Nelder-Mead algoritmus és variánsainak alkalmazása, tesztelése

Diplomamunka

Írta: Babarczy Mónika
Alkalmazott matematikus szak

Témavezető:

Illés Tibor
egyetemi docens
Eötvös Loránd Tudományegyetem
Operációkutatási Tanszék

Eötvös Loránd Tudományegyetem
Természettudományi Kar
2013

Absztrakt

A Nelder-Mead módszert először 1965-ben publikálták. Fél évszázad után is még az egyik legnépszerűbb direkt kereső eljárásként tartják számon az optimalizálás gyakorlati problémamegoldásában, bár az elméleti megközelítésben a módszer hagy maga után néhány nyitott kérdést. A legfontosabb probléma az algoritmussal a konvergenciája. Megalkotása óta többször módosították jobb konvergencia reményében. Bemutatom az eredeti 1965-ös algoritmust, a standarddé vált Lagarias és társai által publikált 1998-as verzióját, név variánsait, ezen eljárások konvergenciáját. Ismertetem ezeken felül két hatékony módosítását a módszernek, majd ezeket ötvözöm egy általam kidolgozott szintén eredményesen használható Nelder-Mead algoritmusban. Numerikus eredményekkel támasztom végül alá a módosítások hatékonyságát.

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Illés Tibornak a számtalan lehetőséget rejtő témáért és a hasznos tanácsokért, észrevételekért.

Köszönettel tartozom Egri Attila doktorandusznak a folyamatos segítségért, a nélkülözhetetlen tanácsokért, remek ötletekért és végtelen türelméért.

Tartalomjegyzék

Ábrák jegyzéke

1. fejezet

Bevezetés

A nemlineáris optimalizálás egy a korunkban egyre nagyobb teret és elismertséget szerző matematikai szakterület. Keletkezése, fejlődése szemléletes példája annak, hogy neves matematikusok több évszázadot átfogó és egymásra épülő sok-sok éves elméleti matematikai kutatásai, felfedezései, majd azok továbbfejlesztései találkozva egy új kor és műszaki fejlődés modern eszközeivel egyszerre az elmélet gyakorlati alkalmazhatóságának széles lehetőségét kínálják.

Az hogy a nemlineáris programozás manapság számtalan iparágban, a szállításban és a logisztikában alkalmazott tudománnyá vált, az több neves matematikus - közte egy magyar - matematikus munkásságának továbbá az elektronikus számítógép múlt századi megalkotásának és robbanásszerű - a világot és így a matematika világát is - alapjaiban megváltoztató fejlődésének és elterjedésének köszönhető.

Jelen szakdolgozatomban is ezen komoly elméleti múlttal bíró matematikai szakterületnek, a nemlineáris optimalizálásnak a mai műszaki és számítógépes fejlődés által lehetővé tett egyik kérdésével, a Nelder-Mead algoritmussal foglalkozom.

A nemlineáris optimalizálás elméleti alapjait Joseph-Louis Lagrange gróf, (Torino,1736. január 25. - Párizs, 1813. április 10.) olasz születésű francia matematikus rakta le Analitikus mechanika című könyvében 1788-ban.

Az elmélet továbbfejlesztéséhez nagyban hozzájárult Farkas Gyula (Pusztasárosd, 1847. március 28. – Pestszentlőrinc, 1930. december 27.) neves magyar matematikus és fizikus, aki kidolgozta és 1902-ben publikálta a Farkas-lemmát (”Theorie der einfachen Ungleichungen. *Crelle Journal,* 1902.”). Ezen módszert a nemlineáris optimalizálásban az optimum szükséges feltételeinek a meghatározására használják.

Harold W. Kuhn (Santa Monica, USA 1925. július 29. –) és Albert W. Tucker (Oshawa, Kanada 1905. november 28. - Hightstown, USA 1995. január 25.) matematikusok a Farkas féle elméletből kiindulva készítették el az optimalitás szükséges feltételeit bemutató értekezésüket [5], melyet 1951-ben a 2. Berkeley Matematika, Statisztika és Valószínűségszámítás Szimpóziumon ismertettek, elsőként használva a nemlineáris programozás elnevezést. Munkájuk során a lineáris programozás dualitás tételét általánosították. A számítógépek ez idő szerinti egyre növekvő megjelenésével a nemlineáris optimalizálás is

jelentős fejlődésnek indult.

A nemlineáris optimalizálás elméleti kutatásaira és addigi eredményeire épülve fejlődött ki a nemlineáris programozás egy részterülete, a direkt kereső eljárás ("direct search method"), melynek alapjait Robert Hooke és T.A. Jeeves rakták le a "Direct Search Solution of Numerical and Statistical Problems" című tanulmányukban [3] az amerikai Számítógép Társaság 1961. évi kiadványában.

Egy évvel később a direkt kereső eljárást W. Spendley, G. R. Hext és F.R. Himswort szimplex alapon dolgozták ki [19] tükrözéses transzformációval a legrosszabb csúcsból, illetve zsugorításos transzformációval a legjobb csúcs irányába. Így az állandó szögek közti éleket használva a szimplexek méretükben igen, alakjukban azonban nem változnak.

John Ashworth Nelder (1924. október 8. – 2010. augusztus 7.) és Roger Mead (1938. május 24. –) ezen módszert továbbfejlesztve és a tágításos és összehúzásos transzformációkkal kiegészítve dolgozták ki és 1965-ben publikálták szimplex módszerüket ["A simplex method for function minimization", Computer Journal 7., USA pp. 308-313] [11], mely szerint a mozgó szimplex méretében és alakjában egyaránt változik. A módszer mintegy húsz évig kevéssé volt használatos, azonban mivel az algoritmus – az időközben megjelenő és gyorsan elterjedő – kis méretű számítógépeken is használható, ezért downhill szimplex módszer, illetve amőba módszer néven az 1980-as években népszerűvé vált a felhasználók között, és bekerült a Numerical Recipes (1992) [21] kézikönyvbe és a Matlab (2008) szoftvercsomagba is.

A létrehozása óta eltelt majd fél évszázad ellenére a Nelder-Mead algoritmust még mindig az egyik legnépszerűbb direkt kereső eljárásként tartják számon a gyakorlati problémamegoldásban. Bár az elméleti megközelítésben a módszer hagy maga után néhány nyitott kérdést (például konvergenciájával kapcsolatban), viszont könnyű érthetőségével, egyszerű alkalmazhatóságával és alacsony tárigényével ideális eszköz a gyakorlati szakember kezében, akinek minél gyorsabban, minél kisebb költséggel, minél jobb optimum kell hogy a rendelkezésére álljon. Használata különösen elterjedt a műszaki tudományok és a gyógyászat területén, de a statisztikában is, a bizonytalan függvényértékeken alapuló paraméterbecslés és a nem folytonos függvények problematikájában.

A matematikai optimalizálás ügyének előmozdítása és a legfejlettebb kutatási eredmények széles körű elterjesztésének céljával működik az 1975-ben philadelphiai székhellyel létrehozott nemzetközi Matematikai Optimalizációs Társaság. A tudományos társaság Matematikai programozás néven szakfolyóiratot is ad ki, és három évente az optimalizálás időszerű kérdéseivel foglalkozó konferenciát rendez.

2. fejezet

Nelder-Mead algoritmus háttere

A Nelder-Mead algoritmus kifejlesztésének célja az volt, hogy megadják vele egy adott nemlineáris, többdimenziós $f : \mathbf{R}^n \to \mathbf{R}$ függvény minimumát egy klasszikus, nem korlátozott optimalizálási feladatban. [S. Singer - J. Nelder] [16]

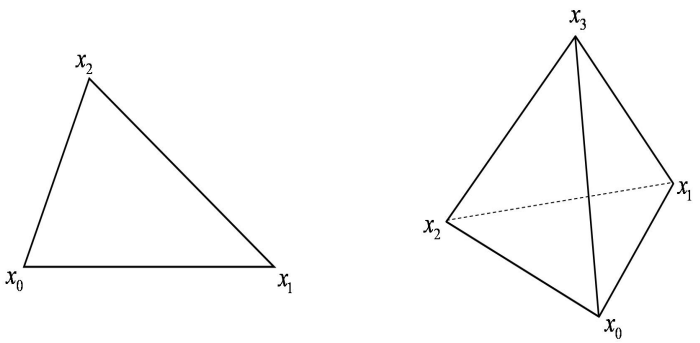
A módszer:

1. csak függvényértékeket használ, néhány pontját $\mathbf{R}^n$–nek, és
2. nem próbál kialakítani egy approximációs gradienst e pontok egyikére sem.

Az algoritmus a direkt kereső eljárások családjába tartozik [Wright, 1996] [22], [Powell, 1998] [12].

A Nelder-Mead eljárás szimplex alapú. A S szimplex az $\mathbf{R}^n$–ben úgy van megadva, mint $n+1$ db csúcs konvex burka: $x_0, \ldots, x_n$. Például a szimplex az $\mathbf{R}^2$–ben egy háromszög, $\mathbf{R}^3$–ben egy tetraéder.

A szimplex alapú direkt kereső eljárások azzal kezdődnek, hogy kiválasztunk $n+1$ pontot a halmazból ($x_0, \ldots, x_n \in \mathbf{R}^n$), amikre úgy tekintünk, mint egy S szimplex csúcsaira, és hasonlóan a függvényértékek halmazára $f_j = f(x_j)$, $j = 0, \ldots, n$. A kezdő S szimplex nem degenerált, azaz az $x_0, \ldots, x_n$ pontoknak nem szabad ugyanabban a hipertérben feküdniük.



2.1. ábra. Két dimenziós szimplex (balra), három dimenziós szimplex (jobbra)

A módszer, amikor végrehajtja a szimplex átalakítások folyamatát, akkor az a cél vezérli, hogy csökkentse a függvényértékeket a csúcsokban. Minden lépésben az átalakítás meghatároz egy vagy több teszt pontot kiszámításra a függvényértékükkel együtt, és ezen függvényértékek összehasonlítását ezekben a meghatározott csúcsokban.

Ez a folyamat behatárolja, mikorra válik elég kicsivé a S szimplex valamilyen értelemben, vagy mikorra lesznek elég közel a f_j függvényértékek valamilyen értelemben (felhasználva, hogy f folytonos).

A Nelder-Mead algoritmus jellemzően megköveteli, hogy csak egy vagy maximum két függvény kiértékelés történjen minden egyes lépésben, amíg sok más direkt kereső módszer n vagy még több függvény kiértékelést használ.

3. fejezet

Az eredeti algoritmus 1965-ből

Ebben a fejezetben a jelen szakdolgozatom kiindulási alapjául szolgáló 1965-ben publikált eredeti Nelder-Mead algoritmusról szóló (”A simplex method for function minimization”) [11] cikket elemzem a mai számítógépes programozást is segítségül hívott szemléltetési verzió bemutatásával. Ebben a részben megtartom az eredeti cikk jelöléseit, elkerülendő, hogy azok összekeveredjenek a következő fejezetekben ismertetett hasonló módszerek jelöléseivel.

A cél egy korlátozás nélküli n változós függvény minimalizálása: legyen $P_0, P_1, ..., P_n$ $n+1$ darab pont az n dimenziós térben, melyeket úgy választunk, hogy szimplexet definiáljanak. Az y_i változóhoz tartozó függvényérték legyen P_i, és definiáljuk

1. h -t, mint indexet: $y_h = \max_i(y_i)$ [h , mint high]
2. l -t, mint indexet: $y_l = \min_i(y_i)$ [l , mint low]

Továbbá, definiáljuk $\overline{P}$-t, mint a pontok centroidát ($i \neq h$), és $[P_iP_j]$ jelölje a P_i és P_j pontok közötti távolságot. Mindegyik szakasz a folyamatban behelyettesít egy új csúcsot P_h; három műveletet használ az algoritmus a futása alatt:

1. tükrözés (reflection),
2. összehúzás (contraction),
3. tágítás/nyújtás (expansion).

Ezeket definiáljuk a következőként:

1. **Tükrözési együttható** (reflection coefficient): a P_h tükrözése meghatározza a P^*-t:
$$P^* = (1+\alpha)\overline{P} - \alpha P_h,$$
ahol α egy pozitív konstans, a tükrözési együttható. Így P^* egy vonalon helyezkedik el P_h-val és $\overline{P}$-sal, $\overline{P}$ túlsó oldalán P_h-ból lehet kapni $[P^*\overline{P}] = \alpha[P_h\overline{P}]$ -sal. Ha y^* az y_h és az y_l között fekszik, akkor P_h -t kicseréljük P^*-gal és kezdjük újra az új szimplexszel.

2. **Tágítási/nyújtási együttható** (expansion coefficient): Ha $y^* < y_l$, azaz, ha a tükrözés létrehoz egy új minimumot, akkor kiterjesztjük P^*-t P^{**}-ra:

$$P^{**} = \gamma P^* + (1 - \gamma)\overline{P}.$$

A tágítási/nyújtási együttható γ - ami nagyobb, mint az egység – a $[P^{**}\overline{P}]$ és a $[P^*\overline{P}]$ távolságok hányadosa. Ha $y^{**} < y_l$, akkor kicseréljük a P_h-t a P^{**}-ra és újra indítjuk a folyamatot; de ha $y_l < y^{**}$, akkor nem sikerült a tágítás/nyújtás, és kicseréljük a P_h-t a P^*-ra, mielőtt újra indítjuk a folyamatot.

3. **Összehúzási együttható** (contraction coefficient): Ha tükrözéskor P-re P^*-ot találjuk, akkor $y_i < y^*$ minden $i \neq h$ -ra, úgymint P-t P^* -ra cserélése közben y^*-ot hagyjuk a maximumnak, akkor definiálunk egy új P_h -t vagy a régi P_h-ra vagy a P^*-ra, amelyik alacsonyabb y értékénél, és alakítjuk ki P^{**}-t:

$$P^{**} = \beta P_h + (1 - \beta)\overline{P}.$$

Az összehúzási együttható β 0 és 1 között helyezkedik el, és a $[P^{**}\overline{P}]$ és a $[P\overline{P}]$ távolságok hányadosa. Mikor elfogadjuk P^{**}-t P_h-ra, újra indítjuk a folyamatot, kivéve ha $y^{**} > \min(y_h, y^*)$, úgymint az összehúzási pont rosszabb, mint a P_h jobbja és a P^*. Minden egyes sikertelen összehúzásra megismételjük minden P_i-t $\frac{(P_i+P_l)}{2}$-vel és újraindítjuk a folyamatot.

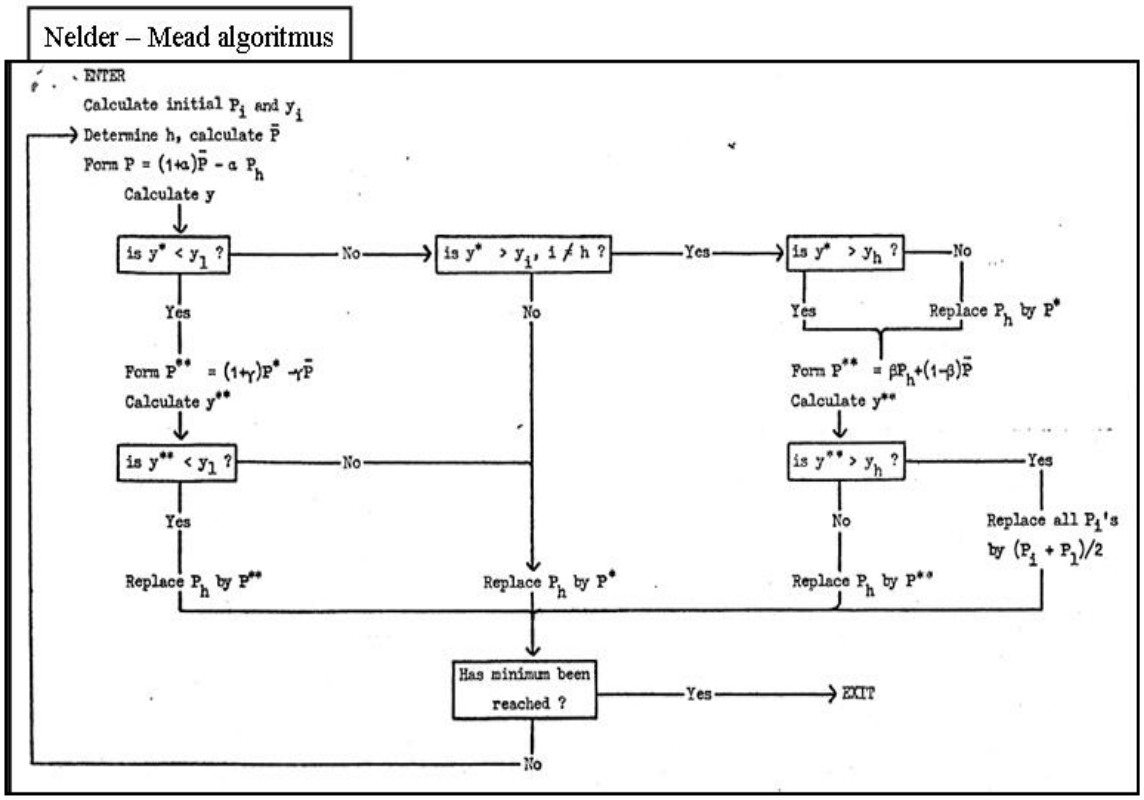
A sikertelen tágítás/nyújtás származhat abból is, hogy egy szerencsés behatolás történt egy völgybe (P^*), de ferdén a völgyben, így hogy a P^{**} a szemben fekvő lejtőn fakad. A sikertelen összehúzás sokkal ritkább, de megtörténhet, amikor a völgy görbül és a szimplex egy pontja távolabb esik a völgy aljától, mint a többi; összehúzás okozhatja a tükrözési pont elmozdulását a völgy aljától ahelyett, hogy afelé mozdulna. A további összehúzások ekkor már hasznavehetetlenek. Az eljárás a szimplex összehúzását tanácsolja a legalsó pont felé, így hozva végül az összes pontot a völgybe. Az együtthatók α, β, γ adják azt a tényezőt, mely változtatja a szimplex alakját a tükrözési, a tágítás/nyújtási, az összehúzási műveleteknek megfelelően. Az eredeti cikk ezt a három választást találta optimálisnak:

$$\alpha = 1, \qquad \beta = \frac{1}{2}, \qquad \gamma = 2.$$

A végső pont attól a kritériumtól függ, amit megállási feltételként adnuk a folyamathoz. A kritérium némiképp módosítva Powell-től (1964) [12] lett adoptálva, úgyhogy a változás a szimplex felett inkább y értékeit érinti, semmint az x értékeit. Az alak kiválasztása az y-ok standard hibáink összehasonlításából adódik, ahol is ez a standard hiba

$$\sqrt{\frac{(\sum (y_i - \overline{y})^2)}{n}}$$

alakban adott előre beprogramozott hibahatárral, és amikor ez alá esik ez az érték, akkor megállunk. A kritérium sikere attól függ, hogy ne váljon a szimplex túl kicsivé a felszín görbületének függvényében, amíg a végső minimumot el nem éri.



3.1. ábra. Az 1965-ös cikkben közölt eredeti folyamatábra

A teljes módszer megadható egy folyamatábrán (3.1 ábra).

Manapság már nem igazán elterjedt ez a szemléltetési verzió, inkább programozási struktúrában adják meg az algoritmusokat. Még inkább elterjedt és használhatóságában is célra vezetőbb egy megírt számítógépes program stilizált vázának képében ismertetni egy-egy módszert. Mint, ahogy a következőekben látszik is, ilyen formában szinte már kész is a számítógépes program, mindössze az adott programozási nyelv sajátosságaihoz kell igazítani: (3.1 Algoritmus).

3.1 Algoritmus:

Nelder-Mead(1965)

Bemenő adatok:
Kezdő szimplex, leállási feltétel

Begin
Kezdő szimplex megadása: számítsuk ki a kezdő P_i-ket és y_i-ket;
While leállási feltétel nem teljesül **do**
Határozzuk meg h-t;
Számítsuk ki $\overline{P}$-t;
$P^* = (1+\alpha)\overline{P} - \alpha P_h$;
Számítsuk ki y^*-t;
If $y^* < y_l$ **then**
$P^{**} = (1+\gamma)P^* - \gamma\overline{P}$;
Számítsuk ki y^{**}-t;
If $y^{**} < y_l$ **then**
Cseréljük P_h -t P^{**}-ra;
else
Cseréljük P_h -t P^*-ra;
endif
else
If $y_i < y^*$ $(i \neq h)$ **then**
If $y_h < y^*$**then**
;
else
Cseréljük P_h -t P^*-ra;
endif
$P^{**} = \beta P_h + (1-\beta)\overline{P}$;
Számítsuk ki y^{**}-t;
If $y_h < y^{**}$ **then**
Cseréljük az összes P_i-t $\frac{(P_i+P_l)}{2}$-re;
else
Cseréljük P_h -t P^{**}-ra;
endif
else
Cseréljük P_h -t P^*-ra;
endif
endif
endwhile
End

4. fejezet

Módosított klasszikus Nelder-Mead algoritmus

A Nelder-Mead módszer megvalósításra sok különféle lehetőség adódik. [16] A kezdő algoritmus néhány kis számítási részletét nem számítva is, a fő különbség a változatos kivitelezésekben a kezdő szimplex megkonstruálásában, és a konvergenciában vagy az iterációs folyamat végére adott leállási feltételek meghatározásában rejlik. Az általános algoritmus így néz ki:

1. Konstruáljuk meg a kezdő S szimplexet.
2. Ismételjük az alábbi lépéseket, amíg a leállási feltételek nem teljesülnek:
   (a) Számoljuk ki a leállási feltételeket;
   (b) Ha a leállási feltételek nem teljesülnek, akkor átalakul az épp aktuális szimplex.
3. Ha a leállási feltételek teljesülnek, akkor az aktuális szimplex legjobb csúcsával tér vissza és a hozzákapcsolódó függvényértékkel.

4.1. Kezdő szimplex

A kezdő S szimplex szokásos megszerkesztése: generáljunk $n+1$ csúcsot $x_0, \ldots, x_n$ egy adott bemenő pont $x_{in} \in \mathbf{R}^n$ köré. A gyakorlatban a legelterjedtebb választás $x_0 = x_{in}$, ami engedi az algoritmus valódi újraindulását. A megmaradó n csúcsból az S alábbi eseteit szokták kialakítani a generálás alatt:

1. S derékszögű az x_0–ra, kiindulópontul szolgál a koordinátatengelyeken, vagy $x_j := x_0 + h_j e_j$, $j = 1, \ldots, n$, ahol h_j egy lépés méret az e_j egység vektor irányába $\mathbf{R}^n$–en.
2. S szabályos szimplex, ahol minden élnek ugyanaz a megadott hossza.

4.2. Szimplex transzformációs algoritmus

A Nelder-Mead algoritmus egy iterációs lépése a következő három lépésből áll:

1. **Sorba rendezés**: Határozzunk meg három indexet: h, a legrosszabb csúcsra; s, a második legrosszabb csúcsra; l, pedig a legjobb csúcsra vonatkozik; mindig az aktuális szimplexre, S –re vonatkozóan. ($f_i = f(x_i)$ jelöléssel) $f_h = \max_j f_j$, $f_s = \max_{j \neq h} f_j$, $f_l = \min_{j \neq h} f_j$. Néhány megvalósításban az S csúcsai sorba vannak rendezve a függvényértékekre vonatkozóan: $f_0 \leq f_1 \leq \ldots \leq f_{n-1} \leq f_n$. Ekkor $l = 0$, $s = n-1$, és $h = n$.

2. **Centroid, avagy középpontosítás**: Kiszámítjuk a legjobb oldal centroidját c. Ez az egyetlen, ami szemben helyezkedik a legrosszabb csúccsal x_h :

$$c := \tfrac{1}{n} \textstyle\sum_{j \neq h} x_j$$

3. **Transzformáció**: Kiszámítjuk az új szimplexet az aktuálisból. Először is próbáljuk csak a legrosszabb csúcsot x_h visszahívni egy jobb ponttal tükrözés, tágítás vagy összehúzás segítségével, figyelemmel a legjobb oldalra. Az összes tesztpont egy vonalon fekszik definiálva x_h és c által, és legfeljebb kettőt közülük kiszámítunk egy iterációs lépésen belül. Ha ez sikert ér el, akkor az elfogadott pont válik a szimplex új csúcsává. Ha sikertelen, akkor zsugorítjuk a szimplexet a legjobb csúcs x_j felé. Ebben az esetben n új csúcs lett kiszámítva.

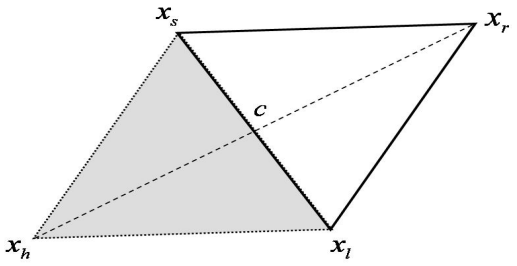
A szimplex transzformációkat a Nelder-Mead módszerben négy paraméter kontrollálja:

1. α : tükrözés (reflection);
2. β : összehúzás (contraction);
3. γ : tágítás/nyújtás (expansion);
4. δ : zsugorítás (shrinkage).

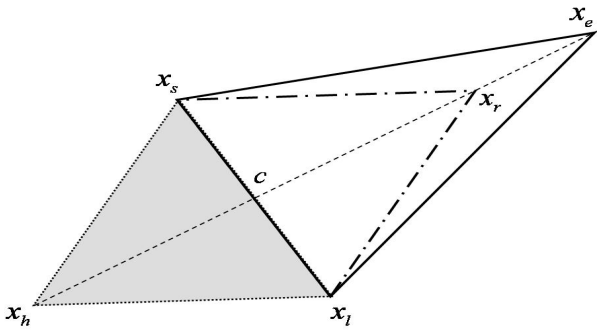
$$0 < \alpha, \qquad 0 < \beta < 1, \qquad 1 < \gamma, \quad \alpha < \gamma, \qquad 0 < \delta < 1.$$

A standard értékek, a legtöbb kivitelezéshez ezeket használják:

$$\alpha = 1, \qquad \beta = \frac{1}{2}, \qquad \gamma = 2, \qquad \delta = \frac{1}{2}.$$



4.1. ábra. Tükrözés



4.2. ábra. Tágítás/nyújtás

4.3. Wright – féle szimplex átalakítások (1996)

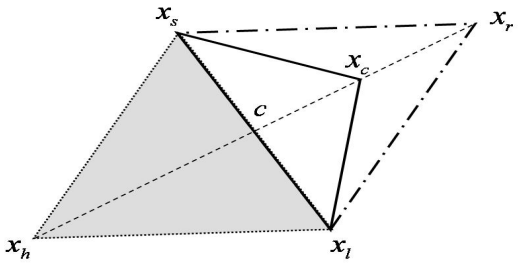
A következő algoritmus bemutatja a szimplex transzformálását három lépésben (Wright, 1996) [22], a különböző átalakulásokat az ábrákon lehet figyelemmel követni. Az új szimplexet mindig a vastagon húzott vonal jelöli, az eredetit a szürkével színezett háromszög.

1. **Tükrözés (reflect):** Számítsuk ki a tükrözési pontot: $x_r := c + \alpha(c - x_h)$ és $f_r := f(x_r)$. Ha $f_l \leq f_r < f_s$, elfogadjuk x_r-t és befejezzük az iterációt. (4.1 Ábra)

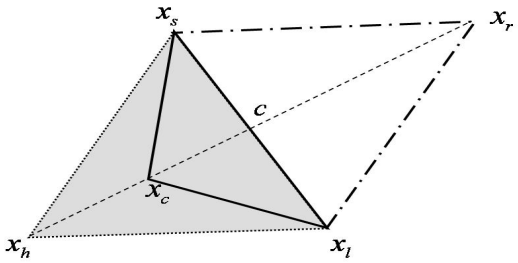
2. **Tágítás/nyújtás (expansion):** Ha $f_r < f_l$, számítsuk ki a nyújtási pontot $x_e := c + \gamma(x_r - c)$ és $f_e := f(x_e)$. Ha $f_e < f_r$, elfogadjuk x_e-t és befejezzük az iterációt. Másképpen ($f_e \geq f_r$), elfogadjuk x_r-t és befejezzük az iterációt. Ez "mohó minimalizálás" ami magába foglalja befejezéskor a két legjobb pontot x_r, x_e az új szimplexben, és a szimplexet csak akkor nyújtjuk, ha $f_e < f_r < f_l$. Ezt használják a legtöbb kivitelezésben és az elméletben szintén (Lagaris, 1998). [7] (4.2 Ábra)

3. **Összehúzás (contract):** Ha $f_r \geq f_s$, kiszámítjuk az összehúzási pontot x_c, amire a két legjobb pontot használjuk x_h-t és x_r-t.

   (a) **Kifelé:** Ha $f_s \leq f_r < f_h$, kiszámítjuk $x_c := c + \beta(x_r - c)$ és $f_c := f(x_c)$. Ha $f_c \leq f_r$, elfogadjuk x_c-t és befejezzük az iterációt. Egyébként, végrehajtunk egy zsugorítási transzformációt. (4.3 Ábra)



4.3. ábra. Összehúzás kifelé



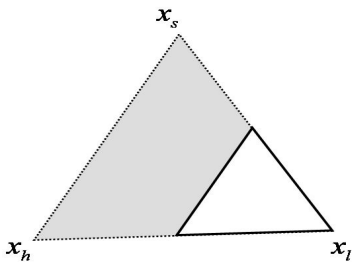
4.4. ábra. Összehúzás befelé

(b) **Befelé:** Ha $f_r \geq f_h$, kiszámítjuk $x_c := c + \beta(x_h - c)$ és $f_c := f(x_c)$. Ha $f_c < f_h$, elfogadjuk x_c-t és befejezzük az iterációt. Egyébként, végrehajtunk egy zsugorítási transzformációt. (4.4 Ábra)

4. **Zsugorítás (shrinkage):** Kiszámítjuk n új csúcsra $x_j := x_l + \delta(x_j - x_l)$ és $f_j := f(x_j)$, $j = 0, ..., n$ és $j \neq l$. (4.5 Ábra)

A zsugorító transzformáció megakadályozza, hogy az algoritmus végrehajtsa a következő esetet, amit az eredeti 1965-ös cikk [11] részletesen ismertet:

”A sikertelen összehúzás sokkal ritkább, de megtörténhet, amikor a völgy görbül és a szimplex egy pontja távolabb esik a völgy aljától, mint a többi; összehúzás okozhatja a tükrözési pont elmozdulását a völgy aljától ahelyett, hogy afelé mozdulna. A további összehúzások ekkor már hasznavehetetlenek. Az eljárás tanácsolja a szimplex összehúzását a legalsó pont felé, így hozva végül az összes pontot a völgybe.”



4.5. ábra. Zsugorítás

4.4. Lagarias – féle szimplex transzformációs algoritmus (1998)

Lagaris-féle leírás [7] és az előbbi Wright-féle [22] között nincs nagy különbség. Mindössze annyi, hogy a szimplex transzformációs algoritmus sorba rendezés lépésénél Lagaris-féle elképzelés szigorúan megköveteli a csúcsok rendezését.

Ilyen megvalósításban az S csúcsai sorba vannak rendezve a függvényértékekre vonatkozóan: $f_0 \leq f_1 \leq \ldots \leq f_{n-1} \leq f_n$. Ekkor az előbbiek így módosulnak: $l = 0$, $s = n - 1$, és $h = n$.

A Matlab "fminsearch" parancsa alatt futó Nelder-Mead algoritmus: az 1998-as Lagarisék által publikált a '65-ös eljárást módosító módszer [7].

4.5. A Módosított algoritmus vázlatos futása

Kezdésnek kiválasztunk véletlenszerűen $n + 1$ pontot, és kiértékeljük ezeket. E pontokat úgy választjuk ki, hogy egy n dimenziós szimplexet definiáljanak. Az algoritmus a következő módon fut le:

1. Rendezzük sorba a csúcsokat az $f(x)$ értékeik szerint.
2. Ha a legrosszabb és a legjobb csúcsok benne vannak egymásnak az adott toleranciájában, akkor STOP.
3. Az iteralizáció kezdéseként próbáljunk kiválasztani egy jobb pontot a legrosszabb csúcs helyett. Találjuk meg a másik n csúcsnak a centroidját (mindegyiknek kivéve a legrosszabbat) és tükrözzük a legrosszabb csúcson keresztül ezt a pontot. Az új pont eshet ugyanarra a távolságra a központtól, mint a régi (legrosszabb) pont.
4. Kiértékeljük ezt a pontot. Ha ez egy javulás (pontosabban: f_{new} jobb, mint a f_{worst}), akkor a régi legrosszabb csúcsot kicseréljük erre.

5. Ha ez az új csúcs jobb, mint az előző, kiterjesztjük az új pontot két tényező között húzódó vonal mentén a pont és a centroid között. Más szavakkal, mozgásban tartjuk a mozgatás irányában.

6. Ha az új csúcs rosszabb, akkor ez a második legrosszabb csúcs, ekkor látjuk, hogy nem csináltunk javulást, és így szükségünk van arra, hogy összehúzzuk a szimplexet egy dimenzió mentén. Csökkentve így a távolságot az új pont és a középpont között a felére.

7. Ha még nem javult az új pont (rosszabb, mint a második legrosszabb), akkor összehúzzuk a szimplexet az összes dimenzió mentén. A távolságot a központ és mindegyik pont között (kivéve a legjobbat) vágjuk a felére.

8. Térjünk vissza az 1. pontra.

4.5.1 Algoritmus:

Módosított klasszikus Nelder-Mead (1996) [22]

Bemenő adatok:
Kezdő szimplex, leállási feltétel
Begin
Kezdő szimplex megadása: számítsuk ki a kezdő x_i-ket és $f(x_i)$-ket;
While leállási feltétel nem teljesül **do**
Határozzuk meg h-t;
Számítsuk ki c-t;
$x_r = c + \alpha(c - x_h)$;
Számítsuk $f(x_r)$-t;
If $f(x_r) < f(x_l)$ **then**
$x_e := c + \gamma(x_r - c)$;
Számítsuk ki $f(x_e)$-t;
If $f(x_e) < f(x_r)$ **then**
Cseréljük x_h -t x_e-re;
else
Cseréljük x_h -t x_r-re;
endif
else
If $f(x_s) \leq f(x_r)$ **then**
If $f(x_r) < f(x_h)$ **then**
$x_c := c + \beta(x_r - c)$;
Számítsuk ki $f(x_c)$-t;
If $f(x_r) < f(x_c)$ **then**
$\forall$ x_j-re $(j \neq l)$: $x_j := x_l + \delta(x_j - x_l)$ és $f(x_j)$;
else
Cseréljük x_r -et x_c-re;
endif
else
$x_c := c + \beta(x_h - c)$;
Számítsuk ki $f(x_c)$-t;
If $f(x_h) < f(x_c)$ **then**
$\forall$ x_j-re $(j \neq l)$: $x_j := x_l + \delta(x_j - x_l)$ és $f(x_j)$;
else
Cseréljük x_h -et x_c-re;
endif
endif
else
Cseréljük x_h -et x_r-re;
endif
endif
endwhile
End

4.5.2 Algoritmus:

Módosított klasszikus Nelder-Mead (1998) [7]

Bemenő adatok:
Kezdő szimplex, leállási feltétel
Begin
Kezdő szimplex megadása: számítsuk ki a kezdő x_i-ket és $f(x_i)$-ket;
While leállási feltétel nem teljesül **do**
Rendezzük sorba az $f(x_i)$-ket;
Számítsuk ki c-t;
$x_r = c + \alpha(c - x_n)$;
Számítsuk $f(x_r)$-t;
If $f(x_r) < f(x_1)$ **then**
$x_e := c + \gamma(x_r - c)$;
Számítsuk ki $f(x_e)$-t;
If $f(x_e) < f(x_r)$ **then**
Cseréljük x_n -t x_e-re;
else
Cseréljük x_n -t x_r-re;
endif
else
If $f(x_{n-1}) \leq f(x_r)$ **then**
If $f(x_r) < f(x_n)$ **then**
$x_c := c + \beta(x_r - c)$;
Számítsuk ki $f(x_c)$-t;
If $f(x_r) < f(x_c)$ **then**
$\forall$ x_j-re $(j \neq 1)$: $x_j := x_1 + \delta(x_j - x_1)$ és $f(x_j)$;
else
Cseréljük x_r -et x_c-re;
endif
else
$x_c := c + \beta(x_n - c)$;
Számítsuk ki $f(x_c)$-t;
If $f(x_n) < f(x_c)$ **then**
$\forall$ x_j-re $(j \neq 1)$: $x_j := x_1 + \delta(x_j - x_1)$ és $f(x_j)$;
else
Cseréljük x_n -et x_c-re;
endif
endif
else
Cseréljük x_n -et x_r-re;
endif
endif
endwhile
End

5. fejezet

Az 1965-ös, 1996-os és 1998-as algoritmusok közötti különbség

Az eddigiekben ismertettem az 1965-ös Nelder és Mead által tervezett szimplex kereső eljárást [11], majd áttértem a 1996-os M. Wright [22] cikkében feltűnő ezen Nelder-Mead algoritmus módosított változatának bemutatására. Mindezek után pedig összefoglaltam, hogy az 1998-as publikációban [7] Lagaris és szerzőtársai ezen 1996-os módszerre még milyen megkötéseket és javításokat javasolnak.

A mostani fejezetben kitérek e három módszer fontosabb eltéréseinek bemutatására.

Nelder és Mead az 1965-ös [11] cikkükben igen nagy vonalakban ismertették híressé vált módszerüket. Ezzel lehetőséget adva különböző lehetséges magyarázatok megszületésére és az eltérő algoritmus változatok megalkotására. Ezek az eredeti módszer hiányosságainak félremagyarázhatóságából jöhettek létre. Ezen hiányosságok legfontosabbjai:

1. Az egyenlőtlenségek pontosságára vonatkozóan
2. "tie-breaking" szabály hiánya

Az 1965-ös [11] cikk nem tér ki a pontok sorba rendezésére azonos függvényértékeknél, ezért Lagarisék a 1998-as [7] publikációjukban a következő "tie-breaking" szabályt adoptálják: az új csúcsnak a legmagasabb lehetséges index-szel rendelkezőt fogadja el, következetesen végig az egész folyamat alatt ($f(x_0^{(k)}) \leq f(x_1^{(k)}) \leq f(x_2^{(k)}) \leq \ldots \leq f(x_n^{(k)})$ $\forall k$-ra , ami a k. iterációt jelenti).

M. Wright az 1996-os [22] cikkében már a zsugorítás lépését más alakban adja meg, hogy elkerülhetővé váljon az 1965-ös Nelder és Mead [11] cikkében ismertetett eset:

> "A sikertelen összehúzás sokkal ritkább, de megtörténhet, amikor a völgy görbül és a szimplex egy pontja távolabb esik a völgy aljától, mint a többi; összehúzás okozhatja a tükrözési pont elmozdulását a völgy aljától ahelyett, hogy afelé mozdulna. A további összehúzások ekkor már hasznavehetetlenek. Az eljárás tanácsolja a szimplex összehúzását a legalsó pont felé, így hozva végül az összes pontot a völgybe."

A tágítási/nyújtási lépés ismertetése a 4. fejezetben:

> **Tágítás/nyújtás (expansion):** Ha $f_r < f_l$, számítsuk ki a nyújtási pontot $x_e := c + \gamma(x_r - c)$ és $f_e := f(x_e)$. Ha $f_e < f_r$, elfogadjuk x_e-t és befejezzük az iterációt. Másképpen ($f_e \geq f_r$), elfogadjuk x_r-t és befejezzük az iterációt.

> Ez "mohó minimalizálás" ami magába foglalja befejezéskor a két legjobb pontot x_r, x_e az új szimplexben, és a szimplexet csak akkor nyújtjuk, ha $f_e < f_r < f_l$. Ezt használják a legtöbb kivitelezésben és az elméletben szintén (Lagaris, 1998). [7]

A klasszikus Nelder-Mead 1965-ös értekezés "mohó tágítást" használ, ahol x_e-t elfogadjuk, ha $f_e < f_l$ és $f_r < f_l$, függetlenül az f_r és f_e közötti összefüggésre. Az történhetett, hogy $f_r < f_e$, így x_r jobb új pont lesz, mint x_e, és még is inkább x_e-t fogadjuk el az új szimplexre. A szimplexet tartsuk olyan nagyként amennyire csak lehetséges a túlságosan korai befejeződést elkerülendő, amint az bizonyos esetekben nagyon hasznos tud a sima függvényekre.

Lagaris-féle leírás [7] és az előbbi Wright féle [22] között nincs nagy különbség. Mindössze annyi, hogy a szimplex transzformációs algoritmus sorba rendezés lépésénél a Lagaris-féle elképzelés szigorúan megköveteli a csúcsok rendezését.

Ilyen megvalósításban az S csúcsai sorba vannak rendezve a függvényértékekre vonatkozóan: $f_0 \leq f_1 \leq \ldots \leq f_{n-1} \leq f_n$. Ekkor az előbbiek így módosulnak: $l = 0$, $s = n - 1$, és $h = n$.

Bár az ebben a fejezetben ismertetett tulajdonságok némelyike már szerepelt az eddigi fejezetekben, de a célom ezek megismétlésével az volt, hogy egy helyen összefoglaljam a főbb különbségeket a három algoritmus között.

6. fejezet

Módszerek a változatos elnevezések mögött

Az eddigiekben láttuk, hogy az eredeti 1965-ös [11] Nelder-Mead módszer mennyiben tér el az 1996-os [22] és 1998-as [7] kissé módosított változataitól.

A témában publikáló cikkek változatos neveken hivatkoznak a Nelder-Mead algoritmusra:

1. Amoeba algorithm
2. Nelder-Mead direct search algorithm
3. Nelder-Mead simplex algorithm
4. Downhill simplex algorithm
5. Simplex search algorithm

Felmerül a kérdés, hogy ezek a módszerek vajon az eredeti 1965-ös [11] algoritmust fedik, 1996 [22] vagy esetleg a 1998-ast [7], avagy egy-egy módosításai a Nelder-Mead algoritmusnak?

1. **Nelder-Mead simplex algorithm**

   1965-ös [11] cikknek Nelder és Mead "A simplex method for function minimization" címet adta, mint a 3. fejezetben láttuk, nem véletlenül. Egy szimplex-szel közelítik a célfüggvény minimumát.

   Margaret H. Wright "Direct Search Methods: Once Scorned, Now respectable" (1996) [22] című cikkében már Nelder-Mead szimplex algoritmusként hivatkozik Nelderék munkájára. Egyszerűen a szerzők nevét használta és módszerük jellegzetességét.

   Később az 1998-as cikkben [7] (Convergence properties of the Nelder-Mead Simplex Algorithm in Low Dimensions) Lagaris és szerző társai szintén e néven nevezik a

módszert. Valószínűleg nem véletlenül, ugyanis a szerzők között ott van Margaret H. Wright is.

Habár mint fent láttuk, mindkét cikk már nem az eredeti Nelder-Mead algoritmust használja, hanem annak egy-egy enyhén módosított verzióját. Tehát tulajdonképpen a Nelder-Mead simplex algorithm a Módosított klasszikus Nelder-Mead módszert fedi, annak ellenére, hogy kezdetben a két cikk így hívja az eredeti algoritmust is.

K.I.M. McKinnon: "Convergence of the Nelder-Mead simplex method to a non-stationary point" [10] cikk is a Módosított klasszikus Nelder-Mead módszert ismerteti Nelder-Mead simplex algorithm-ként.

2. Nelder-Mead direct search algorithm

A Nelder-Mead algoritmus a direkt kereső algoritmusok családjába tartozik. Valószínűleg innen ered, hogy Margaret H. Wright a "Direct Search Methods: Once Scorned, Now respectable" [22] cikkében ilyen néven is nevezi a Módosított klasszikus Nelder-Mead módszert.

Továbbá C. T. Kelley: "Detection and Remediation of stagnation in the Nelder-Mead Algorithm using a sufficient decrease condition" [4] munkájában is hasonlóan jár el. Ő is ilyen néven ismerteti a Módosított klasszikus Nelder-Mead algoritmust.

3. Downhill simplex Algorithm

William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery [Numerical Recipes in C] [21] könyvükben ismertetik a Módosított klasszikus Nelder-Mead algoritmust Downhill simplex algorithm néven.

Nem csak a matematika területén fut változatos neveken a Nelder-Mead algoritmus, sok más tudományágban is különböző neveken hivatkoznak az 1996/98-as Nelder-Mead algoritmusra [22]/ [7]. Csak néhány példa:

(a) Tapas Kanungo and Qigong Zheng: A Downhill Simplex Algorithm for Estimating Morphological Degradation Model Parameters [20]

(b) Maksat Ashyraliyev, Yves Fomekong-Nanfack, Jaap A. Kaandorp, Joke G. Blom: Systems biology: parameter estimation for biochemical models [9]

(c) MacKnight, M.; Horch, E. P.: Calculating Visual Binary Star Orbits with the Downhill Simplex Algorithm (Amoeba) [8]

Viszont akadnak eltérések is. Vannak szerzők, akik kissé másképp ismertetik a Downhill simplex Algorithm-t, esetleg más paraméter értékkel a szimplex átalakításokra:

(a) Satoru Hiwa, Tomoyuki Hiroyasu, Mitsunori Miki: Downhill Simplex method [15]

4. **Amőba algoritmus**

Amint a MatLab-ban a Módosított klasszikus Nelder-Mead algoritmus "fminsearch" néven fut, úgy az előbbi felsorolási pontban említett "Numerical Recipes in C" [21] könyvben pedig "amoeba" néven lehet meglelni. Innen az amőba algoritmus elnevezés, egyébként valószínűleg a szimplex keresés közbeni mozgása (nagyon hasonlóan mozog egy amőba is) ihlette az amőba nevet.

Az amőba algoritmusról Christopher H. Brooks [1] is írt egy értekezést "An Introduction to Amoeba" címmel, ahol ezzel a névvel hivatkozik a Nelder-Mead algoritmusra. Az értekezést kielemezve látjuk, hogy a Módosított klasszikus Nelder-Mead algoritmusról van szó.

5. **Simplex search algorithm**

A módosított klasszikus Nelder-Mead algoritmust Singer és Nelder [Complexity Analysis of Nelder–Mead Search Iterations] [17], [Efficient Implementation of the Nelder-Mead Search Algoritm] [18], [Sasa Singer and John Nelder (2009), Scholarpedia, 4(7) :2928] [16] munkáikban Simplex search algorithm-nak is nevezik.

Tehát a fent említett öt algoritmus csak egy-egy név variációja - a standard Nelder-Mead algoritmusként a köztudatba került - Módosított klasszikus Nelder-Mead algoritmusnak (1998) [7]. Néhány esetben az elnevezések egy korábbi módszert fednek: az 1996-os Nelder-Mead módszert [22]. Viszont ahogy láttuk a 1996-os [22] és 1998-as [7] között a legfontosabb különbség: az utóbbi a függvényértékek sorba rendezését követeli meg.

Amikor tehát általánosságban Nelder-Mead módszerről beszélünk, az már koránt sem az eredeti 1965-ös [11] algoritmus, tehát megállapítható, hogy a fent említett öt elnevezés nem különbözik egymástól, mind ugyanazt a módszert fedik.

7. fejezet

A klasszikus Nelder-Mead algoritmusok konvergenciája

A Nelder-Mead algoritmusok konvergenciájának elemzése során felvetődik a kérdés: mi lenne az ideális állapot? Ha bizonyítható lenne a módszer konvergenciája bárhol és bármely dimenzióban.

Ugyanakkor mi a valóság? Alapvetően általánosságban el lehet mondani, hogy az algoritmus nem konvergens. Ez alól vannak bizonyos megkötésekkel kivételek.

Az 1998-as publikációig [7] nem született igazán említésre méltó eredmény, vagy azok is csak a negatív eredményeket bizonyították, mégpedig, hogy hol és hogyan biztosan nem konvergens a módszer. Lagarisék [7] a cikkükben ismertetnek néhány tételt, amikor konvergens lesz az algoritmus. Vizsgáljuk meg ezeket:

A következő definíciók bármely k. iterációra alkalmazhatóak, ezért a túl sok jelölés miatti tévedést elkerülendő, elhagyjuk a definíciókból a k. iteráció megjelölését. Így a definíciókban szereplő leírások, szimplexek stb. a k. iterációra vonatkoznak.

1. Definíció. *S szimplex legyen egy $n \times (n+1)$ mátrixszal megadva, melynek oszlopai a csúcsok: $S = (x_0, \ldots, x_n) = (B \quad x_n)$, ahol $B = (x_0, \ldots, x_{n-1})$*

2. Definíció. *Bármely S szimplexre $\mathbf{R}^n$-en definiáljuk M -et, mint egy $n \times n$ -es mátrixot, aminek j. oszlopa reprezentálja a szimplex "élét" x_{j-1} és x_n között: $M \equiv (x_0 - x_n \quad x_1 - x_n \quad \ldots \quad x_{n-1} - x_n) = B - x_n e^T$, ahol $e = (1, \ldots, 1)^T$.*

1. Megjegyzés. *n dimenziós szimplex térfogata: $vol(S) = \frac{|\det(M)|}{n!}$*

3. Definíció. *S szimplex nemdegenerált, ha M nemszinguláris, vagy ekvivalensen, ha $0 < vol(S)$.*

2. Megjegyzés. *A szimplex térfogata egyértelműen csak a csúcsok koordinátáitól függnek, és a sorrendjüktől nem.*

3. Megjegyzés. *S szimplex átmérője: $diam(S) = \max_{i \neq j} \|x_i - x_j\|_2$*

4. Definíció (Szigorú konvexitás)**.** *Az f függvény szigorúan konvex $\mathbf{R}^n$-en, ha minden y, z pontpárra ($y \neq z$) , és minden λ skalárra ($0 < \lambda < 1$),*

$$f(\lambda y + (1-\lambda)z) < \lambda f(y) + (1-\lambda) f(z).$$

4. Megjegyzés. *Amikor az f függvény szigorúan konvex $\mathbf{R}^n$-en, és $c = \sum_{i=1}^{l} \lambda_i z_i$, ahol $0 < \lambda_i < 1$ és $\sum_{i=1}^{l} \lambda_i = 1$, akkor $f(c) < \sum_{i=1}^{l} \lambda_i f(z_i)$ és így $f(c) < \max\{f(z_1), \ldots, f(z_l)\}$.*

1. Állítás. *Tegyük fel, hogy f függvény szigorúan konvex $\mathbf{R}^n$-en, és hogy a Módosított klasszikus Nelder-Mead algoritmus az f-re vonatkozóan már az elején nemdegenerált S kezdő szimplexszel rendelkezik. Ekkor nincsenek zsugorító lépések az algoritmus végrehajtása alatt.*

1. Tétel (Egy dimenziós Módosított klasszikus Nelder-Mead algoritmus konvergenciája)**.** *Legyen az f függvény szigorúan konvex $\mathbf{R}^1$-en korlátos szinthalmazokkal. Tegyük fel, hogy a Módosított klasszikus Nelder-Mead algoritmus az f-re vonatkozóan $0 < \alpha$, $0 < \beta < 1$, $1 < \gamma$, $\alpha < \gamma$, $1 \leq \alpha\gamma$ paraméterekkel már az elején nemdegenerált S kezdő szimplexszel rendelkezik. Ekkor a Módosított klasszikus Nelder-Mead intervallum mindkét végpontja konvergál $x_{\min}$-hez.*

1. Lemma. *Legyen az f függvény szigorúan konvex $\mathbf{R}^1$-en korlátos szinthalmazokkal. Tegyük fel, hogy a Módosított klasszikus Nelder-Mead algoritmus az f-re vonatkozóan $0 < \alpha$, $1 < \gamma$, $\alpha < \gamma$, $1 \leq \alpha\gamma$ paraméterekkel már az elején nemdegenerált S_0 kezdő szimplexszel rendelkezik. Ekkor létezik egy legkisebb egészértékű K, amire $K \leq \frac{\left|x_{\min} - x_1^{(0)}\right|}{diam(S_0)}$, úgyhogy $f_2^{(K)} \geq f_1^{(K)}$ és $f_e^{(K)} \geq f_1^{(K)}$. Ebben az esetben, $x_{\min} \in int(x_2^{(K)}, x_e^{(K)})$, és azt mondjuk, hogy $x_{\min}$ az $x_2^{(K)}$ és $x_e^{(K)}$ közé esik. (Az S_k, $x^{(k)}$, $f^{(k)}$,stb.: a k. iterációs lépésre vonatkozó szimplexet, aktuális csúcsokat, függvényértékeket stb. jelöli)*

2. Tétel (Egy dimenziós Módosított klasszikus Nelder-Mead algoritmus lineáris konvergenciája $\alpha = 1$-re)**.** *Legyen f függvény szigorúan konvex $\mathbf{R}^1$-en korlátos szinthalmazokkal. Tegyük fel, hogy a Módosított klasszikus Nelder-Mead algoritmus az f-re vonatkozóan $\alpha = 1$, $0 < \beta < 1$, $1 < \gamma$ paraméterekkel már az elején nemdegenerált S_0 kezdő szimplexszel rendelkezik. Ekkor az egész értékű M csak a β-tól és γ-tól függ, úgyhogy*

$$diam(S_{k+M}) \leq \tfrac{1}{2} diam(S_k), \text{ minden } k \geq K\text{-ra,}$$

ahol K és k iterációs indexek az előző lemmának megfelelően.

3. Tétel (két dimenziós csúcs függvényértékek konvergenciája)**.** *Legyen f függvény szigorúan konvex $\mathbf{R}^2$-en korlátos szinthalmazokkal. Tegyük fel, hogy a Módosított klasszikus Nelder-Mead algoritmus az f-re vonatkozóan $\alpha = 1$, $\beta = \frac{1}{2}$ paraméterekkel már az elején nemdegenerált S_0 kezdő szimplexszel rendelkezik. Ekkor a három határoló csúcs függvény megegyezik, így: $f_1^* = f_2^* = f_3^*$ (ahol az f_i^* meghatározása: $f_i^* = \lim_{k\to\infty} f_i^{(k)}$).*

2. Lemma (Szimplex térfogatok konvergenciája a nullához tart)**.** *Legyen az f függvény szigorúan konvex $\mathbf{R}^2$-en korlátos szinthalmazokkal. Tegyük fel, hogy a Módosított klasszikus Nelder-Mead algoritmus az f-re vonatkozóan $\alpha = 1$, $\beta = \frac{1}{2}$, $\gamma = 2$ paraméterekkel már az elején nemdegenerált S_0 kezdő szimplexszel rendelkezik. Ekkor az algoritmus generálta szimplexekre (S_k-kra) : $\lim_{k\to\infty} vol(S_k) = 0$.*

4. Tétel (Szimplex átmérők konvergenciája a nullához tart)**.** *Legyen f függvény szigorúan konvex $\mathbf{R}^2$-en korlátos szinthalmazokkal. Tegyük fel, hogy a Módosított klasszikus Nelder-Mead algoritmus az f-re vonatkozóan $\alpha = 1$, $\beta = \frac{1}{2}$, $\gamma = 2$ paraméterekkel már az elején nemdegenerált S_0 kezdő szimplexszel rendelkezik. Ekkor az algoritmus generálta szimplexekre (S_k-kra) : $\lim_{k\to\infty} diam(S_k) = 0$.*

5. Megjegyzés. *Az utolsó tételben nincs megerősítve, hogy a szimplexek mindig ugyanabba a pontba konvergálnak. Viszont nincs ismert példa, hogy ne így lenne.*

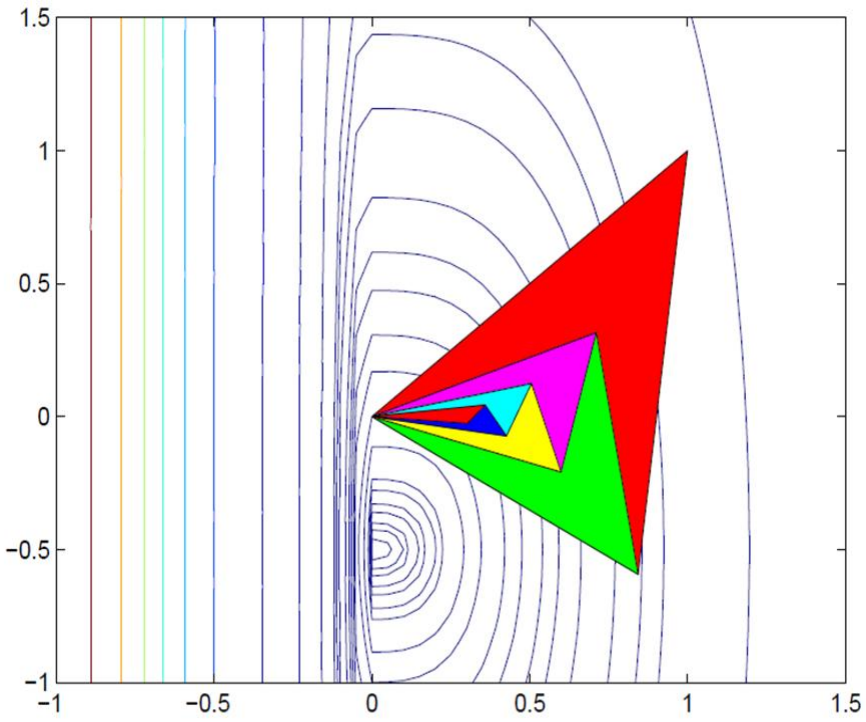
Az itt ismertetett állítások, tételek mind a fent említett 1998-ban Lagaris és szerző társai által publikált cikkből [7] származnak, ahol részletes bizonyításokkal támasztják ezeket alá.

Most nézzünk a "negatív eredmények" közül is példát. Ez talán az egyik leghíresebb és legfontosabb publikáció a Nelder-Mead módszer nem konvergens voltára.

Sokáig az a tévhit létezett, hogy a szigorúan konvex függvényeken két dimenzióban bárhol a Módosított klasszikus Nelder-Mead módszer konvergens. Erre adott Mckinnon [10] egy paraméterekkel megalkotott ellenpéldát. Ez a legsimább példája:

$$f_m(x,y) = \begin{cases} 6x^3 + y + y^2, & ha\ x \geq 0 \\ 2400\,|x|^3 + y + y^2, & ha\ x \leq 0 \end{cases}$$

Legyen a kezdő szimplex csúcsai: $(0,0)$, $(1,1)$ és $((1+\sqrt{33})/8, (1-\sqrt{33})/8))$. Ez az f_m függvény kétszer folytonosan differenciálható, és a Hesse-mátrixa pozitív definit.



7.1. ábra. A Módosított klasszikus Nelder-Mead algoritmus eltéveszti a minimumot a McKinnon ellenpéldán [6]

8. fejezet

Módosított Nelder-Mead algoritmusok

Ahogy láttuk az előző fejezetben a Nelder-Mead algoritmus konvergenciája egyáltalán nem megbízható magasabb dimenzióban, de már két dimenzióban is csak kikötésekkel érvényesül.

Nézzünk most két módszert, melyek Nelder-Mead algoritmussal dolgoznak továbbra is, de ezeket a jobb konvergencia lehetősége érdekében kissé módosították.

Először két dimenzióban, majd magasabb dimenzióban ismertetek egy-egy módosított módszert. Ezután pedig megvizsgálom, hogy ha a két eljárás előnyös tulajdonságait ötvözöm egy általam kidolgozott új Nelder-Mead módszerben, vajon érhető-e el javulás.

8.1. Restricted Nelder-Mead algoritmus (RNM)

8.1.1. Restricted Nelder-Mead algoritmus

Lagaris, Poonen, Wright a 2011-es publikációjukban [6] szeretnék leegyszerűsíteni a sokféle már létező Nelder-Mead algoritmus verziókat. Szerintük inkább korlátozni kellene az engedélyezett mozgások számát, semmint fixálni a módszert.

Az Módosított klasszikus Nelder-Mead [7], [22] algoritmusban négyféle mozgatás engedélyezett: tükrözés, tágítás, összehúzás, zsugorítás. A tágítás standard paraméterek mellett kétszeresére engedi növelni az adott szimplex térfogatát, míg a többi mozgásnál ugyanakkora marad vagy csökken. A szimplexnek és a mozgásának egy idő utáni stagnálását a tágítás létezésében vélik felfedezni. Szerintük gondosan mérlegelt "kis lépésekkel" ki lehetne ezt küszöbölni. Ezért az általuk módosított Nelder-Mead módszer (Restricted Nelder-Mead algoritmus) [6] egyáltalán nem használ tágító lépéseket. Teljesen kitörölték a létezését az eljárásukból. Két dimenzióban megkötésekkel pedig bizonyítják is az új módszerük konvergenciáját. (Ezen konvergencia bizonyítását terjedelmi okok miatt nem ismertetem, de az értekezésük [6] ezt részletesen tartalmazza.)

Tehát ezen leegyszerűsített Nelder-Mead algoritmus verzió szerint:

Adottak:

1. A minimalizálandó függvény $f : \mathbf{R}^n \to \mathbf{R}$.

2. Legyenek $x_1, \ldots, x_{n+1}$ a nemdegenerált szimplex csúcsai $\mathbf{R}^n$-en.

Restricted Nelder-Mead algoritmus egy iterációja standard paraméterekkel:

1. **Sorba rendezés**: Lagarisék [7] 1998-as cikke szerint $f(x_1) \leq f(x_2) \leq \ldots \leq f(x_{n+1})$.

2. **Tükrözés**: kiszámítani a $c = \frac{\sum_{i=1}^{n} x_i}{n}$ ($n+1$. kivételével az n db legjobb átlaga), ezután kiszámítjuk a tükrözési pontot: $x_r = 2c - x_{n+1}$ és kiértékeljük $f_r = f(x_r)$. Ha $f_r < f_n$ elfogadjuk a tükrözési pontot x_r és lezárjuk az iterációt.

3. **Összehúzás**: Ha $f_r \geq f_n$, akkor végrehajtjuk az összehúzást c, a legjobb x_{n+1} és x_r pontok között.

   (a) **Külső összehúzás**: Ha $f_n \leq f_r < f_{n+1}$ (x_r szigorúan jobb, mint x_{n+1}), végrehajtunk egy külső összehúzást. Számítsuk ki a külső összehúzási pontot: $x_{out} = \frac{1}{2}(c + x_r)$, és értékeljük ki $f_{out} = f(x_{out})$. Ha $f_{out} \leq f_r$, akkor elfogadjuk x_{out}-ot és leállítjuk az iterációt, egyébként ugrás a 4. lépésre (zsugorításra).

   (b) **Belső összehúzás**: Ha $f_r \geq f_{n+1}$, végrehajtunk egy belső összehúzást. Számítsuk ki a belső összehúzási pontot: $x_{in} = \frac{1}{2}(c + x_{n+1})$, és értékeljük ki $f_{in} = f(x_{in})$. Ha $f_{in} \leq f_{n+1}$, akkor elfogadjuk x_{in}-ot és leállítjuk az iterációt, egyébként ugrás a 4. lépésre (zsugorításra).

4. **Zsugorítás**: Számítsuk ki f n db pontjára: $v_i = \frac{1}{2}(x_1 + x_i)$ $i = 2, \ldots, n+1$. A szimplex rendezetlen csúcsai a következő iterációban: $x_1, v_2, \ldots, v_{n+1}$.

6. Megjegyzés. *Restricted Nelder-Mead algoritmus ezeket eredményezi:*

1. *Az egyetlen új csúcs – az elfogadott pont – visszahelyezi a legrosszabb csúcsot x_{n+1}-et a csúcsok halmazába a következő iterációban.*

2. *Ha a zsugorítás végrehajtódott n új pontra, ezek együtt a x_1-gyel alakítják a következő iterációban a szimplexet.*

8.1.2. Restricted Nelder-Mead algoritmus konvergenciája

5. Definíció. *Legyen $\mathfrak{F}$: a kétszer folytonosan differenciálható függvények halmaza, $f : \mathbf{R}^2 \to \mathbf{R}$ korlátos szinthalmazokkal és mindenhol pozitív definit Hesse-mátrixszal.*

5. Tétel. *Ha a Restricted Nelder-Mead algoritmust alkalmazzuk az $f \in \mathfrak{F}$-re, egy nemdegenerált kezdő szimplexszel, akkor az algoritmus konvergál az f egyedüli minimumához.*

6. Tétel (Általánosítás)**.** *Ha $f \in \mathfrak{F}$ és $g : \mathbf{R} \to \mathbf{R}$ szigorúan növő függvény, akkor a Restricted Nelder-Mead algoritmus konvergenciája vonatkozik $\tilde{f} = g \circ f$-re is, mert a Restricted Nelder-Mead algoritmus lépései $\tilde{f}$-ra azonosak f-re alkalmazott lépéseivel.*

8.1 Algoritmus:

Restricted Nelder-Mead (2011) (a standard paraméterekkel) [6]

Bemenő adatok:
Kezdő szimplex, leállási feltétel
Begin
kezdő szimplex megadása: számítsuk ki a kezdő x_i-ket és $f(x_i)$-ket;
While leállási feltétel nem teljesül **do**
Rendezzük sorba az$f(x_i)$-ket;
Számítsuk ki c-t;
$x_r = 2c - x_{n+1}$;
Számítsuk $f(x_r)$-t;
If $f(x_n) \leq f(x_r)$ **then**
If $f(x_r) < f(x_{n+1})$ **then**
$x_c := \frac{c+x_r}{2}$;
Számítsuk ki $f(x_c)$-t;
If $f(x_r) < f(x_c)$ **then**
$\forall\ x_j$-re $(j \neq 1)$: $x_j := \frac{(x_j+x_1)}{2}$ és $f(x_j)$;
else
Cseréljük x_r -et x_c-re;
endif
else
$x_c := \frac{c+x_{n+1}}{2}$;
Számítsuk ki $f(x_c)$-t;
If $f(x_{n+1}) < f(x_c)$ **then**
$\forall\ x_j$-re $(j \neq 1)$: $x_j := \frac{(x_j+x_1)}{2}$ és $f(x_j)$;
else
Cseréljük x_{n+1} -et x_c-re;
endif
endif
else
Cseréljük x_{n+1} -et x_r-re;
endif
endwhile
End

3. Lemma. *Tegyük fel, hogy a Restricted Nelder-Mead algoritmust alkalmazzuk egy szigorúan konvex kétváltozós függvényre korlátos szinthalmazokkal. Ekkor bármely nemdegenerált kezdő szimplexre a Restricted Nelder-Mead algoritmus szimplexek átmérője konvergál a* 0*-hoz az algoritmus folyamán.*

7. Megjegyzés. *Az Módosított klasszikus Nelder-Mead algoritmusban (fontos: most csak olyat vegyünk, melynél a folyamat konvergál a minimumhoz) a tágítási lépések soha nem történtek a minimum közelében. A tágítási lépések tipikusan a módszer korai szakaszában történtek, amikor még "helyben körvonalazódik" a szimplex alakformálása. Ezért a Restricted Nelder-Mead algoritmus közel hasonlóan viselkedik a Módosított klasszikus Nelder-Mead módszerhez a minimum közelében.*

8.2. Adaptive Nelder-Mead módszer (ANMS)

8.2.1. Adaptive Nelder-Mead módszer

Most Fuchang Gao és Lixing Han 2010-es publikációját [2] mutatom be.

Kezdésnek vegyünk egy érdekes kérdést: mi történik magasabb dimenzióban a Nelder-Mead módszerrel? Elsőre is bonyolultnak tűnik rá a válasz, de ha megnézzük alaposan a kérdést, akkor értjük meg csak igazán, milyen komplikált ez az ártatlannak látszó kérdés.

Láttuk a 7. fejezetben, hogy már két dimenzióban is bonyolódik a módszer konvergenciájára és viselkedésére vonatkozó eredmények. Vajon akkor hogy alakul ez magasabb dimenzióban?

Lagarisék [7] igazolták, hogy ha f szigorúan konvex és $n = 2$, akkor a Módosított klasszikus Nelder-Mead algoritmus szimplexeinek átmérője konvergál a nullához. Ezt szeretnénk átvinni három dimenzióra és a fölé, azzal a plusz megkötéssel, hogy f erősen konvex.

Erre mindössze sejtéseket lehet felállítani. Nem lehet semmit bizonyítani, csupán numerikus futtatásokkal ellenőrizhetőek a felállított sejtések.

6. Definíció (Erősen konvex függvény)**.** *Az f függvény erősen konvex $\mathbf{R}^n$-en, ha létezik egy szigorúan növekvő függvény $\varsigma : [0, \infty) \to [0, \infty)$, úgyhogy $\varsigma(0) = 0$, és $\forall\ x, y$-ra $(\in \mathbf{R}^n)$ és $\forall\ 0 < t < 1$-re:*

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y) - t(1-t)\varsigma(\|x - y\|),$$

ahol $\|x - y\|$ jelöli az euklideszi távolságot x és y között $\mathbf{R}^n$-en.

8. Megjegyzés. *Az erősen konvex függvények definiálásához egy speciálisan választott függvényt használnak:*

$$\varsigma(t) = \tfrac{c}{2}t^2, \text{ ahol } 0 < c \text{ valamilyen konstans.}$$

2. Állítás. *Ha f kétszer folytonosan differenciálható függvény, akkor f erősen konvex, akkor és csak akkor ha ennek Hesse-mátrixa erősen pozitív definit.*

7. Tétel. *Tegyük fel, hogy $n \geq 2$. Feltesszük, hogy f erősen konvex és a Módosított klasszikus Nelder-Mead algoritmus az általános paramétereket használja. Legyen S egy szimplex $\mathbf{R}^n$-en $\{x_1, \ldots, x_{n+1}\}$ csúcsokkal és $diam(S)$ az átmérő. Legyen*

$$F(S) = f(x_1) + \ldots + f(x_{n+1}).$$

Ha τ egy tágítás, belső összehúzás vagy külső összehúzás a Módosított klasszikus Nelder-Mead algoritmusban, akkor

$$F(\tau S) - F(S) \leq -\frac{(n-1)}{2n^2}\varsigma\left(\frac{1}{2}diam(S)\right).$$

1. Következmény. *Tegyük fel, hogy $n \geq 2$. Ha a célfüggvény f erősen konvex és a standard paraméterekkel a Módosított klasszikus Nelder-Mead algoritmust használja, akkor a szimplexeknek az átmérői konvergálnak a nullához.*

A tételben rámutattunk, hogy egy erősen konvex f-re a tágítási és összehúzási lépéseknek van egy elégséges csökkenő tulajdonsága, ha a szimplex átmérője nem túl kicsi. Ez a tulajdonság Fuchang Gao és Lixing Han szerint segítséget nyújthat néhány új sejtés felállításában, amik adnak néhány új magyarázatot a dimenzió hatásra a Nelder-Mead algoritmusban.

Tegyük fel, hogy f erősen konvex.

A tételből következik, hogy az $F(S)$ funkcionálban a csökkenést az okozza, hogy a tágítási és összehúzási lépések függenek az $\frac{n-1}{2n^2}$ faktortól. Ez csökken $n \geq 2$-re és konvergál nullához ($n \to \infty$). Ez sejteti, hogy a tágítási és összehúzási lépéseknek a hatékonysága csökken, ahogy a dimenzió n száma növekszik.

Szintén láthatjuk, hogy a nagyobb S szimplex átmérője is okoz jelentős csökkenést $F(S)$-ben egy tágítási vagy összehúzási lépésre. A megfigyelés magyarázza, miért segítheti az újrakezdés stratégiája nagyobb szimplexszel javítani a Nelder-Mead módszer végrehajtását magasabb dimenzióban. Szintén ad olyan javaslatot, hogy egy nagyobb szimplexszel a kezdés előnyösebb, ha a kezdőpont túl messze esik a minimumtól.

Az erősen konvex célfüggvényekre a Módosított klasszikus Nelder-Mead algoritmus soha nem használja a zsugorítási lépést. Így a tétel azt sugallja, hogy a Módosított klasszikus Nelder-Mead algoritmus magas dimenzióban való eredménytelensége valószínűleg a tükrözések magas számában keresendő.

8.2.2. Az alkalmazkodó paraméterek

Cél: csökkenteni a használt tükrözési lépések valószínűségét, és kikerülni a gyors csökkenést a szimplex átmérőben, így javítva a Módosított klasszikus Nelder-Mead [7] algoritmus teljesítményét nagy dimenziójú problémákra.

A kiindulópontul szolgáló megfigyelés: válasszuk a tágítási, összehúzási és zsugorítási paramétereket alkalmazkodó módon, és ezek legyenek összhangban a probléma dimenziójával (n).

Gyakorlatban: legyen $n \geq 2$

$$\alpha = 1, \qquad \gamma = 1 + \frac{2}{n}, \qquad \beta = 0,75 - \frac{1}{2n}, \qquad \delta = 1 - \frac{1}{n}.$$

9. Megjegyzés. α*: a tükrözési paraméter*
γ*: a tágítási paraméter*
β*: az összehúzási paraméter*
δ*: a zsugorítási paraméter*

Elnevezése: Adaptive Nelder-Mead módszer [2]

10. Megjegyzés. $n = 2$*-re az Adaptive Nelder-Mead módszer megegyezik a Módosított klasszikus Nelder-Mead [7] módszerrel.*

$\gamma = 1 + \frac{2}{n}$: meggátolja a szimplex rossz irányba való elhajlását, amit a tágító lépések okoznak magas dimenzióban.

$\beta = 0,75 - \frac{1}{2n}$: megkönnyíti a szimplex átmérő csökkenését, amikor a dimenzió nagy.

$\delta = 1 - \frac{1}{n}$: gátolja a szimplex átmérőjének meredek csökkenését, amikor a dimenzió nagy.

Ez a végrehajtandó tágítási és összehúzási lépések számának csökkenéséhez vezet egy adott célfüggvényre, nagyobb összhangban a tétellel.

Numerikus kísérleteken észlelték a tükrözési lépések számának visszaesését erősen konvex függvényekre a fenti módosított paraméterekkel.

8.3. Az általam kidolgozott: Adaptive Restricted Nelder-Mead módszer (ARNMS)

Eddig ismertettem az eredeti 1965-ös [11], a 1996-os [22], jelenleg standard algoritmusnak nevezett 1998-as [7], egy főleg két dimenzióra helyezett hangsúlyú 2011-es [6] és végül egy a többdimenziós esetet szem előtt tartott 2010-es Nelder-Mead algoritmust [2] .

A két legutóbbi algoritmus bemutatásánál észre lehet venni, hogy más és más szemszögből közelít a Módosított klasszikus Nelder-Mead algoritmus [7] továbbfejlesztéséhez.

A két dimenzióra összpontosító Restricted Nelder-Mead módszernek [6] és a több dimenzióban hatékony Adaptive Nelder-Mead algoritmusnak [2] az eredményeit megvizsgálva határoztam el, hogy a két egymástól eltérő megközelítésű módszer jó tulajdonságait ötvözöm egy általam kidolgozott Adaptive Restricted Nelder-Mead módszerben.

Veszem a standard paraméterekkel ismertetett Restricted Nelder-Mead algoritmust [6], és átdolgozom nem standard paraméterezetté: a szokásos α, β, γ és δ általános paraméterekkel.

Adottak: A minimalizálandó függvény $f : \mathbf{R}^n \to \mathbf{R}$. Legyenek $x_1, \ldots, x_{n+1}$ a nemdegenerált szimplex csúcsai $\mathbf{R}^n$-en.

Restricted Nelder-Mead algoritmus egy iterációja ***nem*** standard paraméterekkel (8.2 Algoritmus):

1. **Sorba rendezés**: Lagarisék [7] 1998-as cikke szerint $f(x_1) \leq f(x_2) \leq \ldots \leq f(x_{n+1})$.

2. **Tükrözés**: kiszámítani a $c = \frac{\sum_{i=1}^{n} x_i}{n}$ ($n+1$. kivételével az n db legjobb átlaga), ezután kiszámítjuk a tükrözési pontot: $x_r = c + \alpha(c - x_{n+1})$ és kiértékeljük $f_r = f(x_r)$. Ha $f_r < f_n$ elfogadjuk a tükrözési pontot p_r és lezárjuk az iterációt.

3. **Összehúzás**: Ha $f_r \geq f_n$, akkor végrehajtjuk az összehúzást c, a legjobb x_{n+1} és x_r között.

   (a) **Külső összehúzás**: Ha $f_n \leq f_r < f_{n+1}$ (x_r szigorúan jobb, mint x_{n+1}), végrehajtunk egy külső összehúzást. Számítsuk ki a külső összehúzási pontot: $x_{out} = c + \beta(x_r - c)$, és értékeljük ki $f_{out} = f(x_{out})$. Ha $f_{out} \leq f_r$, akkor elfogadjuk x_{out}-ot és leállítjuk az iterációt, egyébként ugrás a 4. lépésre (zsugorításra).

   (b) **Belső összehúzás**: Ha $f_r \geq f_{n+1}$, végrehajtunk egy belső összehúzást. Számítsuk ki a belső összehúzási pontot: $x_{in} = c + \beta(x_{n+1} - c)$, és értékeljük ki $f_{in} = f(x_{in})$. Ha $f_{in} \leq f_{n+1}$, akkor elfogadjuk x_{in}-ot és leállítjuk az iterációt, egyébként ugrás a 4. lépésre (zsugorításra).

4. **Zsugorítás**: Számítsuk ki f n db pontjára: $v_i = x_1 + \delta(x_i - x_1)$ $i = 2, \ldots, n+1$. A szimplex rendezetlen csúcsai a következő iterációban: $x_1, v_2, \ldots, v_{n+1}$.

8.2 Algoritmus:

Restricted Nelder-Mead (2011) (nem a standard paraméterekkel) [6]

Bemenő adatok:
Kezdő szimplex, leállási feltétel
Begin
kezdő szimplex megadása: számítsuk ki a kezdő x_i-ket és $f(x_i)$-ket;
While leállási feltétel nem teljesül **do**
Rendezzük sorba az $f(x_i)$-ket;
Számítsuk ki c-t;
$x_r := c + \alpha(c - x_{n+1})$;
Számítsuk $f(x_r)$-t;
If $f(x_n) \leq f(x_r)$ **then**
If $f(x_r) < f(x_{n+1})$**then**
$x_c := c + \beta(x_r - c)$;
Számítsuk ki $f(x_c)$-t;
If $f(x_r) < f(x_c)$ **then**
$\forall$ x_j-re $(j \neq 1)$: $x_j := x_1 + \delta(x_j - x_1)$ és $f(x_j)$;
else
Cseréljük x_r -et x_c-re;
endif
else
$x_c := c + \beta(x_{n+1} - c)$;
Számítsuk ki $f(x_c)$-t;
If $f(x_{n+1}) < f(x_c)$ **then**
$\forall$ x_j-re $(j \neq 1)$: $x_j := x_1 + \delta(x_j - x_1)$ és $f(x_j)$;
else
Cseréljük x_{n+1} -et x_c-re;
endif
endif
else
Cseréljük x_{n+1} -et x_r-re;
endif
endwhile
End

Ezek után vegyük az Adaptive Nelder-Mead [2] módszerben meghatározott paramétereket:

$$\alpha = 1, \qquad \gamma = 1 + \frac{2}{n}, \qquad \beta = 0,75 - \frac{1}{2n}, \qquad \delta = 1 - \frac{1}{n}.$$

Mivel a Restricted Nelder-Mead módszerből törölték a tágítást és az Adaptive Nelder-Mead módszerben pedig szó volt róla, hogy a tágítás az egyik olyan műveleti lépés, amelyet kisebbé kell paraméterezni, hogy ne legyen nagyon nagy a tükrözési lépések darabszáma, ezért a fenti négy Adaptive Nelder-Mead módszer paraméterei közül törölni lehet a tágítás paraméterét. Így ezek maradnak:

$$\alpha = 1, \qquad \beta = 0,75 - \frac{1}{2n}, \qquad \delta = 1 - \frac{1}{n}.$$

Ezen paramétereket alkalmazom a nem standard paraméterű Restricted Nelder-Mead módszerre, és így létrejött egy új módszer.

Ezt az új módosított Nelder-Mead módszeremet a két alapjául szolgáló módosított Nelder-Mead algoritmusról neveztem el: Adaptive Restricted Nelder-Mead módszernek. (8.3 Algoritmus)

8.3 Algoritmus:

Adaptive Restricted Nelder-Mead (2013)

Bemenő adatok:
Kezdő szimplex, leállási feltétel
Begin
kezdő szimplex megadása: számítsuk ki a kezdő x_i-ket és $f(x_i)$-ket;
While leállási feltétel nem teljesül **do**
Rendezzük sorba az $f(x_i)$-ket;
Számítsuk ki c-t;
$x_r := 2c - x_h$;
Számítsuk $f(x_r)$-t;
If $f(x_n) \leq f(x_r)$ **then**
If $f(x_r) < f(x_{n+1})$ **then**
$x_c := c + (0,75 - \frac{1}{2n})(x_r - c)$;
Számítsuk ki $f(x_c)$-t;
If $f(x_r) < f(x_c)$ **then**
$\forall\ x_j$-re $(j \neq 1)$: $x_j := x_1 + (1 - \frac{1}{n})(x_j - x_1)$ és $f(x_j)$;
else
Cseréljük x_r -et x_c-re;
endif
else
$x_c := c + (0,75 - \frac{1}{2n})(x_{n+1} - c)$;
Számítsuk ki $f(x_c)$-t;
If $f(x_h) < f(x_c)$ **then**
$\forall\ x_j$-re $(j \neq 1)$: $x_j := x_1 + (1 - \frac{1}{n})(x_j - x_1)$ és $f(x_j)$;
else
Cseréljük x_{n+1} -et x_c-re;
endif
endif
else
Cseréljük x_{n+1} -et x_r-re;
endif
endwhile
End

Két dimenzióban az Adaptive Restricted Nelder-Mead módszer megegyezik a Restricted Nelder-Mead módszerrel [6], de az elképzelésem az, hogy magasabb dimenzióban egyesíteni fogja a felvázolt sejtésem szerint a Restricted Nelder-Mead módszer [6] és az Adaptive Nelder-Mead módszer [2] jó tulajdonságait.

A 9. fejezetben az eddig ismertetett összes Nelder-Mead algoritmust numerikus teszteknek vetem alá, megvizsgálva a gyakorlatban is a teljesítményüket.

9. fejezet

Numerikus eredmények

9.1. Beprogramozott Nelder-Mead algoritmusok

Az általam vizsgálandó Nelder-Mead algoritmusok numerikus teszteléséhez a Matlab szoftvercsomagot alkalmaztam.

A Matlab szoftvercsomagba a Lagaris és társai által módosított algoritmust [7] programozták be a szakemberek, mint a Nelder-Mead módszernek a köztudatban 1998 óta klasszikusként elfogadott verzióját. Mivel a Módosított klasszikus Nelder-Mead módszer [7] bizonyítottan jobban teljesít a több homályos kérdést megválaszolatlanul hagyó 1965-ös eredeti cikkben [11] közölttel szemben, ezért természetes is, hogy az 1998-as [7] algoritmus került be a szoftvercsomagba.

A Matlab-ban a Módosított klasszikus Nelder-Mead módszert [7] "fminsearch" paranccsal tudjuk alkalmazni. Ehhez kapcsolódnak különböző állítható paraméterei a parancsnak.

Legegyszerűbb alakja:

$$xopt = fminsearch(fuggveny, x0)$$

ahol a $fuggveny$ értelemszerűen a minimalizálandó függvény, az $x0$ egy kezdőpont, és végül az $xopt$ a kapott végeredmény, a függvény egy lokális minimumpontja, jó esetben globális minimumpontja.

Nézzünk néhány állítható opciót:

1. $MaxIter$: ezzel lehet beállítani a megengedett maximális iterációszámot. Alap esetben: az optimalizált változók számának 200-szorosa. A tesztfüggvényeimnél a dimenziók számától függetlenül 30000-re állítottam.

2. $MaxFunEvals$: a végeredményhez szükséges függvényhívások darabszámának felső határát lehet behatárolni. Alap esetben: az optimalizált változók számának 200-szorosa. A tesztfüggvényeimnél a dimenziók számától függetlenül 30000-re állítottam.

3. $TolX$: azt szabja meg, hogy mikor számít elég jónak a kapott végeredmény. Milyen kicsi eltérést engedélyezzünk az optimalizálandó változók függvényében, hogy a megoldást már jónak nevezhessük. Alap esetben az értéke 10^{-4}. A tesztfüggvényeimnél megtartottam ezt a beállítást.

4. $TolFun$: az algoritmus folyamatát leállítja, ha a legutóbbi néhány iterációnál az optimalizálandó függvényértéke nem módosult többel, mint a $TolFun$ értéke. Alap esetben az értéke 10^{-4}. A tesztfüggvényeimnél megtartottam ezt a beállítást.

Amikor a kísérlettervezésben azt vizsgáljuk, hogy néhány adott dolog mennyiben különbözik egymástól, akkor természetesen vannak preferenciáink, hogy milyen tulajdonságok különbözőségét szeretnénk megfigyelni. Viszont szeretnénk elkerülni, hogy külső tényezők zavarják a megfigyelésünket. Hívhatjuk ezeket zajoknak is. Egy kísérletben minél kevesebb zajt szeretnénk, hogy minél optimálisabban figyelhessük meg az adott tulajdonságokban kissé eltérő modelljeink viselkedését.

Ezt az elvet tartottam szem előtt akkor is, amikor beprogramoztam az eddig ismertetett Nelder-Mead módszereket. Szerettem volna az algoritmusokban minél kisebbre szorítani a zajok hatását. Ezért mivel a Matlab már rendelkezik egy Nelder-Mead algoritmussal – mégpedig az előbb említett 1998-as Módosított klasszikus Nelder-Mead módszerrel [7] – így ezt vettem alapul, amikor a többi algoritmust is beprogramoztam. Ezen Matlab "fminsearch" nevezetű programot alakítottam úgy át, hogy a többi Nelder-Mead módszer mintegy belesimuljon az alap "fminsearch" program keretei közé.

Így a Nelder-Mead algoritmusok programjaiban a keret továbbra is azonos, tehát ez nem befolyásolja a teszt futtatások eredményeit. Minden program csak a maga jellemző tulajdonságával módosult. Például: a Restricted Nelder-Mead algoritmus [6] annyiban módosult a Módosított klasszikus Nelder-Mead módszerhez [7] képest, hogy töröltem belőle a Módosított klasszikus Nelder-Mead módszer [7] tágító lépését, a paramétereket és egyéb tényezőket hozzáigazítottam, hogy a Restricted Nelder-Mead algoritmust [6] kapjam, de egyéb tekintetben a két algoritmus kerete, leállási feltétele stb. ugyanaz maradt, így csökkentve a felesleges zajok mennyiségét.

Az általam beprogramozott Nelder-Mead módszerek Matlab fájl listája:

1. "fminsearch65": az 1965-ös eredeti cikkben közölt algoritmus [11]
2. "fminsearcha": a Módosított klasszikus Nelder-Mead módszer [7]
3. "fminsearchRNM": a Restricted Nelder-Mead algoritmus [6]
4. "fminsearchANMS": az Adaptive Nelder-Mead módszer [2]
5. "fminsearchARNMS": az általam kidolgozott Adaptive Restricted Nelder-Mead módszer

A Nelder-Mead programok, a tesztfeladatok és egyéb alkalmazott Matlab fájlok listája a Mellékletben megtekinthetőek, a fájlokat pedig mellékeltem a diplomamunkám mellé.

A tesztfeladatokhoz használt kezdőpontokat az adott célfüggvény engedélyezett határai között véletlenül generálom. Ezzel minél tágabb és változatosabb skáláját generálva a vizsgálati mintáknak.

9.2. Numerikus eredmények két dimenzióban

Alacsony dimenzióban bármelyik Nelder-Mead módszernek jobb a teljesítménye, mint magasabb dimenzióban. Ahogy eddig ismertettem, a Nelder-Mead módszereknek csak két dimenzióban van bizonyítható konvergenciájuk, természetesen még ekkor is csak kikötésekkel. Így az elvárásoknak megfelelően két dimenzióban hatásosabban fog optimalizálni, mint három dimenzióban.

A két dimenziós tesztfeladatok elemzése előtt nézzük újra a két legfontosabb dolgot:

1. Két dimenzióban az Adaptive Nelder-Mead módszer [2] megegyezik a Módosított klasszikus Nelder-Mead módszerrel [7].
2. Két dimenzióban az általam kidolgozott Adaptive Restricted Nelder-Mead módszer megegyezik a Restricted Nelder-Mead módszerrel [6].

Ennek gyakorlati következménye, hogy két dimenzióban elég három algoritmust tesztelni:

1. "fminsearch65": az 1965-ös eredeti cikkben közölt algoritmus [11]
2. "fminsearcha": a Módosított klasszikus Nelder-Mead módszer [7]
3. "fminsearchARNMS": az én Adaptive Restricted Nelder-Mead módszerem

A 9.1-es ábrán látszik, hogy 100 véletlenül generált ponton a három vizsgált Nelder-Mead algoritmus milyen hatékonysággal teljesített. Minden célfüggvénynél minden az engedélyezett határok között generált kiindulópontra mindhárom vizsgált Nelder-Mead módszert lefuttattam, és kiválasztottam a legjobb optimum értékűt a három futtatási eredmény közül. Ezt a pontot kihozó Nelder-Mead verzió került be egy ponttal a táblázatba.

Gyakorlatban nézzük az Ackley függvényt az engedélyezett határok között generált 100 darab kiindulópontra:

1. Módosított klasszikus Nelder-Mead módszer [7] egyszer hozott jobb eredmény egy kiindulópontra, mint a másik kettő futtatott algoritmus.
2. Az 1965-ös Nelder-Mead algoritmus [11] ugyanezt kétszer tette meg.
3. Ugyanakkor az általam kidolgozott Adaptive Restricted Nelder-Mead módszer 97-szer hozott jobb eredményt a többihez képest.

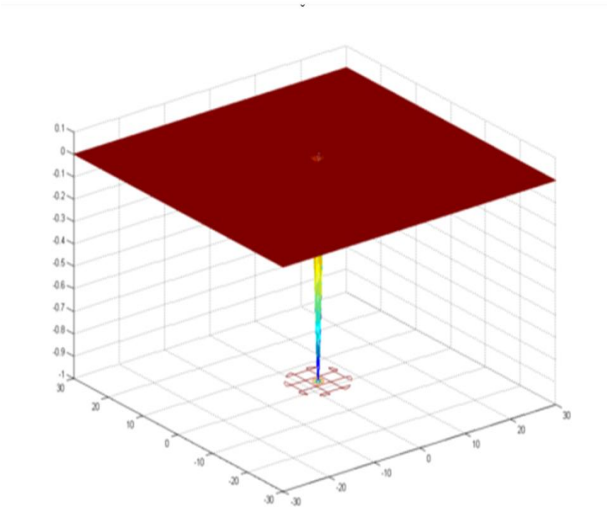
Egyébként a sejtéseknek megfelelően a Restricted Nelder-Mead módszer [6] a legtöbb esetben 70% felett teljesített, nem egy esetben 100% -ot ért el.

Az eredmény kiértékelésnél, ha két vagy több algoritmus azonosan jó minimum pontot hozott ki, akkor az az algoritmus lett kihozva a legjobbnak, amelyik kevesebb iteráció alatt érte ezt el. Ha az iteráció szám is egyezett, akkor a függvénykiértékelések száma döntött, ahol pedig kevesebb volt, az a módszer kapta végül a legjobb besorolást.

A célfüggvényeknél sárgával jelzett módszer hozta végül az adott 100 kezdőpont közül a legkisebb optimumot az adott tesztfeladatra. Viszont azt ki kell emelni, hogy a kihozott

Optimum 2 dim.	NMS	65NMS	ARNMS	f(x_opt)	x_opt_1	2
nem konvex						
Ackley fv.:	1	2	97	0	0	0
Beale fv.:	10	13	77	0	3	0.5
Branin fv.:				0.397887	-3,14	12,275
				0.397887	3,14	2,275
	29	10	61	0.397887	9,42478	2,475
Easom fv.:	0	0	100	-1	3,14	3,14
Goldstein &Price fv.:	17	10	73	3	0	-1
Griewank fv.:	8	5	87	0	0	0
Hump fv.:	43	16	41	0	-0,0898	0,7126
				0	0,0898	-0,7126
Levy fv.:	3	5	92	0	1	1
Michalewics fv.:	21	9	70	-1,80130341	2,2029	1,5708
Perm fv. P(n,b):				0	1	2
	13	4	83	0	1,76922732	1,07693043
Power fv.:	5	2	93	0	1	2
Rastrigin fv.:	1	0	99	0	0	0
Rosenbrock fv. 2.dim.:	5	1	94	0	1	1
Schwefel fv.:	4	2	94	-276,352794	-559,148605	-559,148621
Shubert fv.:	0	0	100	-186,7309	-7,08351838	-1,425137
szigorúan konvex						
Bohachevsky fv. (1):	13	2	85	0	0	0
Bohachevsky fv. (2):	6	1	93	0	0	0
Bohachevsky fv. (3):	5	1	94	0	0	0
Booth fv.:	31	6	63	0	1	3
Matyas fv.:	22	6	72	0	0	0
Sphere fv.:	0	0	100	0	0	0
Sum Squares fv.:	0	0	100	0	0	0
Trid fv.:	25	3	72	-2	2	2
Zakharov fv.:	11	1	88	0	0	0
erősen konvex						
Dixon & Price fv.:	26	4	70	0	1	-0,70710454
Freudenstein & Roth fv.:	22	11	67	0	5	4
				48,9842	11,41	-0,897
Perm fv. P(0,n,b):	18	8	74	0	1	0,5
(a.*x)*(a.*x)T	0	0	100	0	0	0

9.1. ábra. Két dimenziós tesztfeladatok futtatási táblázata



9.2. ábra. Az Easom függvény

optimum értéke mindig szorosan kapcsolódik ahhoz a kezdőponthoz, ahonnan az algoritmus indult. Tehát mindig jobb optimumot fog kihozni egy olyan kiindulási pont, ami közelebb esik az optimumhoz. A közeli kezdőpontból az algoritmus gyorsabb futás idővel fogja elérni a minimumhelyet, mintha egy távoli pontból tenné.

Jó példa erre az Easom függvény: (9.2 ábra).

Az 9.2-es ábrán látszik, hogy a kereső eljárásnak szinte bele kell esnie a lyukba, hogy eljusson a globális optimumhoz. A 100 darab generált kezdőpont közül mindössze egy találta meg a -1 pontban a legjobb lokális minimumát.

Mint a diplomamunkám mellé mellékelt Easom függvényről készült excel táblázatban látszik, a kezdőpont igen közel van a keresett értékhez:

1. a kapott optimum $(3,14161641; 3,141558004)$
2. a kezdőpont $(3,650358883; 1,553731293)$

A 9.1-es ábra jobb oldalán az adott célfüggvényhez tartozó valós optimumértékek vannak. A világos kékkel jelzett függvényeknek a 100 darab generált kezdőpontból egyik Nelder-Mead algoritmusnak se sikerült elérnie a globális minimumpontjaikat. Néhány függvénynél több eredeti globális optimumuk van feltüntetve. Ezeknek a függvényeknek több globális minimumpontjuk van és az algoritmus futtatásoknál hol ez, hol az az optimum került elő. A futtatási eredmények végül a 9.1-es ábrán ahhoz a globális optimumhoz kerültek feltüntetésre, amelyik a legjobb célfüggvény optimumot hozta a 100 darab kezdőpont közül.

A tesztfeladatokat három csoportra osztottam:

1. nem konvex
2. szigorúan konvex
3. erősen konvex

Az eddig ismertetett Nelder-Mead algoritmus hatékonyságok jól megfigyelhetők a három csoporton. A nem konvex csoportba tartozó tesztfüggvényeknél kettőnél is előfordult (Power függvény, Rastrigin függvény), hogy nem jött ki a várt globális optimum, az algoritmusok elakadtak egy-egy lokális optimumban. Sőt a többi ebbe a részbe tartozó tesztfüggvény esetében is gyakran el-elakadt egy-egy lokális minimumpontban.

Sokkal gyorsabban, kevesebb iteráció számmal kielégítő eredményt kaptam a szigorúan konvex csoportra. Erre kiváló példát nyújt a Bohachevsky függvény három típusa. A szigorúan konvex csoportba eső tesztfüggvények sokkal kevesebbszer akadnak el egy helyi minimumpontban.

Az erősen konvex csoportba eső tesztfüggvényeknél pedig nem is találtam nem az elvárt optimumba érkező eredményt. Jellemzően szinte végig kevés iteráció számmal dolgoztak a Nelder-Mead algoritmusok, egy-egy kirívó eset volt, amikor magasra futott az iteráció szám, de ezek inkább csak kivételek, melyek erősítik a szabályt.

Két dimenzióban elenyésző esetben nyújtott rosszabb teljesítményt a Restricted Nelder-Mead módszer [6], mint a két másik vizsgált algoritmus. Jellemzően az összes futtatásra vagy azonos, vagy jobb, de sokszor sokkal jobb eredményeket produkált, mint a másik két Nelder-Mead módszer. Többször látszódott, hogy az 1965-ös Nelder-Mead módszer [11] elkezdett stagnálni, míg a másik két Nelder-Mead módszer dolgozott tovább. A stagnálástól kezdve pedig hasznavehetetlenné válik a módszer.

9.3. Numerikus eredmények több dimenzióban

Magasabb dimenzióban már mind az öt beprogramozott Nelder-Mead algoritmust teszteltem, ugyanis itt már nincsenek azonos Nelder-Mead módszer verziók, mint két dimenzióban.

Tehát akkor az itt tesztelt Nelder-Mead algoritmusok:

1. ”fminsearch65” : az 1965-ös eredeti cikkben közölt algoritmus [11]
2. ”fminsearcha” : a Módosított klasszikus Nelder-Mead módszer [7]
3. ”fminsearchRNM” : a Restricted Nelder-Mead algoritmus [6]
4. ”fminsearchANMS” : az Adaptive Nelder-Mead módszer [2]
5. ”fminsearchARNMS” : az általam kidolgozott Adaptive Restricted Nelder-Mead módszer

Optimum több dim.	NMS	65NMS	RNMS	ANMS	ARNMS	f(x_opt)	x_opt
nem konvex							
Griewank fv. 8D:	11	8	1	80	0	0	(0; ... ;0)
Hartmann fv H(3,4):	2	2	1	0	95	-3,86278	(0,11; 0,55; 0,85)
Hartmann fv H(4,6):	22	18	31	17	12	-3,32237	(0,20; 0,15; 0,47; 0,27; 0,31; 0,67)
Levy fv. 4D:	12	14	14	13	47	0	(1; 1; 1; 1)
Michalewics fv. 5D:	5	0	2	0	93	-4,687658	?
Perm fv. P(n,b) 4D:	21	11	8	45	15	0	(1; 2; 3; 4)
Power fv. 4D :	21	7	4	51	17	0	(1; 2; 3; 4)
Rastrigin fv. 5D:	24	21	25	13	17	0	(0; ... ;0)
Rosenbrock fv. 5D:	29	19	24	18	10	0	(1; 1; 1; 1; 1)
Schwefel fv. 5D:	20	25	27	14	14	0	(1; 1; 1; 1; 1)
Shekel fv. 4D:	22	16	26	9	27	-10,5364	(4; 4; 4; 4)
Griewank fv. 20D:	1	0	5	86	8	0	(0; ... ;0)
Levy fv. 30D:	24	10	7	40	19	0	(1; ... ;1)
szigorúan konvex							
Sphere fv. 3D:	7	1	1	0	91	0	(0; ... ;0)
Sum Squares fv. 8D:	21	10	18	24	27	0	(0; ... ;0)
Trid fv. 6D:	23	20	23	23	11	-50	(6; 10; 12; 12; 10; 6)
Zakharov fv. 6D:	28	8	29	22	13	0	(0; ... ;0)
Sphere fv. 20D:	1	0	0	67	32	0	(0; ... ;0)
Trid fv. 20D:	0	0	0	99	1	-1520	(20; 38; 54; 68; 80; 90; 98; 104; 108; 110; 110; 108; 104; 98; 90; 80; 68; 54;38;20)
erősen konvex							
Colville fv. 4D:	53	34	0	0	13	0	(1; 1; 1; 1)
Dixon & Price fv. 6D:	30	18	18	18	16	0	(1; 0,707; 0,59; 0,54; 0,52; 0,51)
Perm fv. P(0,n,b) 4D:	17	29	10	25	19	0	(1; 0,5; 0,33; 0,25)
Powell fv. 8D:	17	10	8	41	24	0	(3; -1; 0; 1; 3; -1; 0; 1)
Dixon & Price fv. 20D:	0	0	0	58	42	0	(1; 0,707; 0,59; 0,54; 0,52; 0,51; 0,50; 050 ...)
Perm fv. P(0,n,b) 20D:	0	0	4	44	52	0	(1; 0,5; 0,33; 0,25)
Powell fv. 24D:	0	0	0	100	0	0	(3; -1; 0; 1; ... 3; -1; 0; 1)
(a.*x)*(a.*x)T	14	15	20	16	35	0	(0; ... ;0)

9.3. ábra. A többdimenziós tesztfeladatok futtatási táblázata

A 9.3-as ábrán látszik, hogy 100 véletlenül generált ponton az öt vizsgált Nelder-Mead algoritmus milyen hatékonysággal teljesített. Minden célfüggvénynél minden az engedélyezett határok között generált kiindulópontra mind az öt vizsgált Nelder-Mead módszert lefuttattam, és kiválasztottam a legjobb optimum értékűt a három futtatási eredmény közül. Ezt a pontot kihozó Nelder-Mead verzió került be egy ponttal a táblázatban.

Itt, mint a két dimenziós eseteknél, az eredmény kiértékelésnél, ha két vagy több algoritmus azonosan jó minimum pontot hozott ki, akkor az az algoritmus lett kihozva a legjobbnak, amelyik kevesebb iteráció alatt érte ezt el. Ha az iteráció szám is egyezett, akkor pedig a függvénykiértékelések száma döntött, ahol kevesebb volt, az a módszer kapta végül a legjobb besorolást.

A célfüggvényeknél sárgával jelzett módszer hozta végül az adott 100 kezdőpont közül a legkisebb optimumot az adott tesztfeladatra. Viszont azt ki kell emelni, hogy a kihozott optimum értéke mindig szorosan kapcsolódik ahhoz a kezdőponthoz, ahonnan az algoritmus indult. Tehát mindig jobb optimumot fog kihozni egy olyan kiindulási pont, ami közelebb esik az optimumhoz és természetesen gyorsabban is fogja ezt produkálni.

A 9.3-as ábra jobb oldalán az adott célfüggvényhez tartozó valós optimumértékek vannak. A világos kékkel jelzett függvényeknek a 100 darab generált kezdőpontból egyik Nelder-Mead algoritmusnak se sikerült elérnie a globális minimumpontjaikat.

A többdimenziós eseteknél már nagyon körülményes bármilyen konvergenciáról szóló elképzelést is mutatni. Ettől függetlenül a gyakorlati tapasztalat az, hogy a Nelder-Mead módszerek az elméletet meghazudtolva, működnek magasabb dimenzióban is. Nem szabad viszont abba a hibába esni, hogy feltétel nélkül elfogadjuk az algoritmusok eredményét. A Nelder-Mead módszer sokszor még két dimenzióban sem ad helyes végeredményt, nem beszélve a többdimenziós esetekről. Tehát a Nelder-Mead módszert szabad használni, sőt használjuk is, mivel a tapasztalat azt mutatja, hogy jól dolgozik az optimalizálásban, de sokszor csak segítségünkre van az optimum betájolására, de a helyes globális optimumot már nem tudja teljes biztonsággal megadni.

A két dimenziósok mintájára a tesztfeladatokat három csoportra osztottam:

1. nem konvex
2. szigorúan konvex
3. erősen konvex

Minél nagyobb a dimenzió szám, annál instabilabbá válik a módszer és annál könnyebben ad helytelen választ. Erre kiváló példa az erősen konvex csoportban vizsgált kvadratikus függvény, ugyanis alacsony dimenzióban rendben és gyorsan megtalálta a helyes optimumot, de a harminc dimenziós esetben már nagyon félrevitte a megoldást. Konkrétan 0 helyett $34151282,68$. Ez pedig nagyon nagy eltérés.

Érdekes ellenpontja az erősen konvex függvények között a Dixon & Price függvény. Ennek a tesztfeladatnak alacsonyabb, azaz hat dimenziós esetben szinte mindig kijött a helyes optimuma, de a húsz dimenziós esetben már ritkábban. Viszont húsz dimenziós esetben is csak $0,6$-es eltérést lehet látni a helyes és kissé félre mozdult megoldások között. 0 helyett

algoritmus	Iteráció	fv. kiértekelés	f(x_opt)
65NM search	11771	15196	1852,127
NM search	19079	23953	1269,81
RNM search	28031	28306	0,666667
ANM search	9891	12878	2,26E-07
ARNM search	21811	22353	0,666667

algoritmus	Iteráció	fv. kiértekelés	f(x_opt)
65NM search	885	1378	7,68E-08
NM search	389	604	2,01E-09
RNM search	1671	1751	9,12E-09
ANM search	491	811	0,666667
ARNM search	1333	1468	8,51E-09

9.4. ábra. A Dixon & Price függvény felül húsz dimenzióban, alul hat dimenzióban

sokszor $0,66$ jött ki húsz dimenzióban. Ilyen nagyon magas dimenzióban végül csak az Adaptive Nelder-Mead módszer [2] találta meg a helyes 0 értéket. A Restricted Nelder-Mead algoritmus [6] és az általam kidolgozott Adaptive Restricted Nelder-Mead módszer azonosan a $0,66$ -os értéket hozták ki, viszont mindezt az én Adaptive Restricted Nelder-Mead módszerem kissé jobb iteráció számmal tette ezt. Az 1965-ös eredeti cikkben közölt algoritmus [11] és a Módosított klasszikus Nelder-Mead módszer [7] nagyon kirívóan félre érkeztek, 1000-n felüli értéket kihozva. Érdekesség viszont, hogy hat dimenzióban viszont a legjobb értéket kihozó kezdőpontnál az Adaptive Nelder-Mead módszer [2] bukott csak el a $0,66$ -os értékével, a többi Nelder-Mead algoritmus a helyes 0 értékre érkezett. 9.4-es ábrán látszanak ezek az eltérések.

A szigorúan konvex csoportban minden függvénynek megtaláltam az optimumát, viszont ez nem a csoport jellemzője, csak egyszerűen így alakultak az adott tesztfeladatok. Nagyon szépen dolgozott az én Adaptive Restricted Nelder-Mead módszerem a Sphere függvényen, több, mint az esetek 90%-ban ez a Nelder-Mead módszer hozta a legjobb optimumot a kezdőpontokra.

Ebben a csoportban viszont a Trid függvényre húsz dimenzióban a helyes megoldást újra és újra csak az Adaptive Nelder-Mead módszer [2] hozta meg. Jóval lemaradva tőle hozta a megfelelő optimumot még az én Adaptive Restricted Nelder-Mead módszerem. A másik három módszer itt helyes eredményt szinte fel se tudott mutatni.

A nem konvex csoportba pedig a 9.3-as ábrát szemügyre véve azonnal látszik, hogy igen rendszertelen lett a futtatások eredménye. Sok függvénynek nem lett meg az optimuma, soknak pedig meg lett vagy épp, hogy elkerülte az optimalizáló eljárás (pl.: Michalewics függvény 5 dimenzióban). Az eljárások eredményessége is változó volt. Például a három dimenziós Hartmann függvényre a legtöbb, legjobb eredményeket megint csak az én Adaptive Restricted Nelder-Mead módszerem hozta a 95% arányával. Griewank függvényre húsz dimenzióban viszont már az Adaptive Nelder-Mead módszer [2] bizonyult a legjobbnak a maga 86%-val.

Magas dimenzióban jellemzően bizonytalanul dolgozik a Nelder-Mead módszer, és ez csak romlik a dimenzió szám növekedésével. Amit biztosan ki lehet jelenteni, hogy az 1965-ös eredeti cikkben közölt algoritmus [11] szinte mindig nagyon rosszul teljesített. A másik négy módszer között kisebb-nagyobb arányban eloszlik a hatékonysági arány. A tesztfüggvény futtatási adatokat elemezve, arra a megfigyelésre jutottam, hogy az, hogy melyik Nelder-Mead verzió hozza a legjobb eredményt egy célfüggvényre, az magától a célfüggvénytől függ. Kisebb mértékben pedig attól, hogy melyik kezdőpontból indítjuk az eljárást.

Sajnos nem lehet azt mondani a többdimenziós eseteknél, hogy ha egy adott célfüggvény konvex, vagy szigorúan konvex, esetleg erősen konvex, akkor ezt meg ezt a Nelder-Mead algoritmus verziót kell alkalmazni rá, mert az lesz hatékony. Szép lenne, ha egy adott csoportra tudnánk melyik Nelder-Mead verziót kell használni, mivel az adná a csoport függvényeire mindig a legjobb optimumot. Sajnos ez nem igaz. Egyes csoportokon belül is rendszertelen a módszerek hatékonysága. Egyes függvényekre az egyik módszer hatásos, a másikra pedig egy másik Nelder-Mead algoritmus variáns.

10. fejezet

Konkluzió

A diplomamunkámban bemutattam az egyik legnépszerűbb direkt kereső eljárást, amit az egyik leghatékonyabbként tartanak számon a gyakorlati probléma megoldásban. Bár az elméleti szakemberek körében a módszer nem örvend túl nagy népszerűségnek a sok nyitott kérdésével, és a dimenzió szám emelkedésével egyre inkább romló teljesítményével, ennek ellenére könnyű érthetőségével, egyszerű alkalmazhatóságával és alacsony tárigényével ideális eszköz a gyakorlati szakember kezében, akinek minél gyorsabban, minél kisebb költséggel, minél jobb optimum kell hogy a rendelkezésére álljon.

A Nelder-Mead algoritmus többször esett át kisebb nagyobb átalakításokon az 1990-es években, végül az 1998-ban kialakult Lagaris és társai féle Nelder-Mead módszer [7] került be a köztudatba, mint standard Nelder-Mead módszer. Eközben egyre több névvel illették a módszert: amőba algoritmus, downhill módszer stb. Amint azt a diplomamunkámban bemutattam, ezek az esetek legnagyobb részében ennek az 1998-as standard Nelder-Mead módszernek [7] az egyes név variánsai.

Ismertettem a módszer konvergenciáját. Az előbbi standard módszer [7] egy és két dimenzióban képes korlátozott keretek között felmutatni konvergencia tulajdonságot. Ennek fejlesztésére dolgozták ki a Restricted Nelder-Mead algoritmust [6], ami hatékonyabbnak is bizonyult a standardnál [7]. Magasabb dimenzióban már nagyon körülményes bármilyen konvergenciáról beszélni, csak sejtésekre és numerikus eredményekre lehet támaszkodni, semmi bizonyíthatóra. Ennek ellenére megpróbálkoztak szakemberek magasabb dimenzióban is hatékonyabb Nelder-Mead módszert fejleszteni, aminek az Adaptive Nelder-Mead módszer [2] lett a végeredménye, ami két dimenzióban maradt a standard algoritmus [7]. Ennek hatékonyabb futását numerikus eredményekkel és sejtésekre építkezve mutatták be.

A szakemberek továbbfejlesztett Nelder-Mead módszereinek hatékonyságát elemezve merült fel bennem az elképzelés, hogyan lehet ezt a két hatékony Nelder-Mead módszer–variánst ötvözni, hogy egy módszerben dolgozzanak együtt bármilyen dimenzió felett, összeadva ezáltal jó tulajdonságaikat. Ebből hoztam létre az általam elképzelt Adaptive Restricted Nelder-Mead módszert.

Numerikus tesztekkel bizonyítottam a sejtésemet, ami a következő eredménnyel zárult: két dimenzióban az általam kidolgozott Adaptive Restricted Nelder-Mead módszer valóban hatékony, megegyezik a Restricted Nelder-Mead algoritmussal [6]. Magasabb dimenzióban

pedig egyes célfüggvényekre is ez volt a legeredményesebb. Az Adaptive Nelder-Mead módszer [2] és a Restricted Nelder-Mead algoritmus [6] is jól szerepeltek. Változó, tehát hogy melyik célfüggvényre melyik Nelder-Mead módszer variáns a leghatékonyabb, sőt olyan optimalizálandó függvény is akadt magas dimenzióban, amire pedig a standard Nelder-Mead módszer bizonyult a legeredményesebbnek. Az elmondható, hogy az 1965-ös eredeti cikkben közölt – a maga idejében fontos felfedezést jelentő – algoritmus [11] idővel valóban elavulttá vált. A tesztelés során sokkal többet hibázott, mint bármelyik módosítása. Arra vonatkozóan nem találtam semmilyen mintát, hogy milyen módon lehetne csoportokba sorolni a célfüggvényeket, azaz hogy mely típusú függvényekre, melyik Nelder-Mead variáns alkalmazandó, hogy a leghatékonyabb eredményt kapjuk. Ez egy érdekes kérdés a jövőre nézve, hogy egyes Nelder-Mead verziókhoz meghatározott függvény típusokat lehessen hozzárendelni a még hatékonyabb optimalizálás érdekében.

A. függelék

Mellékletek

A diplomamunkám mellé csatolt fájlok:

A.1. Matlab fájlok

1. ackley2d: Ackley fv. 2dim
2. banana: Rosenbrock fv. 2dim
3. beale: Beale fv. 2dim
4. bh1: Bohachevsky fv. (1) 2dim
5. bh2: Bohachevsky fv. (2) 2dim
6. bh3: Bohachevsky fv. (3) 2dim
7. booth: Booth fv. 2dim
8. branin: Branin fv. 2dim
9. colville: Colville fv. 4dim
10. dp: Dixon & Price fv. 6dim
11. dp2: Dixon & Price fv. 2dim
12. dp20d: Dixon & Price fv. 20dim
13. easom: Easom fv. 2dim
14. fminsearch65: 1965-ös Nelder-Mead algoritmus [11]
15. fminsearcha: Módosított klasszikus Nelder-Mead módszer [7]
16. fminsearchANMS: Adaptive Nelder-Mead módszer [2]
17. fminsearchARNMS: Adaptive Restricted Nelder-Mead módszer

18. fminsearchRNM: Restricted Nelder-Mead algoritmus [6]
19. fr: Freudenstein & Roth fv. 2dim
20. fullbigfminsearch2d: két dimenziós esetben Nelder-Mead módszerek összehasonlítása
21. fullbigfminsearchtobbd: többdimenziós esetben Nelder-Mead módszerek összehasonlítása
22. gold: Goldstein &Price fv. 2dim
23. griewank: Griewank fv. 8dim
24. griewank2d: Griewank fv. 2dim
25. griewank20d: Griewank fv. 20dim
26. hart3: Hartmann fv H(3,4) 3dim
27. hart6: Hartmann fv H(4,6) 6dim
28. hump: Hump fv. 2dim
29. levy: Levy fv. 4dim
30. levy2d: Levy fv. 2dim
31. levy30d: Levy fv. 30dim
32. matyas: Matyas fv. 2dim
33. mich: Michalewics fv. 5dim
34. mich2d: Michalewics fv. 2dim
35. perm: Perm fv. P(n,b) 4dim
36. perm0: Perm fv. P(0,n,b) 4dim
37. perm02d: Perm fv. P(0,n,b) 2dim
38. perm2d: Perm fv. P(n,b) 2dim
39. perm020d: Perm fv. P(0,n,b) 20dim
40. powell: Powell fv. 8dim
41. powell24d: Powell fv. 24dim
42. powerr: Power fv. 4dim
43. powerr2d: Power fv. 2dim

44. qqq: (a.*x)*(a.*x)T fv. 2dim
45. qqq30d: (a.*x)*(a.*x)T fv. 30dim
46. rast: Rastrigin fv. 5dim
47. rast2d: Rastrigin fv. 2dim
48. rosen: Rosenbrock fv. 5dim
49. schw: Schwefel fv. 5dim
50. schw2d: Schwefel fv. 2dim
51. shekel: Shekel fv. 4dim
52. shub: Shubert fv. 2dim
53. sphere: Sphere fv. 3dim
54. sphere2d: Sphere fv. 2dim
55. sphere20d: Sphere fv. 20dim
56. sum2: Sum Squares fv. 8dim
57. sum22d: Sum Squares fv. 2dim
58. tesztfv5: két dimenziós tesztfüggvények futtatása
59. tesztfv6: Több dimenziós tesztfüggvények futtatása
60. trid: Trid fv. 6dim
61. trid2d: Trid fv. 2dim
62. trid20d: Trid fv. 20dim
63. zakh: Zakharov fv. 8dim
64. zakh2d: Zakharov fv. 2dim

A.2. Excel fájlok

1. ackley2d: Ackley fv. 2dim
2. banana: Rosenbrock fv. 2dim
3. beale: Beale fv. 2dim
4. bh1: Bohachevsky fv. (1) 2dim
5. bh2: Bohachevsky fv. (2) 2dim

6. bh3: Bohachevsky fv. (3) 2dim
7. booth: Booth fv. 2dim
8. branin: Branin fv. 2dim
9. colville: Colville fv. 4dim
10. dp: Dixon & Price fv. 6dim
11. dp2: Dixon & Price fv. 2dim
12. dp20d: Dixon & Price fv. 20dim
13. easom: Easom fv. 2dim
14. fr: Freudenstein & Roth fv. 2dim
15. gold: Goldstein &Price fv. 2dim
16. griewank: Griewank fv. 8dim
17. griewank2d: Griewank fv. 2dim
18. griewank20d: Griewank fv. 20dim
19. hart3: Hartmann fv H(3,4) 3dim
20. hart6: Hartmann fv H(4,6) 6dim
21. hump: Hump fv. 2dim
22. levy: Levy fv. 4dim
23. levy2d: Levy fv. 2dim
24. levy30d: Levy fv. 30dim
25. matyas: Matyas fv. 2dim
26. mich: Michalewics fv. 5dim
27. mich2d: Michalewics fv. 2dim
28. perm: Perm fv. P(n,b) 4dim
29. perm0: Perm fv. P(0,n,b) 4dim
30. perm02d: Perm fv. P(0,n,b) 2dim
31. perm2d: Perm fv. P(n,b) 2dim
32. perm020d: Perm fv. P(0,n,b) 20dim
33. powell: Powell fv. 8dim

34. powell24d: Powell fv. 24dim
35. powerr: Power fv. 4dim
36. powerr2d: Power fv. 2dim
37. qqq: (a.*x)*(a.*x)T fv. 2dim
38. qqq30d: (a.*x)*(a.*x)T fv. 30dim
39. rast: Rastrigin fv. 5dim
40. rast2d: Rastrigin fv. 2dim
41. rosen: Rosenbrock fv. 5dim
42. schw: Schwefel fv. 5dim
43. schw2d: Schwefel fv. 2dim
44. shekel: Shekel fv. 4dim
45. shub: Shubert fv. 2dim
46. sphere: Sphere fv. 3dim
47. sphere2d: Sphere fv. 2dim
48. sphere20d: Sphere fv. 20dim
49. sum2: Sum Squares fv. 8dim
50. sum22d: Sum Squares fv. 2dim
51. trid: Trid fv. 6dim
52. trid2d: Trid fv. 2dim
53. trid20d: Trid fv. 20dim
54. zakh: Zakharov fv. 8dim
55. zakh2d: Zakharov fv. 2dim

A.3. MO PowerPoint fájl

Tesztfeladatok : Tesztfüggvények

A.4. Tesztfeladatok elérési helyei

http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar
/Hedar_files/TestGO_files/Page364.htm

http://www.math.bme.hu/~bog/GlobOpt/globopt_hun.pdf

B. függelék

Irodalomjegyzék

[1] Brooks CH. An Introduction to Amoeba.
http://www.cs.usfca.edu/~brooks/papers/amoeba.pdf

[2] Fuchang Gao, Lixing Han. Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Computational Optimization and Applications* January 2012; Volume 51, Issue 1, pp 259-277

[3] Hooke R. and Jeeves TA. Direct Search Solution of Numerical and Statistical Problems. *Journal of the ACM* 1961; Vol. 8, pp. 212-229

[4] Kelley C.T. Detection and remediation of stagnation in the Nelder-Mead algorithm using a sufficient decrease condition. *SIAM J. Optim.* 2000; 10, 43–55

[5] Kuhn HW. And Tucker AW. Nonlinear Programming. *Proc. Second Berkeley Symp. on Math. Statist. and Prob. (Univ. of Calif. Press),* 1951; 481-492.

[6] Lagarias JC, Poonen B, Wright MH. Convergence of the restricted Nelder-Mead algorithm in two dimensions. *SIAM J. Optimization 22* 2012; 2, 501-532

[7] Lagarias JC, Reeds JA, Wright MH, and Wright PE. Convergence Properties of the Nelder-Mead simplex method in low dimensions. *SIAM Journal on Optimization* 1998; 9: 112-147.

[8] MacKnight M. Horch E. P. Calculating Visual Binary Star Orbits with the Downhill Simplex Algorithm (Amoeba). *American Astronomical Society Meeting 204, #07.19; Bulletin of the American Astronomical Society, Vol. 36, p.788*

[9] Maksat Ashyraliyev, Yves Fomekong-Nanfack, Jaap A. Kaandorp, Joke G. Blom. Systems biology: parameter estimation for biochemical models. FEBS J. 2009 Feb;276(4) :886-902. doi: 10.1111/j.1742-4658.2008.06844.x. Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands.

[10] Mckinnon KIM. Convergence of the nelder-Mead simplex method to a nonstationary point. *SIAM Journal on Optimization* 1998; 9: 148-158.

[11] Nelder JA and Mead R. A simplex method for function minimization. *Computer Journal* 1965; 7:308-313.

[12] Powell, M. J. D. An efficient method for finding the minimum of a function of se-

veral variables without calculating derivatives. *Computer Journal* 7(2) 1964; 155–162. doi:10.1093/comjnl/7.2.155

[13] Price CJ, Coope ID, and Byatt D. A convergent variant of the Nelder-Mead algorithm. Journal of Optimization Theory and Applications 2002; 113: 5-19.

[14] Rapcsák Tamás. Nemlineáris optimalizálás. 2007;
http://www.oplab.sztaki.hu/tanszek/download/nemlinopt.pdf

[15] Satoru Hiwa, Tomoyuki Hiroyasu, Mitsunori Miki. Downhill Simplex method. Report No. 20060806006 2006;
http://www.mis.doshisha.ac.jp/report/2006/9/25/20060806006/isreport20060806006.pdf

[16] Singer Sasa and Nelder JA. Nelder-Mead algorithm. *Scholarpedia, 4(7) :2928* 2009; doi:10.4249/scholarpedia.2928

[17] Singer, Sanja and Singer, Sasa (2001). Complexity Analysis of Nelder-Mead Search Iterations. in *Proceedings of the 1. Conference on Applied Mathematics and Computation, Dubrovnik, Croatia, ID1999, M1999*, M. Rogina, V. Hari, N. Limi, and Z. Tutek (Eds.), PMF–Matematiki odjel, Zagreb, pp. 185–196.

[18] Singer, Sasa and Singer, Sanja (2004). Efficient Implementation of the Nelder-Mead Search Algorithm. *Appl. Numer. Anal. Comput. Math.* **1**, No. 3, pp. 524–534.

[19] Spendley W. Hext GR. Himsworth FR. Sequential Application of Simplex Designs in Optimisation and Evolutionary Operation. *Technometrics* 1962; Vol. 4, No. 4 pp. 441-461, Section 3.1

[20] Tapas Kanungo and Qigong Zheng. A Downhill Simplex Algorithm for Estimating Morphological Degradation Model Parameters. 2001;
http://www.dtic.mil/dtic/tr/fulltext/u2/a458745.pdf

[21] William H. Press, Saul A. Teukolsky, William T. Vetterling, Brian P. Flannery. Numerical Recipes in C. *Cambridge University Press* 1992;

[22] Wright MH. Direct search methods: once scorned, now respectable. *In Numerical Analysis* Griffiths DF and Watson GA, ed.;, Addison Wesley Longman, Harlow, United Kingdom 1996;191-208.