sciendo

Műszaki Tudományos Közlemények vol. 15. (2021) 11–14.
DOI: Magyar: https://doi.org/10.33895/mtk-2021.15.03
Angol: https://doi.org/10.33894/mtk-2021.15.03

EME MTSZ MTK

Prediktor-korrektor belsőpontos algoritmus az általános lineáris komplementaritási feladatra

Predictor-Corrector Interior-Point Algorithm for the General Linear Complementarity Problem

Darvay Zsolt,[1] Füstös Ágnes[2]

[1] *Babeș–Bolyai Tudományegyetem, Matematika és Informatika Kar, Kolozsvár, Románia, darvay@cs.ubbcluj.ro; Erdélyi Múzeum-Egyesület, Matematikai és Informatikai Szakosztály*

[2] *Babeș–Bolyai Tudományegyetem, Matematika és Informatika Kar, Kolozsvár, Románia, fustosagi@yahoo.com*

Abstract

We study a predictor-corrector interior-point algorithm for solving general linear complementarity problems from the implementation point of view. We analyze the method proposed by Illés, Nagy and Terlaky [1] that extends the algorithm published by Potra and Liu [2] to general linear complementarity problems. A new method for determining the step size of the corrector direction is presented. Using the code implemented in the C++ programming language, we can solve large-scale problems based on sufficient matrices.

Keywords*: interior-point algorithm, general linear complementarity problem, predictor-corrector algorithm, numerical results, sufficient matrix, object-oriented programming.*

Összefoglalás

Prediktor-korrektor általános lineáris komplementaritási feladatra vonatkozó belsőpontos algoritmust vizsgálunk az implementáció szemszögéből nézve. Az Illés, Nagy és Terlaky [1] által megadott módszert elemezzük, amely általános lineáris komplementaritási feladatra terjeszti ki a Potra és Liu [2] által közölt algoritmust. A korrektorirány lépéshosszának meghatározására egy új módszert vezetünk be. A C++ programozási nyelvben implementált kód nagy méretű elégséges mátrixokra alapozott feladatok megoldására is alkalmas.

Kulcsszavak*: belsőpontos algoritmus, általános lineáris komplementaritási feladat, prediktor-korrektor módszer, numerikus eredmények, elégséges mátrix, objektumorientált programozás.*

1. Bevezetés

Lineáris komplementaritási feladatokkal (*LCP*) a gyakorlatban különböző műszaki vagy gazdasági problémák megoldása során találkozhatunk. A feladatot egy M mátrix határozza meg, amely egy lineáris összefüggésben szerepel, de ezenkívül teljesülnie kell egy komplementaritási feltételnek is.

Az *LCP* NP-teljes feladat, ezért nagyon bonyolult egy hatékony általános megoldást adni rá. Léteznek algoritmusok, amelyek polinom időn belül oldják meg az *LCP*-t, ha az M mátrix pozitív szemidefinit. Ennek kiterjesztéseként bevezették az elégséges mátrix fogalmát, és bebizonyították, hogy ilyen tulajdonságú bemeneti M mátrix esetén is megoldható a feladat polinom időben.

Az általános *LCP* (*General Linear Complementarity Problem*, *GLCP*) keretében bevezetett algoritmusok el tudják dönteni, hogy a *GLCP* mátrixa $P_*(\kappa)$ tulajdonságú vagy sem (az elégségességgel egyenértékű tulajdonság).

Az Illés Tibor, Nagy Marianna és Terlaky Tamás által bevezetett módszer [1, 3, 4] alapján a különböző belsőpontos algoritmusok beépített ellenőrzések segítségével oldják meg a *GLCP*-t polinom időben.

Korábban már implementáltunk egy rövid lépéses belsőpontos algoritmust a *GLCP* megoldására [5]. Mivel a prediktor-korrektor módszer általában hatékonyabb, a továbbiakban egy ilyen jellegű algoritmus megvalósítását vizsgáljuk.

2. A lineáris komplementaritási feladat bemutatása

Az *LCP* esetén azokat az $x, s \in \mathbb{R}^n$ vektorokat keressük, amelyekre:

$$\begin{cases} Mx + q = s, \\ xs = 0, \\ x \geq 0, s \geq 0, \end{cases} \tag{1}$$

ahol $q \in \mathbb{R}^n, M \in \mathbb{R}^{n\times n}$ és xs a vektorok komponensenkénti szorzatát jelöli.

3. A $P_*(\kappa)$ tulajdonság

Kojima és társai [6] meghatározásában egy $M \in \mathbb{R}^{n\times n}$ mátrix $P_*(\kappa)$ tulajdonságú ($\kappa \geq 0$), ha bármely $x \in \mathbb{R}^n$esetén teljesül a következő:

$$(1 + 4\kappa)\sum_{i\in\mathcal{I}_+(x)} x_i(Mx)_i + \sum_{i\in\mathcal{I}_-(x)} x_i(Mx)_i \geq 0, \tag{2}$$

ahol $\mathcal{I}_+(x) = \{1 \leq i \leq n : x_i(Mx)_i > 0\}$ és
$\mathcal{I}_-(x) = \{1 \leq i \leq n : x_i(Mx)_i < 0\}$.

Egy $M \in \mathbb{R}^{n\times n}$ mátrix P_* tulajdonságú, ha $P_*(\kappa)$ tulajdonságú valamely $\kappa \geq 0$ értékre. Megjegyezzük, hogy $\kappa = 0$ esetén a pozitív szemidefinit mátrixok osztályát kapjuk vissza.

4. Lokális κ értékek

Az egyes iterációk prediktor-, illetve korrektorlépéseiben az alábbi függvény segítségével kiszámolunk egy lokális κ-t [1, 4], majd az összes ilyen érték közül kiválasztva a maximumot egy alsó korlátot határozunk meg az M mátrixot jellemző κ-ra:

$$\kappa(\Delta x) = -\frac{1}{4}\frac{\Delta x^T M \Delta x}{\sum_{i\in\mathcal{I}_+(\Delta x)} \Delta x_i (M\Delta x)_i}. \tag{3}$$

5. A környezet fogalma

Legyen $\gamma \in (0, 1)$ és

$$\mathcal{F}^0 := \{(x, s) \in \mathbb{R}^n \times \mathbb{R}^n : -Mx + s = q,\ x, s > 0\}.$$

Potra és Liu [2] a környezetet a következőképpen definiálta:

$$D(\gamma) := \left\{(x, s) \in \mathcal{F}^0 : xs \geq \gamma \frac{x^T s}{n} e\right\}, \tag{4}$$

ahol e az 1-esekből álló n dimenziós vektor.

6. A Newton-módszer

Az algoritmus egy kezdeti (x_0, s_0) pontból indul, és a prediktor-, illetve korrektorirányok kiszámolása érdekében az alábbi Newton-rendszert oldjuk meg:

$$\begin{cases} -M\Delta x + \Delta s = -r_q, \\ s\Delta x + x\Delta s = a, \end{cases} \tag{5}$$

ahol $r_q = Mx + q - s$ a megengedettség megőrzése érdekében került bevezetésre.

Megjegyezzük, hogy a prediktorlépésben $a = -xs$, illetve a korrektorlépésben $a = \mu e - xs$ lesz. Ugyanakkor az algoritmus folyamán az alábbi jelöléseket is használjuk:
$x(\alpha) = x + \alpha\Delta x$, $s(\alpha) = s + \alpha\Delta s$, ahol $\alpha > 0$ a lépéshosszt adja meg.

7. Az algoritmus

Az Illés, Nagy és Terlaky [1] által bevezetett algoritmusból kiindulva a *GLCP*-t megoldó prediktor-korrektor algoritmust a következőképpen adjuk meg:

Bemeneti paraméterek:

$\tilde{\kappa} > 0$ felső korlát a κ számára;
$\varepsilon > 0$ pontossági paraméter;
$\gamma \in (0, 1)$ környezetre vonatkozó paraméter;
$(x_0, s_0) \in D(\gamma)$ kezdőpont;
$\sigma \in (0,1]$ a μ csökkentését szabályozó paraméter;
$\rho \in (0,1)$ a lépéshossz csökkentésére vonatkozó paraméter;
$\beta, \beta_{max} > 0$ a korrektorlépéshossz pontosságára vonatkozó paraméterek;

Kimenet:

az (x, s) vektorpár – az *LCP* megoldása – vagy egy üzenet arra vonatkozóan, hogy nem teljesül a $P_*(\kappa)$ tulajdonság.

BEGIN

$x := x_0$; $s := s_0$; $\mu := \sigma\frac{(x_0)^T s_0}{n}$; $\kappa := 0$;

$r_q = Mx + q - s$; $\alpha_p^*(\kappa) = \frac{2\sqrt{(1-\gamma)\gamma}}{(1+4\kappa)n+2}$;

while $\frac{x^T s}{1 + x_0^T s_0} > \varepsilon$ **or** $\frac{\|r_q\|}{1 + \|q\|} > \varepsilon$ **do begin**

Prediktor lépés
$a = -xs$;
$(\Delta x, \Delta s)$ kiszámítása (5) alapján
if *M szinguláris* **then**
return *M nem P_0 tulajdonságú;*
end if
α = *prediktorLepeshossz(γ);*
if $\alpha < \alpha_p^*(\kappa)$ **then**
$\kappa(\Delta x)$ kiszámítása a (3) alapján;

if $\kappa(\Delta x)$ *nem létezik* **then**
return *M nem* P_* *tulajdonságú;*
end if
if $\kappa(\Delta x) > \tilde{\kappa}$ **then**
return *M nem* $P_*(\tilde{\kappa})$ *tulajdonságú;*
end if
$\kappa = \kappa(\Delta x)$;
$\alpha_c^*(\kappa) = \frac{2\gamma}{(1+4\kappa)n+1}$;
end if
$x = x + \rho\,\alpha\,\Delta x$;
$s = s + \rho\,\alpha\,\Delta s$;
$\mu = \sigma\,(x^T s)/n$;
$r_q = Mx + q - s$;

Korrektor lépés
$a = \mu e - xs$;
$(\Delta x, \Delta s)$ *kiszámítása (5) alapján;*
if *M szinguláris* **then**
return *M nem* P_0 *tulajdonságú;*
end if
if $(x(\alpha_c^*(\kappa)), s(\alpha_c^*(\kappa)) \notin D(\gamma)$ **then**
$\kappa(\Delta x)$ *kiszámítása a (3) alapján;*
if $\kappa(\Delta x)$ *nem létezik* **then**
return *M nem* P_* *tulajdonságú;*
end if
if $\kappa(\Delta x) > \tilde{\kappa}$ **then**
return *M nem* $P_*(\tilde{\kappa})$ *tulajdonságú;*
end if
$\kappa = \kappa(\Delta x)$;
$\alpha_p^*(\kappa) = \frac{2\sqrt{(1-\gamma)\gamma}}{(1+4\kappa)n+2}$;
end if
$\alpha = korrektorLepeshossz(\gamma, \beta, \beta_{max})$;
$x = x + \rho\,\alpha\,\Delta x$;
$s = s + \rho\,\alpha\,\Delta s$;
$\mu = \sigma\,(x^T s)/n$;
$r_q = Mx + q - s$;
end
END

7.1. A lépéshossz meghatározása

Az algoritmus implementációja során a lépéshossz kiszámítására sajátos módszereket adtunk meg.

7.1.1. A prediktorlépéshossz kiszámítása

A prediktorlépésben a Potra és Liu [2] által megadott módszerből kiindulva valósítottuk meg az α lépéshossz meghatározását, de figyelembe vettük azt is, hogy a lépéshossz őrizze meg az $x > 0$, $s > 0$ feltételeket. A következő függvény szemlélteti ezt:

function *predictorLepeshossz(γ):*

$$\alpha_1 = min\left\{-\frac{x_i}{\Delta x_i} \;\middle|\; \Delta x_i < 0\right\};$$

$$\alpha_2 = min\left\{-\frac{s_i}{\Delta s_i} \;\middle|\; \Delta s_i < 0\right\};$$

$$\alpha = min\{\alpha_1, \alpha_2\};$$

$$u = \frac{xs}{\mu};\quad v = \frac{\Delta x \Delta s}{\mu};$$

$$t = \frac{1-\gamma}{(1+4\kappa)n+1};$$

$$\alpha = min\left\{\alpha, \frac{2}{1+\sqrt{1-4e^T v/n}}\right\};$$

for $i = 1$ **to** n **do begin**
$b = v_i - (1-t)\gamma e^T v/n$;
$c = u_i - (1-t)\gamma$;
$\Delta = c^2 - 4bc$;
if $\Delta > 0$ **and** $b \neq 0$ **then**
$\alpha = min\left\{\alpha, \frac{2(u_i - (1-t)\gamma)}{u_i - (1-t)\gamma + \sqrt{\Delta}}\right\}$;
end if
end
return α;
end

7.1.2. A korrektorlépéshossz kiszámítása

A korrektorlépés hosszúságának meghatározására a [2] cikkben javasolt módszertől eltérő megoldást adtunk meg. A függvény meghatározza a maximális lehetséges α lépéshosszt, amely nem sérti az x, $s > 0$ feltételt. Ezután a $[0, \alpha]$ intervallumot felosztjuk β egyenlő részre, és vizsgáljuk, hogy a részintervallumok végpontjaiban szereplő értékek esetén vett lépéshossz benne van-e a $D(\gamma)$ környezetben. Ha igen, akkor kiszámítjuk az adott értékre vett új (x, s) pont esetén $\mu(\alpha) = \frac{x(\alpha)^T s(\alpha)}{n}$ értékét, majd az lesz a végleges lépéshossz, amelyre $\mu(\alpha)$ minimális.

Ha nem találtunk olyan pontot, amely benne van a környezetben, akkor a β értékét megduplázzuk, és újból elvégezzük a fenti eljárást. A β értékére megadtunk egy β_{max} felső határt, amelyet ha elérünk, hibával megállunk. A módszert a következő függvény segítségével írhatjuk le:

function *korrektorLepeshossz(γ, β,* β_{max}*):*

$$\alpha_1 = min\left\{-\frac{x_i}{\Delta x_i} \;\middle|\; \Delta x_i < 0\right\};$$

$$\alpha_2 = min\left\{-\frac{s_i}{\Delta s_i} \;\middle|\; \Delta s_i < 0\right\};$$

$\alpha = min\{\alpha_1, \alpha_2\}$;
notfound = true;
while $\beta < \beta_{max}$ *and notfound* **do begin**
for $i = 1$ **to** n **do begin**

$\bar{\alpha} = \frac{\alpha i}{\beta}$;
if $\left(x(\bar{\alpha}), s(\bar{\alpha})\right) \in D(\gamma)$ **and** $\mu(\bar{\alpha}) > \mu(\alpha)$ **then**
$\alpha = \bar{\alpha}$;
notfound = *false*;
end if
end
$\beta = 2\beta$;
end
if $\beta \geq \beta_{max}$ **then return** *"hiba"*;
return α;
end

8. Numerikus eredmények

A megvalósítás C++ programozási nyelvben, Visual Studio környezetben, Windows operációs rendszer alatt történt, egy 1,9 GHz-es processzorral rendelkező számítógépen. A használt paraméterek: $\rho = 0.95$, $\sigma = 0.1$, $\varepsilon = 10^{-5}$, $\gamma = 0.9$, $\beta = 100$, $\beta_{max} = 1000$, $\tilde{\kappa} = 10^{40}$. A kezdeti pontoknak az $x_0 = e$ és $s_0 = e$ vektorokat választottuk.

8.1. Elégséges mátrixok

A [7] cikkben leírt módszerrel generált elégséges mátrixokra alapozott feladatokra az általunk implementált algoritmus legtöbb 6 iteráció alatt megtalálta a megoldást. Az **1. táblázat** az átlagos futási időket tartalmazza (másodpercben) azonos méretű feladatok esetén.

Levonhatjuk a következtetést, hogy a feladat méretének a növekedésével a futási idő jelentősen megnövekedett.

8.2. A Csizmadia-mátrix

A Csizmadia Zsolt által bevezetett mátrixcsalád esetén elméleti úton bizonyították [8], hogy a mérettel arányosan a maximális κ exponenciálisan növekszik. Az általunk kapott lokális κ értékek igazolják ezt az elméleti eredményt (lásd a **2. táblázat**ot).

A kapott κ értékek a [9] cikkbeli eredményhez hasonló változást mutatnak a méret függvényében.

1. táblázat. *Átlagos futási idők a méret függvényében a [7]-beli feladatokra*

n	10	20	50
CPU	0.1209	0.1941	0.5876
n	100	200	500
CPU	2.0456	9.9068	131.9007

2. táblázat. *A maximális lokális κ értékek változása a méret függvényében*

n	10	50	100
κ	347.53	$4.65 \cdot 10^{16}$	$1.89 \cdot 10^{34}$

9. Következtetések

Az [1] publikációban bevezetett prediktor-korrektor *GLCP*-re mutattuk be saját megvalósításunkat, korrektorlépés esetén egy sajátos lépéshosszszámító módszert bevezetve. Ismertettük az algoritmusra kapott numerikus eredményeket elégséges mátrixhoz kapcsolódó bemenetekre. A Csizmadia Zsolt által bevezetett mátrixok esetén a κ változását figyeltük meg a méret függvényében.

Köszönetnyilvánítás

A szerzők köszönetüket fejezik ki az Erdélyi Múzeum-Egyesületnek a kutatási munkához nyújtott támogatásért.

Szakirodalmi hivatkozások

[1] Illés T., Nagy M., Terlaky T.: *Polynomial Interior Point Algorithms for General Linear Complementarity Problems*. Algorithmic Operations Research, 5/1. (2010) 1–12.

[2] Potra F. A., Liu X.: *Predictor-Corrector Methods for Sufficient Linear Complementarity Problems in a Wide Neighborhood of the Central Path*. Optimization Methods and Software, 20/1. (2005) 145–168. https://doi.org/10.1080/10556780512331318038

[3] Illés T., Nagy M., Terlaky T.: *Ep Theorem for Dual Linear Complementarity Problems*. Journal of Optimization Theory and Applications, 140. (2009) 233–238. https://doi.org/10.1007/s10957-008-9440-0

[4] Illés T., Nagy M., Terlaky T.: *A Polynomial Path-Following Interior Point Algorithm for General Linear Complementarity Problems*. Journal of Global Optimization, 47/3. (2010) 329–342. https://doi.org/10.1007/s10898-008-9348-0

[5] Darvay Zs., Füstös Á.: *Numerical Results for the General Linear Complementarity Problem*. Műszaki Tudományos Közlemények, 9. (2019) 43–46. https://doi.org/10.33894/mtk-2019.11.07

[6] Kojima M., Megiddo N., Noma T., Yoshise A.: *A Unified Approach to Interior Point Algorithms for Linear Complementarity Problems*. Springer Verlang, Berlin, 1991.

[7] Illés T., Morapitiye S.: *Generating Sufficient Matrices*. In: *Short Papers of the 8th VOCAL Optimization Conference: Advanced Algorithms held in Esztergom, Hungary*. (Szerk.: Friedler F.) Pázmány Péter Katolikus Egyetem, Budapest, 2018. 56–61.

[8] de Klerk E., E.-Nagy M.: *On the Complexity of Computing the Handicap of a Sufficient Matrix*. Mathematical Programming, 129. (2011) 383–402. https://doi.org/10.1007/s10107-011-0465-z

[9] Darvay Zs., Illés T., Povh J., Rigó P. R.: *Feasible Corrector-Predictor Interior-Point Algorithm for $P_*(\kappa)$-Linear Complementarity Problems Based on a New Search Direction*. SIAM Journal on Optimization, 30/3. (2020) 2628–2658. https://doi.org/10.1137/19M1248972