Gólelosztások

Bevezetés

Tekintsük a következő problémát. Adott n db futballcsapat, mindannyian javaslatot tesznek, hogy összesen hány gólt akarnak rúgni, illetve kapni. A feladatunk egy olyan bajnokság szervezése, amelyben mindenkinek teljesül a kívánsága. Első lépésként tekintsük azt a változatot, amikor a feladat megoldható, ekkor előállítunk egy megoldást, a későbbiekben azonban arra keressük a választ, hogy a nem megoldható feladatokra hogyan tudunk olyan kvázimegoldást adni, amelyben csak egyetlen csapat kívánsága sérül a lehető legkisebb mértékben.

Legyen $n \geq 2$ egész, a, b tetszőleges nemnegatív egészek. A $T_n(a, b)$ n-pontú irányított gráfot (a, b)-versenynek nevezzük, ha minden csúcsa között legalább a és legfeljebb b irányított él vezet. Ha a kivezető éleket azonosítjuk a "nyert" pontokkal, akkor a v_i csúcs kivezető éleinek számát, azaz kifokát d_i, a befokát l_i jelölje. Ekkor a $D = (d_1, d_2, \dots, d_n)$ vektort a verseny pontvektorának, az $L = (l_1, l_2, \dots, l_n)$ vektort pedig a vesztővektorának nevezzük.

1. Tétel (Hakimi) Legyen $n \geq 2$ egész, és legyen $D = (d_1, d_2, \dots, d_n)$, $L = (l_1, l_2, \dots, l_n)$ két nemnegatív egészeket tartalmazó vektor, melyek elemei úgy rendezettek, hogy

$d_i + l_i \leq d_{i+1} + l_{i+1}$ $(i = 1, \dots, n-1)$. Akkor és csak akkor van olyan $(0, \infty)$-verseny, amelynek pontvektora D, vesztővektora pedig L, ha

$$\sum_{i=1}^{n} d_i = \sum_{i=1}^{n} l_i \tag{1}$$

valamint

$$\sum_{i=1}^{n-1} d_i + l_i = d_n + l_n \tag{2}$$

Megjegyzés A fenti tételt átfogalmazhatjuk nem rendezett vektorokra is, ekkor a (2) feltétel helyett azt követeljük meg, hogy

$$\sum_{i=1}^{n} d_i + l_i - \max_{\mathrm{i}}(\mathrm{d_i} + \mathrm{l_i}) \geq \max_{\mathrm{i}}(\mathrm{d_i} + \mathrm{l_i})$$

Jelentse a továbbiakban M azt az indexet, amelyre $d_M + l_M := \max_{\mathrm{i}}(\mathrm{d_i} + \mathrm{l_i})$. Így tehát

$$\sum_{i=1}^{n} d_i + l_i \geq 2 \max_{\mathrm{i}}(\mathrm{d_i} + \mathrm{l_i}) = 2d_M + 2l_M \tag{3}$$

Az (1) és (3) feltételeknek eleget tevő D, L vektorokat realizálhatónak nevezzük (Hakimi).

A feladatunk a következőkben az lesz, hogy előállítsuk egy adott pont- és vesztővektorhoz tartozó versenyt.

Konstrukció

Adott tehát egy $n \geq 2$ pozitív egész, valamint D, L n elemű pont- és vesztővektorok. Az alapgondolat az, hogy ha egy mátrix (i, j) eleme azt jelenti,hogy az i csapat hány pontot szerzett a j csapat elleni mérkőzésen, akkor (j, i) éppen annyi, ahány pontot vesztett ellene ugyanazon a meccsen. Így ha a mátrix elemeit összeadjuk az i sorban, akkor éppen a pontvektor i elemét kapjuk, míg az i oszlopban összeadva az elemeket a vesztővektor megfelelő eleméhez jutunk. Az eredeti feladat tehát felírható úgy, hogy keressük azt az

$$A = \begin{matrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{matrix}$$

$n \times n$-es mátrixot, amelyre teljesül

$$\sum_{j=1}^{n} a_{ij} = d_i \qquad (i = 1,2, \dots n)$$

$$\sum_{i=1}^{n} a_{ij} = d_j \qquad (j = 1,2, \dots n)$$

A feltételezésünk az, hogy nem tejesülnek a fentiek, és ezért bevezetünk egy sorhiba- és oszlophiba vektort, amelyek az eltérést mutatják. Az egyenletek tehát

$$\varepsilon_i + \sum_{j=1}^{n} a_{ij} = d_i \qquad (i = 1,2, \dots n) \tag{4}$$

$$\delta_j + \sum_{i=1}^{n} a_{ij} = l_j \qquad (j = 1,2, \dots n) \tag{5}$$

Ha a feladat teljesíthető, akkor a célunk az, hogy $\|\varepsilon\| + \|\delta\| \to min$, ahol $\|.\|$ jelentse valamely szokásos, vektortér felett értelmezett normát[1]. Mi a továbbiakban l_1 normában gondolkodunk, azaz vektorok elemeinek abszolútérték összegét kívánjuk minimalizálni. Abban az esetben, ha a probléma megoldható, akkor $\|\varepsilon\| + \|\delta\| = 0$ teljesíthető. A továbbiakban algoritmust adunk az A mátrix

[1] Különböző normákkal részben különböző feladatok oldhatók meg, hogy csak a legszokványosabbakat említsük: az 1-es norma csak a felhalmozott hibát kívánja minimalizálni; az euklideszi normával kedvezőbb az az eset, amikor több csapatnál jelentkezik kevesebb hiba, mint ha egynél az összes; maximum normánál az a legjobb eshetőség, ha az összes hibát a lehető legegyenletesebben osztjuk el a lehető legtöbb csapat között.

konstrukciójára, ami természetes módon azonosítható a keresett versenyhez tartozó szomszédsági mátrixszal.

Az MGrow algoritmus

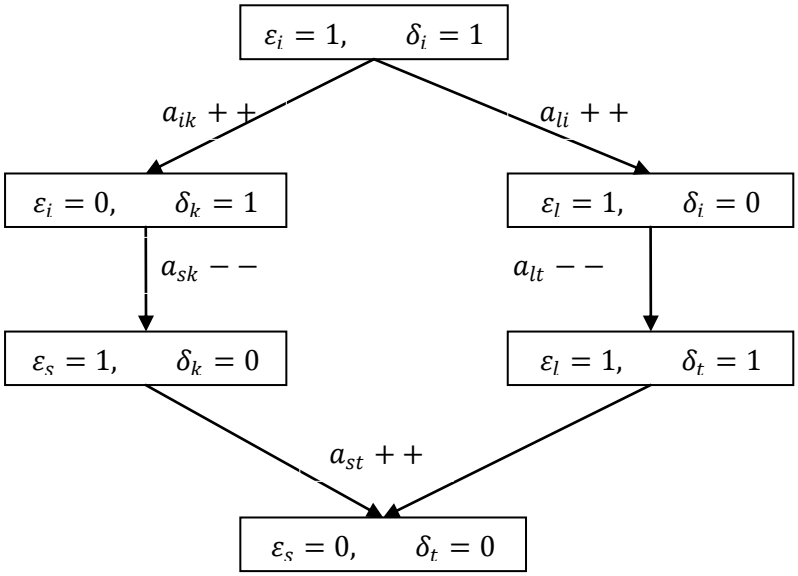
A verseny létrehozását tehát visszavezettük egy mátrix megkonstruálásának problémájára. Induljunk ki a (4), (5) egyenletekből! Legyen kezdetben A az $n \times n$-es nullmátrix, ekkor $\varepsilon_i = d_i$, $\delta_i = l_i$ minden $i = 1..n$ esetén. A célunk tehát $\|\varepsilon\|$ és $\|\delta\|$ minimalizálása. Az alapötlet az, hogy ε és δ vektor egyszerre csökkenthető rendre az i és j pozíciókban, ha pozitívak, valamint $i \neq j$, hiszen ez a szóban forgó csapat saját maga ellen játszott meccsét jelentené, ezt viszont nem engedjük meg. Ekkor ha a mátrix (i, j) pozicióban lévő elemét eggyel növeljük, akkor ε_i és δ_j eggyel csökkenthető, úgy hogy továbbra is fennáll (4), (5)! Sok stratégia létezhet a megfelelő elemek kiválasztására, mi egy sorfolytonos elhelyezést mutatunk a *Growing* algoritmusban.

$Growing(n, \varepsilon, \delta)$
for $i = 1..n$ **do**
  for $j = 1..n$ **do**
    if $\varepsilon_i > 0$ **and** $\delta_j > 0$ **and** $i \neq j$
      $a_{ij} = a_{ij} + 1,\ \ \varepsilon_i = \varepsilon_i - 1,\ \ \delta_j = \delta_j - 1$

A fenti algoritmus egy lefutás alkalmával egy sorban legfeljebb n-1 elemet helyez el a fenti kritériumok betartásával, így természetes módon csökkentve ε és δ vektorokat. Újra és újra lefuttatva a *Growing*-ot mindaddig csökkenthetők a hibák, amíg a hibavektorok különböző $i \neq j$ pozícióiban van pozitív elem. Végül két helyzet állhat elő

> 1) Minden elemet el tudtunk helyezni a mátrixban, azaz ε és δ nullvektorok, ekkor készen vagyunk.
>
> 2) A hibavektorok már csak egyetlen $i = j$ pozícióban tartalmaznak nemnulla elemet, ezért a *Growing* algoritmust nem tud tovább csökkenteni. Mivel az előállt helyzet azzal oldódna meg, hogy nem megengedett módon a mátrix átlójában helyeznénk el elemet, a továbbiakban ezt a problémát diagonális hibának fogjuk nevezni.

A kérdés tehát az, hogy mit tudunk kezdeni a diagonális hibával. Az alábbiakban mutatunk egy módszert, amivel a diagonális hiba eggyel csökkenthető. Tegyük fel, hogy a a hibavektorok már csak az i-edik pozícióban tartalmaznak nemnulla elemet. Ekkor vegyünk két indexet, pl. $k \neq i, l \neq i$. Ha a_{ik} , a_{li} elemeket megnöveljük, akkor ε_i és δ_i csökken, cserébe viszont δ_k és ε_l 0-ról 1-re vált. Ha most választunk további két megfelelő $s \neq t$ indexet, akkor a_{sk} és a_{lt} csökkentésével az előbb elrontott δ_k és ε_l helyreáll, de megjelenik az ε_s , δ_t helyeken a 0 helyett az 1. Mivel $s \neq t$, az a_{st} elemet megnövelhetjük, így eltűntetve az előbb keletkezett hibát. Vegyük észre, hogy a fenti transzformációk végrehajtásával a diagonális hiba eggyel csökken, anélkül hogy máshol hibát okoznánk! Tekintsük át tehát mégegyszer, hogy mi kell ahhoz, hogy végrehajthassuk a diagonális hibát csökkentő transzformációt!



A fentiekkel bebizonyítottuk a következő lemmát.

1. Lemma. Legyen a (4), (5) egyenletekkel adott rendszerben ε és δ vektorok egyetlen nemnulla eleme az i pozícióban. Ha léteznek olyan $k, l, s, t \epsilon [1..n]$ indexek, hogy $k \neq i, k \neq s, l \neq i, l \neq t$, $s \neq t$, valamint $a_{sk}, a_{lt} > 0$, akkor a diagonális hiba 1-gyel csökkenthető.

Az 1. lemma által megfogalmazott vizsgálatot az összes lehetséges elemre elvégzi a *DecDiagErr* algoritmus, és ha talált megfelelő indexeket, akkor csökkenti a diagonális hibát.

$DecDiagErr(n, \varepsilon, \delta)$
$s = 1,\ \ k = 1,\ \ failed = \textbf{true}$
$\textbf{while not}\ jumpoff\ \textbf{and}\ failed$
$failed = \textbf{false}$
  $\textbf{while}\ a_{sk} \leq 0\ \textbf{or}\ s = k\ \textbf{or}\ i = k\ \textbf{do}$
    $\textbf{if}\ s < n\ \textbf{then}$
      $s = s + 1$
    $\textbf{else}$
      $s = 1,\ \ k = k + 1$
    $\textbf{if}\ k > n\ \textbf{then}$
      $jumpoff = \textbf{true}$
      $\textbf{break}$
  l = 1, t = 1
  $\textbf{while}\ a_{lt} \leq 0\ \textbf{or}$ t = l $\textbf{or}$ i = l $\textbf{or}\ t = s$ **do**
    $\textbf{if}\ l < n\ \textbf{then}$
     $l = l + 1$
    $\textbf{else}$
      $l = 1,\ t = t + 1,\ \ failed = \textbf{true}$
    $\textbf{if}\ t > n\ \textbf{then}$
     $t{=}1, k = k + 1$
      $\textbf{break}$
$a_{ik} = a_{ik} + 1,\ \ a_{li} = a_{li} + 1$
$a_{sk} = a_{sk} - 1\ \ a_{lt} = a_{lt} - 1$
$a_{st} = a_{st} + 1$
$\varepsilon_i = \varepsilon_i - 1,\ \ \delta_i = \delta_i - 1$

A módszer kritikus pontja az, hogy találunk-e megfelelő pozícióban lévő szigorúan pozitív elempárt. A *DecDiagErr* algoritmus keres egy pozitív elemet, majd ehhez probál találni egy megfelelő párt, az 1. lemmában megfogalmazott kritériumok betartásával. Ha egy elemhez nem talál megfelelő párt, az még nem jelenti azt, hogy egy másik elemhez sem találhat, ekkor a $failed$ értéke igaz, és a vizsgálatot újrakezdjük a következő pozitív elem keresésével. Ha a $failed$ hamis, akkor találtunk elempárt. Ha végigvizsgálva a mátrix összes elemét sem találunk megfelelő párt, akkor a $jumpoff$ változó igazra állítása mellett befejezzük a futást, ellenkező esetben csökkentjük a diagonális hibát. Az algoritmus mohósága és naivitása miatt elég lassú, a problémával kapcsolatos további észrevételeket az utólagos megjegyzésekben tárgyaljuk.

Az eddig áttekintett két algoritmus egymásutánja megoldhatja tehát a problémát. Azt kell még tisztáznunk, hogy mikor melyik részt kell alkalmazni. Kezdésképpen a $Growing$ algoritmust kell futtatnunk, majd ha az ε és δ vektorok nemnulla elemeinek száma egyaránt 1, akkor diagonális hibánk van, tehát át kell váltanunk a $DecDiagErr$ algoritmusra. Akkor hagyjuk abba transzformálást, amikor a hibavektorok nullvektorok, vagy akkor, ha valami hibát találunk, ami eredhet abból, hogy nem tudjuk kitranszformálni a diagonális hibát, vagy abból, hogy a diagonális hiba nem szimmetrikus, tehát már az egyik hibavektorunk már nullvektor, a másik még nem az. A fentiek figyelembevételével elkészíthetjük az $MGrow$ algoritmust. $NZ(x)$ jelentse az x vektor nemnulla elemeinek számát.

$MGrow(n, D, L)$
  $A = 0,\ \ \varepsilon = D,\ \ \delta = L$
  $diagerr = \boldsymbol{false},\ \ jumpoff = \boldsymbol{false}$
  $\boldsymbol{while}\ \|\varepsilon\| + \|\delta\| \neq 0\ \boldsymbol{and\ not}\ jumpoff\ \boldsymbol{do}$
    $\boldsymbol{if}\ NZ(\varepsilon) = NZ(\delta) = 1\ \boldsymbol{then}$
      $diagerr = \boldsymbol{true}$
    $\boldsymbol{if}\ NZ(\varepsilon)NZ(\delta) = 0$
      $jumpoff = \boldsymbol{true}$
    $\boldsymbol{if\ not}\ diagerr\ \boldsymbol{then}$
      $Growing(n, \varepsilon, \delta)$
    $\boldsymbol{else}$
      $DecDiagErr(n, \varepsilon, \delta)$

Az MGrow algoritmus jó

A továbbiakban azt szeretnénk megmutatni, hogy az $MGrow$ algoritmus megoldja a kitűzött feladatot, tehát terminál, és minden olyan esetben amikor a feladat megoldható, a kiugrási feltétel nem válik igazzá ($jumpoff$), azaz egy helyes megoldást állít elő. Ehhez fel fogjuk használni az 1. Tétel (Hakimi) feltételeit.

2. Tétel. Ha az $n \geq 2$, $D = (d_1, \dots, d_n)$, $L = (l_1, \dots, l_n)$ nemnegatív egészeket tartalmazó vektorok, melyekre teljesülnek az (1), (3) feltételek, akkor az MGrow algoritmus az (n,D,L) inputra egy megengedett megoldást állít elő.

Bizonyítás Az előzőekben láttuk, hogy a *Growing* algoritmus addig csökkenti a hibavektorokat, míg vagy megoldja a feladatot, vagy diagonális hibát hoz létre. Tobábbá adtunk egy elégséges feltételt arra, hogy a diagonális hiba csökkenthető legyen. Ahhoz, hogy belássuk, hogy az MGrow megoldja a megoldható feladatot, ahhoz két dolgot kell megmutatnunk:

i) a *Growing* algoritmus lefutása után a hibavektorokban megmaradt elemek egyenlők

ii) a diagonális hiba 0-ra csökkenthető

Tegyük fel, hogy i) nem teljesül. Ekkor tehát a *Growing* lefutása után a hibavektorokban egyetlen k pozícióban maradt csak egy-egy elem, és ezek nem egyenlők. Jelölje a *Growing* lefutása után előállt hibavektorokat $\tilde{\varepsilon}, \tilde{\delta}$:

$$\tilde{\varepsilon}_i = \tilde{\delta}_i = 0 \quad (i = 1, \dots, n \quad i \neq k)$$

$$\tilde{\varepsilon}_k, \tilde{\delta}_k > 0,\ \ \tilde{\varepsilon}_k \neq \tilde{\delta}_k$$

A *Growing* csak úgy csökkentheti valamely ε_i-t ha valamely δ_j-t is csökkenti, ebből viszont az következik, ha a mátrixba helyeztünk már N elemet, akkor

$$\sum_{i=1}^{n} \varepsilon_i - \tilde{\varepsilon}_k = N = \sum_{i=1}^{n} \delta_i - \tilde{\delta}_k$$

Ekkor viszont $\tilde{\varepsilon}_k \neq \tilde{\delta}_k$ miatt

$$\sum_{i=1}^{n} \varepsilon_i \neq \sum_{i=1}^{n} \delta_i$$

ami azt jelenti, hogy (1) nem teljesül. Hasonlóan beláthatjuk fordítva is az állítást. Ha feltesszük, hogy (1) nem teljesül, akkor tehát valamely $t \neq 0$ egészre

$$t + \sum_{i=1}^{n} \varepsilon_i = \sum_{i=1}^{n} \delta_i$$

Ha a *Growing* algoritmus N lépésben leáll, és diagonális hibát okoz, akkor

$$t + N + \tilde{\varepsilon}_k = N + \tilde{\delta}_k$$

így t nemnulla volta miatt kapjuk a bizonyítandó állítást.

A fentiekben azt láttuk be, hogy az (1) állítás nem teljesülése ekvivalens azzal, hogy a *Growing* algoritmus terminálásakor a sor- és oszlophiba vektorokban maradt elemek nem egyenlők. Ez azt jelenti, hogy minden olyan nem megoldható feladat, amely (1)-et teljesíti és (3)-t nem, valamint a megoldható feladatok mind olyanok, hogy az i) állítást teljesítik.

Az ii) állítás belátásához azt mutatjuk meg, hogy az 1. Lemma feltételei teljesednek. Feltesszük, hogy a *Growing* algoritmus lefutott, és diagonális hibát okozott a k-adik pozícióban. Bizonyítanunk kell, hogy van legalább egy olyan pozitív elempár, hogy az 1. nem a k-adik sorból, a 2. pedig nem a k-adik oszlopból való, továbbá az 1. oszlopa nem egyezik meg a 2. sorával! Tekintsük a következő halmazokat!

$$N_1 = \sum_{i=1}^{n} \varepsilon_i - \varepsilon_k$$

$$N_2 = \sum_{i=1}^{n} \delta_i - \delta_k$$

Az 1. elemet az N_1 halmazból kell kiválasztanunk, a 2. elemet az N_2-ből.

Tegyük fel, hogy $M \neq k$. Ekkor M definíciójából következik, hogy

$$\forall i = 1, \dots, n \quad i \neq k\colon \ \varepsilon_M + \delta_M \geq \varepsilon_i + \delta_i$$

Viszont, ha például k-ra egyenlőség teljesül, az (3) miatt $n \geq 3$ esetén csak úgy lehet, ha az összes többi eleme mindkét vektornak nulla.

$$\exists k \in [1..n]\colon \ \varepsilon_k + \delta_k = \varepsilon_M + \delta_M \overset{(3)}{\Rightarrow} \varepsilon_i = \delta_i = 0 \quad (i \neq k, M)$$

A feladatnak ekkor csak úgy létezik megoldása, ha

$$\varepsilon_k = \delta_k = \varepsilon_M = \delta_M$$

Vegyük észre, hogy ekkor viszont nem kapunk diagonális hibát, mivel a *Growing* algoritmus 0-ig tudja csökkenteni a hibavektorokat!

Ha viszont az egyenlőség nem teljesülhet, akkor a következő teljesül

$$\varepsilon_k + \delta_k < \varepsilon_M + \delta_M$$

amiből

$$N_1 + N_2 = \sum_{i=1}^{n} \varepsilon_i - \varepsilon_k + \sum_{i=1}^{n} \delta_i - \delta_k = \sum_{i=1}^{n} (\varepsilon_i + \delta_i) - \varepsilon_k - \delta_k \geq$$
$$\geq 2(\varepsilon_M + \delta_M) - (\varepsilon_k + \delta_k) > \varepsilon_M + \delta_M$$

Azt láttuk be tehát, hogy elhagyva a mátrixból a k-adik sort és oszlopot, még mindig több gól van bent, mint $\varepsilon_M + \delta_M$, ebből viszont az is következik, hogy az nem fordulhat elő, hogy az összes elem egy sorban és oszlopban van, hiszen ha lenne ilyen m index, akkor erre $\varepsilon_m + \delta_m > \varepsilon_M + \delta_M = \max_i(\varepsilon_i + \delta_i)$ adódna, ami nyilvánvalóan lehetetlen. Kell tehát lennie legalább két olyan elemnek a mátrixban, ami teljesíti az 1. Lemma feltételeit. Ráadásul nem használtunk fel semmit abból, hogy hogyan transzformálódott a mátrix a *Growing* algoritmus alatt, következésképpen ha egy diagonális hibát kitranszformálunk, és megváltozik a mátrix, a fenti bizonyítás ugyanilyen alakú lesz, így a diagonális hiba, nagyságától függetlenül nulláig csökkenthető.

Vizsgáljuk meg azt a lehetőséget, amikor $M = k$, azaz a diagonális hiba a maximális pozícióban jelentkezik. Ekkor nem adhatunk szigorú becslést

$$N_1 + N_2 = \sum_{i=1}^{n} (\varepsilon_i + \delta_i) - \varepsilon_M - \delta_M \geq$$
$$\geq 2(\varepsilon_M + \delta_M) - (\varepsilon_M + \delta_M) = \varepsilon_M + \delta_M$$

Viszont a fentiekben egyszer már alkalmazott gondolatmenetet követve,ha feltesszük, hogy minden elem a megmaradt mátrixban ugyanolyan indexű sorban és oszlopban vannak, akkor az feltételezi egy olyan m indexű játékos jelenlétét, amelyre $\varepsilon_m + \delta_m = \varepsilon_M + \delta_M$, az előzőekben láttuk, hogy ekkor az összes többi játékos 0 gólt rúg és kap, és csak a triviális esetben megoldható. Ekkor viszont a *Growing* már megoldotta volna a problémát, tehát feltehetjük, hogy nincs még egy maximális gólszámú játékos, azaz az $\varepsilon_M + \delta_M$ számú pont legalább két játékos között oszlik el. Ez pedig pontosan azt jelenti, hogy a diagonális hibát csökkentő elemek kiválasztásakor létezik két különböző sorban és oszlopban lévő pozitív elem.

Itt is megjegyezhetjük, hogy nem tettünk említést a mátrix struktúrájára, tehát ez bármely átrendezésre igaz, azaz végrehajtva a diagonális hibát csökkentő eljárást, az mégegyszer végrehajtható. Ez a tulajdonság nem függ a mátrix elemeinek eloszlásától tehát, kizárólag annak a következménye, hogy az elemek teljesítik a (3) feltételt, valamint annak, hogy a *Growing* algoritmus hogyan állítja elő a diagonális hibát.

■

Megjegyzés Felhívnánk mégegyszer a figyelmet arra, hogy az i) bizonyításához elég feltennünk (1)-et, hasonlóan ii) bizonyításához önmagában elegendő (3).

Következmény Egy olyan nem megoldható feladatra, amelyre (1) nem teljesül, (3) viszont igen, a diagonális hibát csökkentő eljárás addig tud csökkenteni, míg az egyik hibavektor nullvektorrá nem válik. Ekkor a másik hibavektorban maradt többlet már nem csökkenthető, és nem lett volna az semmilyen más elrendezés esetén sem (ld. i) bizonyítása). Tehát azokra a feladatokra, melyekre (3) teljesül, (1) pedig nem, az MGrow algoritmus optimális megoldást ad. Az optimális alatt azt értjük, hogy semmilyen más algoritmussal sem lehetett volna olyan (realizálható) bajnoságot szervezni, amelyre az összes csapat által feleslegesen rúgott és kapott gólok száma ennél kisebb.

A továbbiakban az kerül vizsgálódásunk középpontjába, hogyan javítható meg a minimális változtatással egy olyan pont- és vesztővektor, amelyekre (1) (3) nem teljesül.

Legjobb közelítés kis mértékű hibákra

Tekintsük a $D = d_1, \dots, d_n$, $L = l_1, \dots, l_n$ nem realizálható pont- és vesztővektorokat. Ezekre tehát nem teljesül az (1) és/vagy (3) feltétel, tehát azt írhatjuk, hogy

$$\sum_{i=1}^{n} d_i = \sum_{i=1}^{n} l_i + F \tag{6}$$

$$\sum_{\substack{i\neq 1 \\ i\neq M}}^{n} d_i + l_i = d_M + l_M - G \tag{7}$$

ahol F tetszőleges, G nemnegatív egész. Szeretnénk megtalálni azokat a $D^{'}, L^{'}$ vektorokat, amelyek realizálhatók, valamint

$$E(D^{'}, L^{'}) := \|D - D^{'}\|_1 + \|L - L^{'}\|_1$$

hibafüggvény minimális, abban az értelemben, hogy nem léteznek olyan $D^{''}, L''$ vektorok, melyekre igaz (1) és (3), valamint $E(D^{''}, L^{''}) < E(D^{'}, L^{'})$.

Állítsunk tehát elő a $D^{'}, L'$vektorokat a D, L vektorokból azzal a minimális változtatással, amellyel (6), (7) helyrejön, azaz $D^{'}, L'$ vektorok eleget tesznek az (1), (3) állításoknak! A (7) megjavításához csak a d_M, l_M elemek csökkentése szükséges, továbbá a (6) megjavításához is használhatjuk azt a stratégiát, hogy - F előjelétől függően - csökkentjük D vagy L vektor valamely elemét. Nyilvánvaló, hogy a legkisebb bajt akkor okozzuk, ha a d_M, l_M elemeket csökkentjük, máskülönben elronthatjuk (3)-at! Ezért természetes ötletnek tűnik, hogy csak d_M, l_M elemeket

változtatjuk, ez viszont megszorításokat tesz szükségessé az F, G értékeire vonatkozóan. A továbbiakban megköveteljük, hogy

$$\max(|F|, G+1) < \min(d_M, l_M) \tag{8}$$

$$\max(|F|, G+1) < (d_M + l_M) - \max_{i \neq M}(d_i - l_i) \tag{9}$$

A megszorításokról, azok szükségességéről, gyengíthetőségéről, elhagyhatóságáról észrevételeket az utólagos megjegyzésekben teszünk.

Legyen $F > 0$, és

$$d_M^{'} = d_M - F$$
$$l_M^{'} = l_M$$

ezt beírva (6)-ba

$$\sum_{i=1}^{n} d_i = \sum_{i=1}^{n} d_i^{'} + F = \sum_{i=1}^{n} l_i^{'} + F = \sum_{i=1}^{n} l_i + F$$

ebből pedig

$$\sum_{i=1}^{n} d_i^{'} = \sum_{i=1}^{n} l_i^{'}$$

tehát $D^{'}, L^{'}$ (1)-et teljesíti. Továbbá (7) miatt

$$\sum_{\substack{i=1 \\ i \neq M}}^{n} d_i^{'} + l_i^{'} \geq d_M^{'} + l_M^{'} - G + F \quad (F \geq G)$$

Kaptunk tehát $F \geq G \geq 0$ esetén egy olyan helyettesítést, amely eleget tesz (1), (3)-nak. Vizsgáljuk most azt az esetet, amikor $|F| \geq G \geq 0$ és $F \leq 0$. Ekkor legyen

$$d_M^{'} = d_M$$
$$l_M^{'} = l_M - |F|$$

Az előbbiek alapján látható, hogy

$$\sum_{i=1}^{n} d_i = \sum_{i=1}^{n} d_i^{'} = \sum_{i=1}^{n} l_i^{'} = \sum_{i=1}^{n} l_i - |F|$$

valamint

$$\sum_{\substack{i=1 \\ i \neq M}}^{n} d_i^{'} + l_i^{'} \geq d_M^{'} + l_M^{'} - G + |F| \quad (|F| \geq G$$

A helyettesítésünk megvan tehát $|F| \geq G \geq 0$ esetre!

Tekintsük most a $G > |F| \geq 0$ esetet! Ekkor pl. $F > 0$ esetén

$$d_M^{'} = d_M - \mathrm{F} - G_0$$
$$l_M^{'} = l_M - G_0$$

ahol

$$G_0 = \left\lceil \frac{|F| - G}{2} \right\rceil$$

És az előzőek alapján felírhatók a javított egyenletek

$$\sum_{i=1}^{n} d_i = \sum_{i=1}^{n} d_i^{'} + F + G_0 = \sum_{i=1}^{n} l_i^{'} + F + G_0 = \sum_{i=1}^{n} l_i + F$$

$$\sum_{\substack{i=1 \\ i \neq M}}^{n} d_i^{'} + l_i^{'} \geq d_M^{'} + l_M^{'} - G + 2G_0 + F \qquad (G > F)$$

hiszen G_0 definíciója miatt $2G_0$ legalább G, legfeljebb $G + 1$. A fentiekből már analóg módon következik, hogy $G > F$ és $F \leq 0$ esetén a

$$d_M^{'} = d_M - G_0$$
$$l_M^{'} = l_M - |F| - G_0$$

helyettesítés is kielégíti a (1), (3) összefüggéseket, a teljesség kedvéért

$$\sum_{i=1}^{n} d_i = \sum_{i=1}^{n} d_i^{'} + G_0 = \sum_{i=1}^{n} l_i^{'} + G_0 = \sum_{i=1}^{n} l_i - |F|$$

$$\sum_{\substack{i=1 \\ i \neq M}}^{n} d_i^{'} + l_i^{'} \geq d_M^{'} + l_M^{'} - G + 2G_0 + |F| \qquad (G > |F|)$$

Az előzőekben egy olyan módszert adtunk, amely $D^{'}, L'$ vektorokat állít elő a D, L vektorokból, amelyekre

$$E(D^{'}, L^{'}) = \begin{cases} F & ha\ |F| \geq G \\ G, & ha\ |F| < G\ és G_0 \quad páros \\ G + 1, & ha\ |F| < G\ és\ G_0\ páratlan \end{cases}$$

(10)

Észrevételeinket a *RecoverScores* algoritmus összegzi.

$RecoverScores(D, L)$
$M = argmax(D + L)$
$F = sum(D) - sum(L)$
$G = 2(d_M + l_M) - sum(D + L)$
$\boldsymbol{if}\ F > 0$
  $d_M = d_M - F$
$\boldsymbol{elseif}\ F < 0$
  $l_M = l_M + F$
$\boldsymbol{if}\ G > F$
  $G_0 = \left\lceil \frac{|F| - G}{2} \right\rceil$
  $d_M = d_M - G_0$
  $l_M = l_M - G_0$

A *RecoverScores* algoritmus tehát egy olyan transzformációt hajt végre a D, L vektorokon, amellyel már teljesítik az (1), (3) feltételeket, valamint a hibájuk (10). A következőkben megmutatjuk, hogy ez a transzformáció minimális.

3. Tétel Tegyük fel, hogy D, L pont- és vesztővektorok, amelyekre fennáll (6), (7), az ott definiált F, G konstansok pedig a kielégítik a (8), (9) feltételeket. Ekkor ha D', L' vektorokat a *RecoverScores* algoritmussal kaptuk a D, L vektorokból, akkor nem léteznek D'', L'' realizálható vektorok, amelyekre $E(D', L') > E(D'', L'')$.

Bizonyítás A bizonyítás alapgondolatául az az észrevétel szolgál, hogy bármely realizálható rendszer átvihető egy tetszőleges másik realizálható rendszerbe véges sok elemcserével (Ellenkező esetben azt mondhatnánk szemléletesen, hogy ha kapunk egy gráfot, egy radírt, és egy ceruzát, és az a feladatunk, hogy átrajzoljuk a gráfot egy ugyanolyan csúcszámú tetszőleges gráffá, akkor azt nem tudnánk megtenni...). Emiatt feltehetjük, hogy D'', L'' vektorokat egy-egy gól hozzávételével, vagy törlésével kapjuk D', L' vektorokból. Ha ugyanis egy tetszőleges elempár transzformációjára sem csökken az újonnan kapott vektorok hibája, akkor véges sokra sem fog. Továbbá a D', L' vektorok realizálhatók, azaz teljesítik (1)-et, ebből viszont az következik, hogy ha egy gólt hozzáveszünk D'-höz, akkor egyet L'-höz is hozzá kell vennünk, ha az egyikből törlünk, akkor a másikból is kell, csak így maradhat érvényben (1). Tegyük fel tehát, hogy valamely j, k indexekre

$$d_j'' = d_j' \pm 1$$
$$l_k'' = l_k' \pm 1$$

A *RecoverScores* algoritmus csak a d_M, l_M elemeket csökkenti, ezért

$$\sum_{\substack{i=1 \\ i \neq M}}^{n} d_i = \sum_{\substack{i=1 \\ i \neq M}}^{n} d_i'$$

$$\sum_{\substack{i=1 \\ i\neq M}}^{n} l_i = \sum_{\substack{i=1 \\ i\neq M}}^{n} l_i^{'}$$

amiből viszont az következik, hogy

$$\sum_{\substack{i=1 \\ i\neq M}}^{n} |d_i - d_i^{'}| + |l_i - l_i^{'}| = 0 \tag{11}$$

tehát

$$E(D^{'}, L^{'}) = |d_M - d_M^{'}| + |l_M - l_M^{'}| \tag{12}$$

Három esetre választjuk szét a problémát:

1) $j \neq M,\ \ k \neq M$

(11)-ből következik, hogy bármely $j, k \neq M$ esetén $d_j = d_j^{'}$, $l_k = l_k^{'}$. Ebből pedig következik, hogy

$$|d_j - d_j^{''}| = 1, \quad |l_k = l_k^{''}| = 1$$

tehát

$$\sum_{\substack{i=1 \\ i\neq M}}^{n} |d_i - d_i^{''}| + |l_i - l_i^{''}| = 2$$

mivel azonban $d_M^{'} = d_M^{''}$ és $l_M^{'} = l_M^{''}$

$$E(D^{'}, L^{'}) + 2 = E(D^{''}, L^{''})$$

2) $j \neq M,\ \ k = M\ \ (vagy\ \ j = M,\ \ k \neq M)$

Ekkor az 1) pontban leírtak alapján azt mondhatjuk, hogy

$$\left|d_j - d_j^{''}\right| = 1$$

miatt,valamint figyelembe véve, hogy $l_M^{''}$ legfeljebb egyet javíthat a hibán

$$E(D^{'}, L^{'}) \leq E(D^{''}, L^{''})$$

A feladat nyilvánvaló szimmetriája miatt következik a fordított eset is.

3) $j = M,\ \ k = M$

Ebben az esetben azt kell felismernünk, hogy a *RecoverScores* algoritmus konstrukciójából következik, hogy

$$d_M \geq d_M^{'},\ \ l_M \geq l_M^{'}$$

Ez viszont azt jelenti, hogy a $d_M^{''}$, $l_M^{''}$ elemek csak úgy csökkenthetik a hibát, ha

$$d_M^{''} > d_M^{'}, \;\; l_M^{''} > l_M^{'}$$

Feltehetjük tehát, hogy

$$d_M^{''} = d_M^{'} + 1$$
$$l_M^{''} = l_M^{'} + 1$$

Ekkor (10) és (7) szerint

$$\sum_{\substack{i=1 \\ i \neq M}}^{n} d_i^{'} + l_i^{'} = d_M + l_M - H$$

teljesül, ahol

$$H = \{F, G, G + 1\}$$

Ekkor viszont bármelyik esetben

$$\sum_{\substack{i=1 \\ i \neq M}}^{n} d_i^{''} + l_i^{''} = \sum_{\substack{i=1 \\ i \neq M}}^{n} d_i^{'} + l_i^{'} < d_M + l_M + 2 - H$$

Tehát $D^{''}$, L'' megsérti (3)-at..

■

Azt láttuk be tehát, hogy a *RecoverScores* algoritmus olyan realizálható $D^{'}$, L' vektorokat állít elő (6-9) fennállása esetén, amelyeknél nincs jobb realizálható közelítése D, L vektoroknak.

Összegzés

A fentiekben a célkitűzésben megfogalmazott problémának megadtuk egy mátrixalgebrai modelljét, ebben definiáltunk egy algoritmust, amely vagy megoldja a feladatot, vagy egy úgynevezett diagonális hibához vezet, amelynek kitranszformálásához elégségesek a Hakimi tétel feltevései. Ebből arra a következtetésre jutottunk, hogy az MGrow algoritmus megoldja a feladatot, amennyiben az megoldható. A feladat nem megoldhatósága esetén pedig bizonyos megszorítások mellett tudtunk adni egy módszert, amely a pont- és vesztővektort olyan vektorokká alakítja, amelyekre az esetünkben megfogalmazott hiba minimális. Szemléletesen ez a legkevesebb gól, amelyet el kell vennünk 1 csapattól ahhoz, hogy a feladat megoldható legyen, ha tehát az MGrow algoritmussal a transzformált vektorokra adott megoldásvektorokban visszaadjuk ezeknek a csapatoknak az elvett gólokat, akkor a feladat legjobb közelítését kapjuk a bevezetésben megfogalmazott szemlélet alapján.

Utólagos megjegyzések

1) A *RecoverScores* lineáris időben elvégezhető, a *Growing* algoritmus futásideje $O(n^2)$, amely jónak mondható, a *DecDiagErr* algoritmusé viszont $O(n^4)$, mivel végig kell vizsgálnia az összes lehetséges elempárt. A *DecDiagErr* algoritmus gyorsítására azért nem fektettünk nagyobb hangsúlyt, mert egyrészt első közelítésben a feladat megoldása volt a célkitűzés, másrészt a kísérleti tapasztalatok alapján a *DecDiagErr* vagy nem fut le, mert a *Growing* önmagában megoldja a problémát, vagy lefut, de ekkor csak kevésszer, és ekkor is, ha az állapottér elegendően nagy, és a vektorok "kiegyensúlyozottak" akkor átlagosan konstans időben található pozitív elempár. Felmerül azonban annak a lehetősége, hogy az időbonyolultság egy részét tárbonyolultságra terhelve már a *Growing* futása közben regisztráljuk a diagonális hiba csökkentésére alkalmas elempárokat, ez további vizsgálódás tárgyát képezheti.

2) A *RecoverScores* végrehajtásához azért van szükség a (8) feltételre, hogy nehogy nagyobb értéket vonjunk le a d_M, l_M egyikéből, hogy az negatívvá váljon. Nyilvánvalóan ez a feltétel gyengíthető, ha külön kezeljük azt a két esetet, amikor F negatív, vagy pozitív. A (9) feltételre pedig azt mondja ki, hogy az az érték, amit levonunk, nem lehet akkora, hogy az nagyobb legyen, mint a legnagyobb elempár és a második legnagyobb különbsége. Ugyanis ha ezt megengednénk, akkor előfordulhatna, hogy megváltozik a legnagyobb elempár indexe, és ezzel megsértjük a (3)-at. Ez a feltétel is gyengíthető. ha megvizsgáljuk, hogy mennyit lehet levonni úgy, hogy a második legnagyobb elempár is teljesítse (3)-at. A feltételek elhagyhatósága csak az algoritmus módosításokkal válhat lehetségessé. További meggondolások azonban feltehetőleg lehetségessé teszik vagy az effajta feltételek teljes elhagyását, vagy a fentiekben leírtaknál nagyobb mértékű csökkentését.

Irodalomjegyzék

...