Alkalmazott Matematikai Lapok **24** (2007), 163-185.

Új monoton jellegű szimplex algoritmusok elemzése megengedettségi feladatok esetén

Bilen Filiz, Csizmadia Zsolt és Illés Tibor

Anstreicher és Terlaky (1994) monoton szimplex algoritmusának megfogalmazzuk egy olyan új variánsát megengedettségi feladatokra, amelynek a lépésszámát egy a szokásostól gyengébb nem degeneráltsági feltevés mellett $m\Delta$ adja meg, ahol Δ a feladat adataiból kiszámítható konstans, m a feltételek száma. A Δ konstans, a feladat leírásához szükséges számítógépes tárigénynek – az adatok bithosszának – egy polinomjával nem mindig korlátozható.

Erősen degenerált feladatok esetén a végességet olyan index választási szabályok segítségével biztosítjuk, melyek nagyfokú flexibilitást tesznek lehetővé a pivot pozíció meghatározásakor, elősegítve a numerikusan nehéz pivot pozíciók elkerülését. Az algoritmusunk végességének a bizonyításához olyan általános keretet használunk – **s**-monoton pivotálási szabályok tulajdonságait –, amely magában foglalja a szokásos minimál indexes szabály mellett, a "Last In First Out" (LIFO) és a "most-often-selected-variable" (MOSV) szabályt is.

Az **s**-monoton pivotálási szabályok fogalma jól illeszkedik a lineáris programozási pivot algoritmusok témaköréhez is. A megengedettségi feladatok egy pivot algoritmusára (MBU-szimplex módszer új variánsára) bemutatott, az **s**-monoton pivot szabály fogalmát használó végesség bizonyítás természetes módon vihető át több lineáris programozási pivot algoritmus végességének az igazolására is.

Az algoritmus lépésszámának elemzése az eredeti monoton szimplex algoritmus és a primál szimplex algoritmus első, illetve második fázisára, továbbá az úgynevezett exterior-point simplex algoritmusra is átvihető nem degeneráltsági feltevés mellett.

Kulcsszavak: megengedettségi feladat, monoton szimplex algoritmusok, degeneráltság, lineáris programozás, index választási szabályok.

Mathematics Subject Classification 2000: **90C20**.

1. Bevezetés

A lineáris egyenlőtlenségrendszerek vizsgálata és megoldása, amely a XX. század közepén fejlődésnek indult *lineáris optimalizálás* egyik fontos problémája, ma is több szempontból érdekes kutatási terület.

Tekintsük az alábbi lineáris egyenlőtlenségrendszert, az

$$A\mathbf{x} = \mathbf{b}, \quad \mathbf{x} \geq \mathbf{0} \tag{1}$$

*megengedettségi feladat*ot, ahol $A \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m$. Az általánosság korlátozása nélkül feltehetjük, hogy $rang(A) = m$, ugyanis a lineáris feltételrendszer megoldhatóságát ellenőrizhetjük, illetve a redundáns lineáris feltételeket elhagyhatjuk a

Gauss-Jordán elimináció segítségével. A megengedettségi feladatok legismertebb megoldatlan kérdése az, hogy létezik-e erősen polinomiális pivot algoritmus az (1) probléma megoldására. Mi ennél egy könnyebb kérdésre adunk választ: megmutatjuk, hogy az *Anstreicher-Terlaky monoton szimplex algoritmus*nak megfogalmazható egy olyan variánsa az (1) feladatra, amelynek a lépésszámát az $m\Delta$ adja meg, ahol Δ a feladat adataiból kiszámítható konstans. Annak ellenére, hogy a Δ konstans, a feladat leírásához szükséges számítógépes tárigénynek – az adatok bithosszának – egy polinomjával nem mindig korlátozható, és egy speciális *nem degeneráltsági feltevés*sel is élnünk kell az elemzésünk során, eredményünk mégis érdekes, hiszen hasonló nem ismert a szakirodalomból. Sőt, azt is mondhatjuk, hogy a lineáris programozási feladatra, a Dantzig-féle szimplex algoritmus [7, 8] felfedezése óta sem ismert olyan pivot algoritmus, amelynek a lépésszám becslését az algoritmus elemzéséből nyernénk. A lineáris programozási feladat megoldására definiált pivot algoritmusok nagy többségéről megmutatták, hogy a *Klee-Minty feladat*on [15], vagy valamely azzal nagyon hasonló struktúrájú exponenciális ellenpéldán, egy adott bázisból indulva exponenciálisan sok bázis csere segítségével állítják elő az optimális megoldást. A pivot algoritmusok többségénél, a végesség bizonyításán kívül, ez az egyetlen elméleti jellegű támpont az algoritmus hatékonyságára általános lineáris programozási feladat esetén. Másfelől, gyakorlati feladatok megoldása során nyert tapasztalat ([16], 19. oldal), azt mutatja, hogy a lineáris programozási feladatok megoldásához, a szimplex módszerrel $O(m)$ iterációra van szükség, ahol m a feltételek száma. A gyakorlati tapasztalatot számos valószínűségi modellen alapuló vizsgálat támasztja alá, azaz igazolták, hogy számos speciális feladatosztályon és valószínűségi feltételek mellett a szimplex algoritmus várható lépésszáma polinomiális. Ezek közül a vizsgálatok közül – a teljesség igénye nélkül – megemlítjük Borgwardt [4] és Todd [20] műveit. A téma iránt érdeklődők Fukuda és Terlaky [9] cikkéből nyerhetnek további információkat a pivot algoritmusok várható-, átlagos-, illetve legrosszabb lépésszámáról, bőséges irodalmi hivatkozás mellett.

Karmarkar híres cikke [12] után a lineáris optimalizálás kutatás homlokterébe a belsőpontos algoritmusok fejlesztése, új algoritmusok és elemzési módszerek felfedezése került, teljesen háttérbe szorítva a pivot algoritmusok elméleti vizsgálatát, új algoritmusok definiálását.[1] Ritka kivétel Anstreicher és Terlaky [2] *monoton szimplex algoritmusa*, amely számos igen jelentős újítást tartalmaz a korábbi szimplex (pivot) típusú algoritmusokhoz viszonyítva. Az algoritmus főbb jellemzői: (i) a duál változók megengedettségét monoton módon állítja be; (ii) nem feltétlenül őrzi meg az iterációk során a primál megengedettséget; (iii) két lehetséges pivot pozíciót hasonlít össze; (iv) degenerált feladatok esetén szükséges valamilyen degeneráltság elleni technikát, pl. minimál indexes szabályt, alkalmazni; (v) amennyiben elrontja a primál megengedettséget, akkor az ún. vezér változón való pivotáláskor, a primál megengedettség visszaáll.

[1] A pivot algoritmusok kifejlesztésének hőskorát lezáró Klee és Minty [15] dolgozata után megjelent pivot algoritmusokat foglalták össze Terlaky és Zhang [19].

Dolgozatunkban Anstreicher és Terlaky által megfogalmazott monoton szimplex módszer [2] alapötletét felhasználva definiáltuk algoritmusunkat megengedettségi feladat megoldására. Algoritmusunk rendelkezik azzal a tulajdonsággal, hogy ha valamelyik változó a számítási eljárás során megengedetté vált, akkor ezt a megengedettséget mindvégig megőrzi, tehát a feladat megoldása – a változók megengedettségének az elérése – monoton módon történik. Algoritmusunk duál változata analóg módon definiálható.

Pivot pozíció választásunk eltér a megengedettségi (lineáris programozási) feladatoknál megszokottaktól. Részletesebben ezzel a 2. fejezetben foglalkozunk.

Algoritmusunkról – egy nem degeneráltsági feltétel mellett – kimutatjuk, hogy $m\Delta$ iterációra van szükségünk a feladat megoldásához. Tekintettel arra, hogy a nem degeneráltsági feltétel teljesülését nem lehet a priori bizonyítani, így a 4. fejezetben egy általános módszertan segítségével számos flexibilis index választási szabályról bizonyítjuk, hogy algoritmusunk végességét biztosítja. Ezen index választási szabályok tartalmazzák a most-often-selected-variable, LIFO és a szokásos minimál index választási szabályt is. A flexibilis index választási (pl. LIFO) szabályok számos esetben lehetővé teszik numerikusan instabil pivot pozíciók elkerülését.

Algoritmusunk geometriai tulajdonságait az alábbiakban foglaljuk össze. Hasonlóan a criss-cross módszerekhez, az algoritmus a megengedettségi feladat feltételei által alkotott metszésrendszer szomszédos metszéspontjain – nem megengedett bázisokon – halad keresztül. Ha valamely iteráció során az aktuális bázismegoldás egy adott feltétel által meghatározott hipersík megengedett oldalára került, akkor a további iterációkban előállított bázismegoldások sem fognak ezen feltétel nem megengedett oldalára kerülni.

Dolgozatunk 2. fejezetében új pivot pozíció választási szabályt és lokális degeneráltsági fogalmat vezetünk be. A 3. fejezetben megfogalmazzuk az MBU típusú primál szimplex algoritmusunkat, és igazoljuk néhány tulajdonságát. Lokálisan nem erősen degenerált pivot sorozatok esetén kiszámítjuk az algoritmus lépésszámának egy felső korlátját. A 4. fejezetben degeneráltsági megkötés nélkül igazoljuk az MBU típusú szimplex algoritmus végességét.

1.1. Jelölések

Dolgozatunkban a következő jelölésrendszert használjuk. A skalárokat és indexeket kis latin betűk, a vektorokat vastag kis latin betűk, a mátrixokat nagy latin betűk, míg az indexhalmazokat kalligrafikus nagy betűk jelölik.

Egy $\mathcal{G} \subseteq \mathcal{I} := \{1, 2, \ldots, n\}$ indexhalmaz és $\mathbf{v} \in \mathbb{R}^n$ vektor esetén jelölje $\mathbf{v}_\mathcal{G}$ a $\mathbf{v}$ $\mathcal{G}$ által indexelt részvektorát, vagyis $(\mathbf{v}_\mathcal{G})_i := v_i$, $i \in \mathcal{G}$. Egy $T \in \mathbb{R}^{m\times n}$ mátrix, illetve az $\mathcal{F} \subseteq \mathcal{J} := \{1, \ldots, m\}$ és $\mathcal{G} \subseteq \mathcal{I}$ indexhalmazok esetén jelölje $T_{\mathcal{F}\mathcal{G}}$ azt a részmátrixot, amelynek sorai $\mathcal{F}$, illetve oszlopai a $\mathcal{G}$ indexekkel vannak indexelve.

Legyen $B \in \mathbb{R}^{m\times m}$ a teljes sorrangú A mátrix egy nemszinguláris négyzetes részmátrixa, illetve N az A mátrix többi – maradék – oszlopából álló mátrix. Ekkor a B mátrixot a feladat egy bázisának nevezzük, és az $\mathcal{I}_B \subset \mathcal{I}$, illetve $\mathcal{I}_N \subset \mathcal{I}$ indexhalmazokkal a bázisbeli és bázison kívüli változók halmazát

jelöljük. Ha szükséges hangsúlyozni, hogy az indexhalmazok a k. iterációhoz tartoznak, akkor az $\mathcal{I}_{B_k}$ és $\mathcal{I}_{N_k}$ jelöléssel élünk. Ismeretes, hogy ekkor a bázishoz tartozó megoldást a $\bar{\mathbf{b}} := \bar{\mathbf{x}}_{\mathcal{I}_B} := B^{-1}\mathbf{b}$ formulával lehet meghatározni, és $T = B^{-1}N \in \mathbb{R}^{m\times(n-m)}$ adja meg a bázishoz tartozó rövid pivot táblát. Definíció szerint $\bar{\mathbf{x}}_{\mathcal{I}_N} := \mathbf{0}$.

Egy rögzített B bázis esetén a következő partíciókat fogjuk használni:

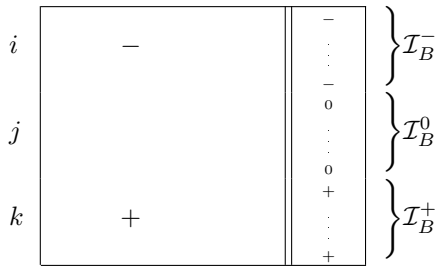
$$\mathcal{I}_B = \mathcal{I}_B^+ \cup \mathcal{I}_B^0 \cup \mathcal{I}_B^-,$$

ahol

$$\mathcal{I}_B^+ = \{i \in \mathcal{I}_B : \bar{x}_i > 0\}, \quad \mathcal{I}_B^0 = \{i \in \mathcal{I}_B : \bar{x}_i = 0\} \quad \text{és} \quad \mathcal{I}_B^- = \{i \in \mathcal{I}_B : \bar{x}_i < 0\}.$$

Időnként, ha csupán a megengedettségét szeretnénk hangsúlyozni bizonyos bázisváltozóknak, akkor az $\mathcal{I}_B^{\oplus} = \mathcal{I}_B^+ \cup \mathcal{I}_B^0$ jelölést használjuk.

A partíciónak megfelelő bázistáblát az 1. ábra mutatja.



1. ábra. A bázis partícionálása.

2. Pivot pozíció választása: néhány új fogalom

A legtöbb szakirodalomból ismert pivot módszer primál megengedett sorban csak pozitív elemen végez báziscserét (pl. primál szimplex), illetve primál nem megengedett sor esetén negatív elemen végez báziscserét (pl. duál szimplex) esetleg a kétféle stratégiát ötvözi (pl. criss-cross módszer). Ezt felismerve Fukuda és Terlaky [9, 10], bevezették az úgynevezett elfogadható báziscsere fogalmát. A mi algoritmusunk ennél általánosabb báziscseréket is fog végezni. A pozitív értékű változó sorában pozitív elemen végzett báziscsere felfogható az elfogadható báziscserék duál oldali pivotjának megengedettségi feladatokra vonatkozó megfelelőjeként. A degeneráltság kezelésekor a degenerált sorokban (vagyis ahol a jobboldal nulla) pozitív, illetve negatív elemen is végezhetünk báziscserét.

2.1. Definíció. Egy adott B bázis és T pivot tábla esetén a t_{ij} elemet *általánosított elfogadható* pivot elemnek nevezzük, ha

1. $i \in \mathcal{I}_B^-$ és $t_{ij} < 0$, vagy
2. $i \in \mathcal{I}_B^+$ és $t_{ij} > 0$, vagy
3. $i \in \mathcal{I}_B^0$ és $t_{ij} \neq 0$.

Algoritmusunkban és a későbbiekben bevezetett részfeladatok megoldásakor általánosított elfogadható báziscseréket használunk. Az algoritmus megfogalmazásához szükségünk lesz a degeneráltság fogalmának a finomítására. Vezessük be $s \in \mathcal{I}_N$ esetén a

$$\mathcal{K}_s = \left\{ i \in \mathcal{I}_B^0 : t_{is} > 0 \right\}$$

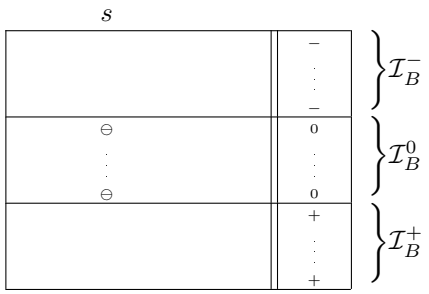
jelölést.

2.2. Definíció. Egy B bázist *nem degenerált*nak nevezünk, ha $\bar{\mathbf{x}}_B$ egyik komponense sem nulla. A B bázis *degenerált*, ha az $\bar{\mathbf{x}}_B$ bázismegoldásnak van nulla komponense.

A degeneráltság jelensége az aktuális bázistól is függ oly módon, hogy az adott bázismegoldás előállítható a dimenziójánál kevesebb bázisbeli elem kombinációjaként. A későbbiekben a degeneráltság két fajtáját fogjuk megkülönböztetni.

2.3. Definíció. A B degenerált bázist *lokálisan gyengén degenerált*nak nevezzük az $s \in \mathcal{I}_N$ indexre vonatkozóan, ha $\mathcal{K}_s = \emptyset$, illetve *lokálisan erősen degenerált*nak nevezzük az s indexre vonatkozóan, ha $\mathcal{K}_s \neq \emptyset$.

Tegyük fel hogy egy adott B bázis esetén eldöntöttük, hogy a bázistábla s. oszlopában kívánunk nem degenerált sorban általánosított elfogadható báziscserét végezni olymódon, hogy a degenerált változók ne váljanak nem megengedetté. Figyeljük meg, hogy ez pontosan akkor lehetséges, ha $\mathcal{K}_s = \emptyset$. Egy ilyen, az s indexre nézve lokálisan gyengén degenerált táblát mutat a 2. ábra.



2. ábra. Lokálisan gyengén degenerált pivot tábla.

2.4. Definíció. Legyen adott egy B bázis és egy $r \in \mathcal{I}_B^-$ index. A megfelelő bázistábla egy t_{ij} elemén végzett báziscserét x_r *növelő*nek nevezünk, ha

1. a t_{ij} elem általánosított elfogadható pivot elem,
2. $\mathcal{I}_B^{\oplus} \subseteq \mathcal{I}_{B'}^{\oplus}$ és
3. $\hat{x}_r > \bar{x}_r$ teljesül,

ahol $\bar{\mathbf{x}}$ az aktuális megoldást, B' az új bázist és $\hat{\mathbf{x}}$ az új bázismegoldást jelöli.

Az x_r növelő báziscserére mutatunk egy egyszerű példát.

2.1. Példa.

	x_4	x_5	
x_1	0	-1	0
x_2	**1**	-1	1
x_3	-1	0	-2

A tábla egyetlen nem megengedett változójának a sorában egy negatív elem található, így a vezérváltozó az x_3, és az x_4 oszlopában végzünk báziscserét. A vezérváltozó sorában a báziscsere nem lenne növelő, mert az x_2 változó nem megenegedetté válna, ellentmondva a definíció 2. feltételének. Mivel a tábla gyengén degenerált ($\mathcal{K}_4 = \emptyset$), így van x_3 növelő báziscsere, azaz az x_2 távozik és az x_4 belép a bázisba.

	x_2	x_5	
x_1	0	-1	0
x_4	1	-1	1
x_3	1	-1	-1

Célunk x_r növelő báziscseréket végrehajtó algoritmust megfogalmazni. Ismereteink szerint a szakirodalomból egyetlen x_r növelő báziscseréket végző algoritmus ismert, Anstreicher és Terlaky [2] általános, primál, illetve duál megengedett megoldásból induló lineáris programozási feladatokra megfogalmazott primál, illetve duál oldali algoritmusa.

Sajnos léteznek olyan bázistáblák, melyeken nincs x_r növelő báziscsere, ahogyan azt a 3. ábra példája is mutatja. A probléma egy megengedett megoldása az $x_1 = x_2 = 0, x_3 = x_4 = 1$, mégsincs az egyetlen negatív értékű sorhoz növelő báziscsere.

	x_3	x_4	
x_1	-1	0	-1
x_2	1	-1	0

3. ábra. Lokálisan erősen degenerált tábla, melyen nincs x_r növelő báziscsere, ahol $r = 1$.

3. Az MBU szimplex típusú algoritmus megengedettségi feladatokra

Ebben a fejezetben megfogalmazzuk az (1) primál megengedettségi feladat megoldására szolgáló MBU szimplex típusú algoritmusunkat, mely Anstreicher és Terlaky [2] algoritmusához hasonló tulajdonságokkal rendelkezik. Ez utóbbi algoritmust általános lineáris programozási feladatra fogalmazták meg, és fő jellemzője, hogy duál megengedett bázisból indulva monoton módon állítja be a primál változók megengedettségét, miközben a duál megengedettség elromolhat, azonban a kiválasztott primál változó megengedettségének elérésekor az aktuális megoldás ismét duál megengedetté válik.

3.1. Algoritmus. MBU szimplex algoritmus megengedettségi feladatokra

Begin
Bemenő adatok: $A \in \mathbb{R}^{m\times n}, \mathbf{b} \in \mathbb{R}^m$,
$B \in \mathbb{R}^{m\times m}$ az A reguláris részmátrixa.
$T := B^{-1}A, \bar{\mathbf{b}} := B^{-1}\mathbf{b}, \mathcal{I}_B^- := \{i \in \mathcal{J} \mid \bar{b}_i < 0\}$.
While $(\mathcal{I}_B^- \neq \emptyset)$ **do**
Legyen $r \in \mathcal{I}_B^-$ tetszőleges (vezérváltozó), $rDone := false$.
While $(rDone = false)$ **do**
$\mathcal{J}_r^- := \{j \in \mathcal{I}_N \mid t_{rj} < 0\}$.
If $(\mathcal{J}_r^- = \emptyset)$ **then**
nincs megengedett megoldás, **Return**
Endif
Legyen $s \in \mathcal{J}_r^-$ tetszőleges, $\mathcal{K}_s := \{i \in \mathcal{I}_B^0 \mid t_{is} > 0\}$.
If $(\mathcal{K}_s \neq \emptyset)$ **then**
$(T, l) = DegEljárás(T, \mathcal{I}_B^0, r)$.
If $(l \in \mathcal{I}_N)$ **then**
$s := l$.
else
nincs megengedett megoldás, **Return**
Endif
Endif
$\Theta_1 := \frac{\bar{b}_r}{t_{rs}}$, $\Theta_2 := \min\left\{\frac{\bar{b}_k}{t_{ks}} \mid k \in \mathcal{I}_B^{\oplus}, t_{ks} > 0\right\}$.
If $(\Theta_1 \leq \Theta_2)$ **then**
pivotálás a t_{rs} elemen, $rDone = true$.
else
$q := \arg_k \min\left\{\frac{\bar{b}_k}{t_{ks}} \mid k \in \mathcal{I}_B^+, t_{ks} > 0\right\}$, pivotálás a t_{qs} elemen.
Endif
Endwhile
$\mathcal{I}_B^- := \{i \mid \bar{b}_i < 0\}$.
Endwhile
$\bar{\mathbf{x}}$ megengedett megoldás.
End

Algoritmusunk bizonyos értelemben az Anstreicher és Terlaky duál MBU-szimplex algoritmusához áll közel, hiszen esetünkben is a primál megengedett változók száma növekszik monoton módon.

Algoritmusunk a monotonitás elérése érdekében a következő szemléletes képet követi. Kiválaszt egy, az adott megoldásban nem megengedett változót, melyet az elkövetkező iterációk során megengedetté kíván tenni. Mindaddig, míg ezen változó megengedetté nem válik, ezt a változót *vezérváltozónak* fogjuk nevezni. Az x_r vezérváltozó megengedettségét monoton módon javítjuk x_r növelő báziscserék segítségével. Látni fogjuk, ha az algoritmus csupán nem degenerált, vagy lokálisan gyengén degenerált bázistáblákon halad, akkor ez mindig megtehető.

Az (1) feladatra megfogalmazott algoritmusunk pseudo kódját a 3.1. Algoritmus, míg annak folyamatábráját az 4. ábra foglalja össze. Az algoritmus lokálisan erősen degenerált bázisok esetén kénytelen degeneráltság elleni eljárást alkalmazni. A degeneráltság elleni eljárást a 4. fejezetben tárgyaljuk, az algoritmusban egyenlőre csak *DegEljárás*ként hivatkozunk rá.

Az x_r növelő báziscsere megtalálásának érdekében a báziscserére kiválasztott oszlopban két hányadosteszt kell végezni, melyek értékének viszonyából állapítható meg, hogy végezhető-e báziscsere a vezérváltozó sorában, vagy általánosabb x_r növelő báziscserére van-e szükség.

Az algoritmus leírásában a két hányadosteszt értékét Θ_1-gyel és Θ_2-vel jelöltük (3.1. Algoritmus). A definíciójukból könnyen látható, hogy $\Theta_1 > 0$ és $\Theta_2 \geq 0$, továbbá ha a megfelelő bázis lokálisan nem erősen degenerált, akkor $\Theta_2 > 0$. Az algoritmus belső ciklusának célja a vezérváltozó megengedettségének előállítása.

Megmutatjuk, hogy az algoritmus kizárólag x_r növelő báziscseréket végez, ha az adott B bázis a kiválasztott nem bázis változóra nézve nem erősen degenerált. A következő állításban azt az esetet vizsgáljuk, amikor éppen a vezérváltozó távozik a bázisból.

3.1. Állítás. *Az MBU szimplex algoritmus egy adott lépésében az aktuális B bázis esetén legyen $r \in \mathcal{I}_B^-$ és $q \in \mathcal{I}_B^\oplus, t_{qs} > 0$. Tegyük fel, hogy*

$$\frac{\bar{b}_q}{t_{qs}} = \Theta_2 \geq \Theta_1 = \frac{\bar{b}_r}{t_{rs}} > 0,$$

ahol $\bar{b}_r < 0$, $t_{rs} < 0$, $\bar{b}_q \geq 0$ és $t_{qs} > 0$. Ekkor az algoritmus a t_{rs} elemen végez báziscserét. Jelölje a báziscsere utáni új bázist B'.

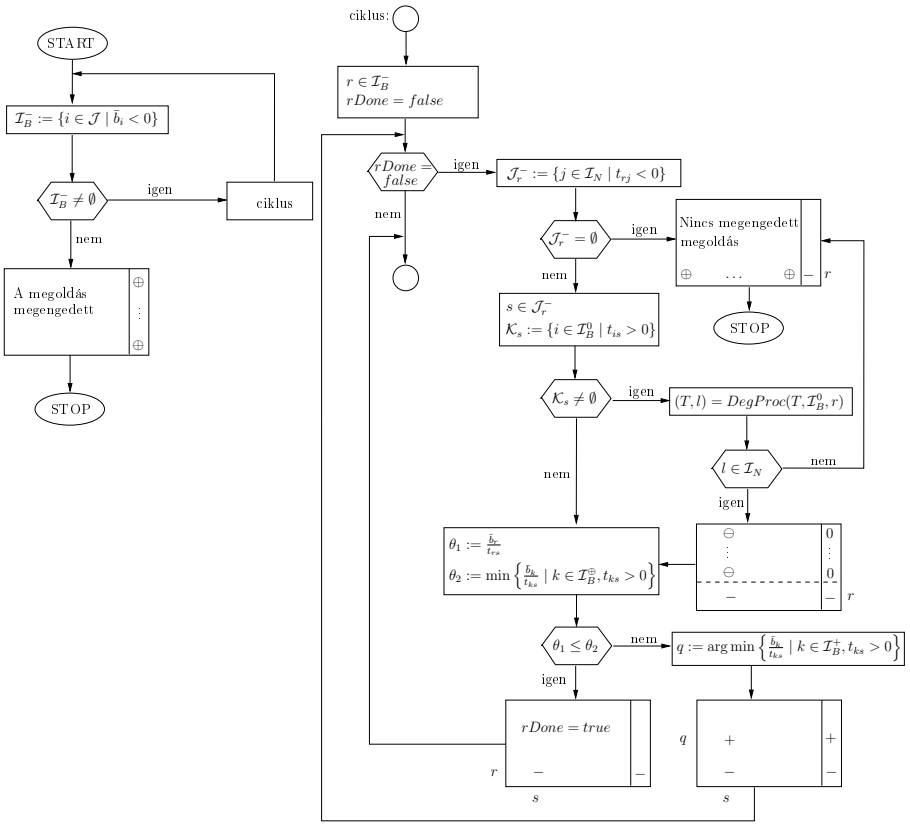
Ekkor

$$\mathcal{I}_B^\oplus \cup \{s\} \subseteq \mathcal{I}_{B'}^\oplus$$

teljesül, és így

$$\left|\mathcal{I}_{B'}^\oplus\right| > \left|\mathcal{I}_B^\oplus\right|.$$

Bizonyítás. A t_{rs} elemen végzett báziscsere során az x_r változó távozik a bázisból, és így új értéke nulla, míg az x_s változó belép a bázisba. Jelölje az új bázishoz tartozó megoldást – és így az új bázistábla jobboldalát – $\mathbf{b}^+$. Bizonyításunk esetszétválasztáson alapszik.



4. ábra. Az MBU szimplex algoritmus folyamatábrája megengedettségi feladatokra.

a) Az s index esetén a megfelelő jobboldal $b_s^+ = \frac{\bar{b}_r}{t_{rs}} = \Theta_1 > 0$, vagyis a vezérváltozó helyére belépő változó megengedett lesz.

b) Az $i \in \mathcal{I}_B^+, i \neq s$ esetén az új jobboldal a $b_i^+ = \bar{b}_i - \frac{t_{is}\bar{b}_r}{t_{rs}}$ képlettel számolható ki. Ha $t_{is} \leq 0$, akkor felhasználva, hogy $\bar{b}_i > 0$, $\bar{b}_r < 0$ és $t_{rs} < 0$, adódik, hogy $\bar{b}_i^+ > 0$, mivel egy nemnegatív számot adunk a már amúgy is pozitív $\bar{b}_i$ jobboldalhoz. Egyébként, ha $t_{is} > 0$, akkor a $b_i^+ = t_{is}\left(\frac{\bar{b}_i}{t_{is}} - \frac{\bar{b}_r}{t_{rs}}\right)$ átalakítás alapján, mivel a vezérváltozó sorában történik a báziscsere, ezért a $\frac{\bar{b}_i}{t_{is}} \geq \Theta_2 \geq \Theta_1 = \frac{\bar{b}_r}{t_{rs}} > 0$ feltétel alapján $b_i^+ \geq 0$.

c) Legyen $i \in \mathcal{I}_B^0$. Az új jobboldal értéke $b_i^+ = \bar{b}_i - \frac{t_{is}\bar{b}_r}{t_{rs}}$. Mivel $\frac{\bar{b}_q}{t_{qs}} = \Theta_2 > 0$, ezért a $q \notin \mathcal{I}_B^0$, vagyis a $\mathcal{K}_s = \emptyset$, és így $t_{is} \leq 0$. Figyelembe véve a $\bar{b}_i = 0$ feltételt is, a $b_i^+ = -\frac{t_{is}\bar{b}_r}{t_{rs}} = -t_{is}\,\Theta_1 \geq 0$ teljesül.

Mivel minden esetet számba vettünk, így megmutattuk, hogy megengedett változó nem válik negatívvá a báziscsere folyamán, de az x_r korábban nem megengedett vezérváltozó megengedetté vált, így $\left|\mathcal{I}_{B'}^{\oplus}\right| > \left|\mathcal{I}_{B}^{\oplus}\right|$. Ezzel állításunkat beláttuk. □

A következő állítás azzal az esettel foglalkozik, amikor nem a vezérváltozó sorában választunk pivot pozíciót. Ha az aktuális bázisunk nem lokálisan erősen degenerált a vizsgált nem bázis változóra nézve, akkor az algoritmus által választott báziscsere továbbra is x_r növelő lesz.

3.2. Állítás. *Az algoritmus egy adott lépésében az aktuális B bázis esetén legyen $r \in \mathcal{I}_B^-$ és $q \in \mathcal{I}_B^{\oplus}$, $t_{qs} > 0$. Tegyük fel, hogy*

$$0 \leq \frac{\bar{b}_q}{t_{qs}} = \Theta_2 < \Theta_1 = \frac{\bar{b}_r}{t_{rs}},$$

ahol $\bar{b}_r < 0$, $t_{rs} < 0$, $\bar{b}_q \geq 0$ és $t_{qs} > 0$. Ebben az esetben az algoritmus a t_{qs} elemen végez báziscserét. Jelölje a báziscsere utáni bázist B'. Ekkor $\mathcal{I}_B^{\oplus}\backslash\{q\} \subseteq \mathcal{I}_{B'}^{\oplus}\backslash\{s\}$ és $0 > b_r^+ \geq \bar{b}_r$. Amennyiben a bázis nem degenerált vagy lokálisan gyengén degenerált az s indexre nézve, akkor $b_r^+ > \bar{b}_r$.

Bizonyítás. Az előző állításban bevezetett jelöléseket és gondolatmenetet használva a hányadosteszekből adódik, hogy tetszőleges $i \in \mathcal{I}_B^{\oplus}$ index esetén $i \in \mathcal{I}_{B'}^{\oplus}$ teljesül $i \neq q$ esetén.

Továbbá, mivel $b_s^+ = \frac{\bar{b}_q}{t_{qs}} \geq 0$, így $s \in \mathcal{I}_{B'}^{\oplus}$ és így

$$\mathcal{I}_B^{\oplus}\backslash\{q\} \subseteq \mathcal{I}_{B'}^{\oplus}\backslash\{s\},$$

ami biztosítja, hogy egy már megengedett változó ne váljon negatívvá. A vezérváltozó megváltozását $b_r^+ = \bar{b}_r - \frac{t_{rs}\bar{b}_q}{t_{qs}}$ formula adja meg, ahol $-\frac{t_{rs}\bar{b}_q}{t_{qs}} \geq 0$, hiszen $t_{rs} < 0$, $t_{qs} > 0$ és $\bar{b}_q \geq 0$. A $\Theta_2 < \Theta_1$ feltevés miatt $0 \geq \frac{t_{rs}\bar{b}_q}{t_{qs}} > \bar{b}_r$ adódik, és így

$$0 > b_r^+ = \bar{b}_r - \frac{t_{rs}\bar{b}_q}{t_{qs}}.$$

Amennyiben a bázis nem degenerált, vagy lokálisan gyengén degenerált az $s \in \mathcal{I}_N$ indexre, akkor definíció szerint $\Theta_2 > 0$ teljesül, tehát $\frac{\bar{b}_q}{t_{qs}} > 0$ és $-t_{rs}\frac{\bar{b}_q}{t_{qs}} > 0$, így

$$0 > b_r^+ = \bar{b}_r - t_{rs}\frac{\bar{b}_q}{t_{qs}} > \bar{b}_r$$

teljesül. Ezzel állításunkat beláttuk. □

Geometriailag a 3.2. Állítást úgy lehet értelmezni, hogy a báziscsere során a vezérváltozó által kijelölt feltételhez közelebbi megoldást kapunk.

A 3.1. és 3.2. Állításokat összefoglalva kapjuk az alábbi következményt.

3.1. Következmény. *Az MBU szimplex algoritmus megengedettségi feladatokra, ha lokálisan nem erősen degenerált bázisokon halad, akkor minden lépésben x_r növelő báziscserét hajt végre a vezérváltozóra nézve, és így véges.*

Bizonyítás. Mivel a lehetséges bázisok száma véges, így elég megmutatni hogy az algoritmus nem ciklizál, vagyis ugyanaz a bázis nem fordulhat elő kétszer. Mivel feltettük, hogy a bázisok, amelyeken az algoritmus végighalad nem lehetnek lokálisan erősen degeneráltak, ezért a 3.1. és 3.2. Állításokból következik, hogy az algoritmus minden lépésben x_r növelő báziscserét hajt végre a vezérváltozóra nézve, így minden lépésben vagy egy újabb változó válik megengedetté, vagy pedig a vezérváltozó értéke növekszik, így kétszer ugyanaz a bázismegoldás nem fordulhat elő. □

A 2.1. Példa folytatásaként bemutatunk egy a vezérváltozó sorában történő pivotálást is. Ezzel az MBU szimplex algoritmusunk mindkét nem degenerált pivot fajtájára (x_r növelő pivotálásra) példát adunk.

3.1. Példa.

	x_2	x_5	
x_1	0	-1	0
x_4	1	-1	1
x_3	1	$\mathbf{-1}$	-1

A tábla továbbra is degenerált, a báziscsere oszlopa egyértelmű, hiszen az x_3 vezérváltozó sorában csak egy negatív elem van. A gyengén degeneráltság miatt és a hányados teszt értelmében, a vezérváltozó sorában elvégezhető a báziscsere.

	x_2	x_3	
x_1	-1	-1	1
x_4	0	-1	2
x_5	-1	-1	1

A 3.1. és 3.2. Állítások és a 3.1. Következmény az MBU szimplex algoritmus pivotálási szabályát jellemző eredmények. Anstreicher és Terlaky [2] cikkükben lineáris programozási feladatra definiált primál algoritmusukra hasonló eredményeket igazoltak.

A következő fejezetben a vezérváltozó értékének növekedésére alsó korlátot adunk, és így az algoritmus lépésszámára felső korlátot állítunk elő.

Legjobb tudomásunk szerint a szakirodalomból eddig még nem ismert olyan általános feladatokra kifejlesztett pivot algoritmus, amelynek a lépésszámára az algoritmus elemzéséből tudtak korlátot kiszámítani. A pivot algoritmusokat az jellemzi, hogy lépésszámukra általában legrosszabb eset elemzés ismert csak, vagy pedig sejtik, hogy exponenciális lépésszámú példa adható rájuk.

Az itt bemutatott módszert hasonló módon lehet elemezni, és hasonló lépésszám becslést lehet adni az MBU algoritmus lineáris programozási megfelelőjére, illetve a szimplex algoritmus megengedettségi feladatok megoldására szolgáló első

fázisára is. Míg a primál MBU algoritmus geometriai interpretációja, hogy egy kiszemelt sérülő feltételhez közeledünk, a primál szimplex algoritmus első fázisa a nem megengedettségek összegét próbálja csökkenteni.

3.1. Lépésszám becslés: lokálisan nem erősen degenerált pivot sorozatok esetén

Ebben a fejezetben feltesszük, hogy algoritmusunk lokálisan nem erősen degenerált bázisokon halad. A degenerált esettel a 4. fejezetben foglalkozunk.

A rövid pivot tábla és a bázismegoldás definiciója alapján a rövid pivot tábla egy oszlopára és az aktuális bázismegoldásra teljesül, hogy $\mathbf{t}_s = B^{-1}\mathbf{a}_s$ és $\bar{\mathbf{b}} = B^{-1}\mathbf{b}$, vagyis a $\mathbf{t}_s$ és $\bar{\mathbf{b}}$ vektorok a $B\mathbf{u} = \mathbf{a}_s$, illetve $B\mathbf{v} = \mathbf{b}$ lineáris egyenletrendszerek egyértelmű megoldásai. A Cramer szabály alapján tetszőleges $i \in I_B$ index esetén

$$t_{is} = \frac{\det(B_{is})}{\det(B)} \quad \text{és} \quad \bar{\mathbf{b}}_i = \frac{\det(B_i)}{\det(B)},$$

ahol a $B_{is} \in \mathbb{R}^{m\times m}$ mátrixot úgy kapjuk, hogy a B bázis i. oszlopát kicseréljük az $\mathbf{a}_s$ vektorra, és hasonlóan, a B_i mátrix a B i. oszlopának $\mathbf{b}$ vektorra cserélésével keletkezik. Egy olyan x_r növelő pivot során, mely nem a vezérváltozó sorában történik, a 3.2. Állításban bizonyítottak alapján

$$b_r^+ = \bar{b}_r - \frac{t_{rs}\bar{b}_q}{t_{qs}} = \frac{\det(B_r)}{\det(B)} - \frac{\frac{\det(B_{rs})}{\det(B)}\frac{\det(B_q)}{\det(B)}}{\frac{\det(B_{qs})}{\det(B)}} = \frac{\det(B_r)}{\det(B)} - \frac{\det(B_{rs})\det(B_q)}{\det(B_{qs})\det(B)},$$

ahol

$$-\frac{\det(B_{rs})}{\det(B_{qs})}\frac{\det(B_q)}{\det(B)} > 0$$

teljesül a bázis lokálisan nem erősen degenerált volta miatt. Jelölje

$$\Delta_A := \min\left\{ -\frac{\det(B_{rs})\det(B_q)}{\det(B_{qs})\det(B)} : \begin{array}{l} B \text{ az A reguláris részmátrixa és} \\ \frac{\det(Brs)}{\det(B)} < 0,\ \frac{\det(B_q)}{\det(B)} > 0,\ \frac{\det(B_{qs})}{\det(B)} > 0 \end{array} \right\}$$

a vezérváltozó értékének minimális növekedését. Felhasználva, hogy csak lokálisan nem erősen degenerált bázisokat vizsgálunk, adódik, hogy $\Delta_A > 0$ véges, és

$$b_r^+ = \bar{b}_r - \frac{t_{rs}\bar{b}_q}{t_{qs}} = \frac{\det(B_r)}{\det(B)} - \frac{\det(B_{rs})\det(B_q)}{\det(B_{qs})\det(B)} \geq \frac{\det(B_r)}{\det(B)} + \Delta_A.$$

Összefoglalva, az x_r növelő pivot vagy megengedetté teszi a vezérváltozót, vagy annak értékét legalább Δ_A-val növeli. Habár a végesség bizonyításához ez már elegendő lenne, a lépésszám felső becsléséhez meg kell becsülnünk a vezérváltozók lehetséges legnagyobb abszolút értékét. Legyen

$$\Delta_{\max} := \max\left\{ -\frac{\det(B_r)}{\det(B)} : \begin{array}{l} sgn(\det(B_r)) = -sgn(\det(B)), \text{ és} \\ B \in \mathbb{R}^{m\times m} \text{ az } A \text{ egy reguláris részmátrixa} \end{array} \right\}$$

a lehetséges legnagyobb abszolútértékű jobboldal a Cramer-szabály alapján. Ha a feladat nem triviális, és van olyan bázis, mely esetén van negatív jobboldal, akkor $\Delta_{\max}$ pozitív és véges. Legyen $\Delta \in \mathbb{Z}$ és $\Delta := \left\lceil \frac{\Delta_{\max}}{\Delta_A} \right\rceil$.

A következő állításban felső korlátot adunk az algoritmus belső ciklusának maximális hosszára, vagyis azon iterációk számára, melyek során a vezérváltozó értéke növekszik ugyan, de nem válik megengedetté.

3.3. Állítás. *Tegyük fel, hogy az algoritmus lokálisan nem erősen degenerált bázisokon halad végig. Legyen $r \in I_B^-$ az aktuális vezérváltozó indexe. Ekkor az algoritmus legfeljebb Δ báziscserét végez mielőtt a vezérváltozó megengedetté válik, vagy legfeljebb Δ báziscsere alatt kimutatja, hogy a megengedettségi feladatnak nincs megoldása.*

Bizonyítás. A definíciók alapján a vezérváltozó legkisebb értéke $-\Delta_{\max}$ lehet, melynek értéke minden iterációban legalább Δ_A-val növekszik, így legfeljebb Δ iterációt végezhet az algoritmus mielőtt a következő x_r növelő pivot a vezérváltozót megengedetté tenné, vagy előállít egy primál infizibilis táblát. □

Készen állunk az algoritmus lépésszámának becslésére.

3.1. Tétel. *Tekintsük az (1) megengedettségi feladatot, és tegyük fel hogy az algoritmus lokálisan nem erősen degenerált bázisokon halad végig. Ekkor az MBU szimplex algoritmus legfeljebb $m\Delta$ báziscserét végez a feladat megoldása során.*

Bizonyítás. A 3.3. Állítás alapján az algoritmus legfeljebb Δ báziscserét végez egy-egy vezérváltozó megengedettségének beállítása előtt, vagy elér egy infizibilis táblát. Az algoritmus indulásakor a nem megengedett változók maximális száma megegyezhet a sorok számával. A 3.1. és 3.2. Állítások alapján, ha egy változó megengedett egy iteráció során akkor az is marad, így az algoritmus nem ciklizálhat, és legfeljebb $m\Delta$ lépésben vagy megoldja a feladatot, vagy kimutatja, hogy a feladatnak nincs megoldása. □

Megmutattuk, hogy a nemdegeneráltsági feltétel teljesülése esetén az algoritmus véges, és korlátot tudtunk adni a szükséges báziscserék maximális számára. Ez a felső korlát gyakran nagyon durva, így érdekes kérdés lenne olyan feladatosztályokat keresni, amelyek esetén a korlát jól számítható, vagyis könnyen meghatározhatók a Δ_A és $\Delta_{\max}$ számok. Egy nyilvánvaló módon felmerülő ilyen feladatosztály a teljesen unimoduláris mátrixok osztálya, azonban a legtöbb ilyen tulajdonságú feladat erősen degenerált, így a becslés csak a lokálisan nem erősen degenerált báziscserék számára ad egyszerű felső korlátot. Ezen felső korlátra a determinánsok meghatározásából $\mathbf{b} \in \mathbb{Z}^m$ esetén egyszerű számolással a $\mathbf{b}$ oszlopa szerinti kifejtést használva $\Delta \le \|\mathbf{b}\|_1$ adódik. Érdekes kérdés lenne megfelelő perturbációkat találni adott feladat osztályokhoz, azonban megjegyezzük, hogy a perturbáció

nagyon erősen befolyásolja a lépésszám becslésében szereplő értékeket. A szakirodalomból ismert eljárások (mint amilyen az ϵ-perturbációs technika is [17, 6]) nem adnak megnyugtató választ a kérdésre[2].

4. Az erős degeneráltság kezelése

Ebben a fejezetben az MBU algoritmus *DegEljárás* (4.1. ábra) részét dolgozzuk ki. A degeneráltságot algoritmikus szempontból leggyakrabban perturbációval vagy indexválasztási szabályokkal kezelik [16].

Az előző fejezetben ismertetett algoritmus elemzésének kulcskérdése az algoritmus által bejárt táblák esetén az erős degeneráltság feltételének a nem teljesülése volt. A degeneráltság elleni eljárás, a vezérváltozó sora és a degenerált részfeladat alapján olyan báziscseréket végez, melyek az aktuális bázismegoldás értékét nem változtatják meg, de úgy transzformálják a pivot tábla belsejét, hogy az vagy primál nem megengedett legyen, vagy legyen olyan oszlopa melyben a vezérváltozó sorában negatív érték szerepel, és a tábla az adott oszlopra nézve lokálisan gyengén degenerált. A degenerált részfeladat kezelése során kizárólag a degenerált sorokban végzünk báziscserét.

Az algoritmus végességét az új típusú index választási szabályok alkalmazása esetén egyszerre, az [1] és [5] dolgozatokban is alkalmazott $\mathbf{s}$ vektor egy általánosításának segítségével bizonyítjuk.

4.1. Az s-monoton pivotálási szabályok

Legyen adott a $B_0, B_1, ..., B_k$ bázis sorozat, amellyel eljutottunk az aktuális B_k bázisba. A bázis sorozat minden bázisához megadható egy $\mathbf{s} \in \mathbb{N}^n_\oplus$ vektor. Hasonlóan ahhoz, hogy a különböző pivotálási szabályokkal nyert $\mathbf{s}$ vektorok, ugyanúgy, ahogyan a bázis sorozatok általában eltérőek, az $\mathbf{s}$ vektorok is különböznek. A különböző pivotálási algoritmusok végességének az elemzése során, hasonló módszerek alkalmazhatóak. Ezeket szeretnénk bizonyos tulajdonságok definiálásával közös keretbe foglalni.

A degeneráció ellenes eljárásunk olyan index választási szabályokat használ, melyek teljesítik a következő tulajdonságot.

4.1. Definíció. Legyen adott egy index választáson alapuló pivotálási szabály, egy $\mathbf{s} \in \mathbb{N}^n_\oplus$ vektor, amelynek a koordinátáit a feladat változóihoz rendeltük, és az algoritmus iterációi során, a pivotálási szabálytól függően módosulhatnak. A pivotálási szabálytól függő $\mathbf{s}$ vektor sorozatra az alábbi elvárásokat fogalmazzuk meg:

[2] Az ϵ-perturbációs technika kritikus mértékben rontja az adódó korlátot.

1. Az $\mathbf{s}$ vektor értékei a báziscserék során nem csökkennek, illetve kizárólag a mozgó változók értéke változhat. Amennyiben a pivotálási szabály több változó bázis állapot változását tenné lehetővé, akkor a mozgó változó indexét az $\mathbf{s}$ vektor maximális értékű elemei közül kell kiválasztani.
2. Ha a pivotálás során előállított valamelyik B_k bázis erősen degenerált, akkor az $\mathbf{s}$ vektor biztosítja, hogy a következő két eset valamelyike előforduljon:
   (i) A B_k bázis után véges sok lépésben véget ér az algoritmus.
   (ii) Ha léteznek olyan változók, melyek a B_k bázis után végtelen sokszor változtatnak bázis állapotot: az ilyen változók indexeinek a halmazát jelölje $\mathcal{I}^*$. A B_k bázis után, véges sok iterációval eljutunk egy olyan B^* bázishoz, amely esetén az $\mathbf{s}$ vektor szerinti legkisebb értékű, bázison kívüli változó egyértelmű az $\mathcal{I}^*$-ra nézve. Jelölje ezt a változót x_l.
3. Amikor a B_k bázis után az x_l változó belép a bázisba, akkor a belépés után addig, amíg az x_l változó újra távozik a bázisból igaz, hogy azon változók $\mathbf{s}$ értéke, amelyek az x_l változó bázisba lépése óta beléptek a bázisba nagyobb, mint az x_l változó $\mathbf{s}$ vektor szerinti értéke.

Azokat a pivotálási szabályokat, amelyekhez tartozó $\mathbf{s}$ vektorokra az 1–3. feltételek teljesülnek $\mathbf{s}$-*monoton* pivotálási szabályoknak nevezzük.

Megmutatjuk, hogy számos index választási szabály eleget tesz az $\mathbf{s}$-monotonitási tulajdonságnak. Megjegyezzük, hogy az index választási szabálytól függően lehet olyan bázis, melyre az $\mathbf{s}$ monotonitás második tulajdonsága teljesül, azonban a harmadik nem, azonban figyeljük meg, hogy ha az $\mathbf{s}$-monotonitási tulajdonság $2/i$ tulajdonsága nem teljesül, akkor olyan B^* bázis létezését kötöttük ki, melyre $2/ii$ és 3 együtt teljesül.

A bizonyítások során csak a nem triviális $2/ii$ és 3 tulajdonságokat bizonyítjuk.

4.1. Lemma. *A Bland-féle minimál index szabály* $\mathbf{s}$-*monoton.*

Bizonyítás. A Bland-féle minimálindex szabály esetén az $\mathbf{s}$ vektort a következő módon konstansnak definiálhatjuk:

$$s_i = n - i, \quad i \in \mathcal{I}.$$

Könnyen belátható, hogy az így definiált $\mathbf{s}$ vektor tetszőleges részvektora szerinti legnagyobb érték minden esetben a legkisebb indexű változóhoz tartozik, így a maximális $\mathbf{s}$ érték kiválasztásának a szabálya tényleg a Bland-féle minimál index szabálynak megfelelő.

Az $\mathbf{s}$-monotonitás első és második feltétele következik abból, hogy $\mathbf{s}$ értékei nem változnak. A B_{k+1} bázis esetén $\arg\min_{i \in \mathcal{I}_{N_{k+1}} \cap \mathcal{I}^*} s_i = u$ egyértelmű, hiszen az s_i értékek különbözőek, $s_u = n - u$, azaz u a legnagyobb indexű elem az $i \in \mathcal{I}_{N_{k+1}} \cap \mathcal{I}^*$

halmazban. Ha $v = \underset{i \in \mathcal{I}_{B_{k+1}} \cap I^*}{\arg\min} s_i$ olyan, hogy $v \geq u$ (ekkor valójában $v > u$), akkor $B^* = B_{k+1}$ és $l = u$, különben B^* az az első olyan bázis, amikor x_v, már bázison kívüli, és ekkor $l = v$.

A szabály harmadik elvárása azért teljesül, mert az $\mathbf{s}$ vektor konstans, így a mozgó változók közötti legkisebb indexű változó végig ugyanaz. □

A LIFO pivotálási szabály szerint, választási lehetőség esetén, azon változókat részesítjük előnyben, melyek időben visszafelé a lehető legközelebbi időpontban mozogtak.

4.2. Lemma. *A LIFO pivotálási szabály* $\mathbf{s}$*-monoton.*

Bizonyítás. Az $\mathbf{s}$ vektort inicializáljuk azonosan nullának. Egy (l, k) báziscsere során, amennyiben ezen báziscsere az r. báziscsere, akkor az $\mathbf{s}$ vektor frissítése a következő szabály alapján történik:

$$s_i' = \left\{ \begin{array}{ll} r & \text{ha } i \in \{l, k\}, \\ s_i & \text{egyébként.} \end{array} \right\}$$

Az $\mathbf{s}$-monotonitási tulajdonság első pontja nyilvánvalóan teljesül. Legyen adott a definíció szerinti B_k bázis.

Legyen $\mathcal{M} = \mathcal{I}_{N_k} \cap \mathcal{I}^*$. Figyeljük az $i \in \mathcal{M}$ esetén az x_i változók mozgását, és ha valamely x_i változó belép a bázisba, akkor töröljük az $\mathcal{M}$ halmazból, azaz $\mathcal{M} := \mathcal{M} - \{i\}$. Mivel $\mathcal{M} \subset \mathcal{I}^*$, ezért véges sok báziscsere után $|\mathcal{M}| = 1$ teljesül. Az $\mathcal{M}$ halmaz utolsó elemét jelöljük l-el. Ennél a bázisnál nyilván $l = \underset{i \in \mathcal{I}_N \cap \mathcal{I}^*}{\arg\min} s_i$, tehát az egyértelmű x_l változóhoz ezt a bázist jelöljük B^*-gal.

A harmadik pont azonnal következik a LIFO definíciójából. □

A "most-often-selected-variable" (MOSV) pivotálási szabály azon változókat részesíti előnyben, melyek addig a legtöbbet mozogtak.

4.3. Lemma. *A "most-often-selected-variable" pivotálási szabály* $\mathbf{s}$*-monoton.*

Bizonyítás. Az $\mathbf{s}$ vektort inicializáljuk azonosan nullának. Egy (l, k) báziscsere során az $\mathbf{s}$ vektor frissítése a következő szabály alapján történik:

$$s_i' = \left\{ \begin{array}{ll} s_i + 1 & \text{ha } i \in \{l, k\}, \\ s_i & \text{egyébként.} \end{array} \right\}$$

Az első tulajdonság azonnal következik az index választási szabály definíciójából.

Legyen $\mathcal{M}_N = I_{N_k} \cap \mathcal{I}^*$ és $\mathcal{M}_B = I^* \setminus \mathcal{M}_N$. Definiáljuk a t_i értéket a következő módon:

$$t_i = \left\{ \begin{array}{ll} s_i, & \text{ha } i \in \mathcal{M}_N, \\ s_i + 1, & \text{ha } i \in \mathcal{M}_B. \end{array} \right.$$

Legyen $\mathcal{P} = \{i \in \mathcal{I}^* : i \in \underset{j \in \mathcal{I}^*}{\arg\min}\, t_j\}$ és $\min\limits_{j \in \mathcal{I}^*} t_j = p$. Folytassuk az iterációt a pivot szabálynak megfelelően. Mivel $\mathcal{P} \subset \mathcal{I}^*$, ezért bármely $i \in \mathcal{P}$ esetén lesz egy

olyan bázis, amikor az x_i változó a B_k után először lép be a bázisba. Ekkor töröljük az indexét $\mathcal{P}$-ből, azaz $\mathcal{P} := \mathcal{P} \setminus \{i\}$. Véges sok báziscsere után előáll egy olyan $\mathcal{P}$ halmaz, mely esetén $|\mathcal{P}| = |\{l\}| = 1$. Az aktuális bázis utáni első bázis, mikor végül x_l belép a bázisba, legyen B^*. Megmutatjuk, hogy az x_l változó kiválasztása ekkor már egyértelmű. Nyilván ebben az esetben $s_l = p$, attól függetlenül, hogy az x_l a B_k bázis esetén bázis változó volt-e vagy sem. A pivotálási szabály miatt $p < s_i$, ha $i \in \mathcal{I}^* \setminus \mathcal{P}$, és mivel minden $i \in \mathcal{P} \setminus \{l\}$ már legalább egyszer belépett a bázisba, ezért ha most bázison kívül van, és a B_k bázis esetén is bázison kívül volt, akkor az $\mathbf{s}$ szerinti értéke legalább $p+2$. Másfelől, ha a B_k bázis esetén bázisban volt, akkor jelenleg az $\mathbf{s}$ szerinti értéke legalább $p+1$. Így a második tulajdonság teljesül.

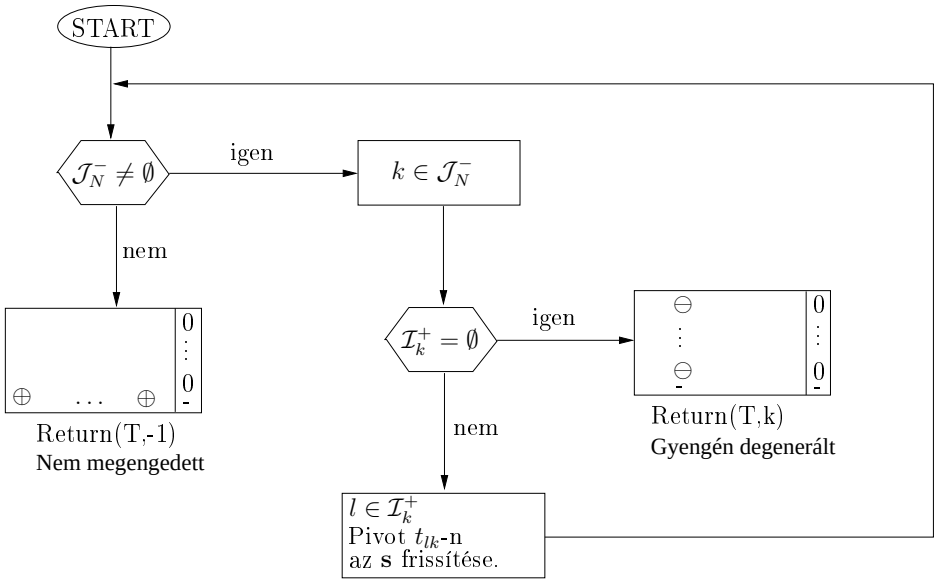
Az $\mathbf{s}$-monoton pivotálási szabály harmadik feltétele is teljesül, mivel a B_k bázis után az x_l változó, a MOSV szabály alkalmazása esetén a fentiek szerint a B^* bázisnál lép be a bázisba, és az I^*-beli később belépők mindegyikének már a B^* bázis esetén is nagyobb volt az $\mathbf{s}$ szerinti értéke. □

4.2. Az algoritmus végessége degenerált esetben

Készen állunk a degeneráltság kezelésére szolgáló algoritmusunk definiálására, amelynek pseudo kódját a 4.1. Algoritmus, míg folyamatábráját a 5. ábra foglalja össze. Megjegyezzük, hogy lehetséges a teljes algoritmus során ugyanazt az $\mathbf{s}$ vektort használni, azonban elegendő annak csupán a degenerált részfeladatok kezelése során történő bevezetése is. Az egyszerűség kedvéért a leírás során ezen utóbbi stratégiát választjuk.

4.1. Algoritmus. Egy lehetséges degeneráltság elleni DegEljárás, az $\mathbf{s}$ vektort használva.

$(T, l) =$ **DegEljárás**$(T, \mathcal{I}_B^0, r)$
Az $\mathbf{s}$ vektor inicializálása.
$\mathcal{J}_N^- := \{j \in \mathcal{I}_N : t_{rj} < 0\}$.
  While $(\mathcal{J}_N^- \neq \emptyset)$ **do**
    Legyen $k \in \mathcal{K} := \left\{i \in \mathcal{J}_N^- : s_i = \max\{s_j : j \in \mathcal{J}_N^-\}\right\}$ tetszőleges index.
    $\mathcal{I}_k^+ := \{i \in \mathcal{I}_B^0 \mid t_{ik} > 0\}$.
    If $\left(\mathcal{I}_k^+ = \emptyset\right)$ **then**
      A hívó tábla gyengén degenerált, **Return**(T, k).
    Endif
    Legyen $l \in \mathcal{L} := \left\{i \in \mathcal{I}_k^+ : s_i = \max\{s_j : j \in \mathcal{I}_k^+\}\right\}$ tetszőleges index.
    Pivotálás a t_{lk} elemen, az $\mathbf{s}$ vektor frissítése.
    $\mathcal{J}_N^- := \{j \in \mathcal{I}_N : t_{rj} < 0\}$.
  Endwhile
  A hívó tábla nem megengedett, **Return**$(T, -1)$.
  Return(T, l).
End



5. ábra. Egy lehetséges degeneráltság elleni eljárás folyamatábrája, az **s** vektort használva. A bázistáblákat a degenerált részre, és a vezérváltozó sorára megszorítva ábrázoltuk.

Figyeljük meg, hogy a degeneráció elleni eljárás mindkét leállási táblája biztosítja a teljes algoritmus végességét, hiszen a teljes táblára nézve az egyik esetben növelő pivot lehetséges, míg a másik esetben nem megengedettség miatt az algoritmus véget ér. Így elegendő bizonyítani hogy a degeneráció elleni eljárás véges.

Bizonyításunk a jól ismert ortogonalitási tételen alapszik. Egy adott B bázis és hozzá tartozó $\mathcal{T}$ rövid pivot tábla esetén definiáljuk a $\mathbf{t}^{(i)}$, $i \in \mathcal{I}_B$, illetve a $\mathbf{t}_j$, $j \in \mathcal{I}_N \cup \{b\}$ vektorokat a következőképpen [13]:

$$\left(\mathbf{t}^{(i)}\right)_k = \begin{cases} t_{ik}, & \text{ha } k \in \mathcal{I}_N \cup \{b\} \\ 1, & \text{ha } k = i \\ 0, & \text{ha } k \in \mathcal{I}_B \setminus \{i\} \end{cases}$$

illetve

$$(\mathbf{t}_j)_k = \begin{cases} t_{kj}, & \text{ha } k \in \mathcal{I}_B \\ -1, & \text{ha } k = j \\ 0, & \text{ha } k \in (\mathcal{I}_N \cup \{b\}) \setminus \{j\} \end{cases}$$

Ezek után az ortogonalitási tételt [13] az alábbi formában mondhatjuk ki:

4.1. Tétel. *Legyen adott az* (1) *megengedettségi feladat két tetszőleges* B', *illetve* B'' *bázisa. Bármely* $\mathbf{t}'^{(i)}$, $i \in \mathcal{I}_{B'}$ *vektor merőleges bármely* $\mathbf{t}''_j$, $j \in \mathcal{I}_{N''}$ *vektorra.*

Készen állunk eljárásunk végességének bizonyítására.

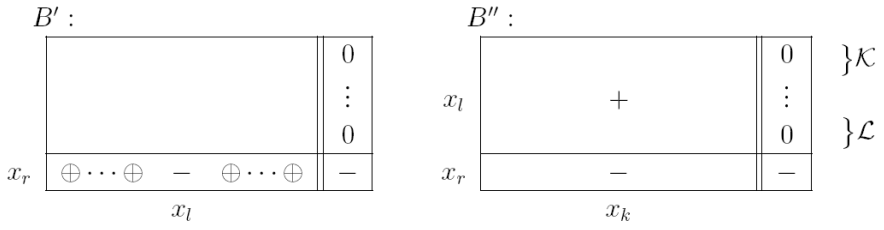
4.2. Tétel. *Ha a degenerációs eljárás* **s***-monoton pivotálási szabályt alkalmaz, akkor véges.*

Bizonyítás. A **DegEljárás** esetén a teljes feladat csupán a degenerált sorokból és a vezérváltozó sorából áll.

Tegyük fel indirekt módon, hogy a degeneráció elleni eljárás nem véges. Mivel a bázisok száma véges, így ez csak úgy lehetséges, ha az eljárás ciklizál[3]. Tekintsünk egy minimális méretű ciklizáló példát. Az **s** monotonitás tulajdonságai miatt egy ilyen feladat esetén minden változó – a vezérváltozó kivételével – végtelen sokszor mozog.

Mivel a pivotálási szabály **s**-monoton, ezért van olyan B' bázis és x_l bázison kívüli változó, melynek az **s** vektor szerinti értéke a bázisba való belépés pillanatában a bázison kívüli változók közül minimális (1. és 2. tulajdonság alapján). Mivel az algoritmus a lehetséges x_i változók közül a legnagyobb s_i értékűt választja ki, ez csak úgy lehetséges, hogy ha a B' bázishoz tartozó $t'^{(r)}$ vektorra $t'_{rl} < 0$, illetve $t'_{ri} \geq 0$ bármely $i \in \mathcal{I} - \{l\}$ esetén.

Tekintsük most azt a B'' bázist, mikor az x_l változó legközelebb kilép a bázisból. A belépő változó legyen az x_k. A degeneráció elleni eljárás szabálya szerint ez azt jelenti, hogy a $\mathbf{t}''_k$ vektor pozitív elemeihez tartozó változók közül az x_l értéke a legnagyobb az $\mathbf{s}''$ vektorban, hiszen egyéb esetben másik változót választott volna az eljárás. A kisebb **s** értékű változók azonban az **s** szabály harmadik pontja alapján kizárólag olyan változók lehetnek, melyek B' bázis óta nem mozogtak. Legyen ezen változók indexeinek halmaza $\mathcal{K}$, vagyis $\mathcal{K} = \{j \in \mathcal{I}_{B''} : t''_{jk} > 0\} \cap \mathcal{I}_{B'}$. Jelölje továbbá $\mathcal{L} = \{j \in \mathcal{I}_{B''} : t''_{jk} \leq 0\}$ indexhalmazt. Nyilván $l \notin \mathcal{L} \cup \mathcal{K}$. A B' és B'' bázisokhoz tartozó pivot táblát a 6. ábra mutatja.



6. ábra. A B' és B'' bázisok.

[3]A flexibilis index választási szabályok esetén ciklizálás alatt csupán annyit értünk, hogy az algoritmus végtelen sokszor visszatér egy adott állapotba (bázisba).

Mivel $t'_{ri} = 0$, ha $i \in \mathcal{K}$, és $t''_{ik} = 0$, ha $i \in \mathcal{I}_{N''} - \{k\}$, így

$$\mathbf{t}'^{(r)^T}\mathbf{t}''_k = \sum_{i\in\mathcal{I}_{N''}} t'_{ri}t''_{ik} + \sum_{i\in\mathcal{K}} t'_{ri}t''_{ik} + \sum_{i\in\mathcal{L}} t'_{ri}t''_{ik} + t'_{rl}t''_{lk} + t'_{rr}t''_{rk} \le t'_{rl}t''_{lk} + t'_{rr}t''_{rk} =$$
$$= t'_{rl}t''_{lk} + t''_{rk} < 0,$$

mivel $t'_{rj} \ge 0$, ha $j \in \mathcal{L}$, és $t'_{ri}t''_{ik} = 0$, ha $i \in \mathcal{K}$, mivel ezek a változók nem mozogtak a két bázis között. Figyelembe véve a $t'_{rl} < 0$ és $t''_{lk} > 0$, illetve $t'_{rr} = 1$ és $t''_{rk} < 0$ feltételeket $t'_{rl}t''_{lk} < 0$ adódik, amely ellentmond az ortogonalitási tételnek. □

Figyelembe véve, hogy a **DegEljárás** során az x_r változó egyetlen egyszer sem lép be a bázisba az iterációk során, valamint a 6. ábrán bemutatott leállási táblák alapján világosan látszik, hogy a degenerált részfeladat megfogalmazható lenne lineáris programozási részfeladatként is, az x_r vezérváltozó sorával, mint célfüggvény sorral.

Másfelől, bármely a lineáris programozási feladatok pivot algoritmussal történő megoldásának a végességét biztosító módszer (pl. lexikografikus szabály) is alkalmazható lenne a **DegEljárás** egy módosított változatában. Erre egy utalás erejéig Anstreicher és Terlaky [2] is kitér a lineáris programozási feladatra megfogalmazott MBU-szimplex módszerről írt cikkükben.

Az előzőek alapján kimondjuk, hogy az **s**-monoton pivotálási szabállyal megfogalmazott algoritmusunk véges.

4.3. Tétel. *Tetszőleges megengedettségi feladatot az MBU-algoritmussal,* **s***-monoton indexválasztási szabály alkalmazása esetén véges sok pivot iterációban megoldhatunk.*

Bizonyítás. Nem erősen degenerált feladatok esetén a növelő báziscserék biztosítják, hogy egy adott bázis ne térhessen vissza. Erősen degenerált pivot táblát a degeneráció elleni eljárás a 4.2. Tétel alapján véges sok pivot lépésben gyengén degenerálttá konvertálja, vagy kimutatja hogy a feladat nem megengedett anélkül, hogy a tábla jobboldalát megváltoztatná. □

4.3. Algoritmus változatok

Gyakorlatban az algoritmus hatékonysága növelhető, ha a vezérváltozó, illetve a bázisba belépő elem választása nem tetszőleges, habár éppen ez a szabadsága hasonlóan az index-választási szabályok flexibilitásához numerikusan rosszul viselkedő feladatok esetén praktikus lehet.

A pivot típusú módszerek legköltségesebb része a bázistábla megfelelő sorainak, illetve oszlopainak az előállítása. Azonban a báziscserék hatékonyságának szempontjából érdekes lehet a vezérváltozó meghatározásakor a megfelelő sorok degeneráltságának vizsgálata, ugyanis könnyen előfordulhat, hogy egy adott vezérváltozóhoz tartoznak lokálisan gyengén, illetve erősen degenerált oszlopok is. Hasonlóan, előfordulhat olyan bázistábla is, melyen választható olyan vezérváltozó,

mely egy lépésben megengedetté tehető, de van olyan vezérváltozó is, melynek megengedetté tételéhez degeneráció ellenes eljárás szükséges.

Ha a bázistábla nagyobb részét ismerjük, olcsón elvégezhető a következő, lokálisan erős degeneráltság elleni taktika. Amennyiben bármelyik lépésben lenne olyan negatív változó, mely sorában lehetséges olyan báziscserét végezni, amely növeli a megengedett változók számát, akkor a vezérváltozót a végesség veszélyeztetése nélkül le lehet cserélni.

Sajnos ez az ötlet az erős degenerált tábláknál nem mindig alkalmazható, ahogy azt a 3. ábra példája is mutatta.

Az MBU algoritmus tulajdonságaira építve kifejleszthető a degeneráltság kezelésére egy rekurzív primál-duál eljárás is [3].

A **DegEljárás** végességének a biztosítására kifejlesztett módszerek egyszerűen átvihetők lineáris programozási pivot algoritmusok végességének az igazolására is. Ebben az értelemben a 3. és a 4. Tételek eredményei az Illés és Mészáros [11], illetve az S. Zhang [21] cikkében szereplő végesség bizonyítások közös általánosítása.

Köszönetnyilvánítás.

Csizmadia Zsolt és Illés Tibor köszönetet mond a MOL Rt.-nek, a számukra biztosított kutatási ösztöndíjért. Ezt a kutatást az OTKA T 049789 számú pályázata és a TÉT SLO-4/2005 számú pályázata támogatta.

Hivatkozások

[1] Akkeleş A. A. Balogh L. és Illés T.: *A véges criss-cross módszer új variánsai biszimmetrikus lineáris komplementaritási feladatra.* Alkalmazott Matematikai Lapok **21**:1–25, (2003).

[2] Anstreicher, K. M. and Terlaky T.: *A Monotonic Build-Up Simplex Algorithm for Linear Programming.* Operations Research **42**:556-561, (1994).

[3] Bilen F., Csizmadia Zs. and Illés T.: *Anstreicher-Terlaky type monotonic simplex algorithms for linear feasibility problems.* Optimization Methods and Software, Vol. **22**, No. **4**, 679–695, (2007).

[4] Borgwardt, K. H.: *The Simplex Method: A Probabilistic Analaysis.* Algorithms and Combinatorics, Vol. **1**, Springer, Berlin, (1987).

[5] Csizmadia Zs. and Illés T.: *New criss-cross type algorithms for linear complementarity problems with sufficient matrices.* Optimization Methods and Software, Vol. **21**, **2**:247–266, (2006).

[6] Chvatal, V.: *Linear programming.* W. H. Freeman and Company, New York, (1983).

[7] Dantzig, G. B.: *Programming in a Linear Structure.* Comptroller, USAF, Washington D.C., (1948).

[8] Dantzig, G. B.: *Linear Programming and Extensions.* Princeton University Press, Princeton, New Jersey, (1963).

[9] Fukuda K. and Terlaky T.: *Criss-cross methods: A fresh view on pivot algorithms.* Mathematical Programming **79**:369–395, (1997).

[10] Fukuda K. and Terlaky T.: *On the existence of a short admissible pivot sequence for feasibility and linear optimization problems.* Pure Mathematics with Applications 10 no. **4**, 431–447, (1999).

[11] Illés T. és Mészáros K.: *A Farkas-lemma egy új és elemi bizonyítása.* Komlósi S. és Szántai T. (szerkesztők): *Új utak a magyar operációkutatásban: In memoriam Farkas Gyula.* Dialóg Campus Kiadó, Pécs, (1999).

[12] Karmarkar, N.: *A new polynomial-time algorithm for linear programming.* Combinatorica **4**:373–395, (1984).

[13] Klafszky E. és Terlaky T.: *A pivot technika szerepe a lineáris algebra néhány alapvető tételének a bizonyításában.* Alkalmazott Matematikai Lapok **14**:425–448, (1989).

[14] Klafszky E. és Terlaky T.: *Magyar módszer típusú algoritmusok lineéris programozási feladatok megoldására.* Alkalmazott Matematikai Lapok **12**, 1–14, (1986).

[15] Klee, V. és Minty, G. J.: *How good is the simplex algorithm?* In O. Shisha, editor, Inequalities III, Academic Press, New York, (1972).

[16] Maros I.: *Computational Techniques of the Simplex Method.* Kluwer Academic Publishers, Boston, (2003).

[17] Murty, K. G.: *Linear and combinatorial programming.* Robert E. Krieger Publishing Co. Inc., Melbourne, (1985).

[18] Terlaky T.: *A convergent criss-cross method.* Mathematische Operationsforschung und Statistik Series Optimization , Vol. **16**, No. **5**, 683–690, (1985).

[19] Terlaky T. and Zhang, S.: *Pivot rules for linear programming: A survey on recent theoretical developments.* Annals of Operations Research **46**:203–233, (1993).

[20] Todd, M. J.: *Polynomial expected behavior of a pivoting algorithm for linear complementarity and linear programming problems.* Mathematical Programming **35**:173–192, (1986).

[21] Zhang, S.: *A new variant of criss-cross pivot algorithms for linear programming.* European Journal of Operational Research, **116**:607–614, (1999).

(Beérkezett: 2005. március 21.)

CSIZMADIA ZSOLT, ILLÉS TIBOR
EÖTVÖS LORÁND TUDOMÁNYEGYETEM
OPERÁCIÓKUTATÁSI TANSZÉK
1117 BUDAPEST, PÁZMÁNY PÉTER SÉTÁNY 1/C.
csisza@math.elte.hu, illes@math.elte.hu

BILEN, FILIZ
EASTERN MEDITERRANEAN UNIVERSITY
DEPARTMENT OF MATHEMATICS AND COMPUTER SCIENCE
FAMAGUSTA, NORTH-CYPRUS
filiz.erbilen@mail.emu.edu.tr

A NEW ANALYSIS FOR MONOTONIC TYPE SIMPLEX ALGORITHMS FOR FEASIBILITY PROBLEMS

Zsolt Csizmadia, Filiz Bilen and Tibor Illés

A variant of Anstreicher and Terlaky's (1994) monotonic build-up (MBU) simplex algorithm for linear feasibility problems is defined. Under a nondegeneracy assumption weaker than the usual one, the complexity of the algorithm can be given by $m\Delta$, where Δ is a constant determined by the input data of the problem and m is the number of constraints. The constant Δ cannot be bounded in the general case by a polynomial of the bit length of the input data.

Flexible index selection rules provide finiteness for strongly degenerate problems. The flexibility of the rules provides possibility to avoid numerically instable pivots. The proof of finiteness is presented in a unified framework – using the **s**-monoton property of pivot rules, defined in this paper –, that incorporates the usual minimal index rule, the Last-In-First-Out and the most-often-selected-variable index rules.

Key words: feasibility problem, monoton simplex algorithms, degeneracy, linear programming, index selection rules.