Eötvös Loránd Tudományegyetem
Informatikai Kar
Komputeralgebra Tanszék

Párhuzamos folyamatok ütemezése

Készítette: Szendrei Rudolf
Témavezető: Dr. Iványi Antal

Budapest, 2007. Május

Tartalomjegyzék

1. Bevezetés

Kombinatorikusok körében ismert a Mozipénztáros problémája. A pénztáros nyitáskor egy üres kasszát kap és meg kell próbálnia eladni a jegyeket. A jegyek egységesen 500 forintba kerülnek és a sorbanállók 500 vagy 1000 forintossal szeretnének fizetni. A kérdés az, hogy a sor elakad-e, azaz a pénztáros vissza tud-e adni azoknak akik 1000 forintossal fizetnének.

A dolgozatban ezt a problémát általánosítjuk úgy, hogy folyamatok fognak versengeni egy olyan kölcsönös kizárást igénylő erőforrásért, amely egyszerre r folyamatot tud kiszolgálni. A folyamatok határidős tevékenységeket szeretnének végezni az erőforrás segítségével és ezen szándékukat előre tudják. Éppen ezért megengedettnek tekintjük, ha ezt a munkát korábban végzik el, azonban későbbre nem halaszthatják.

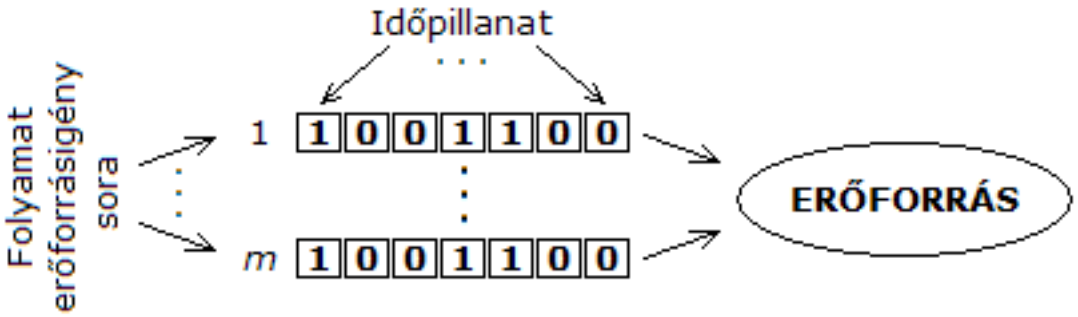
- Vizsgálatainkat $r = 1$ esetén fogjuk megtenni.
- Algoritmusokat készítünk a probléma megoldhatóságának eldöntésére.
- Bemutatunk egy a Gács Péter által javasolt algoritmusnak egy ekvivalens, de élszám-csökkentett gráffal rendelkező változatát.
- Megmutatjuk, hogy a gráfok, illetve mátrixok közötti összefüggések segítségével hogyan lehet lényegesen csökkenteni a futásidőt, ha egy időintervallum összes esetét kívánjuk vizsgálni.
- A feladat megoldhatóságához megfogalmazunk egy olyan szükséges feltételt, amelynek bebizonyítjuk a kritikus valószínűségét.

Megjegyzés: A probléma fellelhető kombinatorikusok [3, 6, 7, 18, 19] és fizikusok [1, 4, 9, 13, 15] egyik népszerű kutatási témájaként, mint különböző gráfokon való bolyongás. A problémát szokás perkolációnak [6, 7, 13, 15] nevezni (az angolok ezt a kifejezést eredetileg a hagyományos kávéfőzőben lévő gőz bolyongó útjára használták).

2. A feladat reprezentációja

Adott egy kölcsönös kizárást igénylő erőforrás és *m* folyamat, melyek bizonyos időpontokban az erőforrást használni szeretnék. Minden folyamathoz tartozik egy sor, amelylyel az igényeit reprezentáljuk. Esetünkben a reprezentáció nem eseményalapú, hanem azonos időközönként készített ”pillanatképekkel” értelmezhető.

A probléma megfeleltetéshez egy olyan bináris mátrixot használunk, amelynek egyes sorai egy-egy folyamat kérelemsorát jelentik. Az oszlopok az összes folyamat igényéről azonos időpillanatban készült ”pillanatképét” adják meg. A mátrixban az első oszlop jelenti a vizsgálat kezdetét és az utolsó a végét.



2.1. **ábra.** Folyamatok erőforrásigényének mátrixa

3. Alapfogalmak

Egy $m \times n$ méretű bináris mátrix

- *r-jó*, ha minden oszlopában legfeljebb r egyes van;
- *r-biztos*, ha tetszőleges k-ra igaz, hogy a mátrix első k oszlopában legfeljebb kr darab egyes van.
- *r-ütemezhető*, ha nulla elemek törlésével jó mátrixszá alakítható; azaz létezik a nulláknak olyan részhalmaza, hogy ha ezen nullák esetén az adott sorban lévő nullát a sor végére visszük és az utána lévő elemeket egyel balra léptetjük, akkor a mátrix *jó* lesz.

Legyen Z olyan $m \times n$ méretű mátrix, melynek elemei egymástól független valószínűségi változók, és mindegyik valószínűségi változó p valószínűséggel az 1 és $1 - p$ valószínűséggel a 0 értéket veszi fel. Az $m \geq 1$ esetben a jó mátrixok segítségével alsó, a biztos mátrixok segítségével felső korlátokat adunk annak aszimptotikus valószínűségére, hogy a Z mátrix egy konkrét realizációja 1-*ütemezhető*, és megadjuk azt a kritikus valószínűséget, amelynél kisebb p-re a Z mátrix pozitív valószínűséggel 1-*biztos*.

Néhány példa az egyes mátrixtípusokra

0	0	0	1	0	1	0	0
0	1	1	0	0	0	1	1

3.1. **ábra.** Jó mátrix

1	0	0	0	0	1	1	0
0	1	0	0	1	1	1	1

3.2. **ábra.** Biztos és ütemezhető, de nem jó mátrix

1	0	0	0	0	1	1	0
0	1	0	0	1	1	1	1

3.3. **ábra.** Biztos, de se nem jó, se nem ütemezhető mátrix

4. A feladat megfogalmazása

Legyenek m és n pozitív egészek, legyen r ($0 \leq r \leq m$) valós szám és Z olyan z_{ij} független valószínűségi változókat tartalmazó mátrix, melyek közös eloszlása
$P(z_{ij} = 1) = p$ és $P(z_{ij} = 0) = 1 - p$

Legyen A a Z mátrix egy konkrét megvalósítása.

A jó, biztos és ütemezhető mátrixok definíciója a következő.

Az A mátrixot ***r*-jónak** nevezzük, ha minden oszlopa legfeljebb r darab egyest tartalmaz. A különböző $m \times n$ méretű ***r*-jó** mátrixok számát $G_r(m, n)$-nel, a Z mátrix jóságának valószínűségét pedig $g_r(m, n, p)$-vel jelöljük.

Az A mátrixot ***r*-biztosnak** nevezzük, ha $\sum_{i=1}^{m} \sum_{j=1}^{k} a_{ij} \leq kr \qquad \forall 1 \leq k \leq n.$

A különböző $m \times n$ méretű ***r*-biztos** mátrixok számát $S_r(m, n)$-nel, a Z mátrix biztosságának valószínűségét pedig $s_r(m, n, p)$-vel jelöljük.

Ha $a_{ij} = 0$, akkor az a_{ij} elem törölhető az A mátrixból. a_{ij} törlése azt jelenti, hogy az $a_{i,j+1}, ..., a_{im}$ elemek második indexét eggyel csökkentjük és az $a_{im} = 0$ elemet hozzáírjuk az A mátrix i-edik sorának végéhez.

Az A mátrixot **Winkler *r*-ütemezhető**nek (röviden ***r*-ütemezhetőnek** vagy ***r*-kompatibilisnek**) nevezzük, ha törlésekkel átalakítható r-jó mátrixszá. A különböző $m \times n$ méretű, r-ütemezhető mátrixok számát $W_r(m, n)$-nel, a Z mátrix r-ütemezhetőségének valószínűségét $w_r(m, n, p)$-vel jelöljük. A $w_r(m, n, p)$ függvényt ***r*-ütemezhetőségi függvény**nek nevezzük. A $g_r(m, n, r)$, $w_r(m, n, r)$ és $s_r(m, n, r)$ függvényeket *sűrűségfüggvényeknek* nevezzük. A jó, biztos és ütemezhető mátrixok **aszimptotikus sűrűségét** a

$$\begin{aligned} g_r(m, p) &= \lim_{n \to \infty} g_r(m, n, p), \\ s_r(m, p) &= \lim_{n \to \infty} s_r(m, n, p), \\ w_r(m, p) &= \lim_{n \to \infty} w_r(m, n, p). \end{aligned}$$

határértékekként definiáljuk.

A következő, kritikus valószínűségeknek nevezett szuprémumok több alkalmazásban jelentős szerepet játszanak:

$$\begin{aligned} w_{crit,r}(m) &= \sup\{p \mid w_r(m, p) > 0\}, \\ g_{crit,r}(m) &= \sup\{p \mid g_r(m, p) > 0\}, \\ s_{crit,r}(m) &= \sup\{p \mid s_r(m, p) > 0\}. \end{aligned}$$

Ennek a dolgozatnak a célja az ütemezhető mátrixok különböző tulajdonságainak elemzése – elsősorban számítógépes szimuláció segítségével.

Vizsgálataink kiinduló pontja Gács Péter [7] cikke, amely szerint elég kis p értékre $w_1(2, p) > 0$. A tétel bizonyításából adódik, hogy $w_{crit,1}(2) \geq 10^{-400}$. Ebben a cikkben Gács polinomiális algoritmust javasol annak eldöntésére, hogy adott A mátrix ütemezhető-e.

Míg a két dimenziós perkolációnak gazdag irodalma van, kevés eredmény ismert több dimenzióban. A [10] cikkben szerepel a Winkler-modell kiterjesztése tetszőleges $m \geq 2$ dimenzióra.

Gács algoritmusánál gyorsabb algoritmust javaslunk a két dimenziós esetre, és ezt az algoritmust kiterjesztjük tetszőleges dimenziós mátrixok vizsgálatára.

Megjegyezzük, hogy a dolgozatban vizsgált probléma a [11] cikkből származik.

5. A feladat értelmezése

Bár a Winkler–modellt a perkoláció leírására javasolták, a problémák egy–egy lehetséges informatikai értelmezését mutatjuk be. m folyamatnak időnként ugyanarra az erőforrásra van szüksége, amelyből r egység van. Az i-edik folyamat erőforrásigényét az $a_{i1}, a_{i2}, ..., a_{im}$ sorozattal adjuk meg. Ha ennek a sorozatnak az a_{ij} eleme egyes, akkor az i-edik folyamat $[j-1, j)$ intervallumban igényli az erőforrást. Ha $a_{ij} = 0$, akkor ugyanabban az intervallumban a folyamat nem igényli az erőforrást, mert későbbre halasztható háttérmunkát végez - ez magyarázza, hogy az ütemezhetőség érdekében a nullák törölhetők.

Az $m = 1$ és $r = 1$ speciális eset az ismert jegyváltási probléma [14, 18] és szavazási probléma [5], az $m = 2$ és $r = 2$ speciális eset pedig a Winkler-féle perkolációs modell [7, 19].

A jó mátrixok törlés nélkül ütemezhetők. A nem jó mátrixok egy része törlés(ek) segítségével jóvá alakítható, azaz ütemezhető. A biztosság az ütemezhetőség szükséges feltétele. Ezért a jó mátrixok száma alsó korlát, a biztos mátrixok száma pedig felső korlát az ütemezhető mátrixok számára.

Mivel a problémát informatikai problémaként kezeljük, ezért a továbbiakban elsősorban Feller [5] informatikai (tömegkiszolgálási) terminológiáját használjuk.

6. Algoritmusok, gráfmegfeleltetések

Könnyen ellenőrizhető, hogy tetszőleges m, n és r mellett egy mátrix jó-e és hogy biztos-e.

6.1. Jó mátrixok

Csak azt kell megvizsgálnunk, hogy az oszlopok elemeinek összege kisebb-e r-nél – és csak az első olyan oszlopig kell elmennünk, amelyre a feltétel nem teljesül.

JÓ(m, n, r)

```
1 for i = 1 to n
2   s = 1
3   for j = 1 to m
4     s = s + a[i, j]
5     if s > r
6       return "a mátrix nem jó"
7 return "a mátrix jó"
```

Rögzített m és r esetén ez az algoritmus legrosszabb esetben $\Theta(n)$ ideig fut, legjobb és várható esetben pedig $\Theta(1)$ ideig.

6.2. Biztos mátrixok

Itt oszlopfolytonosan számoljuk az egyeseket és számukat oszloponként összehasonlítjuk az adott oszlop indexének r-szeresével.

BIZTOS(m, n, r)

```
1 s = 0
2 for i = 1 to n
3   for j = 1 to m
4     s = s + a[j, i]
5     if s > i * r
6       return "túl sok 1-es van az i-ik oszlopig"
7 return "a mátrix biztos"
```

Rögzített m és r esetén ennek az algoritmusnak legrosszabb esetben $\Theta(n)$, legjobb esetben pedig $\Theta(1)$ a futási ideje.

A további vizsgálatokat az $r = 1$ speciális esetben tesszük.

6.3. Ütemezhetőség

Adott A mátrix javíthatóságát nyers erővel például úgy vizsgálhatjuk, hogy a benne lévő nullák halmazának minden részhalmazát (külön–külön) töröljük, és az így kapott mátrixok jóságát vizs-gáljuk. Ebből a legrosszabb esetre nézve $\Theta(n * 2^{2n})$ ellenőrzési idő adódik. Ennél ismert sokkal jobb módszer is [7]. Rendeljük hozzá A-hoz azt a $G(A)$ irányított gráfot, melynek $(n+1)^2$ csúcsa van: a koordinátarendszer (i, j) koordinátájú pontjai, ahol $0 \leq i, j \leq n$. A gráf éleit a következőképpen definiáljuk:

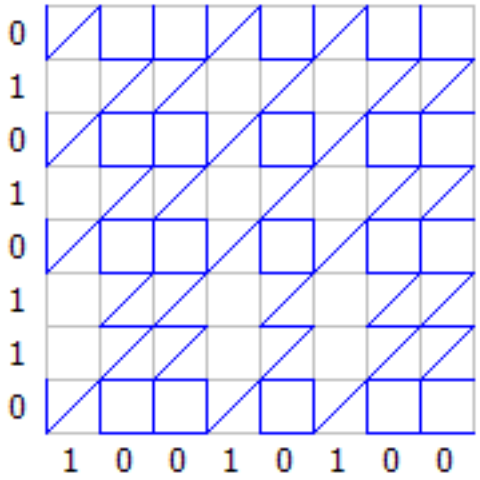
1. ha $x_i = y_j = 1$, akkor az (i, j) csúcsból nem indul ki él;

2. ha $x_i = y_j = 0$, akkor az (i, j) csúcsból két él indul ki: az egyik az $(i + 1, j)$, a másik pedig az $(i, j + 1)$ csúcsban végződik;

3. ha $x_i = 0$ és $y_j = 1$, akkor az (i, j) csúcsból két él indul ki: az egyik az $(i + 1, j)$, a másik pedig az $(i + 1, j + 1)$ csúcsban végződik;

4. ha $x_i = 1$ és $y_j = 0$, akkor az (i, j) csúcsból két él indul ki: az egyik az $(i, j + 1)$, a másik pedig az $(i + 1, j + 1)$ csúcsban végződik.

ahol $0 \leq i, j < n$.



6.1. **ábra.** Példa mátrixnak megfeleltetett gráfra
(a koordinátatengelyeken az origóból indulva olvashatóak le a mátrix sorai)

Gács Péter cikkében [7] szerepel a következő állítás:

Lemma. *Az A mátrix akkor és csak akkor ütemezhető, ha a $G(A)$ gráfban van az origóból induló és vagy egy (n, i) vagy pedig egy (i, n) pontban végződő irányított út.*

Ahhoz, hogy egy fent említett utat megtaláljunk, használhatjuk mondjuk azt az algoritmust, ahol az origót betesszük egy sorba és egy halmazba. (A halmaz segít nekünk abban, hogy egy pontot legfeljebb egyszer terjesszünk ki, a sor pedig abban, hogy a pontokat átlósan terjesszük ki a futás során.)
Ismételjük a következő lépést, amíg a sor ki nem ürül, vagy olyan ponttal nem találkozunk, amelynek van legalább egy koordinátája, amely n:

Vegyük ki a sor első elemét. Vizsgáljuk meg, hogy a csúcsból hova vezet él és az esetleges csúcsok szerepelnek-e már a halmazban. Ha még nem szerepelnek, akkor tegyük bele a halmazba és a sorba is.

Ha a sor kiürül anélkül, hogy találkoztunk volna olyan csúccsal, amelynek legalább egyik koordinátája n, akkor nem létezik út. Ha futás közben találkozunk ilyen csúccsal, akkor kész vagyunk, találtunk utat.

Korábban említettük, hogy ennél az algoritmusnál jobbat fogunk mutatni. Ezt úgy tehetjük meg, hogy a fenti bijektív megfeleltetésnél behúzott koordinátatengellyel párhuzamos éleket nem húzzuk be, amikor átlós élek indulnak ki egy csúcsból. Míg az eredeti megfeleltetésnél átlagosan 6/4 él indul ki egy csúcsból, addig jelen esetben csak 4/4. Az, hogy ezeknek az éleknek az elhagyása jogos, – azaz megtehető anélkül, hogy az ütemezhetőség eldöntését befolyásolná – bizonyítanunk kell.

Állítás: *Az eredetileg definiált gráfos megfeleltetésből elhagyott élek az út létezését nem befolyá-solják.*

Bizonyítás: Az eldöntés ekvivalenciáját négy kisebb állításként írhatjuk fel, melyeket egyenként bizonyítunk.

1. segédállítás: *Ha az új gráfban létezik út, akkor az eredetiben is, amely az origó csúcsból indul és olyan csúcsba vezet melynek a koordinátája (i, n) vagy (n, i).* Az állítás könnyen adódik, hiszen az új gráfot az eredetiből élelhagyással hoztuk létre.

2. segédállítás: *Ha az eredeti gráfban nincs az origó csúcsot (i, n) vagy (n, i) koordinátájú csúccsal összekötő út, akkor az új részgráfban sincs.* Ez adódik részgráf tulajdonságból.

3. segédállítás: *Ha az eredeti gráfban létezik az origó csúcsból induló* (i, n) *vagy* (n, i) *csúcsba vezető út, akkor az új gráfban is.*

Bizonyítás: A bizonyításhoz élhalmazok egymásba skatulyázásának közrefogási elvét fogjuk alkalmazni. Ehhez a következő indukciós eljárást hajtsuk végre:

1. Vegyünk egy tetszőleges két soros, n oszlopos bináris mátrixot és állítsuk elő Gács javaslata alapján a neki megfeleltetett gráfot
2. Vegyük a gráf azon csúcsait, amelyeknek legalább egyik koordinátája n
3. Válasszuk ki közülük azokat, melyekbe vagy átlós él vezet, vagy olyan él, amelynek forrás-csúcsából nem indul ki átlós él. Ez a csúcsra vonatkozóan nem kizáró vagyot jelent!
4. Most válasszuk ki ezeknek a csúcsoknak a szülőit és színezzük kékre az őket összekötő éleket. Ekkor minden olyan csúcsot kiválasztunk, melyeknek legalább egyik koordinátája $n-1$ és belőlük el lehet jutni olyan pontba, melynek legalább egyik koordinátája n. Ez nyilvánvaló, hiszen ha egy csúcsból merőleges élek indulnak ki, akkor rá következő csúcsuk kiválasztásra került a 3. pontban. Ha pedig átlós él is indulna ki, akkor visz jobbra és felfele is. Azokat a pontokat, amelyeknek egyik koordinátája $n-1$ és a másik pedig legfeljebb $n-1$, azok közül csak azokat hagyjuk meg, amelyeket a határpontok szülőcsúcsainak választottunk, a többit töröljük a hozzájuk vezető éllel együtt.
5. Csökkentsük n értékét egyel és ha még pozitív, akkor ugorjunk a 2. pontra.

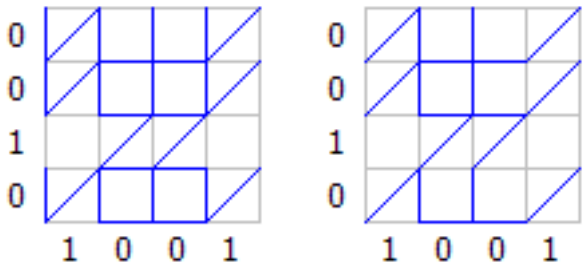
A fent definiált eljárás jól látható módon visszavágja az eredeti gráfot úgy, hogy eltávolítja azokat a pontokat, melyekből zsákutcába futnánk, továbbá ha egy pontból indul ki átlós él, akkor csak ezt hagyja meg. Ez az eljárás tehát az új gráf éleinek csak egy részhalmazát hagyja meg a Gács szerinti gráfon. Mivel az általunk kékkel színezett út egyben egy lehetséges ütemezése is a mátrixnak, ezért az új gráfban is létezik legalább egy ütemezés, ha az eredetiben is létezett.

4. segédállítás: *Ha az új gráf szerint nem ütemezhető a mátrix, akkor az eredeti szerint sem.*

Bizonyítás: Indirekt tegyük fel, hogy az új gráf alapján nem ütemezhető a mátrix, de az eredeti szerint igen. Végezzük el a 3. állítás bizonyításához adott eljárást. Ekkor a visszavágott gráfban kellene lennie egy olyan kék éleket tartalmazó élsorozatnak, amely egy helyes ütemezés. Ugyanakkor az új gráfnak ezt az élsorozatot szükségszerűen tartalmaznia kellene, így ellentmondásra jutottunk.

A négy segédállítással lefedtük az ekvivalenciára adott állítás összes lehetséges esetét. Ezek bizonyításával pedig megadtuk az eredeti állítás bizonyítását is.

A következő két ábrán jól látható ezek alapján Gács gráfmegfeleltetése és az újonnan, élelhagyással kapott gráf közötti különbség



6.2. **ábra.** Különbség megfeleltett gráfok között.

A Gács által adott lemma és az előbbiekben bizonyított állítás alapján megvalósítottuk azt a TERMÉSZETES algoritmust, amely csökkentett élszámmal dolgozik és $n = 1, 2, ..., 17$ értékekre meghatároztuk a jó $G_1(2,n)$, a biztos $S_1(2,n)$ és az ütemezhető mátrixok $W_1(2,n)$ számát, valamint – a $p = 0.5$, $p = 0.3$ és $p = 0.25$ értékek mellett – a $G_1(2,n,p)$, $S_1(2,n,p)$ és $W_1(2,n,p)$ valószínűségeket.

TERMÉSZETES(m,n,p)

```
 1  OSZLOPOS(m,n)
 2  G_1(m,n) = jo_matrix
 3  S_1(m,n) = biztos_matrix
 4  W_1(m,n) = komp_matrix
 5  G_1(m,n,p) = 0
 6  S_1(m,n,p) = 0
 7  W_1(m,n,p) = 0
 8  for i = 0 to n*m
 9     G_1(m,n,p) = G_1(m,n,p) + (p^i + (1-p)^(n*m-i)) * jo_szam[i]
10     S_1(m,n,p) = S_1(m,n,p) + (p^i + (1-p)^(n*m-i)) * biztos_szam[i]
11     W_1(m,n,p) = W_1(m,n,p) + (p^i + (1-p)^(n*m-i)) * komp_szam[i]
```

jo_matrix, *biztos_matrix* és *komp_matrix* tárolják a jó, biztos, illetve kompatibilis mátrixok értékét. *jo_szam*[*i*], *biztos_szam*[*i*] és *komp_szam*[*i*] megegyezik azon mátrixok számával, melyben pontosan *i* darab egyes található és jó, biztos, illetve kompatibilis.

A változók kiszámításához az OSZLOPOS algoritmus a következőképpen adódik:

OSZLOPOS(m, n)

```
 1 jo_matrix = 0
 2 biztos_matrix = 0
 3 komp_matrix = 0
 4 for i = 0 to n * m
 5   jo_szam[i] = 0
 6   bizt_szam[i] = 0
 7   komp_szam[i] = 0
 8 for i = 0 to n - 1
 9   oszlopok[i] = 0
10   egyesek_az_oszlopban[i] = 0
11   egyesek_idaig[i] = 0
12 STOP = 0
13 for i = 0 to m - 1
14   STOP = (STOP<<1) + 1
15 while oszlopok[0] < STOP
16   biztos_matrix = biztos_matrix + 1
17   bizt_szam[egyesek_idaig[n - 1]] = bizt_szam[egyesek_idaig[n - 1]] + 1
18   jo = igaz
19   i = 0
20   while i < n && jo
21     if 1 < egyesek_az_oszlopban[i]
22       jo = hamis
23     i = i + 1
24   if jo
25     jo_matrix = jo_matrix + 1
26     komp_matrix = komp_matrix + 1
27     jo_szam[egyesek_idaig[n - 1]] = jo_szam[egyesek_idaig[n - 1]] + 1
28     komp_szam[egyesek_idaig[n - 1]] = komp_szam[egyesek_idaig[n - 1]] + 1
29   else if Ütemezhető(oszlopok, m, n)
30     komp_matrix = komp_matrix + 1
31     komp_szam[egyesek_idaig[n - 1]] = komp_szam[egyesek_idaig[n - 1]] + 1
32   for i = n - 1 downto 0
33     BIZTOS = hamis
34     while oszlopok[i] < STOP
35       egyesek = 0
36       oszlopok[i] = oszlopok[i] + 1
37       oszlop = oszlopok[i]
38       while oszlop > 0
39         if oszlop & 1
40           egyesek = egyesek + 1
41           oszlop = oszlop >> 1
```

```
42        if egyesek_idaig[i]+egyesek − egyesek_az_oszlopban[i]<= i + 1
43          BIZTOS = igaz
44          egyesek_idaig[i]= egyesek_idaig[i]+egyesek−
45                                  egyesek_az_oszlopban[i]
46          egyesek_az_oszlopban[i]= egyesek
47          for j = i + 1 to n − 1
48            egyesek_idaig[j]= egyesek_idaig[i]
49          if i < n − 1
50            for j = i + 1 to n
51              oszlopok[j]= 0
52              egyesek_az_oszlopban[j]= 0
53          break
54      if BIZTOS
55        break
```

A változók értelmezése:

oszlopok[i]	mátrix i. oszlopa, 2-es számrendszerben ábrázolva
egyesek_az_oszlopban[i]	az i. oszlopban lévő egyesek száma
egyesek_idaig[i]	egyesek száma a mátrix első i+1 oszlopában
STOP	csupa egyest tartalmazó oszlopot reprezentáló szám
jo	igaz, ha az aktuális mátrix jó, különben hamis
BIZTOS	igaz, ha az aktuális mátrix biztos, különben hamis

Az OSZLOPOS algoritmus lényege, hogy előállítja azokat a mátrixokat, amelyek biztosak, és csak ezek közül vizsgálja meg az ÜTEMEZHETŐ algoritmussal azt, hogy melyek azok, amelyek valóban Winkler-ütemezhetők.

ÜTEMEZHETŐ(oszlopok, m, n)

```
 1 if m < 2
 2   return igaz
 3 if n < 2
 4   return igaz
 5 koord1 = (0, ..., 0)
 6 H = {}
 7 H = H ∪ koord1
 8 while H ≠ {}
 9   koord1 = min(H)
10   H = H − koord1
11   iranyok_szama = 0
12   hataron = 0
13   for i = 0 to m − 1
14     leptetes = koord[i]
15     egyes = (oszlopok[leptetes] >> i) & 1
16     if leptetes ≥ n
17       hataron = hataron + 1
18     if egyes ≠ 0
19       iranyok_szama = iranyok_szama + 1
20   if hataron > m − 2
21     return igaz
22   if iranyok_szama = 0
23     for i = 0 to m − 1
24       if koord1[i] < n
25         koord2 = koord1
26         koord2[i] = koord2[i]+1
27         H = H ∪ koord2
28   else if iranyok_szama = 1
29     koord2 = koord1
30     for i = 0 to m − 1
31       if koord2[i] + 1 < n
32         koord2[i] = koord2[i]+1
33       else
34         koord2[i] = n
35     H = H ∪ koord2
36 return hamis
```

A változók értelmezése:

koord1, koord2	m dimenziós koordináták
H	halmaz, mely m dimenziós koordinátákat tartalmaz, a halmazon a rendezés koordináták szerint lexikografikus
iranyok_szama	az adott koordinátapontból hány 1-es címkéjű él vezet ki
hataron	*koord1* pontnak hány koordinátája n
leptetes	hány bit lett feldolgozva az aktuális irányban
egyes	*koord1* pontból i. koordinátatengely mentén vezet-e ki él. 1, ha nem és 0, ha igen
min	visszaadott értéke a megadott halmazban lévő legkisebb elem

A lemma szerint a $G(A)$ gráf belső csúcsaiból átlagosan 1.5 él indul ki. Amikor azonban x_{i-1} és y_{j-1} különböző, nincs szükség a tengelyekkel párhuzamos élre, elegendő csak az (i, j) pontba vezető él. Az így kapott $H(A)$ gráf csúcsaiból másfél helyett csak 1 él indul ki. Vizsgáltuk a két dimenziós Winkler-modellnek a [10] cikkben javasolt m-dimenziós általánosítását is. Az m soros mátrixok ütemezhetőségének vizsgálatát így már a fentebb bemutatott algoritmusban megvalósítottuk. Mivel az algoritmus az n^m térfogatú térrész rácspontjait legfeljebb egyszer terjeszti ki (és akkor legfeljebb m kimenő élet vizsgál), így futási ideje legrosszabb esetben $O(m * n^m)$. Ha a vizsgált mátrix minden eleme nulla, akkor például három dimenzióban a $(0, 0, 0)$, $(n, 0, 0)$, $(0, n, 0)$ és $(0, 0, n)$ pontok által határolt hasábban lévő rácspontokat kell kiterjeszteni - ebből adódik, hogy legrosszabb esetben a futási idő $\Theta(m * n^m)$.

7. Gráfbejárások hatásai

A korábbiakban mátrixok és gráfok közötti kölcsönös megfeleltetést mutattunk be. Ezután bebizonyítottuk, hogy ahhoz, hogy a megfeleltetett gráfban találjunk utat az origó és valamely határpont között – amelynek legalább $m - 1$ koordinátája n –, ahhoz egy redukált gráf is elegendő. Nem tértünk ki még arra a fontos tényre, hogy mindez a megfeleltetés csupán az útkeresés során használható élek halmazát befolyásolja és ettől függetlenül tetszőleges útkereső algoritmust választhatunk az ütemezhetőség eldöntésére. Már most leszögezzük, hogy csak olyan útkereső algoritmusokkal fogunk foglalkozni, amelyek irányított élekkel dolgoznak. Ekkor a feladatunk speciális volta miatt két dolgot is kiküszöbölhetünk. Egyrészt nincsenek körök, hurkok a gráfban és ezért biztosan terminálni fog az algoritmus, másrészt a futási idő a mátrix méretének tekintetében felülről korlátos lesz. Mindezek alapján megmutatjuk, hogy az egyes bejárási módszerek miként befolyásolják az útkeresés idejét és a felhasznált erőforrások mennyiségét.

7.1. Gráfbejárási típusok

Alapvetően két kategóriába sorolhatjuk a bejárásokat. Az elsőbe esnek azok, amelyek függetle-nül az élek helyétől, adott csúcstól vett távolságától, stb. működnek, azaz statikus viselkedésűek. A második kategóriát reprezentálják azok a bejárást megvalósító algoritmusok, amelyek segéd-információk segítségével megpróbálnak online módon döntést hozni egy következő lépést illetően. Ezeket a továbbiakban heurisztikáknak fogjuk nevezni.

7.2. Példa statikus viselkedésű bejárásra

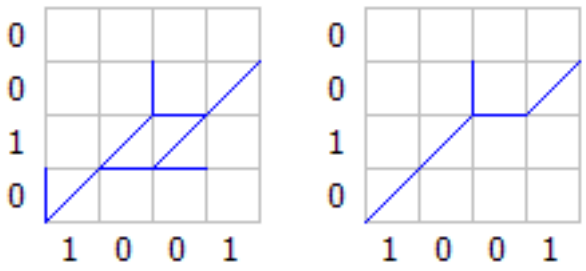
Legyen adott egy kétsoros, bináris, ütemezhetőségi mátrixhoz rendelt gráf. Ennek a bejárását az origótól kezdjük és olyan csúcsba akarunk eljutni, melynek koordinátája (i, n) vagy (n, i). Legyen a bejárás alapötlete az, hogy az origóban megvizsgáljuk, hogy mely szomszédba vezet út és azokat a csúcsokat betesszük egy halmazba. Ezután adott i esetén sorban kivesszük a halmazból azokat a csúcsokat, amelyeknek a két koordinátájuk közül a maximális i és megvizsgáljuk, hogy mely szomszédjaiba léphetünk tovább, majd azokat betesszük a halmazba. Esetünkben i az $[1..n]$ intervallumot futja be. Ennek eredményeképpen a bejárásnál minden lehetséges csúcsot elérünk amelyet lehet, mielőtt (n, i) vagy (i, n) csúcshoz érnénk. Értelemszerűen a futási idő az origóból elérhető csúcsok számával egyenlő. Ilyen esetben az új algoritmus a kevesebb él használata miatt kevesebb lépést kell végezzen. Ezt a módszert ha tetszik, nevezhetjük egy speciális szélességi bejárásnak.

7.3. Példa heurisztikus bejárásra

Előre kell bocsátanom, hogy a heurisztikák között egyértelmű győztest nem lehet hirdetni. Továbbá megjegyzem, hogy a heurisztikák esetleges további számításai miatt amit nyerhetünk futási időben vizsgálatok számában, azt könnyen el is veszthetjük az egy csúcs megvizsgálására fordított többletköltségeken.

Legyen ismételten adott, egy kétsoros, bináris, ütemezhetőségi mátrixhoz rendelt gráf. Az előzőekben említett módon most is vizsgáljuk meg, hogy az origó mely szomszédjaiba tudunk eljutni. Ezeket a csúcsokat tegyük bele egy halmazba. Értelmezzünk egy olyan rendezést a halmazon, amely szerint az a kisebb elem, amely koordinátáit lexikografikusan vizsgálva nagyobb. Két megállási feltételt kell rögzíteni. Az egyik, hogy ha kiürült a halmaz, akkor nem találtunk utat az origó és egy (n,i) vagy (i,n) koordinátájú csúcs között. A másik lehetőség, hogy pont ilyen koordinátájú csúcsot veszünk ki éppen a halmazból. Ekkor találtunk utat, vagyis az eredeti feladat értelmében ütemezhetőnek ítélhetjük a mátrixot.

Az algoritmus futása közbeni egyes lépések a következő módon adottak: vegyük ki a legkisebb elemet a halmazból és próbáljuk meg kiterjeszteni. Azokat a szomszédos csúcsokat, amelyekbe így el tudunk jutni, betesszük a halmazba. Ezt az algoritmust egy mélységi keresésnek foghatjuk fel, amely a mohó algoritmusok kategóriájába tartozik. Kézenfekvő módon a korábbi statikus algoritmushoz képest ez legfeljebb ugyanannyi csúcsot és élet vizsgál meg, de általában véve töredékét, hiszen már az első út megtalálásával megelégszik. Élszám kiterjesztési lépések számát mérve noha azt gondolnánk, hogy kevesebb él mellett rövidül a futási idő, az tapasztalható, hogy az újonnan megfeleltett gráfon az eldöntő algoritmusnak több lépésre van szüksége átlagosan, mintha az eredeti megfeleltetett gráfon történt volna a vizsgálat.



7.1. **ábra.** Heurisztikusan bejárt eredeti és redukált gráf

7.4. Következtetések

Az utóbb tárgyalt heurisztikánál jelenleg nem találtunk futási időt tekintve gyorsabbat, ami természetesen nem jelenti azt, hogy nem létezik.

Az, hogy ezen heurisztikus algoritmus futási ideje nagyobb a redukált gráfon, mint az eredetin, csupán azt bizonyítja, hogy több mérce szerint is mérhetünk optimalitást. Ha több memória áll rendelkezésünkre, inkább a régi algoritmust érdemes alkalmazni, ellenkező esetben a gép processzorának többletszámításai kompenzálhatják a memóriahiányt adott esetben. Tulajdonképpen ez jelenti a feladat problémájának bermudaháromszögét: erőforrások, eldöntési módszer, bejárási heurisztika. A dolgozat határain túlmutató elemzéseket kívánna az, hogy olyan algoritmus legyen adható, amely ezen három paraméter mellett keressen hatékony megoldást. Említésképpen megjegyzem, hogy az ilyen algoritmusokat legtöbbször alkalmazkodónak szokták nevezni (adaptive) és leginkább a képfeldolgozás területén elterjedtek a szűrési eljárásokban.

A fentebb tárgyalt heurisztikus útkereső algoritmus futási idejeit élszámkiterjesztésben mérve az alábbi táblázat adja meg a két gráfmegfeleltetés mellett. A vizsgálat két soros bináris mátrixok adott n érték melletti összes előfordulását véve lettek kiszámítva.

n	redukált gráf	eredeti gráf	arány
1	0	0	1,000
2	1	1	1,000
3	24	18	1,333
4	236	154	1,532
5	1 676	995	1,684
6	10 161	5 555	1,829
7	56 066	28 409	1,973
8	290 377	137 193	2,116
9	1 436 086	636 776	2,255
10	6 857 070	2 872 374	2,387
11	31 852 936	12 685 224	2,511
12	144 746 322	55 128 796	2,625

7.2. **ábra.** Heurisztikus útkereső algoritmus élszámkiterjesztései eredeti és redukált gráfon

8. Matematikai eredmények

Ebben a részben elsősorban azt vizsgáljuk - különböző módszerekkel - hogyan függ a biztos mátrixok aszimptotikus sűrűsége az egyesek előfordulásának p valószínűségétől és a sorok m számától.
A vizsgált $g_r(m, n, p)$, $w_r(m, n, p)$ és $s_r(m, n, p)$ függvények tulajdonságai:

1. $n \in \mathbb{N}$, $r \in \mathbb{R}$ és $r \in [0, m]$, $p \in \mathbb{R}$ és $p \in [0, 1]$;
2. n szerint monoton csökkenők
3. p szerint monoton csökkenők
4. m szerint monoton csökkenők
5. r szerint monoton növekvők

A továbbiakban mindenütt feltesszük, hogy $r = 1$, azaz a jó mátrixok oszlopaiban legfeljebb egy egyes, a biztos mátrixok k hosszúságú prefixeiben pedig legfeljebb k egyes lehet. Mivel r értéke mindenütt ugyanaz, a továbbiakban elhagyjuk az r indexet.

8.1. Előzetes eredmények

A későbbiekben felhasználjuk az alábbi állításokat.

Jelöljük C_n-nel ($n \in \mathbb{N}$) azon $a_1, a_2, ..., a_{2n}$ bináris sorozatok számát, amelyekben n darab egyes és n darab nulla van úgy, hogy minden $a_1, a_2, ..., a_k (1 \leq k \leq 2n)$ kezdősorozatban legfeljebb annyi egyes van, mint nulla.

8.1.1. lemma. *Ha $n \geq 0$, akkor*

$$C_n = \frac{1}{n+1}\binom{2n}{n}$$

Megjegyezzük, hogy C_n az n-edik Catalan-szám, melynek explicit alakja számos tankönyvben és cikkben [2, 11, 12, 14, 16, 18] megtalálható.

8.1.2. lemma. *Ha* $0 \leq x \leq 1$*, akkor*

$$f(x) = x \sum_{k=0}^{\infty} \frac{1}{k+1} \binom{2k}{k} (x(1-x))^k = \begin{cases} \dfrac{x}{1-x}, & \text{ha } 0 \leq x \leq \dfrac{1}{2} \\ 1, & \text{ha } \dfrac{1}{2} \leq x \leq 1 \end{cases}$$

Bizonyítás.

Ha $m \geq 2$, akkor a csupa nullát tartalmazó oszlopot fehérnek nevezzük, a pontosan egy egyeset tartalmazót szürkének az ettől többel rendelkezőket pedig feketének hívjuk.

Ha $m \geq 2$, akkor az A mátrix minden oszlopa $q^m + mq$ valószínűséggel lehet fehér vagy szürke, ezért $g(m, n, p) = ((q^m + mq))^n$.

Ha $p > 0$, akkor

$$g(m, p) = \lim_{n \to \infty} (q^m + mq)^n = 0, \qquad (8.1.2)$$

tehát az oszlopok számának növekedtével a jó mátrixok sűrűsége tart a nullához.

Ha az $m = 2$ esetben egy jó mátrixból töröljük a fehér oszlopokat, akkor csak szürke oszlopok maradnak, azaz a mátrix két sora egymásnak a komplementere.

Vizsgálataink szempontjából fontos szerepet játszik a következő egyszerű állítás.

8.1.3. lemma. *Ha* $m \geq 2$*, akkor a jó mátrixok ütemezhetők, az ütemezhető mátrixok pedig biztosak.*

Bizonyítás. Ha az A mátrix minden oszlopában legfeljebb egy egyes van, akkor a mátrix első k oszlopában összesen legfeljebb k egyes van.

Ha van olyan $k (1 \leq k \leq n)$, amelyre A mátrix első k oszlopában több egyes van, mint k, akkor a skatulyaelv szerint az első k oszlop között van olyan, amelyikben legalább két egyes van. Ha az A mátrixból törlünk egy nullát, ezzel az első k oszlopban lévő egyesek száma nem csökken - tehát A nem ütemezhető.

Az állításnak hasznos következménye az alábbi.

8.1.4. következmény. *Ha* $m \geq 2$*, akkor*

$$g(m, n, p) \leq w(m, n, p) \leq s(m, n, p)$$
$$g(m, p) \leq w(m, p) \leq s(m, p)$$
$$g_{crit}(m) \leq w_{crit}(m) \leq s_{crit}(mp)$$

Az $m \times n$ méretű biztos mátrixok elemzését az $m \geq 4$ esetben az $m = 3$ esethez hasonló módon végezhetjük el.

A bolyongó pont a legalább $b \geq 2$ egyest tartalmazó oszlop esetén $(b-2)$-t ugrik balra, két egyest tartalmazó oszlop esetén egyet lép balra, egy egyest tartalmazó oszlop esetén helyben marad és az m nullát tartalmazó oszlop esetén egyet lép jobbra.

A $(b-2)$-vel balra ugrás valószínűsége $\binom{m}{b} p^{b-2} q^{n-b+2}$, a balra lépésé $\binom{m}{2} p^{m-2} q^2$, a helyben maradásé $\binom{m}{1} pq^{m-1}$ és a jobbra lépésé $\binom{m}{0} q^m$, ezért a következő egyenleteket írhatjuk fel.

Az $m = 2$ és $m = 3$-ra kapott korábbi eredményeket is figyelembe véve ezzel a következő eredményt kaptuk.

8.1.5. tétel. *Ha* $m \geq 2$ *és* $0 \leq p \leq 1$, *akkor*

$$s(m, p) = 1 - \sum_{i=2}^{m} \binom{m}{i} \left(\frac{p}{1-p} \right)^i (i-1) \qquad (8.1.6)$$

és

$$s_{crit} = \frac{1}{m} \qquad (8.1.7)$$

9. Szimulációs eredmények

9.1. Kétsoros mátrixok

Az egyszerűség kedvéért az $s(2,n,p)$ függvény helyett az $u(2,n,p) = 1 - s(2,n,p)$ függvényt elemezzük. Először megadunk egy zárt képletet $u(2,n,p)$ meghatározására.

9.1.1. lemma. *Ha $n \geq 1$, akkor*

$$u(2,n,p) = \sum_{i=1}^{n} \sum_{j=0}^{[(i-1)/2]} 2^{i-1-2j} C_j \binom{i-1}{2j} 4^{n-i} \qquad (9.1.1)$$

Az $u(2,n,p)$-re kapott (9.1.1) képlet nehezen kezelhetőnek látszik. Ezért bemutatunk egy kombinatorikus és két bolyongásos módszert $s(2,p)$ explicit alakjának levezetésére.

9.1.2. lemma. *Ha $0 \leq p \leq 1$, akkor*

$$u(2,p) = \begin{cases} \dfrac{p^2}{q^2}, & \text{ha } 0 \leq p \leq \dfrac{1}{2} \\ 1, & \text{ha } \dfrac{1}{2} \leq p \leq 1 \end{cases} \qquad (9.1.2)$$

Mátrixaink vizsgálatának hasznos módszere, ha minden mátrixhoz hozzárendelünk egy – az x-tengelyen való – bolyongást.

A bolyongó pont a k-adik időpontban a $P_k(b_k, 0)$ pontban van, ahol b_k a mátrix első k oszlopában előfordult nullák és egyesek számának különbségét jellemzi.

Ha a bolyongás az origóból indul, akkor

$$b_k = \begin{cases} -1, & \text{ha } \exists k \text{ úgy, hogy } k < \sum_{i=1}^{k}(a_{i1} + a_{i2}) \\ k - \sum_{i=1}^{k}(a_{i1} + a_{i2}), & \text{egyébként} \end{cases} \qquad (9.1.3)$$

Bemutatunk egy olyan módszert, amelyet az $r = 1$ esetben tetszőleges $m \geq 2$ értékre alkalmazni tudunk.

Bizonyítás. Jelöljük x_k-val ($k = -1, 0, 1, 2, ...$) annak a valószínűségét, hogy a k pontból induló bolyongó pont elnyelődik az $x = -1$ helyen. A két egyest tartalmazó, p^2 valószínűséggel előforduló oszlopnak feleltessünk meg balra lépést, a $2pq$ valószínűséggel előforduló vegyes oszlopokhoz tartozzon helyben maradás és a két nullát tartalmazó, q^2 valószínűséggel előforduló oszlopoknak feleljen meg jobbra lépés.

Ekkor a következő egyenleteket írhatjuk fel.

$$
\begin{aligned}
x_0 &= q^2 x_1 + 2qpx_0 + p^2 \\
x_1 &= q^2 x_2 + 2qpx_1 + p^2 x_0 \\
x_2 &= q^2 x_3 + 2qpx_2 + p^2 x_1 \\
x_3 &= q^2 x_4 + 2qpx_3 + p^2 x_2
\end{aligned}
\qquad (9.1.4)
$$

$$\vdots$$

Legyen

$$G(z) = \sum_{i=1}^{\infty} x_i \, z^i \qquad (9.1.5)$$

az $x_0, x_1, x_2, \ldots$ sorozat generátorfüggvénye. Az (7.4) egyenletrendszer x_i-vel kezdődő egyenleteit rendre $z - i$-vel beszorozva és az egyenleteket összeadva a

$$G(z) = q\frac{G(z) - x_0}{z} \; + \; 2pqG(z) \; + \; p^2 \, (1 \; + \; z \, G(z)). \qquad (9.1.6)$$

Innen $G(z)$ kifejezhető

$$G(z) = \frac{P(z)}{Q(z)} \qquad (9.1.7)$$

alakban, ahol

$$P(z) = q^2 - p^2 z \qquad (9.1.8)$$

és

$$Q(z) = p^2 z^2 \; + \; 2pq \; + \; p^3 \; - \; z \qquad (9.1.9)$$

A $Q(z)$ polinom legfeljebb egy abszolút értékű zérushelyein a Cauchy-Hadamard-tétel [17] szerint a $P(z)$ polinomnak a nulla értéket kell felvennie. A $Q(z) = 0$ egyenletet

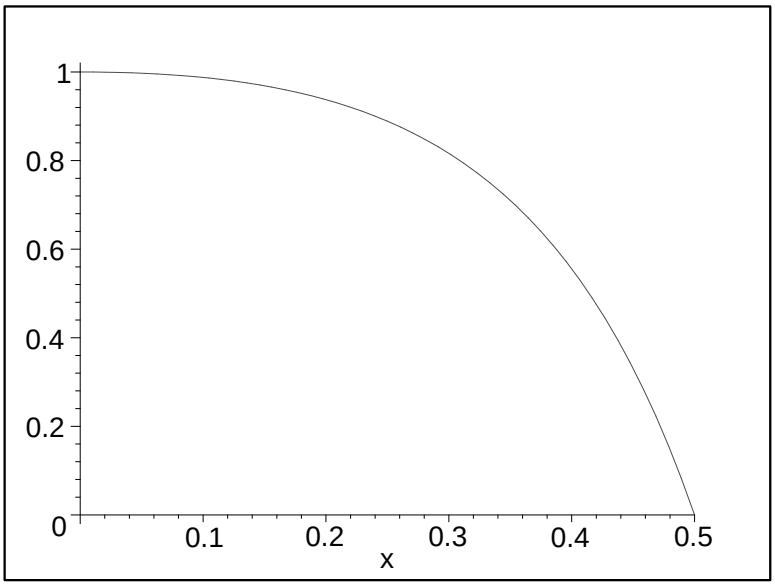
$$(pz \; + \; q)^2 = 1 \qquad (9.1.10)$$

alakban felírva közvetlenül adódik, hogy $z = 1$ gyöke a $Q(z)$ polinomnak. A $P(1) = 0$ egyenletből az

$$x_0 = \frac{p^2}{q^2} \qquad (9.1.11)$$

megoldást kapjuk.

A 9.1.1 ábra mutatja a $[0, 1]$ intervallumban értelmezett $s(2, p)$ függvény görbéjének a $[0, \frac{1}{2}]$ intervallumhoz tartozó részét.



9.1.1. **ábra.** Az $s(2, p)$ ütemezhetőségi függvény görbéje.

A $g(2, p)$ és $s(2, p)$ függvények alapján az $m = 2$ értékhez tartozó kritikus valószínűségekre fennáll

$$0 = g_{crit}(2) \ \leq \ w_{crit}(2) \ \leq \ s_{crit}(2) = \frac{1}{2} \qquad (9.1.12)$$

n	$G(2,n)$	$\frac{G(2,n)}{T(2,n)}$	$W(2,n)$	$\frac{W(2,n)}{T(2,n)}$	$S(2,n)$	$\frac{S(2,n)}{T(2,n)}$	$\frac{W(2,n)}{S(2,n)}$
1	3	0.750	3	0.750	3	0.750	1.000
2	9	0.562	10	0.625	10	0.625	1.000
3	27	0.452	35	0.547	35	0.547	1.000
4	81	0.316	124	0.484	126	0.492	0.984
5	243	0.237	444	0.434	462	0.451	0.961
6	729	0.178	1 592	0.389	1 716	0.419	0.927
7	2 187	0.133	5 731	0.350	6 435	0.393	0.890
8	6 561	0.100	20 671	0.315	24 310	0.371	0.850
9	19 683	0.075	74 722	0.285	92 378	0.352	0.808
10	59 049	0.056	270 521	0.258	352 716	0.336	0.767
11	177 147	0.042	980 751	0.234	1 352 078	0.322	0.725
12	531 441	0.032	3 559 538	0.212	5 200 300	0.310	0.684
13	1 594 323	0.022	12 931 155	0.193	20 058 300	0.299	0.646
14	4 782 969	0.018	47 013 033	0.175	77 558 760	0.289	0.606
15	14 348 907	0.013	171 036 244	0.159	300 540 195	0.280	0.568

9.1.2. **ábra.** A $p = 1/2$ valószínűséghez tartozó kerekített adatok.

Emlékeztetünk Gács eredményére, amely szerint $w_{crit}(2) \geq 10^{-400}$.

A 9.1.2. táblázatban megadjuk a jó, ütemezhető és biztos mátrixok számát és részarányát – ami megfelel a $p = 0.5$ értéknek. A táblázatban jellemzett mátrixok oszlopainak száma $1, 2, \ldots, 15$. Jelöljük a $m \times n$ méretű bináris mátrixok számát $T(m, n)$-nel.

Ekkor $T(m, n) = 2^{mn}$.

Az eddigi eredmények alapján mind a $G(m, n)/T(m, n)$, mind pedig a $W(m, n)/T(m, n)$ és $S(m, n)/T(m, n)$ értékeknek nullához kell tartani n növekedtével.

Nyitott kérdés a $W(2, n)/S(2, n)$ hányados viselkedése.

A 9.1.3. táblázat $s(2, n, 0.4)$ oszlopában a számoknak az $5/9$ határértékhez kell tartaniuk – ami még messze van.

A 9.1.4. táblázatban az $s(2, n, 0.35)$ oszlop számainak a $120/169 \sim 0.7101$ határértékhez kell tartaniuk.

n	$T(2,n)$	$g(2,n,0.4)$	$w(2,n,0.4)$	$s(2,n,0.4)$	$\frac{w(2,n,0.4)}{s(2,n,0.4)}$
1	4	0.8400	0.8400	0.8400	1.0000
2	16	0.7056	0.7632	0.7632	1.0000
3	64	0.5927	0.7171	0.7171	1.0000
4	256	0.4979	0.6795	0.6862	0.9902
5	1 024	0.4182	0.6487	0.6639	0.9771
6	4 096	0.3513	0.6206	0.6470	0.9592
7	16 384	0.2951	0.5957	0.6339	0.9397
8	65 536	0.2479	0.5731	0.6234	0.9193
9	262 144	0.2082	0.5524	0.6149	0.8984
10	1 048 576	0.1749	0.5332	0.6078	0.8773
11	4 194 304	0.1469	0.5155	0.6019	0.8565
12	16 777 216	0.1234	0.4988	0.5967	0.8359
13	67 108 864	0.1037	0.4832	0.5924	0.8157
14	268 435 456	0.0871	0.4685	0.5886	0.7960
15	1 073 741 824	0.0731	0.4545	0.5854	0.7764
16	4 294 967 296	0.0644	0.4412	0.5825	0.7574
17	17 179 869 184	0.0516	0.4286	0.5800	0.7390

9.1.3. **ábra.** A $p = 0.4$ valószínűséghez tartozó kerekített adatok.

n	$T(2,n)$	$g(2,n,0.35)$	$w(2,n,0.35)$	$s(2,n,0.35)$	$\frac{w(2,n,0.35)}{s(2,n,0.35)}$
1	4	0.8775	0.8775	0.8775	1.0000
2	16	0.7700	0.8218	0.8218	1.0000
3	64	0.6757	0.7901	0.7901	1.0000
4	256	0.5929	0.7645	0.7699	0.9930
5	1 024	0.5203	0.7441	0.7561	0.9841
6	4 096	0.4565	0.7255	0.7462	0.9723
7	16 384	0.4006	0.7094	0.7389	0.9601
8	65 536	0.3515	0.6949	0.7334	0.9475
9	262 144	0.3085	0.6817	0.7291	0.9350
10	1 048 576	0.2707	0.6696	0.7258	0.9226
11	4 194 304	0.2375	0.6585	0.7231	0.9107
12	16 777 216	0.2084	0.6481	0.7210	0.8989
13	67 108 864	0.1839	0.6383	0.7192	0.8875
14	268 435 456	0.1605	0.6291	0.7178	0.8764
15	1 073 741 824	0.1401	0.6204	0.7166	0.8658
16	4 294 967 296	0.1236	0.6122	0.7156	0.8555

9.1.4. **ábra.** A $p = 0.35$ valószínűséghez tartozó kerekített adatok.

n	$T(3,n)$	$G(3,n)$	$W(3,n)$	$w(3,n,0.5)$	$S(3,n)$	$s(3,n,0.5)$	$\frac{w(3,n,0.5)}{s(3,n,0.5)}$
1	8	4	4	0.5000	4	0.5000	1.0000
2	64	16	19	0.2969	19	0.2969	1.0000
3	512	64	98	0.1914	98	0.1914	1.0000
4	4 096	256	525	0.1282	531	0.1296	0.9892
5	32 768	1 024	2 884	0.0880	2 974	0.0907	0.9702
6	262 144	4 096	16 043	0.0612	17 060	0.0651	0.9401
7	2 097 152	16 384	90 091	0.0429	99 658	0.0475	0.9032
8	16 777 216	65 536	507 520	0.0303	590 563	0.0352	0.8594

9.1.5. **ábra.** Az $m = 3$ és $p = 0.5$ paraméterekhez tartozó kerekített adatok.

n	$T(3,m)$	$g(3,n,0.3)$	$w(3,n,0.3)$	$s(3,n,0.3)$	$\frac{w(3,n,0.3)}{s(3,n,0.3)}$
1	8	0.7840	0.7840	0.7840	1.0000
2	64	0.6147	0.6795	0.6795	1.0000
3	512	0.4819	0.6153	0.6153	1.0000
4	4 096	0.3778	0.5682	0.5710	0.9951
5	32 768	0.2962	0.5313	0.5380	0.9875
6	262 144	0.2322	0.5004	0.5125	0.9764
7	2 097 152	0.1821	0.4739	0.4919	0.9634

9.1.6. **ábra.** Az $m = 3$ és $p = 0.3$ értékekhez tartozó kerekített adatok.

n	$T(3,m)$	$g(3,n,0.25)$	$w(3,n,0.25)$	$s(3,n,0.25)$	$\frac{w(3,n,0.25)}{s(3,n,0.25)}$
1	8	0.8437	0.8437	0.8437	1.0000
2	64	0.7119	0.7712	0.7712	1.0000
3	512	0.6007	0.7286	0.7286	1.0000
4	4 096	0.5068	0.6981	0.7004	0.9967
5	32 768	0.4276	0.6748	0.6804	0.9917

9.1.7. **ábra.** Az $m = 3$ és $p = 0.25$ értékekhez tartozó kerekített adatok.

9.2. Háromsoros mátrixok

Az $m = 3$ esetben $3 : 0, 2 : 1, 1 : 2$ és $0 : 3$ lehet a nullák és egyesek aránya. A vizsgált mátrixhoz olyan bolyongást rendelünk, amelyikben a csupa egyes oszlop p^3 valószínűségével ugrunk balra kettővel, a két egyest tartalmazó oszlop $3p^2q$ valószínűségével lépünk balra, az egy egyest tartalmazó oszlopok $3p^2$ valószínűségével maradunk helyben és a három nullát tartalmazó oszlop q^3 valószínűségével lépünk jobbra.

9.3. Általános eset

Ha $r = 1$ és $m \geq 2$, akkor a biztos mátrixok vizsgálata olyan aszimmetrikus bolyongásra vezet, amely az origóból indul, az $x = -1$ pontban van egy nyelő, és a mozgás a mátrix oszlopainak felel meg: ha az oszlopban egy egyes van – ennek valószínűsége $\binom{m}{1}pq^{m-1}$ – akkor a pont helyben marad; ha az oszlopban $k > 1$ egyes van – ennek valószínűsége $\binom{m}{k}p^kq^{m-q}$ – akkor $k - 1$ helyet ugrunk jobbra; ha pedig az oszlopban csupa nulla van – ennek valószínűsége q^m – akkor egyet lépünk balra.

10. Az összes mátrix hatékony vizsgálatának módszerei

Mindezidáig olyan algoritmusokról esett szó, amelyek külön-külön alkalmasak arra, hogy eldöntsék egy mátrixról annak tulajdonságait. Vizsgálataink során adott m, n és r paraméter mellett szerettük volna meghatározni az egyes kategóriákba eső mátrixok számát. Most összefoglaljuk azokat az alapismereteket, amelyek a *jóságot, biztosságot* és *ütemezhetőséget* ellenőrző algoritmusok "összeépítésére" vonatkoznak:

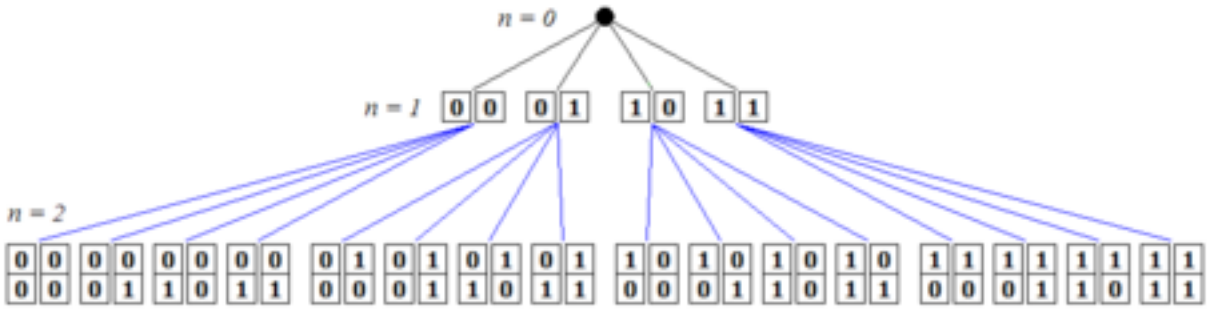
- Szeretnénk kiszámítani az egyes kategóriákba eső mátrixok számát
- A jó mátrix ütemezhető
- Az ütemezhető mátrix biztos
- Ha a mátrix valamely prefixe nem biztos, akkor a mátrix nem biztos
- Ha a mátrix valamely prefixe nem ütemezhető, akkor a mátrix sem ütemezhető

A fenti ismeretekből származó megállapítások:

- Elegendő előállítani a biztos mátrixokat és ellenőríznünk, hogy melyekre teljesül a jóság és ütemezhetőség definíciója.
- A jó mátrixoknál nem kell vizsgálnunk ütemezhetőséget
- A prefix tulajdonságok kihasználására az összes mátrix előállítását oszloponként kell végeznünk

10.1. Mátrixok fába rendezése

Ha a mátrixot a fentiek értelmében oszloponként $(1..n)$ állítjuk elő, akkor ehhez a következő faszerkezetet rajzolhatjuk fel $m = 2$ esetben:



10.1. **ábra.** Az összes mátrix oszlopos előállítását reprezentáló fa
(A fában a mátrixok elfektetve láthatóak)

Vezessük be a következő fogalmat: ha egy mátrix első k oszlopára teljesül a biztosság, akkor a mátrixot *k-prefix-biztosnak* nevezzük. Ennek analógiájára építjük a *k-prefix-ütemezhetőség* definícióját is.

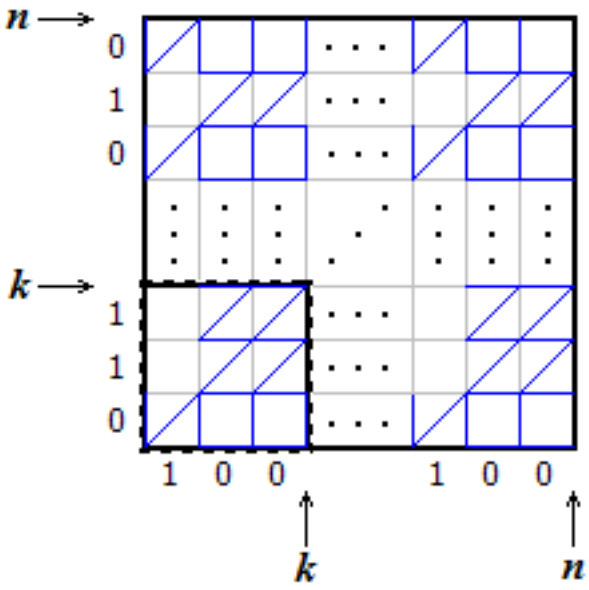
A prefix tulajdonságok a fán úgy értelmezhetőek, hogy egy adott k szinten lévő csúcs összes gyereke rendelkezik a szülőjének *k-prefix* tulajdonságával.

Ekkor a következő eredmények adódnak:

- Amennyiben egy csúcs nem *k-prefix-biztos*, úgy a belőle kiinduló ágakat levághatjuk
- Ha egy csúcs nem rendelkezik a *k-prefix-ütemezhetőséggel*, akkor a gyerekeinél nem kell megvizsgálnunk az ütemezhetőséget
- Az előző két megállapítás kihasználásához a fa csúcsait inorder bejárás mellett kell vizsgálni (elöször a csúcs vizsgálata, majd csak ezután a gyerekeké)

10.2. Mátrixok prefixei a gráfos megfeleltetéskor

Egy m soros, n oszlopos mátrix m dimenziós koordinátarendszerben történő gráfmegfeleltetése esetén a *k-prefix* tulajdonság jelentése megegyezik azon gráf adott tulajdonságával, amelyet úgy kapunk, hogy az eredeti gráfnak a koordinátarendszer $(0, ..., 0)$, $(k, ..., k)$ koordinátájú pontjai által meghatározott m-dimenziós kockával vesszük a metszetét.



10.2. **ábra.** Példa *k-prefixségre* $m = 2$ esetén

10.3. Mátrixok korrelációja

A mátrixok előállításához javasolt inorder fabejárás során az egymás után előállított mátrixok közötti korrelációt nem hanyagolhatjuk el, ha hatékony algoritmust akarunk készíteni. Kihasználhatjuk ugyanis azt, hogy korábbi számításaink eredményeit figyelembevéve aszimptotikusan csökkenthetjük akár az elvégzendő kiértékelések számát. Ezen lehetőségeket a biztosságot, illetve ütemezhetőséget ellenőrző algoritmusoknál külön tárgyaljuk a következőkben.

10.4. Korreláció felhasználása biztosság ellenőrzéshez

Tartsuk nyilván minden egyes szint esetében, hogy egy i-ik szinten lévő csúcsnál, – amelyen keresztül elérhető a vizsgált k-ik szintén lévő csúcs, – hány 1-es fordult elő. Más szavakkal, hány 1-est tartalmaz a mátrix első i oszlopa. Ebből n szint esetén pontosan n adatot kapunk. Ekkor a következő egyszerű feltétellel ellenőrízhető az adott – k-ik szintű – csúcsnál a *k-prefix-biztosság*

- Legyen $e[1..n]$ egy tömb, ahol $e[k]$ értéke a mátrix első k oszlopában lévő egyesek száma
- Ha $k = 1$, akkor legyen $e[k] = 0$, egyébként pedig $e[k] = e[k-1]$
- Legyen l a fában az adott csúcsnál a mátrix k-ik oszlopában lévő egyesek száma.
- Legyen $e[k] = e[k] + l$
- A mátrix akkor és csak akkor *k-prefix-biztos*, ha $e[k] <= k$.

Korábbi megállapítás alapján, ha ekkor úgy találjuk, hogy nem teljesül a *k-prefix-biztos* tulajdonság, akkor nem kell elvégeznünk a vizsgálatot az azonos prefixű mátrixokra, a fa bejárása folytatódhat úgy, mintha már bejártuk volna az adott csúcs összes gyerekcsúcsát.

További lehetőség van a számítási igény csökkentésére akkor, ha adott k mellett teljesülnek a mátrixra a következő feltételek

- *k-prefix-biztos*
- $(k-1)$-*prefix-ütemezhető*
- *nem k-prefix-ütemezhető*

Ekkor az m, k paraméterű mátrixot reprezentáló csúcs gyerekeinél nem kell megvizsgálni az ütemezhetőséget, mert azok garantáltan nem lesznek ütemezhetőek.

10.5. Számítási igény vizsgálata

Induljunk ki abból, hogy megvizsgáljuk az összes lehetséges mátrixot m, n paraméterek esetén. Ekkor az összes lehetséges bináris mátrix száma 2^{m*n}. Az egyszerűség kedvéért számoljuk azt, hogy hány oszlopot kell megvizsgálnunk. Jelöljük ezt rögzített m esetén $T(n)$-nel.

- Brute-Force esetben: $T_{Brute-Force}(n) = n * 2^{m*n}$.

- Prefixadatok ismeretében:

  Az n-ik szinten 2^{m*n} mátrix van, mindegyiknek egyetlen oszlopát vizsgáljuk csak meg. Az $n-1$-ik szinten $2^{m*(n-1)}$ oszlopot kell vizsgálnunk, stb.

  Összesen $\sum_{k=1}^{n} 2^{m*k}$ oszlopot vizsgálunk meg.

A mértani sorozat összegképletét véve

$$T_{Prefix}(n) = \sum_{k=1}^{n} 2^{m*k} = 2^m * \frac{2^{m*n} - 1}{2^m - 1}$$

Rögzített m mellett a következő konstans szorzót emelhetjük ki

$$c = \frac{2^m}{2^m - 1}.$$

Megjegyezzük, hogy m szerint határértéket véve

$$\lim_{m \to \infty} \frac{2^m}{2^m - 1} = 1.$$

Ezek alapján elmondhatjuk, hogy

$$T_{Brute-Force}(n) = c * n * T_{Prefix}(n).$$

Eszerint a prefixadatokat felhasználó algoritmus aszimptotikusan $\Theta(n)$-szer kevesebb oszlopot vizsgál meg.

$$T_{Brute-Force}(n) = n * \Theta(T_{Prefix}(n)).$$

10.6. Korreláció felhasználása ütemezhetőség ellenőrzéshez

Korábbi eredményeink alapján egy mátrix ütemezhetőségének eldöntéséhez legrosszabb esetben $\Theta(n^m)$ csúcs vizsgálatára volt szükség. Ezt az aszimptotikus értéket próbáljuk meg csökkenti a már megvizsgált mátrixok segítségével. Ehhez két ötletet fogunk felhasználni.

- az ütemezhetőséghez szükség van a *k-prefix-ütemezhetőségre* ($k = 1, ..., n-1$).

- Ha egy n oszlopos mátrix első $n-1$ oszlopa ütemezhető, akkor ha ismerjük az összes olyan csúcsot, amelyeknek mindkét koordinátája kisebb egyenlő, mint $n-1$ és legalább egyik koordináta ezek közül $n-1$ és irányított út vezet hozzá az origóból, akkor elegendő megvizsgálni, hogy ezen csúcsokból indul-e él olyan csúcsba, amelynek legalább $m-1$ koordinátája n.

Emlékeztetésül megjegyezzük, hogy egy mátrix akkor és csak akkor ütemezhető, ha a neki megfeleltett gráfban vezet irányított út az origóból olyan csúcsba, melynek legalább $m-1$ koordinátája n, továbbá a korábban ismertett két gráfmegfeleltetést tetszés szerint alkalmazhatjuk a feladat megoldása során.
Az új, ütemezhetőséget ellenőrző algoritmus ezek alapján a következő

- Válasszunk egy gráfmegfeleltetési módszert az ismertetettek közül

- Miközben az összes mátrixnak megfeleltetett fát inorder módon járjuk be, az aktuális úton fentről lefele minden csúcsnál tartsuk nyilván a neki megfelelő mátrix azon csúcsait, melyekbe vezet út az origóból és a koordináták közül legalább $m-1$ darab n

- Amikor a fában a következő szintre lépünk, akkor az adott csúcs szülőjénél nyilvántartott csúcsokat kiterjesztjük az aktuális csúcsnak megfelelően és az új csúcsokat felvesszük az adott csúcs nyilvántartásába

- Az algoritmust az első szinten kezdjük és a képzeletbeli 0. szinten a rendelkezésre álló nyilvántartott csúcs az origó

- Ha a k-ik szinten nem tudunk kiterjeszteni egy $k-1$ szinten tárolt csúcsot sem, akkor a mátrix nem *k-prefix-ütemezhető*, így nem szükséges a fában a levelek felé továbbhaladni a vizsgálattal

- Ha a vizsgálat megszakítására került sor, akkor a fában történő visszalépéskor a csúcs által tárolt adatokra már nincs szükség, így azokat töröljük

- Ha az n-ik szinten egy csúcs ütemezhető, akkor a neki megfelelő mátrix ütemezhető

10.7. Számítási igény vizsgálata

A futási időt a vizsgált csúcsok számának ismeretében fogjuk értelmezni. Legrosszabb esetben egy mátrix ütemezhetőségének vizsgálatához a megfeleltett gráfot teljes egészében be kellett járnunk, amely $\Theta(n^m)$ csúcsot jelentett, vagyis az összes mátrix esetében ez $\Theta(n^m * 2^{n*m})$. Nevezzük ezt a módszert Brute-Force-nak.

Ha feltesszük, hogy nem alkalmazzuk a fában a visszavágási lehetőséget, akkor a következőket mondhatjuk el most, a prefix tulajdonságok esetében.

- A k-ik szinten egy adott facsúcsnál $m * (k-1) + 1$ gráfcsúcsot kell megvizsgálni
- A k-ik szinten 2^{m*k} mátrix található

Ezek alapján a k-ik szinten megvizsgált gráfcsúcsok száma

$$(m * (k-1) + 1) * 2^{m*k}.$$

Tehát összesen

$$\sum_{k=1}^{n} (m * (k-1) + 1) * 2^{m*k}.$$

Ezt egyszerűbb alakra hozva kapjuk, hogy

$$\sum_{k=1}^{n} (m * (k-1) + 1) * 2^{m*k} =$$

$$= 2^m * \frac{2^{m*n} - 1}{2^m - 1} + m * \sum_{k=1}^{n} (k-1) * (2^m)^k =$$

$$= 2^m * \frac{2^{m*n} - 1}{2^m - 1} + m * 2^m * \sum_{k=0}^{n-1} k * (2^m)^k =$$

$$= 2^m * (\frac{2^{m*n} - 1}{2^m - 1} + m * \frac{2^{m(n+1)}(n2^m - n - 1) + 2^m}{(2^m - 1)^2})$$

Rögzített m paraméter mellett – elhagyva a konstans szorzókat –, a következő aszomptotikus értéket kapjuk

$$\Theta(n * 2^{mn} * 2^m) = \Theta(n * 2^{mn}).$$

Amikor külön-külön vizsgáltuk az összes mátrixot, akkor az alábbi aszimptotikus értéket kaptuk

$$\Theta(n^m * 2^{mn}).$$

Vizsgálataink során olyan mátrixokkal foglalkozunk, ahol $2 \leq m$, ezért a két érték hányadosának határértéke

$$\lim_{n\to\infty} \frac{n * 2^{mn}}{n^m * 2^{mn}} = \lim_{n\to\infty} \frac{n}{n^m} = 0.$$

Másképpen megfogalmazva, a futási időkre a következő összefüggés adódik

$$T_{Brute-Force}(n) = \Theta(n^m * T_{Prefix}(n)).$$

11. Javasolt implementációs módszerek és eszközök

Ebben a fejezetben ismertetjük azokat a fontosabb meggondolásokat, melyeket célszerű az implementáció során figyelembe venni, illetve implementációs (programozási) eszközöket javasolunk egy konkrét szimulációs program elkészítéséhez.

1. Mivel a legtöbb programozási nyelvben a számok ábrázolása egy n bites egészként realizálódik, ezért sokszor 2^{mn} értéke nem ábrázolható. Ezen korlátok kiküszöbölésére saját magunknak kell megvalósítanunk a "nagy számok" típust.

   - A szimulációban kiszámolandó valószínűségek számolása végett a "nagy számok" típusnak el kell készíteni a lebegőpontos változatát is.
   - Külön valósítsuk meg az egész és a lebegőpontos típust.
   - A gyors számításokhoz implementáljuk úgy az egész típust, hogy az eggyel növelés (inkrementálás), hatékony művelet legyen.

2. A "fa" bejárásához választhatunk a rekurzió és annak iteratívvá transzformált változata között.

   - Rekurzió használata mellett számolnunk kell a programozási nyelveknél a fordító-programok által létrehozott plusz műveletekkel: függvényhívásoknál verem felépítése, aktivációs rekordok elkészítése, karbantartása...
   - Iteráció esetében megspórolhatjuk a függvényhívások idejét, viszont számolnunk kell azzal, hogy a rekurzió esetében megírt függvényeink lokális változóit elmentsük. Mivel a szimuláció indulásakor n értéke ismert, ezért ennek egy egyszerű módszere az, ha a lokális változókat n dimenziós tömbökként hozzuk létre.

3. A feladat jellegéből adódóan a bináris mátrixok egy oszlopa felfogható egy binárisan ábrázolt számként. Ez azért előnyös, mert a programozási nyelvekben rendelkezésre álló egész szám típus helyettünk "csoportosítja" azokat.
   Amennyiben m "nagy", úgy egy oszlop ábrázolásához k bites számítógép architektúra esetén $\lceil m/k \rceil$ méretű egészeket tartalmazó tömb szükséges.

12. Összefoglalás

Az $m \geq 2$ sort tartalmazó mátrixokra megadtuk az ütemezhetőségi függvény explicit alakját és meghatároztuk az $s_{crit}(m)$ kritikus valószínűségeket. $s_{crit}(2)$ értéke a több perkolációs modellre jellemző 1, a további kritikus valószínűségek értéke m növekedtével csökken.

A szimulációs vizsgálatok szerint a kritikus ütemezhetőségi valószínűségek közel vannak a kapott felső korlátokhoz: az 9.1.2. táblázat a $p = 0.5$, az 9.1.3. táblázat a $p = 0.4$, az 9.1.4. táblázat pedig a $p = 0.35$ valószínűséghez tartozó adatokat mutatja.

A táblázatok adatai alapján azt sejtjük, hogy a [7] cikkben szereplő korlátnál jobb is adható: ha $p < 0.4$, akkor $w(2, p) > 0$.

A $w(m, p)/s(m, p)$ hányadosok viselkedése még további elemzést kíván.

A cikkben szereplő alsó korlátnál nagyobbakat is meg tudunk adni, de azokból is csak a természetes nulla alsó korlát adódik. A kritikus valószínűségek pontosabb jellemzéséhez a jó mátrixoknál hasznosabb mátrixokra van szükség.

Jelölési konvenciók

Fejezetek, számozás, ábrák

A fejezetek, ábrák és képletek számozási stílusa, valamint az ábrák és táblázatok elrendezése, a hozzájuk kapcsolódó megjegyzésekkel együtt az Informatikai algoritmusok című könyv konvencióit követik.

Pszeudokódok

A pszeudokódok jelölésrendszere úgy került kialakításra, hogy az ne csak absztrakt programkódként legyen értelmezhető, hanem a konkrét implementáláshoz segítséget nyújtson.

A C nyelv szabályait követi az értékadási jel és a vezérlési szerkezetek nevei, valamint a vezérlési szerkezetekhez tartozó kódblokkok beljebb kezdéssel való jelölése. Az említett kódblokk jelölési technika alkalmazása miatt a **then**, **do** kifejezések elhagyásra kerültek.

Az alacsonyszintű műveleteket, – mint a biteltolás, összeadás, stb. – a C nyelv műveletei jelölik. Azok a műveletek, amelyek pedig mind egy programozási nyelv, mind pedig egy pszeudokód számára absztraktak (lásd halmazműveletek), azok a matematikai jelölésrendszer segítségével kerültek leírásra. Ez a jelölésmód nagyban elősegíti azt, hogy a kód olvasásakor szembetűnően látszódjék az, hogy melyek azok a programrészek, amelyek a futásidőt nagyban befolyásolják (gépközeli hatékony leírás), és melyek azok, amelyek ezen szempontból kevésbé fontosak.

Irodalomjegyzék

1. P. Balister, B. Bollobás, M. Walters (2004), Continuum percolation with steps in an annulus. *Ann. Appl. Probab.* **14/4** 1869–1879. http://arxiv.org/find. 2

2. A. Bege, Z. Kása (2001), Coding objects related to Catalan numbers, Studia Univ. Babeş-Bolyai, *Informatica*, Volume **XLVI** (1), 31–39.
http://www.cs.ubbcluj.ro/ studia-i/contents.php. 11

3. B. Bollobás, O. Riordan (2005), A short proof of the Harris-Kesten theorem. Electronic manuscript, http://arxiv.org/find. 2

4. I. Derényi, G. Palla, T. Vicsek (2005), Clique percolation in random networks. *Phys. Rev. Lett.* **94** 160–202. http://arxiv.org/find. 2

5. W. Feller, (1968), An Introduction to Probability Theory and its Applications. *John Wiley and Sons*, New York. 3

6. P. Gács (2002), Clairvoyant scheduling of random walks (submitted to *Electronic Journal of Probability*). Short version: Clairvoyant scheduling of random walks. In: *Proc. of the Thirty-Fourth Annual ACM Symposium on Theory of Computing*, 99–108 (electronic), ACM, New York, 2002. http://www.cs.bu.edu/fac/gacs/recent-publ.html. 2

7. P. Gács (2004), Compatible sequences and a slow Winkler percolation. *Combin. Probab. Comput.* **13/6**, 815–856. http://www.cs.bu.edu/fac/gacs/recent-publ.html. 2, 3, 5, 20

8. T. E. Harris (1960), A lower bound for the critical probability in a certain percolation process. *Proc. of the Cambridge Phil. Soc.* **56** 13–20

9. A. Hunt (2005), Percolation Theory for Flow in Porous Media. *Lecture Notes in Physics*, Vol. 674. 2

10. A. Iványi, 2-biztos mátrixok sűrűsége (kézirat). 3, 11

11. Z. Kása (2003), *Combinatorică cu aplicaţii.* Presa Universitară Clujeană, Cluj-Napoca. 3, 11

12. Z. Kása (2004), Rekurzív egyenletek. *Informatikai algoritmusok. 1*
(Szerk. A. Iványi). ELTE Eötvös Kiadó, Budapest, 14–37. http://elek.inf.elte.hu/. 11

13. H. Kesten (1982), *Percolation Theory for Mathematicians.* Birkhäuser, Boston. 2

14. Cs. Láng (1994), *Bevezető fejezetek a matematikába 1.* ELTE Eötvös Kiadó, Budapest. 3, 11

15. D. Schauffer (1985), *Introduction to Percolation Theory.* Taylor and Francis, London, 1985. 2

16. R. P. Stanley (1999), *Enumerative Combinatorics*, Vol 2. *Cambridge Studies in Advanced Mathematics*, 62. Cambridge University Press, Cambridge. 11

17. P. Szász (1951), *A differenciál- és integrálszámítás elemei.* Közoktatásügyi Kiadóvállalat, Budapest. 15

18. N. J. Vilenkin (1971), *Kombinatorika.* Műszaki Könyvkiadó, 2, 3, 11

19. P. Winkler (2000), Dependent percolation and colliding random walks, *Random Structures & Algorithms* 16/1 58–84.
http://www.math.dartmouth.edu/ pw/papers/pubs.html. 2, 3