalon három, majd négy végül

öt lap van felfordítva, a játékos kezében pedig kettő lap van. A játékos a felfordított öt lap közül hármát kell kiválasztania a kezében tartott két lapjához. A flush esélyét, vagyis annak, hogy öt egyforma színű lapot tud kiválasztani, annak ismeretében, hogy hány lapot nem lát a játékos és azok közül hánynak megfelel a színe az ő számára, a hipergeometrikus eloszlás segítségével tudja kiszámítani minden lap felfordítása után.

A polihipergeometrikus eloszlás akkor alkalmazható, ha a kimenetek száma kettőnél több. Például, ha egy nyereményjátékban a játékos körönként véletlenszerűen szerezheti meg a különböző értékű nyereményeket. Az egyes körökben a nyeremény várható értékét polihipergeometrikus eloszlással számíthatjuk ki.

További fontos alkalmazási terület a homogenitás vizsgálat. Amikor annak az eldöntése a cél, hogy az egyes kategóriákból egyenlő számú elem van-e a sokaságban, amiből a mintánk származik. Ennek eldöntésére használják a polihipergeometrikus eloszlás különböző közelítéseit.

A polihipergeometrikus eloszlás alkalmazási területe tehát igen széles, azonban körülményes kiszámítási módja erősen korlátozza a felhasználhatóságát. Az összes valószínűségértékének előállításához szükséges, hogy a húzások eseményeit fel tudjuk sorolni. Amennyiben a fenti képletet alkalmazzuk ez egybeesik a  $q_1, q_2, \dots, q_n$  számok előállításával. Ebben az esetben a binomiánális együthatók számításakor ügyelnünk kell a számítógép ábrázolási tartományára, valamint a hányadost is érdemes úgy képezni, hogy felváltva osztunk a nevezővel és szorzunk a számláló tényezőivel.

Ebből a dolgozatból kiderül, hogy az eloszlás megadható a 3.1. Tétel alapján az elemi események valószínűségeinek összegeként is.

### 3. fejezet

## A probléma megoldása

#### 3.1. Particionálás és ismétléses permutáció együttes alkalmazása

Az említett két megadás közül a  $q_1, q_2, \dots, q_n$  paraméterek előállítása két lépésben oldható meg:

- 1.) a kihúzott elemek számának összegre bontása, particionálása,
- 2.) a részösszegek permutálása.
- 3.) iteráció a húzások számával 1-től  $N$ -ig az 1.) és 2.) pontokkal.

Az particionálási lépés során azonban figyelembe kell venni, hogy  $q_1 \leq k_1, q_2 \leq k_2, \dots, q_n \leq k_n$  feltételeket, illetve a részösszegnek  $n$ -nél nem lehet több tagja. Az általunk particionálásra használt algoritmus a felbontásokban a részösszegeket csökkenő sorrendben adja meg. Tehát kínálkozik az a lehetőség, hogy a legnagyobb  $k_i$  és a felbontás első elemének összevetésével kiszűrjük azokat, melyeknek egyetlen permutációja sem fogja teljesíteni a  $q_i \leq k_i$  feltételeket ( $i = 1, 2, \dots, n$ ). Azonban a megmaradt felbontások minden permutációjára ellenőriznünk kell őket. Azokat az eseteket amelyekre nem teljesül az összes feltétel szintén el kell hagynunk. A megmaradt elemek pontosan a lehetséges húzások lesznek. Ezek valószínűségei meghatározhatóak a korábban bevezett (2.1) formulával.

Tehát az említett két algoritmus együttes alkalmazásával sikerült a polihipergeometrikus eloszlás valószínűségértékeit megadnunk, még hozzá minden eseményt egyszer vizsgálva.

A továbbiakban közöljük az említett két módszer együttes használatának algoritmusát.

---

**Requires:**  $A, n$  (  $a_1 \geq a_2 \geq \dots \geq a_h, n > 0$  )

---

```

1:   $h \leftarrow \text{length}(A)$ 
2:   $k \leftarrow h - 1$ 
3:   $H_{h-1} \leftarrow n$ 
4:  while  $k < h$  do
5:     $x \leftarrow B_{k+1} + 1$ 
6:     $y \leftarrow B_k - 1$ 
7:     $B_k \leftarrow 0$ 
8:     $k \leftarrow k + 1$ 
9:    while  $x \leq y$  and  $k > 1$  do
10:      $B_k \leftarrow x$ 
11:      $y \leftarrow y - x$ 
12:      $k \leftarrow k - 1$ 
13:  end while
14:   $B_k \leftarrow x + y$ 
15:  if  $k > 1$  then
16:     $B \leftarrow [B_k, B_{k+1}, \dots, B_h, B_1, B_2, \dots, B_{k-1}]$ 
17:  end if
18:  for  $j \leftarrow 1$  to  $h$ 
19:     $p \leftarrow A_j - B_j$ 
20:    if  $p < 0$  then break end if
21:  end for
22:  if  $p \geq 0$  then
23:     $i \leftarrow 1$ 
24:    if  $B_1 \neq B_h$ 
25:       $B \leftarrow [B_h, B_1, B_2, \dots, B_{h-1}]$ 
26:    visit  $B$ 
27:    while  $i < h$  do
28:      if  $i \leq h - 2$ 
29:        if  $B_{i+2} > B_i$  then
30:           $B \leftarrow [B_{i+1}, B_1, B_2, \dots, B_i, B_{i+2}, B_{i+3}, \dots, B_h]$ 
31:        else
32:           $B \leftarrow [B_{i+2}, B_1, B_2, \dots, B_{i+1}, B_{i+3}, B_{i+4}, \dots, B_h]$ 
33:        end if
34:      else
35:         $B \leftarrow [B_h, B_1, B_2, \dots, B_{h-1}]$ 
36:      end if
37:      if  $B_1 < B_2$  then
38:         $i \leftarrow 1$ 
39:      else
40:         $i \leftarrow i + 1$ 
41:      end if

```

---

```

42:          for  $j \leftarrow 1$  to  $h$ 
43:               $p \leftarrow A_j - B_j$ 
44:              if  $p < 0$  then break end if
45:          end for
46:          if  $p \geq 0$  then
47:              visit  $B$ 
48:          end if
49:      end while
50:  end if
51: end if
52: if  $k > 1$  then
53:      $B \leftarrow [B_{h-k+2}, B_{h-k+3}, \dots, B_h, B_1, B_2, \dots, B_{h-k+1}]$ 
54: end if
55: end while

```

---

A fenti algoritmus bemenetként egy tömböt és egy számot vár. A tömb elmei, legyenek csökkenő sorrendbe rendezve, ahol a tömb elemei az alapsokaság kategóriáinak a számai. A másik bemenetként megadott szám, pedig annak a száma, hogy hány elemet emelünk ki belőle. Kimeneteli értéként a  $B$  tömbök egy felsorolását kapjuk, melyek a kiemelt elemek kategóriánkénti számait tartalmazzák.

### 3.2. Hálóval történő kódolás, a háló szintjeinek felsorolása

Az előző alfejezetben ismertetett algoritmust hasonlítsuk össze egy fastruktúrával történő kódolással, mely minden elem kihúzására egy újabb ágot eredményez. A leírásból is érezhető a fa redundanciája. Ez szolgáltatta a kiindulópontot a következő megközelítéshez. A továbbiakban az elemi események egy felsorolását adjuk meg, az ismétlődőket továbbra is ugorjuk át. Ehhez először egy elem kihúzását szükséges értelmeznünk.

#### 3.1. DEFINÍCIÓ (csökkentés):

Tekintsünk a  $\mathbf{q} \in \mathbb{N}^n$ , ahol  $n \in \mathbb{N} \setminus \{0\}$  vektort, aminek az egyes komponensei tárolják a különbözőféle húzások számát, továbbá jelölje  $n$  a komponensek számát.

A húzások leírására vezessük be a következő egyszerű műveletet:

$$\sigma(\mathbf{q}, i) = \begin{cases} (q_1, q_2, \dots, q_i - 1, \dots, q_n) & , \text{ ha } q_i \neq 0, \\ \text{nincs értelmezve} & , \text{ ha } q_i = 0, \end{cases}$$

$$\text{ahol } \mathbf{q} = (q_1, q_2, \dots, q_i, \dots, q_n).$$

Ez a művelet az  $i$ -edik komponenst csökkenti eggyel.

Példának vegyük a  $\mathbf{q} = (3, 2, 1, 1)$  vektort. Ebben az esetben az  $i = 1, 2, 3, 4$  lehet. Tekintsük továbbá az  $\mathbf{r}_1 = (3, 2, 0, 1)$  és  $\mathbf{r}_2 = (2, 2, 1, 1)$  vektorokat, amik előállíthatóak  $\mathbf{q}$ -ból.

Ekkor a  $\sigma(\mathbf{q}, i)$ -re ( $i = 1, 2, 3, 4$ ) a következők adódnak:

$$\begin{aligned} \sigma(\mathbf{q}, 1) &= (2, 2, 1, 1), & \sigma(\mathbf{q}, 2) &= (3, 1, 1, 1), \\ \sigma(\mathbf{q}, 3) &= (3, 2, 0, 1), & \sigma(\mathbf{q}, 4) &= (3, 2, 1, 0). \end{aligned}$$

Hasonlóan  $\sigma(\mathbf{r}_1, i)$  és  $\sigma(\mathbf{r}_2, i)$ -re kapott vektorok:

$$\begin{aligned} \sigma(\mathbf{r}_1, 1) &= (2, 2, 0, 1), & \sigma(\mathbf{r}_1, 2) &= (3, 1, 0, 1), & (3.1) \\ \sigma(\mathbf{r}_1, 3) &= \perp, & \sigma(\mathbf{r}_1, 4) &= (3, 2, 0, 0), \\ \sigma(\mathbf{r}_2, 1) &= (1, 2, 1, 1), & \sigma(\mathbf{r}_2, 2) &= (2, 1, 1, 1), \\ \sigma(\mathbf{r}_2, 3) &= (2, 2, 0, 1), & \sigma(\mathbf{r}_2, 4) &= (2, 2, 1, 0). \end{aligned}$$

Azt az esetet amikor a művelet nincs értelmezve  $\perp$  jellel jelöljük.

### 3.2. DEFINÍCIÓ (követés):

Legyen  $\mathbf{q}$  és  $\mathbf{r} \in \mathbb{N}^n$  vektor. Azt mondjuk, hogy a  $\mathbf{q}$  követi  $\mathbf{r}$  vektort (jelölés:  $\mathbf{q} \kappa \mathbf{r}$ ), ha  $\mathbf{q} - \mathbf{r} = \mathbf{e}$ , ahol  $\mathbf{e}$  az egyik egységvektor.

Továbbá a  $\delta : \{0, 1\}^n \rightarrow \mathbb{N} \setminus \{0\}$  függvény adja meg a csökkentett komponens indexét, azaz

$$\begin{aligned} \delta(\mathbf{e}) &= i, \text{ ha } e_i = 1, \text{ és } e_j = 0, \text{ ha } i \neq j, \\ \text{ahol } \mathbf{e} &= (e_1, e_2, \dots, e_n), \text{ és } e_1, e_2, \dots, e_n \in \{0, 1\}. \end{aligned} \quad (3.2)$$

Jellemezzük a  $\mathbf{q} \kappa \mathbf{r}$  relációt a  $w$  súllyal a következők szerint:

$$w(\mathbf{q}, \mathbf{r}) = \begin{cases} \frac{q_j}{s} & , \text{ ha } \mathbf{q} \kappa \mathbf{r} , \\ 0 & , \text{ egyébként,} \end{cases}$$

$$\text{ahol } \mathbf{q} = (q_i)_{i=1}^n , \quad s = \sum_{i=1}^n q_i , \quad 1 \leq i \leq n, \quad \text{és } j = \delta(\mathbf{q} - \mathbf{r}) .$$

Például az előbbi (3.1)-el jelölt példában szereplő vektorokra a  $w$  értékei a következők:

Tekintsük először a  $\mathbf{q} = (3, 2, 1, 1)$  vektort. Ekkor:

$$\begin{aligned} w(\mathbf{q}, (2, 2, 1, 1)) &= \frac{3}{7}, & w(\mathbf{q}, (3, 1, 1, 1)) &= \frac{2}{7}, \\ w(\mathbf{q}, (3, 2, 0, 1)) &= \frac{1}{7}, & w(\mathbf{q}, (3, 2, 1, 0)) &= \frac{1}{7}. \end{aligned}$$

Hasonlóan kapjuk meg a  $w$  értékeit  $\mathbf{r}_1 = (3, 2, 0, 1)$  és  $\mathbf{r}_2 = (2, 2, 1, 1)$  esetén is:

$$\begin{aligned} w(\mathbf{r}_1, (2, 2, 0, 1)) &= \frac{3}{6} = \frac{1}{2}, & w(\mathbf{r}_1, (3, 1, 0, 1)) &= \frac{2}{6} = \frac{1}{3}, & (3.3) \\ w(\mathbf{r}_1, (3, 2, 0, 0)) &= \frac{1}{6}, \\ w(\mathbf{r}_2, (1, 2, 1, 1)) &= \frac{2}{6} = \frac{1}{3}, & w(\mathbf{r}_2, (2, 1, 1, 1)) &= \frac{2}{6} = \frac{1}{3}, \\ w(\mathbf{r}_2, (2, 2, 0, 1)) &= \frac{1}{6}, & w(\mathbf{r}_2, (2, 2, 1, 0)) &= \frac{1}{6}, \\ w(\mathbf{r}_1, (1, 2, 1, 1)) &= 0, & w(\mathbf{r}_2, (3, 1, 0, 1)) &= 0. \end{aligned}$$

Könnyen belátható, hogy  $w(\mathbf{q}, \mathbf{r})$  pontosan annak a valószínűségével egyezik meg, mikor egy olyan alapsokaságból húzunk ki egy elemet, aminek elemei úgy oszthatóak szét kategóriákba, hogy azok számossága  $\mathbf{q}$  elemeit adják.

Az előbbi definíciókat felhasználva az eloszlás valószínűségeit a következő tétel szerint tudjuk felbontani az elemi események valószínűségeinek összegére.

### 3.1. TÉTEL :

Jelölje  $A_i^{(h)}$  azokat az eseményeket, amikor egy alapsokaságból  $h = 0, 1, \dots, N$  darab elemet húzunk ki, ahol  $i = 1, 2, \dots, m_h$ , és  $m_h$  a lehetséges húzások száma. Továbbá az

$A_i^{(h)}$  révén előálló halmazok kategóriáinak számosságait tartalmazzák az  $\mathbf{a}_k^{(h)}$  vektorok. A valószínűségeket, melyek azonos számú elem kihúzásához tartoznak tároljuk vektorokban a következőképpen:  $\mathbf{p}^{(h)} = \left\{ \mathbf{P} \left( A_i^{(h)} \right) \right\}_i$ .

Ekkor  $\mathbf{p}^{(h)}$  előáll a következő alakban:

$$\mathbf{p}^{(h)} = \left( \left( \mathbf{p}^{(h-1)} \right)^\top \mathbf{W}^{(h-1)} \right)^\top.$$

Kihasználva a rekurziót a következő formulát kapjuk:

$$\mathbf{p}^{(h)} = \underbrace{\left( \underbrace{\mathbf{W}^{(1)} \mathbf{W}^{(2)} \mathbf{W}^{(3)} \dots \mathbf{W}^{(h-1)}}_{(\mathbf{p}^{(2)})^\top} \right)^\top}_{(\mathbf{p}^{(3)})^\top} = \left( \prod_{l=1}^{h-1} \mathbf{W}^{(l)} \right)^\top,$$

ahol  $\mathbf{W}^{(h)} = \left\{ \mathbf{P} \left( A_i^{(h)} \middle| A_j^{(h-1)} \right) \right\}_{i,j} = \left\{ w \left( \mathbf{a}_i^{(h)}, \mathbf{a}_j^{(h-1)} \right) \right\}_{i,j}$ , azzal a megkötéssel, hogy  $\mathbf{W}^{(h)} = \left\{ \mathbf{w}_i^{(h)} \right\}_i$  oszlopai ugyanabban a sorrendben feleltethetők meg az  $A_i^{(h)}$  eseményeknek, mint  $\mathbf{W}^{(h+1)}$  sorai.

### 3.1. BIZONYÍTÁS :

Könnyen belátható, hogy az  $A_i^{(h)} \middle| A_j^{(h-1)}$  esemény megfeleltethető egy elem kihúzásának, aminek a valószínűsége éppen a  $w \left( \mathbf{a}_j^{(h-1)}, \mathbf{a}_i^{(h)} \right)$ . Az állítás szerint a  $\mathbf{p}^{(h)}$  egy komponense megkapható  $p_i^{(h)} = \left( \mathbf{p}^{(h-1)} \right)^\top \mathbf{w}_i^{(h)}$  vektorok szorzataként, ahol

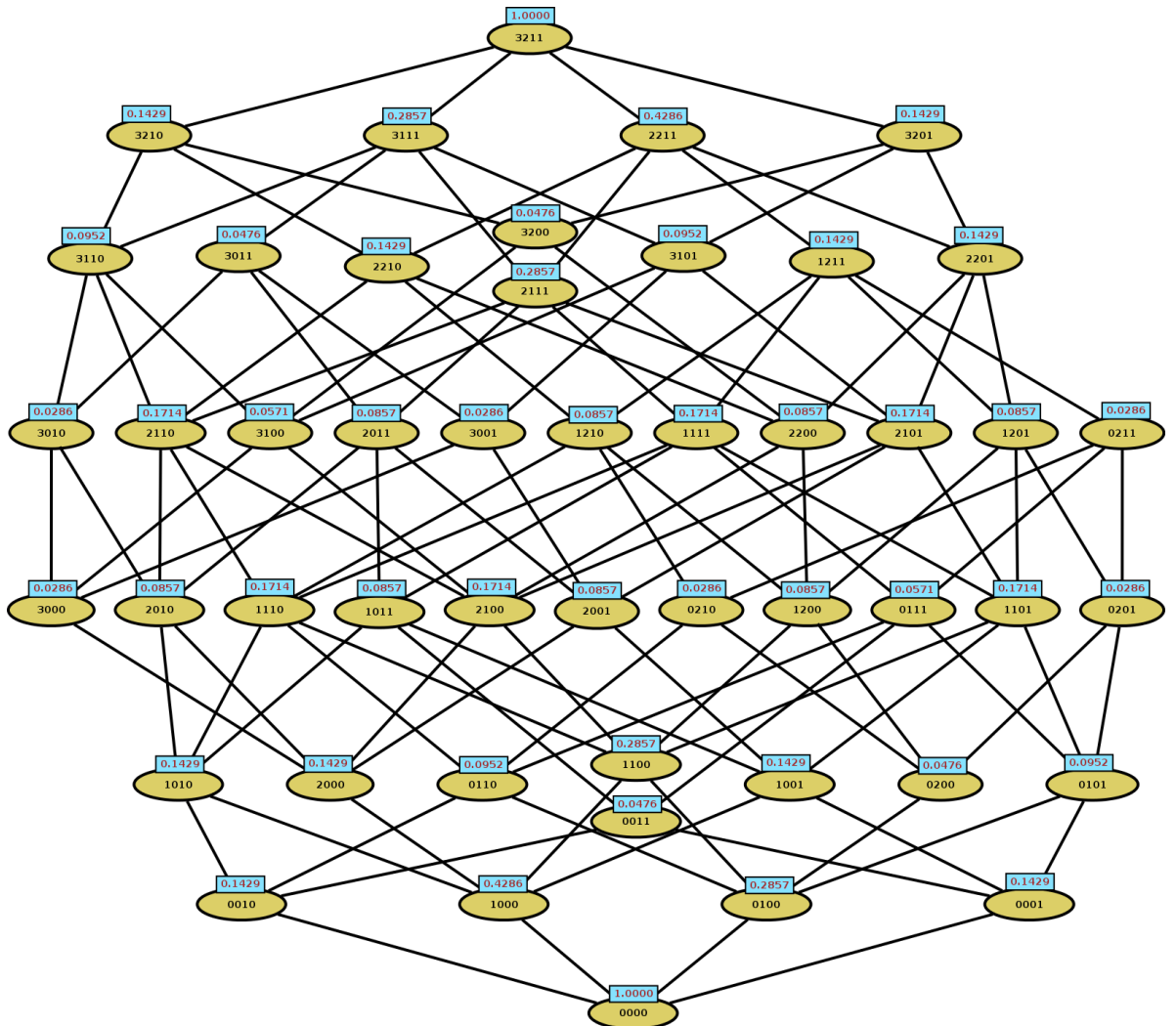
$$\begin{aligned} p_i^{(h)} &= \sum_j^{m_h} w_{i,j}^{(h)} \cdot p_j^{(h-1)} = \\ &= \mathbf{P} \left( A_i^{(h)} \right) = \sum_j^{m_h} \mathbf{P} \left( A_i^{(h)} \middle| A_j^{(h-1)} \right) \mathbf{P} \left( A_j^{(h-1)} \right). \end{aligned}$$

Innen a tétel állítását megkaphatjuk a teljes valószínűség tételéből, ahol  $m_h$  annak a száma, hogy hányféleképpen tudjuk kihúzni a  $h - 1$  elemet.

Ezen  $\mathbf{W}^{(h)}$  mátrixok az  $A_k^{(h)}$  események, illetve a nekik megfeleltethető  $\mathbf{a}_k^{(h)}$  vektorok egy hálóba történő rendezését írják le. Ez a háló szintekre bontható aszerint, hogy hány

elemet húztunk ki. A legfelső szinten maga az alapsokaság áll, alatta azok a vektorok, amelyekből egy elemet húztunk ki, majd ezután azok a vektorok, melyeből két elemet húztunk ki. Ezek közül azok, amelyekből két különbözőt húztunk a megelőző szint két halmazát is követi, ugyanis ezek előállhatnak akár úgy, hogy előbb az egyik féleből húzunk egyet, majd a másikból, illetve fordítva is. A többi szintekre is igaz, hogy a vektorok az előző szint annyi vektorát követik, ahány komponensük eltér az alapsokaság vektorának megfelelő komponensétől.

A következő ábrán az elipszisekben az  $\mathbf{a}_i^{(h)}$  vektorok, a felettük lévő téglalapokban, pedig a hozzájuk tartozó valószínűségük szerepel. Egy elipszis akkor van összekötve egy felette állóval, ha abból egy elemet kihúzva előállítható. Az egyes szinteken belül a sorrendet a követések áttekinthetősége érdekében választottuk meg így.



A háló egyes éleit súlyozhatjuk a két végpontjához rendelhető  $w$  értékkel. Egy csúcshoz tartozó valószínűség kiszámítható a 3.1 Tétel alapján úgy, hogy a bemenő élek másik végpontjainak valószínűségeit a  $w$ -kel súlyozva összegezzük.

A fentiekben a (3.1) példában ismertetett értékekkel a  $(2,2,0,1)$  valószínűsége, ami az ábrán a második szinten jobbról az első vektor, a következőképpen alakul:

$$\mathbf{P}(\mathbf{r}_1)w(\mathbf{r}_1(2,2,0,1)) + \mathbf{P}(\mathbf{r}_2)w(\mathbf{r}_2(2,2,0,1)) = \frac{1}{7} \cdot \frac{1}{2} + \frac{3}{7} \cdot \frac{1}{6} = \frac{1}{7} \approx 0,1429. \quad (3.4)$$

Az eddigiek arra nem adnak választ, hogy a mátrix oszlopait milyen sorrendbe töltsük ki. Tehát arra kell megoldást találni, hogy azokat a vektorokat soroljuk fel hatékonyan, amelyekből az alapsokasághoz képest  $h$  elemet húztunk ki. Mégpedig úgy, hogy minden vektort pontosan egyszer sorolunk fel.

### 3.3. DEFINÍCIÓ (hozzáfűzés):

Jelöljük  $\oplus$ -al azt a műveletet, amely a felsorolás végéhez hozzáfűz egy elemet.

$$\mathbf{S}_h \oplus \mathbf{a}_i^{(h)} = \begin{cases} \mathbf{S}_h & , \text{ ha } \mathbf{a}_i^{(h)} \text{ nincs értelmezve,} \\ \langle \mathbf{S}_h, \mathbf{a}_i^{(h)} \rangle = \langle \mathbf{a}_1^{(h)}, \mathbf{a}_2^{(h)}, \dots, \mathbf{a}_{i-1}^{(h)}, \mathbf{a}_i^{(h)} \rangle, & \text{ egyébként,} \end{cases}$$

ahol  $\mathbf{S}_h = \langle \mathbf{a}_1^{(h)}, \mathbf{a}_2^{(h)}, \dots, \mathbf{a}_{i-1}^{(h)} \rangle$ .

Az így bevezett művelettel lehetőség nyílik egy korábbi  $h$  szint vektorain végighaladva előállítanunk a  $(h+1)$ -dik szinten lévő vektorokat a  $\sigma$  csökkentési művelet segítségével.

Ennek algoritmus a következő képpen alakul:

- 1.) Legyen  $h = 0$ , valamint induljunk ki az alaphalmazból oly módon, hogy a kategóriák számosságait csökkenő sorrend szerint rendezve írjuk az  $\mathbf{a}^{(0)}$  vektor komponenseibe. Az így felírt  $\mathbf{a}^{(0)}$  vektort helyezzük el az  $\mathbf{S}_0$  szinten.
- 2.) Ezután állítsuk elő a következő,  $h+1$  szintet úgy, hogy vesszük a  $h$ -dik szint első vektorát és a komponenseit az utolsóval kezdve egymás után csökkentjük egyel, majd ezt elvégezzük a szint többi vektorán is. Közben a már felsoroltakat ugorjuk át.

3.) Növeljük eggyel a  $h$  értékét ( $h = h + 1$ ), és ismételjük meg a 2.) pontot egészen addig amíg az összes elemet ki nem húzzuk.

Az előző példánkat tekintve a  $(3,2,1,1)$ -ből kiindulva a következő sorrend adódik.

A nulladik szinten egyedül az alapsokaság vektora áll, azaz:  $(3,2,1,1)$ . Ezért az első szinten 4 vektor fog szerepelni. Mivel az első ezek közül a  $(3,2,1,0)$ , így ebből csak három vektort tudunk előállítani a második szinten, a következő a  $(3,2,0,1)$ , de mivel a  $(3,2,0,0)$ -át a  $(3,2,1,0)$ -ból már előállítottuk, így ezt átugorva ebből már csak két újat tudunk felsorolni. A  $(3,1,1,1)$ -ből újra csak kettő komponens adódik, mert a  $(3,1,1,0)$  a  $(3,2,1,0)$ -ból valamint a  $(3,1,0,1)$  a  $(3,2,0,1)$ -ből már előállt. Ezt folytatjuk egészen addig amíg meg nem kapjuk a  $(0,0,0,0)$  vektort.

| $S_0$     | $S_1$     | $S_2$     | $S_3$     | $S_4$     | $S_5$     | $S_6$     | $S_7$     |
|-----------|-----------|-----------|-----------|-----------|-----------|-----------|-----------|
| $3,2,1,1$ | $3,2,1,0$ | $3,2,0,0$ | $3,1,1,0$ | $3,0,0,0$ | $2,0,0,0$ | $1,0,0,0$ | $0,0,0,0$ |
|           |           |           |           | $2,1,0,0$ | $1,1,0,0$ | $0,1,0,0$ |           |
|           |           |           | $2,2,0,0$ | $1,2,0,0$ | $0,2,0,0$ |           |           |
|           |           | $3,1,1,0$ | $3,0,1,0$ | $2,0,1,0$ | $1,0,1,0$ | $0,0,1,0$ |           |
|           |           |           | $2,1,1,0$ | $1,1,1,0$ | $0,1,1,0$ |           |           |
|           |           | $2,2,1,0$ | $1,2,1,0$ | $0,2,1,0$ |           |           |           |
|           | $3,2,0,1$ | $3,1,0,1$ | $3,0,0,1$ | $2,0,0,1$ | $1,0,0,1$ | $0,0,0,1$ |           |
|           |           |           | $2,1,0,1$ | $1,1,0,1$ | $0,1,0,1$ |           |           |
|           |           | $2,2,0,1$ | $1,2,0,1$ | $0,2,0,1$ |           |           |           |
|           | $3,1,1,1$ | $3,0,1,1$ | $2,0,1,1$ | $1,0,1,1$ | $0,0,1,1$ |           |           |
|           |           | $2,1,1,1$ | $1,1,1,1$ | $0,1,1,1$ |           |           |           |
|           | $2,2,1,1$ | $1,2,1,1$ | $0,2,1,1$ |           |           |           |           |

Ennek leírásához felhasználjuk a következőket:

### 3.4. DEFINÍCIÓ (átugrás):

A korábbi (3.2) egyenletben bevezett  $\delta$  függvény segítségével, mely követési relációban lévő vektorok különbségére a követést megvalósító komponens indexét adta meg, vezessünk be egy  $\gamma$  függvényt. Ez az egyes  $\mathbf{a}_i^{(h)}$  vektorokhoz hozzárendel egy számot, amit  $n$ -ből kivonva éppen azoknak az  $\mathbf{a}_j^{(h+1)}$  vektoroknak a számát adja meg, melyek követnek egy

a felsorolásban előrébb szerepelő  $\mathbf{a}_l^{(h)}$  ( $l < j$ ) vektort is. Tehát a következő szint felsorolásakor, annak érdekében, hogy minden elem egyszer szerepeljen ezeket át fog kellni ugranunk.

$$\gamma(\mathbf{a}_i^{(h)}) = \begin{cases} n & , \text{ ha } h = 0, \\ \delta(\mathbf{a}_j^{(h-1)} - \mathbf{a}_i^{(h)}) & , \text{ egyébként,} \end{cases}$$

ahol  $j = \min(l)$ , ha  $\mathbf{a}_l^{(h-1)} \kappa \mathbf{a}_i^{(h)}$ , és  $\mathbf{a}_i^{(h-1)} \in \mathbf{S}_{h-1}$ .

### 3.2. TÉTEL :

Az eddigi jelöléseket felhasználva az összes  $\mathbf{a}_j^{(h+1)}$  vektort pontosan egyszer állítja elő a következő felsorolás:

$$\mathbf{S}_{h+1} = \bigoplus_{j=1}^{m_h} \bigoplus_{i=k}^1 \sigma(\mathbf{a}_j^{(h)}, i), \text{ ahol } m_h = |\mathbf{S}_h|, k = \gamma(\mathbf{a}_j^{(h)}).$$

Ez a sorrend számunkra azért kitüntetett mivel, az átugrott elemeknek, melyek a felsorolásban már szerepelnek, sorrendben elfoglalt indexét is meg tudjuk adni. Tehát a  $\mathbf{W}^{(h+1)}$  mátrixokat a felsorolás közben előállíthatjuk. A (3.1) példában szereplő  $w$  értékek mátrixban történő felírásakor a következőket kapjuk:

$$\mathbf{W}^{(1)} = \begin{pmatrix} \frac{1}{7} & \frac{1}{7} & \frac{2}{7} & \frac{3}{7} \end{pmatrix},$$

$$\mathbf{W}^{(2)} = \begin{pmatrix} \frac{1}{6} & \frac{1}{3} & \frac{1}{2} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{6} & 0 & 0 & \frac{1}{3} & \frac{1}{2} & 0 & 0 & 0 \\ 0 & \frac{1}{6} & 0 & \frac{1}{6} & 0 & \frac{1}{6} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{6} & 0 & \frac{1}{6} & 0 & \frac{1}{3} & \frac{1}{3} \end{pmatrix},$$

$$\mathbf{W}^{(3)} = \begin{pmatrix} \frac{2}{5} & \frac{3}{5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{5} & 0 & \frac{1}{5} & \frac{3}{5} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{5} & 0 & \frac{2}{5} & \frac{2}{5} & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{5} & 0 & 0 & 0 & 0 & \frac{1}{5} & \frac{3}{5} & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{5} & 0 & 0 & 0 & 0 & \frac{2}{5} & \frac{2}{5} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{5} & 0 & 0 & \frac{1}{5} & 0 & 0 & \frac{3}{5} & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{5} & 0 & 0 & \frac{1}{5} & 0 & \frac{1}{5} & \frac{2}{5} & 0 \\ 0 & 0 & 0 & 0 & \frac{2}{5} & 0 & 0 & \frac{1}{5} & 0 & \frac{1}{5} & \frac{1}{5} \end{pmatrix}.$$

A felírást a következőképpen végezhetjük. Mivel a sorok az  $\mathbf{a}_i^{(h)}$  vektorokkal feltehetőek meg, soronként haladhatunk miközben, ha a felsoroláshoz egy új  $\mathbf{a}_j^{(h+1)}$  elemet írunk, a mátrixhoz hozzáírunk egy új csupa nullából álló oszlopot, kivéve az éppen aktuális sort, ahova a  $w(\mathbf{a}_i^{(h)}, \mathbf{a}_j^{(h+1)})$  értékét írjuk. Ha átugrunk egy elemet az indexét megkeresve a megfelelő oszlopban javítunk az előbb leírt módon számított értékkel.

A következő algoritmus megvalósítja a fent leírtakat.

Bemenetében a megelőző szintet egy kétdimenziós tömb formájában várja, a gamma értékeit egy másik tömbben várja. Az első tömb soraiban a szint vektorai, az oszlopaiban az elemek kategóriák szerinti számai legyenek. A gammák tömbjében minden sorban két elemet vár, az első  $n - \gamma(\mathbf{a}_i^{(h)})$ , ahol  $\mathbf{a}_i^{(h)}$  az első tömb megfelelő sorában lévő vektor. A gamma tömb egy sorának második eleme,  $n - \gamma(\mathbf{a}_l^{(h-1)})$ , ahol  $l = \min(j)$ , ha  $\mathbf{a}_j^{(h-1)} \kappa \mathbf{a}_i^{(h)}$ .

---

```

1:  function [ weights, cols, sch] = next_level_find( rows, psch )
2:  % NEXT_LEVEL_FIND  Creates the next level and the weight matrix for %
3:  %                  a polihypergeometric lattice %
4:  %  [weights, cols, sch ] = NEXT_LEVEL_FIND( rows, psch ) %
5:  %  rows              previous level %
6:  %  psch              step children and step sisters %
7:  %                  of the previous level's elements, %
8:  %                  level zero has [0,0] %
```

```

9:      %   cols           next level                               %
10:     %   sch           step children and step sisters           %
11:     %                               of the next level's elements %
12:
13:     cols = []; sch = []; weights = []; indexes = [];
14:     [ 1, k ] = size(rows);
15:     ssis = zeros(2,k);
16:     denum = sum(rows(1,:));
17:     % iteration through rows %
18:     for i=1:l
19:         pidx = size(cols,1) + 1;
20:         % to keep psch intact while being able to decrement %
21:         step = psch(i,:);
22:         % to find the children of rows' ith element %
23:         indexes = [indexes; pidx];
24:         % number of sisters %
25:         nsis = 0;
26:         % number of non-zero categories %
27:         nz = sum(sign(rows(i,:)));
28:         w = zeros(1, pidx + nsis - 1 - psch(i,1) + nz);
29:         weights = [weights, zeros(i-1, nz - psch(i,1)) ];
30:         % to find step sisters' indexes %
31:         if psch(i,2) == psch(i,1)
32:             ssis( :, psch(i,2) + 1 ) = [pidx + nz - psch(i,1); pidx];
33:         end
34:         % iterating through categories %
35:         for j=k:-1:1
36:             dz = sign(rows(i,j)) - sign(rows(i - step(1),j));
37:             % creating child while keeping parent intact %
38:             tmp = rows(i,:);
39:             % don't choose from which isn't exist any more %
40:             if tmp(j) ~= 0
41:                 % skip stepchildren ( which already enumerated ) %
42:                 if step(1) == 0
43:                     w(pidx+nsis) = tmp(j) /denum;
44:                     tmp(j)=tmp(j)-1;
45:                     sch = [ sch; nsis + psch(i,1), psch(i,1) ];
46:                     cols=[cols;tmp];
47:                     nsis = nsis + 1;
48:                 else
49:                     % finding stepsisters %
50:                     if step(2) > 0
51:                         % rewinding step sister index due to the lack of current category %
52:                         % there'd be a still better way to do this %

```

```
53:             if ssis(1,step(2)) > pidx+nsis-1 || ...
54:             sum(abs(rows(i,:)-cols(ssis(1,step(2)),:)))~=1
55:                 ssis(1,step(2)) = ssis(2,step(2));
56:             end
57:             w(ssis(1,step(2))) = tmp(j) /denum;
58:             ssis(1,step(2)) = ssis(1,step(2)) + 1;
59:             step(2) = step(2) - 1;
60:         % finding sisters %
61:         else
62:             w(indexes(i-step(1))+step(1)-dz)=tmp(j)/denum;
63:         end
64:         step(1) = step(1) - 1;
65:     end
66: end
67: end
68: weights = [weights;w];
69: end
70: end
```

---

## 4. fejezet

# Alkalmazási lehetőségek

Az ismertetett algoritmus alkalmazhatóságát elsődlegesen futási sebessége határozza meg. Ezt a ciklusszámtól függ, azaz hogy hány vektort kell előállítani. Ennek felső becslését adjuk meg, annak az esetnek a számával, mikor  $h < k_i$ ,  $(i = 1, 2, \dots, n)$ . Ezt azért tehetjük meg, mert azok a komponensek, melyek kisebbek  $h$ -nál elfogynak, a vektorok megfelelő komponense ezért nulla lesz és kevesebb vektor fogja követni ezeket, a  $k_i < j \leq h$  szinteken. Annak megadása érdekében, hogy hány vektort soroltunk fel a  $h$ -dik szinten a fenti feltétel mellett, fogalmazzuk át a problémát. Mivel mindegyik elemből kitudunk húzni  $h$ -t ezt tekinthetjük *ismétléses kombinációnak*, ahol  $n$ -féle dologból  $h$ -t húzunk ki.

$$|\mathbf{S}_h| \leq \binom{n+h-1}{h} =$$

Ezt a binomiális együthatókra vonatkozó nevezetesség szerint írhatjuk a következő formában:

$$\begin{aligned} &= \binom{n+h-1-1}{h-1} + \binom{n+h-1-1}{h} = \\ &= \binom{n+h-1-1}{h-1} + \binom{n+h-2-1}{h-1} + \binom{n+h-2-1}{h} = \\ &\vdots \\ &= \binom{n+h-1-1}{h-1} + \binom{n+h-2-1}{h-1} + \dots + \binom{n+h-i-1}{h-1} + \binom{n+h-i-1}{h} = \\ &= \sum_{i=1}^{n-1} \binom{n+h-i-1}{h-1} + \binom{h}{h} = \sum_{i=1}^{n-1} \binom{n+h-i-1}{h-1} + \binom{h-1}{h-1} = \\ &= \sum_{i=1}^n \binom{n+h-i-1}{h-1}. \end{aligned}$$

Elvégezve a lehetséges egyszerűsítéseket:

$$|\mathbf{S}_h| \leq \frac{\sum_{i=1}^n \prod_{j=1}^{h-1} (n - i + j)}{(h-1)!}$$

alakra hozható. Érdekes, hogy az összeg első tagja éppen  $|\mathbf{S}_{h-1}|$ .

Jól látszik, hogy ez a felsőbecslés nem függ a  $k_1, k_2, \dots, k_n$  kategóriák számosságaitól, sem  $N$  összegüktől. Ez belátható, ha arra gondolunk, hogy úgy kell választanunk  $n$ -féle elemből  $h$  darabot, hogy bármelyikből ki tudjunk választani  $h$  egyformát. Ekkor ugyanannyiféleképpen tudunk választani attól függetlenül, hogy mennyivel nagyobbak a  $k_1, k_2, \dots, k_n$ -nek  $h$ -nál. A teljes háló kiszámításához természetesen szükséges egytől  $N$ -ig minden szintet számba venni. Az is jól látszik a formulából, hogy  $n$ -től viszont az összes elem száma exponenciálisan függ.

Az eddigi megállapítások alapján a bemutatott algoritmus képes kis  $n$ -ek mellett akár igen nagy  $N$ -re is egyszerűen megadni a polihipergeomertikus eloszlás értékeit. Eközben a lehetséges vektorokra egy felsorolást valósít meg. Ezért alkalmas lehet, olyan problémák megoldására amikor particionálás és permutáció egyszerre történő alkalmazására szükséges. Ezeket egy hálóval történő kódolással teszi, így alkalmas lehet nagyméretű hálók generálására is. A felsorolás közben a sorrend indexelését is elvégzi, ezért a keresés is hatékonyan megoldható.

## 5. fejezet

# Összefoglalás

A polihipergeomertikus eloszlás értékeinek megadására két megoldás ismertettünk. Az első két meglévő algoritmus együttes alkalmazásával oldjuk meg a feladatot. Második megoldásban a lehetséges események felsorolása révén mátrixok szorzataként adjuk meg a valószínűségértékeket. A mátrixokban az események feltételes valószínűségérékei szerepelnek. A mátrixok kitöltését a felsorolásban való hatékony keresés segíti elő. Bemutattuk, hogy ezek a mátrixok az események hálóban történő rendezését írják le. Az algoritmus idő bonyolultságára a húzások számának függvényében felső becslést állapítottunk meg. Számos kérdés megválaszolása további vizsgálatot igényel.

## 6. fejezet

# Nyitott problémák

Vizsgálataink számos új problémát vetettek fel, amelyekkel a későbbiekben szeretnénk foglalkozni.

- a.) A felsorolás teljeségének bizonyítása.
- b.) Az indexelés formalizált leírása, bizonyítással.
- c.) Az elemszámra adott becslés pontosítása.
- d.) Az algoritmus változóinak újra értelmezése a művelet szám csökkentése érdekében.
- e.) Az előállított sorrend összevetése a korábban ismertetett felsorolásokkal, a kódolások közötti elhelyezkedésének vizsgálata.

## Irodalomjegyzék

- [1] A. Childs, N. Balakrishnan, (2000), Some approximations to the multivariate hypergeometric distribution with applications to hypothesis testing, *Computational Statistics & Data Analysis*, **35**, 137-154.
- [2] J. Kelleher, B. OSullivan, (2009), Generating All Partitions: A Comparison Of Two Encodings, *CoRR*, abs/0909.2331.
- [3] A. Rényi, (1966), *Valószínűesszámitás*, Tankönyvkiadó, Budapest.
- [4] G. Stoyan, (2005), *MatLab numerikus módszerek*, TypoT<sub>E</sub>X Kiadó, Budapest.
- [5] A. Williams, (2009), Loopless Generation of Multiset Permutations using a Constant Number of Variables by Prefix Shifts, *SODA '09 Proceedings of the twentieth Annual ACM-SIAM Symposium on Discrete Algorithms*, 987-996.