POZITÍV MÁTRIXOK DOMINÁNS SAJÁTVEKTORÁNAK SZÁMÍTÁSA A CIKLIKUS KOORDINÁTÁK MÓDSZERÉVEL[1]

ÁBELE-NAGY KRISTÓF – FÜLÖP JÁNOS
Budapesti Corvinus Egyetem – SZTAKI

Pozitív mátrixok domináns sajátértékének és sajátvektorának számítása több alkalmazásban fontos feladat. Kisméretű mátrixok esetén ez egy gyorsan megoldható feladat, nagyméretű mátrixok esetén azonban lassú lehet. Egy új iteratív algoritmust mutatunk be, amely a ciklikus koordináták módszerén alapul, és kifejezetten nagyméretű mátrixokra van szabva.

Kulcsszavak: domináns sajátérték, pozitív mátrix, ciklikus koordináták módszere, hatvány módszer, Collatz–Wielandt-tétel

1 Bevezetés

Pozitív, illetve általánosabb esetben irreducibilis (ld. 1. Definíció) mátrixok domináns sajátértékének és sajátvektorának számítására számos alkalmazásban szükség van. MacCluer [13] jó néhány alkalmazást felsorol: ezek a Markov-láncoktól kezdve a Leontief input/output modellen át a walrasi egyensúlyi modellig terjednek, valamint számos egyéb alkalmazást is említ (például topológia, epidemiológia és statisztikai mechanika). Ezeken túl egy igen fontos alkalmazás a többszempontú döntéselmélet [18], melyről bővebben is szó lesz.

Míg kisméretű mátrixok esetén gyakorlatilag mindegy, milyen módszert alkalmazunk a domináns sajátérték, illetve sajátvektor kiszámítására, nagyméretű mátrixok esetén az alkalmazott módszerek igen lassúak lehetnek. Az alapvető módszer domináns sajátérték és sajátvektor számításra a hatvány módszer [10, 105–110. oldal], mely általános esetben diagonalizálható mátrixokra működik, azonban nemnegatív mátrixok esetén az irreducibilitás (1. Definíció) elég. A hatvány módszer egy $\mathbf{A}$ $n \times n$-es mátrix esetén a $\mathbf{v}^{(k+1)} = \mathbf{A}\mathbf{v}^{(k)}$ szorzatot számolja minden lépésben (ahol $\mathbf{v}$ az iterált vektor). A konvergencia sebessége az abszolút értékben második legnagyobb sajátérték és a domináns sajátérték hányadosától függ, ami nem ellenőrizhető könnyen előre. Más módszerek, mint például az LU és QR algoritmusok [10, 110–118. oldal] az összes sajátértéket számolják, ám jóval nagyobb műveleti költséggel.

1. Definíció ([14, 671. oldal]). Egy $\mathbf{A}$ négyzetes mátrix irreducibilis, ha nem hozható sorok és oszlopok egyszerre történő cseréivel a következő blokk

[1]E-mail: kristof.abele-nagy@uni-corvinus.hu, fulop.janos@sztaki.hu. Beérkezett: 2019. augusztus 3.

felső háromszög alakra:

$$\begin{pmatrix} \mathbf{B} & \mathbf{C} \\ \mathbf{0} & \mathbf{D} \end{pmatrix}$$

ahol $\mathbf{B}$ és $\mathbf{D}$ nem 0 méretű négyzetes mátrixok.

Jelölje $\lambda_{\max} = \lambda_{\max}(\mathbf{A})$ a domináns sajátértéket, $\mathbf{w}^*$ pedig a hozzá tartozó domináns jobb oldali sajátvektort, azaz $\mathbf{A}\mathbf{w}^* = \lambda_{\max}\mathbf{w}^*$. Ekkor az alábbi összefüggés teljesül:

1. Tétel (Collatz–Wielandt, [14, 666. és 673. oldal]). *Legyen* $\mathbf{A} \geq 0$ *egy irreducibilis* $n \times n$*-es mátrix. Ekkor* $\lambda_{\max} > 0$*, és (pozitív skalárral való szorzástól eltekintve) egyetlen pozitív sajátvektor tartozik hozzá, valamint*

$$\lambda_{\max} = \max_{\mathbf{w}>0} \min_{i=1,\ldots,n} \frac{(\mathbf{A}\mathbf{w})_i}{w_i} \tag{1}$$

$$\lambda_{\max} = \min_{\mathbf{w}>0} \max_{i=1,\ldots,n} \frac{(\mathbf{A}\mathbf{w})_i}{w_i}. \tag{2}$$

Az 1. Tételt az irodalomban gyakran mint Perron–Frobenius tételt, vagy mint annak egy részét mutatják be.

Innentől feltesszük, hogy $\mathbf{A} > 0$, tehát szigorúan pozitív mátrixról van szó. Az 1. Definícióból adódóan minden pozitív mátrix irreducibilis.

Az ebben a dolgozatban javasolt algoritmus a (2) formulát használja a $\lambda_{\max}$ közelítésére, azonban ez a választás önkényes: az eljárás könnyen átalakítható úgy, hogy az (1) összefüggést használja. Később azonban mindkét alakot felhasználjuk a megállási kritérium meghatározásához.

Az eljáráshoz a ciklikus koordináták módszerét [12, 253–254. oldal] alkalmazzuk, mely egy numerikus optimalizálási módszer. A ciklikus koordináták módszerének lényege, hogy egy többváltozós optimalizálási feladatban egyszerre mindig csak egy változót tekintünk ténylegesen változónak, a többi változó az előző lépésben számolt értéken van rögzítve. Annak az elemnek, amelyik ténylegesen változik, az új értéke az a szám lesz, ahol az optimalizálási feladat a többi elem változatlansága melletti optimumát felveszi. Azt, hogy melyik változót tekintjük egy adott lépésben ténylegesen változónak, ciklikusan változtatjuk, azaz először az elsőt, majd a másodikat stb., majd amikor az utolsó változón is túl vagyunk, visszaugrunk az elsőre és ugyanígy folytatjuk, amíg el nem érjük a leállási kritériumot. Az, hogy mi a leállási kritérium, a feladattól függ.

Az (1) és (2) feladatban szereplő lineáris törtfüggvények kvázilineárisak, azaz kvázikonvexek és kvázikonkávak is [2]. Mivel kvázikonvex függvények pontonkénti maximuma szintén kvázikonvex, illetve kvázikonkáv függvények pontonkénti minimuma kvázikonkáv, ezért az (1) feladat egy kvázikonkáv függvény maximalizálását, a (2) pedig egy kvázikonvex függvény minimalizálását jelenti az n-dimenziós pozitív ortáns felett. A [2] könyvben bemutatott 4.1 Algoritmus, amely egy felezéses eljárás, alkalmas az (1)-(2) feladat megoldására, viszont az egy általános eljárás, amely nem használja ki a feladat speciális tulajdonságait.

A (2) feladat minimalizálandó célfüggvénye sajnos nemcsak, hogy nem konvex, hanem nem is folytonosan differenciálható, tehát a konvex optimalizálási algoritmusok nagy része nem alkalmazható rá közvetlenül. Ezen hátrányok viszont kivédhetők a feladat speciális alakját kihasználva. Legyen $v_i = \log w_i$, azaz $w_i = \exp(v_i), i = 1, \ldots, n$. Ezt a koordináta-transzformációt alkalmazva (2) lineáris törtfüggvényei konvex exponenciális függvények összegének alakját öltik, ezek pontonkénti maximuma viszont szigorúan konvex függvény lesz. Viszont ez is lehet még nem folytonosan differenciálható. Ugyanakkor, mivel véges számú sima konvex függvény maximumát kell minimalizálni, a feladat a diszkrét minimax feladatok osztályába tartozik. [16, 2.3 fejezet] az ilyen típusú feladatokra mutat be gradiens alapú algoritmusokat. Ezeknél az algoritmusoknál azonban az iránymenti keresésnél a lépéshossz meghatározása nem végezhető el hatékonyan, szemben az általunk ebben a tanulmányban javasolt módszerrel. A ciklikus koordináták módszere különösen hatékonynak bizonyult nagyon nagy méretű feladatok megoldásánál [15].

2 Az algoritmus

A változóink az eljárásban a domináns sajátvektor, $\mathbf{w}^*$ elemei: $w_1^*, \ldots, w_n^*$. Ezek aktuális értékét a továbbiakban $\mathbf{w} = (w_1, \ldots, w_n)^\top$ jelöli. A ciklikus koordináták módszere, az előző részben leírtak szerint minden lépésben csak egy változót tekint ténylegesen változónak. Jelölje ennek a változónak az indexét k, így minden lépésben w_k lesz az aktuális változónk, míg a többi változó értéke ideiglenesen azok előző lépésben számolt értékein van rögzítve.

A fentiek alapján (2) szerint minden lépésben a változó, w_k új értéke az a $\hat{w}_k$ érték lesz, amire teljesül, hogy

$$\hat{w}_k = \arg\min_{w_k} \max_{i=1,\ldots,n} \frac{(\mathbf{Aw})_i}{w_i}. \tag{3}$$

Mivel a többi, w_j, $j \neq k$ érték fix, ezért a (3) kifejezés minden i-re csupán w_k-tól függ. Így bevezethetjük a következő jelölést. Legyen

$$f_i(w_k) = \frac{(\mathbf{Aw})_i}{w_i} = \frac{a_{i1}w_1 + \ldots + a_{ik}w_k + \ldots + a_{in}w_n}{w_i}, \quad i = 1, \ldots, n. \tag{4}$$

Amit keresünk tehát egy lépésben, az a w_k új értéke $\hat{w}_k > 0$, amelyre

$$\hat{w}_k = \arg\min_{w_k} \max_{i=1,\ldots,n} f_i(w_k),$$

azaz ahol az f_i függvények felső burkolójának a minimumpontja van. Az $f_i(\hat{w}_k)$ függvényérték pedig a $\lambda_{\max}$ közelítése (felső korlátja) lesz.

Mivel

$$f_i(w_k) = \frac{(\mathbf{Aw})_i}{w_i} = \sum_{j=1}^{n} \frac{a_{ij}w_j}{w_i}, \quad i = 1, \ldots, n, \tag{5}$$

ezért a vizsgált f_i függvények $i \neq k$-ra affin (lineáris plusz konstans) függvények, hiszen a változó, w_k, csak az n tagú összeg számlálójában szerepel.

Hasonlóan, $i = k$-ra $f_k(w_k)$ egy hiperbolikus függvény, mert w_k a nevezőben szerepel, illetve egyszer a számlálóban is, ami ebben a tagban így 1-re egyszerűsödik.

Az f_i függvények tehát felírhatóak a következő alakban:

$$f_i(w_k) = \frac{(\mathbf{Aw})_i}{w_i} = a_i w_k + b_i, \quad i \neq k \tag{6}$$

valamilyen $a_i > 0$ és $b_i > 0$, k-tól függő konstansokkal. Hasonlóan, $i = k$-ra

$$f_k(w_k) = \frac{(\mathbf{Aw})_k}{w_k} = \frac{c}{w_k} + d \tag{7}$$

valamilyen $c, d > 0$, k-tól függő konstansokkal. A konstansok egzakt alakja meghatározható az (5) formula alapján:

$$a_i = \frac{a_{ik}}{w_i} \tag{8}$$

$$b_i = \sum_{j \neq k} \frac{a_{ij} w_j}{w_i} \tag{9}$$

$$c = \sum_{j \neq k} a_{kj} w_j \tag{10}$$

$$d = a_{kk}. \tag{11}$$

A következő állítás a felső burkoló minimumpontjának meghatározásához visz közelebb:

1. Állítás. *A (3) összefüggést teljesítő $\hat{w}_k$-ra*

$$f_i(\hat{w}_k) = f_k(\hat{w}_k) \tag{12}$$

valamilyen $i \neq k$-ra.

Bizonyítás. Mivel $a_i, b_i > 0$ $\forall i \neq k$-ra a (6) képletben, ezért az $f_i(w_k)$, $i \neq k$ affin függvények szigorúan monoton növekvők, így a felső burkolójuk is szigorúan monoton növekvő. Hasonlóan, mivel $c, d > 0$ a (7) képletben, és mert $w_k > 0$, az $f_k(w_k)$ hiperbolikus függvény szigorúan monoton csökkenő.

Tekintsük a következő, elemi úton adódó határértékeket:

$$\lim_{w_k \to 0+} f_k(w_k) = \infty$$

$$\lim_{w_k \to \infty} f_k(w_k) = d < \infty$$

$$\lim_{w_k \to \infty} \max_{i \neq k} f_i(w_k) = \infty,$$

továbbá

$$\max_{i \neq k} f_i(0) < \infty$$

is teljesül.

Ezek alapján, és mivel az $f_i(w_k), i = 1, \ldots, n$ függvények folytonos, szigorúan monoton függvények, az $f_i(w_k), i \neq k$ függvények felső burkolója és az $f_k(w_k)$ egyetlen $\hat{w}_k$ pontban metszik egymást. Így az előbbi monotonitási tulajdonságok szerint a $\max_{i=1,\ldots,n} f_i(w_k)$ minimumpontja ebben a metszéspontban, $\hat{w}_k$-ban van, ahol $\max_{i\neq k} f_i(\hat{w}_k) = f_k(\hat{w}_k)$. Mivel $\max_{i\neq k} f_i(w_k)$ véges sok affin függvény felső burkolója, ezért létezik egy $i \neq k$, amire $f_i(\hat{w}_k) = f_k(\hat{w}_k)$. □

Az 1. Állításból következik, hogy elegendő csak az f_k függvény metszéspontjait kiszámolni minden f_i, $i \neq k$ függvénnyel. Az f_k szigorúan monoton csökkenése miatt, az a $\hat{w}_k > 0$, amelyik eleget tesz a (3) feltételnek, az a legkisebb w_k lesz, amelyik eleget tesz a (12) összefüggésnek. A következő állítás a metszéspont kiszámítására vonatkozik.

2. Állítás. *Az $f_i(w_k) = f_k(w_k)$ metszéspont $w_k > 0$ helye*

$$w_k = \frac{d - b_i + \sqrt{(b_i - d)^2 + 4a_i c}}{2a_i}.$$

Bizonyítás. A (6) és (7) képleteket a (12) összefüggésbe helyettesítve a következőt kapjuk:

$$a_i w_k + b_i = \frac{c}{w_k} + d.$$

Ezt átrendezve

$$a_i w_k^2 + (b_i - d) w_k - c = 0.$$

A másodfokú megoldóképletet alkalmazva

$$w_k = \frac{d - b_i \pm \sqrt{(b_i - d)^2 + 4a_i c}}{2a_i}.$$

Mivel $\sqrt{(b_i - d)^2 + 4a_i c} > d - b_i$ csak a $+\sqrt{(b_i - d)^2 + 4a_i c}$ tagot tartalmazó képlet ad pozitív megoldást. □

Az eddig leírtak képezik az algoritmus magját. Pár további részleten kívül, mint a megállási kritérium és az induló értékek, ez már elegendő az implementációhoz. További gyorsítási lehetőségeket is kihasználhatunk, melyek különösen nagyméretű mátrixok esetén lehetnek jelentősek. Az eredeti célunk is az algoritmus nagyméretű mátrixokra szabása volt. A következő gyorsítási lehetőségek kisméretű mátrixok esetén valószínűleg valójában lassítások, mivel olyan elemeket tartalmaznak, amelyeknek konstans vagy a mérettel lassan növekedő a műveletigényük. Nagyméretű mátrixok esetén azonban ezek már a futásidő elenyésző hányadát teszik ki és a használatukból eredő előnyök már összességében gyorsítást eredményeznek.

Figyeljük meg, hogy nem feltétlenül szükséges az összes metszéspontot kiszámolnunk: csak az affin függvények maximumának a hiperbolikus függvénnyel vett metszéspontjára van szükségünk. Így azok az affin függvények, amelyeknek nincs közös pontja az affin függvények felső burkolójával, érdektelenek. Formálisan, ha létezik $i \neq k$ és $j \neq k$, amelyre $a_i \geq a_j$ és

$b_i \geq b_j$ a (6) képletben, valamint a két egyenlőtlenség közül legalább az egyik szigorú, akkor $f_i(w_k) > f_j(w_k)$ minden $w_k > 0$-ra. Így f_j-nek nem lesz közös pontja $\max_{i \neq k} f_i(w_k)$-val, azaz f_k-nak az f_j-vel vett metszéspontját nem szükséges kiszámolni.

Szemléletesen ez azt jelenti, hogy ha egy egyenes a pozitív síknegyeden teljes egészében egy másik „felett" halad, akkor a „lejjebb lévő" egyenest elhagyhatjuk. Ilyen pontosan akkor következik be, hogy ha két egyenesnek nincsen a pozitív síknegyedben metszéspontja.

Ha a metszéspontot minden f_i, $i \neq k$-ra kiszámolnánk, akkor $n-1$ metszéspontot kellene kiszámolnunk. Ha csak azokra az f_i-kre számoljuk ki, amelyek nem érdektelenek, akkor potenciálisan jóval kevesebb, mint $n-1$ metszéspontot szükséges kiszámolnunk, amely különbség főleg nagyméretű mátrixok esetén lehet látványos.

Térjünk most rá a megállási kritériumra. A megállási kritérium a $\lambda_{\max}$ (2) és (1) szerinti minimax és maximin tulajdonságát is felhasználja. Minden iteráció végén, w_k meghatározása után kiszámoljuk az

$$f_i(w_k) = f_k(w_k) = a_i w_k + b_i = \frac{c}{w_k} + d = \frac{(Aw)_i}{w_i} = \frac{(Aw)_k}{w_k} \tag{13}$$

értéket a megfelelő i-vel, azaz azzal, amelyikre f_i-nek az f_k-val való metszéspontja a legközelebb van 0-hoz. A fenti (13) egyenlőségsorozatban az első egyenlőség az 1. Állítás miatt igaz, míg az azt követő formák a (12) összefüggés átfogalmazásai (6), (7), illetve (4) alapján. Így kapjuk a (3) feltételnek eleget tevő $\hat{w}_k$ értéket. Az alsó burkoló maximumát, a $\max_{w'_k} \min_{i=1,\dots,n} \frac{(Aw)_i}{w_i}$ értéket és az ahhoz tartozó $\hat{w}'_k$ pontot is hasonlóan számoljuk, tulajdonképpen a fenti számolások tükörképeivel. Pontosabban az f_k hiperbola összes olyan $f_i, i \neq k$-val vett metszéspontját, azaz az $f_k(w'_k) = f_i(w'_k)$ egyenlőségnek eleget tevő értéket kiszámoljuk, ami minimum értelemben nem érdektelen, azaz amire nincs olyan másik egyenes, ami „alatta" halad. Ezután ezek közül azt a $\hat{w}'_k$ értéket választjuk, ami a legnagyobb, azaz ami a legmesszebb van 0-tól. Így $\hat{w}'_k$-re

$$\max_{w'_k} \min_{i=1,\dots,n} \frac{(Aw')_i}{w'_i} = f_k(\hat{w}'_k) \tag{14}$$

teljesül az aktuális iterációban. Az algoritmus megáll, ha (3) és (14) közelebb vannak egymáshoz, mint egy előre meghatározott küszöbérték.

A fentiekből látható, hogy a megállási kritérium ellenőrzése az alsó burkoló maximumának számolása miatt többletszámításokat igényel. Annak érdekében, hogy ezeknek a számításoknak a számát csökkentsük, két fázisra bontjuk az algoritmust. A fenti (14) értéket, azaz az alsó burkoló maximumát csak a második fázisban számoljuk. Az első fázisban csak a (3) értékét, azaz a felső burkoló minimumát számoljuk, és azt ellenőrizzük, hogy az előző iterációban felvett értékéhez képest többet változik-e, mint egy előre meghatározott küszöbérték. Ha nem, akkor a második fázisba lépünk, és onnantól kezdve már a (14) értékét is számoljuk, és a két értéket egymással hasonlítjuk össze. A

két fázisra bontásnak köszönhetően az algoritmus elején így nem szükséges az alsó burkoló maximumának számolását is elvégezni: az első fázisban csak a felső burkoló minimumát, a második fázisban mindkét burkoló szélsőértékét kiszámítjuk.

Tisztázásra vár még a kezdőértékek kérdése. Kezdőértéknek elvileg bármely pozitív súlyvektor megfelel. Ha egyszerűen akarunk eljárni, használhatjuk a $w_i^{(0)} = 1, i = 1, \ldots, n$ csupa 1-es kezdőértékeket. Páros összehasonlítás mátrixok esetén azonban a keresett domináns sajátvektor általában közel van a logaritmikus legkisebb négyzetek módszere által adott soronként vett mértani középhez [11]. Így a kezdőértékeket ebben az esetben érdemes a következő módon választani:

$$w_i^{(0)} = \prod_{j=1}^{n} \sqrt[n]{a_{ij}}. \tag{15}$$

Ezt a kezdőértéket általános pozitív mátrixok esetén is használhatjuk a csupa egyesekből álló kezdőérték helyett.

Szükségünk van még a w_i értékek normalizálására is, hogy ne fordulhassanak elő túl nagy vagy túl kicsi számok. A normalizálást több módon is meg lehet tenni, például a w_i értékek összegét 1-re beállítani, vagy akár a $w_1 = 1$ módon is. A konkrét implementációban ez utóbbi normalizálás szerepel, melynek előnye, hogy eggyel csökkenti a szabad változók számát. A normalizálást minden iteráció végén elvégezzük.

Összefoglalva, a teljes algoritmus a következő lépésekből áll, ahol a $\cdot^{(p)}$ felső index egy függvényt, változót vagy értéket jelöl a p-edik iterációban.

1. Beállítjuk a $w_i^{(0)}, i = 1, \ldots, n$ kezdőértékeket (15) alapján. Legyen az iteráció indexe $p = 1$, és a fázis legyen az első fázis.

2. Legyen az aktuális változó indexe $k = 1$.

3. Az aktuális változó $w_k^{(p)}$, míg a többi változó értéke rögzítve van a $w_i^{(p)}, i = 1, \ldots, k-1$ és $w_i^{(p-1)}, i = k+1, \ldots, n$ értékeken.

4. Számoljuk ki $w_k^{(p)}$-t úgy, hogy az összes nem érdektelen $f_i^{(p)}$ közül azt az értéket választjuk, amire $f_k^{(p)}(w_k^{(p)}) = f_i^{(p)}(w_k^{(p)})$ a legkisebb.

5. $k = k + 1$. Ha $k < n$, térjünk vissza a 3. lépéshez.

6. Ha az első fázisban vagyunk, ellenőrizzük $f_n^{(p-1)}(w_n^{(p-1)}) - f_n^{(p)}(w_n^{(p)}) < T$ teljesülését. Ha a második fázisban vagyunk, ellenőrizzük $f_n^{(p)}(w_n^{(p)}) - f_n^{(p)}(w_n'^{(p)}) < T$ teljesülését. Itt T az előre meghatározott küszöbérték.

7. Normalizáljunk.

8. Ha a 6. lépésben az ellenőrzés eredménye igaz, és az első fázisban vagyunk, akkor lépjünk a második fázisba és térjünk vissza a 2. lépéshez.

Ha az ellenőrzés eredménye igaz, és a második fázisban vagyunk, akkor menjünk a 9. lépésre. Ha az ellenőrzés eredménye hamis, akkor legyen $p = p + 1$ és térjünk vissza a 2. lépéshez.

9. Eredményként adjuk ki a $w_i, i = 1, \ldots, n$ értékeket a domináns sajátvektor elemeiként és az $f_k(w_k)$ értéket a domináns sajátértékként.

A fent leírt algoritmus egy új eljárás a domináns sajátvektor és sajátérték számításra, mely nagyméretű mátrixokra van szabva és egyszerűségét a ciklikus koordináták módszere, valamint az aritmetikailag egyszerű számítások adják.

3 A konvergencia biztosítása

Ha a célfüggvény folytonosan differenciálható, akkor a ciklikus koordináták módszere biztosan konvergál [12, 252–253. oldal], azonban ebben az esetben ez nem teljesül. Bár a numerikus tesztek során mindannyiszor a valódi domináns jobb oldali sajátértékhez és sajátvektorhoz konvergált az eljárás, egy segédfeladat megoldásával elkerülhető az, hogy esetlegesen nem a megfelelő pontba konvergál az algoritmus. Fontos, hogy ezt a segédfeladatot általában nem kell megoldanunk, kizárólag akkor, ha úgy tűnik, hogy nem a megfelelő helyre konvergál az eljárás.

A segédfeladat egy lineáris programozási feladat formáját ölti. A célunk az lesz, hogy ellenőrizzük, hogy a domináns sajátérték, $\lambda_{\max}$ nem halad meg egy fix v értéket. A Collatz–Wielandt-formula (1. Tétel) értelmében a $\lambda_{\max}$ egyenlő a (2) kifejezéssel, azaz a cél a következő ellenőrzése:

$$\min_{\mathbf{w}>0} \max_{i=1,\ldots,n} \frac{(\mathbf{Aw})_i}{w_i} \leq v. \tag{16}$$

A (16) feltétel pontosan akkor teljesül, ha

$$\exists \mathbf{w} > 0 \quad \forall i\text{-re, hogy} \quad \frac{(\mathbf{Aw})_i}{w_i} \leq v. \tag{17}$$

Ez utóbbi átalakítható úgy, hogy a w_i értékekkel átszorzunk, így minden i-re egy lineáris egyenlőtlenséget kapunk: $(\mathbf{Aw})_i \leq v w_i$. Arra is szükség van azonban, hogy az összes változó, azaz a $\mathbf{w} = (w_1, \ldots, w_n)^\top$ határozottan pozitív legyen. Ez úgy érhető el, hogy egy új t változót bevezetve a következő lineáris programozási feladatot oldjuk meg:

$$\begin{aligned} &\max t && && \\ &(\mathbf{Aw})_i && \leq && v w_i \quad i = 1, \ldots, n \\ &w_i && \geq && t \quad\;\; i = 1, \ldots, n \end{aligned} \tag{18}$$

Amennyiben a (18) LP feladatnak plusz végtelen az optimumértéke, akkor az éppen azt jelenti, hogy létezik olyan $\mathbf{w}$, amire az összes $\frac{(\mathbf{Aw})_i}{w_i} \leq v$ (az

első feltételcsoport miatt), valamint az összes w_i választható pozitívnak (a második feltételcsoport miatt). Ez utóbbi azért igaz, mert ha az összes $w_i > 0$, akkor a feltételek bármekkora pozitív konstanssal szorozva is teljesülnek, tehát tetszőlegesen nagy t-nél nagyobbak lehetnek. Az első feltételcsoporton pedig a w_i-k konstanssal való szorzása nem változtat. Az, hogy a (18) LP feladatnak plusz végtelen az optimumértéke, ekvivalens azzal, hogy a következő LP-nek az optimumértéke pozitív:

$$\begin{aligned} &\max t && && (19)\\ &(\mathbf{Aw})_i && \leq \quad vw_i \quad && i = 1, \ldots, n\\ &\quad w_i && \geq \quad t \quad && i = 1, \ldots, n\\ &\sum_{i=1}^{n} w_i && = \quad 1 \end{aligned}$$

Ha (16) nem teljesül, akkor pedig a (18) esetén az optimumérték nem pozitív (az, hogy az összes változó 0, mindig lehetséges megoldás). Ha a (19) feladatról beszélünk, akkor a feladatnak vagy nincsen lehetséges megoldása, vagy az optimumérték nem pozitív.

Jelöljük λ-val az aktuális domináns sajátértékjelöltet, ami az előző fejezetben leírtak szerint az aktuális iterációban $f_k(w_k)$ módon adódik. A fent leírt (18) vagy (19) LP-kben $v = \lambda - T$ (ahol T az előre meghatározott toleranciaküszöb) választással tehát ellenőrizhető a

$$\lambda_{\max} \leq \lambda - T \tag{20}$$

feltétel. Ha a (20) feltétel hamis, akkor $T > \lambda - \lambda_{\max}$, azaz a toleranciaküszöbnél kisebb távolságra vagyunk a valódi domináns sajátértéktől, tehát az algoritmust célt ért. Ha azonban a (20) feltétel igaz, akkor $T \leq \lambda - \lambda_{\max}$, tehát a toleranciaküszöb kisebb, mint a jelenlegi becslésnek a domináns sajátértéktől vett távolsága, azaz a jelenlegi λ nem optimális érték.

Ha olyan ponthoz konvergálna tehát az algoritmus, ahol a (2) és az (1) kifejezések egyenlősége az adott T toleranciaszint mellett sem teljesül (azaz (2) − (1) nagyobb T-nél), akkor ennek a pontnak a környezetéből úgy léphetünk ki, hogy megoldjuk a (19) LP-t $v = \lambda - T$-vel, mely egy ilyen pontban minden esetben (a fentiek alapján) csupa pozitív megoldást fog adni, és a megoldásként kapott w_i, $i = 1, \ldots, n$ értékekből kiindulva folytatjuk az eljárást.

Az algoritmus tehát úgy módosul, hogy a 2. fejezet végén lévő pontokban a 6. lépést ki kell egészíteni a következővel. Ha a második fázisban a domináns sajátértékjelölt már nem változik egy $T' < T$ küszöbértéken belül, azaz $f_n^{(p-1)}(w_n^{(p-1)}) - f_n^{(p)}(w_n^{(p)}) < T'$, és a felső burkoló minimuma és az alsó burkoló maximuma továbbra is messze vannak egymástól, azaz $f_n^{(p)}(w_n^{(p)}) - f_n^{(p)}(w_n'^{(p)}) \geq T$, akkor a segéd LP-t lefuttatva az új w_i értékekből folytatjuk az algoritmust.

4 A segéd LP alkalmazása a Saaty-féle 10%-os szabályra

A 3. fejezetben bemutatott (18) LP egy további lehetséges alkalmazásának bemutatására kerül sor ebben a részben.

A többszempontú döntéselméletben igen fontos szerepe van a Saaty [18] által bevezetett páros összehasonlítás mátrixoknak. Ezek (például) szempontok összehasonlításakor a „Hányszor fontosabb az i szempont a j szempontnál?" módon feltett kérdésekre adott a_{ij} válaszokat egy mátrixba rendezve reprezentálják. Az ilyen mátrixok reciprok-szimmetrikusak, azaz $a_{ij} = 1/a_{ji}$ minden $i, j = 1, \ldots, n$- re, valamint pozitívak, tehát az ebben a tanulmányban bemutatott algoritmus alkalmazható rájuk. Az is ismert, hogy a domináns sajátértékre $\lambda_{\max} \geq n$ [18].

Egy páros összehasonlítás mátrixból egy úgynevezett súlyvektort kell számolni, amely (az előző példánál maradva) megadja a szempontok végső fontossági súlyértékét a döntésben. A súlyvektor számítására számos módszert alkalmaznak, azonban a legrégebbi és az egyik legnépszerűbb a szintén Saaty [18] által bevezetett sajátvektor módszer. Ez a módszer a páros összehasonlítás mátrix domináns sajátértékéhez tartozó jobb oldali sajátvektort tekinti súlyvektornak, tehát az algoritmus közvetlenül alkalmazható ennek megoldására. Saaty a páros összehasonlítás mátrixokat eredetileg az Analytic Hierarchy Process (AHP) [18] részeként vezette be, ahol a legnagyobb mátrixméretnek a 9×9-et javasolta, azóta azonban előfordulnak ennél sokkal nagyobb páros összehasonlítás mátrixok is. Különböző más alkalmazások [7, 17] mellett gyakran fordulnak elő sporttal kapcsolatosan, mind klasszikus kitöltött [17, 20], mind nem teljesen kitöltött mátrixok esetében [3, 8]. A nem teljesen kitöltött mátrixokra javasolt iteratív módszerek pozitív mátrixok egy sorozatán lépkednek, és minden mátrix esetén szükség van a legnagyobb sajátérték meghatározására [1, 4].

Egy páros összehasonlítás mátrixot konzisztensnek nevezünk, ha a kardinális tranzitivitási tulajdonság teljesül rá, azaz $a_{ik}a_{kj} = a_{ij}$ minden $i, j, k = 1, \ldots, n$ indexhármasra. Ha ez nem teljesül, akkor inkonzisztensnek nevezzük. Az inkonzisztencia nagyságát sok különböző módon javasolták mérni az irodalomban [5, 6], azonban a sajátvektor módszerhez szorosan kapcsolódó, emiatt szintén az egyik legrégebbi és legnépszerűbb a CR (Consistency Ratio) index [18]. Ennek meghatározásához először a CI (Consistency Index) értéket kell kiszámolni:

$$CI = \frac{\lambda_{\max} - n}{n - 1}.$$

A következő lépésben az RI (Random Index) értékkel kell ezt elosztani, amely véletlen-generált $n \times n$-es páros összehasonlítás mátrixokra számított átlagos CI érték. Ez n-től függ, viszont minden n-re egy egyszerű valós szám, amelyek táblázatba foglalhatóak [21, Table 1].

Saaty [18] javaslata alapján azok a páros összehasonlítás mátrixok tekinthetőek elfogadható inkonzisztenciájúnak, amelyekre $CR \leq 0{,}1$. Ez a szabály az úgynevezett 10%-os szabály, amelynek ellenőrzéséhez tehát a $\lambda_{\max}$

egy lineáris transzformációjának 0,1 alatt maradását kell ellenőrizni. Azaz a következő összefüggés ellenőrizendő:

$$\frac{CI}{RI} = \frac{\lambda_{\max} - n}{RI(n-1)} \leq 0,1.$$

Ez átrendezve a következő összefüggést adja:

$$\lambda_{\max} \leq 0{,}1RI(n-1) + n.$$

Tehát a (18) LP-t $v = 0{,}1RI(n-1) + n$ választással megoldva kapjuk az ellenőrzést arra vonatkozóan, hogy a mátrix eleget tesz-e a 10%-os szabálynak.

5 Teszteredmények

Ebben a fejezetben az algoritmus futási eredményeit hasonlítjuk össze a hatvány módszerével. Fontos megjegyezni, hogy a 3. fejezetben bemutatott segéd LP-t sohasem kellett ténylegesen lefuttatni, az algoritmus minden esetben konvergált. Így egyrészt a segéd LP semmilyen formában nem lassította az algoritmus futását, másrészt a tesztek nagy száma miatt igen valószínű, hogy az algoritmus minden esetben konvergál, így az LP csupán elméleti jelentőséggel bír.

Három mátrixtípuson teszteltünk. Az első csoport a „pozitív mátrixok", ezeknek minden eleme véletlen egész szám 1 és 9 között. A második csoport a „véletlen páros összehasonlítás mátrixok". Páros összehasonlításról mátrixokról (angolul Pairwise Comparison Matrix, röviden PCM) volt már szó a 4. fejezetben. Ezek reciprok-szimmetrikus pozitív mátrixok, azaz $a_{ij} = 1/a_{ji}$ minden $i, j = 1, \ldots, n$-re (következésképp $a_{ii} = 1$ minden $i = 1, \ldots, n$-re). A „véletlen páros összehasonlítás mátrixok" elemeit a Saaty [18] által javasolt módon az $1, \ldots, 9$ és reciprokaik közül választjuk véletlenszerűen (a konkrét implementációban először minden felső háromszögbeli elem esetén egy 1 és 9 közötti egészet generálunk, majd 1/2 eséllyel a reciprokát vesszük). A harmadik típus az „ordinálisan rendezett páros összehasonlítás mátrix", ami olyan, mint az előző, de a főátló feletti összes elem ≥ 1 (így a főátló alatti összes elem, a fentiek reciprokai lévén ≤ 1). Ez annak felel meg döntéselméleti szempontból, mintha az összehasonlítandó elemek ordinálisan csökkenő sorrendbe lennének rendezve. Ezek generálása úgy történt, hogy a felső háromszögben 1 és 9 között generáltunk egy véletlen egész számot.

A toleranciaküszöb 10^{-6}, az indulóértékek pedig a 2. fejezetben leírtak alapján, (15) szerint a sorok mértani közepei voltak.

Az *1. táblázat* a különböző mátrixok esetén az átlagos iterációszámokat tartalmazza. Az n a mátrix méretét jelöli. A típus a fenti három típus valamelyike, ahol a „PM" a „pozitív mátrix"-ot jelöli, a „PCM" a „véletlen páros összehasonlítás mátrix"-ot, az „OPCM" pedig az ordinálisan rendezett páros összehasonlítás mátrixot jelöli. Az „Első fázis", „Második fázis" és „Iterációszám" az algoritmus átlagos iterációszámát jelöli rendre az első és második fázisban, illetve a teljes iterációszámot (ami az előző kettő összege).

A „Hatvány m." a hatvány módszer átlagos iterációszámát jelöli. A tesztek minden méretre 1000 véletlen mátrixon lettek futtatva, kivéve $n = 500$ esetén, ahol 100-on (pozitív mátrixoknál még néhányat ki kellett venni a mintából, mivel egyik algoritmus sem futott le az előzetesen meghatározott maximálisan 200 iteráció alatt).

n	Típus	Első fázis	Második fázis	Iterációszám	Hatvány m.
10	PM	7,079	1,332	8,411	8,265
20	PM	7,736	1,814	9,550	7,212
50	PM	8,115	2,496	10,611	6,049
100	PM	8,106	2,972	11,078	5,518
200	PM	8,075	3,468	11,543	4,911
500	PM	7,943	4,011	11,954	4,523
10	PCM	17,852	5,559	23,411	20,225
20	PCM	18,950	6,948	25,898	14,498
50	PCM	19,287	8,977	28,264	10,409
100	PCM	19,150	10,286	29,436	8,690
200	PCM	18,988	11,365	30,353	7,828
500	PCM	18,440	12,800	31,240	6,090
10	OPCM	4,679	6,863	11,542	19,518
20	OPCM	5,983	6,084	12,067	19,123
50	OPCM	6,845	6,278	13,123	19,205
100	OPCM	7,369	6,401	13,770	19,582
200	OPCM	7,423	7,039	14,462	19,993
500	OPCM	7,650	7,340	14,990	20,000

1. táblázat. Átlagos iterációszámok összehasonlítása

Az eredményekből látszik, hogy általános pozitív, és általános véletlen páros összehasonlítás mátrixokon, ahol a hatvány módszer igen kevés iterációval végez, az itt bemutatott algoritmus határozottan lassabb. Azonban az ordinálisan rendezett páros összehasonlítás mátrixok esetén a javasolt algoritmus jelentősen kevesebb iterációt felhasználva éri el az optimumot.

6 Összefoglalás

A fentiek alapján tehát egy olyan új algoritmust adtunk pozitív mátrixok domináns sajátértékének és sajátvektorának számítására, amely kifejezetten nagyméretű mátrixokra van szabva. A vizsgált három esetből kettőben (általános pozitív véletlen mátrixok és véletlen páros összehasonlítás mátrixok) az algoritmus több iterációt igényelt, mint a hatvány módszer, azonban ordinálisan rendezett páros összehasonlítás mátrixok esetén kevesebbet.

Ez utóbbi eredmény alapján további vizsgálatok tárgyát azon esetek képezhetik, ahol szintén kevesebb iteráció igénybevételével fut le az algoritmus, mint a hatvány módszer. Továbbá a konvergencia formális bizonyítása hasznos továbblépésnek bizonyulhat, ami alapján ezek az esetek is könnyebben felderíthetőek.

Az algoritmus kiterjeszthető továbbá nemnegatív irreducibilis mátrixokra is. Egy másik kiterjesztési lehetőség az úgy nevezett nem teljesen kitöltött páros összehasonlítás mátrixok [9] esete, amikor magában a mátrixban is van-

nak hiányzó elemek, és a minimális domináns sajátértékű kitöltéshez tartozó sajátvektort keressük [4, 19].

Köszönetnyilvánítás

A kutatást az OTKA K 111797 pályázat és a Budapesti Corvinus Egyetem támogatta. Köszönet illeti továbbá az eredmény korábbi változataihoz tett értékes hozzászólásaiért, konferencia vagy szeminárium előadás és értekezés tervezet bírálatáért Bozóki Sándort, Csóka Pétert, Duleba Szabolcsot, Hegedűs Csabát és Temesi Józsefet, illetve a két anonim bírálót.

Irodalom

1. Ábele-Nagy K. (2015) Minimization of the Perron eigenvalue of incomplete pairwise comparison matrices by Newton iteration. *Acta Universitatis Sapientiae, Informatica,* 7(1):58–71.
2. Boyd S. és L. Vandenberghe (2004) *Convex Optimization.* Cambridge University Press.
3. Bozóki S., L. Csató, és J. Temesi (2016) An application of incomplete pairwise comparison matrices for ranking top tennis players. *European Journal of Operational Research,* 248(1):211–218.
4. Bozóki S., J. Fülöp, és L. Rónyai (2010) On optimal completion of incomplete pairwise comparison matrices. *Mathematical and Computer Modelling,* 52(1-2):318–333.
5. Brunelli M., L. Canal, és M. Fedrizzi (2013) Inconsistency indices for pairwise comparison matrices: a numerical study. *Annals of Operations Research,* 211(1):493–509.
6. Brunelli M. és M. Fedrizzi (2015) Axiomatic properties of inconsistency indices for pairwise comparisons. *Journal of the Operational Research Society,* 66(1):1–15.
7. Corcoran K., B. Dent, J. Smith, és P. Lara (1997) Location of optimal areas for the development of an alternative livestock species: the cashmere goat. Lsird Nafplio Conference Papers. The Macaulay Institute.
8. Csató L. (2012) Ranking by pairwise comparisons for swiss-system tournaments. *Central European Journal of Operations Research,* 21(4):783–803.
9. Harker P. (1987) Incomplete pairwise comparisons in the Analytic Hierarchy Process. *Mathematical Modelling,* 9(11):837–848.
10. Hämmerlin G. és K.-H. Hoffman (1991) *Numerical Mathematics.* Springer New York.
11. Kwiesielewicz M. (1996) The logarithmic least squares and the generalized pseudoinverse in estimating ratios. *European Journal of Operational Research,* 93(3):611–619.
12. Luenberger D. G. és Y. Ye (2008) *Linear and Nonlinear Programming,* volume 116 of International Series in Operations Research & Management Science. Springer, 3rd edition.
13. MacCluer C. R. (2000) The many proofs and applications of Perron's theorem. *SIAM Review,* 42(3):487–498.

14. Meyer C. (2000) *Matrix Analysis and Applied Linear Algebra.* SIAM.

15. Nesterov Y. (2012) Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization,* 22(2):341–362.

16. Nesterov Y. (2018) *Lectures on Convex Optimization.* Springer International Publishing.

17. Poesz A. (2018) Inkonzisztencia a döntéshozatalban. Ph.D. értekezés, Budapesti Corvinus Egyetem, Általános és Kvantitatív Közgazdaságtan Doktori Iskola.

18. Saaty T. L. (1977) A scaling method for priorities in hierarchical structures. *Journal of Mathematical Psychology,* 15(3):234–281.

19. Shiraishi S., T. Obata, és M. Daigo (1998) Properties of a positive reciprocal matrix and their application to AHP. *Journal of the Operations Research Society of Japan,* 41(3):404–414.

20. Sinuany-Stern Z. (1988) Ranking of sports teams via the AHP. *Journal of the Operational Research Society,* 39(7):661–667.

21. Wedley W. C. (1993) Consistency prediction for incomplete AHP matrices. *Mathematical and Computer Modelling,* 17(4-5):151–161.

COMPUTING THE DOMINANT EIGENVECTOR OF POSITIVE MATRICES BY THE METHOD OF CYCLIC COORDINATES

Computing the dominant (or principle, or Perron-) eigenvalue and eigenvector of positive matrices is an important task in many applications. For small matrices they can be determined fast, however it could be slow in case of large matrices. A new algorithm is presented, which is based on the method of cyclic coordinates, and is intended for application on large matrices. The standard method for computing the dominant eigenvector of diagonalizable matrices is the power method [10, pages 105–110.]. In the case of nonnegative matrices, irreducibility, which is always true for strictly positive matrices, is a sufficient condition for the applicability of the power method. Let $\lambda_{\max} = \lambda_{\max}(\mathbf{A})$ be the dominant eigenvalue, and $\mathbf{w}^*$ the corresponding dominant right eigenvector, thus $\mathbf{A}\mathbf{w}^* = \lambda_{\max}\mathbf{w}^*$. The Collatz–Wielandt theorem [14, pages 666 and 673] states, that if $\mathbf{A} \geq 0$ is an irreducible $n \times n$ matrix, then $\lambda_{\max} > 0$ and there is a unique corresponding eigenvector (up to a positive scalar multiplier). Furthermore,

$$\lambda_{\max} = \max_{w>0} \min_{i=1,\ldots,n} \frac{(\mathbf{A}\mathbf{w})_i}{w_i} \tag{1}$$

$$\lambda_{\max} = \min_{w>0} \max_{i=1,\ldots,n} \frac{(\mathbf{A}\mathbf{w})_i}{w_i} \tag{2}$$

From now $\mathbf{A} > 0$ is assumed. Note that every positive matrix is irreducible. The variables in the algorithm are the elements of the dominant eigenvector (denoted by $\mathbf{w}^*$): $w_1^*, \ldots, w_n^*$. Their current values are denoted by $\mathbf{w} = (w_1, \ldots, w_n)^\top$. The algorithm uses the method of cyclic coordinates [12], where only one variable is actually changing at a time, while the rest are fixed on their values calculated in the previous step. When the optimum for the current variable is found, the variable that changes shifts to the next one cyclically. Let the index of the variable actually

changing be k. Thus in every step the variable will be w_k. Considering (2), in every step the new value of w_k will be the $\hat{w}_k$ for which

$$\hat{w}_k = \arg\min_{w_k} \max_{i=1,\ldots,n} \frac{(\mathbf{Aw})_i}{w_i}. \tag{3}$$

As all w_j, $j \neq k$ are fixed, (3) depends only on w_k for all i. Thus the notation:

$$f_i(w_k) = \frac{(\mathbf{Aw})_i}{w_i} = \frac{a_{i1}w_1 + \ldots + a_{ik}w_k + \ldots + a_{in}w_n}{w_i}, \quad i = 1, \ldots, n. \tag{4}$$

Therefore in every step we are looking for the new value $\hat{w}_k > 0$ of w_k, for which $\hat{w}_k = \arg\min_{w_k} \max_{i=1,\ldots,n} f_i(w_k)$, in other words, the minimum point of the maximum of the f_i functions. The $f_i(\hat{w}_k)$ function value will be the approximation (upper bound) of $\lambda_{\max}$. Considering (4), the f_i for $i \neq k$ are affin (linear plus constant) functions, while $f_k(w_k)$ is a hyperbolic function. It also holds, that for the value $\hat{w}_k$, which satisfies (3), $f_i(\hat{w}_k) = f_k(\hat{w}_k)$ for some $i \neq k$. Thus it is sufficient to calculate the intersection points of f_k with all f_i, $i \neq k$. Because f_k is strictly monotonically decreasing, the $\hat{w}_k > 0$, which satisfies (3), will be the smallest w_k which satisfies $f_i(\hat{w}_k) = f_k(\hat{w}_k)$. The intersection points can be calculated using the quadratic formula.

The above is enough for implementation. However, there are some modifications to speed up the process. As we need only the intersection point of the maximum of the f_i functions with f_k, those f_i can be disregarded which have no common points with the maximum function. Because f_i, $i \neq k$ are linear, if one has a 'starting point' (its value at 0) below another one, which also has a higher slope, then the lower function can be ignored. The stopping criterion holds, when the minimum point of the maximum function (corresponding to (2)) is closer than a T tolerance threshold to the maximum point of the minimum function (corresponding to (1). The latter also has to be calculated though. Thus a further modification for speed up is to divide the algorithm into two phases. In the first phase, only the minimum point of the maximum function is calculated in each step. If this doesn't change much, we start the second phase and start calculating the maximum point of the minimum function only in this phase. The starting values could be any positive values (e.g. all ones), however the geometric means of each row are proposed for this, because in the case of pairwise comparison matrices [18], these values are typically close to the values of the dominant right eigenvector [11]. This is also a more sophisticated starting value in the general case as opposed to all ones. There is a normalization needed in each step as well. One possibility is to set the sum of the elements of the eigenvector to 1. Another one is to set the first element as 1. In the implementation the second one was chosen.

As the objective function is not continuously differentiable, the convergence of the cyclic coordinates method is not trivial. Thus, an LP is suggested to get around those instances where the algorithm would possibly not converge to the correct eigenvalue. The LP can be written as

$$\begin{aligned} &\max t \\ &(\mathbf{Aw})_i \leq v w_i \quad i = 1, \ldots, n \\ &\quad w_i \geq t \qquad i = 1, \ldots, n \\ &\sum_{i=1}^{n} w_i = 1 \end{aligned} \tag{19}$$

with $v = \lambda - T$, where λ is the current approximation of the dominant eigenvalue. If the algorithm converges somewhere wrong (which can be checked similarly to the stopping criterion), it solves the LP and continues from the solutions w_i. The LP can also be used to check if a pairwise comparison matrix is below the 10% CR inconsistency threshold [18].

The LP was never needed to be run in any of the tests, the algorithm always converged, which suggests it actually converges in every case. There were three classes of matrices inspected in the tests. First, positive matrices (PM), which had their values randomly generated as an integer between 1 and 9 for every entry. Second, pairwise comparison matrices (PCM, which are reciprocally symmetric matrices), which had their entries in the upper triangle randomly generated as an integer between 1 and 9, and taking its reciprocal with 50% chance. The diagonal is all ones, and the lower triangle takes the reciprocal of the opposite entry in the upper triangle. Third, 'ordinally consistent' pairwise comparison matrices (OPCM), which are the same as PCMs, but there is no chance to take the reciprocal in the upper triangle, thus, their entries in the upper triangle are all above 1, and the entries in the lower triangle are all below 1. The test results show, that in the case of positive and PCM matrices the power method uses much fewer iterations. However, in the case of the ordinally consistent pairwise comparison matrices, the power method uses significantly more iterations than the proposed algorithm.