Műszaki Tudományos Közlemények vol. 22. (2025) 27–31.
DOI: Magyar: https://doi.org/10.33895/mtk-2025.22.05
Angol: https://doi.org/10.33894/mtk-2025.22.05

MAGFÜGGVÉNYEN ALAPULÓ BELSŐPONTOS ALGORITMUS MEGVALÓSÍTÁSA SÚLYOZOTT LINEÁRIS KOMPLEMENTARITÁSI FELADATOKRA

IMPLEMENTATION OF AN INTERIOR-POINT ALGORITHM BASED ON A KERNEL FUNCTION FOR WEIGHTED LINEAR COMPLEMENTARITY PROBLEMS

Darvay Zsolt,[1,2] Máthé Edina[1,2]

[1] *Babeş-Bolyai Tudományegyetem, Matematika és Informatika Kar, Kolozsvár, Románia, zsolt.darvay@ubbcluj.ro*

[2] *Erdélyi Múzeum-Egyesület, Matematikai és Informatikai Szakosztály*

Abstract

We consider the solution of weighted linear complementarity problems using interior-point algorithms, where the search directions are determined by a specific kernel function. To achieve a more efficient implementation, we modify the theoretically well-performing algorithm. The resulting methods are analyzed using a code written in the Python programming language. We present numerical results for problems whose solutions, based on previous theoretical findings, would require a very high number of iterations.

Keywords*: interior-point algorithm, kernel function, weighted linear complementarity problem, search direction, Newton's method.*

Összefoglalás

Súlyozott lineáris komplementaritási feladatok megoldását vizsgáljuk olyan belsőpontos algoritmusokkal, melyekre a keresési irányokat egy sajátos magfüggvénnyel határozzuk meg. Hatékonyabb megvalósítás érdekében az elméleti szempontból jól működő algoritmuson módosításokat végzünk. Az így kapott módszereket egy Python programozási nyelvben írt kóddal elemezzük. Numerikus eredményeket mutatunk be olyan feladatokra, amelyeknek a megoldása az eddigi elméleti eredmények alapján csak nagyon magas iterációszámmal volna lehetséges.

Kulcsszavak*: belsőpontos algoritmus, magfüggvény, súlyozott lineáris komplementaritási feladat, keresési irány, Newton-módszer.*

1. Bevezetés

A súlyozott lineáris komplementaritási feladat (WLCP) fogalmát Potra [1] vezette be 2012-ben. A WLCP a következő alakban fogalmazható meg. Keressük azt az $(x, s) \in \mathbb{R}^n \times \mathbb{R}^n$ vektorpárt, amelyre

$$\begin{aligned} -Mx + s &= q, \\ xs &= w, \\ x, s &\geq 0, \end{aligned} \tag{1}$$

ahol $q \in \mathbb{R}^n$ egy adott vektor, $w \geq 0$ egy adott súlyvektor és $M \in \mathbb{R}^{n \times n}$ egy adott mátrix. A WLCP valójában a lineáris komplementaritási feladat (LCP) kiterjesztése, hiszen az a speciális eset, amikor a súlyvektor csupa nullásokat tartalmaz, egy LCP-t eredményez. A WLCP alkalmazási területei rendkívül szélesek. Sajátosan az LCP alkalmazható mechanikai kölcsönhatások vizsgálatára, akadályproblémák esetén vagy szerkezettervezési problémák esetén is [2]. Ezenkívül a WLCP hasznos lehet olyan rendszerekben, ahol a különböző változók vagy feltételek eltérő fontossággal bírnak, valamint gyakran alkalmazzák gazdasági

modellezések során, például a Fisher-féle piaci egyensúlyprobléma esetén is [1].

Az (1) összefüggésben leírt feladat megoldhatóságának biztosításához az M mátrixnak teljesíteni kell bizonyos feltételeket. Ezen feltételek egy olyan változatát vizsgáljuk meg, melyben az M mátrix $P^*(\kappa)$-tulajdonságú. Egy M mátrixról akkor mondjuk, hogy $P^*(\kappa)$-tulajdonságú, ha létezik olyan κ nem negatív valós szám, amelyre tetszőleges $x \in \mathbb{R}^n$ vektor esetén fennáll az alábbi egyenlőtlenség:

$$(1+4\kappa)\sum_{i\in I_+} x_i(Mx)_i + \sum_{i\in I_-} x_i(Mx)_i \geq 0,$$

ahol $I = \{1, 2, ..., n\}$,
$I_+ = \{i \in I: x_i(Mx)_i \geq 0\}$ és
$I_- = \{i \in I: x_i(Mx)_i < 0\}$.

A feladat megoldására belsőpontos algoritmust fogunk alkalmazni. Szükségünk lesz néhány jelölésre. Az $\mathbb{R}^n_+$ jelöli a nem negatív valós n-dimenziós számok halmazát, míg $\mathbb{R}^n_{++}$ a pozitívokét. Az $\mathcal{F}$ fogja megadni a megengedett megoldások halmazát, míg az $\mathcal{F}^0$ halmaz a szigorúan megengedett párokat tartalmazza:

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

Ezenkívül adott $(x^0, s^0) \in \mathcal{F}^0$ esetén meghatározzuk a súlyozott utat egy $t \in [0,1]$ paraméter függvényében. Ennek érdekében vezessük be az alábbi jelölést:

$$w(t) = (1-t)w + tx^0s^0. \qquad (2)$$

Ha az (1) rendszerben a súlyvektort helyettesítjük az előzőekben megadott $w(t)$ kifejezéssel, akkor egy olyan megoldható rendszert kapunk, amely meghatározza a súlyozott centrális utat:

$$\begin{aligned} -Mx + s &= q, \\ xs &= w(t), \\ x, s &\geq 0. \end{aligned} \qquad (3)$$

Ha M egy $P^*(\kappa)$-mátrix és $\mathcal{F}^0 \neq \emptyset$, akkor a fenti paraméterezett egyenletrendszernek bármilyen $t \in [0,1]$ paraméterre egyértelmű megoldása van, melyet $(x(t), s(t))$-vel jelölünk. Ezen megoldások halmaza fogja megadni a feladathoz rendelt súlyozott centrális utat.

Ha Newton módszerét alkalmazzuk a (3) egyenletrendszerre, az alábbi rendszert kapjuk:

$$\begin{aligned} -M\Delta x + \Delta s &= 0, \\ s\Delta x + x\Delta s &= w(t) - xs, \end{aligned} \qquad (4)$$

amelyből meghatározhatóak a keresési irányok. A továbbiakban bevezetjük a $v = \sqrt{\frac{xs}{w(t)}}$ variánciavektort, amelyet a súlyozott centrális úttól való eltérés mértékének meghatározására használhatunk.

A feladat megoldására a magfüggvényen alapuló belsőpontos algoritmusok módszerét fogjuk használni. Egy ψ függvényt magfüggvénynek nevezünk, ha teljesíti az alábbi feltételeket:

$\psi: \mathbb{R}_{++} \to \mathbb{R}_+$ kétszer folytonosan differenciálható, $\psi(1) = \psi'(1) = 0$ és $\psi''(t) > 0$ minden $t > 0$ esetén. Egy adott magfüggvényre a hozzá kapcsolódó $\Psi: \mathbb{R}^n_{++} \to \mathbb{R}_+$ barrierfüggvényt az alábbi módon definiáljuk:

$$\Psi(v) = \sum_{i=1}^n \psi(v_i).$$

Magfüggvények segítségével megadható a keresési irányok rendszere:

$$\begin{aligned} -M\Delta x + \Delta s &= 0, \\ s\Delta x + x\Delta s &= -w(t)\, v \nabla \Psi(v). \end{aligned} \qquad (5)$$

Észrevehető, hogy a $\psi(t) = \frac{t^2-1}{2} - \log(t)$ logaritmikus magfüggvényt használva az (5) rendszerben, visszakapjuk az eredeti (4) egyenletrendszert, melyet a Newton-módszer alkalmazásával kaptunk.

A továbbiakban a $\psi(t) = \frac{t^2-1}{2} + \frac{t^{-1}-1}{2} - \frac{t-1}{2}$ magfüggvényt fogjuk használni a keresési irányok meghatározására. Erre vonatkozó részletes elméleti elemzést a Chi, Wang és Lesaja [3] által publikált cikkben találunk.

2. Elméleti algoritmus

A [3] cikkben bevezetett elméleti algoritmust mutatjuk be, majd ezt követően az implementáció szempontjából elvégzett módosításokat tárgyaljuk. Az algoritmus bizonyos lépéseit külön függvényekkel adjuk meg. A keresési irányok meghatározása érdekében behelyettesítjük a ψ magfüggvényt az (5) lineáris egyenletrendszer jobb oldalán szereplő kifejezésbe, ezt követően megoldjuk a rendszert, és visszatérítjük a kapott irányokat. Ezeket a következő függvénnyel végezzük:

function ***keresésiIrány*** (x, s, t)
begin
  Megoldjuk az (5) egyenletrendszert.
  return $(\Delta x, \Delta s)$;
end.

A teljes Newton-lépés végrehajtása érdekében az aktuális pontokhoz hozzáadjuk a keresési irányokat:

function ***teljesNewtonLépés*** $(x, s, \Delta x, \Delta s)$
begin
  return $(x, s) + (\Delta x, \Delta s)$;
end.

A súlyozott centrális utat meghatározó paraméter frissítését az alábbi függvénnyel végezhetjük:

function ***frissítThetaSegítségével*** (t, θ)
begin
  return $(1-\theta)t$;
end.

Mindezeket felhasználva az alábbi módon adható meg az elméleti algoritmus.

1. algoritmus (Chi, Wang és Lesaja [3])
Adott az $(x^0, s^0) \in \mathcal{F}^0$ vektorpár, amelyre $x^0 s^0 \geq w$.
Legyen $\varepsilon > 0$ a pontossági paraméter.
Legyen $\theta = \bar{\theta}_{min}$ a frissítési paraméter, ahol

$$\bar{\theta}_{min} = \frac{\gamma}{3\gamma + (18\bar{\kappa} + 10)\beta},$$

$$\bar{\kappa} = \frac{(1 + 4\kappa) max(x^0 s^0)}{4\, min(w)} - 0.25,$$

$\gamma = min(w)$, $\beta = \|x^0 s^0 - w\|$.
Legyen $\tau > 0$ a küszöbparaméter.
Legyen $t^0 = 1$, úgy, hogy $\delta(x^0, s^0; t^0) \leq \tau$, ahol

$$\delta(x, s; t) = \left\| v\left(\frac{\Delta x}{x} + \frac{\Delta s}{s}\right)\right\| = \left\| \frac{e}{2} + \frac{e}{2v^2} - v \right\|.$$

$x = x^0$, $s = s^0$, $t = t^0$;
while $\|xs - w\| > \varepsilon$ **do**
begin
  $(\Delta x, \Delta s) =$ ***keresésiIrány*** (x, s, t);
  $(x, s) =$ ***teljesNewtonLépés*** $(x, s, \Delta x, \Delta s)$;
  $t =$ ***frissítThetaSegítségével*** (t, θ);
end.

A hatékony megvalósítás érdekében az 1. algoritmuson számos módosítást végeztünk.

3. Módosított algoritmus

Mivel az algoritmus iterációi során megtörténhet, hogy nem megengedett pontba jutunk, bevezetjük az

$r_q = -Mx + s - q$

hibavektort. A keresési irányokat az (5) rendszer helyett az alábbi módosított rendszer segítségével adjuk meg:

$$-M\Delta x + \Delta s = -r_q,$$
$$s\Delta x + x\Delta s = -w(t)v\nabla\Psi(v). \qquad (6)$$

Az ehhez rendelt függvény pedig a következő:

function ***módosítottKeresésiIrány*** (x, s, t)
begin
  $r_q = -Mx + s - q$;
  Megoldjuk a (6) egyenletrendszert.
  return $(\Delta x, \Delta s)$;
end.

Nem teljes Newton-lépéssel lépünk tovább a következő pontra, hanem egy α változóban kiszámoljuk a maximális lépést a határig, majd ezt csökkentjük egy $\rho \in (0, 1)$ tényező segítségével:

function ***NewtonLépés*** $(x, s, \Delta x, \Delta s, \rho)$
begin
  $\alpha_x = min\left\{\frac{-x_i}{\Delta x_i} \middle| \Delta x_i < 0\right\}$;
  $\alpha_s = min\left\{\frac{-s_i}{\Delta s_i} \middle| \Delta s_i < 0\right\}$;
  $\alpha = min\{\alpha_x, \alpha_s\}$;
  return $(x, s) + \rho\alpha(\Delta x, \Delta s)$;
end.

A t paraméter csökkentése érdekében két módszert valósítottunk meg. Az első megegyezik az elméleti algoritmusban megadott változattal, amely a θ segítségével csökkenti a t értékét:

function ***frissítParaméter*** (t, θ, x, s, ζ)
begin
  return ***frissítThetaSegítségével*** (t, θ);
end.

A második variáns a [4] cikkben bevezetett módszer segítségével frissíti a t paramétert, egy $\zeta \in (0,1]$ skálázási tényezőt felhasználva:

function ***frissítParaméter*** (t, θ, x, s, ζ)
begin
  return ***frissítZetaSegítségével*** (x, s, ζ);
end
function ***frissítZetaSegítségével*** (x, s, ζ)
begin
  return $t = \zeta \left| \frac{x^T s - w^T e}{(x^0)^T s^0 - w^T e} \right|$;
end.

Feltételezve, hogy $x^0 s^0 \neq w$, megállapíthatjuk, hogy a fenti kifejezés nevezője nem lehet 0. A továbbiakban bemutatjuk a módosított algoritmust.

2. algoritmus. Adott az $(x^0, s^0) \in \mathcal{F}^0$ pontpár úgy, hogy $x^0 s^0 \geq w$, $x^0 s^0 \neq w$.
Legyen $\varepsilon > 0$ a pontossági paraméter. Továbbá $\theta \in (0, 1)$, illetve $\zeta \in (0, 1]$ frissítési paraméterek.
Legyen $\rho \in (0, 1)$ a lépés skálázási paramétere.
$t^0 = 1$, $x = x^0$, $s = s^0$, $t = t^0$;
while $\|xs\text{-}w\| > \varepsilon$ **do**
begin
  $(\Delta x, \Delta s) =$ ***módosítottKeresésiIrány*** (x, s, t);
  $(x, s) =$ ***NewtonLépés*** $(x, s, \Delta x, \Delta s, \rho)$;
  $t =$ ***frissítParaméter*** (t, θ, x, s, ζ);
end.

A fenti algoritmust felhasználva, különböző numerikus eredményeket mutatunk be.

4. Numerikus eredmények

Az előbbiekben bemutatott algoritmus Python programozási nyelven került implementálásra, felhasználva a NumPy könyvtárat a lineáris algebrai műveletek elvégzéséhez. Minden futtatás során az

$$M = \begin{pmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & \dots & 0 \\ -1 & -1 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ -1 & -1 & -1 & \dots & 1 \end{pmatrix}$$

Csizmadia-mátrixot [5] használtuk, amelyre a [6] cikkben igazolták, hogy $n \geq 4$ esetén $\kappa = 2^{2n-8} - 0.25$. Ennek a mátrixnak azért van nagyon fontos szerepe a $P^*(\kappa)$-WLCP vizsgálatában, mert a κ érték a feladat mérete függvényében exponenciálisan növekszik.

Azt is feltételeztük, hogy $q = [0, 1, ..., n-1]^T$. Továbbá az $x^0 = s^0 = e$ kezdeti értékekkel dolgoztunk, ahol e a csupa egyesekből álló n-dimenziós vektor. Figyeljük meg, hogy ebben az esetben fennáll az $(x^0, s^0) \in \mathcal{F}^0$ feltétel.

Az $x^0 s^0 \geq w$ egyenlőtlenség teljesítése érdekében a w súlyvektor értékeit véletlenszerűen generált, egynél kisebb valós pozitív számoknak választottuk. Mivel több szempontból megvizsgáltuk az algoritmust, ezért különböző megszorítások mellett generáltuk a w elemeit. Továbbá az $\varepsilon = 10^{-6}$, $\rho = 0.9$ értékeket használtuk. Ami a θ paramétert illeti, az elméleti szinten meghatározott $\bar{\theta}_{min}$ értéknél nagyobbakat is megvizsgáltuk, az alábbi halmazból:

$\theta \in \{\bar{\theta}_{min}, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 0.01, 0.02, ..., 0.1, 0.2, ..., 0.9\}$.

Azt tapasztaltuk, hogy általában a θ növelése esetén is helyes eredményt kapunk, az algoritmus pedig gyorsabban konvergál a megoldáshoz. Ennek érdekében a 4×4-es Csizmadia-mátrix esetén tekintsük a kapott iterációszámokat, illetve CPU-időket másodpercben kifejezve. A futtatásokhoz egy Intel Core i5-1035G1 processzorral és 8 GB RAM-mal rendelkező gépet használtunk, Windows 10 operációs rendszer alatt, Python 3.11.6-os környezetben (1. táblázat).

A következő táblázatban a különböző θ értékekre kapott minimális iterációszámot mutatjuk be a dimenzió függvényében. Látható, hogy a dimenzió növekedésével lineárisan nő a minimális iterációszám. Az utolsó sorban feltüntettük, hogy mely θ értéktől kezdve kapjuk ezt a minimális iterációszámot. Megállapítható, hogy a nagyobb dimenziók esetében az iterációszám nem minden esetben csökkenthető a θ értékének növelésével (2. táblázat).

A további eredményeket azzal az implementációval kaptuk, melyben a t paraméter csökkentését egy ζ skalár segítségével végezzük. Ennek a módszernek a segítségével is hasonló eredményeket értünk el a 2. algoritmus iterációszámát tekintve (1. ábra).

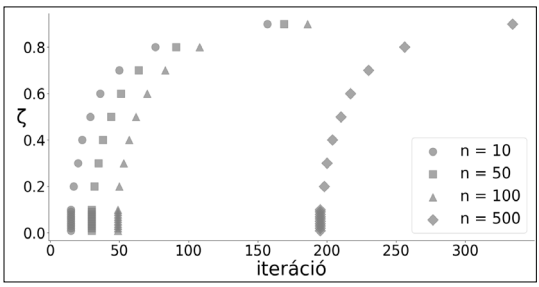
A 2. ábra az $n = 10$, $\zeta = 0.1$ esetben azt hivatott bemutatni, hogy a súlyvektor különböző intervallumokba való kategorizálása hogyan befolyásolja a megoldásban szereplő s értékeket. Tudjuk, hogy ha a súlyvektor nulla, akkor visszakapjuk az s vektorban a q vektor által reprezentált növekvő természetes számokat. Abban az esetben, ha a w értékei nullához közeli intervallumba es-

1. táblázat. *Az iterációszám θ függvényében.*

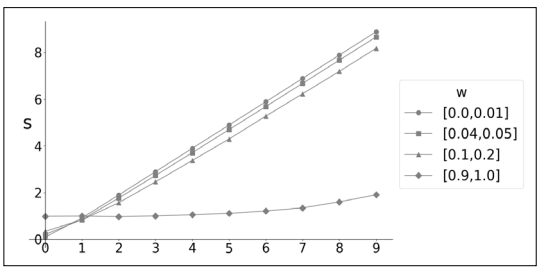
Θ	Iterációszám	CPU
0.1	139	0.0156
0.01	1445	0.0625
10^{-3}	14498	0.7031
10^{-4}	145029	6.8281
10^{-5}	1450342	107.3906
10^{-6}	14503470	1035.1250
$\bar{\theta}_{min} = 3.86 \cdot 10^{-7}$	37573477	3006.2031

2. táblázat. *Minimális iterációszám a dimenzió függvényében, ahol D a dimenziót és I a minimális iterációszámot jelöli.*

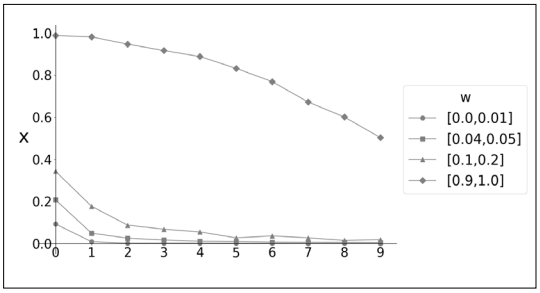
D	20	30	40	50	100	200	300	400	500
I	18	22	26	29	48	84	121	158	195
Θ	0.7	0.6	0.5	0.5	0.3	0.2	0.2	0.2	0.09



1. ábra. *Az iterációszám ζ függvényében*



2. ábra. *Az s vektor értékei a súlyvektor függvényében*



3. ábra. *Az x vektor értékei a súlyvektor függvényében*

nek, megkapjuk a q vektor közelítő értékeit. Ha a súlyvektor értékei számára olyan intervallumot választunk, melynek elemei a nullától távolabb esnek, akkor az s értékei is eltolódnak.

Ez a tendencia fellelhető az x esetében is, viszont ennek a vektornak az elemei nullához konvergálnak, ha a súlyvektor értékeit nullához közelebbinek választjuk (**3. ábra**).

5. Következtetések

A Chi, Wang és Lesaja **[3]** által publikált magfüggvényre alapozott belsőpontos algoritmust módosítottuk a $P^*(\kappa)$-WLCP hatékony megoldása érdekében. A súlyozott centrális utat meghatározó paraméter csökkentésére vonatkozóan két változatot vizsgáltunk. Az általunk fejlesztett Python-kódot felhasználva különböző numerikus eredményeket mutattunk be.

Köszönetnyilvánítás

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

Szakirodalmi hivatkozások

[1] Potra F. A.: *Weighted Complementarity Problems – a New Paradigm for Computing Equilibria*. SIAM Journal on Optimization, 22/4. (2012) 1634–1654. https://doi.org/10.1137/110837310

[2] Ferris M. C., Pang J. S.: Engineering and Economic Applications of Complementarity Problems. SIAM Review, 39/4. (1997) 669–713. https://doi.org/10.1137/S0036144595285963

[3] Chi X., Wang G., Lesaja G.: *Kernel-Based Full-Newton Step Feasible Interior-Point Algorithm for $P^*(\kappa)$-Weighted Linear Complementarity Problem.* Journal of Optimization Theory and Applications, 202/1. (2024) 108–132. https://doi.org/10.1007/s10957-023-02327-9

[4] Darvay Zs., Orbán A.-Sz.: *Implementation of the Full-Newton Step Algorithm for Weighted Linear Complementarity Problems*. Papers on Technical Science, 15. (2021) 15–18. https://doi.org/10.33894/mtk-2021.15.04

[5] 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

[6] E.-Nagy M., Illés T., Povh J., Varga A., Zerovnik J.: *Sufficient Matrices: Properties, Generating and Testing*. Journal of Optimization Theory and Applications, 202/1. (2024) 204–236. https://doi.org/10.1007/s10957-023-02280-7