Balogh János

GLOBÁLIS OPTIMALIZÁLÁSI ALKALMAZÁSOK ÉS SZEMI-ON-LINE LÁDAPAKOLÁS

Doktori értekezés

Témavezető: Dr. Csendes Tibor

Szegedi Tudományegyetem

Szeged, 2007.

Tartalomjegyzék

Bevezetés **1**

1. Globális optimalizálási problémák Stiefel–sokaságokon **4**
1.1. Bevezetés, előzmények 5
1.2. Néhány bevezető példa és megoldásaik – speciális esetekben 7
1.3. Numerikus eredmények 17
1.4. Tesztfüggvények megadása $M_{n,k}$-n 28
1.5. A lehetséges megoldások halmazának megszorítása – diszkretizálása 31
1.6. A fejezet összegzése 35

2. Az RPCRS párhuzamos globális optimalizálási módszer **37**
2.1. Bevezetés 37
2.2. Az RPCRS, mint a CRS algoritmus egy párhuzamos változata 39
2.3. Az RPCRS-t értékelő teszt egyprocesszoros környezetben 45
2.4. Az RPCRS párhuzamos környezetbeli numerikus értékelése 49
2.5. A fejezet összegzése 51

3. Fázisstabilitási problémák globális optimalizálása **53**
3.1. Bevezetés, előzmények 53
3.2. A célfüggvény és az alkalmazott módszer 56
3.3. A módszer tesztelése 57
3.4. A fejezet összegzése 61

4. Korlátok szemi-on-line ládapakolási feladatokhoz **63**
4.1. Bevezetés, előzmények és definíciók 63
4.2. Alsó korlátok 68
4.3. Alsó korlát konstrukciónk speciális esetei 82
4.4. Felső korlátok 88
4.5. A fejezet összegzése 100

Összefoglalás **102**

Irodalomjegyzék **108**

Bevezetés

A kutatási feladatokról és az alkalmazott módszerekről

A jelen dolgozat a szerzőnek a globális optimalizálás és alkalmazásai területén, és egy diszkrét optimalizálási feladaton elért néhány eredményét tartalmazza. A dolgozat négy fő részből áll: az 1., a 2., és a 3. fejezet témája a globális optimalizálás és alkalmazásainak körébe sorolható. A 4. fejezet egy diszkrét optimalizálási problémát tárgyal, alsó és felső korlátokat adva meg a ládapakolási feladat egy szemi-on-line variánsára. Ennek alsó korlátokat tárgyaló részében szintén használunk globális optimalizálási módszereket. Azaz, bár a folytonos és diszkrét optimalizálási problémák alapfeladata különböző, mégis, a kettő között létezik „átjárás". Az 1. és a 4. fejezetben egy folytonos, és egy diszkrét optimalizálási feladat vizsgálatánál is különböző – akár éppen a másik területről származó – módszereket vegyítünk, azaz jellemzően összefonódnak a különböző eszköztárak.

Az 1. fejezetben a statisztikában (faktoranalízis), a dinamikus rendszereknél (közgazdasági, tőzsdei idősorok) és a környezetvédelmi problémákban is számos alkalmazással bíró, az n-dimenziós egységhipergömb ortogonális vektorrendszerein, az ún. Stiefel–sokaságokon értelmezett nemlineáris optimalizálási feladatokat vizsgálunk. A bevezetés, a feladat definíciója és motivációja (1.1. alfejezet) után az 1.2. alfejezetben először egyszerű példákon keresztül, majd a sík egységkörén tanulmányozzuk a problémát. Ezután numerikus, globális optimalizálási szoftverekkel vizsgáljuk azt (1.3. alfejezet). A feladat nehézsége miatt érdemes ennek speciális eseteit is tekinteni. Ennek keretében adekvát globális optimalizálási tesztfeladatokat definiálunk (1.4. alfejezet), majd végül az általunk tárgyalt folytonos probléma egy diszkretizálását vetjük föl (1.5. alfejezet), és elemezzük.

A 2. fejezet egy gyakorlati alkalmazás: egy heurisztikus globális optimalizálási módszer párhuzamosíthatóságát vizsgáljuk. Megadjuk és numerikusan teszteljük a javasolt párhuzamos változatot. Globális optimalizálási módszerként az itt tárgyalt feladatunk a folytonos optimalizálás tárgykörébe tartozik.

A 3. fejezet témája szintén egy gyakorlati alkalmazás: egy sztochasztikus klaszterező globális optimalizálási módszer alkalmazása kémiai fázisstabilitási problémák megoldására, az irodalomból vett konkrét példákon végzett tesztekkel illusztrálva.

A 4. fejezetben a szemi-on-line ládapakolási feladatok területén elért eredményeinket tárgyaljuk. A kapcsolódó szakirodalombeli korábbi eredményeket javító alsó, valamint új felső korlátokat adunk meg. Az előkészületek után ennek első, az alsó korlátokat tárgyaló (4.2.) részében is nemlineáris optimalizálási problémához vezet a megadott modell vizsgálata. A diszkusszió lineáris algebrai módszerek használatát, majd egy szélsőértékfeladat megoldását, azaz analitikus eszközök alkalmazását igényli. Az optimum értékét a Lambert–féle W függvény segítségével fejezzük ki. A 4.3. alfejezetben a tárgyalt feladat speciális eseteit vizsgálva is használunk globális optimalizálási eszközöket. Végül, a 4.4. alfejezetben tisztán diszkrét optimalizálási módszerekkel adunk felső korlátokat ugyanezen problémára.

Az egész dolgozaton végigvonul a problémák (akár több eszköztárat igénylő) numerikus vizsgálata. Az egyik eszköztár éppen a különböző, globális optimalizálási módszerek, szoftverek alkalmazása – akár sztochasztikus, akár garantált megbízhatóságú intervallumos módszerekről legyen is szó. A nemlineáris optimalizálási eszközökön kívül és a fentebb említett technikák mellett felhasználunk más lineáris programozási (LP) eredményeket és technikákat is a dolgozat 1. és 4. fejezetében (az 1. fejezetben a feladat diszkretizálása során). A különböző területekről származó módszerek összeforrasztásával érdekes elemzéseket és eredményeket nyerünk.

A fejezetek alapját adó publikációkról

Az egyes fejezetek a szerző következő, legtöbbször társszerzőkkel közös publikációira épülnek: a Stiefel-sokaságok feletti optimalizálási problémákat vizsgáló 1. fejezet a [2, 6, 7, 11] cikkek, illetve előzetes változataik alapján készült, melyeket a megfelelő fázisokban különböző konferenciákon adtam elő. A globális optimalizálás két gyakorlati alkalmazását tárgyaló 2. és 3. fejezetből az előbbi, párhuzamos módszert tárgyaló rész a [80] munkára, míg a fázisstabilitási problémával foglalkozó 3. rész a [8, 9] cikkekre épül. A szemi-on-line ládapakolási feladatokat tárgyaló 4. fejezet alapját a [3, 4, 10] dolgozatok, illetve ezek előzetes verziói adták, utóbbiak többségükben konferenciaelőadások voltak.

Köszönetnyilvánítás

Itt szeretnék köszönetet mondani témavezetőmnek, Csendes Tibornak, aki azon túl, hogy irányította és egyengette munkámat, inspirált, és mindig a legjobb érdekeimet figyelembe véve adott tanácsokat. Példamutató hozzáállása, lelkesedése a tudomány művelése iránt valamennyi tanítványára, köztük rám is kihatott, amiért minden tanítványa részéről, így részemről is köszönet illeti. Munkája és ötletei nélkül a sikeres nemzetközi és hazai tudományos pályázatoknak, konferenciarészvételeknek csak töredékrésze valósulhatott volna meg.

Nagyon hálás vagyok Rapcsák Tamásnak, aki azon kívül, hogy javasolta és inspirálta az 1. részben tárgyalt téma kutatását, hosszú ideje nagy lelkesedéssel buzdított ezen disszertáció elkészítésére. Hozzáállásán, sok segítségén, határtalan lelkesedésén és emberségén kívül a vele való találkozások is mindig nagy lendületet adtak munkámnak.

Köszönöm Galambos Gábor támogatását, akitől nemcsak tudományos gondolkodásmódot tanultam, hanem biztos hátteret biztosított munkámhoz. Köszönet Békési Józsefnek, akinek egy húsz perces szemináriumi előadása során fölvetett problémák olyan kutatásokat indítottak meg, amelyekkel csaknem három éve foglalkozunk. Köszönettel tartozom kollegáimnak jóságukért és segítségükért, akik közül Csallner András és Mágoriné Huhn Ágnes szerzőtársam is.

Köszönettel tartozom külföldi tartozkodásaim kapcsán Inmaculada Garcíának (Almeríai Egyetem), Roumiana P. Statevának (Bolgár Tudományos Akadémia), Hans Kellerernek és Ulrich Pferschynek (Grazi Egyetem), Gerhard Reineltnek (Heidelbergi Egyetem) is, akik nemcsak külföldi útjaim lehetőségét és infrastruktúráját biztosították, hanem számos téma diszkutálására, előadására is lehetőséget adtak, és részt is vettek ezekben. Többjük szerzőtársam is. Szerzőtársaim közül külön köszönet illeti Markót Mihály Csabát, aki ötleteket és konkrét javaslatokat adott a disszertáció elkészítéséhez is.

Ezen kutatás elvégzéséhez segítséget nyújtott a jelenleg is folyó Országos Tudományos Kutatási Alapprogramok (OTKA T 048377 és T 046822 számú) valamint a MÖB-DAAD Magyar-Német Kutatócsere Program (21/05-ös számú) projektjeinek támogatása. Ezekért a szerző ezúton is szeretné kifejezni köszönetét, csakúgy, mint a már befejeződött Országos Tudományos Kutatási Alapprogramok (OTKA T 034350 projekt) 2001 és 2004, valamint az 5. Európai Uniós Keretprogram APPOL II. Tematikus Hálózati Projektjének (FP5, IST-2001-32007) és az OMFB Magyar-Spanyol Bilaterális SP-25/01 projektjének 2001 és 2003 közötti támogatásáért, továbbá a Tempus Közalapítványnak korábbi támogatásaiért is.

Végül, de legfőképpen szerető családomnak köszönök mindent. Tőlük az élet tudományát tanultam – tanulgatom. Mindezért szerény és csekély kis hála ezen munka is, melyet ajánlok

Balogh Martinnak, végtelen · végtelen szeretettel.

1. fejezet

Néhány globális optimalizálási probléma Stiefel–sokaságokon

A dolgozat ezen fejezetében a Stiefel–sokaságokon adott problémák globális optimalizálási feladatait tárgyaljuk a [2, 6, 7, 11] cikkekben publikált eredményeink alapján. Stiefel–sokaságok fölötti optimalizálást tárgyalt és elemzett Rapcsák néhány korábbi dolgozatában [88, 89, 90]. Az általa, valamint Bolla és szerzőtársai által [15] tanulmányozott feladatot vizsgáljuk.

A Stiefel–sokaságokon megadott globális optimalizálási problémát elméletileg és numerikusan, különböző globális optimalizálási módszerekkel elemezzük. A globális optimalizálási algoritmusok, szoftverek tesztelésére jól használható tesztfüggvényeket adunk meg. Megvizsgáljuk a probléma néhány speciális esetét, továbbá a lehetséges megoldások halmazának egy speciális megszorításával kapott feladatot is.

A fejezet a következő módon tagolódik: a bevezetés és a motiváció után először az optimumpontok struktúráját adjuk meg néhány speciális esetben (lásd még [2]) a [15, 88, 90]-ben és itt is tárgyalt kvadratikus feladatra, amiből néhány redukciós ötlet és eredmény adódik. Külön foglalkozunk azzal az esettel, amikor a célfüggvény együtthatómátrixai diagonálisak.

Ezután a [11] alapján megoldási módszereket és technikákat adunk meg a tárgyalt probléma numerikus optimalizálásához. A feladat numerikus szempontból történő vizsgálata különböző technikákat használó szoftverekkel készült. A számítógépes tesztfuttatások eredményeit az 1.3. alfejezetben foglaljuk össze. Motivációként, illetve a probléma numerikus nehézségeinek megvilágítása céljából elemezzük a numerikus vizsgálatok tapasztalatait.

Numerikus vizsgálataink eredményei, a tapasztalati nehézségek tanulságai szolgálhatnak motivációul is. Ez lesz a kiindulópontja annak, hogy az 1.4. alfejezetben a feladat ismert globális minimumhelyekkel és minimumértékekkel rendelkező speciális eseteit vizsgáljuk. A fejezet egyik fő eredménye, hogy ennek kiterjesztéseként tesztfüggvényeket adunk meg. A globális optimalizálásban nagy jelentőséggel bír, hogy adekvát tesztfeladatok álljanak rendelkezésre [31, 32]. Kiemeljük, hogy tesztproblémáink minden $M_{n,k}$ $(2 \leq k \leq n)$ Stiefel-sokaságon definiálhatóak.

A fejezet másik fő elméleti eredményeként a lehetséges megoldások halmazának megszorításával a folytonos feladat egy diszkretizálását vetjük fel és elemezzük (1.5. alfejezet). Megmutatjuk, hogy ez a jól ismert hozzárendelési feladattal ekvivalens problémát eredményez.

1.1. Bevezetés, előzmények

Stiefel 1935-ben vezette be azt a differenciálható sokaságot, amely az összes $\mathbf{x}_1$, $\mathbf{x}_2,\ldots,\mathbf{x}_k \in R^n$ ortonormált vektorrendszert tartalmazza, ahol R^n az n-dimenziós euklidészi tér [102]. A Stiefel-sokaságok gyakorlati problémákhoz kapcsolódnak. Tekintsünk többdimenziós idősorokból származó megfigyeléseket, feltételezve, hogy a dimenziószámhoz viszonyítva kevés tényező határozza meg a viselkedésüket. A statisztikai vizsgálatok egyik célja ezeknek a hatásoknak az azonosítása. A statisztika hagyományos eszköze erre a célra a faktoranalízis, ami többdimenziós idősorok esetén pontatlan vagy hibás eredményeket is adhat, különösen akkor, ha az idősorok komponensei között időtől függő összefüggés van. Ennek az az oka, hogy a faktoranalízist független megfigyelésekre dolgozták ki, és a függetlenség nem jellemző a többdimenziós idősorokra. Mindez azon új módszertan kidolgozását tette szükségessé, ami figyelembe veszi a megfigyelések dinamikáját is. A dinamikus faktoranalízis fogalmát Geweke [41], Bánkövi, Veliczky és Ziermann ([12, 13]) vezették be többdimenziós gazdasági idősorok vizsgálatára. Az azóta ebben a témában megjelent munkák a hagyományos faktoranalízis bizonyos tulajdonságainak általánosításait tartalmazzák.

Bánkövi, Veliczky és Ziermann [12, 13] megközelítését alapul véve Bolla, Michaletzky, Tusnády és Ziermann heterogén kvadratikus alakok maximalizálását vizsgálták Stiefel-sokaságokon, feltételezve, hogy a heterogén kvadratikus alakok pozitív definitek. A feladat struktúrális vizsgálata során megadták az elsőrendű és a másodrendű szükséges optimalitási feltételeket, valamint egy globális másodrendű szükséges feltételt, és globálisan konvergens, iterációs algoritmust dolgoztak ki kritikus pont meghatározására [15]. Eredményeiket elsősorban mátrixelméleti eszközök felhasználásával érték el. Rapcsák a fenti optimalizálási feladatot kvadratikus célfüggvényű és kvadratikus egyenlőségfeltételeket tartalmazó optimalizálási feladatként fogalmazta meg, amire a Riemann-geometria eszköztárát és ezen keresztül a globális Lagrange-multiplikátor-szabályt alkalmazva lokális és globális, első- és másodrendű, szükséges és elegendő optimalizálási feltételeket kapott, valamint klasszikus, nemlineáris optimalizálási módszerek erre a feladatra vonatkozó kiterjesztéseit [88, 89].

Márkus, Berke, Kovács és Urfer [71] a dinamikus faktoranalízis módszerének alkalmazását és a kapott eredményeket ismertetik többdimenziós környezeti idősorok esetén. Helmke és Moore könyvében [50] a dinamikus rendszerek strukturális stabilitásának vizsgálata a Rayleigh-hányados és az általánosított Rayleigh-hányados Stiefel-sokaságokon történő optimalizálására vezet. Edelman és társszerzői elektronikus rendszerek tervezését említik, mint a Stiefel-sokaságokon történő optimalizálás egyik lehetséges alkalmazását [28]. A statisztikai alkalmazások közül a főkomponens analízist és a statisztikai vizualizálást lehetne még megemlíteni [90]. Megjegyezzük, hogy gyakorlati problémák független változócsoportjainak modellezésében a Stiefel-sokaságokat alkotó ortogonális vektorrendszerek hatékony eszközt jelenthetnek.

Tekintsük a következő optimalizálási problémát:

$$\min \sum_{i=1}^{k} \mathbf{x}_i^T A_i \mathbf{x}_i \tag{1}$$

$$\mathbf{x}_i^T \mathbf{x}_j = \delta_{ij}, \qquad 1 \leq i,j \leq k, \tag{2}$$

$$\mathbf{x}_i \in \mathbb{R}^n, \quad i = 1,\ldots,k, \qquad n \geq 2,$$

ahol A_i, $i = 1,\ldots,k$, szimmetrikus mátrixok, és δ_{ij} a Kronecker-féle delta. A továbbiakban jelölje $M_{n,k}$ azt a halmazt, amely az n-dimenziós euklidészi tér összes k elemű ortonormált vektorrendszeréből áll.

Vezessük be a következő jelöléseket:

$$\begin{aligned} \mathbf{x} &= (\mathbf{x}_1^T, \mathbf{x}_2^T, \ldots, \mathbf{x}_k^T)^T \in \mathbb{R}^{kn}, \\ f(\mathbf{x}) &= \sum_{i=1}^{k} \mathbf{x}_i^T A_i \mathbf{x}_i. \end{aligned}$$

A Stiefel–sokaságok szerkezete a következő módon jellemezhető. A következő tételt James bizonyította először [60], de Rapcsák Tamás is adott rá egy szép bizonyítást [89] munkájában:

1. Tétel. *Az $M_{n,k}$ halmaz kompakt, $kn - \frac{k(k+1)}{2}$ dimenziós, végtelen sokszor differenciálható sokaság minden olyan pozitív egészekből álló (n,k) párra, amelyekre $k \leq n$ teljesül. A Stiefel–sokaságok a $k < n$ esetekben összefüggők, a $k = n$ esetben két összefüggő komponensből állnak.*

Az (1) probléma (2) korlátozó feltételei (nyilvánvalóan) úgy is felírhatóak, hogy

$$\begin{aligned} \boldsymbol{x}_i^T \boldsymbol{x}_i &= 1, \qquad i = 1,\ldots,k, & (3) \\ \boldsymbol{x}_i^T \boldsymbol{x}_j &= 0, \qquad i,j = 1,\ldots,k,\ i \neq j, & (4) \\ \boldsymbol{x}_i &\in \boldsymbol{R}^n, \quad i = 1,\ldots,k,\ n \geq 2. & \end{aligned}$$

Az ortogonalitás előírásából következően $n \geq k \geq 2$.

A vizsgálandó (1)-(2) optimalizálási probléma speciális típusú, kvadratikus egyenlőségfeltételekkel adott kvadratikus célfüggvényű feladat. Az optimalizálási irodalom szerint nehéz erre általános esetben jó közelítő megoldást adó hatékony módszert adni [53]. Elsőként a fenti, (1)-(2) típusú probléma egy partikuláris esetét tekintjük (a legkisebb méretű érdekes esetben), és azt vizsgáljuk, hogy milyen típusú elemi geometriai eszközök alkalmazhatók a feladat egyszerűsítésére. Ezt követően elemezhetők a választott numerikus eszközök és megvizsgálható, hogy ezek közül melyek alkalmazhatók ilyen típusú példákra, és melyek az általános, tetszőleges dimenziós esetekben.

1.2. Néhány bevezető példa és megoldásaik – speciális esetekben

Bevezetésként néhány egyszerű példát elemzünk. Megadjuk az optimumpontokat speciális diagonális együtthatómátrixú problémák esetén. (1) ezen speciális esetei gyakran előfordulnak valódi alkalmazásokban [89]. Ez segít a feladat szerkezetének feltárásában, másrészt lehetőséget ad néhány lehetséges egyszerűsítési, redukciós lehetőség bemutatására is. (Mint azt majd a későbbiekben látjuk, a nagy numerikus számításigény miatt ezek használata elengedhetetlennek látszik.)

1. Példa. *Tekintsük a következő feladatot, [89] egy feladatát:*

$$\min f(\mathbf{x}) = x_1^2 + \frac{1}{2}x_2^2 + x_3^2 + \frac{3}{2}x_4^2 \qquad , \tag{5a}$$

feltéve, hogy

$$\begin{aligned} &x_1^2 + x_2^2 = 1, \\ &x_3^2 + x_4^2 = 1, \\ &x_1x_3 + x_2x_4 = 0, \\ &\mathbf{x} = (x_1, x_2, x_3, x_4)^T \in \mathbb{R}^4. \end{aligned} \tag{5b}$$

Megmutatjuk, hogy $M_{2,2}$-n a feladat két fontos keresett jellemzője, a minimumhelyek és a maximumhelyek halmaza (a célfüggvény megfelelő függvényértékeivel együtt) explicit módon kiszámíthatók az (5a)-(5b) problémához.

Az egyenlőségfeltételekből következik, hogy

$$\begin{aligned} x_1^2 &= 1 - x_2^2, \\ x_3^2 &= 1 - x_4^2, \\ x_1^2x_3^2 &= x_2^2x_4^2, \end{aligned} \tag{6}$$

innen $x_1^2(1 - x_4^2) = (1 - x_1^2)x_4^2$, azaz,

$$x_1^2 = x_4^2 \quad \text{és} \quad x_2^2 = x_3^2. \tag{7}$$

Összevetve ezt az (5b) korlátozó feltételekkel azt nyerjük, hogy

$$\left.\begin{array}{c} x_1 = -x_4 \\ x_2 = x_3, \end{array}\right\}, \text{ vagy } \left\{\begin{array}{c} x_1 = x_4 \\ x_2 = -x_3. \end{array}\right. \tag{8}$$

Ily módon az $M_{2,2}$ Stiefel–sokaság két komponense a következő alakban áll elő:

$$\begin{aligned} M_{2,2} = \{\mathbf{x} \in \mathbb{R}^4 \mid x_1^2 + x_2^2 = 1,\ x_1 + x_4 = 0,\ x_2 - x_3 = 0\} \quad \cup \\ \{\mathbf{x} \in \mathbb{R}^4 \mid x_1^2 + x_2^2 = 1,\ x_1 - x_4 = 0,\ x_2 + x_3 = 0\}. \end{aligned}$$

Kihasználva a fenti egyenleteket, azt kapjuk, hogy a célfüggvény $M_{2,2}$-n a következő alakra hozható:

$$x_1^2 + \frac{1}{2}x_2^2 + x_3^2 + \frac{3}{2}x_4^2 = \frac{5}{2}x_1^2 + \frac{3}{2}x_2^2 = x_1^2 + \frac{3}{2}(x_1^2 + x_2^2) = x_1^2 + \frac{3}{2}.$$

A globális minimumhely – könnyen látható módon – az $x_1 = 0$ pont, és a globális minimum értéke $\frac{3}{2}$. Így az eredeti feladat globális minimumértéke szintén ennyi, minimumhelyeiként pedig a

$$(0, \pm 1, \pm 1, 0)^T$$

alakú $M_{2,2}$-beli pontok adódnak. Az $f(\mathbf{x})$ globális maximumhelyei is könnyen adódnak: ezek a

$$(\pm 1, 0, 0, \pm 1)^T$$

pontok, a globális maximum értéke pedig $\frac{5}{2}$. Megjegyezzük, hogy itt mindegyik esetben az előjelekre az összes eset lehetséges, nincs egyéb függőség, kapcsolat a megfelelő értékek között.

2. Példa. *Tekintsük a következő optimalizálási feladatot:*

$$\min f(\mathbf{x}) = 2x_1^2 + x_2^2 - 2x_3^2 + x_4^2, \quad \mathbf{x} \in M_{2,2}.$$

Kihasználva ismét a (6–7) során alkalmazott technikát, az f függvény

$$2x_1^2 + x_2^2 - 2x_3^2 + x_4^2 = 3x_1^2 - x_2^2$$

alakban adható meg $M_{2,2}$-n.

Az első korlátozó feltételt kihasználva (lásd (5b), 1. Példa) egyváltozós célfüggvényt kapunk. Egy másik lehetőség közvetlenül a $3x_1^2 - x_2^2$ függvényt tekinteni; az egységkörön ennek szóba jöhető legkisebb lehetséges értéke egyenlő -1-gyel. A célfüggvény ezt az alsó korlátot a $(0, \pm 1)^T$ pontokban veszi fel, és mivel (pl. (6)-ból láthatóan) a hozzájuk tartozó négy

$$(0, \pm 1, \pm 1, 0)^T$$

pont az eredeti négyváltozós feladat lehetséges megoldásait adja, így ezek globális minimumhelyek is, és ebből következően a minimum értéke $M_{2,2}$-n -1-gyel egyenlő. Könnyen belátható (például az utolsó, $3x_1^2 - x_2^2$ függvényalakból), hogy minden más pontban a függvényérték nagyobb, így az eredeti, négyváltozós feladat négy megoldásával annak összes megoldását megkaptuk.

Hasonlóan belátható az is, hogy az eredeti négyváltozós feladat globális maximumhelyei a

$$(\pm 1, 0, 0, \pm 1)^T$$

pontok, és csak azok, az ezekben felvett maximum értéke pedig 3.

Ennek kiterjesztéseként a diagonális A_i, $i = 1, \ldots, k$, együtthatómátrixok esete könnyen tárgyalható teljesen analóg módon, ekkor az optimumpontok minden koordinátája szintén a $\{-1, 0, +1\}$ halmazból kerül ki (kivéve azt az elfajuló esetet, amikor minden lehetséges megoldás egyben optimális megoldás is).

Az általános (1)-(2) feladat optimumpontjai szerkezetének jellemzése $M_{2,2}$-n szintén történhet elemi eszközökkel, például függvénydiszkusszióval; [11]-ben ennek segítségével kritériumot adtunk az optimumpontok számának végességére $M_{2,2}$-n. Ezt tárgyalja a most következő rész.

A probléma megoldása $M_{2,2}$-n

Ebben a szakaszban az (1) alakú, kvadratikus célfüggvényű globális optimalizálási feladat optimumpontjainak szerkezetét adjuk meg az $n = 2$, $k = 2$ esetben. Mivel ekkor az együtthatómátrixok tetszőlegesek, így vizsgálatunk vissza fogja adni az előző példákbeli azon speciális esetet, amikor a feladat együtthatómátrixai diagonálisak. Ennek során látni fogunk néhány redukciós lehetőséget is. (Ezek egy része nem más, mint az előző példák esetén is látott eliminációs lehetőségek általánosítása.)

$M_{2,2}$-n az (1) feladat a következőképpen írható fel általánosan: a minimalizálandó (vagy maximalizálandó) célfüggvény

$$ax_1^2 + bx_2^2 + cx_3^2 + dx_4^2 + ex_1x_2 + fx_3x_4, \tag{9a}$$

alakú, míg a korlátozó feltételek a következők:

$$\begin{aligned} x_1^2 + x_2^2 &= 1, \\ x_3^2 + x_4^2 &= 1, \\ x_1x_3 + x_2x_4 &= 0, \\ \mathbf{x} = (x_1, x_2, x_3, x_4)^T &\in \mathbb{R}^4. \end{aligned} \tag{9b}$$

A következő állítás segítségével (9) optimalizálási feladatát $M_{2,2}$-n egy vele ekvivalens egydimenziós, a $[0, 1]$ intervallumon definiált optimalizálási problémára redukáljuk, amelynek célfüggvénye viszont két, egyidejűleg optimalizálandó függvényt tartalmaz.

1. Lemma. *Tekintsük a*

$$\min\{Ax_1^2 + Bx_1\sqrt{1 - x_1^2} + C, Ax_1^2 - Bx_1\sqrt{1 - x_1^2} + C\}, \tag{10}$$

illetve a

$$\max\{Ax_1^2 + Bx_1\sqrt{1 - x_1^2} + C, Ax_1^2 - Bx_1\sqrt{1 - x_1^2} + C\}, \tag{11}$$

függvényeket ahol $x_1 \in [0, 1]$, $A = a + d - b - c$, $B = e - f$ és $C = b + c$ konstansok. (9a) minimalizálási illetve maximalizálási feladata $M_{2,2}$-n ekvivalens (10) minimalizálási illetve (11) maximalizálási feladatával a $[0, 1]$ intervallumon.

BIZONYÍTÁS. Az állítást két lépésben bizonyítjuk. Mindkét lépésben a (9b) korlátozó feltételeket használjuk. Először 2 korlátozó feltételt és 2 változót eliminálunk, majd a bizonyítás második lépésében még egy változót és az első lépés után maradt egyetlen korlátozó feltételt is.

1. lépés. Az 1. Példa megoldása során látott átalakításokat elvégezve, és a (8)-beli egyenleteket felhasználva a (9a) célfüggvény egyszerűsítésére, nyerjük a vele ekvivalens

$$(a + d)x_1^2 + (b + c)x_2^2 + (e - f)x_1x_2 \tag{12}$$

célfüggvényt, a korlátozó feltétel pedig (9b) első korlátozó feltétele marad; eliminálva ezzel két változót, és velük (9b) második és harmadik korlátozó feltételét.

A (12) alakja még tovább egyszerűsíthető x_2^2 kifejezésével, ez $1-x_1^2$ a (9b) első korlátozó feltételéből. Így az első lépés eredményeként $M_{2,2}$-n az általános, (9a-9b) probléma a következő, egyszerűbb formára hozható:

$$\begin{gathered} Ax_1^2 + Bx_1x_2 + C, \\ x_1^2 + x_2^2 = 1 \quad (x_1, x_2 \in \mathbb{R}), \end{gathered} \tag{13}$$

ahol $A = a + d - b - c$, $B = e - f$ és $C = b + c$.

2. lépés. A feladat (13) alakjából a megmaradt utolsó korlátozó feltételt is eliminálhatjuk, elvégezve az $x_2 = \pm\sqrt{1-x_1^2}$ helyettesítést. Ezzel megkapjuk az az állításban szereplő, $[0,1]$ intervallumon definiált függvényeket. Így a (9) optimalizálási feladatával ekvivalens, az állításbeli, egyváltozós, de két függvénnyel megadott optimalizálási feladatot nyerjük. □

Megjegyezzük, hogy nyilván hasonló kifejezéseket kaphattunk volna x_2-ben, x_3-ban és x_4-ben is, mint amilyen kifejezést kaptunk az 1. Lemmában x_1-ben, hasonló átalakításokkal.

1. MEGJEGYZÉS**.** *(9b) első két korlátozó feltétele (magasabb dimenzióban az első k korlátozó feltétel) közvetlenül használható minden „diagonális típusú" probléma esetén (azaz amelyeknél $\forall i$-re A_i diagonális mátrix). Ebben az esetben a célfüggvényben csak négyzetes tagok szerepelnek (lásd [2]), így k darab változó eliminálható a feladatból. Mint az majd a numerikus eredményekből kiviláglik, a korlátozó feltételek és/vagy a változók számának csökkenése a felmerülő számítási költségigényben jelentős csökkenést okozhat. Ezért fontos az ilyen jellegű redukciós lehetőségek kihasználása, ahol az lehetséges.*

2. MEGJEGYZÉS**.** *Az 1. Tételből következően $M_{2,2}$-nek – mivel a tétel szerint minden $M_{n,k}$-nak $n = k$ esetén – két komponense van. Az ezeken való optimalizálásnak felel meg az, ha a feladatot úgy oldjuk meg, hogy a célfüggvényben szereplő két függvényt különkülön optimalizáljuk. Érdemes megemlíteni, hogy az, hogy az alkalmazott egyszerűsítések, ekvivalens átalakítások után a feladatunk egyváltozós, az szintén megfelel az 1. Tétel eredményének, amelyből kiszámolható, hogy $M_{2,2}$ dimenziószáma 1.*

Az 1. Lemmából kaptuk az $M_{2,2}$-n megadott feladat immáron korlátozó feltételek nélküli alakját. Ennek megoldásában a probléma szimmetriája lesz segítségünkre: elegendő lesz a célfüggvényben szereplő két függvény egyikének szélsőértékhelyeit meghatározni a $[0,1]$ intervallumon, ennek során adódik a két függvénnyel megadott feladat megoldása is. Ebből szimmetria tulajdonságokat felhasználva adódik az eredeti feladat minden megoldása. Ehhez először az eredeti, $M_{2,2}$-n megadott (9) feladat egy olyan tulajdonságát mondjuk ki, amely ugyan (8)-ra épül, de megfogalmazása megkönnyíti az eredeti feladat megoldásainak automatikus meghatározását a (10) (maximalizálási feladat esetén a (11)) feladat megoldásaiból.

1. Tulajdonság. *Az általános, $M_{2,2}$-n megadott (9a) alakban felírt probléma szimmetrikus abban az értelemben, hogy ha $(x_1, x_2, -x_2, x_1) \in M_{2,2}$, akkor $(x_1, x_2, x_2, -x_1)^T$, $(-x_1, -x_2, x_2, -x_1)^T$, $(-x_1, -x_2, -x_2, x_1)^T \in M_{2,2}$, és*

$$\begin{aligned} f(x_1, x_2, -x_2, x_1) &= f(x_1, x_2, x_2, -x_1) = \\ f(-x_1, -x_2, x_2, -x_1) &= f(-x_1, -x_2, -x_2, x_1). \end{aligned}$$

BIZONYÍTÁS. (8)-ból megfigyelhető, hogy ha $(x_1, x_2, -x_2, x_1)^T \in M_{2,2}$, akkor az állításban szereplő pontok mindegyike $M_{2,2}$-beli. A függvény (13) alakjából látszik, hogy az ezekben a pontokban fölvett függvényértékek egyenlőek. □

Megjegyezzük, hogy a tulajdonság kimondásában szereplő pontok csak akkor alkotnak négy különböző pontot, ha egyik koordináta sem zérus, egyébként két különböző pontról van szó. Az 1. Tulajdonság nyilvánvaló következménye, hogy ha a (9) feladatnak ismert egy minimum/maximumhelye, és erre $x_1, x_2 \neq 0$, akkor az 1. Tulajdonság további 3 különböző minimum/maximumhelyet szolgáltat. $x_1 = 0$ vagy $x_2 = 0$ esetén további egyet. Ez azt is jelenti, hogy ha az 1. lemmabeli (10) feladatot (vagy maximalizálási feladatnál a (11) feladatot) az $x_1 \in [0, 1]$ intervallumon optimalizáljuk, akkor az 1. Tulajdonság alapján a (9) feladat összes megoldását megkapjuk.

1. Állítás. *Ha $A \neq 0$, vagy $B \neq 0$, akkor a* (9) *probléma optimumpontjait pontosan 4 minimumhely és pontosan 4 maximumhely alkotja. Az ettől eltérő esetben a feladatnak kontinuum sok megoldása van, mivel $M_{2,2}$ minden pontja minimum- és maximumpont is egyben.*

BIZONYÍTÁS. Három lépésben bizonyítjuk az állítást, megkülönböztetve az $A = 0$ és $B = 0$, az $A \neq 0$ és $B = 0$, valamint a $B \neq 0$ eseteket.

1. Lépés. Ha $A = 0$ és $B = 0$, akkor (10)-ből könnyen látható, hogy a célfüggvény konstans, így az adott Stiefel-sokaság minden pontja minimumhely és maximumhely is egyben.

2. Lépés. Ha $A \neq 0$ és $B = 0$, akkor problémánk együtthatómátrixa diagonális, és megoldásunk az 1. és 2. Példában látottakhoz hasonló lesz. (10) alakja ebben az esetben:

$$Ax_1^2 + C, \;\; x_1 \in [0, 1], \tag{14}$$

ahol $A = a + d - b - c$, $C = b + c$. (14) optimumpontjai nyilvánvalóan az $x_1 = 0$ és az $x_1 = 1$ pontok.
1. Ha $A > 0$ (azaz $a + d > b + c$), akkor a (14)-beli függvény, és így (9a) is akkor veszi föl $C(= b + c)$ minimumértékét, amikor $x_1 = 0$. Ebből következően a (9) feladat minimumhelyei pontosan a

$$(0, \pm 1, \pm 1, 0)^T$$

pontok, így ekkor (9)-nek pontosan 4 minimumhelye van. A (9a) maximumértéke egyenlő $a + d\ (= A + C)$-vel, és azt a

$$(\pm 1, 0, 0, \pm 1)^T$$

pontokban veszi föl.
2. Ha $A < 0$ (azaz $a + d < b + c$), akkor (9)-nek az $A > 0$ esetbeli minimumhelyei lesznek a maximumhelyei, és fordítva. A minimum értéke $A + C$-re vált, míg a maximumérték C-re.

3. Lépés. Tegyük fel, hogy $B \neq 0$. Jelöljük $g_1(x_1)$-gyel és $g_2(x_1)$-gyel a (10) (maximalizálási feladatnál (11)) célfüggvényében szereplő két függvényt, azaz

$$g_1(x_1) = Ax_1^2 + Bx_1\sqrt{1 - x_1^2} + C, \quad \text{és} \quad g_2(x_1) = Ax_1^2 - Bx_1\sqrt{1 - x_1^2} + C,$$

ahol $x_1 \in [0, 1]$ mindkét esetben. Helyettesítsük x_1-et $\sqrt{y}$-nal ($y = x_1^2$, $x_1 \geq 0$), és definiáljuk h_1-et és h_2-t y függvényében a $[0.1]$ intervallumon úgy, hogy

$$\begin{aligned} h_1(y) &:= g_1(x_1) = Ay + B\sqrt{y}\sqrt{1 - y} + C &&= Ay + B\sqrt{y - y^2} + C, \\ &&\text{és}& \\ h_2(y) &:= g_2(x_1) = Ay - B\sqrt{y}\sqrt{1 - y} + C &&= Ay - B\sqrt{y - y^2} + C. \end{aligned} \tag{15}$$

(10) minimalizálási illetve (11) maximalizálási feladata a $[0, 1]$ intervallumon nyilvánvalóan ekvivalens a hasonló, a $h_1(y)$ és $h_2(y)$ függvényekkel megadható minimalizálási illetve maximalizálási feladattal (azaz amelyekben a célfüggvény $\min\{h_1(y), h_2(y)\}$ illetve $\max\{h_1(y), h_2(y)\}$, $y \in [0, 1]$).

Az utóbbi megoldását megkapjuk, ha meghatározzuk h_1 és h_2 optimumhelyeit ezen az intervallumon. Ehhez azonosítanunk kell, hogy $h_1'(y)$ és $h_2'(y)$ hol zérus, valamint ellenőriznünk kell az $y = 0$ és az $y = 1$ pontokat is, ahol a deriváltfüggvények nem értelmezettek, ugyanis

$$h_1'(y) = A + B\frac{1 - 2y}{2\sqrt{y - y^2}} \quad \text{és} \quad h_2'(y) = A - B\frac{1 - 2y}{2\sqrt{y - y^2}}, \quad y \in (0, 1).$$

Ebből következően

$$h_1'(y) = 0 \text{ akkor és csak akkor, ha } -2A\sqrt{y - y^2} = B(1 - 2y). \tag{16}$$

Mindkét oldalt négyzetre emelve és az egyenletet átrendezve kapjuk, hogy

$$(4B^2 + 4A^2)y^2 - (4B^2 + 4A^2)y + B^2 = 0. \tag{17}$$

Mivel most $B \neq 0$, így a következő egyenlet adódik:

$$y^2 - y + \frac{B^2}{4(B^2 + A^2)} = 0,$$

melynek gyökei

$$y_{1,2} = \frac{1 \pm \sqrt{1 - \frac{B^2}{B^2+A^2}}}{2} = \frac{1}{2}\left(1 \pm \sqrt{\frac{A^2}{B^2+A^2}}\right).$$

Mivel a (16)-beli egyenlet mindkét oldalát négyzetre emelve kaptuk (17)-et, észrevehető, hogy a megoldás egy plusz, hamis gyököt szolgáltatna. De ugyanakkor az is észrevehető, hogy ezt nem kell elhagynunk, mivel a feladat most mindkét függvény szélsőértékhelyeinek meghatározása, és a másik gyök éppen $h_2'(y)$ gyöke. Ha $A \neq 0$, akkor $y_1 > \frac{1}{2}$ és $y_2 < \frac{1}{2}$; ebben az esetben $h_1'(y)$ és $h_2'(y)$ alakjából látszik, hogy ha A és B megegyező előjelűek, akkor y_1 a gyöke h_1'-nek és y_2 a gyöke h_2'-nek, míg ha ellenkező előjelűek, akkor fordítva. Az $A = 0$ esetben $y_1 = y_2 = \frac{1}{2}$ gyöke mindkettőnek.

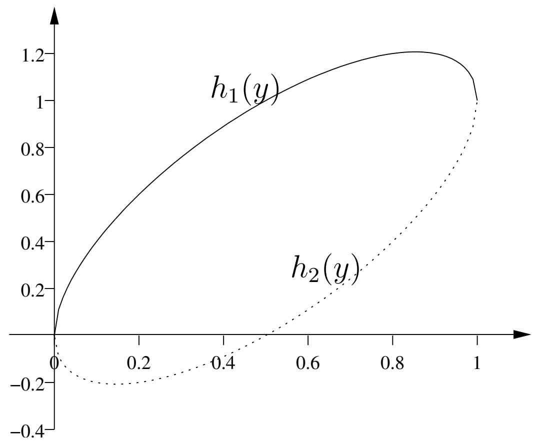
Deriválva a $h_1'(y)$ függvényt,

$$\begin{aligned} h_1''(y) &= (A + B\frac{1-2y}{2\sqrt{y-y^2}})' = B\frac{-4\sqrt{y-y^2} - \frac{(1-2y)^2}{\sqrt{y-y^2}}}{4(y-y^2)} = \\ &= \frac{-B}{4\sqrt{y-y^2}(y-y^2)} = -\frac{B}{4}\left(y-y^2\right)^{-\frac{3}{2}}, \ \ y \in (0,1), \end{aligned}$$

továbbá $h_2''(y) = -h_1''(y)$.

Következésképpen a h_1 és h_2 függvények kétszer folytonosan differenciálhatók a $(0,1)$ intervallumon. Ha $B > 0$, akkor a $(0,1)$ intervallumon a $h_1''(y)$ függvény negatív, azaz h_1 szigorúan konkáv, míg ha $B < 0$ akkor $h_1''(y)$ pozitív és így h_1 szigorúan konvex ott, és hasonlóan, h_2 szigorúan konvex, illetve szigorúan konkáv a $(0,1)$ intervallumon rendre, amikor B negatív, illetve pozitív. Ebből következően a fent kapott gyökeink egyike sem inflexiós pont a $(0,1)$ intervallumon. A fentiek alapján, $B > 0$ és $A \geq 0$ esetén y_1 a h_1 függvény maximumhelye, y_2 a h_2 minimumhelye a $(0,1)$ intervallumon ($A < 0$ esetén y_1 és y_2 szerepet cserél). $B < 0$ és $A \geq 0$ esetén y_2 a h_1 minimumhelye és y_1 a h_2 maximumhelye a $(0,1)$ intervallumon ($A < 0$ esetén y_1 és y_2 szerepet cserél).

Hátramaradt annak megmutatása, hogy a két gyök a két függvénnyel adott minimalizálási, illetve maximalizálási feladat egy-egy optimumpontját szolgáltatja, és nem csak a $(0,1)$ nyitott, hanem a $[0,1]$ zárt intervallumon is. Ennek igazolásához vegyük észre, hogy az $y = 0$ és az $y = 1$ pontokban a függvényértékek egyenlőek: $h_1(0)=h_2(0)$ és $h_1(1)=h_2(1)$ (például (15)-ből könnyű ezt látni). Az nyilvánvaló, hogy a 0 pontban jobbról, az 1 pontban balról folytonos mindkét függvény. Így a függvények görbéi ugyanabból a pontból indulnak ki, és ugyanabban a pontban végződnek. Továbbá, $h_1'(y)$ jobboldali határértéke a 0 pontban ∞ és baloldali határértéke az 1 pontban $-\infty$, míg $h_2'(y)$-é rendre $-\infty$ és ∞. (Ez a tulajdonság megfigyelhető az 1. ábrán is, az $A = 1$, $B = 1$, $C = 0$ eseteket ábrázoló példa képén.) Így az $y = 0$ pontnak van olyan jobboldali, az $y = 1$ pontnak pedig olyan baloldali, pozitív sugarú környezete, ahol az egyik függvény szigorúan monoton csökkenő, míg a másik szigorúan monoton növekvő. Ezért azt kapjuk,



1. ábra. A $h_1(y)$ és $h_2(y)$ függvények grafikonja az $A = 1$, $B = 1$ és $C = 0$ esetben.

hogy a $0, 1$ pontok egyike sem lehet optimumhely a fenti minimalizálási, illetve maximalizálási feladatra sem.

A fentieket összevetve azzal, hogy B előjelétől függően $h_1(y)$ és $h_2(y)$ közül az egyik szigorúan konvex, a másik szigorúan konkáv $(0, 1)$-en, mindebből következik, hogy az $y_{1,2} \in (0, 1)$ pontok – és csak azok – a globális optimumhelyek, A és B előjelétől függően a fent részletezett módon. A $g_1(x_1)$ és $g_2(x_1)$ $[0, 1]$-en való együttes optimalizálásának megoldásai visszahelyettesítéssel adódnak.

Végül az eredeti, $M_{2,2}$-n adott (9) feladat összes optimumpontját megkapjuk az 1. Tulajdonság felhasználásával. Mivel $x_1 \neq 0$ és $x_1 \neq 1$, így az $4 - 4$ különböző megoldást szolgáltat a minimalizálási és maximalizálási feladatra is. Ezektől különböző optimumhely pedig nincs. Ez indirekt úton könnyen bizonyítható az 1. Tulajdonság alapján. (Ha lenne a kapott 4 maximumhely vagy 4 minimumhely mindegyikétől különböző maximumhely vagy minimumhely, akkor az 1. Tulajdonság és a fentiek alapján egy ilyen pont egy új, $y \in [0, 1]$ intervallumba eső gyökét szolgáltatná az egydimenziós feladatnak is, amivel ellentmondásra jutnánk.) Ezzel állításunk bizonyított. □

Megjegyezzük, hogy a (16) összefüggésből minden esetben szemléletesen származtatható az optimumpontok struktúrája. Ezt illusztráljuk a különböző, A és B értékektől függő esetekben az ábrákon (amikor A vagy B nem zérus, akkor értéküket 1-re álítottuk be):

– Az $A \neq 0$, $B \neq 0$ esetet az 1. Állítás bizonyításában elemeztük. (Lásd a 2. ábra első grafikonját: egy egyenes és egy görbe metszéspontjai.)

– Az $A = 0$, $B \neq 0$ esetben két egyenes szakaszunk van, amelyek az $y = \frac{1}{2}$ pontban metszik egymást. (Lásd a 2. ábra második grafikonját.)

– Az $A \neq 0$, $B = 0$ esetben, a (16)-beli egyenlet jobboldalán a $B(1 - 2y)$ egyenes konstans zérus függvény és ezt metszi a görbe a 0 és 1 pontokban. (Lásd a 3. ábra első grafikonját).

2. ábra. Az $A = 1$, $B = 1$, illetve az $A = 0$, $B = 1$ esetek.

3. ábra. Az $A = 1$, $B = 0$, illetve az $A = 0$, $B = 0$ esetek.

– Az utolsó esetben, amikor $A = 0$ és $B = 0$, az egyenes szakasz és a görbe is megegyezik a konstans zérus függvénnyel a $[0, 1]$ intervallumon. Ezért a $[0, 1]$ intervallum minden pontja optimumpont. (3. ábra, második grafikon.)

A h_2 függvény esete teljesen hasonló: az egyik görbe helyett annak -1-szeresét, és a másik görbével való metszésponto(ka)t kell tekinteni. Ezután a bizonyításbeli helyettesítéseket elvégezve, figyelembe véve az 1. Tulajdonságból adódó multiplicitásokat is, a metszéspontokból a (9) összes optimumhelye meghatározható.

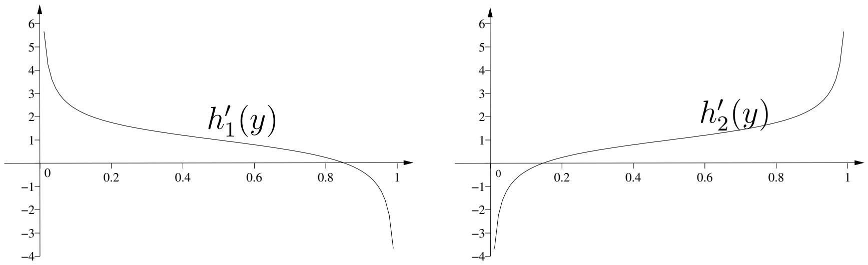
Nézzük meg ezt külön az első esetre, egy önmagában is szemléletes konkrét példán demonstrálva a megoldást ebben az esetben. Tekintsük azt a konkrét (10) és (11) alakú (minimalizálási, illetve maximalizálási) feladatot, amelyben $A = 1$, $B = 1$ és $C = 0$. (Megjegyezzük, hogy egy ilyen, A, B, C paraméterekkel megadott feladat valójában nyilvánvalóan számtalan, különböző a, b, c, d, e, f együtthatókkal $M_{2,2}$-n megadott (9a) célfüggvényű feladatból származhat, melyek mindegyike ugyanazon, A, B, C paraméterű (10), illetve (11) feladatra vezet az alkalmazott egyszerűsítések után.)

3. Példa. *Tekintsük a (10) és (11) feladat azon esetét, amikor* $A = 1$, $B = 1$ *és* $C = 0$. *(A feladat az optimalizálás a* $[0, 1]$ *intervallumon.)*

A fenti bizonyítás 3. lépését kell végrehajtani. A $h_1(y)$ és $h_2(y)$ függvények (15) alapján a következők lesznek:

$$h_1(y) = y + \sqrt{y}\sqrt{1-y} \;\text{ és }\; h_2(y) = y - \sqrt{y}\sqrt{1-y},$$

ahol $y \in [0, 1]$. Az 1. ábrán látható a h_1 és h_2 függvények grafikonja.



4. ábra. A deriváltfüggvények grafikonjai az $A = 1,\ B = 1$ beállítások esetén.

A deriváltfüggvények (lásd 4. ábra) a következők:

$$h'_1(y) = 1 + 1\frac{1-2y}{2\sqrt{y-y^2}} \quad \text{és } h'_2(y) = 1 - 1\frac{1-2y}{2\sqrt{y-y^2}},\ \ y \in (0,1),$$

Az optimumpontok

$$y_1 = \frac{1}{2} + \frac{\sqrt{2}}{4} = 0,85355... \text{ és } y_2 = \frac{1}{2} - \frac{\sqrt{2}}{4} = 0,14644...,$$

ahonnan

$$x_{1,1} = \sqrt{\frac{1}{2} + \frac{\sqrt{2}}{4}} = 0,92387... \text{ és } x_{1,2} = \sqrt{\frac{1}{2} - \frac{\sqrt{2}}{4}} = 0,38268....$$

Az 1. Állítás bizonyítása alapján az y_1, illetve az y_2 pont rendre a $h_1(y)$ és $h_2(y)$ függvények $[0,1]$ intervallumon való egyidejű maximalizálási, illetve minimalizálási feladatának megoldása, és nincs egyéb megoldás. A maximum értéke $h_1(y_1) = 1,20711$, míg a minimumé $h_2(y_2) = -0,207107$.

Az eredeti feladatnak $M_{2,2}$-n így pontosan 4 megoldása van a minimalizálási és maximalizálási feladat esetén is, ezek az 1. Tulajdonságból könnyen adódnak visszahelyettesítés után.

Összegezve, az 1. Állításban nyertük az $M_{2,2}$-n definiált (9) probléma megoldását. Láttuk ennek bizonyításának 1-3. eseteiben, hogy az $M_{2,2}$ Stiefel–sokaság két fontos komponense, a globális maximum- és minimumhelyek – és az optimumértékek – explicit módon számíthatók bármely (9) típusú problémához. Érdekes kérdés, hogy a megoldási technika alkalmazható vagy általánosítható-e adekvát módon nagyobb dimenziós feladatokra is.

A most következő numerikus vizsgálódásaink célja olyan tapasztalatok szerzése, amelyek nagyobb méretű, nagyobb n és k értékekhez tartozó feladatok megoldásában segíthetnek. A numerikus tesztek másik fő célja annak megvilágítása, hogy milyen numerikus technika hogyan alkalmazható a probléma megoldásában.

1.3. Numerikus eredmények

A probléma numerikus vizsgálatára két, különböző típusú globális optimalizálási eszközt választottunk. Egyrészt, egy hagyományos sztochasztikus globális optimalizálási módszert, Csendes Tibor GLOBAL nevű algoritmusát [23]. Másrészt, egy megbízható, intervallum-aritmetikát alkalmazó globális optimalizálási eszközt, a Baker Kearfott által tervezett és fejlesztett GlobSol [63] nevű optimalizálási módszert alkalmaztuk a numerikus vizsgálatok második részében. Ennek megbízható eljárására [63, 105] építve célunk garantált megbízhatóságú eredmények elérése volt. Ezen utóbbi vizsgálatok nehézségei és ezek okai egyben motivációt is jelenthetnek a fejezet további részeinek vizsgálataihoz, ezt ott röviden összefoglaljuk.

Empirikus numerikus elemzésünkben a számításigény mérésére valamennyi esetben megadjuk a vizsgált feladatok megoldásához szükséges függvénykiértékelések számát (beleértve a gradiens- és korlátozó feltétel kiértékelésekét is) és a felhasznált CPU-időt.

A numerikusan elsőként vizsgálandó problémánk legyen az $M_{2,2}$-n adott 1. Példa feladata, (nem-zérus) diagonális együtthatókkal:

1. Probléma. *Vizsgáljuk a következő, [89]-beli problémát:*

$$\min f(\mathbf{x}) = x_1^2 + \frac{1}{2}x_2^2 + x_3^2 + \frac{3}{2}x_4^2,$$

az

$$\begin{aligned} x_1^2+x_2^2 &= 1, \\ x_3^2+x_4^2 &= 1, \\ x_1x_3+x_2x_4 &= 0, \\ \mathbf{x} = (x_1, x_2, x_3, x_4)^T &\in \mathbb{R}^4 \end{aligned}$$

korlátozó feltételek mellett.

Az előző alfejezetben demonstráltuk, hogy az 1. Probléma egydimenziós problémává egyszerűsíthető $M_{2,2}$-n. Az eredeti, 1. problémabeli alak vizsgálata azonban numerikus szempontból is érdekes. Ebben az alfejezetben először erre adunk meg numerikus eredményeket.

Sztochasztikus globális optimalizálási módszer alkalmazásával nyert megoldások

A sztochasztikus globális optimalizálási módszerek közül a GLOBAL nevű nyílt forráskódú módszer alkalmazását választottuk. Ez egy összehasonlító kutatás [79] eredményeképpen a legjobbnak bizonyult a hasonló szoftverek között. A GLOBAL módszer Boender és társai globális optimalizálási módszerének [14] adaptált, továbbfejlesztett változata. Az algoritmus röviden a következőképpen írható le [23]:

1. lépés: Generáljunk egyenletes eloszlással M pontot az $S(= [0, 1]^n)$ értelmezési tartományban, és adjuk ezeket az aktuális C mintához. Konstruáljuk meg a D transzformált mintát, megtartva C pontjainak legalacsonyabb függvényértékű γ százalékát.

2. lépés: Alkalmazzuk a klaszterezési eljárást D-re. Ha D összes pontja valamely klaszterhez rendelhető, akkor ugorjunk a $4.$ lépésre.

3. lépés: Alkalmazzuk a helyi kereső eljárást T még klaszterezetlen pontjaira. Ismételjük a $3.$ lépést addig, amíg minden pont klaszterhez rendeltté válik. Ha találtunk egy új lokális minimumot, akkor ugorjunk az $1.$ lépésre.

4. lépés: Határozzuk meg a talált legkisebb lokális minimumértéket, és STOP.

A 2. lépésben alkalmazott klaszterezési eljárás leírása megtalálható a [23] közleményben. Ez az úgynevezett „single linkage" klaszterezési eljárás, amely a feladatunkra alkalmasnak bizonyult. Ennek célja azon mintapontok felismerése, melyekből a helyi keresőt elindítva a lehetséges eredmény egy már ismert lokális minimumhely lenne. A 3. lépésben két különböző helyi kereső eljárás használható az implementációban, vagy egy kvázi-Newton-eljárás a DFP (Davidon-Fletcher-Powell) update formulával [42], vagy egy véletlen sétát használó módszer, az UNIRANDI (lásd [61]).

Az itt alkalmazott módszer (bármelyik lokális kereső alternatívát alkalmazzuk is) egy úgynevezett direkt módszer, ami azt jelenti, hogy nem igényli a deriváltfüggvény explicit meghatározását. Sima célfüggvényekre a kvázi-Newton-módszer jó választásnak tűnik, míg olyan problémák esetén, ahol a függvény, vagy a deriváltfüggvény nem folytonos, a robusztus UNIRANDI véletlen kereső módszer ajánlható [22]. Ezért a lehetséges lokális kereső alternatívák közül mi a kvázi-Newton-típusú lokális keresőt használtuk.

A beállítható vezérlőparamétereket [22, 23] az adott feladatosztályra hangoltan oly módon választottuk, módosítva a javasolt „default" paramétereket, hogy minden iterációban 200 mintapontot generáltunk, ezek mindegyikéből lokális keresést indítva. Ez a választás [8, 9]-beli sikeres tapasztalatunkkal összhangban történt erre a feladatra is. Ott a kémiai fázisstabilitási probléma megoldására alkalmaztuk a módszert, hasonló beállításokkal. Ezt ebben a dolgozatban a 3. fejezetben tárgyaljuk majd, de a valós időrend fordított volt, ez a tapasztalat ekkor már rendelkezésre állt.

A módszer csak úgynevezett *„bound constrained"* problémáknál alkalmazható, azaz amelyekben csak intervallumos korlátozó feltételek szerepelnek (ez gyakorlatilag azt jelenti, hogy a korlátozó feltételeket csak a változókra adott alsó és felső korlátok jelentik). Ahhoz, hogy feladatunkból ilyen jellegű problémát állítsunk elő, az – az egyik leggyakrabban alkalmazott technikával – például büntetőfüggvény használatával lehetséges.

A büntetőfüggvény használatának célja a korlátozó feltételektől való „megszabadulás", amely azoknak a célfüggvénybe „olvasztásával" történik úgy, hogy az optimális megoldás ne változzék. A p büntetőfüggvény a célfüggvényben egy nemnegatív additív tagot jelent, azaz a célfüggvényből ily módon a származtatott optimalizálandó függvény ekkor $f + p$ ($p \geq 0$) alakú. p előírtan a korlátozó feltételek megsértésének valamely nemnegatív mértéke kell hogy legyen, azaz értéke pontosan azokban a pontokban zérus, ahol a korlátozó feltételek teljesülnek. Könnyen látható, hogy az $f + p$ minimumértéke megegyezik f-ével, és a minimumhelyek is ugyanazok mindkét függvény esetén, tehát a két optimalizálási feladat ekvivalens egymással. Esetünkben a $p(\mathbf{x}) = 2|x_1^2 + x_2^2 - 1| + 2|x_3^2 + x_4^2 - 1| + (x_1x_3 + x_2x_4)^2$, $\mathbf{x} \in \mathbb{R}^4$ büntetőfüggvényt alkalmaztuk.

A globális minimumértékek és a globális minimumhelyek, amelyeket ezzel a globális optimalizálási eszközzel találtunk ezen 4-dimenziós függvényre a következők:

	$g(\mathbf{x})$	$\mathbf{x}$
1.	$1,5001255$	$(\ 0,0059719,\ \ 0,9999686,\ \ 0,9999517, -0,0084435)^T$
2.	$1,5007071$	$(\ 0,0120176,\ \ 0,9996325, -0,9999300,\ \ 0,0116300)^T$
3.	$1,5003991$	$(\ 0,0029898, -0,9999863,\ \ 0,9998251, -0,0012770)^T$
4.	$1,5007045$	$(-0,0109279, -0,9999999, -0,9996446, -0,0146223)^T$.

f maximalizálásakor (ezt a gyakorlatban úgy végeztük, hogy $-f$-et minimalizáltuk):

	$g(\mathbf{x})$	$\mathbf{x}$
1.	$-2,4998629$	$(\ 0,9999981, -0,0030527,\ 0,0028565,\ \ 1,0001186)^T$
2.	$-2,4996702$	$(\ 1,0001611,\ \ 0,0009282,\ 0,0013075, -1,0000044)^T$
3.	$-2,4993594$	$(-1,0003144, -0,0014572,\ 0,0012710,\ \ 0,9999993)^T$
4.	$-2,4993250$	$(-1,0002619,\ \ 0,0030735,\ 0,0037596, -1,0000762)^T$.

A numerikus teszteket egy 500 MHz-es Pentium III-as PC-n (256 MB RAM, Linux operációs rendszer alatt) végeztük el. Az esetleges jobb összehasonlíthatóság céljából megadjuk az úgynevezett *STU*-t (*Standard Time Unit*), ez gépünkön $0,00047$ másodperc. Az STU a globális optimalizálási irodalomban szokásos módon használt viszonyítási alap, egyfajta mértékegység, amely a Shekel–5 függvény értékének a $(4,0;4,0;4,0;4,0;4,0)^T$ pontban $1\,000$-szer történő kiértékeléséhez szükséges CPU-idő. Ez lehetőséget teremt arra, hogy a különböző számítógépeken végzett tesztek CPU-idői összehasonlíthatók legyenek, csak az STU-khoz viszonyított relatív értékeket figyelembe véve.

A minimalizálási, illetve a maximalizálási feladat megoldásához az algoritmus összesen $88\,360$, illetve $78\,762$ függvénykiértékelést hajtott végre. A teljes végrehajtási idő mindkét esetben 1 másodperc alatt volt, $0,477$ illetve $0,420$ másodperc.

Az optimalizálási feladat azon megoldásai, amelyeket a GLOBAL algoritmus alkalmazásával nyertünk jó egyezést mutatnak az előző, 1.2. alfejezetben (5) optimalizálására nyert elméleti eredményeinkkel. Pontosabban fogalmazva, az optimumpontok közelítési pontossága elfogadhatóan kicsi, figyelembe véve a módszer sztochasztikus voltát.

Mindezen tapasztalataink azt mutatják, hogy kisebb megbízhatóság és kisebb pontosság rövidebb számítási időt és kevesebb műveletigényt jelent – ez a tény voltaképpen elvárt és indokolt. Amit itt levonhatunk következtetésül az az, hogy ennek a módszernek a használata egyszerű, nem igényli a derivált explicit megadását, továbbá gyorsnak, hatékonynak is bizonyult. Ellenben a módszer jellege miatt a szolgáltatott megoldások nem lehetnek megbízhatóak.

Megbízható, intervallum-matematikán alapuló módszerrel nyert numerikus eredmények

Az 1. Probléma egy 4-változós függvény minimalizálását jelenti, 3 korlátozó feltétellel. Tanulságos ezt először egy úgynevezett „black box" feladatként megoldani egy megbízható intervallumos globális optimalizálási módszer használatával.

Az intervallumos numerikus eszközök közül kevés képes kezelni bemenetként korlátozó feltételeket is, és ezek közül is csak néhány nemlineáris korlátozó feltételeket. Itt mi egy ilyen, intervallum-matematikán alapuló, Branch-and-Bound (B&B) technikát használó optimalizálási eszközt, a GlobSolt alkalmaztuk, amelyet Baker Kearfott tervezett és fejlesztett [63]. Ebben nem csak lineáris, hanem nemlineáris korlátozó feltételeket is előírhatunk, közvetlenül az input részeként. Ez egy speciális, intervallum-matematikán (lásd [48, 77, 91]) alapuló módszer. Az intervallum-matematika hasznos és hatékony eszköz folytonos függvények d-dimenziós zárt intervallumokon (hipertéglalapokon) felvett értékkészletének befoglalására. Az intervallum-matematikai szubrutinok különböző számítógéptípusokhoz, számítási környezetekhez is rendelkezésünkre állnak. Ezek a szubrutinok kifinomult, *irányított, kifelé kerekítő eljárások*at használnak, melyeknek köszönhetően az intervallumos műveletvégzés eredménye matematikai szigorral ellenőrzött, másképpen mondva garantált megbízhatóságú lesz lebegőpontos gépi aritmetikák alkalmazásakor is.

A Branch-and-Bound típusú globális optimalizálási módszerek (lásd pl. [53]) a függvénynek a keresési tartomány bizonyos részhalmazai fölött fölvett értékkészletének korlátait igénylik. A B&B módszerek egyes típusaira ezek a részhalmazok hipertéglalapok (más néven *box*ok). Ekkor az értékkészletek befoglalására intervallum-matematikai módszereket alkalmazhatunk.

Az 1. és a 2. táblázatok az ezzel az eszközzel végzett numerikus tesztek eredményeit tartalmazzák, f minimalizálási, illetve maximalizálási feladatára. (A maximalizálási feladat esetében $-f$-et minimalizáltuk. Ezért ekkor a ténylegesen kapott optimumérték $-5/2$ volt a táblázatban szereplő $5/2$ helyett.) A teszteket egy IBM RS/6000 típusú számítógépen hajtottuk végre, AIX 4.3 operációs rendszer alatt. A felhasznált CPU-idő összehasonlíthatósága céljából mért STU $0,06$ másodperc volt ezen a számítógépen.

Az előírt pontosságot 10^{-10}-re állítottuk be, azaz a lehetséges maximális beállítási pontosságra, amit a program képes kezelni. Helyhiány miatt itt csak a számítások költségigényére vonatkozó adatokat, illetve az optimumértékre a szoftver által talált legjobb közelítő értéket adjuk meg. Egyéb numerikus részletek a következő internetes címen találhatók:

`http://www.jgytf.u-szeged.hu/~balogh/Reszletes_adatok.ps.`

1. táblázat. Számítási költségigény az 1. Probléma *minimalizálásakor* a GlobSol program használatával. A megállási feltételként előírt pontossági követelmény 10^{-10} volt.

Függvénykiértékelések száma:	1 548
Gradiens kiértékelések száma:	718
Korlátozó feltétel kiértékelések száma:	2 927 955
Az optimumértékre nyert legjobb becslés:	1,50000029396437820
CPU idő (s):	429,63

2. táblázat. Számítási költségigény az 1. Probléma *maximalizálásakor* a GlobSol program használatával. A megállási feltételként előírt pontossági követelmény 10^{-10} volt.

Függvénykiértékelések száma:	4 300
Gradiens kiértékelések száma:	3 104
Korlátozó feltétel kiértékelések száma:	5 513 034
Az optimumértékre nyert legjobb becslés:	2,49999970603216810
CPU idő (s):	861,37

Az ott szereplő táblázatok tartalmazzák a minimumpontokra és a minimumértékre kapott befoglaló intervallumokat (X, $F(X)$). Ezek összefoglalásaként elmondhatjuk, hogy a numerikus tesztek eredményei pontosak voltak: az az X intervallumokban 5 jegyre, míg az $F(X)$-ekben 4 jegyre voltak pontosak. Pontosabban, az ismert megoldás minden koordinátáját tartalmazza egy legfeljebb $0,000122$ szélességű intervallum.

Ezen a ponton meg kell említenünk azonban, hogy bár a program által adott megoldások az előre ismert megoldásnak igen jó közelítését adják, és a megoldásként kapott intervallum minden esetben tartalmazza a közelítő megoldás egy befoglaló intervallumát, továbbá a program minden megoldást megtalált, mégis a következő üzenetet adta: „There were no boxes corresponding to verified feasible points." Ez azt jelenti, hogy a program nem tudott egy, a feltételeknek megfelelő megoldáspontot (azaz lehetséges megoldást) azonosítani a megadott intervallumban ("boxban"). Így a talált pont csak közelítő megoldásnak minősül. Ennek akár az is lehetett volna az oka, hogy a megkövetelt, beállított pontossági szint (10^{-10}) kisebb, mint amit kaptunk. Kérdés volt, hogy ha lazítjuk a megoldás pontosságára vonatkozó előírásunkat, akkor ennek árán a szoftver képes-e verifikált megoldásokat szolgáltatni. Azonban az előírt pontossági szint csökkentésével végzett teszteknél (10^{-8}, 10^{-5}, 10^{-4}, 10^{-3}, és 10^{-2} paramétereket beállítva megállási feltételként) sem kaptunk kielégítő eredményeket: a megoldások ekkor sem váltak ellenőrzötté.

Az 1. és 2. táblázatokban megfigyelhető, hogy a korlátozó feltétel kiértékelések száma az elsődleges, szinte kizárólagos okozója a teljes számítási költségnek (kivéve annak kb. egy ezrelékét). Ez azt mutatja, hogy a probléma numerikus megoldásának nehézségét leginkább a korlátozó feltételek okozzák. Ha számításba vesszük, hogy ez a feladat az egységkörön definiált – ami itt a lehetséges legkisebb dimenzió –, akkor magasabb dimenziós egységgömbökön tekintett problémáknál a számítások költsége a várakozásunk szerint elfogadhatatlan mértékben megnő. Ez felvet egy másik kérdést is: ezen korlátozó feltételek mellett a módszer alkalmazhatóságának kérdését. Ez motivál bennünket áttérni valamelyest kedvezőbb, könnyebben kezelhető alakra, kevesebb korlátozó feltétellel. Ezt elemezzük a következőkben.

Numerikus eredmények az 1. Probléma polárkoordinátás alakjára

Ebben a részben a feladat polárkoordinátás alakjára adunk meg numerikus eredményeket. A polár alak nagyobb méretű, nagyobb dimenziószámú feladatnál azt jelenti, hogy kevesebb változónk van (egy változóval, vektoronként). Ez egyszerűsítést jelent. A célfüggvényben viszont ezek szinusz és koszinusz függvényei jelennek meg, amelyek kiértékelése nagyobb számítási költséget jelent a célfüggvényérték vagy befoglalás meghatározásakor.

2. Probléma. *Az 1. Probléma polárkoordinátás transzformált alakja.*

$$\min\ \cos^2\alpha + \frac{1}{2}\sin^2\alpha + \cos^2\beta + \frac{3}{2}\sin^2\beta, \tag{18a}$$

a

$$\cos\alpha\cos\beta + \sin\alpha\sin\beta = 0,\ \ \alpha,\beta\in[0,2\pi] \tag{18b}$$

korlátozó feltételek mellett.

Ezt a formulát az 1. Problémából az $x_1 = \cos\alpha, x_2 = \sin\alpha, x_3 = \cos\beta, x_4 = \sin\beta$ helyettesítésekkel kaptuk. (18b)-ből következően $\cos(\alpha-\beta) = 0$, amely azt jelenti, hogy

$$\left.\begin{array}{r}\sin\alpha = \cos\beta,\\ \cos\alpha = -\sin\beta;\end{array}\right\}\ \text{vagy}\ \left\{\begin{array}{l}\sin\alpha = -\cos\beta,\\ \cos\alpha = \sin\beta.\end{array}\right. \tag{19}$$

Ezt kihasználva a (18a) függvény egyszerűsíthető, nyerve a

$$\min\ \sin^2\beta + \frac{3}{2},\ \ \beta\in[0,2\pi]$$

feladatot.

Az általános esetben (9a) úgy írható át, hogy

$$\min\ (a+d-b-c)\sin^2\beta + c + b + (e-f)\cos\alpha\cos\beta, \tag{20a}$$

és ekkor az egyetlen korlátozó feltétel

$$\cos(\alpha - \beta) = 0. \tag{20b}$$

(19) alapján (20b)-ből egy további változó elmininálható, azonban ezután két ágat kapunk ismét, azaz két, egyidejűleg optimalizálandó függvényt. Ebben az esetben (20) most úgy változik, hogy

$$\min\ (a + d - b - c)\sin^2\beta + c + b \pm (e - f)\sin\beta\cos\beta, \quad \beta \in [0, 2\pi],$$

amely ekvivalens a

$$\min\ (a + d - b - c)\sin^2\beta + c + b - |(e - f)\sin\beta\cos\beta|, \quad \beta \in [0, 2\pi]$$

feladattal. Ezáltal így itt egy egydimenziós függvényünk van, amely korlátozó feltételek nélkül minimalizálandó. (Csak intervallum-korlátunk van.)

A GlobSol használatával a 2. Problémára, azaz az 1. Probléma polárkoordinátás transzformáltjára kapott tapasztalati eredményeket mutatja a 3. és 4. táblázat, a minimalizálási illetve a maximalizálási feladatra. Az előállított megoldások mindkét esetben ellenőrzöttek, ugyanis az output tartalmazta a „List of boxes containing verified feasible points" üzenetet. A 3. és a 4. táblázatban az áttekinthetőség kedvéért csak a számítási igényre vonatkozó összefoglaló adatokat közöljük. A teljes, numerikus adatokat is tartalmazó output szintén elérhető a következő, fent is megadott internetes címen:

`http://www.jgytf.u-szeged.hu/~balogh/Reszletes_adatok.ps`.

3. táblázat. Számítási költségigény a 2. Probléma *minimalizálásakor* a GlobSol program használatával. A megállási feltételként előírt pontossági követelmény 10^{-10} volt.

Függvénykiértékelések száma:	2 003
Gradiens kiértékelések száma:	747
Korlátozó feltétel kiértékelések száma:	163 188
Az optimumértékre nyert legjobb becslés:	1,50000000000000220
CPU idő (s):	118,85

A polárkoordinátás alak használatának előnyei és hátrányai

A polár átalakítás előnye, hogy elvégzése után a változók száma k-val csökken, azaz ennél a konkrét feladatnál 2-vel. Ez kis n és k értékeknél kedvező. A jelen feladatnál a polárkoordinátás alakra áttérve a GlobSol szoftvercsomag a minimalizálási és a maximalizálási feladatra is jó pontosságú eredményeket, és ellenőrzött, matematikai szigorral is megbízható megoldásokat adott. Nagyméretű feladatoknál a polár transzformáció annyit jelent, hogy nk helyett $(n-1)k$ a változók száma,

4. táblázat. Számítási költségigény a 2. Probléma *maximalizálásakor* a GlobSol program használatával. A megállási feltételként előírt pontossági követelmény 10^{-10} volt.

Függvénykiértékelések száma:	2 269
Gradiens kiértékelések száma:	1 591
Korlátozó feltétel kiértékelések száma:	165 279
Az optimumértékre nyert legjobb becslés:	-2,49999999999999420
CPU idő (s):	116,82

viszont a polár átalakítás után ezek szinusz és koszinusz függvényei jelennek meg a célfüggvényben.

Mint a numerikus eszközök általában, a GlobSol program is kompakt halmaz megadását követeli meg keresési tartományként, ezért ennél a feladatnál is a $[0, 2\pi]$ intervallumot adtuk meg. Azonban így a megoldásokban szereplő zérus szögek kétszer reprezentáltak, úgy is mint 0, és úgy is mint 2π. Ez az oka annak, hogy a fenti internetes címen, a részletes adatoknál található táblázatokban hat minimumpont adatait tüntettük fel, holott tudjuk, hogy a feladatnak 4 megoldása van. Általános esetben is el kell végeznünk az ilyen multiplicitások kiszűrését, a megoldások azonosítását, hogy különböző vagy egyező megoldásokról van-e szó az eredeti x vektorok $[-1, 1]$ intervallumba eső koordinátáira való visszatérés után.

A feladat numerikus vizsgálata $M_{3,3}$-on konkrét példákkal és teszteredményeikkel

Tekintsük az (1) minimalizálási feladatot most $M_{3,3}$-on. Vizsgáljuk ezt két konkrét eseten keresztül, amelyekben az A_i, $i = 1, 2, 3$ együtthatómátrixok legyenek ismét diagonálisak:

3. Probléma. *Tekintsük a következő együtthatómátrixokkal adott (1) feladatot $M_{3,3}$-on:*

$$A_1 = diag(1, 1/2, -1/2), A_2 = diag(1, 3/2, 3), A_3 = diag(1, -5, -6).$$

4. Probléma. *Legyen ez a problémánk az*

$$A_1 = diag(1, 1/2, -1/2), A_2 = diag(1, -3/2, -3), A_3 = diag(1, -1, -1)$$

együtthatómátrixokkal megadott (1) alakú feladat $M_{3,3}$-on.

Kiírva a két példa célfüggvényeit (3 vektorral és ezek 9 koordinátájával, mint változókkal, bármely tag vagy változó eliminálása nélkül) ezek a következő alakúak:

$$f_1(\mathbf{x}) = x_{1,1}^2 + \frac{1}{2}x_{1,2}^2 - \frac{1}{2}x_{1,3}^2 + x_{2,1}^2 - \frac{3}{2}x_{2,2}^2 - 3x_{2,3}^2 + x_{3,1}^2 - 1x_{3,2}^2 - 1x_{3,3}^2,$$

$$f_2(\mathbf{x}) = x_{1,1}^2 + \frac{1}{2}x_{1,2}^2 - \frac{1}{2}x_{1,3}^2 + x_{2,1}^2 + \frac{3}{2}x_{2,2}^2 + 3x_{2,3}^2 + x_{3,1}^2 - 5x_{3,2}^2 - 6x_{3,3}^2.$$

A (3) korlátozó feltételek itt 3 – általános esetben $k(k-1)/2$ – egyenletet jelentenek:

$$x_{i,1}^2 + x_{i,2}^2 + x_{i,3}^2 = 1, \quad i = 1, \ldots, k, \tag{21}$$

míg a (4) korlátozó feltételek ($k(k-1)/2 =$) 3 *egyenlőségfeltételt* eredményeznek:

$$x_{1,1}x_{2,1} + x_{1,2}x_{2,2} + x_{1,3}x_{2,3} = 0, \tag{22a}$$

$$x_{1,1}x_{3,1} + x_{1,2}x_{3,2} + x_{1,3}x_{3,3} = 0, \tag{22b}$$

$$x_{2,1}x_{3,1} + x_{2,2}x_{3,2} + x_{2,3}x_{3,3} = 0. \tag{22c}$$

Analóg módon követve az előző alfejezetbeli érvelésünket adódik, hogy az első példa f_1 célfüggvényének globális minimumhelyei az $\mathbf{x}_1 = (\pm 1, 0, 0)^T$, $\mathbf{x}_2 = (0, 0, \pm 1)^T$ és $\mathbf{x}_3 = (0, \pm 1, 0)^T$ vektorok által ebben a sorrendben az összes lehetséges módon alkotott 8 különböző vektorrendszer – és egyéb megoldás nincs. A globális minimum értéke -3. A második függvény amiatt is bonyolultabb, mert az globális minimumértékét ($-5, 5$-et) 16 különböző pontban veszi fel.

Ezen probléma megoldására is a GlobSolt használtuk. A számítógépes környezet és a beállítások megegyeznek az eddig leírt GlobSol tesztek paramétereivel. A problémát egyszerűsítés végrehajtása nélkül, „black box" feladatként kezeltük. Ekkor a szoftver futása a 3. Problémán (az f_1 célfüggvénnyel és a (21)-(22) korlátozó feltételekkel) több mint egy hét futási idő után állt meg, de használható eredmény nélkül. A 4. Probléma numerikus eredményei azt jelzik, hogy ez a feladat nehezebb, mert ez 20 napig futott – bármilyen konkrét eredmény szolgáltatása nélkül.

Finomíthatjuk megközelítésünket, ha a célfüggvényekből elimináljuk az $x_{1,1}$, $x_{2,1}$ és $x_{3,1}$ változókat a (21) korlátozó feltételeket felhasználva. Ezt végrehajtva f_1 és f_2 új, az előzővel ekvivalens formája:

$$f_1(\mathbf{x}) = 3 - \frac{1}{2}x_{1,2}^2 - \frac{3}{2}x_{1,3}^2 - \frac{5}{2}x_{2,2}^2 - 4x_{2,3}^2 - 2x_{3,2}^2 - 2x_{3,3}^2, \tag{23}$$

$$f_2(\mathbf{x}) = 3 - \frac{1}{2}x_{1,2}^2 - \frac{3}{2}x_{1,3}^2 + \frac{1}{2}x_{2,2}^2 + 2x_{2,3}^2 - 6x_{3,2}^2 - 7x_{3,3}^2.$$

A korlátozó feltételeket nem módosítottuk, azok ugyanazok maradtak, mint fent.

Ez a kézenfekvő egyszerűsítési „trükk" nem működött a 4. Probléma esetén, mert egy hét futás után sem kaptunk semmiféle eredményt. Azonban a teszt a 3. Probléma f_1 alakú új célfüggvényével $3, 5$ nap futási idő után megoldásokat adott. Annak ellenére, hogy ezek nem ellenőrzöttek és ebből fakadóan nem voltak megbízhatóak, elmondható, hogy az eredmény 36 box tartalmazza a 8 megoldás mindegyikét. A 36 talált intervallum teljes listáját helyhiány miatt nem mellékeljük, csak azok összegző adatait soroljuk föl.

- A GlobSol által az optimum értékére adott 36 intervallum közül mindegyik tartalmazta a -3 optimumértéket, és ezen intervallumok mindegyikét pedig a $[-3, 0002000000000860,\ -2, 98800849999998520]$ intervallum – mint legszűkebb ilyen.

5. táblázat. Számítási eredmények a 3. Probléma (23) alakjára, a GlobSol használatával.

Függvénykiértékelések száma:	2 612 700
Gradiens kiértékelések száma:	12 430 812
Korlátozó feltétel kiértékelések száma:	109 597 416
Az optimumértékre nyert legjobb becslés:	-2,99999833797237470
CPU idő :	3,5 nap

- A 36 közelítő megoldásnál az ismert megoldások 0 koordináta értékeire kapott legrosszabb abszolút közelítés (ezek között vannak negatív és pozitív értékek is) a $-0,100000017449$, míg a megoldások ± 1-koordináta értékeire a $0,9994974918$ pont volt. Így a 36 közelítő megoldás mindegyikében minden „0-koordinátát" egy kisebb, mint $0,20000035$ szélességű, míg a ± 1-es értékű x_1, x_6 és x_8 koordináták mindegyikét egy legfeljebb $0,00100051$ méretű intervallum tartalmazza. Mivel az optimális megoldásban 6 zérus koordináta, és 3 darab ± 1-koordináta van, így a 8 optimumhely mindegyikét egy $6,41 \cdot 10^{-14}$ térfogatú hipertéglalap tartalmazza. Ezek térfogatainak összege $5,13 \cdot 10^{-13}$. Tehát az ismert optimumpontokat tartalmazó boxok a 36 output intervallumból nemcsak hogy mind tartalmazzák koordinátánként a megfelelő optimumpontok koordinátáit, hanem ezáltal az eredeti keresési tartomány mérete $1,0015 \cdot 10^{-15}$-szörösére csökkent. (Mivel az eredeti keresési hiperkocka, „box" mérete 2^9.)

- Az outputban az optimumértékek közelítésére kapott megoldáspontok mindegyikét a $[-2,99400099949735310, \ -2,99400412500000710]$ intervallum tartalmazta – mint szintén a legszűkebb ilyen. Ez azt jelenti, hogy (a korlátozó feltételeket ugyan nem biztos, hogy kielégítő), de az optimumpontok legjobb közelítéseként outputként kapott 36 pont mindegyikében fölvett célfüggvényérték az optimumérték $0,006$ pontosságú közelítését adta.

A szoftver számára a numerikus eredmények előállításához szükséges számítások költségigényének összegző adatatait tartalmazza az 5. táblázat. Ebből jól látható, hogy a számítási költségigény „hatalmas" volt, de annak is legnagyobb részét a korlátozó feltételek kiértékelésének nagy száma okozza (kb. $109,6$ millió ilyen típusú művelet volt ennél a feladatnál).

A numerikus eredmények értékelése és a redukciós ötletek néhány általánosítása

Az előzőekben megbízható megoldásokat célzó numerikus vizsgálataink során láttuk, hogy a legnagyobb számítási erőfeszítést a korlátozó feltételek teljesülésének ellenőrzéséhez szükséges kiértékelések okozzák. Ezek egy megbízható numerikus eszköz számára annak ellenőrzéséhez szükségesek, hogy egy adott intervallum egy pontja lehetséges megoldás-e, azaz kielégíti-e a korlátozó feltételeket. Az általunk a számításigény mérésére megadott korlátozófeltétel-kiértékelések száma több millió volt az $M_{3,3}$, vagy akár az $M_{2,2}$ feladatára is. A kapott intervallumok nem ellenőrzöttek, bár megvizsgálva őket kimutatható, hogy tartalmazzák az optimális megoldást. Érdemes megemlíteni, hogy a GlobSol [63] csak a polárkoordinátás helyettesítések után adott garantált megoldást. (Ekkor k változó eliminálható a feladatból, amely kisebb méretű feladatoknál számottevő.) Ilyen transzformáció nélkül egy $M_{3,3}$ fölött definiált hasonló problémán a program több napig futott – garantált megoldás elérése nélkül.

Ezen tények együtt azt indokolják, hogy ésszerű a numerikus eszközök megbízhatóvá és hatékonyabbá tételéhez a redukciós lépések használata. Összhangban ezzel a Gould–Toint szerzőpáros is ugyanerre a konkluzióra jutott, a Mathematical Programming szakfolyóiratbeli közleményükben [43], melyben kvadratikus feladatokat, pontosabban azok „előfeldolgozását" diszkutálták. Ahogyan ők is, mi is arra a következtetésre jutottunk, hogy a korlátozó feltételek ilyen szempontból történő vizsgálata indokolt és szükséges. Amennyiben lehetséges, a korlátozó feltételek és a változók számának csökkentése vagy egyéb egyszerűsítést okozó feltételek felhasználása célszerű az optimalizálási módszer hatékonyságának növelésére. Az előfeldolgozás fontossága a [43] tanulmányból is kiderül; ha az adott feladat kvadratikus korlátozó feltételekkel rendelkezik, ez a feladat jellegét kihasználva kell, hogy megtörténjen (mint ahogy azt [43] is hangsúlyozza).

Emiatt nagyobb méretű problémák esetén feladatunknál elkerülhetetlennek látszik valamilyen redukciós lehetőség megvizsgálása a numerikus eszközök alkalmazásának hatékonyabbá tételéhez. A lehetséges egyszerűsítésekhez feladatunknál célszerű az optimalizálási feladat ($M_{n,k}$) korlátozó feltételeiben és a célfüggvény szimmetriájában rejlő redukciós lehetőségek felhasználása. Ezt illusztrálva néhány lehetséges egyszerűsítést adtunk meg, amelyek az általánosítás lehetőségével bírnak.

A korlátozó feltételek esetén két lehetőség kínálkozik a feladat méretének csökkentésére: felhasználni a (4), és a (3)-beli korlátozó feltételeket. Ezzel csökkenthetjük a változók számát vagy/és a korlátozó feltételek számát is. A változók eliminálására adódó lehetőségek kihasználása azonban a – megmaradt vagy újonnan bevezetett – változóknak a korábbinál komplexebb kifejezését eredményezi. Ez történik akkor is, ha polárkoordinátás transzformációt használunk. Az ugyan nyilvánvaló, hogy polárkoordinátás transzformáció alkalmazása után a változók száma k-val csökken, azonban n dimenzióban egy-egy változó helyett különböző változók szinuszának és koszinuszának (többtényezős) szorzatai jelennek meg. Ez a gyakorlatban nehezebben kezelhetővé teszi a feladatot. A fenti ötletek alkalmazhatóak tetszőleges méretű Stiefel–sokaságokon, és diagonális vagy tetszőleges együtthatómátrixú feladatok esetén is.

A fentiekből következően a megoldáshoz szükséges számításigény csökkentése céljából létjogosultsága lehet más technikák alkalmazásának. Ezek helyett tekinthetünk más, numerikusan ekvivalens problémákat is, például büntetőfüggvényeket, illetve a redukciós lehetőségekben rejlő felgyorsítást felhasználva. Felmerülhet más, megbízható számítási eszköz alkalmazási lehetősége [21], illetve olyan módszereké, eszközöké, amelyek korlátozó feltételeket is képesek kezelni.

1.4. Tesztfüggvények megadása $M_{n,k}$-n

Az ebben az alfejezetben tárgyalandó tesztfüggvények definiálásához az előzőben tárgyalt numerikus tapasztalatok is motivációt jelentettek. Ahogy azt illusztráltuk, a tárgyalt optimalizálási probléma garantáltan megbízható numerikus megoldását tűzve ki célul a számítógépes műveletigény hatalmas. A megoldásként kapott értékek helyessége azonban nagyon nehezen ellenőrizhető, mivel ez a feladat ekvivalens az eredeti problémával.

Ezért szükséges olyan speciális problémák vizsgálata $M_{n,k}$-n az algoritmusok hatékonyságának és megbízhatóságának tesztelésére, amelyek esetén az optimum helyei és értéke ismertek. Ezek a tesztfeladatok azzal az előnnyel járnak, hogy könnyebben kezelhetők – ezen szempontból. Ilyeneket adtunk meg [6]-ban, de az előzőekben vizsgált partikuláris esetek feladatai is szolgálhatnak hasonló célokat.

A tesztfüggvények megadása fontos szerepet játszik a globális optimalizálásban [26, 31, 32]. A Floudas–Pardalos könyvben [31] megadott tesztproblémák között nem túl sok korlátozó feltételekkel rendelkező probléma található, és az ezen körbe sorolhatók is túlnyomórészt ipari alkalmazásokból származnak [53].

A következőkben a fejezet egyik fő eredményeként ilyen, az optimalizálási módszerek numerikus vizsgálatánál jól használható tesztproblémákat adunk meg tetszőleges $M_{n,k}$ Stiefel–sokaságon. Az általunk javasolt tesztfüggvények előzetes kézi számítással könnyen meghatározható minimumhelyekkel és minimumértékekkel rendelkeznek. Az így definiált feladatok megoldásainak koordinátái szintén a $\{-1, 0, +1\}$ halmazból valók.

Ezek megadásához először tekintsük a következő problémát $M_{3,3}$-n, amelyet azután majd általánosítunk. Ez egy, 8 darab ismert globális optimális megoldással rendelkező feladat.

4. Példa. *Tekintsük azt az (1)–(2) típusú minimalizálási problémát $M_{3,3}$-n, amely az*

$$A_1 = \mathrm{diag}(-1, 2, 3), \quad A_2 = \mathrm{diag}(4, -5, 6), \quad A_3 = \mathrm{diag}(7, 8, -9)$$

együtthatómátrixokkal definiált.

A (2) korlátozó feltételek azt is kifejezik, hogy minden x_i, $(i = 1, 2, 3)$ az egységgömb felületén helyezkedik el. Ebből egy alsó korlát nyerhető a célfüggvény értékére a következő módon:

$$f(\mathbf{x}) = (-1)x_{11}^2 + 2x_{12}^2 + \ldots + (-9)x_{33}^2 \geq (-1) \cdot 1 + 2 \cdot 0 + \ldots + (-9) \cdot 1 \geq -15,$$

minden $\mathbf{x} \in M_{3,3}$-ra. A becslésben kihasználtuk, hogy $0 \leq x_{ij}^2 \leq 1$ teljesül minden $1 \leq i, j \leq n$ esetén. Könnyen belátható az, hogy ez a célfüggvényérték felvevődik a

$$(\pm 1, 0, 0); \quad (0, \pm 1, 0); \quad (0, 0, \pm 1),$$

pontokban: ahol a ± 1 értékek minden lehetséges kombinációja globális minimumpontot ad. Az is könnyen látható, hogy $M_{3,3}$ bármely más, ezektől különböző pontjában a függvény felvett értéke nagyobb, mint a globális minimum értéke, -15. Azaz más, a fentiektől különböző globális minimumpont nincs.

Ez a módszer, az $M_{3,3}$-on tesztfüggvény generálására szolgáló eljárás tetszőleges n, k $(n \geq k \geq 2)$ értékekre általánosítható úgy, hogy az a kedvező tulajdonság érvényben maradjon, hogy a minimumhelyek és minimumértékek könnyen meghatározhatók. Az optimumpontok koordinátái ebben az esetben is a $\{-1, 0, +1\}$ halmaz elemeiből fognak állni (mint azt bizonyítani fogjuk). Bármely lehetséges n, k párra ezen problémák a következőképpen adhatók meg az $M_{n,k}$ Stiefel–sokaságon:

5. Példa. *Tekintsük az (1)–(2) problémát $M_{n,k}$-n, legyenek ebben*

$$\begin{aligned}
A_1 &= \operatorname{diag}(l_1, 2, 3, \ldots, n), \\
A_2 &= \operatorname{diag}(n+1, l_2, n+3, n+4, \ldots, 2n), \\
&\ldots \\
A_i &= \operatorname{diag}(n(i-1)+1, \ldots, n(i-1)+i-1, l_i, n(i-1)+i+1, \ldots, ni), \\
&\ldots \\
A_k &= \operatorname{diag}(n(k-1)+1, \ldots, nk-1, l_k)
\end{aligned} \tag{24}$$

diagonális együtthatómátrixok, és $l_i < 0, \ i = 1 \ldots k$.

A 4. Példa egy olyan speciális esete az 5. Példának, amikor $n = 3, \ k = 3$; $l_1 = -1, l_2 = -5$ és $l_k = l_3 = -9$. Egy másik választás lehet, amikor $l_i = -1$, $i = 1, \ldots, k$, és az összes többi értéket változatlanul hagyjuk. A minimális függvényérték ekkor is az l_i-k összege, azaz $-k$. A következő állítás azt mutatja, hogy a globális minimumhelyek ebben az esetben is megadhatók ahhoz hasonló módon, mint ahogy azt az 1. Példa esetén tettük. Vezessük be a következő jelöléseket.

Jelölje $\pm E_{n,k}$ az összes olyan k számú vektor n-esből álló ortonormált rendszert, ahol az i-dik vektor $(i = 1, \ldots, k)$ az i-dik egységvektor, vagy annak -1-szerese. Azaz, mindegyik $\pm E_{n,k}$ egy 2^k ortonormált vektorrendszerből álló halmaz ($k \geq 2$, $n \geq k$ tetszőleges). Ily módon mindegyik vektorrendszerbeli vektornak egyetlen nemzérus koordinátája van, az i-dik vektornak az i-dik $(i = 1, \ldots, k)$ koordinátája 1 vagy -1. Ha e_i jelöli az i-dik egységvektort, akkor

$$\pm E_{n,k} = \left\{ \ (\pm e_1, \ldots, \pm e_k) \ \mid \ \pm e_i \in \{e_i, -e_i\}, \ \ k \geq 2, \ n \geq k \ \right\}.$$

2. Állítás. *A (24) mátrixokkal és a (2) korlátozó feltételekkel megadott (1)–(2) típusú problémák globális minimumpontjainak halmaza pontosan $\pm E_{n,k}$.*

BIZONYÍTÁS. Elegendő azt bizonyítani, hogy egy $\mathbf{e}^* \in \pm E_{n,k}$ vektorrendszer lehetséges megoldás, továbbá, hogy bármely más $\mathbf{x} \notin \pm E_{n,k}$ lehetséges megoldás esetén a célfüggvény $\mathbf{x}$-ben felvett értéke nagyobb, azaz $f(\mathbf{x}) > f(\mathbf{e}^*)$.

A) Annak belátásához, hogy egy $\mathbf{e}^* \in \pm E_{n,k}$ vektor lehetséges megoldás, meg kell mutatni, hogy $M_{n,k}$-beli. Könnyen látható, hogy $\pm E_{n,k}$ minden vektorrendszere nyilvánvalóan az n-dimenziós egységgömbön helyezkedik el, és mindegyik ortogonális rendszer is egyben. Ezért $\pm E_{n,k}$ minden eleme kielégíti a (2)-beli egyenlőségfeltételeket, azaz a korlátozó feltételeket, és így lehetséges megoldás.

B) Mutassuk meg most azt, hogy bármely más lehetséges megoldás esetén a célfüggvény ott felvett értéke nagyobb, mint egy tetszőleges $\mathbf{e}^* \in \pm E_{n,k}$-ban felvett értéke. Tekintsünk egy ilyen lehetséges megoldást, egy $(\mathbf{y}_1, \ldots, \mathbf{y}_k) \in M_{n,k}$ vektorrendszert, amelyre $(\mathbf{y}_1, \ldots, \mathbf{y}_k) \notin \pm E_{n,k}$ teljesül. Osszuk fel két diszjunkt S és T halmazra az $\{1, \ldots, k\}$ indexhalmazt, $S \cup T = \{1, \ldots, k\}$, úgy, hogy S tartalmazza az összes olyan vektor indexét, amelyre $\mathbf{y}_s = \mathbf{e}_s$, vagy $\mathbf{y}_s = -\mathbf{e}_s$. T tartalmazza a többi indexet, ezért $\mathbf{y}_t \neq \pm\mathbf{e}_t$ teljesül bármely $t \in T$ esetén. A feltétel miatt $T \neq \emptyset$ kell, hogy teljesüljön. Ezért a célfüggvényérték a választott $\mathbf{y}$ pontban

$$\begin{aligned}\sum_{m\in\{1,\ldots,k\}} \mathbf{y}_m^T A_m \mathbf{y}_m &= \sum_{s\in S} \mathbf{y}_s^T A_s \mathbf{y}_s + \sum_{t\in T} \mathbf{y}_t^T A_t \mathbf{y}_t = \\ &= \sum_{s\in S} \mathbf{e}_s^T A_s \mathbf{e}_s + \sum_{t\in T} \mathbf{y}_t^T A_t \mathbf{y}_t\end{aligned}$$

alakban írható fel. Tekintve a különbséget az $\mathbf{e}^*$-ban felvett függvényértéktől:

$$\sum_{s\in S} \mathbf{e}_s^T A_s \mathbf{e}_s + \sum_{t\in T} \mathbf{y}_t^T A_t \mathbf{y}_t - \sum_{s\in S} \mathbf{e}_s^T A_s \mathbf{e}_s - \sum_{t\in T} \mathbf{e}_t^T A_t \mathbf{e}_t =$$

$$= \sum_{t\in T} \mathbf{y}_t^T A_t \mathbf{y}_t - \sum_{t\in T} \mathbf{e}_t^T A_t \mathbf{e}_t = \sum_{t\in T} \left(\mathbf{y}_t^T A_t \mathbf{y}_t - \mathbf{e}_t^T A_t \mathbf{e}_t\right) = \sum_{t\in T} \sum_{i=1}^{n} a_{ii}^t (y_{ti}^2 - e_{ti}^2) =$$

$$= \underbrace{\underbrace{\sum_{i=1,\ldots,n,} \underbrace{a_{ii}^i}_{<0} \underbrace{(y_{ii}^2 - 1)}_{\leq 0}}_{\geq 0} + \underbrace{\sum_{t\in T} \sum_{i=1,\ldots,n;\, i\neq t} \underbrace{a_{ii}^t y_{ti}^2}_{\geq 0}}_{\geq 0}}_{\geq 0} \geq 0.$$

Feltevéseink miatt azonban legalább egy kifejezés a fenti összegben zérustól különböző, azaz az pozitív. Az is látható, hogy a célfüggvény $\pm E_{n,k}$ minden vektorrendszerében azonos értéket vesz fel, így az a minimum értéke. Ezzel a B) rész, és így az egész állítás bizonyított. $\square$

Néhány technikai megjegyzés

Korlátozó feltételekkel ellátott globális optimalizálási tesztproblémák egy végtelen sorozatát állítottuk elő. Ezek közül mindegyik egy, az $M_{n,k}$ Stiefel–sokaságon definiált $(n \geq k \geq 2)$ és mindegyiknek 2^k megoldása van, azaz a globális minimumpontok száma k-ban exponenciális. A globális minimum értéke is könnyen, az A_i $(i = 1, \ldots, k)$ negatív együtthatóinak összegeként adódik. Ezen tulajdonság előnyös bizonyos optimalizáló algoritmusok tesztelésénél, főleg például azok esetén, amelyek használják az optimum értékét [16, 17, 70].

Ezen tesztproblémákra a nagyszámú globális (és lokális) minimum által okozott nehézség csökkenthető, ha megfelelően választott $x_{ij} \cdot x_{ik}$ kifejezéseket adunk hozzá a célfüggvényhez. Kivonva például minden $x_{ij}x_{il}$-t, $i = 1, \ldots, k; j, l = 1, \ldots, n; j \neq l$ a célfüggvényből csak kettő megoldásunk marad és ezeknél az összes nemzérus koordináta vagy $+1$, vagy -1. (Méghozzá dimenziónként egy koordináta ilyen, és az összes többi koordináta értéke 0.) Sajnos, a kapott probléma már nem lesz diagonális típusú ezután.

Ahhoz pedig, hogy maximalizálási feladatot kapjunk, $-A_i$ mátrixokat kell tekinteni a feladatnál A_i helyett, minden i indexre.

6. Példa. *Tekintsük azt az (1)–(2) problémát, amelyben*

$$\begin{aligned} A_1 &= \operatorname{diag}(-1, 2), \\ A_2 &= \operatorname{diag}(3, -1). \end{aligned}$$

A célfüggvény ekkor

$$f(\mathbf{x}) = -x_{11}^2 + 2x_{12}^2 + 3x_{21}^2 - x_{22}^2$$

lesz, és a megoldáspontok halmaza

$$\mathbf{x}^* = ((\pm 1, 0)^T, (0, \pm 1)^T).$$

Azaz ez egy $2^2 = 4$ pontot tartalmazó halmazzal adható meg. Az optimum értéke a diagonálisban lévő negatív értékek összege, azaz -2.

1.5. A lehetséges megoldások halmazának megszorítása – diszkretizálása

Ebben a fejezetben $M_{n,k}$-n a lehetséges megoldások halmazának egy megszorítását – pontjainak egy speciális megszorításán keresztül – vetjük fel. A (2) feltételek speciális diszkretizálása kvadratikus célfüggvény esetén NP-nehéz probléma, amiből kvadratikus hozzárendelési probléma [94] adódik.

Az előző alfejezetekben példáinkban több helyütt az optimális megoldások minden vektora egy koordinátatengelyen helyezkedett el (azaz egy koordinátájuk 1 vagy -1, és a többi $n-1$ darab koordináta zérus volt). Mint azt az előzőekben beláttuk, $M_{2,2}$-n az optimális megoldások típusa is ilyen, ha az együtthatómátrixok diagonálisak (kivéve azt az elfajuló esetet, amikor a függvény konstans az egész $M_{2,2}$-n). Ilyen esetekben és az általunk konstruált tesztproblémáknál is minden megoldás az n-dimenziós egységgömb és a koordinátatengelyek metszéspontjainak halmazából kerül ki. Másként fogalmazva, nemcsak az adott teszt problémák rendelkeznek ilyen típusú megoldásokkal, de mások is ebbe az osztályba tartozhatnak. Az általunk tárgyalt feladatok azzal a közös tulajdonsággal bírnak, hogy a célfüggvény csak négyzetes tagokat tartalmaz.

Ezen tény szolgált motivációul ahhoz, hogy megszorítsuk a probléma lehetséges megoldásainak halmazát. A lehetséges megoldások halmaza legyen $M_{n,k}$ egy olyan részhalmaza, ahol a lehetséges megoldások pontjaira még azt a feltételt is előírjuk, hogy az n-dimenziós koordinátengelyeknek is pontjai legyenek (mind a k vektor). Az így kapott probléma vizsgálatát – ahol továbbra is az (1) célfüggvény szerepel, de már $M_{n,k}$-nak ezen, a továbbiakban L'-vel jelölt megszorításán definiálva – három lépésben fogjuk elvégezni. Elsőként olyan $n \times n$-es problémákat tekintünk, amelyek esetén a célfüggvényben csak négyzetes tagok szerepelnek (ez a diagonális mátrixok esete). Ennek megoldása után a hasonló típusú, de ezúttal már $n \times k$ méretű problémák esetét vizsgáljuk meg, és végül az általános feladatot. Mindegyik tárgyalt esetben a lehetséges megoldások halmaza a fönti módon megszorított. Amikor a célfüggvény nem (1) típusú, akkor a megoldáshalmaz szerkezete nyitott kérdés még abban az esetben is, ha a lehetséges megoldásokat megszorítjuk az L'-vel jelölt halmazra.

Mire jó egy ilyen megszorítás? Egyrészt, felhasználható az f^* optimumérték közelítésére, másrészt, ezen közelítés alkalmazható néhány optimalizálási eszköz konvergenciájának gyorsítására is (lásd, pl. [16, 17, 70]).

Térjünk vissza a fönt említett három lépéses eljárásra. (2) lehetséges megoldásainak fönti, L' megszorítása a hozzárendelési feladatra vezeti vissza a problémát.

3. Állítás. *Ha az (1)-(2) problémában $M_{n,k}$ lehetséges pontjainak halmazát megszorítjuk arra az L' halmazra, amely az n-dimenziós hipergömb koordinátatengelyekkel vett metszéspontjainak halmaza, akkor az (1) és az így L'-re megszorított (2) által megadott feladat ekvivalens a hozzárendelési problémával.*

BIZONYÍTÁS. A bizonyítást három lépésben végezzük. Elsőként azt a speciális esetet vizsgáljuk, amikor az $M_{n,n}$ sokaságon $(k = n)$ vagyunk, és az (1) célfüggvény csak diagonális A_i $(i = 1, \ldots, k = n)$ mátrixokat tartalmaz. Ezután felhasználva az első állítást, az könnyen általánosítható $M_{n,k}$-ra (ahol immár n és k tetszőleges), még mindig a diagonális együtthatómátrixok esetére. A harmadik lépésben az első és második állítást kihasználva befejezzük a bizonyítást. Ez a bármely $M_{n,k}$ fölötti tetszőleges alakú (1)-beli mátrixok esete.

1. Az (1)-(2) probléma $M_{n,n}$-en ekvivalens a hozzárendelési feladattal, ha az együtthatómátrixok diagonálisak, és a lehetséges megoldások L' halmaza az n-dimenziós hipergömb és R^n koordinátatengelyeivel vett metszéspontjainak halmaza.

A jól ismert hozzárendelési feladat a következőképpen adható meg:

$$\min \sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} a_{ij}, \tag{25}$$

$$\sum_{t=1}^{n} x_{it} = 1 \quad (i = 1, \ldots, n), \tag{26a}$$

$$\sum_{t=1}^{n} x_{tj} = 1 \quad (j = 1, \ldots, n), \tag{26b}$$

$$x_{ij} \in \{0, 1\} \quad (i = 1, \ldots, n;\ j = 1, \ldots, n), \tag{26c}$$

ahol az a_{ij} együtthatók egy $n \times n$-es A' mátrix elemei. Azt kell megmutatnunk, hogy az (1) célfüggvény a (2) korlátozó feltételek mellett (25-26)-ot eredményezi ezen új, megszorított L' halmazon.

Először is megjegyezzük, hogy minden x_i pontosan $n-1$ darab zérus komponenssel rendelkezik, és pontosan egy eleme egyenlő 1-gyel, vagy -1-gyel.

Feleltessük meg a hozzárendelési feladat A' mátrixának a_{ij} elemeit az (1) $(A_i)_{jj}$ elemeinek, továbbá a hozzárendelési feladat x_{ij} változóinak az (1)-beli $\mathbf{x}_i$ j-dik elemét. Ezen feltételek mellett a hozzárendelési feladat (26) feltételei egybeesnek az eredeti probléma (2) korlátozó feltételeivel, míg a (25) célfüggvény pontosan az eredeti (1) célfüggvénnyel egyezik meg.

2. A fönti állítás kiterjeszthető az $M_{n,k}$ Stiefel–sokaságra is: az (1)-(2) probléma $M_{n,k}$-n az előző megszorítás mellett ekvivalens egy hozzárendelési feladattal.

Ennek megmutatásához az eredeti, $k \times n$ méretű probléma helyett tekintsünk egy olyan hozzárendelési problémát, amelyben a mátrix mérete $n \times n$, úgy, hogy töltsük ki a hiányzó $n-k$ sort $M+1$ értékekkel, ahol M az első k oszlop elemeinek abszolútérték-összege.

Megoldva az ily módon nyert hozzárendelési feladatot a „magyar módszer" alkalmazásával [66], és a megoldásból elhagyva az utolsó $n-k$ vektort, a megmaradó k darab vektor-n-esből álló rendszer pontosan az eredeti probléma megoldása lesz, amint az könnyen látható.

Térjünk vissza a célfüggvény szempontjából az általános esetre: tekintsük az általános (1)-(2) problémát, megszorítva $M_{n,k}$-t a koordinátatengelyekkel vett metszéspontjaira.

3. Elegendő azt megmutatni, hogy az utóbbi probléma – azaz amikor az (1)-(2) feladatot vizsgáljuk, ahol (1) immáron tetszőleges alakú, (2) pedig L'-re megszorított – ekvivalens egy olyan problémával, ahol a célfüggvény csak $x_i x_i$ kifejezéseket tartalmaz, amelyek együtthatója $(A_i)_{ii}$, minden i-re. Ezen második feladat ugyanis, mint azt fönt már beláttuk, ekvivalens a hozzárendelési feladattal, így ezzel az egész állítás belátható.

Vizsgáljuk meg ehhez (1)-et az L' részhalmazon. L' minden vektora speciális tulajdonságú, nevezetesen egy kivételével minden koordinátája 0. Ezért, az (1) célfüggvényben minden $x_{ij}x_{jl}$ kifejezés együtthatója zérus, kivéve az x_{ii}^2 típusú kifejezéseket.

Formálisan:

$$\sum_{i=1}^{k} \mathbf{x}_i^T A_i \mathbf{x}_i = \sum_{i=1}^{k} (\sum_{j=1}^{n} \sum_{l=1}^{n} x_{ij}(A_i)_{jl} x_{il}) = \sum_{i=1}^{k} x_{ii}(A_i)_{ii} x_{ii} = \sum_{i=1}^{k} x_{ii}^2 (A_i)_{ii}. \tag{27}$$

Ebből következik, hogy a nyert probléma ekvivalens a hozzárendelési feladattal, ahogy azt állítottuk. □

Visszatérve az általános esetre, felvethető a kérdés: hogyan oldható meg egy általános feladat (ahol a célfüggvény nem feltétlenül az (1) alakban adott) $M_{n,k}$ fenti L' részhalmazára megszorítva?

Ebben a tetszőleges célfüggvényű esetben, ha $n = k$, és a feladat $0 - 1$ értékű megszorítását tekintjük a $\{0, \pm 1\}$ megszorítás helyett, akkor nyilván szintén egy hozzárendelési feladat feltételrendszerét kapjuk, azaz egy olyan feladatot, amelynek feltételrendszere a hozzárendelési feladatéval egyezik meg. Az optimalizálási feladatunk ekkor: keresendő a célfüggvény minimuma ezen speciális $0 - 1$ (26) feltételek mellett. A hozzárendelési feladathoz hasonlóan a lehetséges megoldások halmaza megadható az $\{1 \ldots n\}$ halmaz egy permutációjával is. (Ez mátrix alakban úgy is megkapható, hogy 1-eseket írunk a megfelelő helyekre, míg az összes többi helyre 0-kat.) Ha $n \neq k$, akkor a feladatnak – akár lineáris célfüggvényt tekintve is – több speciális esete, alkalmazása van, például a szállítási feladat, ami a lineáris hozzárendelési feladat sok speciális esete közül az egyik [64].

Ha a célfüggvény tetszőleges kvadratikus alakú, tekintsük például a

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{t=1}^{n}\sum_{r=1}^{n} a_{ij}\mathbf{x}_{it}\mathbf{x}_{jr}b_{tr}$$

függvényt, akkor a korlátozó feltételek fenti $(0 - 1)$ értékű megszorítása esetén is NP-nehéz problémát nyerünk, az ún. kvadratikus hozzárendelési feladatot (lásd pl. [81] vagy [94]). Erre számos fontos elméleti eredmény található az irodalomban. Pardalos és szerzőtársai áttekintő tanulmányában 254 hivatkozás található a kapcsolódó publikációkra.

Az általunk tekintett (1)-(2) probléma vizsgálata önmagában is érdekes $M_{n,k}$ ezen L' megszorításával, másrészt ennek a vizsgálatnak számos haszna, következménye lehet. Mint azt fentebb láttuk, a megoldás diagonális együtthatómátrixok esetén megadható. Általánosítva ezt, ezen technika tetszőleges együtthatómátrixú feladatokra is alkalmazható: megoldva az L'-re megszorított feladatot, az optimum egy közelítése nyerhető. Ugyanakkor megjegyezzük, hogy ezen – a lehetséges pontok halmazának ezen L' megszorításával kapott – feladat megoldása tetszőlegesen távol lehet az eredeti, az $M_{n,k}$-n értelmezett feladatétól. Egy másik nyitott kérdés az utóbbi észrevétellel kapcsolatban, hogy vajon adható-e jó heurisztika az optimum értékének, f^*-nak a közelítésére abban az esetben, ha a probléma lehetséges megoldásainak halmazát megszorítjuk a koordinátatengelyek pontjaira, hasonlóan a csak négyzetes tagokkal rendelkező problémák esetéhez.

7. Példa. *Tekintsünk egy feladatot az $M_{3,2}$ sokaságon, a következő célfüggvénnyel:*

$$3x_{11}^2 + 6x_{12}^2 + 4x_{13}^2 + 7x_{11}x_{12} - 4x_{11}x_{13} - 18x_{12}x_{13} -$$

$$-3x_{21}^2 + 5x_{22}^2 + 3x_{23}^2 - 6x_{21}x_{22} + 7x_{21}x_{23} - 12x_{22}x_{23},$$

ami minimalizálandó a $0 - 1$ koordinátaértékű pontokra megszorított halmazon.

A feladat tulajdonképpen egy

$$A_1 = \begin{pmatrix} 3 & 7/2 & -2 \\ 7/2 & 6 & -9 \\ -2 & -9 & 4 \end{pmatrix},$$

$$A_2 = \begin{pmatrix} -3 & -3 & 7/2 \\ -3 & 5 & -6 \\ 7/2 & -6 & 3 \end{pmatrix}$$

mátrixokkal adott (1)-(2) típusú feladat ((2)-nek $0-1$ megszorítása mellett).

Az utolsó eredmény szerint ezen feladat optimális megoldása

$$\mathbf{x}^* = ((0,0,1)^T, (1,0,0)^T)$$

lesz, és az optimum értéke 1.

1.6. A fejezet összegzése

A fejezet egy, a Stiefel–sokaságokon definiált feladat számítási megközelítéseit, illetve azok támogatásait tárgyalta, mind elméleti mind numerikus úton, továbbá speciális esetek elemzését is szolgáltatta, melyeket példákkal illusztrált. Először megadtuk az (1) módon definiált probléma megoldását $M_{2,2}$-n (1.2. alfejezet). Érdekes további kérdés, hogy hogyan lehetne kritériumot adni az optimumhelyek számának végességére $M_{n,k}$-n, azaz jellemezni a megoldások struktúráját tetszőleges esetben.

Megmutattuk az 1.3. alfejezetben, a probléma számítógépes numerikus elemzésében, hogy egy egyszerű, $M_{2,2}$-n definiált, diagonális együtthatómátrixú feladat megoldása kb. 3 millió függvénykiértékelést igényel (amelyből 2,9 millió ún. „dense constraint" korlátozó feltétel kiértékelés) a GlobSol szoftver használatával. Ráadásul az eredményként kapott intervallumok nem ellenőrzöttek így ez alapján nem lehetünk biztosak abban, hogy vajon azok-e a tényleges megoldások. Ellenőrzött megoldásokat csak a poláralakos tesztfuttatások adtak.

Egy másik, nem túl bonyolult, $M_{3,3}$-on adott optimalizálási probléma pedig kb. $3,5$ napnyi CPU időt kívánt számítógépünkön és 36 különböző nem-ellenőrzött megoldást adott outputként. A kapott megoldás-intervallumok ebben az esetben nem-ellenőrzöttek, így a megoldás megbízhatóságára nincs garancia. A 36 kapott intervallum tartalmazza a 8 ismert optimumpontot, és az outputbeli intervallumok össztérfogata 10^{-15}-ször kisebb, mint az eredeti keresési intervallumé, tehát ilyen szempontból keresésünk sikeresnek mondható. Ennek ellenére, ha nem ismernénk a tényleges megoldásokat, akkor a megoldás helyességét nehéz lenne ellenőrizni, hiszen ez az eredetivel ekvivalens feladat lenne.

Ez fölveti olyan, adekvát tesztfeladatok definiálásának igényét, amelyek a numerikus eszközök teszteléséhez használhatók – akár valamiféle előfeldolgozás után, akár anélkül alkalmazzuk ezeket. A globális optimalizálási szakirodalomban megszokott módon tesztfüggvények szükségesek ennek elősegítéséhez: olyan feladatok, amelyek ismert optimumhelyekkel és optimumértékkel rendelkeznek. Az 1.4. alfejezetben ilyen tesztproblémákat adtunk meg az általunk vizsgált feladatra, minden $M_{n,k}$ sokaságon (azaz tetszőleges n és k értékekre).

Visszatérően foglalkoztunk a tárgyalt probléma azon speciális eseteivel, amikor annak A_i együtthatómátrixai $(i = 1, \ldots, k)$ diagonálisak.

A probléma egy másik speciális esetét vizsgáltuk az 1.5. alfejezetben, ahol a feladatot $\{-1, 0, +1\}$ értékű feladatként tekintettük. Bebizonyítottuk, hogy a feladat ezen megszorítása ekvivalens a hozzárendelési feladattal.

Záró megjegyzések. A fejezet kutatásainak alapötletét és inspirációját Rapcsák Tamás adta. Az 1.3. alfejezetben a GlobSol szoftverrel végzett futtatásokat a [11] közleményben szerzőtársam Tóth Boglárka végezte. Ezen kívül a fejezet többi eredménye saját. Végső technikai megformálásukban, leírásukban és megszövegezésükben valamennyi szerzőtársam fontos szerepet játszott.

2. fejezet

Az RPCRS – egy párhuzamos, sztochasztikus globális optimalizálási módszer

Ebben a fejezetben egy heurisztikus, sztochasztikus globális optimalizálási algoritmust, az RPCRS-t adjuk meg. Tárgyaljuk az RPCRS viselkedését, egyprocesszoros és többprocesszoros (párhuzamos) számítógépeken végzett tesztek segítségével elemezve azt.

Az RPCRS algoritmus a CRS (Controlled Random Search) algoritmus egy általunk javasolt új, finomított párhuzamos változata. A CRS eredetileg egy szekvenciális algoritmus. Így először a CRS-be bevezethető párhuzamosítás mértékének vizsgálatát végeztük el, ahol, vizsgálataink első fázisaként, az RPCRS tesztelését és végrehajtási eredményeinek értékelését a párhuzamos implementáció szimulálásával végeztük. Az RPCRS-nek a CRS-hez viszonyított párhuzamosítási foka egy felhasználó által megadott paraméterrel szabályozható. Ezt a paramétert a párhuzamos számítógépes rendszerhez hangolva kell beállítani. Megmutatjuk, hogy nagyobb érték, nagyobb párhuzamosítási fok esetén a szekvenciális és a párhuzamos futtatások is kedvezőbb végrehajtási eredményeket szolgáltatnak.

2.1. Bevezetés

A globális optimalizálás feladata a következő módon írható fel általánosan:

$$\min f(s),\ s \in S \subset R^n, \tag{28}$$

ahol az $f(s)$ *célfüggvény* egy valós, általában nemlineáris függvény, amely folytonos az S kompakt halmazon. S az ún. *keresési tartomány*. Ezen feltételek mellett jól ismert, hogy az

$$f^* \equiv \min f(s), s \in S$$

optimális megoldás létezik, és felvevődik S-en, azaz

$$S^* \equiv \{s \in S : f(s) = f^*\} \neq \emptyset.$$

A globális optimalizálási módszerek jellegük szerint általánosan két különböző csoportba oszthatók. Az első csoportot a *determinisztikus módszerek* alkotják, amelyek minden lépése meghatározható az előzőkből. Ezek általában a feladat bizonyos matematikai struktúráját, jellemzőit követelik meg. A másik a *sztochasztikus módszerek* csoportja, amelyek a lehetséges megoldások pontjainak egy véletlenszerűen választott mintaponthalmazával operálnak, és lokális nemlineáris optimalizálási eljárásokat tartalmaznak. A módszerek további osztályozásáról egy alapos tárgyalás található Törn és Žilinskas könyvében [107], illetve a globális optimalizálási módszerek aprólékos leírását és pontos elemzését nyújtó Handbook of Global Optimization című műben [53]. Mi ebben a fejezetben csak egyetlen sztochasztikus módszer tárgyalására szorítkozunk. Az ezt megelőző és az ezt követő fejezetekben alkalmazási szinten felhasznált egyéb, létező módszereket ott a szükséges mértékben, vázlatosan ismertetjük.

A gyakran elsőként felmerülő kérdés, amikor egy gyakorlati, globális optimalizálási problémát kell megoldanunk: *milyen típusú módszer, illetve milyen speciális algoritmus illeszkedik legjobban a probléma jellegéhez?* Más szóval: milyen típusú és melyik módszer alkalmazható leghatékonyabban az adott feladatra? Az adekvát válaszhoz a konkrét probléma mélyebb elemzése szükséges. A téma tárgyalása megtalálható az [51] munkában, ahol a probléma jellemzőinek, modellezésének és az alkalmazott globális optimalizálási módszerek kapcsolata mélyrehatóan elemzett. Röviden összefoglalva az mondható el, hogy a determinisztikus módszerek akkor hatékonyabbak, mint a sztochasztikus típusúak, amikor f is, és f valamely analitikus jellemzője(i) is rendelkezésre áll(nak). Az utóbbi(ak) lehet(nek) f deriváltja(i), és/vagy egyéb tulajdonságok, korlátok. Viszont, ha f egy úgynevezett „black-box" függvény (esetleg csak implicit módon, például egy őt kiszámító szubrutinnal adott), akkor a determinisztikus módszerek nem alkalmazhatóak hatékonyan.

A sztochasztikus globális optimalizálási módszerek ellenben nem követelik meg, hogy f bármilyen speciális struktúrával (jó tulajdonsággal, például a derivált létezésével vagy explicit megadottságával) rendelkezzék. Csak azt igénylik, hogy bármely S-beli pontban kiértékelhető legyen a függvény. Ekkor viszont az is elegendő, ha egy kiszámítási eljárás (pl. akár csak annak programkódjával) rendelkezésre áll. Ha ez, vagy bármilyen eszköz, akár mérési eredmények tetszőleges kívánt pontban tudják szolgáltatni a függvény kiértékelését, akkor a sztochasztikus módszerek alkalmazhatók az adott probléma megoldására, függetlenül attól, hogy a célfüggvény nem folytonos, vagy nem differenciálható S-en.

Ennek köszönhetően a legtöbb globális optimalizálási probléma megoldásánál alkalmazhatók sztochasztikus módszerek. Az olyan függvények esetén, amelyek kiértékelése költséges, a sztochasztikus globális optimalizálási módszerek használata versenyképesnek bizonyult. Ennek az az oka, hogy a determinisztikus módszerekkel szemben általában kevesebb függvénykiértékelés járul a (28) probléma megoldásához.

Gyakran alkalmazott ugyanakkor valamely *hibrid módszer* is: amikor (például intervallum-matematikán alapuló) determinisztikus optimalizálási módszerek alkalmazása előtt sztochasztikus módszereket [24] használnak. Ennek célja lehet a globális optimum értékére egy jó becslés nyerése. Egyes módszerek kifejezetten igénylik, hogy az optimum értéke ismert legyen, vagy annak egy jó becslése rendelkezésre álljon. A Branch-and-Bound típusú módszereknél [21, 53, 69] például jól használható, ha egy ilyen megfelelően jó becslés kezdettől fogva rendelkezésre áll [16, 17, 70].

A globális optimalizálás feladata nehéz (speciális függvényosztályokat kivéve általában NP-nehéz) feladat. Ebből kifolyólag a globális optimalizálási feladatoknál felmerülhet a szuperszámítógépek használatának igénye, főként olyan függvények esetén, amelyek kiértékelése költségigényes.

Ebben a fejezetben először leírjuk a W.L. Price [83, 84, 85] által tervezett CRS (Controlled Random Search) nevű sztochasztikus, klaszterező globális optimalizálási algoritmust, majd annak egy új, finomított változatát, az RPCRS-t adjuk meg és elemezzük. Az RPCRS-t ugyan eredetileg párhuzamos, többprocesszoros rendszerre terveztük, de megmutatjuk, hogy teljesítménye még egyprocesszoros környezetben is felülmúlja a CRS-ét.

Analízisünk egy numerikus esettanulmány, amelyben az RPCRS-sel elért gyorsítás („speed up") mértékét adtuk meg a szekvenciális CRS algoritmussal összehasonlítva. Elemzésünk kizárólag a gyakorlati futtatásokból nyert tapasztalati eredményeken alapszik, a globális optimalizálás szokásos módszerével egy tesztfüggvénysoron elemezve ezt. Annak ellenére, hogy eredményeink validálására széles körből választott nagyszámú tesztfüggvények egy csoportját használtuk, munkánk természetesen nem jelenti annak elméleti igazolását, hogy ugyanilyen eredményeket kapnánk más körből választott függvényekre. Megjegyezzük azonban, hogy a használt módszer, és a választott tesztfüggvények köre megfelel a globális optimalizálás szakirodalmában kialakított standard, hasonló módszerek tesztelésére bevett módon használt módszernek és „benchmark" függvénykörnek [26, 53].

Az eredeti módszeren elért gyorsítás mérését két különböző gyakorlati teszt csoporton végeztük. A tesztek első csoportja a párhuzamos jelleg előnyeinek kimutatására szolgál, míg a második fajta tesztek azt hivatottak kimutatni, hogy egy párhuzamos számítógépes környezetben működve algoritmusunk milyen gyakorlati teljesítményre képes.

Ez a fejezet a következőképpen épül föl: a 2.2. alfejezetben először leírjuk a CRS algoritmust, majd megadjuk ennek párhuzamos változatát, az RPCRS-t. A 2.3. alfejezetben az RPCRS-sel a CRS-hez viszonyítva elért gyorsítás mértékére nyert tapasztalati eredményeket mutatunk be egy olyan szabályozó paraméter függvényében, amely meghatározza a párhuzamosság fokát. Végül a 2.4. alfejezetben az RPCRS egy CRAY T3D típusú többprocesszoros számítási környezetben végzett párhuzamos futtatásainak numerikus eredményeit szolgáltatjuk.

2.2. Az RPCRS, mint a CRS algoritmus egy párhuzamos változata

A CRS-nek korábban már több párhuzamos megközelítését javasolták, különböző párhuzamos stratégiákat és párhuzamos számítógépeket használva [27, 37, 39, 75, 86, 104, 112]. Az RPCRS egy továbbfejlesztett változata a [38]-ban javasolt – centralizált, mester-szolgák kommunikációs modellt alkalmazó – PCRS (Parallel Controlled Random Search) algoritmusnak. Ez a CRS egy olyan párhuzamos változata, amelynek implementációja hatékonynak bizonyult egy olyan, képfeldolgozási területről származó függvény globális optimalizálására, amelynek kiértékelése nagyon költséges [37, 38].

Az általunk javasolt RPCRS algoritmus az eredeti szekvenciális CRS változat kis módosítása, amely annak jobb párhuzamosítását célozza. Az ahhoz szükséges módosítást hajtottuk végre rajta, hogy egy többprocesszoros számítógép processzorait állandóan munkával tudjuk ellátni, azaz, hogy egyidejűleg kiegyensúlyozottan leterhelve dolgozhassanak. Az RPCRS azt a lehetőséget biztosítja, hogy a CRS-nél jóval több mintapontban végezhessük el egyidejűleg a célfüggvény kiértékelését. Az erre a célra szolgáló módosításon kívül a CRS általános stratégiáját nem módosítottuk. A jobb áttekinthetőség céljából az RPCRS algoritmus tárgyalása előtt megadjuk a CRS algoritmus leírását.

A CRS algoritmus egy klaszterezési technikán alapuló, heurisztikus, direkt globális optimalizálási módszer, amelyet korlátozó feltételek nélküli és korlátozó feltételekkel ellátott feladatok megoldására javasolt Price. A CRS-t egyszerűségéhez képest jó gyakorlati eredményei miatt számos gyakorlati problémánál alkalmazták, és a gyakorlati alkalmazások során megbízhatónak és hatékonynak bizonyult. Talán éppen egyszerűségének is köszönhető ezen körbeli népszerűsége. Az is igaz ugyanakkor globális optimalizálás elméleti szempontból, hogy elméleti eszközökkel nem bizonyítottak rá konvergenciatulajdonságok.

A jelen dolgozatban feltesszük, hogy a megoldandó globális optimalizálási feladat (28) alakú, ahol S egy hipertéglalap. (Ez a feltevés a többi fejezetben is igaz a numerikusan is vizsgálatra kerülő problémák esetén.)

A CRS általunk használt pontos változatát az 1. Algoritmus írja le. Az algoritmus N darab, egyenletes eloszlás szerint véletlenül választott S-beli pontban a célfüggvény kiértékelésével kezdődik. Ez az algoritmus **Inicializálás** fázisa. Az N elemből álló mintaponthalmaz koordinátái és célfüggvényértékei egy $R = R^0, \ldots, R^{N-1}$ tömbben tárolódnak. Ezután az **Aktualizálás** lépésben kiszámítjuk azt, hogy függvényérték szerint melyik a legrosszabb és a legjobb mintapont (R^W és R^B). A **Generálás** lépésben új mintapontok ($\overline{P}$) generálódnak és értékelődnek ki. Az algoritmus addig iterálja az **Aktualizálás** és **Generálás** lépéseket, amíg a megállási feltétel nem teljesül.

Az algoritmus **Generálás** fázisában két különböző típusú mintapontot számítunk: elsődleges és másodlagos mintapontokat. Mindkét fajtájú pont a mintapontok egy $n+1$ elemű részhalmazából ($R^{j_0}, \ldots, R^{j_n}$) számítva generálódik, amely R^{j_i} $(i = 0, \ldots, n)$ pontok az R-ben tárolt N darab mintapontból véletlen módon választott pontok. (Megjegyezzük, hogy a CRS tekinthető az első olyan algoritmusnak, amely a pontok egy populációját használja.)

Az elsődleges pontok a Nelder-Mead módszerhez [78] hasonló módon generálódnak, tükrözve egy mintapontot (R^{j_n}-t) a ponthalmaz többi pontjának ($R^{j_0}, \ldots, R^{j_{n-1}}$) középpontján, $\overline{G}$-n át. Egy másodlagos pont viszont az R^{j_n} pont és a $\overline{G}$ középpont között helyezkedik el, az általuk alkotott szakasz felezőpontja. Míg az elsődleges pontok azt a célt szolgálják, hogy a keresés során amennyire csak lehet az egész keresési tartományt felderítsük *(globális keresés)*, addig a másodlagos pontok a konvergencia irányításában játszanak döntő szerepet *(lokális keresés)*. Másod-

1. Algoritmus: CRS($f, S, N, NF_{max}, \epsilon$)

begin (Inicializálás)
    $Iter := 0$; $D_{max} := \epsilon + 0.1$; $ns := 0$;
    Generáljunk egy N elemű mintaponthalmazt, R-et véletlen módon, egyenletes eloszlás szerint $(R = \{R^i \mid 0 \leq i < N\})$;
    Számítsuk ki az $f(R^i)$ értékeket $(0 \leq i < N)$;
end; **(Inicializálás)**
while $(D_{max} > \varepsilon$ **or** $f(R^W) - f(R^B) < \delta$ **or** $Iter \leq NF_{max})$ **do**
    Aktualizálás: Határozzuk meg azokat a W és B mintapontokat, melyekre: $f(R^W) \geq f(R^i) \geq f(R^B)$ $(0 \leq i < N)$;
    $Iter := Iter + 1$;
    begin (Generálás)
        Válasszunk $n + 1$ darab pontot véletlen módon egyenletes eloszlás szerint (R^{j_i}), az R halmazból;
        $\overline{G} := \sum_{i=0}^{n-1} R^{j_i}/n$;
        $\overline{P} := 2 \times \overline{G} - R^{j_n}$;   (Elsődleges pontok)
        if $\overline{P} \in S$ **and** $f(\overline{P}) < f(R^W)$ **then**
            $R^W = \overline{P}$; $ns = ns + 1$;
        if $RS = ns/Iter < 0.5$ **then**   (Sikerességi ráta teszt)
            $\overline{P} := (\overline{G} + R^{j_n})/2$;   (Másodlagos pontok)
            $Iter := Iter + 1$;
            if $f(\overline{P}) < f(R^W)$ **then**
                $R^W = \overline{P}$; $ns = ns + 1$;
    end; **(Generálás)**
endwhile;
return $\{R^B, f(R^B)\}$; $(f(R^B) \leq f(R^i);\ 0 \leq i < N)$

lagos pont kiszámítására csak akkor kerül sor, ha az aktuális elsődleges pont nem bizonyul javításnak, azaz a függvényértéke $f(R^W)$-nél, azaz a legrosszabb klaszterbeli ponténál nem jobb. Ehhez nem elegendő az, hogy az utolsó pont sikertelen ilyen szempontból, hanem még annak a feltételnek is teljesülnie kell, hogy az addig megvizsgált pontok kevesebb, mint az 50%-a javít.

Az eddig általánosan leírt eljárás számtalan módon variálható, amiatt is, mert különböző megállási feltételek használhatók. Az implementációnkban (lásd 1. Algoritmus) használt megállási feltétel a következő. Az algoritmus pontosan akkor áll le, ha a következő három feltétel valamelyike bekövetkezik:

(i) az R halmaz bármely két pontja közötti (euklidészi) távolság elegendően kicsire csökkent ($D_{max} = \max\{d(R^i, R^j);\ \forall\, 0 \leq i, j < N;\ i \neq j\} \leq \epsilon$),

(ii) az $f(R^i)$ értékek elég közel vannak egymáshoz $f(R^W) - f(R^B) < \delta$; ahol $f(R^W) = \max\{f(R^i)\}$, $f(R^B) = \min\{f(R^i)\}$, $0 \leq i, j < N$;

(iii) a függvénykiértékelések száma meghaladt egy előre megadott maximális értéket, NF_{max}-ot.

Az (i) feltétel célja azt biztosítani, hogy az összes mintapont egy kis méretű klaszterben helyezkedjék el. Ha ez nagyszámú függvénykiértékelés után sem következne be, akkor a (ii) feltétel megengedi, hogy az algoritmus végrehajtása befejeződjön akkor is, ha az összes mintapont függvényértéke majdnem egyenlő egymással. Ez a feltétel jól használható sok globális minimumponttal rendelkező függvény esetén. Végül, a (iii) feltétel azt az esetet kezeli, amikor az algoritmus nem konvergál. Ebben az esetben azt biztosítja, hogy a futása leálljon.

Észrevehető az, hogy a CRS algoritmusa erősen szekvenciális jellegű. Minden új $\overline{P}$ mintapont az N darab mintapont adatait tartalmazó R halmaz egy részhalmazának adataiból számítva generálódik. Ez a halmaz aktuálisan mindig az iterációs eljárásban addig talált legjobb pontok adatait tartalmazza (azok koordinátáit és felvett függvényértéküket). Biztosítva azt, hogy az $R^0, \ldots, R^{N-1}$ értékei egyidőben generálódhatnak, és az $f(R^0), \ldots, f(R^{N-1})$ értékek konkurens módon számítódhatnak, az algoritmusba már az **Inicializálás** fázisban bevezethető párhuzamos jelleg.

Az eredeti CRS algoritmus párhuzamosításához a módszer párhuzamos jellegének növelése céljából bevezetett átalakítás lényege, hogy az **Inicializálás** fázis után ugyanezen, a kezdeti N mintapontot tartalmazó R halmaz elemei alapján b számú elsődleges mintapontot generálunk. Ezeket egy A, FIFO-típusú bufferben ($A = A^0, \ldots, A^{b-1}$) tároljuk. $f(A^0)$ kiszámítása után ha $f(A^0) < f(R^W)$ teljesül, azaz A^0 „javít", akkor R^W-t kicseréljük A^0-ra a klaszterben. A bufferből töröljük A^0-t, és egy új pontot, amelyet a **Generálás** eljárásban kapunk, elhelyezünk a FIFO-buffer végén. Az egyetlen lényeges különbség az 1. Algoritmushoz képest, hogy egy b mintapontból álló ponthalmaz elemei mindig rendelkezésre állnak kiértékelésre. Ezen stratégia használatával a holtidő csökkenthető, és a párhuzamosság foka növelhető.

Az ilyen jellegű párhuzamos algoritmusok implementálására a centralizált modell jól alkalmazható. Modellünk ilyen, „mester–szolgák" kommunikációs kapcsolati sémát alkalmaz. A mesterprocesszor hajtja végre magát az optimalizálási algoritmust, állítja elő az új mintapontokat, és biztosítja azokat a szolgaprocesszoroknak. A szolgaprocesszorok feladata csak az, hogy kiértékeljék a célfüggvényt a mintapontokban és visszaküldjék az eredményt a mesterprocesszornak [39].

García és szerzőtársai [38, 39] hasonló stratégiát implementáltak, amely azonban teljesen aszinkron volt: a mesterprocesszor nem kezdett új – akár elsődleges, akár másodlagos – mintapont generálásába, amíg az aktuális mintaponthalmaz minden eleme ki nem értékelődött. A szolgaprocesszorok mindig csak egyetlen mintapont információit kapták meg és tárolták a visszaküldésig. A megközelítés teljesen aszinkron jellege ellenére gyakran üresjárati holtidőbe kerül számos szolgaprocesszor, várva azt, hogy a mesterprocesszor új mintapontot küldjön neki. Ez a hátrány kiküszöbölhető, ha a szolgaprocesszorok egy bufferben tárolnak több, kiértékelésre váró mintapontot. Így, amikor egy szolgaprocesszor befejez egy kiértékelést, és az eredményt visszaküldi a mesterprocesszornak, akkor ha van a lokális bufferében kiértékelendő pont, akkor a munkáját egy új, a bufferjében tá-

2. Algoritmus: Procedure Mester_RPCRS($f, S, N, p, npoints, NF_{max}, \varepsilon$)

1. **begin (Inicializálás)**
2. $Iter := 0$; $D_{max} := \varepsilon + 0.1$; $ns := 0$;
3. Generáljuk véletlen módon, egyenletes eloszlással N darab mintapont R halmazát, $R = \{R^i | 0 \leq i < N\}$;
4. **send** N/p darab mintapontot R-ből minden szolgaprocesszornak;
5. **receive** N darab pontot és függvényértékeiket a p szolgaproceszortól;
6. **end**; **(Inicializálás)**
7. **do** $j = 1$ **to** p
8. **do** $k = 1$ **to** $npoints$ $\qquad (b = p \times npoints)$
9. $Iter := Iter + 1$;
10. $\overline{G}^k := \sum_{i=0}^{n-1} R^{j_i}/n$;
11. $\overline{P}^k := 2 \times \overline{G^k} - R^{j_n}$; $\qquad$ (Elsődleges pontok)
12. **send** $\overline{P}^k$ a j. processzornak;
13. **while** ($D_{max} > \varepsilon$ **or** $f(R^W) - f(R^B) > \delta$ **or** $Iter \leq NF_{max}$) **do**
14. **Aktualizálás:** Határozzuk meg, hogy melyek azok az R^W és R^B mintapontok, amelyekre $f(R^W) \geq f(R^i) \geq f(R^B)$ $(0 \leq i < N)$;
15. $Iter := Iter + 1$;
16. $\overline{P}_{new}$:= **mintapont_generál()**;
17. **receive** $\{\overline{P}, f(\overline{P})\}$ az idp. processzortól; $(1 \leq idp \leq p)$
18. **send** $\overline{P}_{new}$ az idp. processzornak;
19. **if** $f(\overline{P}) < f(R^W)$ **then**
20. $R^W := \overline{P}$; $ns := ns + 1$;
21. **endwhile**;
22. **return** $\{R^B, f(R^B)\}$; $\qquad (f(R^B) \leq f(R^i);\ (0 \leq i < N))$

rolt mintapont kiértékelésével folytathatja, és nem kell arra várnia, hogy egy ilyen pont érkezzék a mesterprocesszortól.

Párhuzamos implementációnk két különböző folyamatból áll: a **Mester_RPCRS** (2. Algoritmus) és a **Szolga_RPCRS** (3. Algoritmus).

A **Mester_RPCRS** folyamat három különböző fázisból áll:

(i) az **Inicializálás** fázisból, ahol az N mintapontból álló R halmaz elemei véletlen módon generálódnak S-ből,

(ii) egy olyan fázisból, melyben $b = p \times npoints$ darab elsődleges mintapont számítódik,

(iii) a konvergenciaciklusból, ahol az elsődleges vagy másodlagos mintapontok generálódnak, az 1. Algoritmus stratégiájához hasonló módon.

Az **Inicializálás** lépésben a mesterprocesszor ciklikusan elosztja az N mintapontot a szolgaprocesszorok között. Ha a szolgaprocesszorok száma, p nagyobb,

3. Algoritmus: Procedure Szolga_RPCRS($f, S, N, p, npoints$)

1. **receive** N/p darab mintapontot és tároljuk el ezeket a szolgaprocesszor A_{idp} FIFO-bufferébe (idp a szolgaprocesszor sorszáma);
2. Számítsuk ki $f(A^i)$-t, $1 \leq i \leq N/p$;
3. **send** $\{A^i, f(A^i)\}$, $1 \leq i \leq N/p$ a mesterprocesszornak;
4. **receive** $npoints$ darab pontot a mesterprocesszortól és tároljuk el ezeket $(A^{first}, \ldots, A^{last})$;
5. **while** (**Mester_RPCRS** működik) **do**
6.   **while** van tárolt kiértékelendő pont $(A \neq \emptyset)$ **do**
7.   Értékeljük ki $f(A^{first})$-öt;
8.   **send** $\{A^{first}, f(A^{first})\}$ a mesterprocesszornak;
9.   **while** érkezett új pont a mesterprocesszortól **do**
10.   **receive** A^{last};
11.   **endwhile**;
12.   Várjunk egy új pontot a mesterprocesszortól;
13.   **receive** A^{last};
14. **endwhile**;

mint N, akkor a mesterprocesszor ugyan p darab mintapontot generál, de a szolgaprocesszoroktól visszakapva ezek függvényértékeit, ezekből csak a függvényértékek alapján a legjobb N darab mintapontot tárolja el R-ben. Ezek után a mesterprocesszor b darab elsődleges mintapontot állít elő, és minden egyes szolgaprocesszornak elküld ezekből $npoints$ darabot A $\overline{P}$ pontok b különböző mintapont, $\overline{G}$ és R^{j_n} felhasználásával számítódnak ki.

A konvergenciaciklusban meghatározzuk az $R^0, \ldots, R^{N-1}$ halmaz legrosszabb (legmagasabb függvényértékű) R^W pontját. Az algoritmus egy új, $\overline{P}_{new}$-val jelölt pontot is kiszámít a **mintapont_generál()** eljárásban. Ez az eljárás egy elsődleges vagy egy másodlagos mintapontot generál, ugyanazt a stratégiát követve, mint a CRS. Ezután a mesterprocesszor vár, várja valamely elküldött pontnak és kiszámított célfüggvényértékének a visszaérkezését. Ez bármelyik szolgaprocesszortól jöhet. Amint érkezik ilyen, még ennek feldolgozása előtt küld egy új, $\overline{P}_{new}$ mintapontot a küldő szolgaprocesszornak. Ezután ellenőrzi, hogy a kapott mintapontot elfogadja-e, és meghatározza, hogy a következő generált pontnak elsődleges vagy másodlagos mintapontnak kell-e lennie.

A **Szolga_RPCRS** folyamat is tartalmaz egy inicalizáló fázist, melyben a szolgaprocesszor N/p mintapontot kap a mesterprocesszortól, kiértékeli azokat, és visszaküldi őket – az eredménnyel együtt – a mesterprocesszornak. Ezután a szolgaprocesszor $npoints$ darab pontot kap kiértékelésre. Ezeket eltárolja saját, A_{idp} FIFO-bufferében. Amikor a szolgaprocesszor kiértékelt egy mintapontot, akkor visszaküldi ezt és ennek függvényértékét a mesterprocesszornak, és ellenőrzi, hogy érkezett-e közben új mintapont. Ha érkezett új pont, akkor a szolgaprocesszor kiolvassa az új ponto(ka)t és berakja lokális FIFO-bufferébe. Ha nem érkezett pont, akkor a szolgaprocesszor a bufferjebeli következő pont kiértékelésével folytatja a munkáját. Ha a FIFO-buffere üres, és a mesterprocesszor is dolgozik még, akkor a szolgaprocesszornak várnia kell egy új mintapontra.

A stratégia használatával a szolgaprocesszorok üresjárati ideje csökken – vagy akár teljesen megszűnik – és a kommunikáció pluszköltsége kisebb lesz, mivel a kommunikációs és számítási tevékenységek átlapolódnak.

2.3. Az RPCRS-t értékelő teszt egyprocesszoros környezetben

A párhuzamos implementációból fakadó problémák kiküszöbölésére, úgy mint az intenzív kommunikáció okozta kommunikációs pluszköltség *(overhead)*, vagy a szűk keresztmetszetek *(bottlenecks)*, az RPCRS egy gépfüggetlen értékelését valósítottuk meg; azaz az ebben az alfejezetben elemzett RPCRS futtatásokat egyprocesszoros környezetben végeztük. Az RPCRS teljesítményét a futásai során végrehajtott függvénykiértékelések számában mértük. Ezen tapasztalati analízis célja annak meghatározása, hogy az RPCRS viselkedése milyen a CRS-hez viszonyítva a bufferméret (b) függvényében. Fontos megjegyezni, hogy a $b = 1$ esetben az RPCRS megegyezik az eredeti, szekvenciális CRS algoritmussal. Így az RPCRS(b) algoritmus teljesítményének értékelését az RPCRS(1) eredményeivel összehasonlítva adjuk meg.

Egy huszonkét tesztfüggvényből álló halmazt használtunk az RPCRS konvergenciájának és párhuzamos viselkedésének, teljesítményének tesztelésére. Az algoritmus sztochasztikus jellege miatt a végrehajtott függvénykiértékelések száma függ az adott, egyedi futtatástól. Ezért mindegyik paraméterbeállítás esetén az algoritmust 100-szor futtatva kapott adatokból álló statisztikai mintát vizsgáltuk. Kiszámítottuk a *függvénykiértékelések átlagos számát*, és a hozzájuk tartozó *(95%-os) konfidenciaintervallumot* (lásd [96]) is.

Annak érdekében, hogy az RPCRS gyakorlati viselkedésének elemzését megkönnyítsük, a tapasztalati tesztek első fázisában csak hat, a globális optimalizálás irodalmában legklasszikusabbnak számító [26, 107] „benchmark" tesztfüggvényt használtunk: a Goldstein–Price, a Hartman3, a Hartman6, a Shekel5, a Shekel7 és a Shekel10 függvényeket.

Az RPCRS-nek két, a végrehajtásban meghatározóan fontos szerepet játszó paramétere van: a mintapontok alkotta klaszter pontjainak száma (N), és a buffer b mérete. A tesztek ezen első csoportjában a teljesítményt N és b függvényében is mértük. N-et mindig az adott probléma n dimenziószámának függvényében állítottuk be, az $N = 25 \times n$, $N = 50 \times n$, $N = 75 \times n$, és az $N = 100 \times n$ értékekre. A használt b bufferméretek 1, 2, 3, 4, 8, 16, 32 és 64 voltak. Az RPCRS megállási feltételét az $\varepsilon = 10^{-5}$, $\delta = 10^{-5}$ és $NF_{max} = 10^6$ paraméterértékekkel adtuk meg minden esetben.

A 6. táblázat numerikus eredményei két célfüggvény tesztjének adatait tartalmazzák, minden b,N értékpárra megadva. A táblázat az RPCRS száz futtatásából álló mintájának átlagos függvénykiértékelés-számát mutatja (a továbbiakban: *függvénykiértékelések száma, Fgvk*), továbbá a megfelelő konfidenciaintervallu-

6. táblázat. Az RPCRS futtatási eredményei a Goldstein–Price és a Shekel10 tesztfüggvényeken. Az (1) és (2) esetekben 98%-os és 97%-os volt a sikerességi ráta.

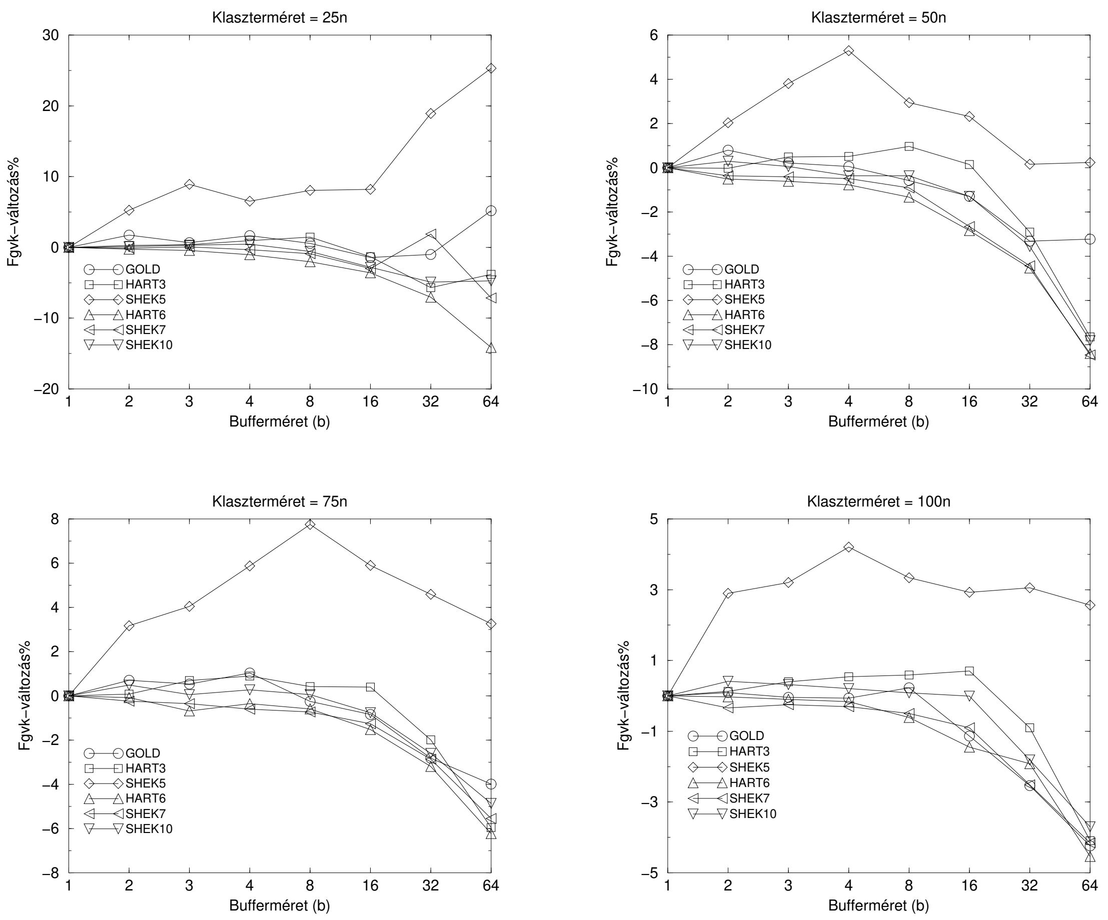
Goldstein–Price tesztfüggvény						
	Klaszterméret = $25 \times n$		Klaszterméret = $50 \times n$		Klaszterméret = $100 \times n$	
b	$Fgvk$	$Konf.int.$	$Fgvk$	$Konf.int.$	$Fgvk$	$Konf.int.$
1	$^{(1)}$1 622	[1 604, 1 640]	3 254	[3 230, 3 278]	6 505	[6 467, 6 544]
2	1 650	[1 632, 1 667]	3 261	[3 238, 3 284]	6 511	[6 479, 6 543]
3	1 633	[1 609, 1 657]	3 261	[3 238, 3 284]	6 502	[6 473, 6 532]
4	1 649	[1 630, 1 669]	3 256	[3 229, 3 283]	6 501	[6 464, 6 539]
8	1 631	[1 612, 1 649]	3 235	[3 207, 3 264]	6 520	[6 483, 6 558]
16	1 599	[1 575, 1 623]	3 212	[3 186, 3 237]	6 431	[6 394, 6 468]
32	1 606	[1 574, 1 637]	3 146	[3 112, 3 180]	6 340	[6 301, 6 378]
64	1 706	[1 674, 1 739]	3 149	[3 112, 3 186]	6 229	[6 195, 6 264]
Shekel10 tesztfüggvény						
1	5 310	[5 262, 5 357]	10 614	[10 562, 10 666]	21 454	[21 367, 21 540]
2	$^{(2)}$ 5 315	[5 260, 5 370]	10 645	[10 592, 10 698]	21 543	[21 475, 21 611]
3	5 326	[5 283, 5 369]	10 620	[10 562, 10 678]	21 523	[21 433, 21 613]
4	5 335	[5 295, 5 375]	10 576	[10 506, 10 646]	21 498	[21 425, 21 571]
8	5 279	[5 229, 5 329]	10 575	[10 509, 10 642]	21 472	[21 390, 21 553]
16	5 161	[5 117, 5 206]	10 477	[10 409, 10 545]	21 452	[21 371, 21 532]
32	5 049	[4 945, 5 152]	10 237	[10 171, 10 303]	21 066	[20 987, 21 145]
64	5 059	[4 905, 5 213]	9 784	[9 728, 9 839]	20 660	[20 577, 20 743]

mokat ($Konf.int.$).

A 6. táblázatban megfigyelhető, hogy a legkisebb klaszterméreteknél ($N = 25 \times n$) az RPCRS sikerességi aránya nem volt minden esetben 100% (lásd az ottani (1) és (2) megjegyzéseket). Az algoritmus konvergenciájának biztosításához nagyobb N klaszterméret értékeket kell választanunk. A 6. táblázatból látható, hogy a függvénykiértékelések száma csökkenő tendenciát mutat az N klaszterméret növekedésével. Ez alól kivétel, hogy a Goldstein–Price függvénynél, a klaszterméret legkisebb ($N = 25 \times n$) értéke mellett ez a tendencia nem figyelhető meg.

Az 5. ábra mutatja az RPCRS végrehajtásának eredményeit a hat tesztfüggvénynél a fenti négy különböző klaszterméret mellett. Mivel a függvénykiértékelések száma $1\,000$ és $40\,000$ között változik, így nem a függvénykiértékelések számát ábrázoltuk grafikonunkon, hanem ezek változását a $b = 1$ esethez képest százalékosan adtuk meg (a *Fgvk-változás*$\%$ = $\frac{Fgvk(b) - Fgvk(b=1)}{Fgvk(1)} \times 100\%$ képlettel számolva).

Az 5. ábra a klaszterméret különböző értékei mellett mutatja a *Fgvk-változás*$\%$ értékeit. Egy-egy grafikonon egy-egy rögzített klaszterméret esetén a bufferméretek függvényében ábrázoljuk a változást. A Shekel5 tesztfüggvény esetén mindig pozitív a megfelelő érték (növekedést kaptunk), de nagyobb klaszterméreteknél a



5. ábra. A függvénykiértékelések ($Fgvk$) számának százalékos növekedése több különböző N klasztérméret esetén. *Fgvk-változás*$\%$ = $\frac{Fgvk(b)-Fgvk(b=1)}{Fgvk(1)} \times 100\%$.

függvénykiértékelések számának növekedése csökkenésbe vált, és a bufferméret növekedésével kezd eltűnni a ($b = 1$ esethez képesti) növekedés.

A többi tesztfüggvény esetén az látható, hogy amikor a bufferméret nem nagyobb, mint 8, akkor a *Fgvk-változás*$\%$ érték alig módosul b függvényében, azaz a függvénykiértékelések átlagos száma csaknem konstans marad. Ennél nagyobb bufferméretek esetén – a legkisebb klasztérméret kivételével – a *Fgvk-változás*$\%$ értékek negatívak, és a klasztérméretek növekedésével monoton csökkenők. A negatív értékek azt jelentik, hogy ezekben az esetekben a számítási költség kisebb, mint az eredeti CRS algoritmus esetén. A $b = 64$ bufferméretnél a Goldstein–Price tesztfüggvény $N = 25 \times n$ klasztérmérethez tartozó esetét kivéve a többi esetben *Fgvk-változás*$\%$ értéke negatív. A bufferben tárolt pontok száma ezekben az esetekben ekkor már relatíve nagy, nagyobb, mint a buffer mérete. A Goldstein–Price tesztfüggvény kétdimenziós, emiatt a klaszter mérete, $(25 \times n =)\ 50$ ebben az esetben kisebb, mint a bufferméret ($b = 64$); azaz $\frac{N}{b} < 1$.

A gyakorlati tesztek ezen csoportja alapján a következő következtetéseket vonhatjuk le:

(i) az algoritmus elegendően nagy klaszterméret esetén konvergál a globális optimumponthoz,

(ii) amikor a bufferméret kisebb, mint a klaszterméret, de 8-nál nagyobb, akkor az RPCRS nem igényel nagyobb számítási költséget, mint az eredeti CRS, vagy az ahhoz hasonló mértékű marad.

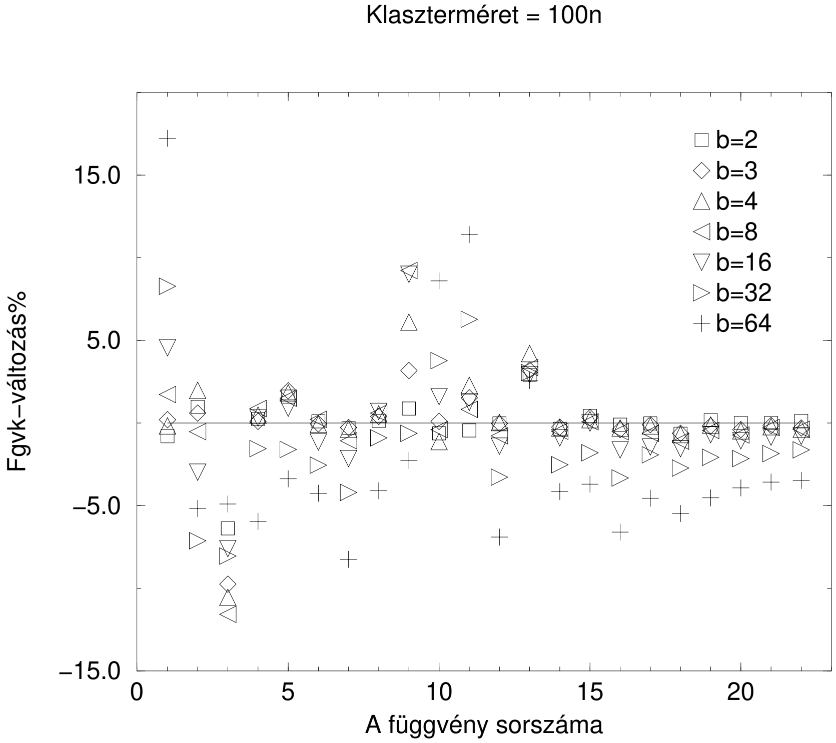
A tapasztalati tesztek második körében az RPCRS teljesítményének mérését és értékelését egy széles körből választott tesztfüggvény halmazon végeztük. A tesztfüggvények ezen csoportja az előzőleg alkalmazott függvények mindegyikét tartalmazza, és ezeken kívül még további tizenhét függvényt is. A függvényeket olyan szempontok alapján választottuk, hogy egyrészt n változzék (1-től 10-ig), hasonlóan a globális optimumhelyek száma (1-től nagyobb, mint 10-ig változik), és a függvények között legyen olyan, amelynek több, mint 1000 lokális optimumhelye van. Ezen tesztfüggvények részletesebb leírása szintén a fent említett:
`http://www.jgytf.u-szeged.hu/~balogh/Reszletes_adatok.ps`
internetes címen található.

A klaszterméretet minden esetben $N = 100 \times n$-re állítottuk be. Ezzel biztosítottuk, hogy $N > b$ teljesüljön minden tesztfüggvényre. Minden egyes tesztfüggvény esetén 100% volt az optimális megoldás megtalálási sikeressége az RPCRS-sel.

A 6. ábra mutatja az RPCRS futtatási eredményeit a 22 tesztfüggvényből álló halmazon. A grafikonon a vízszintes tengelyen ábrázoljuk a tesztfüggvény indexét. A tesztfüggvények a $b = 1$ esetben ($Fgvk(b = 1)$ a globális megoldás eléréséhez szükséges függvénykiértékelések száma szerinti növekvő sorrendbe rendezettek. Minden tesztfüggvényhez minden (b) bufferméret esetén kapott eredményt feltüntettünk egy függőleges egyenesen.

A kapott numerikus eredményeket úgy foglalhatjuk össze, hogy amikor a szükséges függvénykiértékelések száma elég nagy, akkor az RPCRS teljesítménye felülmúlja a CRS-ét; azaz $Fgvk(b) < Fgvk(b = 1)$ a b értékek zömére. Azon tesztfüggvények esetén, amelyekre a függvényértékek kiszámítási költsége alacsony, az algoritmus egy konkrét végrehajtása erősen függ a buffer méretétől, emiatt jobban változnak a megfelelő értékek (nagyobb a szórásuk). A 22 tesztfüggvényből csak 5 függvényen teljesít rosszabbul az RPCRS, mint a CRS, azaz $Fgvk(b) > Fgvk(b = 1)$ ekkor.

Ezen összehasonlító tesztek alapján elmondható, hogy a CRS párhuzamos jellegének RPCRS-t eredményező módosítása során nem csorbultak a CRS jellemző, kedvező tulajdonságai. Sőt, a számításilag költségigényes függvényekre az RPCRS felülmúlja a CRS-t. Meg kell jegyeznünk, hogy annak ellenére, hogy a módszer tulajdonságainak validálására felhasznált tesztfüggvények a szakirodalom „benchmark" tesztfüggvényei, ez nem jelenti az eddigi eredményeink elméleti igazolását, azaz más, speciális függvényeken vagy függvénykörön esetleg más eredményeket kaphattunk volna.

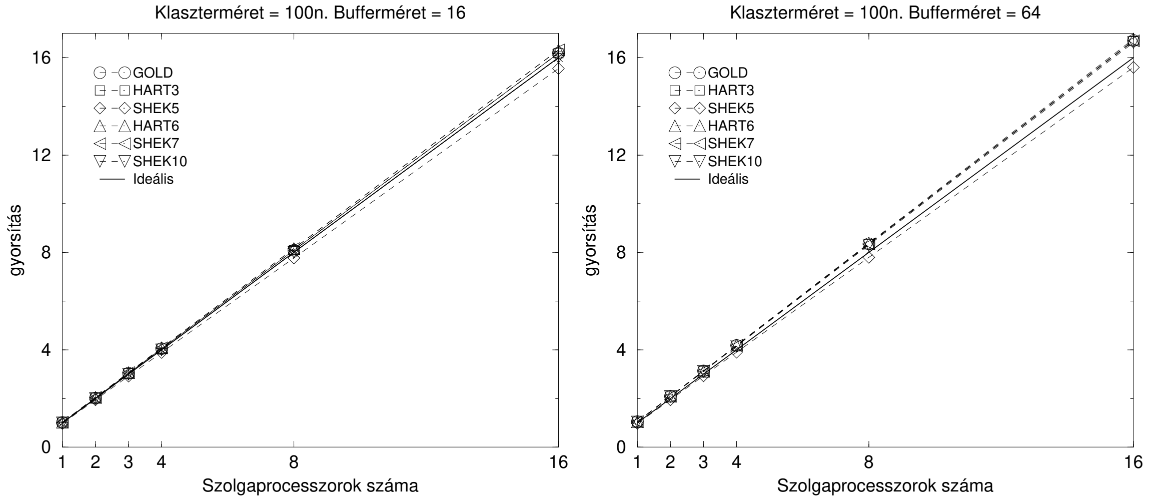


6. ábra. A függvénykiértékelések számának relatív (százalékos) növekedése a $b = 1$ esethez viszonyítva, $N = 100 \times n$ esetén a bufferméret (b) különböző értékei mellett. *Fgvk-változás*$\% = \frac{Fgvk(b) - Fgvk(b=1)}{Fgvk(1)} \times 100$.

2.4. Az RPCRS párhuzamos környezetbeli numerikus értékelése

Ebben az alfejezetben az RPCRS teljesítményét egy párhuzamos, Cray T3D számítógép-rendszeren (ahol a processzorok száma max. 16 lehetett) adjuk meg és értékeljük.

A program párhuzamos viselkedésének és teljesítményének gyakorlati vizsgálatához itt a klaszterméretet $N = 100 \times n$-re választottuk, továbbá a $b = 16$ és $b = 64$ bufferméreteket használtuk. Habár aszinkron párhuzamos programunk tervezésének fő célja volt a számításra és a kommunikációra fordított időintervallumok átlapoltatása, az algoritmus feltételezi, hogy a függvény kiszámítási költsége nagyobb, mint a mester- és a szolgaprocesszorok közötti adatcseréhez szükséges kommunikáció ideje. Ha a célfüggvény kiszámítási költsége túl alacsony, akkor a mesterprocesszornál szűk keresztmetszet jelentkezhet. A gyorsan kiértékelhető függvények esetén nem szükséges párhuzamos rendszert használni, így élhetünk feltevésünkkel. Ezért az RPCRS párhuzamos tesztjeinél azt szimulálva, hogy a tesztfüggvények rendelkezzenek a szükséges számítási költséggel, bevezettük azt a módszert, hogy a függvénykiértékelés idejéhez egy plusz időkésleltetést adtunk hozzá. Az általunk vizsgált 6 tesztfüggvény esetében két ilyen speciá-



7. ábra. Az RPCRS párhuzamos futtatásának eredményei: a szekvenciális esethez viszonyított *gyorsítás(b,p) = t(RPCRS(b=1,p=1))/t(RPCRS(b,p))* értékek. *Késleltetés = 0,03 másodperc.*

lis késleltetést alkalmaztunk: a 0,003 másodperces és 0,03 másodperces késleltetési beállításokat.

A 7. ábra mutatja a párhuzamos futtatások végrehajtási idejét az eredeti szekvenciális $CRS(= RPCRS(b = 1, p = 1))$ algoritmuséhoz viszonyítva kapott ún. *gyorsítás* értékeit. Ez a következő módon definiált:

$$\textit{gyorsítás(b,p)} = t(RPCRS(b = 1, p = 1))/t(RPCRS(b, p)),$$

ahol p a szolgaprocesszorok száma, b a bufferméret.

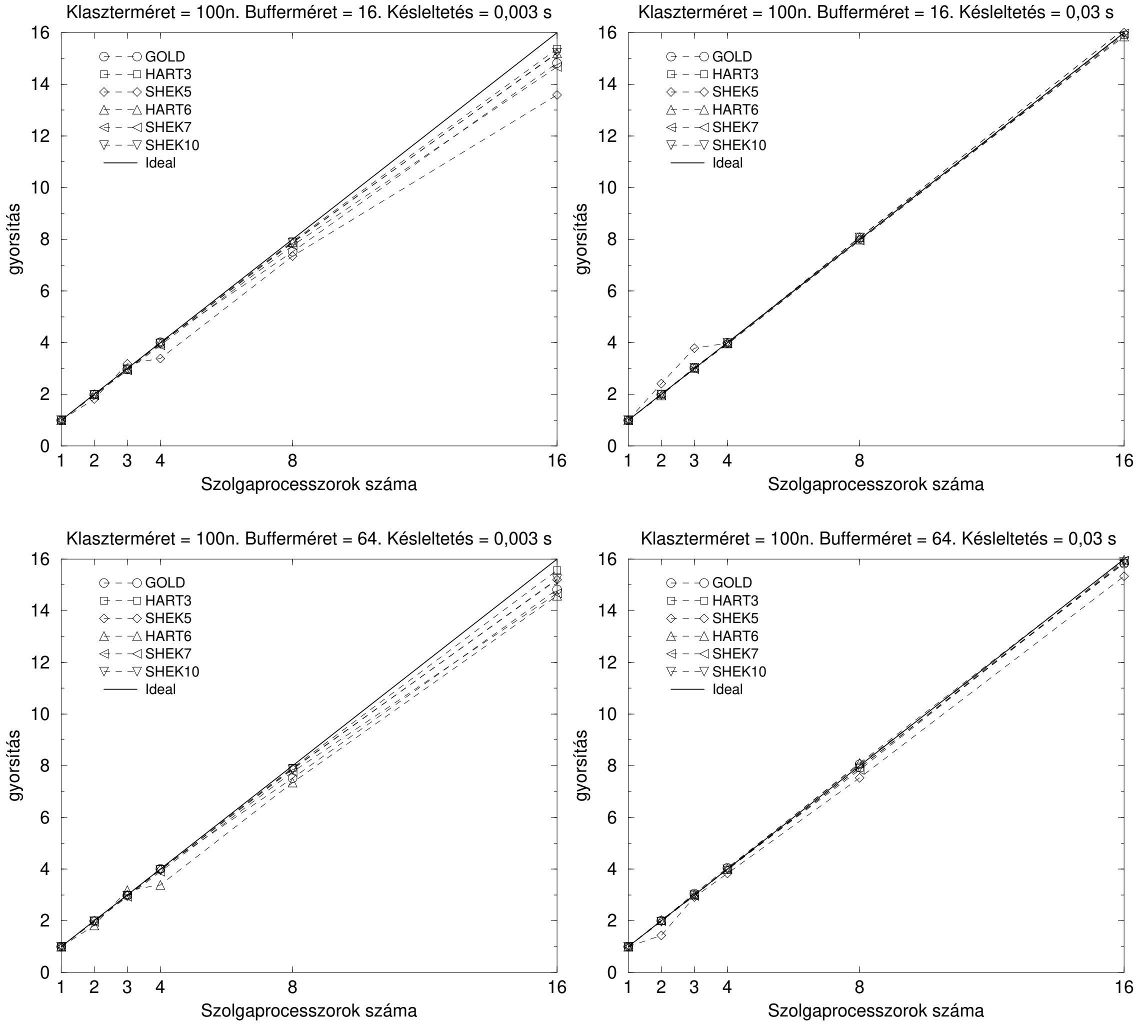
Az eredményekből kitűnik, hogy párhuzamos programunk implementációja sikeres, hiszen a legtöbb esetben a lineárisnál nagyobb gyorsítást kaptunk eredményül. Ezen, ún. szupergyorsítási ráták a fentiek mellett annak a tulajdonságnak köszönhetők, hogy a $b = 16$ és a $b = 64$ bufferméretek esetében az RPCRS algoritmus végrehajtásához szükséges függvénykiértékelések száma kisebb, mint a $b = 1$ esetben (lásd 5. ábra).

A 8. ábra mutatja a *gyorsítás* értékeket abban az esetben, amikor a párhuzamos program végrehajtási idejét az ugyanazon bufferméretet használó szekvenciális eset végrehajtási idejével hasonlítottuk össze, azaz, amikor egy-egy rögzített bufferméretnél a

$$\textit{gyorsítás(p)} = t(RPCRS(b, p = 1))/t(RPCRS(b, p))$$

definícióval dolgoztunk.

Az eredményekből kitűnik, hogy ebben az esetben majdnem lineáris gyorsítást kapunk minden függvényre, de nem kaptunk szupergyorsítást. A gyorsítási értékek a 0,03 másodperces késleltetés esetén közelebb állnak a lineáris gyorsításhoz, mint a 0,003 másodperces késleltetésnél.



8. ábra. A párhuzamos változat gyorsítása a szekvenciális esethez viszonyítva. *gyorsítás(p)* $= t(RPCRS(b, p = 1))/t(RPCRS(b, p))$.

2.5. A fejezet összegzése

Ebben a fejezetben a CRS nevű heurisztikus globális optimalizálási algoritmus párhuzamos implementációját, az RPCRS-t írtuk le és értékeltük. Megmutattuk, hogy az RPCRS kiszámításilag kedvezőbb költségű változat, mint az eredeti CRS. Elmondható továbbá, hogy az RPCRS aszinkron modellként könnyen implementálható valós párhuzamos, vagy elosztott számítógépes rendszerben.

Mint rámutattunk, annak ellenére, hogy nagy számú standard tesztfüggvényen végzett tesztfuttatásokat használtunk annak validálására, hogy az RPCRS felülmúlja a CRS-t, nincs elméleti módszerekkel igazolt okunk kimondani, hogy ez minden függvénynél igaz. Megjegyezzük, hogy az elméleti támogatás ilyen hiánya szinte valamennyi heurisztikus módszerre igaz. Az ilyen típusú elméleti vizsgálatok nehézségét jelzi, hogy az ebbe a csoportba tartozó, legismertebb és talán egyik legnépszerűbb Nelder–Mead féle módszerhez (amely a MATLAB programcsomag [52] keresőjeként is funkcionál) is csak évekkel, sőt évtizedekkel később születtek ilyen jellegű eredmények (lásd például [67]). Egy szép jövőbeli munka lehet eredményeink igazolása, megértésének elméleti alátámasztása matematikai módszerekkel.

Záró megjegyzések. A kutatások alapötletének adója a [80] publikációbeli szerzőtársam, Inmaculada García volt. A fejezet Inmaculada Garcíával és Pilar M. Ortigosával közös eredményeket tartalmaz.

3. fejezet

Fázisstabilitási problémák megoldása sztochasztikus globális optimalizálási módszerrel

Ezen fejezet témája egy sztochasztikus globális optimalizálási módszer alkalmazása fázisstabilitási problémák megoldására. Összehasonlítást nyújt más módszerekkel az irodalomból vett, mára már klasszikusnak számító tesztpéldákon keresztül, ismertetve az adott probléma nehézségeit.

Ezzel kapcsolatos munkáinkban [8, 9] és itt is egy módosított érintősík–távolsági függvényt használunk, és egy sztochasztikus klaszterező módszert ennek optimalizálására. A módszer hatékonyságát és robusztusságát az irodalomból vett tesztpéldákon demonstráljuk: a kénhidrogén–metán és a nitrogén-metán-etán rendszeren, amelyek a Soave-Redlich-Kwong–féle köbös állapotegyenlettel modellezettek. A kapott eredmények a módszer más sztochasztikus módszerekhez viszonyítottan nagyfokú megbízhatóságát igazolják. Az eljárás felhasználóbarát és jól hangolható. Az új módszerrel szignifikánsan csökkentjük a számítási igényt a felhasznált függvénykiértékelések száma és a CPU-idő igény tekintetében. Ez nagyon kedvező más sztochasztikus módszerekkel, pl. a véletlen pontból indított Newton-eljárással való összehasonlításban.

3.1. Bevezetés, előzmények

Az utóbbi időben nagy érdeklődés övezi új (mind elméleti, mind algoritmikus) megoldási módszerek kifejlesztését vegyipari műveletek tervezési, optimalizálási és irányítási feladataira. A fázisstabilitás elemzésének problémája az egyik legdinamikusabban változó terület, ennek formalizált leírása – történjen az akár az érintősík-távolsági függvény vagy a Gibbs-féle szabadenergia minimalizálásával – egyaránt megbízható, robusztus numerikus technikákat igényel a globális megoldás meghatározásához.

Lényegében két megközelítést használhatunk annak igazolására, hogy egy egyensúlyi megoldás megfelel a szabadenergia minimumának. Ezek: közvetlenül a szabadenergia minimalizálása, és az érintősík-távolsági függvény minimalizálása (érintősík-kritérium [1, 76, 111]). A második megközelítés nagyon hatékony lehet, valódi egyensúlyi megoldás megtalálási esélyének növelésében, mint ahogyan azt McDonald és Floudas [72, 73] írja.

A fázisegyensúlyi stabilitás elemzésére az érintősík-kritériumot több, mint 100 évvel ezelőtt vezette be Gibbs. A megvalósítás kiszámítási nehézségei miatt a módszer csak 1982-ben került a kutatások középpontjába. Az érintősík–távolsági függvény módszert egymástól függetlenül Baker és szerzőtársai [1], valamint Michelsen [76] diszkutálták 1982-ben. Utóbbi adta a módszer első numerikus implementációját. Ezek a cikkek forradalmasították a fázisegyensúlyi számításokat, számos olyan cikket és algoritmust eredményezve, amelyek finomították a módszert.

Legyen adott egy N komponensű rendszer, megadott T hőmérséklet és P nyomás mellett, ahol a kezdeti **z** komponenskeverék az úgynevezett input mólfrakció vektorral adott, azaz $\mathbf{z} = (z_1, z_2, \ldots, z_N)^T$ ($\sum_{i=1}^{N} z_i = 1,\ z_i \geq 0,\ i = 1, \ldots, N$), ahol z_i a kezdeti a komponenskeverékben az i. összetevő mólmennyiségének aránya az összes mólmennyiséghez képest. Az F érintősík–távolsági függvény minden lehetséges $\mathbf{y}$ pontban, azaz amelyre $\sum_{i=1}^{N} y_i = 1,\ y_i \geq 0\ (i = 1, \ldots, N)$ definiált; $F(\mathbf{y})$ a Gibbs-energiafüggvény $g(\mathbf{y})$-ban felvett értékének, $g(\mathbf{y})$-nak és és a **z** ponthoz tartozó L érintősík $L(\mathbf{y})$ pontjának távolsága, azaz $F(\mathbf{y}) = g(\mathbf{y}) - L(\mathbf{y})$.

Ezen feltételek mellett az érintősík-kritérium a következő: a rendszer egyensúlyának, azaz az induló fázis stabil voltának szükséges és elegendő feltétele, hogy az $F(\mathbf{y})$ érintősík–távolsági függvény nemnegatív legyen az N-komponensű termodinamikai fázistérben, azaz az összes lehetséges $\mathbf{y}$ vektorban ($\sum_{i=1}^{N} y_i = 1$, $y_i \geq 0, i = 1, \ldots, N$). Másként fogalmazva, ha $F(\mathbf{y})$ negatív bármely y pontban, akkor (és csak akkor) instabil a rendszer.

Ismert, hogy

$$F(\mathbf{y}) = \sum_{i=1}^{N} \mathbf{y}_i \big(\mu_i(\mathbf{y}) - \mu_i(\mathbf{z})\big),$$

ahol μ_i, az i. összetevő kémiai potenciálja. Michelsen rámutatott, hogy a rendszer stabilitása igazolt, ha $F(\mathbf{y})$ nemnegatív az összes stacionárius pontban, és speciálisan a minimumpontokban, mivel így mindenütt nemnegatív lesz [76]. Ha azonban a számítógépes eljárás nem tudja teljes megbízhatósággal megtalálni az összes globális minimumpontot, *akkor nincs elméletileg megalapozott garancia* a stabilitásra. Más szavakkal, még akkor is, ha nem kaptunk negatív értéket F-nek a megtalált stacionárius pontokban felvett értekei között, a vizsgált konfiguráció lehet instabil.

Michelsen megmutatta, hogy egy $\mathbf{y}$ stacionárius pontra a

$$\mu_i(\mathbf{y}) - \mu_i(\mathbf{z}) = K, \quad i = 1, \ldots, N, \tag{29}$$

feltétel teljesül, ahol K az i-től független konstans érték. Ilymódon az eredeti rendszer stabil, ha $K \geq 0$.

A kémiai potenciál kifejezésében valamely állapotegyenletre áttérve, mint amilyen például a Soave-Redlich-Kwong-féle állapotegyenlet [97]:

$$\frac{F(\mathbf{y})}{RT} = \sum_{i=1}^{N} y_i \big(\ln \varphi_i(\mathbf{y}) + \ln y_i - \ln \varphi_i(\mathbf{z}) - \ln z_i\big),$$

ahol φ_j az i. összetevő ún. fugacitási együtthatója (lásd pl. [97, 111, 113]).

Ekkor a stacionaritásra vonatkozó feltétel a következő lesz:

$$\ln \varphi_i(\mathbf{y}) + \ln y_i - h_i = k, \qquad i = 1, 2, \ldots, N,$$

ahol $h_i = \ln z_i + \ln \varphi_i(\mathbf{z})$ $(i = 1, 2, \ldots, N)$, és $k = K/RT$.

Az érintősík-távolsági függvény minimalizálása nehéz és nagy kihívást jelentő probléma. Michelsen a stacionaritási feltételeket egy szukcesszív helyettesítéses módszerrel oldja meg. Habár az algoritmus meglehetősen megbízhatónak bizonyult, tartalmaz egy körülményes inicializációs eljárást, és nem ad az összes megoldás megtalálására vonatkozó elméleti garanciát. Ez alatt azt értjük, hogy ha a módszer hibázik egy olyan $\mathbf{y}'$ helyi minimumpont azonosításában, melyre $g(\mathbf{y}') < 0$, akkor a rendszer hibásan stabilként azonosítja a konfigurációt. Számos kísérlet született ezen probléma megoldására, különböző numerikus technikákon alapuló módszereket és algoritmusokat bevezetve. Néhányat felsorolunk ezek közül, teljesebb áttekintés megtalálható [111]-ben.

Sun és Seider [103] egy homotópia-folytatási módszert alkalmazott, amely igen gyakran megtalálja az összes stacionárius pontot. Habár az ő módszerük könnyebben inicializálható, mint Michelsen módszere, ez mégis inicializálásfüggő, és nem szolgáltat az összes megoldás megtalálására vonatkozó elméleti garanciát. Ilyen típusú garancia itt csak speciális esetekben, korlátozó feltétel nélküli polinomiális rendszereknél [30] adható.

McDonald és Floudas voltak az elsők, akik átfogalmazták az alapproblémát a szabadenergia minimalizálására és az érintősík-kritériumra, a fázis- és kémiai reakcióegyensúlyt egy globális optimalizálási problémaként fölfogva. Egy dekompozíció-alapú megközelítést javasoltak a (29) egyenletekből eredményképpen kapott bikonvex problémákra, és korlátozás és szétválasztás típusú megközelítéseket adtak a különböző állapotegyenletekhez tartozó feladatok megoldására [30, 72, 73, 74]. Bevezették a szabadenergia minimalizálási probléma egy stabilitási kritériummal kombinált változatát, és kifejlesztettek ennek megoldására egy speciális célú programot, a GLOPEQ-t, amellyel alapos numerikus számítási vizsgálatokat végeztek nehéz fázisegyensúlyi problémákon. Ennek ellenére nem adtak javaslatot az összes fázis állapotegyenlet modellezésére.

Viszonylag új kutatási irány az intervallum-matematika használata a fázisstabilitási elemzéshez. Az intervallumos Newton-módszerek, egy általánosított felezéses megközelítéssel kombinálva alkalmasak lehetnek ilyen feladatok megoldására. Ezt az eljárást Hua és szerzőtársai alkalmazták a fázisstabilitási probléma megoldásában. A módszer megbízhatóságát számos publikációjuk [54, 55, 56, 57] elemzi. Ezen algoritmus alkalmazása azonban nagyon költségessé válhat – a tesztelt rendszerek komponensszámának növekedésével a számítási hatékonyság egyre kevésbé előrejelezhetővé válik. Ennek oka, mint bármely bináris fát használó technika esetén, hogy a módszer legrosszabb eset komplexitása exponenciális [57]. Néhány esetben a kiszámíthatósági helyzet még ennél is rosszabb lehet: adott számítógépmemóriával a bonyolultabb problémák nem oldhatóak meg.

Ily módon a fázisstabilitási számítások végrehajtásához fölmerül az igény olyan hatékony, általános célú módszerek fejlesztésére és alkalmazására, amelyek megbízható módon felhasználhatóak tetszőleges állapotegyenlethez. (A különböző állapotegyenletekről lásd pl. [93, 110]).

A jelen munka is egy ilyen megközelítés a fázisstabilitási probléma megoldására. Egy eltérő célfüggvény és egy hatékony globális optimalizálási módszer alkalmazását javasoljuk, és teszteljük. Bár a modell és az optimalizálási módszer ötvözésével kapott eljárás általános célú, azonban itt kizárólag köbös állapotegyenletekkel adott feladatok vizsgálatára koncentrálunk nevezetesen Soave-Redlich-Kwong–féle *(SRK)* köbös állapotegyenletekre [97].

A fejezet a következőképpen tagolódik: az előkészületek során először a javasolt eljárásban alkalmazott célfüggvényt és módszert írjuk le röviden. Ezután konkrét példákon végzett tesztekkel demonstráljuk eljárásunk robusztusságát és hatékonyságát. Először az irodalom tipikusan kihívást jelentő tesztpéldáján, a kénhidrogén–metán rendszer [103], majd a nitrogén–metán–etán rendszer termodinamikai stabilitáselemzésének példáján keresztül elemezzük.

3.2. A célfüggvény és az alkalmazott módszer

A stabilitás elemzésére használt módszerünk az érintősík-kritériumon nyugszik [1, 76], egy R.P. Stateva és St.G. Tsvetkov által bevezetett módosított érintősík-távolsági függvény a célfüggvény (lásd [98, 99, 100, 101]):

$$\Phi(\mathbf{y}) = \sum_{i=1}^{N} [k_{i+1}(\mathbf{y}) - k_i(\mathbf{y})]^2, \tag{30}$$

ahol

$$k_i(\mathbf{y}) = \ln \varphi_i(\mathbf{y}) + \ln y_i - h_i, \qquad i = 1, 2, \ldots, N,$$

$$h_i = \ln z_i + \ln \varphi_i(\mathbf{z}), \qquad i = 1, 2, \ldots, N,$$

és $k_{N+1}(\mathbf{y}) := k_1(\mathbf{y})$.

A (30) egyenletből közvetlenül adódik, hogy $\mathbf{y} = \mathbf{y}^*$ esetén, ha

$$k_1(\mathbf{y}^*) = k_2(\mathbf{y}^*) = \ldots = k_N(\mathbf{y}^*),$$

akkor

$$\Phi(\mathbf{y}^*) = 0,$$

ami a minimum értéke.

A $\Phi(\mathbf{y})$ zérushelyei, az $\mathbf{y}^*$ pontok a Gibbs-energiafelület olyan pontjainak felelnek meg, ahol az érintő hipersík párhuzamos a $\mathbf{z}$-hez tartozóval. Minden $\mathbf{y}^*$ vektorhoz egy k^* mennyiség tartozik, amelyre

$$k^* = k_i^*(y^*) = \ln \varphi_i(y^*) + \ln \mathbf{y}_i^* - h_i, \quad \text{minden } \; i = 1, 2, \ldots, N\text{-re},$$

amely k^* geometriailag a Gibbs-energiafelületen a $\mathbf{z}$-beli és az $\mathbf{y}^*$-beli érintősíkok távolságát adja (lásd pl. [111]). Ha a kiszámított k^* nemnegatív minden zérushelynél, akkor a rendszer stabil. Különben, ha legalább egy zérushelynél negatív k^* érték adódik, akkor a rendszer instabil.

A leglényegesebb lépés technikánkban $\Phi(\mathbf{y})$ összes zérushelyének meghatározása. A függvény olyan, hogy ennek zérushelyei a minimumhelyei is egyben. A feladat megoldására az 1.3. alfejezetben leírt GLOBAL nevű sztochasztikus, klaszterező globális optimalizálási algoritmust alkalmaztuk, Boender és társai globális optimalizálási módszerét [14], amelyet eredetileg Csendes Tibor adaptált [22, 23] és implementáltuk, hangoltuk a problémára. A két változat, a két alternatív módon választható lokális kereső algoritmus közül a tesztpéldákon [79, 106] az UNIRANDI módszer robusztusabbnak, bár kevésbé hatékonynak, míg a kvázi-Newton módszer a kezdőpontra érzékenyebbnek, bár pontosabbnak bizonyult [23, 79].

3.3. A módszer tesztelése

A kénhidrogén + metán rendszer példája

A módszer robusztusságát és hatékonyságát a kénhidrogén + metán rendszer példáján fogjuk illusztrálni, $T = 190$ K és $P = 40,53$ bar (40 atm) paraméterek mellett. Azért választottuk ezt a speciális példát, mivel ez az irodalom egy „benchmark" problémája [49, 56, 57, 76, 103, 113]. A probléma, amelyet Michelsen vetett fel [76], jól modellezi, hogy a stabilitási probléma kis rendszereknél is nehéz feladat lehet (lásd még [49, 103]). Habár a módszer alkalmazásával kapott számítási költségigény kedvező, hangsúlyozzuk, hogy nem adunk közvetlen összehasonlítást a lenti táblázatokban más módszerek alkalmazásával nyert adatokkal, éppen a módszerek eltérő jellege miatt. Ezért az itt szereplő adatok – a különböző betáplálások esetén az összes gyök megtalálásához szükséges függvénykiértékelések száma és CPU-ideje – inkább információul szolgálnak, semmint közvetlen összehasonlítás alapjaként.

Ugyanazokat a karakterisztikus paramétereket használtuk a kénhidrogén–metán rendszernél, mint Hua és szerzőtársai [55, 57], és ugyanazokat a komponenskeverékeket (input betáplálásokat) vizsgáltuk. Ez 6 darab különböző betáplálást jelent, a $\mathbf{z} = (0,0115; 0,9885)^T$, a $\mathbf{z} = (0,0187; 0,9813)^T$, a $\mathbf{z} = (0,07; 0,93)^T$, a $\mathbf{z} = (0,5; 0,5)^T$, a $\mathbf{z} = (0,888; 0,112)^T$, és a $\mathbf{z} = (0,89; 0,11)^T$ betáplálásokat. Az első érték a kénhidrogén (H_2S), második a metán (CH_4) aránya a komponenskeverékben. A betáplálásokra sorrendben, úgy mint $1., 2., \dots, 6.$ betáplálás fogunk hivatkozni.

$\Phi(\mathbf{y})$-nak a különböző betáplálásokra általunk kapott gyökei megegyeznek az irodalomban találhatókkal. Ezeket terjedelmi okokból nem közöljük, a pontos adatok a [9] közleményben, illetve ennek

`http://www.jgytf.u-szeged.hu/~balogh/fpe.doc`

internetes címen elérhető kéziratában megtalálhatók. A gyökök száma valamennyi

7. táblázat. A GLOBAL futtatási eredményei a kénhidrogén + metán rendszer példáján $T = 190$ K és $P = 40,53$ bar paraméterek mellett.

Alk. lok. kereső	Kvázi-Newton			UNIRANDI		
1. beállítás: $M = 200$, $D = 200$						
$Input$	*S. ráta* (%)	$Fgvk$	$CPU(s)$	*S. ráta* (%)	$Fgvk$	$CPU(s)$
1. betáplálás	99,90	2 730	0,1959	99,65	3 244	0,1755
2. betáplálás	99,90	2 646	0,1989	99,80	4 124	0,2270
3. betáplálás	99,85	2 704	0,1945	100	3 950	0,2283
4. betáplálás	99,95	2 596	0,2006	99,85	4 123	0,2202
5. betáplálás	99,95	2 649	0,2360	100	4 040	0,2778
6. betáplálás	99,90	2 627	0.2359	100	4 225	0,2850
2. beállítás: $M = 300$, $D = 300$						
$Input$	*S. ráta* (%)	$Fgvk$	$CPU(s)$	*S. ráta* (%)	$Fgvk$	$CPU(s)$
1. betáplálás	100	3 584	0,2585	100	5 261	0,3298
2. betáplálás	100	3 671	0,2800	100	6 129	0,3674
3. betáplálás	100	3 689	0,2688	100	6 120	0,3366
4. betáplálás	99,85	3 409	0,2641	100	6 122	0,3394
5. betáplálás	100	3 848	0,3126	100	6 543	0,4213
6. betáplálás	100	3 862	0,3168	100	6 442	0,4307

esetben 3 és 5 között van (3 csak az 1., $\mathbf{z} = (0,0115; 0,9885)^T$ betáplálás esetén).

A következőkben az alkalmazott módszer robusztusságát és hatékonyságát tárgyaljuk. A 7. táblázat mutatja az eredményeket, melyeket a kvázi-Newton és az UNIRANDI lokális kereső módszerek alkalmazásával kaptunk. A mintavételek $3\,000$ független futtatás eredményei voltak (mindegyik betáplálásnál és mindkét lokális kereső módszernél). Szintén tartalmazza a 7. táblázat a számítás során végrehajtott függvénykiértékelések átlagos számát $(Fgvk)$, és az átlagosan szükséges összes CPU-időt egy egyprocesszoros Pentium III, 500 MHz-es számítógépen, 256 MB RAM és Linux operációs rendszer használatával.

Mivel a módszer sztochasztikus jellegű, így nem szolgáltat az összes lokális minimum megtalálására vonatkozó elméleti garanciát. Ha egy konkrét futtatásnál nem találunk meg minden gyököt, akkor azt a futtatást sikertelennek nevezzük, egyébként sikeresnek. Az $S.\ ráta$ oszlopokban az összes független keresés megtalálási sikeresség arányait tüntetjük fel mindegyik táblázatban. Így a feltüntetett megbízhatósági ráták $0,05 - 0,35\%$-os hibaszázaléknak felelnek meg. A fentiekből következően, ha akár csak egyetlen gyököt nem sikerült megtalálni a konkrét keresés során, akkor azt mondjuk, hogy az sikertelen volt. Megjegyezzük, hogy gyakorlatunkban minden sikertelen keresésnél egyetlen, és minden esetben a legnehezebben megtalálható, kis konvergenciasugarú $\mathbf{y} = (0,01...; 0,98...)^T$ gyököt nem sikerült megtalálni.

A módszer nem kíván meg semmiféle előzetes információt sem a célfüggvényről, sem a derivált függvényekről. Szintén nem szükséges előzetes információ a minimum értékéről sem (azaz, hogy ez zérus). A módszer későbbi továbbfejlesztésénél ez utóbbi kihasználása gyorsíthat a számítási sebességen.

A program lehetőséget biztosít a számítás során a szignifikáns tizedesjegyek számának beállítására, amely az eredményben is egy minimálisan garantált pontosságot ad. Az általunk használt beállítás 5 tizedesjegy volt, amely viszont az optimumnál nem 10^{-5} pontossági értéket jelentett, hanem $10^{-8} - 10^{-15}$ értéket (a kisebb, 10^{-8} pontosságot a legnehezebben megtalálható $0,01...$ gyöknél), mely itt egyben a gyökök függvényértékében a zérus megközelítését jelenti. Ez a pontosság az alkalmazásban elegendő, de jóval nagyobb megkívánt pontosság is elérhető a számítási igény növekedése árán.

Megváltoztattuk a GLOBAL program alapértelmezett algoritmusparamétereinek beállításait is, miután a helyi kereső eljárás kezdőpontjaként kiválasztott mintapontok számában az aktuális minta C mérete legfeljebb 20 volt, és az M mérete legfeljebb $10\,000$ (lásd [22, 23]). A konkrét vizsgált esetnél a $0,01...$ gyök völgyének szélessége viszonylag kicsi. Ez azt jelenti, hogy például a hatodik, $z = (0,89; 0,11)^T$ betáplálásnál $10\,000$ darab, egyenletes eloszlás szerint generált mintapont esetén az első 28 legkisebb függvényértékű pont a $10\,000$ mintapontból a többi gyök völgyében helyezkedik el. (Csak a 29. legjobb pont származik a $0,0192$ gyök völgyéből, ami rossz esélyt ad ezen gyök megtalálására.)

Az algoritmusparaméterek legkedvezőbb választásának kérdését [9]-ben elemeztük, kísérleti úton. Különböző paraméterbeállítások hatásait megvizsgálva a sikerességi arányra és a számítási költségre, az $M/D = 1$ választást ajánlottuk, amely szintén eltér a [22]-ban javasolttól.

Ezen belül kétféle beállítással kapott eredmények adatait is feltüntetjük a 7. táblázatban: amikor M és D mérete egyaránt 200 volt, illetve amikor mindkettőé 300. Érdekes tény, hogy mindkét paraméterbeállítás esetén a módszer a sztochasztikus módszerekhez képest nagyfokú sikerességi arányt produkált, mégis ezen belül a második beállítás ($M = 300$, $D = 300$) esetén ez nagyobb volt, magasabb számítási költség árán. Mindebből kitűnik, hogy a paraméterbeállítások és a megkívánt pontosság a felhasználó által szabályozható a probléma jellegének függvényében, illetve attól függően, hogy a felhasználó a nagyobb fokú megbízhatóságot vagy inkább a számítási idő csökkentését részesíti előnyben.

A 7. táblázat értékeit magas megbízhatósági rátához tartozó algoritmusparaméterekkel nyertük. Ha megelégszünk alacsonyabb megbízhatósági szinttel, akkor a számítások költsége kisebb lesz. [9]-ben a legjobb paraméterbeállítás(ok) megtalálását célzó kísérleteink során ilyenek voltak az $M/D = 100/25$, $100/50$, $100/75$, $100/100$ beállítások. Az ezekhez tartozó eredményeket is közöltük az ottani táblázatban. Ezekben a számításigény jóval kevesebb, de a megbízhatóság nem elfogadható, kivéve az utolsó $M/D = 100/100$ beállításhoz tartozó – 90% fölötti megtalálási sikerességű – eredményeket. Ez azt is jelenti, hogy kisebb megbízhatóság árán, más paraméter-beállításokkal a számításigény csökkenthető. Más, különböző beállításokkal, mint azt a [8, 9] közleményekben taglaltuk, ennek megfelelően, például, a kvázi-Newton módszer alkalmazása során, ha $99,83\%$-os megbízhatósági mutatóval megelégedünk ($99,90\%$-os helyett) a függvénykiértékelések számában, illetve a CPU időben mért számítási költség $23 - 32\%$-kal kevesebb lesz, nevezetesen $2\,066$ függvénykiértékelés elegendő lesz átlagosan 2666 helyett, és $0,2820$ másodperc CPU-idő $0,4295$ másodperc helyett. Az UNIRANDI módszer használata esetén, $99,66\%$ megbízhatósági mutató esetén ($99,82\%$ helyett) a számítási költségigény $1,6\%$-kal csökken.

A nitrogén + metán + etán rendszer példája

Harding és Floudas [49], valamint Hua és szerzőtársai [57] vizsgálták a nitrogén + metán + etán rendszer 4 különböző betáplálását $T = 270$ K hőmérsékleten és $P = 76$ bar nyomás mellett. A betáplálások, mint az a [9] munkában is megtalálható, a $\mathbf{z} = (0,3; 0,1; 0,6)^T$, a $\mathbf{z} = (0,15; 0,3; 0,55)^T$, a $\mathbf{z} = (0,08; 0,38; 0,54)^T$, és a $\mathbf{z} = (0,05; 0,05; 0,9)^T$, komponenskeverékek voltak (a három koordináta rendre a nitrogén, a metán és az etán arányát jelzi). A négy betáplálásra ebben a sorrendben, mint 1., 2., 3., és 4. betáplálásra hivatkozunk. A gyökök száma az első két betáplálásnál három-három, az utolsó kettőnél egy-egy. Ezen minimumhelyek pontos értékei szintén a [9] publikációban, vagy ennek kéziratában, a világhálón a fent említett címen található meg.

Mi ugyanezeken a feladatokon teszteltük módszerünket, a GLOBAL-t alkalmazva itt is, a kvázi-Newton és az UNIRANDI rutinokkal. A hardveres környezet és a független futtatások száma $(3\,000)$ megegyezett az előzőekkel $(3\,000)$.

A tesztek eredményeiben a lokális minimumok száma mind a négy tesztelt betáplálás esetén megegyezett a [49, 57]-beliekkel. A lokális minimumhelyek koordinátái (komponensei) némiképpen eltértek attól, de ez annak a következménye, hogy mi a rendszer termodinamikai modellezésére az SRK (Soave-Redlich-Kwong–féle) köbös állapotegyenleteket [97] használtuk, míg [49]-ben és [57]-ben PR (Peng-Robinson–féle) köbös állapotegyenleteket [82] alkalmaztak. (A különböző módszerekről, állapotegyenletekről lásd pl. a [111] áttekintő tanulmányt.)

A [49]-ben alkalmazott lokális keresővel és a [57]-ben használt intervallumos Newton–módszerrel szemben a GLOBAL algoritmussal végzett tesztjeink során nem tapasztaltunk semmilyen nehézséget az *összes* globális minimumhely meghatározásában. Még az említett publikációkban alkalmazott eltérő módszerek számára kihívást jelentő második, $\mathbf{z} = (0,15; 0,3; 0,55)^T$ betáplálásnál sem ütköztünk nehézségbe a gyökök nagy pontosságú meghatározásakor. Ezen betáplálás esetén a problémát az említett egyéb módszerek számára az jelenti, hogy az egyik zérushely, a $(0,1629; 0,3107; 0,5264)^T$ pont a triviális megoldáshoz, a $(0,15; 0,3; 0,55)^T$ zérushelyhez közel található. Sőt, ezen túlmenően az is elmondható, hogy nem volt szignifikánsan nagyobb a szükséges függvénykiértékelések száma, és az átlagos CPU-idő sem ekkor, mint a másik három betáplálásnál.

Hasonlóan az előzőleg vizsgált kénhidrogén + metán tesztpéldához, itt is az $M/D = 1$ beállítás ajánlható, szintén vagy az $M/D = 200/200$ paraméterekkel, vagy az $M/D = 300/300$ beállítással, függően a felhasználói prioritástól: az első kisebb számítási költséggel, a második nagyobb sikerességi rátával rendelkezik. A 8. táblázatban az ezekkel a paraméterbeállításokkal kapott eredményeket jelenítettük meg, mind a kvázi-Newton típusú, mind az UNIRANDI lokális kereső

8. táblázat. A GLOBAL futtatási eredményei a nitrogén + metán + etán rendszer példáján $T = 270$ K és $P = 76$ bar paraméterek mellett.

Alk. lok. kereső	Kvázi-Newton			UNIRANDI		
1. beállítás: $M = 200$, $D = 200$						
$Input$	*S. ráta (%)*	$Fgvk$	$CPU(s)$	*S. ráta (%)*	$Fgvk$	$CPU(s)$
1. betáplálás	100	8 124	0,7057	100	15 012	1,1581
2. betáplálás	100	7 117	0,6535	100	14 791	1,1297
3. betáplálás	99,55	7 080	0,5539	100	15 958	1,4085
4. betáplálás	100	7 789	0,6318	100	7 456	0,5635
2. beállítás: $M = 300$, $D = 300$						
$Input$	*S. ráta (%)*	$Fgvk$	$CPU(s)$	*S. ráta (%)*	$Fgvk$	$CPU(s)$
1. betáplálás	100	8 124	0,7057	100	15 012	1,1581
2. betáplálás	100	7 117	0,6535	100	14 791	1,1297
3. betáplálás	99,55	7 080	0,5539	100	15 958	1,4085
4. betáplálás	100	7 789	0,6318	100	7 456	0,5635

eljárások használatával.

További paraméterbeállításokat itt sem alkalmaztunk, mert érdekes módon, (ugyanezt tapasztaltuk a kétkomponensű rendszernél is) az $M/D = 400/400$ vagy az $M/D = 500/500$ beállítások a függvénykiértékelések számának és a CPU-időnek fölösleges emelkedésével járnak, az eredmények megbízhatóságának további javulása nélkül.

3.4. A fejezet összegzése

A fejezetben egy sztochasztikus globális optimalizálási módszer alkalmazását vizsgáltuk meg kémiai fázisstabilitási problémák megoldására. A feladat esetén fontos az összes gyök (az összes minimum, amelyek az általunk alkalmazott modell célfüggvényénél egybeesnek a zérushelyekkel) megtalálása, enélkül a fázis stabil vagy nem stabil volta nem értékelhető helyesen.

A GLOBAL megbízhatóságát és hatékonyságát az irodalom „benchmark" tesztpéldáin vizsgáltuk, a számítási költségigény és az összes globális minimumpont megtalálásának sikeressége által kifejezve ezeket. Elemeztük a felhasználó által hangolható algoritmusparamétereket, ezeket a tárgyalt módon beállítva a független futtatásokból kapott gyakorlati tapasztalati eredmények kedvezőnek mutatkoznak, a számítási igény és a megtalálási sikeresség által kifejezve.

A fejezet tárgyalása a Csendes Tiborral és Roumiana P. Statevával közös publikációkra épült. A tárgyalt modellt és a célfüggvényt Roumiana P. Stateva és Tsvetkov vezették be [99, 100] publikációikban. Az alkalmazott módszer Csendes Tibor módszerének adaptált változata.

Önálló eredmények: A GLOBAL-lal végzett tesztek végrehajtása, az ajánlott algoritmusparaméterek hangolása és a legmegfelelőbb kiválasztása voltak. Hosszú programozási munkát igényelt a FORTRAN nyelvű forráskódokban elérhető célfüggvény kiszámításához szükséges számítások, valamint a GLOBAL forráskódjának összeillesztése (a két különböző szoftver FORTRAN-ban adott forráskódú rutinjait ehhez C nyelvre kellett fordítani), továbbá a numerikus tesztek elvégzése.

Egy további, friss publikációról. Megjegyezzük továbbá, hogy a fázisstabilitási feladat megoldására több különböző módszertan létezik. Az itt tárgyalt alkalmazás az érintősík-távolsági függvényt használó módszerek csoportjába tartozik. Egy másik ilyen módszer az úgynevezett *területi módszer (Area method)*, melynek elméleti szempontú általánosítását adja egy frissen megjelent, a brit R.J.B. Cravennel és a bolgár R.P. Stateván közös publikációm [5]. Ennek gyakorlati alkalmazása további, jövőbeli vizsgálatokat igényel.

4. fejezet

Alsó és felső korlátok szemi-on-line ládapakolási feladatokhoz

A dolgozat ezen fejezetében a [3, 4, 10] munkák alapján alsó és felső korlátokat adunk az itt definiált c-átpakolásos szemi-on-line ládapakolási feladatra. Az alsó korlátok érvényesek lesznek rokon feladatokra és 1-nél nagyobb dimenzióban is.

4.1. Bevezetés, előzmények és definíciók

A ládapakolási feladat több, mint 35 éve kutatott problémája a diszkrét optimalizálás területének. Ezen belül a klasszikus egydimenziós ládapakolási feladat az egyik legismertebb, intenzíven kutatott kombinatorikus optimalizálási probléma. Az alapfeladat hétköznapi szavakkal is könnyen megfogalmazható: tárgyak sorozatát kell elpakolni a lehető legkevesebb számú, azonos méretű (kapacitású) ládába. Ennek egydimenziós változatában a *tárgy mérete*ként csak egyetlen paramétert (a példa kedvéért mondjuk csak a térfogatot) vesszük figyelembe, a többit elhanyagoljuk. Nyilvánvalóan egy ládába nem rakhatunk nagyobb összméretű elemeket, mint a láda kapacitása, valamint a tárgyak nem „fedhetik át" egymást. A ládák kapacitását az irodalomban szokásos módon egységnyinek definiáljuk, az érkező tárgyak sorozatát pedig a $(0, 1]$ intervallumba eső számok egy listájával adjuk meg.

Formalizálva a feladat a következő: adott olyan tárgyak (a továbbiakban tárgyak helyett az *elemek* szót fogjuk használni) egy $L = \{x_1, x_2, \ldots, x_n\}$ listája, ahol az elemek mérete a $(0, 1]$ intervallumba esik. Adott továbbá egység kapacitású ládáknak egy végtelen listája. Az L lista minden egyes x_i elemét hozzá kell rendelni pontosan egy ládához úgy, hogy a ládában lévő elemek méreteinek összege nem haladhatja meg a láda kapacitását. (Ahol a szövegkörnyezetben nem érthető félre, ott egy x_i elem méretét szintén x_i-vel jelöljük, különben $s(x_i)$-vel.) A cél a felhasznált (a pakolás befejeztekor nem üres) ládák számának minimalizálása. Jól ismert az a tény, hogy optimális pakolást találni *NP-nehéz* feladat [40]. Az utóbbi évtizedekben nagy számú, elfogadható közelítést nyújtó algoritmust publikáltak a téma kutatói.

A különböző algoritmusok hatékonyságának mérésére két általános módszer használt: a legrosszabb eset viselkedés vizsgálata, vagy – feltéve az elemek valamely valószínűség-eloszlását – az algoritmusok átlagos eset (valószínűségi) vizsgálata. Jelen dolgozatban csak az algoritmusok legrosszabb eset viselkedési vizsgálatára szorítkozunk, ezen belül is ennek aszimptotikus vizsgálatára. Ezt jól jellemzi az irodalomban szokásos módon használt *aszimptotikus legrosszabb–eset hányados*, vagy másképpen *aszimptotikus versenyképességi hányados, AVK*. Ez a következő módon definiálható egy A algoritmusra. Egy adott L lista esetén jelöljük rendre $A(L)$-lel és $OPT(L)$-lel az A algoritmus által használt, illetve egy optimális pakolás által használt ládák számát. Ekkor

$$AVK(A) := \limsup_{l \to \infty} \left\{ \max_{L} \left\{ \frac{A(L)}{l} \,\middle|\, OPT(L) = l \right\} \right\}, \tag{31}$$

$AVK(A)$ az A algoritmus aszimptotikus versenyképességi hányadosa.

Az algoritmusok egy osztályát alkotják az *off-line algoritmusok*. Az off-line algoritmusok teljes információval rendelkeznek a listáról, ezt a stratégiájuk kialakításánál figyelembe is vehetik. Sokszor rendezik a listát (legtöbbször az elemek mérete szerint csökkenő sorrendbe), majd ezután valamilyen on-line technikát alkalmaznak. Off-line típusú algoritmusokra a legjobb eredményt [29] és [62] szerzői bizonyították: hogy tetszőleges $\varepsilon > 0$-hoz, vannak olyan lineáris idejű algoritmusok, amelyek AVK-ja $1 + \varepsilon$.

Az *on-line algoritmusoknak* az elemeket érkezésük sorrendjében kell elpakolnia, úgy, hogy semmilyen információval nem rendelkezik a lista további elemeiről: sem azok számáról (így arról sem, hogy érkezik-e még egyáltalán elem), sem a később érkező elemek méretéről. Minden egyes tárgy érkezésekor azonnal dönteni kell annak végleges elhelyezéséről. Az algoritmus dönthet úgy, hogy egy olyan ládába rakja, amelyikben már vannak elemek (már megnyitott a láda) és ebbe az érkező tárgy még „belefér", vagy új ládát nyit és abban helyezi el. Az egyszer elpakolt elemeket többet nem mozgathatja az algoritmus.

A jelenleg ismert legjobb (AVK-jú) on-line algoritmust, a Harmonic++ nevűt az ezredforduló után publikálta Seiden [95]. Seiden bizonyította, hogy ennek AVK-ja legfeljebb $1,58889$. Mivel a Harmonic++ algoritmus a Seiden által ugyanebben a cikkben definiált Super Harmonic algoritmusosztályba tartozik, és ezen osztály minden algoritmusának aszimptotikus versenyképességére érvényes a [87]-ben bizonyított $1,58333$ alsó korlát, így $1,58333 \leq AVK(Harmonic{+}{+}) \leq 1,58889$. Megjegyezzük, hogy 1985 óta minden, aktuálisan legjobb algoritmus Super Harmonic típusú [18, 25]. Az on-line feladatra, azaz bármely on-line algoritmus aszimptotikus versenyképességére ismert legjobb alsó korlát $1,5401$, amelyet van Vliet bizonyított [109]. Mindezek azt is jelentik, hogy szakítani kell az utóbbi húsz évben használt Super Harmonic technikával, ha $1,58333$-nál versenyképesebb on-line algoritmust szeretnénk adni. Nyitott kérdés, hogy egyáltalán lehet-e ilyet adni. Egy másik lehetőség, amelyre $1,5401$-nél versenyképesebb algoritmus megadásához már biztosan szükség van, lazítani az on-line feladat előírásán.

Az úgynevezett *szemi-on-line algoritmusok* az on-line és az off-line algoritmusok között helyezkednek el [18]. Az on-line algoritmusokhoz hasonlóan ezek sem rendelkeznek minden információval a listáról. Az on-line feladat definíciójától eltérően azonban valamilyen részleges információval igen. A szemi-on-line algoritmusoknál a következő műveletek legalább egyike megengedett pluszban az on-line esethez képest:

- elemek átpakolása [34, 36, 58, 59],
- néhány következő elem megvizsgálása („előrenézése", [44, 45]),
- vagy néhány elem (elő)rendezése [33].

Ezért tekinthetünk egy ilyen feladatot az on-line feladat egy meglazításának.

Az általunk vizsgált ún. c-átpakolásos szemi-on-line ládapakolási feladat az on-line feladattól annyiban különbözik, hogy minden tárgy érkezésekor a korábban elpakolt tárgyak közül legfeljebb c darab tárgy újrapakolása (már nyitott ládák közötti mozgatása vagy új láda nyitása és abba áthelyezése) is megengedett a pakoló, azaz az algoritmus számára.

Időrendi sorrendben az első szemi-on-line ládapakolási algoritmust Galambos adta meg [33] az on-line ládapakolási feladat azon megszorítására, amikor csak korlátos számú láda lehet egyszerre nyitva (vagy más szóval *aktív*, a szintén az irodalomban szokásosan használt terminológiában). Egy láda lezárt, ha már nem pakolhatunk bele később. Ez az algoritmus két bufferládát használ az elemek átmeneti tárolására. Ezt javítva Galambos és Woeginger [34] definiált egy 3 bufferládát és szintén átpakolást használó, szemi-on-line algoritmust. Ennek AVK-ja $1,69103...$, amely bizonyítottan optimális a korlátos számú nyitott ládát használó ládapakolási algoritmusok között.

Néhány évvel később Gambosi és szerzőtársai [35, 36] visszatértek bizonyos, szemi-on-line típusú algoritmusok elemzéséhez. Az általuk megadott algoritmusokban átpakolást használtak, de ennek alkalmazását és költségét speciális módon definiálták. Az általuk adott algoritmus az „elég nagy" elemeket egyesével pakolja át (mozgatja ládák között), míg a „kis" elemekből csoportokat, kötegeket képez. Ezután egy-egy köteget egyszerre, egy lépésben pakolhatja át, és egy, ilyen köteg ládák közötti mozgatása is egységnyi költségűnek definiált. Azaz egy ilyen mozgatás, ugyanúgy mint egy nagy elem átpakolása, szintén 1 átpakolásnak számít. Ebben az értelemben ezek az algoritmusok egy lépésben akár $O(n)$ darab elemet is mozgathatnak. A [36] cikk szerzői két algoritmust elemeztek. Ezek közül a gyorsabb, lineáris idejű A_1 algoritmus aszimptotikus versenyképességére $\frac{3}{2}$-es, míg a másik $O(n \log n)$ futási idejű A_2 algoritmuséra $\frac{4}{3}$-os felső korlátot bizonyítottak.

Kapcsolat állítható fel a ládapakolás, valamint a számítógépes programok (jobok) memóriaterület-foglalásának ütemezési feladata között. Egy job (byte-okban adott) mérete egy elem méretének felel meg, míg egy memória partíció mérete megfelel a láda kapacitásának. A legfontosabb különbség azonban, hogy egy jobhoz hozzárendelt memóriaterület a job futásának befejeztekor felszabadítható a memóriában. Ezt is modellezve, a ládapakolási algoritmusok új osztálya adható meg, a *dinamikus ládapakolási feladatok*é, ahol a listabeli elemekre *„Törlés"* művelet is előírható, azaz az elemek távozása is megengedett, azok érkezése (*Beszúrás* művelet) mellett. Ez az előírás az input része, azaz egy input lista nem más, mint Beszúrás és Törlés műveletek véges sorozata; azaz amíg el nem érjük a lista végét, addig a következő input művelet (lépés) mindig vagy egy új elem beszúrása, vagy egy korábban beszúrt (érkezett) és még nem törölt elem törlésének egyike lehet. Hangsúlyozzuk, hogy törlés csak az input része lehet, azaz az algoritmus nem törölhet elemet ennél a feladatnál sem, csak az input szolgáltató – nevezzük *Ütemező*nek ezt a szereplőt – teheti ezt meg. Egy *A dinamikus ládapakolási algoritmus* által felhasznált ládák számát úgy definiáljuk, mint az *összes lépés során felhasznált (nemüres) ládák számának a maximuma*. $A(L)$-t ily módon definiálva, akkor ezután már egy A dinamikus ládapakolási algoritmus versenyképességi hányadosa, $AVK(A)$ is a korábbiakhoz teljesen hasonló módon, szintén a (31) formulával definiálható. Könnyen látható, hogy az eredeti ládapakolási feladat a dinamikus ládapakolási feladat azon speciális esete, amikor a lista csak Beszúrás műveletekből áll, az itteni terminológiában fogalmazva. A dinamikus ládapakolási feladatot Coffman, Garey és Johnson definiálták és vizsgálták is [19]-ben, a feladatra közelítő algoritmust adva és többféleképpen elemezve azt.

Ivkovič és Lloyd a ládapakolási feladatok fentebbi két változatát kombinálták. A dinamikus ládapakolási feladatban az algoritmus számára átpakolás használatát megengedve definiálták a *teljesen dinamikus ládapakolási feladatot (fully-dynamic bin packing, FDBP)*. Hasonlóan a [36]-ban használt technikához, az általuk az így kapott problémára adott algoritmusuk szintén felhasználta a kis elemek „kötegelésének" technikáját. Az általuk elért aszimptotikus versenyképességi hányados $\frac{5}{4}$ volt.

Az utóbbi évtizedig nem publikáltak a triviálison kívüli alsó korlátokat szemi-on-line algoritmusok hatékonyságára. Az ilyen szempontból első alsó korlátot adó dolgozatot szintén Ivkovič és Lloyd jegyzi [58]. A teljesen dinamikus ládapakolási feladat azon változatára bizonyítottak alsó korlátot, amikor az átpakolás lehetőségét úgy szorítjuk meg, hogy lépésenként itt is csak konstans számú elem pakolható át. Nevezzük az így kapott feladatot *c-átpakolásos teljesen dinamikus ládapakolási feladatnak* (*c-átpakolásos FDBP feladat*). Bizonyították, hogy nincs olyan c-átpakolásos FDBP algoritmus, amelynek aszimptotikus versenyképességi hányadosa jobb lenne, mint $\frac{4}{3}$. A konstrukció kis módosítással átvihető, és az adott $\frac{4}{3}$-os alsó korlát érvényes a c-átpakolásos szemi-on-line feladatra is, ahogy ez megtalálható Csirik és Woeginger [25] tanulmányában is. Fontos megjegyezni, hogy a korlát minden c-re, és mindkét feladatra érvényes.

Gutin és szerzőtársai 2005-ben megjelent cikkükben [45] adtak alsó korlátot az ott bevezetett *batched ládapakolási feladatra* (*BBP*, a probléma angol *batched bin packing* elnevezésének rövidítéséből). Ez, a szintén a szemi-on-line ládapakolási feladatok körébe sorolható feladat a klasszikus ládapakolási feladat azon módosítása, amikor az input lista részekre, úgynevezett batchekre osztott. Minden batch csak az előző batch feldolgozása után elérhető. Egy-egy batch elpakolása önmagában történhet *off-line* módon, azonban egy-egy batch elpakolása után a batch elemei már nem mozgathatók a további batchek feldolgozása során Ha minden batch pontosan egy elemből áll, akkor a probléma speciális eseteként visszakapjuk a klasszikus on-line ládapakolási feladatot. Ha az input pontosan m batch-ből áll, akkor a BBP feladat ezen megszorítását *m-BBP feladatnak* nevezzük. Az ilyen típusú feladatokra adott algoritmusok aszimptotikus versenyképességi hányadosa hasonló módon definiálható, mint a fentiekben.

A [45] cikkben a szerzők a 2-BBP problémát tárgyalták részletesebben. A 2-BBP problémára itt $1,3871\ldots$-es alsó korlátot bizonyítottak, továbbá korlátokat adtak a probléma azon speciális eseteire, amikor a listában szereplő *különböző méretű* elemek száma felülről korlátozott egy rögzített $p \geq 2$ egész számmal. Bizonyították egy másik, [46]-beli eredményüket felhasználva a szerzők, hogy a $p = 2$ esetre megadott korlátjuk optimális. Nyitott kérdésként felvetették, hogy az ettől különböző p értékekre megadott korlátjaik optimálisak-e.

A fejezet szervezése és eredményei

A 4.2. alfejezetben először alsó korlátot adunk a a c-átpakolásos szemi-on-line feladatra. Az itt nyert $1,3871$-es alsó korláttal javítjuk Ivkovič és Lloyd 1996-ban publikált $\frac{4}{3}$-os alsó korlátját. Ezt a c-átpakolásos szemi-on-line feladata bizonyítjuk, azonban ez érvényes a c-átpakolásos teljesen dinamikus ládapakolási feladatra is. A modell felírása után eredményünket az adódó speciális nemlineáris optimalizálási probléma elemzése és megoldása révén nyerjük. Ennek megoldását egy szélsőérték feladat megoldása után a Lambert–féle W függvény segítségével fejezzük ki pontosan. Alsó korlát konstrukciónk a [45]-beli és az [58]-beli modellek általánosításának is tekinthető.

A 4.3. alfejezetben a probléma azon speciális esetével foglalkozunk, amikor a feladatban – a [45]-belihez hasonlóan – még egy feltételt kirovunk: a megengedett különböző elemméretek maximális számát is korlátozzuk, egy előre rögzített p ($p \geq 2$) konstanssal. Az itt nyert alsó korlátok szintén érvényesek mindhárom, fentebb említett szemi-on-line feladat, azaz a c-átpakolásos szemi-on-line, a c-átpakolásos teljesen dinamikus és a 2–BBP feladat ezen speciális esetére. A $p \geq 3$ esetekre bizonyítjuk, hogy a 2-BBP feladatra Gutin és szerzőtársai által [45]-ben megadott korlátok javíthatók, negatívan megválaszolva ezzel az ott felvetett, ezen korlátaik optimalitására vonatkozó kérdést. A bizonyítás globális optimalizálási eszközökkel történik. Konkrét, $p \geq 3$ értékekre megbízható, intervallum-matematikán alapuló, Branch-and-Bound technikát használó módszerrel adódik a megoldás. A megbízhatóság azt jelenti, hogy bár egy ilyen megoldást számítógépes eszközök használatával kapunk, az eredmény mégis matematikai szigorral ellenőrzött, garantált megoldás, azaz kiküszöböli a kerekítési hibákból (vagy azok felhalmozódásából) származó esetleges pontatlanságokat is.

A 4.2. és a 4.3. alfejezetekben adott alsó korlátok érdekessége, hogy habár a feladat tisztán diszkrét optimalizálási, a megoldáshoz speciális, folytonos nemlineáris optimalizálási problémák megoldása szükséges, és ehhez globális optimalizálási eszközöket használunk. Bizonyítjuk, hogy bár korlátainkat egy dimenzióban fogalmaztuk meg, a konstrukció és az eredmények több dimenzióban is érvényesek.

A fejezet záró, 4.4. alfejezetében felső korlátokat adunk a c-átpakolásos szemi-on-line ládapakolási feladatra, amelynél már „tisztán" diszkrét optimalizálási, azaz a ládapakolás irodalmában klasszikusnak számító algoritmus-elemzési eszközöket (pl. súlyfüggvény-technikát [108]) alkalmazunk. A megadott felső korlátok bizonyítják, hogy az on-line feladat meglazításával kapott szemi-on-line feladatban jól kihasználható a lépésenként c elem átpakolhatóságának lehetősége. Bizonyítjuk, hogy már $c = 2$-re adható az aktuálisan legjobb on-line algoritmusnál versenyképesebb, 2-átpakolásos szemi-on-line ládapakolási algoritmus. Kiemelendő a $c = 4$ esetre adott algoritmusunk is, ez az általunk megadott algoritmussorozatban az első olyan, amelynek az aszimptotikus versenyképességi hányadosa az on-line algoritmusokra bizonyított legjobb alsó korlát [109] értéke alatt van. Ez azt is jelenti, hogy biztosan versenyképesebb minden on-line algoritmusnál – nemcsak minden ma ismert on-line algoritmusnál, hanem minden, a jövőben tervezettnél is. Ismét utalunk arra, hogy ez természetesen csak úgy lehetséges, hogy az általunk adott algoritmusok nem on-line, hanem szemi-on-line algoritmusok, és a feladat meglazítása által rendelkezésre álló átpakolási lehetőség már kis c értékeknél is jól kihasználható. Ez egyben a feladat definiálásának létjogosultságát is mutatja, azaz azt, hogy érdemes ilyen feladatokat vizsgálni.

4.2. Alsó korlátok

Ebben a fejezetben az [58]-beli, viszonylag egyszerűen adódó $\frac{4}{3}$-os alsó korlátot javítjuk $1,3871$-re a c-átpakolásos szemi-on-line ládapakolási feladatra. Az elemzés során különböző eszköztárakból származó technikákat használunk: lineáris programozási eredményeket fogunk kombinálni lineáris algebrai eszközökkel, végezetül egy nemlineáris optimalizálási feladatot oldunk meg.

A kapcsolódó lineáris programozási feladat konstrukciója

$k \geq 1$, $n \geq 1$ egészek esetén legyenek $x_1, x_2, \ldots, x_k$ $\left(\frac{1}{2} \leq x_1 < x_2 < \ldots < x_k < 1\right)$ rögzített valós számok, és $c \geq 1$ tetszőleges egész. Definíció szerint legyen $y_i = 1-x_i \, (i = 1, \ldots, k)$ és $y_{k+1} = 0$. Legyen továbbá $\varepsilon < \min\limits_{j=1,\ldots,k} \{y_j - y_{j+1}\}$ egy tetszőleges pozitív szám, és $\varepsilon_j := \frac{\varepsilon}{\lceil \frac{n}{2y_j} \rceil}$.

Jelöljük az első listát L_0-lal. Definiáljuk L_0-t N darab a méretű elem listájaként, ahol $a < \varepsilon_k/(\lceil \frac{n}{y_k} \rceil c)$, és $N := \lfloor \frac{\frac{n}{2}-\varepsilon}{a} \rfloor$. Az L_0 listabeli kis elemek összmérete $\frac{n}{2}$-höz tart alulról, ha $n \to \infty$. Jelölje L_i $(1 \leq i \leq k)$ a nagy (legalább $\frac{1}{2}$ méretű) elemeket tartalmazó listákat. Definiáljuk L_i-t $\lceil \frac{n}{2y_i} \rceil$ darab $x_i + \varepsilon_i$ méretű tárgy listájaként. Ez azt jelenti, hogy egy-egy listán belül, az L_0 és az L_i listákban $(i = 1, \ldots, k)$ is egyenlő méretűek az elemek. Ezáltal a konstrukciónkban $k + 1$ a különböző elemméretek száma. (A 4.3. alfejezetben a $p = k + 1$ beállítással fogunk vizsgálódni.)

Vegyük észre, hogy L_0 célunknak megfelelően, alkalmas módon elegendően kicsire választott méretű elemeket tartalmaz. Egy L_i lista érkezése alatt, $\lceil \frac{n}{2y_i} \rceil$ $(i = 1, \ldots, k)$ lépésben átpakolható elemek száma $c\lceil \frac{n}{2y_i} \rceil$. Az L_0 listabeli elemek fenti a méretét éppen ezért úgy definiáltuk, hogy L_i feldolgozása alatt az átpakolható kis elemek összmérete kisebb, mint ε_k legyen, amely a legkisebb érték az $\varepsilon_1, \ldots, \varepsilon_k$ értékek közül. Ez az észrevétel fontos szerepet fog játszani a bizonyítás során.

Tegyük fel, hogy c-átpakolásos szemi-on-line ládapakolási algoritmust alkalmazunk a listák elpakolására. Megvizsgáljuk, hogy egy tetszőleges, c-átpakolásos szemi-on-line ládapakolási algoritmus hogyan pakolja el az L_0 és L_0L_i $(i = 1,\ldots,k)$ listákat. Az alsó korlátot egy lineáris programozási feladat megoldásával adjuk meg.

2. Lemma.

$$\limsup_{n\to\infty}\frac{A\,(L_0)}{OPT\,(L_0)} \geq 1+\sum_{j=1}^{k}2z_j\left(\frac{1}{y_j}-1\right).$$

BIZONYÍTÁS. Tekintsünk egy tetszőleges A c-átpakolásos szemi-on-line algoritmust, amelynek először az L_0 listát kell elpakolnia. A célunk az, hogy $A\,(L_0)$-ra alsó, $OPT(L_0)$-ra felső korlátot adjunk. Először becsüljük $A\,(L_0)$-t alulról!

Elemzésünkben $szint(B)$ jelöli egy B ládabeli elemek méreteinek összegét $0 \leq szint(B) \leq 1$. Egy B ládát y_i-típusúnak nevezünk, ha $szint(B) \in (y_{i+1}, y_i]$. Vezessük be továbbá a z_i jelölést; z_in azon elemek méreteinek összege, amelyek y_i-típusú ládákba lettek elpakolva $(i = 1,\ldots,k)$. Ennek segítségével becslést adhatunk minden i-re az y_i-típusú ládák (összes) számára; könnyen látható, hogy ezek száma legalább $\frac{z_in}{y_i}$. Az olyan elemek méreteinek összege, amelyek egyetlen y_i-típusú ládába sincsenek elhelyezve, $Na - n\sum_{j=1}^{k} z_j$. Ezek alapján a kívánt becslés a következő módon adható meg:

$$A\,(L_0) \geq \sum_{j=1}^{k}\frac{z_jn}{y_j} + Na - n\sum_{j=1}^{k}z_j = Na + n\sum_{j=1}^{k}z_j\left(\frac{1}{y_j}-1\right).$$

N definíciója miatt tudjuk, hogy $Na \geq \frac{n}{2} - \varepsilon - a$. Ez azt jelenti, hogy

$$A\,(L_0) \geq n\left(\frac{1}{2}+\sum_{j=1}^{k}z_j\left(\frac{1}{y_j}-1\right)\right) - \varepsilon - a. \tag{32}$$

A következőkben tanulmányozzuk az L_0 lista optimális pakolását! L_0 optimális pakolásában minden láda nagyobb, mint $1-a$ szintig megtöltött ($\forall B$ ládára $szint(B) > 1-a$). (Egyébként bele lehetne tenni még egy elemet.) Ez alapján

$$OPT\,(L_0) \leq \frac{Na}{1-a}+1 \leq \frac{\frac{n}{2}-\varepsilon}{1-a}+1 \leq \frac{\frac{n}{2}}{1-a}+1. \tag{33}$$

(32)-ből és (33)-ból következően

$$\limsup_{n\to\infty}\frac{A\,(L_0)}{OPT\,(L_0)} \geq 1+\sum_{j=1}^{k}2z_j\left(\frac{1}{y_j}-1\right).$$

□

3. Lemma. *Minden* $i = 1, \ldots, k$*-ra:*

$$\limsup_{n\to\infty} \frac{A\left(L_0 L_i\right)}{OPT\left(L_0 L_i\right)} \geq 1 + y_i + 2y_i \left(\sum_{j=1}^{i-1} z_j \left(\frac{1}{y_j} - 1 \right) - \sum_{j=i}^{k} z_j \right).$$

BIZONYÍTÁS. Legyen $i \in \{1, \ldots, k\}$ tetszőleges. Tekintsük most az $L_0 L_i$ konkatenált listának az A általi elpakolását! Legyen B egy olyan láda, amely tartalmaz L_i-beli elemet. Mivel ezen nagy elem mérete $x_i + \varepsilon_i$, így a B ládában lévő kis elemek összmérete legfeljebb $y_i - \varepsilon_i$. Persze, egy ilyen láda úgy is kialakulhatott az $L_0 L_i$ lista elpakolása során, hogy L_0 pakolásának befejeztekor a láda még több, akár $y_i - \varepsilon_i$-nél nagyobb összmennyiségű elemet tartalmazott (azaz egy L_i-beli nagy elem így nem is férne bele). L_i érkezése alatt ugyanis, használva az átpakolás lehetőségét, az algoritmus elemeket pakolhat át ebből a ládából más ládákba úgy, hogy ezután már bele tud rakni a B ládába egy L_i-beli nagy elemet.

Miután az L_i lista pontosan $\lceil \frac{n}{2y_i} \rceil$ darab elemet tartalmaz, így L_i feldolgozása alatt összesen legfeljebb $c\lceil \frac{n}{2y_i} \rceil$ darab elemet átpakolhat az algoritmus. Mindegyik kis elem mérete egységesen a, így a és ε_i definíciójából adódik, hogy azon kis elemek méreteinek összege, amelyeket A ez alatt átpakolhat az legfeljebb ε_i. Így L_0 elpakolása után (pontosabban annak befejeztekor: azaz L_0 utolsó elemének feldolgozása után, de L_i első elemének érkezése előtt) B magasságára a $szint(B) \leq y_i$ egyenlőtlenségnek kellett teljesülnie.

Ebből következően az olyan kis elemek méreteinek összmennyisége, amelyek nagy, L_i-beli elemek ládáiban vannak az egész $L_0 L_i$ lista elpakolása után, legfeljebb $\sum_{j=i}^{k} z_j n$. Más, L_0-beli elemek lehetnek egyrészt y_j-típusú ládákban, ahol $j = 1, \ldots, i-1$. Másrészt a kimaradó (sem a nagy, L_i-beli elemek ládáiban, sem a kis, L_0-beli elemek y_j-típusú ládáiban nem lévő) L_0-beli elemek méreteinek összege legalább $Na - n\sum_{j=1}^{k} z_j$. Így ezek alapján a következő módon becsülhetjük alulról az A által felhasznált ládák számát:

$$A\left(L_0 L_i\right) \geq \frac{n}{2y_i} + n\sum_{j=1}^{i-1} \frac{z_j}{y_j} + Na - n\sum_{j=1}^{k} z_j, \ \ (i = 1, \ldots, k)\,. \tag{34}$$

$L_0 L_i$ $(1 \leq i \leq k)$ egy optimális pakolása által használt ládák számát megvizsgálva kapjuk, hogy

$$OPT\left(L_0 L_i\right) \leq \left\lceil \frac{n}{2y_i} \right\rceil + S_i \leq \frac{n}{2y_i} + S_i + 1, \tag{35}$$

ahol S_i az optimális pakolás azon ládáinak száma, amelyek csak L_0-beli elemet tartalmaznak. Az L_i-beli elemet tartalmazó ládákban lévő kis elemek méreteinek összmennyiségét $kis(B_i)$-vel jelölve, az a következőképpen becsülhető:

$$kis(B_i) \ \ \geq \ \ \left\lceil \frac{n}{2y_i} \right\rceil (y_i - \varepsilon_i - a) \geq \frac{n}{2} - \left\lceil \frac{n}{2y_i} \right\rceil \varepsilon_i - \left\lceil \frac{n}{2y_i} \right\rceil a > Na - 2\varepsilon,$$

mivel $\left\lceil \frac{n}{2y_i} \right\rceil \varepsilon_i = \varepsilon$ és $a < \varepsilon / \lceil \frac{n}{2y_i} \rceil$. Ezért a csak L_0-beli kis elemeket tartalmazó S_i darab ládában lévő elemek összmennyisége kevesebb, mint 2ε, így

$$S_i \leq 1. \tag{36}$$

(35) és (36) alapján

$$OPT\left(L_0 L_i\right) \leq \frac{n}{2y_i} + 2. \tag{37}$$

Együtt tekintve (34)-et és (37)-et azt kapjuk, hogy

$$\limsup_{n\to\infty} \frac{A\left(L_0 L_i\right)}{OPT\left(L_0 L_i\right)} \geq 1 + y_i + 2y_i \left(\sum_{j=1}^{i-1} z_j \left(\frac{1}{y_j} - 1 \right) - \sum_{j=i}^{k} z_j \right), \quad (i = 1, \ldots, k).$$

□

2. Tétel. *A következő lineáris programozási feladat megoldása alsó korlátot szolgáltat bármely c-átpakolásos szemi-on-line ládapakolási algoritmus AVK-jára:*

$$\begin{aligned}
&\min b, \\
b &\geq 1 + y_i + 2y_i \left(\sum_{j=1}^{i-1} z_j \left(\frac{1}{y_j} - 1 \right) - \sum_{j=i}^{k} z_j \right), \quad (i = 1, \ldots, k), \\
b &\geq 1 + \sum_{j=1}^{k} 2z_j \left(\frac{1}{y_j} - 1 \right), \\
z_i &\geq 0, (i = 1, \ldots, k), \\
\sum_{j=1}^{k} z_j &< \frac{1}{2}.
\end{aligned} \tag{38}$$

BIZONYÍTÁS. z_j $(j = 1, \ldots, k)$ definíciójából közvetlenül adódik a $\sum_{j=1}^{k} z_j < \frac{1}{2}$ korlátozó feltétel. A tétel állítása ezután a 2. és 3. Lemmákból következik. □

A lineáris programozási feladat megoldása

Modellünk egyszerűsítésére bevezetünk egy új, z változót, amely a z_j-k összege, azaz $z = \sum_{j=1}^{k} z_j$. Ezt behelyettesítve (38)-ba, és átrendezve az egyenleteket, (38) a következő alakban írható föl:

$$\begin{aligned}
&\min\ b, \\
-2y_i z + 2y_i \sum_{j=1}^{i-1} \frac{z_j}{y_j} - b &\leq -\left(1 + y_i\right), \quad (i = 1, \ldots, k), \\
-2z + 2\sum_{j=1}^{k} \frac{z_j}{y_j} - b &\leq -1, \\
-z + \sum_{j=1}^{k} z_j &= 0, \\
z_i &\geq 0, \quad (i = 1, \ldots, k), \\
z &< \frac{1}{2}.
\end{aligned} \tag{39}$$

(39) helyett először egy, a problémánkhoz kapcsolódó lineáris egyenletrendszert oldunk meg. Az egyenletek száma ebben $k+2$, az egyenletrendszer változói pedig $z, z_1, \ldots, z_k, b$, míg $y_1, \ldots, y_k$ a fentebb előírt módon rögzített paraméterek, amelyek kielégítik a 2. Tétel korlátozó feltételeit.

$$\begin{aligned} -2y_i z + 2y_i \sum_{j=1}^{i-1} \frac{z_j}{y_j} - b &= -(1+y_i), \ \ (i=1,\ldots,k), \\ -2z + 2\sum_{j=1}^{k} \frac{z_j}{y_j} - b &= -1, \\ -z + \sum_{j=1}^{k} z_j &= 0. \end{aligned} \tag{40}$$

4. Lemma. *A (40) egyenletrendszernek létezik megoldása, és az egyértelmű.*

BIZONYÍTÁS. Legyen M a (40) mátrixa, ekkor

$$M = \begin{bmatrix} -2y_1 & 0 & \ldots & 0 & 0 & -1 \\ -2y_2 & 2\frac{y_2}{y_1} & \ldots & 0 & 0 & -1 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -2y_k & 2\frac{y_k}{y_1} & \ldots & 2\frac{y_k}{y_{k-1}} & 0 & -1 \\ -2 & \frac{2}{y_1} & \ldots & \frac{2}{y_{k-1}} & \frac{2}{y_k} & -1 \\ -1 & 1 & \ldots & 1 & 1 & 0 \end{bmatrix}. \tag{41}$$

Jelölje $\det(M)$ a szokásos módon az M mátrix determinánsát. Belátjuk, hogy $\det(M) < 0$, és így a lemma állítása következik a Cramer-szabályból. Fejtsük ki $\det(M)$-et M utolsó oszlopa szerint. Azt kapjuk, hogy

$$\det(M) = \sum_{i=1}^{k+1} -1\,(-1)^{k+2+i} \det(M_i) = \sum_{i=1}^{k+1} (-1)^{k+1+i} \det(M_i), \tag{42}$$

ahol M_i egy $(k+1) \times (k+1)$ méretű mátrix, amelyet M-ből, annak utolsó oszlopának és i. sorának törlésével kapunk. Fejtsük ki $\det(M_i)$-t M_i utolsó sora szerint:

$$\det(M_i) = (-1)(-1)^{k+2} \det(M_{i1}) + \sum_{j=2}^{k+1} (-1)^{k+1+j} \det(M_{ij}), \tag{43}$$

ahol M_{ij} egy $k \times k$ méretű mátrix, amelyet M_i-ből nyerünk, törölve ennek utolsó sorát és j. oszlopát. Most vegyük szemügyre M_{ij}-t. Ennek alakja szerint folytatjuk vizsgálatainkat.

$$M_{ij}=\begin{bmatrix} -2y_1 & 0 & \dots & 0 & 0 & \dots & 0 & 0\\ -2y_2 & 2\frac{y_2}{y_1} & \dots & 0 & 0 & \dots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots\\ -2y_{i-1} & 2\frac{y_{i-1}}{y_1} & \dots & 2\frac{y_{i-1}}{y_{j-2}} & 2\frac{y_{i-1}}{y_j} & \dots & 0 & 0\\ -2y_{i+1} & 2\frac{y_{i+1}}{y_1} & \dots & 2\frac{y_{i+1}}{y_{j-2}} & 2\frac{y_{i+1}}{y_j} & \dots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots\\ -2y_k & 2\frac{y_k}{y_1} & \dots & 2\frac{y_k}{y_{j-2}} & 2\frac{y_k}{y_j} & \dots & 2\frac{y_k}{y_{k-1}} & 0\\ -2 & \frac{2}{y_1} & \dots & \frac{2}{y_{j-2}} & \frac{2}{y_j} & \dots & \frac{2}{y_{k-1}} & \frac{2}{y_k} \end{bmatrix}.$$

A. eset: Tegyük fel, hogy $j \neq i$ és $j \neq i+1$, valamint tekintsük kizárólag az M_{ij} $i.$ és $i+1.$ oszlopait. A két oszlop a következő:

$$\begin{bmatrix} 0 & 0\\ \vdots & \vdots\\ 0 & 0\\ 2\frac{y_{i+1}}{y_{i-1}} & 2\frac{y_{i+1}}{y_i}\\ \vdots & \vdots\\ 2\frac{y_k}{y_{i-1}} & 2\frac{y_k}{y_i}\\ \frac{2}{y_{i-1}} & \frac{2}{y_i} \end{bmatrix}$$

Könnyű észrevenni, hogy ezek az oszlopok nem függetlenek, így $\det(M_{ij})=0$, ha $j \neq i$ és $j \neq i+1$.

B. eset: Ha $j=i$, akkor M_{ii} egy alsó trianguláris mátrix:

$$M_{ii}=\begin{bmatrix} -2y_1 & 0 & \dots & 0 & 0 & \dots & 0 & 0\\ -2y_2 & 2\frac{y_2}{y_1} & \dots & 0 & 0 & \dots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots\\ -2y_{i-1} & 2\frac{y_{i-1}}{y_1} & \dots & 2\frac{y_{i-1}}{y_{i-2}} & 0 & \dots & 0 & 0\\ -2y_{i+1} & 2\frac{y_{i+1}}{y_1} & \dots & 2\frac{y_{i+1}}{y_{i-2}} & 2\frac{y_{i+1}}{y_i} & \ddots & 0 & 0\\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots\\ -2y_k & 2\frac{y_k}{y_1} & \dots & 2\frac{y_k}{y_{i-2}} & 2\frac{y_k}{y_i} & \dots & 2\frac{y_k}{y_{k-1}} & 0\\ -2 & \frac{2}{y_1} & \dots & \frac{2}{y_{i-2}} & \frac{2}{y_i} & \dots & \frac{2}{y_{k-1}} & \frac{2}{y_k} \end{bmatrix}.$$

Ebből következően, $\det(M_{11})=2^k\frac{1}{y_1}$, $\det(M_{(k+1)(k+1)})=-2^k y_k$, és $2 \leq i \leq k$ esetén $\det(M_{ii})=-2^k\frac{y_{i-1}}{y_i}$.

C. eset: Hasonló módon, ha $j=i+1$, akkor $M_{i(i+1)}$ szintén egy alsó trianguláris mátrix:

$$M_{i(i+1)} = \begin{bmatrix} -2y_1 & 0 & \dots & 0 & 0 & \dots & 0 & 0 \\ -2y_2 & 2\frac{y_2}{y_1} & \dots & 0 & 0 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots & \vdots \\ -2y_{i-1} & 2\frac{y_{i-1}}{y_1} & \dots & 2\frac{y_{i-1}}{y_{i-2}} & 0 & \dots & 0 & 0 \\ -2y_{i+1} & 2\frac{y_{i+1}}{y_1} & \dots & 2\frac{y_{i+1}}{y_{i-2}} & 2\frac{y_{i+1}}{y_{i-1}} & \ddots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \ddots & \vdots \\ -2y_k & 2\frac{y_k}{y_1} & \dots & 2\frac{y_k}{y_{i-2}} & 2\frac{y_k}{y_{i-1}} & \dots & 2\frac{y_k}{y_{k-1}} & 0 \\ -2 & \frac{2}{y_1} & \dots & \frac{2}{y_{i-2}} & \frac{2}{y_{i-1}} & \dots & \frac{2}{y_{k-1}} & \frac{2}{y_k} \end{bmatrix}.$$

Könnyű látni, hogy $\det(M_{i(i+1)}) = -2^k$, ha $i \neq 1$, és $\det(M_{12}) = 2^k$.

Ezen eredményeket felhasználva, (43)-ból kapjuk, hogy

$$\begin{aligned} \det(M_1) &= -2^k \left((-1)^{k+2}\frac{1}{y_1} + (-1)^{k+3} \right), \\ \det(M_i) &= -2^k \left((-1)^{k+1+i}\,\frac{y_{i-1}}{y_i} + (-1)^{k+1+i+1} \right), \quad i = 2, \dots, k, \\ \det(M_{k+1}) &= -2^k y_k. \end{aligned}$$

Ezt (42)-be visszaírva nyerjük, hogy

$$\begin{aligned} \det(M) = &\ -2^k \left((-1)^{2(k+1)}\,\frac{1}{y_1} + (-1)^{2(k+1)+1} \right) + \\ + &\ -2^k \left(\sum_{i=2}^{k} \left((-1)^{2(k+1+i)}\,\frac{y_{i-1}}{y_i} + (-1)^{2(k+1+i)+1} \right) + (-1)^{2(k+1)}\,y_k \right) = \\ = &\ -2^k \left(\frac{1}{y_1} + \sum_{i=2}^{k} \frac{y_{i-1}}{y_i} + y_k - k \right). \end{aligned}$$

Miután feltételeink szerint $y_{i-1} > y_i$ $(i = 2, \dots, k)$ és $\frac{1}{2} \geq y_1$ teljesül, ez szolgáltatja, hogy $\frac{1}{y_1} + \sum_{i=2}^{k} \frac{y_{i-1}}{y_i} > k$, és így $\det(M) < 0$. Ezzel a lemma állítását igazoltuk. □

5. Lemma. *A (40) egyenletrendszer megoldása kielégíti a (39) lineáris programozási feladat feltételrendszerét, és*

$$b = 1 + \frac{1 - y_k}{\frac{1}{y_1} + \sum\limits_{i=2}^{k} \frac{y_{i-1}}{y_i} + y_k - k}.$$

BIZONYÍTÁS. Behelyettesítve (40) jobb oldalát M-be, a következő M_b mátrixot kapjuk:

$$M_b = \begin{bmatrix} -2y_1 & 0 & \dots & 0 & 0 & -1-y_1 \\ -2y_2 & 2\frac{y_2}{y_1} & \dots & 0 & 0 & -1-y_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -2y_k & 2\frac{y_k}{y_1} & \dots & 2\frac{y_k}{y_{k-1}} & 0 & -1-y_k \\ -2 & \frac{2}{y_1} & \dots & \frac{2}{y_{k-1}} & \frac{2}{y_k} & -1 \\ -1 & 1 & \dots & 1 & 1 & 0 \end{bmatrix},$$

és

$$\det(M_b) = \det(M) + \sum_{i=1}^{k} y_i \, (-1)^{k+1+i} \det(M_i) = \det(M) - 2^k \, (1-y_k) \, .$$

Ismét a Cramer-szabályt alkalmazva:

$$b = \frac{\det(M_b)}{\det(M)} = 1 + \frac{1-y_k}{\frac{1}{y_1} + \sum\limits_{i=2}^{k} \frac{y_{i-1}}{y_i} + y_k - k}.$$

Hasonló elemzéssel belátható, hogy (39) többi feltétele szintén teljesül. Példaként csak a következő egyenlőtlenség fennállását igazoljuk:

$$z = \frac{\det(M_z)}{\det(M)} = \frac{\frac{1}{2}\det(M) - 2^{k-1}\frac{y_k-1}{y_1}}{\det(M)} = \frac{1}{2} + \frac{y_k - 1}{2y_1\left(\frac{1}{y_1} + \sum\limits_{i=2}^{k} \frac{y_{i-1}}{y_i} + y_k - k\right)} < \frac{1}{2},$$

ahol

$$M_z = \begin{bmatrix} -1-y_1 & 0 & \dots & 0 & 0 & -1-y_1 \\ -1-y_2 & 2\frac{y_2}{y_1} & \dots & 0 & 0 & -1-y_2 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ -1-y_k & 2\frac{y_k}{y_1} & \dots & 2\frac{y_k}{y_{k-1}} & 0 & -1-y_k \\ -1 & \frac{2}{y_1} & \dots & \frac{2}{y_{k-1}} & \frac{2}{y_k} & -1 \\ 0 & 1 & \dots & 1 & 1 & 0 \end{bmatrix}.$$

A többi egyenlőtlenség fennállása könnyen igazolható. □

6. Lemma. *A (40) egyenletrendszer megoldása egy optimális megoldása a (39) lineáris programozási feladatnak.*

BIZONYÍTÁS. Tekintsük a (40) megoldásvektorát, jelöljük ezt v-vel. Az 5. Lemma miatt ez egy lehetséges megoldása (39)-nek. Legyen v bázismegoldás. Mivel a célfüggvény ekkor nem más, mint b (amelynek nyilvánvalóan pozitív az együtthatója), így kielégíti a lineáris programozás jól ismert optimum kritériumának feltételeit. Ezzel a lemma állítása bizonyított. □

Az adódó alsó korlát meghatározása

A dolgozat ezen részében az $y_1, y_2, \ldots, y_k$ változók optimális megválasztásának kérdését tárgyaljuk. Ezek optimális megválasztásához a következő nemlineáris optimalizálási feladatot kell megoldanunk:

$$\max\left\{1+\frac{1-y_k}{\frac{1}{y_1}+\sum\limits_{i=2}^{k}\frac{y_{i-1}}{y_i}+y_k-k}\right\},$$
$$\frac{1}{2}\geq y_1>\ldots>y_k>0. \tag{44}$$

Ez a feladat fix k értékre analitikus módszerekkel nehezen megoldhatónak tűnik. Numerikusan azonban megoldható globális optimalizálási eszközökkel. Ezt (a fixált k értékekhez tartozó feladat megoldását) tárgyaljuk majd a következő, 4.3. részben. Ebben a részben a továbbiakban (44) aszimptotikus vizsgálatát végezzük el (azaz a $k \to \infty$ esetet elemezzük).

7. Lemma. *Ha $k \to \infty$, akkor a (44) feladat optimális megoldása az*

$$f(x) := 1+\frac{1-x}{x+1+\ln\left(\frac{1}{x}\right)-\ln(2)}$$

függvény $\left(0, \frac{1}{2}\right]$ intervallumbeli maximumához tart.

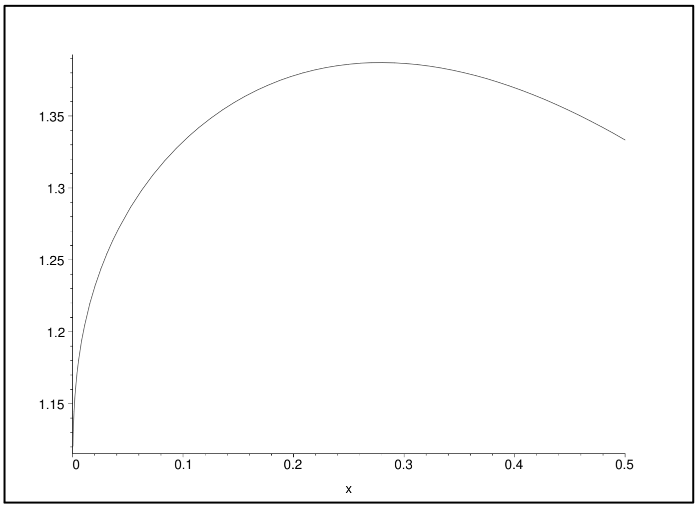
BIZONYÍTÁS. Két esetet különböztetünk meg:

1. eset: Ekkor azt tételezzük fel, hogy $y_2 \leq \frac{y_1}{2}$, azaz $2 \leq \frac{y_1}{y_2}$. Felhasználva, hogy $y_1 \leq \frac{1}{2}$, $y_2 \leq \frac{1}{4}$, $y_k \leq y_2$, $4 \geq 2 + y_k$, és alkalmazva a számtani és mértani közép közötti egyenlőtlenséget az $\frac{y_2}{y_3}, \frac{y_3}{y_4}, \ldots, \frac{y_{k-1}}{y_k}, y_k$ számokra (44) nevezőjében, a következő felső becslés adódik:

$$1+\frac{1-y_k}{\frac{1}{y_1}+\sum\limits_{i=2}^{k}\frac{y_{i-1}}{y_i}+y_k-k} \leq 1+\frac{1-y_k}{4+\sum\limits_{i=3}^{k}\frac{y_{i-1}}{y_i}+y_k-k} \leq$$
$$\leq 1+\frac{1-y_k}{2+y_k-k+(k-1)\sqrt[k-1]{y_k}}.$$

Mivel $\lim\limits_{k\to\infty}(k-1)\sqrt[k-1]{x}-k = \ln(x)-1$ és $\ln(x) < -\frac{5}{4}$, ha $x \leq \frac{1}{4}$, azt kapjuk, hogy:

$$\lim_{k\to\infty} 1+\frac{1-y_k}{\frac{1}{y_1}+\sum\limits_{i=2}^{k}\frac{y_{i-1}}{y_i}+y_k-k} \leq 1+\frac{1-y_k}{y_k+1+\ln(y_k)} < 1.$$



9. ábra. A 7. lemmabeli $f(x)$ függvény grafikonja a $\left(0, \frac{1}{2}\right]$ intervallumban.

2. eset: Tegyük fel, hogy $y_2 > \frac{y_1}{2}$. Ebből kapjuk, hogy $2y_2\,(1-2y_1) \geq y_1\,(1-2y_1)$, és így

$$\frac{1}{y_1} + \frac{y_1}{y_2} \geq 2 + \frac{1}{2y_2}. \tag{45}$$

Tekintve (45)-öt, az $y_1 \leq \frac{1}{2}$ egyenlőtlenséget, és alkalmazva ismét a számtani és mértani közepek közötti egyenlőtlenséget az $\frac{1}{2y_2}, \frac{y_2}{y_3}, \ldots, \frac{y_{k-1}}{y_k}$ számokra (44) nevezőjében, nyerjük az

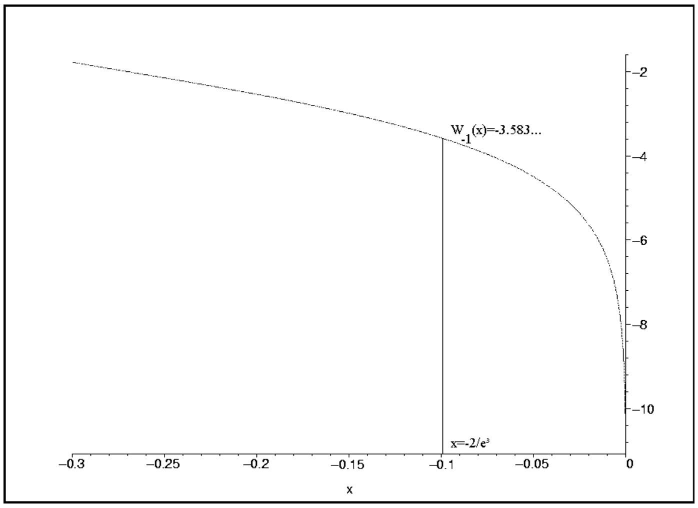
$$1 + \frac{1-y_k}{\frac{1}{y_1} + \sum\limits_{i=2}^{k} \frac{y_{i-1}}{y_i} + y_k - k} \leq 1 + \frac{1-y_k}{2 + \frac{1}{2y_2} + \sum\limits_{i=3}^{k} \frac{y_{i-1}}{y_i} + y_k - k} \leq$$
$$\leq 1 + \frac{1-y_k}{2 + y_k - k + (k-1)\sqrt[k-1]{\frac{1}{2y_k}}}.$$

felső becslést. Mivel $\lim\limits_{k\to\infty} (k-1)\sqrt[k-1]{\frac{1}{2x}} - k = \ln\left(\frac{1}{x}\right) - \ln(2) - 1,$ azt kapjuk, hogy

$$\lim_{k\to\infty} 1 + \frac{1-y_k}{\frac{1}{y_1} + \sum\limits_{i=2}^{k} \frac{y_{i-1}}{y_i} + y_k - k} \leq 1 + \frac{1-y_k}{y_k + 1 + \ln\left(\frac{1}{y_k}\right) - \ln(2)}.$$

Miután $(1-x)/(x+1+\ln(\frac{1}{x}) - \ln(2)) \geq 0,$ ha $x \in (0, \frac{1}{2}]$, ezért $f(x) \geq 1$ ezen az intervallumon (lásd 9. ábra). Így azt kapjuk, hogy a határérték maximumát a 2. eset szolgáltatja, ami igazolja lemmánk állítását.

Itt kihasználtuk azt, hogy ha kiszámítjuk $f(x)$ adott intervallumbeli maximumát, akkor konstruktívan megadhatók az $y_1, y_2, \ldots, y_k$ értékek, amelyek (44) optimális megoldását adják. Ez a konstrukció a következő: y_1-et beállítjuk $\frac{1}{2}$-re, míg a többi érték az $y_i = \frac{1}{2}(2y_k)^{\frac{i-1}{k-1}}$, $i = 2, \ldots, k$, y_k-val végződő geometriai sorozat elemeiként adódik. □



10. ábra. A $W_{-1}(x)$ függvény grafikonja a $[-0,3,\ 0)$ intervallumban.

A feladat megoldását a Lambert-féle W függvény [20] egyik valós ágának egy függvényértéke segítségével fejezzük ki. A Lambert-féle W függvény a $W(z)e^{W(z)} = z$ egyenlet megoldása. Általánosan, komplex z értékekre definiálva a feladatot a megoldás többértékű, komplex. Valós x-ekre $W(x)$-nek $x \geq 0$ esetén egyetlen valós ága van, míg $-1/e \leq x \leq 0$ esetén kettő. Ezek közül a $-1 \leq W(x)$-et kielégítő ágat W_0-val, míg a másik ágat, amelyre $W(x) \leq -1$ teljesül $W_{-1}(x)$-szel (lásd 10. ábra) jelöljük. Ha $x < -1/e$, akkor $W(x)$-nek ebben az esetben is több komplex ága van.

8. Lemma. *Az $f(x)$ függvény maximuma a $\left(0, \frac{1}{2}\right]$ intervallumban* $1 - \frac{1}{W_{-1}\left(\frac{-2}{e^3}\right)+1} \approx 1,3871$*, ahol $W_{-1}(x)$ a Lambert-féle W függvény azon valós ága, amelyre $W(x) \leq -1$.*

BIZONYÍTÁS. A bizonyítás egyszerűen adódik, tekintve az $f(x)$ függvény derivált-függvényét, és analitikusan elemezve azt. $\square$

Az eredmények néhány kiterjesztése

Az alsó korlátot eredményező modellünk lényege az volt, hogy konstrukciónk szinte majdnem teljesen „kikapcsolta" az átpakolás szerepét. A bizonyítás alapgondolatát felelevenítve az triviális egy L_i $(i = 1, \ldots, k)$ lista elemeinek $x_i + \varepsilon_i$ méret-definíciójából, hogy egy ilyen nagy elem csak egy olyan ládában lehet, amelyben a többi elem összmérete legfeljebb $y_i - \varepsilon_i$ $(y_i = 1 - x_i)$. A lehetséges átpakolásokat is figyelembe véve – a fenti érvelés nyomán – egy nagy, $x_i + \varepsilon_i$ méretű elemet tartalmazó láda szintje legfeljebb y_i lehetett L_0 elpakolásának befejeztekor. Egy B ládát y_i-típusúnak neveztünk, ha $szint(B) \in (y_{i+1}, y_i]$, és $z_i n$ jelölte az összes olyan kis elem méretének összegét, amelyek y_i-típusú ládákban vannak L_0 elpakolása után. Ezek alapján ezután becslést adhattunk (31)-re csak az L_0, illetve az

$L_0L_j,\ j = 1, \ldots, k$ listákat adva inputként. Ez vezetett a 2. és 3. Lemmákon keresztül a 2. Tételhez, és az abban szereplő (38) feladathoz.

A továbbiakhoz érdemes megjegyezni, hogy tekintve egy bármilyen, akár véletlen módon generált, a fenti feltételeknek megfelelő y-rendszert, azaz tetszőleges módon rögzítve az $y_1, \ldots, y_k$ értékeket úgy, hogy $0 < y_k < \ldots < y_1 \leq \frac{1}{2}$ teljesüljön, a (38) nemlineáris optimalizálási feladat egy LP-vé redukálódik. Minden, így adódó LP megoldásából kapott optimumérték pedig egy alsó korlátot szolgáltat a c-átpakolásos szemi-on-line ládapakolási feladatra. A következő állítás azt mutatja, hogy ugyanezek az értékek nem csak 1, hanem minden d ($d \geq 1$, egész) dimenzióban is érvényes alsó korlátok a c-átpakolásos szemi-on-line ládapakolási feladatra, bármely $c \geq 1$ egészre. A bizonyításban a fentiekkel analóg konstrukciót és jelöléseket alkalmazva – méret alatt mindenütt d-dimenziós térfogatot értve – ugyanis azt kapjuk, hogy a konstrukció kis módosítása érvényes d-dimenzióban is, és ugyanahhoz a (38) optimalizálási feladathoz vezet. Így végső soron a konstrukcióból adódó optimalizálási probléma és az annak aszimptotikus vizsgálatából származó $1,3871$-es alsó korlát is érvényes minden d dimenzióban.

9. Lemma. *Legyen c tetszőleges pozitív egész és legyenek $y_1, \ldots, y_k$ olyan valós számok, amelyekre $0 < y_k < \ldots < y_1 \leq \frac{1}{2}$ teljesül. Ekkor a (38) feladat megoldása alsó korlátot jelent a c-átpakolásos szemi-on-line ládapakolási feladatra, bármely d dimenzióban (ahol $d \geq 1$ egész szám).*

BIZONYÍTÁS. Az egydimenziós c-átpakolásos szemi-on-line feladatra adott alsó korlát konstrukciónk analógiájára tekintsünk egy d-dimenziós *kockapakolási* feladatot. Az ε és ε_i $(i = 1, \ldots, k)$ értékek legyenek olyanok, hogy ezekre teljesüljenek a fenti előírások, míg az L_0 listabeli elemek méretének megadásához használt a érték itt legyen olyan, hogy a fenti, $a < \frac{\varepsilon_k}{\lceil \frac{n}{y_k} \rceil c}$ előíráson felül $a \leq (\frac{\varepsilon}{\lceil \frac{n}{2y_k} \rceil d \cdot (2^d - 1)})^d$ is teljesüljön rá. Ezek után, a listáink legyenek ugyanazok, mint a fenti konstrukcióban, azzal a különbséggel, hogy egy adott méretű elem helyett tekintsünk egy ugyanolyan *térfogatú* d-dimenziós kockát. Így az L_0, $L_1, \ldots, L_k$ listák elemei olyan d-dimenziós kockák lesznek, amelyek térfogatai rendre a, $x_1 + \varepsilon_1, \ldots, x_k + \varepsilon_k$ ($x_i = 1 - y_i,\ i = 1, \ldots, k$), azaz éleik rendre $\sqrt[d]{a}$, $\sqrt[d]{x_1 + \varepsilon_1}, \ldots, \sqrt[d]{x_k + \varepsilon_k}$ hosszúak. A fentivel analóg módon, a z_j $(j = 1, ..., k)$ értékek jelentsék az L_0 elpakolása után a ládákban elfoglalt megfelelő *térfogat* összegeket.

Követve az egydimenziós esetbeli bizonyítás gondolatmenetét, megmutatjuk, hogy konstrukciónk ezen, d-dimenziós általánosítása ugyanahhoz a (38) optimalizálási feladathoz vezet, mint az. Ehhez emlékeztetünk arra, hogy (38) első egyenlőtlensége a 3. Lemmából második egyenlőtlensége a 2. Lemmából adódtak: $\dfrac{A(L_0L_i)}{OPT(L_0L_i)}$ $(i = 1, \ldots, k)$, illetve $\dfrac{A(L_0)}{OPT(L_0)}$ becsléséből.

A továbbiakban legyen $i \in \{1, \ldots, k\}$ tetszőleges. Az első észrevételünk az, hogy az $A(L_0)$-ra és $A(L_0L_i)$-re a 2. és 3. Lemma bizonyításában adott alsó becsléseink ekkor is érvényben maradnak. Ez abból a tényből következik, hogy az ottani, kvantitatív elemzések szóról szóra érvényesek, elemek mérete és a ládában maradt kitöltetlen helyek alatt ugyanazon *térfogatokat*, és láda szintje alatt a ládában lévő elemek *térfogatainak* összegét értve. Így a bizonyításban ezek után elegendő $OPT(L_0)$-t és $OPT(L_0L_i)$-t felülről becsülnünk, és megmutatni, hogy ezekre ugyanazok a felső korlátok érvényesek, mint egy dimenzióban (lásd 2. és 3. Lemma bizonyítása).

Először $OPT(L_0)$-t fogjuk becsülni. Emlékeztetünk, hogy L_0 kis, egyenlő méretű elemeket tartalmaz. (Nevezzük ezeket *kis elemeknek*.) Megadjuk ezen elemek egy lehetséges pakolását. Ehhez egy egyszerű, szintenként haladó Next Fit típusú stratégiát fogunk használni. (LNF-fel jelöljük ezt az algoritmust (*Leveled Next Fit*)). Az LNF algoritmus egy láda első elemét elhelyezi a láda „bal alsó" csúcsához illesztve, majd szintenként annyi kis elemet pakol, a lehető legszorosabban egymás mellé, amennyit csak lehetséges, szintről szintre haladva. Ha nem maradt annyi hely az aktuális ládában, ahová a következő kis elem elférne, akkor az algoritmus az aktuális ládát telinek nyilvánítja és új ládát nyit.

Ez a szabály azt eredményezi, hogy L_0 elemeivel minden teli ládán belül egy $1-\sqrt[d]{a}$ élhosszúságú d-dimenziós kocka teljesen meg lesz töltve, így a kitöltetlen hely legfeljebb $a' = d \cdot \sqrt[d]{a}$ minden ládában, legfeljebb az utolsóként megnyitott ládát kivéve. Így

$$OPT\,(L_0) \leq LNF\,(L_0) \leq \frac{Na}{1-a'} + 1 \leq \frac{\frac{n}{2}-\varepsilon}{1-a'} + 1 \leq \frac{\frac{n}{2}}{1-a'} + 1,$$

azaz a 2. Lemma állítása teljesül a d-dimenziós konstrukcióra is.

Ahhoz, hogy $OPT(L_0L_i)$-t felülről becsüljük, szintén L_0L_i egy lehetséges pakolását fogjuk megadni. Először elpakoljuk L_i elemeit, mindegyiket egy-egy külön ládába (nevezzük ezeket a ládákat *nagy ládáknak*), az elemet a láda „bal alsó" csúcsához illesztve. Ezután elpakoljuk L_0 elemeit, először a nagy ládákba, LNF szabály szerint, majd ha maradt ki kis elem, akkor egy új ládát nyitva, és abba pakolva a kimaradó kis elemeket, szintén LNF szabállyal. Nevezzük ezt az algoritmust Dupla LNF-nek (DLNF).

Megjegyezzük, hogy a nagy ládákba a kis elemeket nem csak a nagy elem tetejére pakoljuk, hanem amennyire csak lehet, kitöltjük a lapjai melletti szabad helyeket is. A nagy ládákban a nagy elem, mint d-dimenziós kocka lapjai által meghatározott hipersíkok 2^d darab d-dimenziós téglalapra osztják a d-dimenziós egységkockát, amiből egy maga a nagy elem d-dimenziós kockája. Észrevehető, hogy egy nagy elem elhelyezése után $y_i - \varepsilon_i$ térfogatú, míg a kis elemek elpakolása után legfeljebb $a" = d \cdot (2^d - 1)\sqrt[d]{a}$ térfogatú kitöltetlen hely marad annak ládájában. A 3. Lemma bizonyításához hasonló módon, jelölje $kis(B_i)$ az összes, nagy ládában lévő összes kis elem térfogatának összegét. Ekkor

$$kis(B_i) \;\geq\; \left\lceil \frac{n}{2y_i} \right\rceil (y_i - \varepsilon_i - a") \geq \frac{n}{2} - \left\lceil \frac{n}{2y_i} \right\rceil \varepsilon_i - \left\lceil \frac{n}{2y_i} \right\rceil a" \geq Na - 2\varepsilon.$$

Itt kihasználtuk, hogy $\left\lceil \frac{n}{2y_i} \right\rceil \varepsilon_i = \varepsilon$ és $a \leq \left(\frac{\varepsilon}{\left\lceil \frac{n}{2y_i} \right\rceil d\cdot(2^d-1)}\right)^d$. Ebből következően, a csak kis elemeket tartalmazó ládákban lévő elemek összmennyisége legfeljebb 2ε, így ezen ládák (korábban S_i-vel jelölt) száma legfeljebb 1, így (37) teljesül, azaz $OPT\,(L_0 L_i) \leq \frac{n}{2y_i} + 2$ ebben az esetben is. Mivel i tetszőleges $\{1, \ldots, k\}$-beli volt, ezzel beláttuk, hogy a 3. Lemma állítása teljesül erre az esetre is.

Így ebben az esetben a (38) feladat érvényes marad, és optimuma alsó korlátot szolgáltat a d-dimenziós kockapakolási feladatra is. Mivel a d-dimenziós kockapakolási feladat speciális esete az állításbeli d-dimenziós c-átpakolásos szemi-on-line ládapakolási feladatnak, így ezzel az állítást bizonyítottuk. □

A következő lemmával azt mutatjuk meg, hogy ez a konstrukció – ugyanazon alsó korlát értékeket adva – érvényes más szemi-on-line ládapakolási problémákra is, azaz az általunk tárgyalt 3 szemi-on-line feladat bármelyikére.

10. Lemma. *Legyenek c tetszőleges pozitív egész és $y_1, \ldots, y_k$ olyan valós számok, amelyekre $0 < y_k < \ldots < y_1 \leq \frac{1}{2}$ teljesül. Ekkor a (38) feladat megoldása egy alsó korlátot ad nemcsak a c-átpakolásos szemi-on-line, hanem a c-átpakolásos teljesen dinamikus, és a 2-batched ládapakolási (2-BBP) feladatra is.*

BIZONYÍTÁS. A c-átpakolásos szemi-on-line ládapakolási feladat a c-átpakolásos teljesen dinamikus ládapakolási feladat speciális esete. A különbség csak az, hogy az utóbbi feladatnál Törlés művelet is megengedett az Ütemező számára, azaz korábban érkezett elemek távozását is előírhatja az inputban. Mivel a fenti konstrukcióban a mi listáink nem használják a Törlés műveletet, látható, hogy a fenti konstrukció, és így a (38) feladat is érvényes, és ugyanazokat az alsó korlát értékeket adja a c-átpakolásos teljesen dinamikus ládapakolási feladatra.

A *2-BBP* probléma esetén az Ütemező minden inputnál legfeljebb két egymást követő lista (batch) érkezését írhatja elő, míg az algoritmus számára az egyetlen megkötés az, hogy második lista érkezésekor nem mozgathatja az első lista elemeit. Mivel a mi konstrukciónk legfeljebb két listából álló listakonkatenációkat használ (alternatív módon), így egyszerre mindig legfeljebb két batchről beszélhetünk. Könnyen látható, hogy egy-egy batch érkezése alatt a batch egymással megegyező méretű saját elemeinek átpakolási lehetőségétől eltekinthetünk, hiszen annak szerepét figyelmen kívül hagytuk, mivel a bizonyítás során – lásd 2. és 3. Lemma – mindig csak az egy-egy lista elpakolásának befejeztével fennálló állapotot tekintettük. Így ez a speciális konstrukció érvényes a *2-BBP* feladatra is. Ezzel az állítást beláttuk. De megjegyezzük, hogy konstrukciónk általánosabb, hiszen akkor is érvényes maradna, ha a második lista minden elemének érkezésekor az első lista elemeiből legfeljebb c darab átpakolását is megengednénk az algoritmus számára. Így alsó korlátaink érvényesek lennének a 2-BBP feladat ezen meglazítására is, amely a fenti analógiára *c-átpakolásos 2-BBP* feladatnak nevezhető. □

A 9. Lemmához hasonló módon bizonyítható, hogy utolsó állításunk minden d dimenzióban igaz, azaz a két lemma eredményei kombinálhatóak. Ezért a (38) feladatba bármely, a feltételeknek megfelelő y-rendszert beírva, annak optimális megoldása alsó korlátot jelent a 3 különböző, 10. Lemmában említett szemi-on-line ládapakolási feladatra, bármely d dimenzióban, így mindezekre az $1,3871$-es alsó korlát is érvényes.

4.3. Alsó korlát konstrukciónk speciális esetei

Ebben az alfejezetben a [3] cikkünk alapján a 10. Lemmában említett 3 különböző szemi-on-line ládapakolási feladat, azaz

1. a c-átpakolásos szemi-on-line feladat,
2. a c-átpakolásos teljesen dinamikus ládapakolási feladat, és a
3. a 2-BBP feladat

azon speciális eseteire adunk alsó korlátokat, amikor a különböző elemméretek száma legfeljebb p lehet. ($c \geq 1$ tetszőleges, $p \geq 2$ előre rögzített egész szám.) Ennek gyakorlati motiváltsága kézenfekvőnek tűnik, hiszen ha egy cég termékeinek teherautókon való elszállítása a feladat, akkor világos, hogy korlátozott számú különböző terméktípust kell elpakolni.

A p különböző elemméret konstrukciónkban a $k = p - 1$ beállítással biztosítható. Ehhez azt kell észrevenni, hogy mivel a konstrukciónk $k + 1$ darab listát alkalmaz ($L_0,\ L_1, \ldots, L_k$), és minden egyes listán belül az elemek méretei megegyeznek, így az $k + 1$ darab különböző elemméretet használ.

Ezért ahhoz, hogy a feladatok ezen speciális, legfeljebb p különböző elemméretet megengedő esetére alsó korlátot adjunk egy adott p esetén, ezt a feladatot konstrukciónk, és az ebből adódó optimalizálási feladatok rögzített k értékhez tartozó esetének megoldására vezetjük vissza.

A $k = p-1$ beállítással a fentiekhez hasonlóan bármely, a $0 < y_1 < \ldots < y_k \leq \frac{1}{2}$ feltételnek megfelelő, rögzített y-rendszert beírva (38)-ba, és megoldva azt, érvényes alsó korlátokat nyerünk, méghozzá a 10. Lemma figyelembevételével a három feladat bármelyikének ezen p-hez tartozó esetére. Azonban a korábbiakhoz hasonló módon, egy rögzített k-hoz tartozó legjobb alsó korlát nyeréséhez az y-rendszert optimálisan kell választani, azaz a

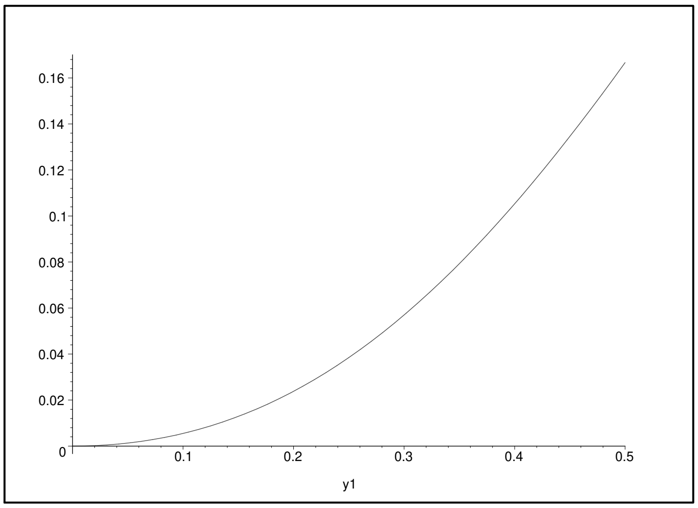
$$\max_{0 < y_k < \ldots < y_1 \leq \frac{1}{2}} b^*(y_1, \ldots, y_k), \tag{46}$$

„max-min" típusú feladatot kell megoldani, ahol $b^*(y_1, \ldots, y_k)$ jelöli a (38) feladatnak az $y_1, \ldots, y_k$ rendszerhez tartozó optimumértékét.

Mielőtt ezt megvizsgálnánk – ezt is előkészítve – megmutatjuk, hogy a [45]-beli és az [58]-beli eredményeket a mi konstrukciónk speciális eseteiben visszaadja.

A k=1 eset

Először azt mutatjuk meg, hogy az [58]-beli $\frac{4}{3}$-os alsó korlátot eredményező konstrukció a mi konstrukciónk speciális eseteként adódik. (Ezt a c-átpakolásos teljesen dinamikus ládapakolási feladatra adták meg az [58] cikk szerzői, Ivkovič és Lloyd.) Ez konstrukciónk $k = 1$ esete, azaz amikor – a mi jelölésrendszerünkben – csak az L_0 és L_1 listákat alkalmazzuk, és az ottani elemméretek is megfelelnek



11. ábra. Az f_1 függvény grafikonja a $\left(0, \frac{1}{2}\right]$ intervallumban.

az általunk használtnak. Megmutatjuk, hogy ebben az esetben ($k = 1,\ p = 2$) a mi konstrukciónkkal adható legjobb alsó korlát, azaz (46) optimumértéke $\frac{4}{3}$. Másrészt megjegyezzük, hogy [45]-ben az is bizonyított, hogy a $p = 2$ esetben adott $\frac{4}{3}$-os alsó korlát a 2-BBP feladatra optimális – egy [46]-beli eredmény felhasználásával, ahol megmutatták, hogy ez felső korlát is.

11. Lemma. *A $k = 1$ esetben a (46) feladat megoldása $\frac{4}{3}$, melyet az $y_1 = \frac{1}{2}$-nél vesz fel.*

BIZONYÍTÁS. $k = 1$-re a (38) feladat a következő alakú:

$$\min \quad b, \tag{47}$$
$$b \geq 1 + y_1 - 2y_1 z_1, \tag{48}$$
$$b \geq 1 + 2z_1\left(\frac{1}{y_1} - 1\right), \tag{49}$$
$$0 \leq z_1 < \frac{1}{2}. \tag{50}$$

Ha rögzítjük y_1-et, akkor (48) és (49) jobb oldala két, a z_1 változóban lineáris függvény. Könnyen látható, hogy (47)-(50) az optimális megoldását a két egyenes metszéspontjában veszi fel. Egymással egyenlővé téve a két jobb oldalt, és megoldva az így kapott egyenletet, azt kapjuk, hogy az optimális megoldás az $\frac{1}{2}\frac{y_1^2}{y_1^2-y_1+1}$ pont. Tekintsük ezután az $f_1(y_1) := \frac{1}{2}\frac{y_1^2}{y_1^2-y_1+1}$ függvényt a $\left(0, \frac{1}{2}\right]$ intervallumon (lásd 11. ábra)! Ahhoz, hogy $k = 1$ esetén megkapjuk (46) optimumértékét, y_1-et úgy kell választanunk, hogy f_1 a $(0, \frac{1}{2}]$ intervallumbeli maximumát ebben a pontban vegye fel. Deriválva f_1-et könnyen ellenőrizhető, hogy f_1 monoton növekvő ebben az intervallumban, így az $y_1 = \frac{1}{2}$ pontban veszi fel maximumát. Visszahelyettesítve ezt a (48) és (49) jobb oldalába, majd megoldva az LP-t kapjuk, hogy a $k = 1$ esetben (46) optimumértéke, azaz a keresett, lehetséges legjobb alsó korlát értéke $\frac{4}{3}$. □

A $k = 1$ eset vizsgálatával ezzel megmutattuk, hogy modellünk általánosítja az [58]-beli modellt.

A $\mathbf{k \geq 2}$ eset

A $k = 1$ esetet azért is tárgyaltuk külön, mert ez $k \geq 2$ esetén egy további feltételt eredményez: a következő lemma alapján vizsgálatainkban $y_k \geq \frac{1}{4}$ is feltehető lesz, így elegendő lesz az $\frac{1}{4} \leq y_k < \ldots y_1 \leq \frac{1}{2}$ feltétel mellett optimalizálni (46)-ot, és az így adódó feladat az eredetivel ekvivalens marad. Numerikusan ez a feltevés azzal az előnnyel jár, hogy ez után az y értékek már egy *zárt*, az $[\frac{1}{4}, \frac{1}{2}]$ intervallumban helyezkednek el.

12. Lemma. *A $k \geq 2$ esetekben (46) optimalizálásakor $y_j \geq \frac{1}{4}$ ($j = 1, \ldots, k$) is feltehető, és az így kapott feladat az eredetivel ekvivalens marad.*

BIZONYÍTÁS. A 11. Lemmában láttuk hogy a $k = 1$ esetben (46) optimumának értéke az $y_1 = \frac{1}{2}$ ponthoz tartozó alsó korlát, $\frac{4}{3}$. Ebből következően a $k \geq 2$ esetekre (46) optimumának értéke legalább $\frac{4}{3}$ lesz, ami azt jelenti, hogy ilyen (vagy ennél jobb) alsó korlátok eléréséhez a konstrukciónk $L_1, \ldots, L_k$ listáiban elegendő $\frac{3}{4}$-nél kisebb vagy egyenlő méretű x_j ($j = 1, \ldots, k$) elemeket tekinteni. Ellenkező esetben az algoritmus a $\frac{3}{4}$-nél nagyobb méretű elemeket bárhogyan pakolhatja – akár úgy, hogy nem is helyez az őket tartalmazó ládába más elemet –, ezek a ládák triviális módon $\frac{3}{4}$-ig tele lesznek. Ez azt jelenti, hogy a legalább $\frac{3}{4}$ méretű elemek, azaz az ilyen nagy elemeket tartalmazó listák elhagyása nem rontja az alsó korlátot. Így $k \geq 2$ esetén feltehetjük, hogy $y_j \geq \frac{1}{4}$ ($j = 1, \ldots, k$). □

Gutin, Jensen és Yeo [45] tanulmánya a 2-BBP problémára szolgáltatott megoldásokat, arra az esetre, amikor legfeljebb p különböző elemméret megengedett. Most annak bemutatására térünk rá, hogy modellünk általánosítja az ottani konstrukciót,vagy másként fogalmazva, speciális eseteiben visszaadja azt, és az ottani eredményeket.

Az előzőek alapján (láttuk, hogy a p különböző elemméret a $k := p - 1$ beállítással biztosítható, a 10. Lemma alapján pedig konstrukciónk érvényes a 2-BBP feladatra) ahhoz, hogy egy konkrét $p \geq 3$ értékre alsó korlátot adjunk meg, elegendő egy tetszőleges módon választott $y_1, \ldots, y_k$ ($k = p - 1$, $0 < y_k \leq \ldots y_1 \leq \frac{1}{2}$) rendszert tekinteni, és ezt beírni (38)-ba. A korábbiak alapján az így adódó LP optimumértéke alsó korlátot szolgáltat a [45]-ben vizsgált problémára.

Az y_j értékekre a legegyszerűbbnek tűnő választás egy ekvidisztáns beosztásból származó pontrendszert tekinteni. A 11. Lemma alapján a $\frac{4}{3}$ egy érvényes alsó korlát érték bármely ilyen feladatra. Ennél jobb alsó korlátok adásához a 12. Lemma bizonyítása alapján elegendő az y-rendszert az $[\frac{1}{4}, \frac{1}{2}]$ intervallumon tekinteni. Így ezt az intervallumot osztottuk fel $k + 1$ osztóponttal ekvidisztáns módon, és az első osztópontot, $\frac{1}{4}$-et nem vettük bele az y-rendszerbe, azaz $y_j := \frac{1}{2} - \frac{1}{4}\frac{j-1}{k}$, $j = 1, \ldots, k$. Az ezekhez tartozó (38) feladatok (LP-k) optimumértékei bármely,

9. táblázat. A 2-BBP probléma azon változatára adott alsó korlátok, amikor csak előre rögzített számú, maximum $p(=k+1)$ különböző elemméret megengedett.

k	*Alsó korlátok ekvidisztáns y-rendszer esetén*	*Alsó korlátok tetszőleges y-ok megengedésekor*
1	1,3333...	—
2	1,3658...	1,36602540378
3	1,3738...	1,37393876913
4	1,3773...	1,37753136189
5	1,3793...	1,37958528769
6	1,38051	1,38091512540
7	1,38136	1,38184652163
8	1,38198	1,38253525895
9	1,38246	1,38306525702
10	1,38296	1,38348573275
20	1,38509	1,38534022765
50	1,38631	1,38642436208
100	1,38673	1,38678113846
1000	1,38709	1,38710030535

standard LP-megoldóval meghatározhatók: néhány konkrét k $(= p-1)$ értékre az így kapott alsó korlát értékeket jelenítettük meg a 9. táblázat második oszlopában.

A kapott alsó korlát értékek megegyeznek a Gutin és szerzőtársai által a 2-BBP feladatra kapott megfelelő értékekkel; pontosabban az első 6 érték, mivel ott csak ezeket közölték a szerzők. Emiatt az első 6 érték megadásánál az ott használt 4 tizedesjegy pontosságot és kerekített jegyeket használtuk. A táblázatban néhány további k-hoz tartozó, általunk kiszámolt megoldásértéket is közlünk. $k = 10$-ig minden alsó korlát értéket megadunk, és kiragadott példaként feltüntetjük még a $k = 20, 50, 100, 1000$ esetekben nyert értékeket is, 5 jegyre kerekítve – az általunk használt LP-megoldó ily módon szolgáltatott eredményt.

A [45]-ben alkalmazott alsó korlát konstrukció hasonló. Az ottani elemméretek értékei is megfelelnek a mi konstrukciónk feltételeinek (az ottani L_0 lista s, $0 < s < \frac{1}{2}$ méretű, míg az L_j listák n/j darab $1 - js$, $j = 1, \ldots, m := \lceil \frac{1}{2s} \rceil - 1$ méretű elemeket tartalmaznak). Emiatt a [45]-beli modell az általunk megadott modellnek egy olyan, speciális esete, amikor (a mi terminológiánkban) az y-rendszer egy konkrétan előírt ekvidisztáns pontrendszer. Ezért modellünk a [45]-beli modell általánosításának tekinthető.

A kapcsolódó nemlineáris optimalizálási feladat elemzése és megoldása

Rátérünk annak a vizsgálatára, hogy rögzített k $(k \geq 2)$ értékre mi a konstrukciónkhoz kapcsolódó feladatok megoldásával megadható legjobb alsó korlát. Kaphatunk-e jobb alsó korlátokat, mint ekvidisztáns beosztásból való y-rendszer választása esetén? Ennek megválaszolására a (46) nemlineáris optimalizálási feladat optimális megoldását rögzített k $(k \geq 2)$ értékek mellett kell meghatározni.

A kérdésre igenlő választ adunk. Ezzel egyben megválaszolunk egy [45]-ben felvetett kérdést is. Nevezetesen azt, hogy a 2-BBP feladat p számú különböző elemméretet megengedő változatának $p \geq 3$ eseteire is optimálisak-e az ott megadott korlátok. Mivel $k \geq 2$-re $(p = k + 1)$ a (46) feladat megoldásai érvényes alsó korlátokat szolgáltatnak erre a feladatra is, így ezekkel javítjuk a [45]-ben a konkrét $p \geq 3$-hoz tartozó esetekre megadott alsó korlát értékeket. A javítás mértéke ugyan nem túlságosan nagy, azonban a [45]-beli fenti kérdésre elméleti szempontból választ jelent, mivel az ott megadott korlátok így nem lehetnek optimálisak. Megjegyezzük, hogy a feladat megoldásából adódó alsó korlátok a 2-BBP probléma mellett érvényesek lesznek a fenti, másik két c-átpakolásos feladat hasonló, legfeljebb p különböző elemméretet megengedő változatára is, és a 2-BBP probléma hasonló módon definiálható c-átpakolásos variánsára is.

Az 12. Lemma miatt a (46) feladattal ekvivalens, következő feladatot kapjuk az $\frac{1}{4} \leq y_k$ plusz feltevés felhasználásával:

$$\max_{\frac{1}{4} \leq y_k < \ldots < y_1 \leq \frac{1}{2}} \quad b^*(y_1, \ldots, y_k), \tag{51}$$

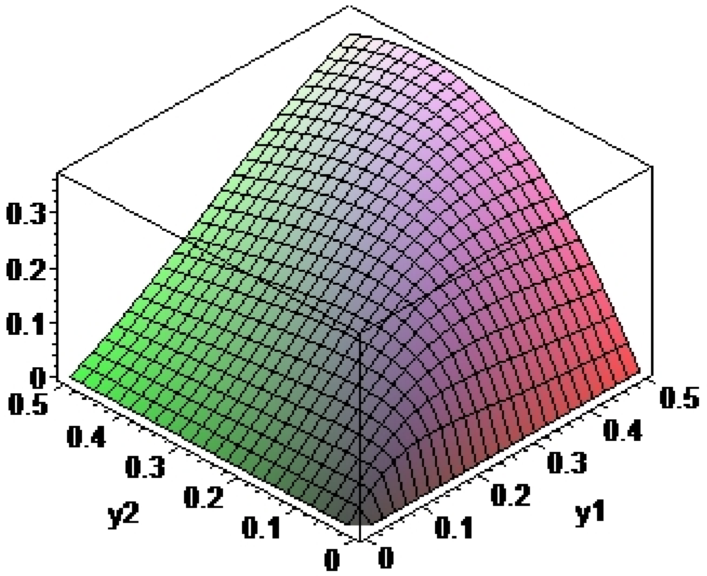
ahol $b^*(y_1, \ldots, y_k)$ itt is a (38) feladatnak az $y_1, \ldots, y_k$ rendszerhez tartozó optimumértékét jelöli. A kérdés az, hogy rögzített k érték mellett melyik y-rendszert kell tekintenünk (51) optimumának eléréséhez. A kérdés megválaszolásához globális optimalizálási módszereket használunk.

Az (51) feladat az így megadott alakban globális optimalizálási szempontból nagyon nehezen megoldhatónak tűnik. Felhasználva az előző alfejezetbeli átalakítások eredményét, az ezek után adódó feladatot fogjuk megoldani, egy numerikus, globális optimalizálási módszert alkalmazva erre. Követve az ott részletesen leírt transzformációkat, a feladat (44) formáját nyerjük, azzal a különbséggel, hogy korlátozó feltételként az $y_1, \ldots y_k$ változókra a $[0, \frac{1}{2}]$ intervallum helyett az írható elő, hogy azok az $[\frac{1}{4}, \frac{1}{2}]$ intervallumba esnek. Mivel ez egy zárt intervallum, így ez lényegesen könnyebben kezelhetővé teszi a feladatot. A feladat most tehát:

$$\max_{\frac{1}{4} \leq y_k < \ldots < y_1 \leq \frac{1}{2}} \quad f_k(y) = 1 + \frac{1 - y_k}{y_k + \frac{1}{y_1} + \sum_{i=2}^{k} \frac{y_{i-1}}{y_i} - k}, \tag{52}$$

ahol $k \geq 2$, előre rögzített egész. (A 12. ábra a $k = 2$ esetbeli f_2 grafikonját mutatja.)

A 7. Lemma bizonyítása alapján (itt már a feltételek miatt elegendő annak a bizonyításának csak a második, $y_2 \geq \frac{y_1}{2}$ ágát tekinteni) a következő állítás közvetlenül adódik:



12. ábra. A $k = 2$ esethez tartozó f_2 függvény grafikonja.

1. Következmény. *Legyen $k \geq 2$ egy tetszőleges, rögzített egész szám. Tegyük fel, hogy a*

$$\max_{\frac{1}{4} \leq y_k \leq \frac{1}{2}} t(y_k) = 1 + \frac{1-y_k}{y_k+2-k+(k-1)(2y_k)^{-1/(k-1)}}, \tag{53}$$

egydimenziós optimalizálási feladatnak egyetlen $y_k = y^* \in [\frac{1}{4}, \frac{1}{2}]$ optimális megoldása van, és $t(y^*) = t^*$. Ekkor a k–dimenziós (52) feladatnak szintén egyetlen optimális megoldása van, és ez a következő módon adható meg:

$$y_1 = \frac{1}{2}, \;\; y_i = \frac{1}{2}(2y^*)^{\frac{i-1}{k-1}}, \; i = 2, \ldots, k. \tag{54}$$

Továbbá, az eredeti feladat optimális megoldásának értéke is t^*-gal egyenlő.

A fenti eredményeket összegezve; azt kaptuk, hogy ha az (53) feladat maximumpontja egyértelmű, akkor az eredeti (52) feladat maximumpontja ugyancsak egyértelmű, és ez az utóbbi megoldás az (54) összefüggések által meghatározott. Továbbá, a két probléma maximumának értéke is megegyezik. Az (53) feladat maximumpontját és annak egyértelműségét a következő rész numerikus módszereivel fogjuk igazolni. Az eredeti probléma ezen módosítására is érvényesek a 7. és a 8. Lemma állításai, amelyek azt mutatják, hogy ha a k tart végtelenbe, akkor az ezen k-khoz tartozó alsó korlátokból álló sorozat tart $1,3871...$-hez.

Az optimalizálási probléma numerikus megoldása

Az (53) feladat numerikus megoldására a [69]-ben leírt globális optimalizálási módszert alkalmaztuk. Ez egy speciális, intervallum-matematikán (lásd [48, 77, 91]) alapuló, Branch-and-Bound [53] típusú globális optimalizálási módszer. A felhasznált módszer a Profil/BIAS [65] intervallum-matematikai könyvtári környezetben megvalósított, felhasználva a „C–XSC Numerical Toolbox for Verified Computing" [47] B&B eljárásának elemeit. A szoftvert a következő alakban felírható, ún. *bound-constrained* optimalizálási problémák megoldására tervezték:

$$\max_{x \in X_0} \quad f(x),$$

ahol $X_0 \subset \mathbb{R}^d$ egy d-dimenziós box és $f : \mathbb{R}^d \to \mathbb{R}$ kétszer folytonosan differenciálható függvény. A program nagy pontossággal képes kiszámítani *minden globális* optimumpont, valamint a globális optimum értékének intervallumos befoglalását.

Az optimumpontokat tartalmazó intervallumok meghatározásához a módszer használ továbbá egy *automatikus ellenőrző* lépést [92] megvalósító eljárást. Ez az eljárás a B&B algoritmus befejezésekor kapott intervallumokban a maximumpont létezésének, valamint lokális (és esetleg globális) egyértelműségének igazolására alkalmas.

Az (53) probléma megoldásait a $k = 2, \ldots, 10,\ 20,\ 50,\ 100,\ 1000$ paraméterértékekre kerestük meg. Minden egyes esetben sikerült nagy pontosságú befoglalásokat megadnunk mind az y^*, mind pedig a t^* értékeire, emellett az y^*-ot tartalmazó intervallumban az optimális megoldás létezését és egyértelműségét is igazoltuk. Az y^*-ot befoglaló intervallum lehetőséget adott az $y^* \leq \frac{1}{2}$ előfeltétel ellenőrzésére is. Ilymódon az 1. Következmény alapján az eredeti, (52) feladat *egyértelmű optimális megoldását* tartalmazó box konstruktív megadását is elvégezhettük.

A 9. táblázat harmadik oszlopa tartalmazza az eredeti probléma optimumának értékeit a megoldott példák esetén. A megadott értékek 11 tizedesjegyig pontosak. Ez azt jelenti, hogy a konstrukciónkkal a konkrét k (és $p = k + 1$) értékekre adható legjobb alsó korlátok *garantáltan* benne vannak a táblázatban a megadott érték, és az adott értékhez 10^{-11}-et hozzáadva kapott szám által meghatározott intervallumban.

4.4. Felső korlátok

A következőkben c-átpakolásos szemi-on-line ládapakolási algoritmusok egy sorozatát adjuk meg, minden $c \in N^+$ értékre egy, a c-től függő, HFR-c nevű algoritmust. Az elnevezés az algoritmus angol nevének rövidítéséből *(a Harmonic Fit type algorithm with Repacking of at most c number of elements per step)* származik. Bebizonyítjuk, hogy *AVK(HFR-c)* $= \frac{3}{2} + \frac{b_c}{1-b_c}$ teljesül, ahol b_c a $2cb_c^2 - (6c+3)b_c + 1$ másodfokú egyenlet $(0, \frac{1}{6c}]$ intervallumba eső gyöke. Az algoritmus végrehajtása során minden ládaosztályban – mindegyikben legfeljebb egy-egy ládát kivéve – $\frac{1}{2} + cb_c$ szintig megtöltött ládák elérését tűzzük ki célul. Két esetet fogunk megkülönböztetni. Ha ez az előbbi cél sikerül, akkor nyilván *AVK(HFR-c)* $\leq \frac{1}{\frac{1}{2}+cb_c}$. Ha nem, akkor a klasszikus súlyfüggvénytechnikát [108] használva belátjuk, hogy

AVK(HFR-c) $\leq \frac{3}{2} + \frac{b_c}{1-b_c}$. Mivel b_c paraméter, így a két jobboldalt egyenlővé téve kapjuk a fenti másodfokú egyenletet és abból a b_c érték beállítását.

Az algoritmus versenyképessége a $c = 1$ esetre irreleváns, mivel a jelenleg ismert legjobb AVK-jú on-line algoritmus jobb AVK-val rendelkezik. Bár megjegyzésre érdemes, hogy a HFR-1 jóval kevesebb ládaosztályt használ annál. Az első érdekes eredmény a $c = 2$ paraméterhez tartozik, mivel erre *AVK(HFR-2)* $\leq 1,5728...$ teljesül, ami jobb, mint a ma ismert legjobb on-line algoritmus AVK-ja [95]. Ennek másik értéke, hogy alatta van az úgynevezett „Harmonic Fit" típusú on-line ládapakolási algoritmusok $1,58333$-as alsó korlátjának [87]. A $c = 4$ esetben szintén javítást értünk el, mivel a *HFR-4* algoritmus jobb AVK-val rendelkezik, mint a legjobb ma ismert on-line alsó korlát. Ha c tart végtelenbe, akkor *AVK(HFR-c)* tart $1,5$-hez. Ez az eredmény azt bizonyítja, hogy egy, lépésenként konstans számú elem átpakolását megengedő szemi-on-line algoritmus versenyképessége jobb lehet bármely on-line algoritmusénál. Ez lényegében azt jelenti, hogy a rendelkezésünkre álló információt jól ki lehet használni szemi-on-line algoritmusok versenyképességének javítására.

A HFR-c algoritmus és bonyolultsága

Ezt a szakaszt néhány jelöléssel és definícióval kezdjük. Egy megnyitott B *láda szintje* $szint(B)$-vel jelölt, és a tartalmazott elemek méreteinek összege, így $0 < szint(B) \leq 1$. Egy elemet *kis elemnek* nevezünk, ha mérete a $(0, \frac{1}{2}]$ intervallumba esik, míg ha $x \in (\frac{1}{2}, 1]$, akkor *nagy elemnek*. *Nagy ládának* mondunk egy B ládát, ha tartalmaz egy nagy elemet.

A továbbiakban b_c-vel, illetve, ha c rögzített, akkor egyszerűen b-vel fogunk jelölni egy tetszőleges olyan értéket, amelyre $cb_c \in \left(0, \frac{1}{6}\right]$. Elemzésünk végén rögzíteni fogjuk ezt az értéket, minden c esetén úgy megválasztva, hogy az általunk megadott algoritmusok versenyképességi hányadosa a lehető legjobb legyen. Hangsúlyozzuk azonban, hogy addig a pontig elemzésünk bármely $b_c \in \left(0, \frac{1}{6c}\right]$ értékre igaz.

A továbbiakban legyen c egy tetszőleges, rögzített pozitív egész szám. Osszuk fel a $(0, 1]$ intervallumot $2c + 3$ darab részintervallum diszjunkt uniójára a következő módon:

$$(0, b] =: I_1,$$

$$\left(b, \frac{1}{2} - cb\right] =: I_2,$$

$$\left(\frac{1}{2} - cb, \frac{1}{2}\right] = \bigcup_{j=1,\ldots,c} \left(\frac{1}{2} - jb, \frac{1}{2} - (j-1)b\right] =: \bigcup_{j=1,\ldots,c} I_{c+3-j},$$

$$\left(\frac{1}{2}, \frac{1}{2} + cb\right] = \bigcup_{j=1,\ldots,c} \left(\frac{1}{2} + (j-1)b, \frac{1}{2} + jb\right] =: \bigcup_{j=1,\ldots,c} I_{c+2+j},$$

$$\left(\frac{1}{2} + cb, 1\right] =: I_{2c+3}.$$

Minden egyes intervallumhoz egy-egy ládaosztályt rendelünk hozzá, amelyeket rendre $\mathcal{B}_j$-vel ($j = 1, \dots 2c+3$) jelölünk. Egy B láda a $\mathcal{B}_j$ ládaosztályba kerül, ha a megnyitásakor a benne elhelyezett első elem mérete az I_j intervallumba esik. Ha $B \in \mathcal{B}_j$, $c+3 \leq j \leq 2c+2$, akkor a B láda az algoritmus futása alatt átkerülhet egy másik ládaosztályba is. Az átsorolás akkor történik meg, ha B ezen ládaosztályok egyikének, mondjuk $\mathcal{B}_l$-nek (aktuális) utolsó ládája, és olyan lépés hajtódik végre, amelynek során a lista éppen feldolgozás alatt lévő kis elemét belerakjuk B-be, vagy ha korábban saját osztályába elpakolt kis elemet pakolunk át B-be, és erre az új B-re már $szint(B) \notin I_l$.

Ennek az a következménye, hogy egy lista elpakolása során a felhasznált ládák száma dinamikusan változhat: ládákat nyithatunk meg, és ládák üresedhetnek meg. Fontos, hogy $A(L)$ számításakor csak azokat a ládákat tekintjük megnyitottnak, amelyek nem üresek.

Két ládaosztályt, $\mathcal{B}_j$-t és $\mathcal{B}_l$-t ($1 \leq j \leq c+2$ és $c+3 \leq l \leq 2c+3$) *komplemenseknek* nevezünk, ha az osztályokhoz rendelt I_j és I_l intervallumok tetszőleges elemeinek összege nem nagyobb, mint 1. Egy $\mathcal{B}_i$ ládaosztály komplemenseinek halmazát $C(\mathcal{B}_i)$-vel jelöljük ($1 \leq i \leq 2c+3$). Vegyük észre, hogy $C(\mathcal{B}_{c+2}) = \emptyset$ és $C(\mathcal{B}_{2c+3}) = \emptyset$, valamint hogy egy $\mathcal{B}_j$ ($1 \leq j \leq c+1$) ládaosztály legkisebb indexű komplemense $\mathcal{B}_{c+3}$, legnagyobb indexű $\mathcal{B}_l^*$, ahol $l^* = \min\{2c+2, 2c+4-j\}$. Hasonlóan egy $\mathcal{B}_l$ ($c+3 \leq l \leq 2c+2$) ládaosztály legnagyobb indexű komplemense $\mathcal{B}_{2c+4-l}$, legkisebb indexű $\mathcal{B}_1$.

A következőkben vázlatosan ismertetjük algoritmusunkat: a HFR-c alapvetően a „Harmonic Fit" (HF) típusú algoritmusoknál alkalmazott szabály szerint [87] pakolja el minden osztály elemeit. Az egy osztályba eső elemeken belül ez „Next Fit" (NF) szabály szerinti pakolást jelent. Ez konkrétan azt jelenti, hogy ha az érkező elem méretére $x \in I_j$ teljesül, és az utoljára megnyitott $\mathcal{B}_j$-beli B ládára $szint(B)+x \leq 1$, akkor x-et elpakoljuk B-be. Különben nyitunk egy új $\mathcal{B}_j$ osztálybeli ládát, és x-et ebben a ládában helyezzük el. Fontos, hogy a fenti, HF eljárásnál alkalmazott szabálytól annyiban térünk el, hogy az új láda megnyitásakor nem zárunk be ládát az adott osztályban, tehát azok tartalma később is hozzáférhető lesz. Mivel az algoritmusunk alapvetően Harmonic Fit típusú szabályt alkalmaz, bár osztópontjaink nem a $[0,1]$ intervallum harmonikus, hanem ekvidisztáns beosztásából származnak. (Mivel az irodalomban ettől függetlenül az ilyen típusú módszereket Harmonic Fit típusúaknak nevezik, ezért ez is Harmonic Fit típusúnak nevezhető – amely ugyan átpakolást használ.)

A HF szabály alkalmazásával minden olyan ládát, amely a $\mathcal{B}_i,\ i \leq c+2$, ládaosztályokba esik – legfeljebb az osztályok utoljára megnyitott ládáitól eltekintve – és a $\mathcal{B}_{2c+3}$ ládaosztályok ládáit is biztos, hogy legalább $1/2 + cb$ szintig telepakoljuk. Így, ha a lista nem tartalmazna más intervallumból elemeket, akkor a célként kitűzött aszimptotikus versenyképességi hányadost már el is érnénk. Mivel azonban a $\mathcal{B}_{c+3}, \dots, \mathcal{B}_{2c+2}$ osztályba tartozó ládákba a HF szabály egyetlen nagy elemet helyez, így ezek a ládák nem érik el a kívánt szintet. Ezért ezekre a ládákra finomítanunk kell az algoritmust: ha a lista elemeinek méretei ezt lehetővé teszik, akkor gondoskodnunk kell arról, hogy ezek is fel legyenek töltve a megfelelő szintig. Két esetben térünk el a HF szabálytól.

4. Algoritmus: HFR-c algoritmus

while van még elem az L listában **do**
    input a listában következő x;
    if $x \in I_l$, $c+2 \leq l \leq 2c+3$ **then**
        $x_{NF} \rightarrow \mathcal{B}_l$;
        if $x \notin I_{2c+3}$ **and** $x \notin I_{c+2}$ **then**
            call Átpakolás(l, $2c+3-l$);
        endif;
    endif;
    else if $x \in I_j$, hogy $1 \leq j \leq 2c+1$ **then**
        call Feltöltés(x);
    endelse;
endwhile;

- **1. szabály (Feltöltés)**: Ha egy olyan $x \in I_j$ kis elem érkezik, amelyre $1 \leq j \leq c+1$ teljesül, akkor megvizsgáljuk a $\mathcal{B}_j$ komplemens osztályait indexeik szerint növekvő sorrendben oly módon, hogy az első nem üres $\mathcal{B}_l$ osztály esetében annak utoljára megnyitott B ládájába elhelyezzük x-et. Az világos, hogy x elpakolása előtt $szint(B) \in I_l$. Ha az elpakolás után $szint(B) + x \notin I_l$, akkor B-t átsoroljuk abba a $\mathcal{B}_t$ ládaosztályba, amelyre $szint(B) + x \in I_t$.

  Ha a komplemensek mindegyike üres, akkor a kis elemet saját ládaosztályába ($\mathcal{B}_j$-be) helyezzük el az NF szabály használatával.

- **2. szabály (Átpakolás)**: Ha egy olyan nagy elem érkezik, amelyre $x \in I_l$, ahol $c+3 \leq l \leq 2c+2$ teljesül, akkor az elem elpakolásához először nyitunk egy $\mathcal{B}_l$ osztálybeli nagy ládát, és abba elpakoljuk x-et. Ezután indexeik szerint csökkenő sorrendben megvizsgáljuk, hogy van-e olyan $\mathcal{B}_j \in C(\mathcal{B}_l)$-beli láda, amely nem üres. Ha találunk ilyen ládát, akkor ebből a ládából egy elemet (a legfelsőt) áthelyezünk a nagy ládába. Amennyiben a nagy láda szintje az elpakolás után intervallumot vált, akkor a Feltöltési szabályban leírtakhoz hasonlóan járunk el. Figyelnünk kell arra, hogy ebben az esetben az a láda, amely a kis elemet tartalmazta, kiürülhet.

Az algoritmus és a két fenti szabály pontos leírását a megadott pszeudokódok tartalmazzák. A leírás során az $x_{NF} \rightarrow \mathcal{B}_t$, illetve $x \rightarrow \mathcal{B}_t$ jelölést használjuk akkor, ha az aktuális tárgyat a Next Fit szabállyal pakoljuk el a $\mathcal{B}_t$ ládaosztályba, illetve ha berakjuk a $\mathcal{B}_t$ ládaosztály utolsó ládájába. Hasonlóan, a $B \rightarrow \mathcal{B}_t$ jelölést használjuk annak rövidítésére, hogy a B ládát átsoroljuk a $\mathcal{B}_t$ ládaosztályba.

Fontos megjegyezni, hogy mindkét szabályban újabb átpakolással is megpróbálkozunk abban az esetben, ha egy $\mathcal{B}_l$-beli ($c+3 \leq l \leq 2c+2$) láda osztályt vált,

5. Algoritmus: Procedure Feltöltés(x)

$j :=$ annak az osztálynak az indexe, melyre $x \in I_j$;
$l^* := \min\{2c{+}4{-}j, 2c{+}2\}$; ($\mathcal{B}_j$ legnagyobb indexű komplemensének indexe)
do $l = c + 3$ **to** l^*
  if $\mathcal{B}_l \neq \emptyset$ **then**
    $x \rightarrow \mathcal{B}_l$;
    if $szint(B) \in I_p$, $p \neq l$ **then**
      $B \rightarrow \mathcal{B}_p$;
      if $p < 2c + 3$ **then**
        call Átpakolás(p,$2c + 4 - p$);
      endif;
    endif;
    return; (nincs osztályváltás vagy Átpakolásból tértünk vissza)
  endif;
enddo;
$x \rightarrow_{NF} \mathcal{B}_j$ osztály ládájába; (nem találtunk komplemenst)

6. Algoritmus: Procedure Átpakolás(l, j)

Megjegyzés: Az eljárás megpróbál átpakolni egy kis elemet egy $C(\mathcal{B}_l)$ osztálybeli ládából $\mathcal{B}_l$ utolsó ládájába ($l \geq c + 3$). A j paraméter adja meg azt, hogy a $C(\mathcal{B}_l)$ melyik elemétől kell kezdeni a keresést.

do $t = j$ **to** 1
  if $\mathcal{B}_t \neq \emptyset$ **then**
    $x \in B \in \mathcal{B}_t \rightarrow \mathcal{B}_l$; ($B$ a $\mathcal{B}_t$ utolsó ládája, x ennek legfelső eleme);
    if $szint(B) \in I_s$, $l + 1 \leq s \leq 2c + 3$ **then**
      $B \rightarrow \mathcal{B}_s$;
      if $s < 2c + 3$ **then**
        call Átpakolás(p, s);
      endif;
    endif;
  endif;
enddo;

és az új ládaosztály nem $\mathcal{B}_{2c+3}$. Ez a 2. szabályt pontosító Átpakolás eljárásban önmagának a rekurzív hívását jelenti.

A következő három állítás az algoritmus bonyolultságának vizsgálatához nyújt segítséget.

13. Lemma. *A HFR-c algoritmus véges.*

BIZONYÍTÁS. A főprogram pontosan n-szer fut le és a Feltöltés eljárás listaelemenként legfeljebb egyszer hajtódik végre.

A főprogramból vagy a Feltöltés eljárásból hívott Átpakolás eljárás ugyan rekurzívan meghívhatja önmagát, de a rekurzív hívás csak akkor következik be, ha az aktuálisan használt láda szintje intervallumot vált, de nem haladja meg az $\frac{1}{2} + cb$ szintet. Ennek következménye, hogy egy Átpakolás eljárás első paraméterének, l-nek az értéke – amely a nagy láda magasságának intervallumát jelzi – a rekurzióban egymást követő Átpakolás eljárások során szigorúan monoton nő. A rekurziót indító Átpakolás eljárásban $l \geq c+3$, az utolsónak hívottban legfeljebb $2c+2$. Ezért a rekurzív hívások száma listalemenként legfeljebb $c-1$, ami adja, hogy az Átpakolás eljárás legfeljebb c-szer hívódik meg listaelemenként.

Mivel az Átpakolás eljárás listaelemenként csak a főprogram és a Feltöltés eljárások legfeljebb egyikéből hívódik meg, így állításunk bizonyított. □

A bizonyításból közvetlenül adódik:

2. Következmény. *Ha c egy rögzített, pozitív egész szám, akkor a HFR-c algoritmus legfeljebb c elemet pakol át lépésenként.*

14. Lemma. *A HFR-c algoritmus futása során a ládaosztályok tartalmának vizsgálata ($B_i \neq \emptyset$ összehasonlítás) összesen legfeljebb $(2c+1)n$-szer történik meg.*

BIZONYÍTÁS. Azt fogjuk belátni, hogy listalemenként legfeljebb $2c+1$-szer történik meg az állításban említett összehasonlítás.

A főprogramban nincs, a Feltöltés eljárásban egy listaelem elhelyezésekor legfeljebb c-szer történhet ilyen vizsgálat.

A főprogramból vagy a Feltöltés eljárásból hívott Átpakolás eljárás első paramétere, l, második paramétere $2c+4-l$. $l = c+2+i$ alakú, ahol $1 \leq i \leq c$. A $2c+4-l$-nél nem nagyobb indexű ládaosztályok száma $c+2-i$. Az Átpakolás eljárások egymást követő rekurzív hívásai során a második paraméterek alkotta sorozat – nem szigorú értelemben ugyan, de – monoton csökkenő sorozatot alkot. Mivel egy adott listaelem elhelyezésekor az Átpakolás eljárás legfeljebb $c+1-i$-szer hívódik (beleértve a rekurzív hívásokat is), így – multiplicitásokkal számolva – ezekben összesen legfeljebb $(c+2-i)+(c+1-i) = 2c+3-2i$ számú kérdéses vizsgálat történik.

Ha az adott listaelemre az első Átpakolás eljáráshívás Feltöltés eljárásból történt, akkor $i \geq 2$ (hiszen ott a nagy láda ekkor osztályt is kellett, hogy váltson), és a Feltöltés eljárásban is pluszban legfeljebb $i-1$ darab ilyen vizsgálat volt. Ebben az esetben összesen tehát legfeljebb $2c+2-i$, azaz legfeljebb $2c$ darab vizsgálat.

Ha az Átpakolás a főprogramból hívódott, akkor a főprogramban egyetlen kérdéses összehasonlítás sem történt, ebben az esetben $i \geq 1$, így a kérdéses vizsgálatok száma ekkor összesen legfeljebb $2c + 1$. □

A három állítás következményeként kimondható az alábbi tétel:

3. Tétel. *A HFR-c algoritmus időbonyolultsága mind az input elemek mind az átpakolható elemek számában lineáris, pontosan* $T(\text{HFR-c}) = \mathcal{O}(cn)$.

A HFR-c algoritmus aszimptotikus versenyképessége

A HFR-c algoritmus AVK-jának vizsgálatakor két esetet különböztetünk meg attól függően, hogy az algoritmusnak egy konkrét inputon, egy L listán való lefuttatása után

- **A.**: a $\mathcal{B}_{c+3}, \ldots, \mathcal{B}_{2c+2}$ ládaosztályok mindegyike üres, vagy
- **B.**: a $\mathcal{B}_{c+3}, \ldots, \mathcal{B}_{2c+2}$ ládaosztályok valamelyike nem üres.

Az **A.** esetre a következő állítás mondható ki:

15. Lemma. *Ha egy L listainputon a HFR-c algoritmus leállásakor a $\mathcal{B}_{c+2+i}$, $i \in \{1, \ldots c\}$ ládaosztályok mindegyike üres, akkor HFR-c$(L) \leq \frac{1}{\frac{1}{2}+cb} OPT(L) + (c+2)$.*

BIZONYÍTÁS. Ebben az esetben minden láda legalább $\frac{1}{2} + cb$ szintig tele van, a csak *kis* elemeket tartalmazó ládaosztályok ($\mathcal{B}_1, \ldots, \mathcal{B}_{c+2}$) legfeljebb $1-1$ ládájától eltekintve. Az ilyen ládaosztályok száma pedig maximum $c + 2$. Ebből a Lemma állítása azonnal következik. □

A **B.** eset vizsgálatához az irodalomból jól ismert eszközt, az úgynevezett súlyfüggvény-technikát alkalmazzuk (lásd pl. [18, 108]). A súlyfüggvény definiálásakor az algoritmus AVK-jának elemzésében a következő állítás fontos szerepet fog játszani.

16. Lemma. *Ha a $\mathcal{B}_{c+2+i}$, $i \in \{1, \ldots c\}$ legalább egyike nem üres, és i^* a legkisebb ilyen tulajdonságú index, akkor a $\mathcal{B}_{c+2+i^*}$ összes komplemens osztálya üres, kivéve $\mathcal{B}_1$-et.*

BIZONYÍTÁS. A bizonyítás indirekt módon történhet. Tegyük fel, hogy $\mathcal{B}_{c+2+i^*}$-nak valamely komplemens osztálya nem üres, és az nem a $\mathcal{B}_1$ osztály. Legyen x a legnagyobb indexű ilyen osztály utolsó nemüres ládájának legfelső eleme. Mivel $\mathcal{B}_{c+2+i^*}$ nem üres, így tartalmaz legalább egy nemüres ládát. Legyen ez B.

Két eset lehetséges. Vagy az x érkezett hamarabb, mint ahogy B-be utoljára elemet pakoltunk (azaz B ezen formájának kialakulása, B tartalma utolsó változásának időpillanata előtt), vagy fordítva.

- Az első esetben az algoritmus rápakolta volna a kis elemet egy nagy ládára, hiszen akkor volt legalább egy, a feltételeknek megfelelő nagy láda.

- A második esetben a szóban forgó nagy, $\mathcal{B}_{c+2+i^*}$-beli ládára került volna átpakolásra egy kis elem, hiszen volt legalább egy átpakolható kis elem akkor.

Mindkét eset ellentmondáshoz vezet. □

Legyen $I_{2,1} = \{x \in I_2 \mid x \leq (i^* - 1)b\}$ és $I_{2,2} = \{x \in I_2 \mid x > (i^* - 1)b\}$, ahol i^* a 16. lemmabeli egész érték. A **B.** eset elemzésében használt súlyfüggvény pontos definíciója a következő:

$$w(x) := \begin{cases} \frac{1}{1-b}x, & x \in \{I_1 \cup I_{2,1}\}, \\ \frac{i^*-1}{1-b}b, & x \in \{I_{2,2} \cup I_3 \cup \ldots \cup I_{c+2-i^*}\}, \\ \frac{1}{2}, & x \in \{I_{c+3-i^*} \cup \ldots \cup I_{c+2}\}, \\ 1 - \frac{1}{1-b}(\frac{1}{2} + (i^* - 1)b - x), & x \in \{I_{c+3} \cup \ldots \cup I_{c+1+i^*}\} \ (= \emptyset, \textit{ ha } \ i^* = 1), \\ 1, & x \in \{I_{c+2+i^*} \cup \ldots \cup I_{2c+3}\}. \end{cases}$$

A HFR-c algoritmus aszimptotikus versenyképességének meghatározásához kimondunk néhány további lemmát.

17. Lemma. *Ha az algoritmus lefutása után van legalább egy olyan i index, $i \in \{1, \ldots, c\}$, hogy a $\mathcal{B}_{c+2+i}$ osztály tartalmaz legalább egy nemüres ládát, és $c + 2 + i^*$ a legkisebb ilyen index, akkor létezik egy olyan $0 \leq M_2 < \infty$ konstans, hogy bármely L listára $w(L) = \sum_{x \in L} w(x) \geq A(L) - M_2$.*

BIZONYÍTÁS. Azt fogjuk belátni, hogy – ládaosztályonként legfeljebb egy-egy ládától eltekintve – minden ládában a benne lévő elemekhez rendelt súlyok összege legalább 1. (Egy B láda $w(B)$ súlya alatt pontosan ezt, a benne elhelyezett elemekhez rendelt súlyok összegét értjük.)

- A $\mathcal{B}_1$ osztály ládái, legfeljebb 1 kivételtől eltekintve legalább $1-b$ szintig tele vannak. Itt minden elem súlya a méretének $\frac{1}{1-b}$-szerese, így a legfeljebb egy ládától eltekintve minden láda súlya legalább 1.
- A $\mathcal{B}_2$ osztály üres.
- A $\mathcal{B}_3, \ldots, \mathcal{B}_{c+2}$ osztályok közül amelyek nem üresek, azok ládái – osztályonként legfeljebb $1-1$ kivételtől eltekintve – pontosan két elemet tartalmaznak, mindkettő súlya $\frac{1}{2}$, így a láda súlya pontosan 1.
- A $\mathcal{B}_{c+2+i^*}, \ldots, \mathcal{B}_{2c+3}$ osztályok ládái – kivétel nélkül – pontosan egy „nagy elemet" tartalmaznak. Legyen B egy ilyen láda, és legyen x a benne lévő nagy elem. Két esetet különböztetünk meg. Ha $x > \frac{1}{2} + (i^* - 1)b$, akkor ennek súlya önmagában 1, így $w(B) \geq 1$. Ha $x \in \left(\frac{1}{2}, \frac{1}{2} + (i^* - 1)b\right]$, akkor mivel $szint(B) > \frac{1}{2} + (i^* - 1)b$, így a láda tartalmaz legalább $\frac{1}{2} + (i^* - 1)b - x$ összmennyiségű kis elemet, amelyek összsúlya legalább $\frac{1}{1-b} \cdot \left(\frac{1}{2} + (i^* - 1)b - x\right)$, így $w(B) \geq 1$ ekkor is. Azaz, mindkét esetben, minden láda összsúlya legalább 1 (kivétel nélkül).

Figyelembe véve, hogy a kivétel ládák száma az összes – nemüres – ládaosztályban legfeljebb $c+1$, így az $M_2 = c+1$ konstanssal a lemma állítása teljesül. □

18. Lemma. *Ha az algoritmus lefutása után létezik legalább egy olyan i index, $i \in \{1,\ldots,c\}$, hogy a $\mathcal{B}_{c+2+i}$ osztály tartalmaz legalább egy, nemüres ládát, akkor minden olyan S elemhalmazra, melyre $|S| := \sum_{x\in S} x \le 1$, a következő teljesül:*

$$w(S) = \sum_{x\in S} w(x) \le \frac{3}{2} + \frac{b}{1-b}.$$

BIZONYÍTÁS. A kulcs ennek a résznek a bizonyításához is a 16. Lemma.

Ha az S halmaz csak kis elemet tartalmaz, akkor a súlyfüggvény definíciójából látható, hogy minden S-beli x elemnek a súlya, $w(x) \le \frac{3}{2}x$. Ebből következően ekkor $w(S) \le \frac{3}{2}$.

Ha az S tartalmaz egy $\frac{1}{2}+cb$-nél nagyobb elemet, legyen ez x_1, akkor a halmazbeli többi elem összmérete kisebb, mint $\frac{1}{2}-cb$. Mivel $w(x_1) = 1$ és egy $\frac{1}{2}-cb$-nél kisebb x elem súlya, $w(x) \le \frac{1}{1-b}x$, így

$$w(S) := \sum_{x\in S} w(x) \le 1 + \sum_{x\in S, x\ne x_1} \frac{1}{1-b}x \le 1 + \frac{1}{1-b}\left(\frac{1}{2}-cb\right) =$$

$$= 1 + \frac{1}{1-b}\left(\frac{1}{2} - \frac{1}{2}b + \frac{1}{2}b - cb\right) = \frac{3}{2} - \frac{(c-\frac{1}{2})b}{1-b} \le \frac{3}{2} - \frac{b}{2(1-b)} < \frac{3}{2}.$$

Állításunkat már csak arra az esetre kell bizonyítani, amikor az S halmaz tartalmaz egy $(\frac{1}{2}, \frac{1}{2}+cb]$ intervallumbeli nagy elemet. Ekkor pontosan egy x_1 elemet tartalmaz onnan, és az S halmaz csak kis elemeket tartalmazhat x_1-en kívül.

Feltételünk szerint ekkor létezik olyan i pozitív egész ($1 \le i \le c$), hogy $x_1 \in I_{c+2+i}$. x_1 mérete alapján két esetet különböztetünk meg.

a) Az első eset, amikor $x_1 > \frac{1}{2} + (i^*-1)b$, azaz $i \ge i^*$. Az S-beli második legnagyobb elem x_2 mérete alapján szintén két további lehetőségünk van.

   – Az első lehetőség, hogy x_2 nem eleme az I_j, $j \in \{c+3-i,\ldots,c+2\}$ intervallumok egyikének sem, azaz a maradék x_1-től különböző S-beli elemek az I_j, $j \in \{1,\ldots,c+2-i\}$ intervallumba esnek. Az ilyen kis x elemek súlya, $w(x) \le \frac{1}{1-b}x$, így ekkor

$$w(S) = \sum_{x\in S} w(x) = w(x_1) + \sum_{x\in S, x\ne x_1} w(x) \le 1 + \frac{1}{1-b}\cdot\frac{1}{2} =$$

$$= 1 + \frac{1}{2-2b} = \frac{3-2b}{2-2b} = \frac{3}{2} + \frac{b}{2(1-b)}.$$

   – A második lehetőség az, hogy x_2 az I_j, $j \in \{c+3-i,\ldots,c+2\}$ intervallumok valamelyikébe esik. Ez az intervallum azonban kizárólag az I_{c+3-i}

intervallum lehet, mert ezek közül bármely más intervallumbeli elem méretéhez hozzáadva x_1-et 1-nél nagyobb értéket kapnánk. Viszont ekkor S x_1-en és x_2-n kívüli elemeinek összmérete legfeljebb $1 - x_1 - x_2 \le 1 - (\frac{1}{2} + (i-1)b) - (\frac{1}{2} - ib) = b$ lehet. (Ez egyben azt is jelenti, hogy ekkor az S halmaz x_1-en és x_2-n kívül legfeljebb csak az I_1 intervallum elemeit tartalmazhatja.) Így

$$w(S) = \sum_{x \in S} w(x) = w(x_1) + w(x_2) + \sum_{x \in S, x \ne x_1, x \ne x_2} w(x) \le 1 + \frac{1}{2} + \frac{b}{1-b} = \frac{3}{2} + \frac{b}{1-b}.$$

b) A második eset az, amikor $x_1 \le \frac{1}{2} + (i^* - 1)b$, azaz $x_1 \in I_{c+2+i}$, ahol $1 \le i < i^*$. Az S-beli második legnagyobb elem, x_2 mérete alapján szintén két lehetőségünk van.

– Ha $x_2 > \frac{1}{2} - i^* b$, akkor $w(x_2) = \frac{1}{2}$, és S-ben az összes többi (x_1-től és x_2-től különböző) x elem összmérete legfeljebb $\sum_{x \in S, x \ne x_1, x \ne x_2} x \le 1 - x_1 - x_2 < 1 - x_1 - (\frac{1}{2} - i^* b) = \frac{1}{2} + i^* b - x_1$.

Így

$$\begin{aligned} w(S) &= \sum_{x \in S} w(x) = w(x_1) + w(x_2) + \sum_{x \in S, x \ne x_1, x \ne x_2} w(x) \le \\ &\le 1 - \frac{1}{1-b} \cdot \left(\frac{1}{2} + (i^* - 1)b - x_1 \right) + \frac{1}{2} + \frac{1}{1-b} \left(\frac{1}{2} + i^* b - x_1 \right) = \\ &= \frac{3}{2} + \frac{b}{1-b}. \end{aligned}$$

– A másik lehetőség, hogy $x_1 \le \frac{1}{2} + (i^* - 1)b$, de $x_2 \le \frac{1}{2} - i^* b$. Ebben az esetben S bármely x_1-től különböző x elemére (x_2-t is beleértve) $w(x) \le \frac{1}{1-b} x$ teljesül. Így az a) eset első ágához hasonló módon $w(S) \le \frac{3}{2} + \frac{b}{2(1-b)}$. □

Most térünk vissza egy tetszőleges c-hez tartozó $b = b_c$ megválasztási módjának definiálására. A 15. lemmabeli egyenlőtlenség jobb oldalán szereplő szorzótényezőt egyenlővé téve a 18. lemmabeli egyenlőtlenség jobb oldalával, a b_c-t válasszuk úgy, hogy

$$\frac{2}{1 + 2cb_c} = \frac{3}{2} + \frac{b_c}{1 - b_c}$$

teljesüljön, és a $b_c \in \left(0, \frac{1}{6c}\right]$ feltétel szintén. Könnyen ellenőrizhető, hogy a fenti másodfokú egyenlet ekvivalens a $2cb_c^2 - (6c+3)b_c + 1 = 0$ egyenlettel, melynek van a megkívánt intervallumba eső gyöke, és pontosan egy ilyen van. A 10. táblázat mutat néhány különböző c-hez tartozó b_c értéket.

A fentiek alapján a következő tétel mondható ki:

4. Tétel. *Bármely pozitív, de rögzített $c \ge 1$ egész esetén AVK(HFR-c) $\le \frac{3}{2} + \frac{b_c}{1-b_c}$. Ha c tart végtelenbe, akkor AVK(HFR-c) tart $\frac{3}{2}$-hez.*

10. táblázat. Néhány különböző c-hez tartozó b_c értékek, és az ezen c-khez tartozó $AVK(HFR - c)$-re nyert alsó és felső korlátok, valamint ezek különbsége (valamennyi érték az első hat tizedesjegyig kiírva).

c	$\mathbf{b_c}$	**Alsó korlát**	**Felső korlát**	**Eltérés**
1	$0,113999$	$1,601704$	$1,628666$	$0,026962$
2	$0,067895$	$1,564496$	$1,572841$	$0,008345$
3	$0,048285$	$1,546743$	$1,550734$	$0,003991$
4	$0,037452$	$1,536580$	$1,538909$	$0,002329$
5	$0,030586$	$1,530026$	$1,531551$	$0,001524$
6	$0,025846$	$1,525457$	$1,526532$	$0,001074$
7	$0,022378$	$1,522092$	$1,522890$	$0,000797$
8	$0,019729$	$1,519511$	$1,520127$	$0,000615$
9	$0,017642$	$1,517469$	$1,517958$	$0,000489$
10	$0,015953$	$1,515813$	$1,516212$	$0,000398$
11	$0,014560$	$1,514444$	$1,514775$	$0,000330$
12	$0,013390$	$1,513293$	$1,513572$	$0,000278$
17	$0,009553$	$1,509505$	$1,509645$	$0,000140$
34	$0,004838$	$1,504826$	$1,504862$	$0,000056$
42	$0,003926$	$1,503918$	$1,503942$	$0,000024$
56	$0,002952$	$1,502961$	$1,502947$	$0,000013$
84	$0,001973$	$1,501971$	$1,501977$	$0,000006$
167	$0,000995$	$1,500994$	$1,500996$	$0,000001$

BIZONYÍTÁS. A 15. Lemmával beláttuk, hogy az **A.** esetbeli minden L listára létezik olyan $M_1(= c + 2)$ konstans, hogy *HFR-c*$(L) \le \frac{2}{1+2cb_c} \cdot OPT(L) + M_1$.

A **B.** esetben minden L listára a 18. Lemma miatt a lista $w(L)$ súlyára, amely a benne elhelyezett elemek súlyainak összegeként definiált, a következő teljesül:

$$w(L) = \sum_{x \in L} w(x) = \sum_{B \in OPT(L)} \sum_{x \in B} w(x) = \sum_{B \in OPT(L)} w(B) \le \left(\frac{3}{2} + \frac{b_c}{1 - b_c}\right) OPT(L).$$

Ugyanakkor a 17. Lemmából adódik, hogy a **B.** esetbeli L listákra létezik olyan M_2 konstans, hogy *HFR-c*$(L) \le w(L) + M_2$, amiből azt kapjuk, hogy *HFR-c*$(L) \le \left(\frac{3}{2} + \frac{b_c}{1-b_c}\right) \cdot OPT(L) + M_2$.

Az $M := \max\{M_1, M_2\} = c + 2$ választással adódik, hogy bármely L listára *HFR-c*$(L) \le \left(\max\{\frac{3}{2} + \frac{b_c}{1-b_c}, \frac{2}{1+2cb_c}\}\right) \cdot OPT(L) + M$.

Viszont b_c-t úgy konstruáltuk, hogy $\frac{2}{1+2cb_c} = \frac{3}{2} + \frac{b_c}{1-b_c}$, így a b_c érték fenti választása miatt, és mert $b_c \in \left(0, \frac{1}{6c}\right]$, a tétel állítása bizonyított. $\square$

A most következő tételben megadjuk a legjobb alsó korlátokat, amelyeket algoritmussorozatunkra bizonyítani tudtunk.

5. Tétel. *Bármely rögzített $c \in N^+$ esetén*

$$AVK(HFR\text{-}c) \geq \frac{1-2b_c}{(1-b_c)^2} \cdot \left(\frac{3}{2} + \frac{b_c}{1-b_c}\right).$$

BIZONYÍTÁS. Minden $t \in N^+$-ra konstruálunk egy $N = 4n + 8t - 3$ elemszámú L listát, ahol $n = 2t(\lfloor \frac{1}{b_c} \rfloor - 1)$.

Álljon L négy (rész)lista konkatenációjából, azaz legyen $L = L_1L_2L_3L_4$, ahol

$L_1 = (L_{11}L_{12}L_{13})^{2t}$, ahol a kitevő a konkatenált részlisták ismétlésszámát jelenti, és
L_{11}: $\lfloor \frac{1}{b_c} \rfloor - 1$ darab egyenlő, $b_c - 2\varepsilon$, méretű elem,
L_{12}: 1 darab $1 - b_c - \frac{n}{2t}(b_c - 2\varepsilon)$ méretű elem,
L_{13}: 3 darab ε méretű elem, ahol

$$\varepsilon < \min\left\{\frac{b_c}{n}, \frac{1}{6t+n-3}, \frac{2b_ct + \frac{nb_c}{2} - t}{n}\right\}.$$

L_2: $n - 3$ darab ε méretű elem,
L_3: n darab $\frac{1}{2} - b_c + \varepsilon$ méretű elem, és
L_4: n darab $\frac{1}{2} + \varepsilon$ méretű elem.

A HFR-c algoritmus úgy működik ezen az inputon, hogy L_1^* elemeit a Next Fit szabály szerint egy-egy $\mathcal{B}_1$-beli ládába pakolja. Ezek magassága pontosan $1 - b_c + 3\varepsilon$, így a következő L_1^* (azaz L_{11}) első eleme már nem fér bele. Így nyitnia kell egy új ládát. Ezért L_1 elemeinek NF szabály szerinti elpakolásához az algoritmus $2t = n/\lfloor \frac{1-b_c}{b_c} \rfloor$ darab ládát használ fel.

Ezután L_2 elemeit, mivel azok is I_1-ből valók, az algoritmusnak $\mathcal{B}_1$-be kell pakolnia. Mivel az utolsó $\mathcal{B}_1$-beli láda magassága $1 - b_c + 3\varepsilon$, így L_2 összes eleme az ε méretük definíciója miatt befér az utolsó, L_1 érkezésekor létrehozott ládába. Így az algoritmus bepakolja ebbe az NF szabály szerint.

Ezt követően az algoritmus elpakolja L_3 elemeit NF szabállyal $\mathcal{B}_{c+2}$-be, $n/2$ darab ládát felhasználva.

Végül, L_4 minden elemének érkezésekor az algoritmus nyit egy új ládát $\mathcal{B}_{c+3}$-ban, belepakolja az elemet, majd átpakol $\mathcal{B}_1$ utoljára megnyitott ládájából egy ε méretű elemet ebbe a ládába. Így ez újabb n darab ládát jelent.

A *HFR-c* algoritmus tehát összesen

$$HFR\text{-}c(L) = n + \frac{n}{2} + \frac{n}{\lfloor \frac{1-b_c}{b_c} \rfloor} \leq n\left(\frac{3}{2} + \frac{b}{1-b}\right)$$

ládát használ fel az L lista elpakolásakor.

Másrészt, mivel az elemek összmérete

$$n + 2t\left(1 - b_c - \frac{n}{2t}(b - 2\varepsilon)\right) + (6t + n - 3)\varepsilon \leq n + 2tb + 1,$$

így azt kapjuk, hogy

$$OPT(L) \leq n + \frac{nb}{\lfloor\frac{1}{b}\rfloor - 1} + 1 \leq n + \frac{nb}{\frac{1}{b} - 2} + 1 \leq n \cdot \left(1 + \frac{b^2}{1-2b}\right) + 1 = n\frac{(1-b)^2}{1-2b} + 1.$$

Ezért

$$\mathit{AVK(HFR\text{-}c)} = \lim_{n\to\infty} \frac{n(\frac{3}{2} + \frac{b}{1-b})}{n\frac{(1-b)^2}{1-2b} + 1} = \lim_{n\to\infty} \frac{\frac{3}{2} + \frac{b}{1-b}}{\frac{(1-b)^2}{1-2b} + \frac{1}{n}} =$$

$$= \left(\frac{3}{2} + \frac{b}{1-b}\right)\left(\frac{1-2b}{(1-b)^2}\right),$$

amivel a tétel állítása bizonyított. □

3. Következmény. *Bármely rögzített* $c \in N^+$*-ra*

$$\left(\frac{3}{2} + \frac{b_c}{1-b_c}\right)\left(\frac{1-2b_c}{(1-b_c)^2}\right) \leq \mathit{AVK(HFR\text{-}c)} \leq \frac{3}{2} + \frac{b_c}{1-b_c}.$$

A 10. táblázat mutatja a legfontosabb, különböző c-khez tartozó b_c, valamint az *AVK(HFR-c)*-re kapott felső- és alsó korlát értékeket. Habár az eltérés az általunk bizonyított alsó és felső korlát értékek között kicsi, már $c = 1$-re is, elméleti szempontból mégis érdekes, hogy különbségük csökkenthető-e, azaz javítható-e valamelyik érték.

4.5. A fejezet összegzése

Ebben a fejezetben a klasszikus on-line ládapakolási feladat minden lépésében legfeljebb c darab (c tetszőleges, rögzített konstans) elem átpakolását megengedő ún. c-átpakolásos szemi-on-line ládapakolási feladatot vizsgálva először az Ivkovič és Lloyd által 1996-ban adott alsó korlátot javítottuk meg. Konstrukciónk egy nemlineáris optimalizálási problémához vezetett, amelynek megoldását átalakítások után megadtuk. Modellünk alsó korlátjának megadásához a Lambert-féle W függvény tulajdonságait használtuk fel. Az alsó korlát érvényes a teljesen dinamikus ládapakolási feladat c-átpakolásos változatára is, valamint minden d dimenzióban ($d \geq 2$).

A 4.3. alfejezetben modellünk vizsgálatával arra a speciális esetre adtunk alsó korlátokat, amikor a különböző elemméretek száma egy előre adott p ($p \geq 2$, egész) számmal korlátozott felülről. Az adódó nemlineáris optimalizálási problémát globális optimalizálási technikával oldottuk meg, egy megbízható, intervallum-matematikán alapuló módszerrel meghatározva ennek konkrét, speciális eseteinek megoldásait. Ezzel megválaszoltunk egy, Gutin, Jensen és Yeo 2005-ös közleményében felvetett kérdést: a 2-batched ládapakolási feladat szintén legfeljebb p különböző elemméretet megengedő esetére javítva a $p \geq 3$ esetekben az általuk megadott alsó korlátokat és így bizonyitva, hogy azok nem optimálisak. (Megjegyezzük, hogy az általunk megadott alsó korlát értékek ezen feladat akár c-átpakolásos akár átpakolás nélküli változatában is érvényesek.) A konkrét p-khez

($p \geq 2$) a konstrukció működik, és a megadott alsó korlátok is érvényesek a másik két tárgyalt c-átpakolásos szemi-on-line ládapakolási feladatra is.

A 4.2. és a 4.3. alfejezetben adott és elemzett alsó korlát konstrukció és az adódó feladat megoldása egy szép kapcsolatot mutat az eredeti diszkrét optimalizálási feladat, és a folytonos globális optimalizálás között. Megjegyezzük továbbá, hogy konstrukciónk korábbi alsó korlát konstrukciókat általánosít.

A 4.4. alfejezetben egy algoritmussorozatot adtunk meg, minden pozitív egész c értékre egy HFR-c nevű algoritmust. Bizonyítottuk (4. Tétel), hogy a HFR-c algoritmusok aszimptotikus versenyképességi hányadosaiból képzett sorozat $1,5$-hez tart, ha c tart a végtelenbe. Két konkrét esetet érdemes kiemelni ezek közül: a $c = 2$ és a $c = 4$ eseteket. A $c = 2$ esetben a HFR-2 algoritmus aszimptotikus versenyképességi hányadosa kisebb, vagy egyenlő, mint $1,5728...$, ami a Harmonic++ nevű, legjobb ismert on-line algoritmusra Seiden által bizonyított aszimptotikus versenyképességi hányados ($1,58889$) alatt van. Továbbá $AVK(HFR - 2)$ kisebb, mint a Harmonic Fit típusú on-line ládapakolási algoritmusokra bizonyított $1,58333$-as alsó korlát, amely érvényes a Seiden által Super Harmonic Fit típusúnak nevezett algoritmusokra is. Megjegyezzük, hogy nemcsak a Harmonic++ algoritmus Super Harmonic Fit típusú, hanem az utóbbi 20 évben mindegyik „aktuálisan legjobb" algoritmus ilyen típusú volt. A $c = 1$ esetre adott algoritmusunk eredménye irreleváns abból a szempontból, hogy az aktuálisan legjobb on-line algoritmus [95] versenyképesebb. Megjegyezzük azonban, hogy a mi 1-átpakolásos algoritmusunk jóval kevesebb ládaosztályt használ, mint az. A $c = 4$ esethez tartozó HFR-4 algoritmus AVK-ja jobb van Vliet $1,5401$-es on-line algoritmusokra vonatkozó alsó korlátjánál.

Számos nyitott kérdés maradt. Ezekből hármat említünk meg. Az első, hogy adjunk lépésenként egy elem átpakolását megengedő, $1,58333$-nál kisebb AVK-jú szemi-on-line ládapakolási algoritmust! A második, hogy adjunk lépésenként 4-nél kevesebb elem átpakolását megengedő, $1,5401$-nél kisebb AVK-jú algoritmust! Továbbá nyitott a kis c értékekre (pl. $c = 1, 2$) az alsó korlátok javítása.

Záró megjegyzések. A fejezetben a 4.2. alfejezet eredményei Békési Józseffel közös eredmények. Ez alól kivétel a 6. Lemma, amely Galambos Gábor, míg a 7. Lemma alapötlete Markót Mihály Csaba nevéhez fűződik. Ez az alfejezet a SIAM Journal on Computingban megjelenés alatt lévő publikációra épül.

A 4.3. alfejezet Békési Józseffel és Markót Mihály Csabával közös eredményeket, míg a 4.4. alfejezet saját eredményeket tartalmaz. Az eredmények végső technikai megformálásában, leírásukban és megszövegezésében valamennyi szerzőtársam fontos szerepet játszott.

Összefoglalás

A globális optimalizálás feladata egy (általában kompakt) halmazon értelmezett (általában valós értékű) függvény minimumának (vagy ami ezzel hasonlóan kezelhető maximumának) meghatározása. A dolgozatban mindegyik tárgyalt témánál globális optimalizálási módszerek alkalmazásai is tárgyalásra kerültek.

Globális optimizálizálás Stiefel–sokaságokon

Az 1. fejezetben egy Stiefel–sokaságokon adott kvadratikus optimalizálási feladat megoldására globális optimalizálási módszereket és technikákat tárgyaltunk.

Az $n = 2$, $k = 2$ esetben megadtuk az optimumpontokat, illetve kritériumot adtunk az optimumpontok számának végességére. A diagonális együtthatómátrixú esetben az optimumpontok koordinátái a $\{-1, 0, +1\}$ halmazból kerülnek ki (kivéve azt az elfajuló esetet, amikor az összes lehetséges megoldás egyben optimális is).

A feladatot ezután numerikus szempontból vizsgáltuk meg, intervallumos globális optimalizálási eszközzel, és hagyományos sztochasztikus globális optimalizálással is. A feladat megoldása megbízható módon nagy számításigényű, már az $M_{2,2}$, vagy az $M_{3,3}$ Stiefel–sokaságon definiált feladat megoldása is. Az utóbbi több napnyi CPU-időt jelenthet. A tapasztalati eredmények azt mutatják, hogy a numerikus nehézségek ebben az esetben a korlátozó feltételek ellenőrzésében rejlenek. Ez az oka annak, hogy a felgyorsítási lehetőségeket érdemes elméletileg vizsgálni, akár geometriai redukciókkal akár numerikus módszerekkel. Ehhez megfelelő tesztfeladatok szükségesek.

A dolgozatban a gyakorlati, számítógépes vizsgálatok támogatására ismert optimális megoldásokkal rendelkező tesztproblémákat javasoltunk. Ezek a numerikus eszközök tesztelésére, hatékonyságának vizsgálatára alkalmazhatók.

A fejezet másik fő elméleti eredményeként a lehetséges megoldások halmazának egy érdekes megszorításával keletkező feladatot definiáltuk és vizsgáltuk. Bizonyítottuk, hogy ez a hozzárendelési feladattal ekvivalens feladatot ad, amely minimumpontjainak száma véges, bár exponenciális k-ban.

Az RPCRS: egy párhuzamos sztochasztikus (klaszterező) globális optimalizálási algoritmus

A globális optimalizálás hagyományos determinisztikus és sztochasztikus direkt keresőket alkalmazó algoritmusain kívül léteznek megbízható intervallum-matematikán alapuló módszerek, illetve az előzőek kevert (adaptív), illetve heurisztikus változatai is. Nagy számításigényű feladatok esetén indokolt ezek párhuzamos változatainak használata.

W.L. Price 1978-ban javasolt egy, az akkori számítógépeken és PC-ken kiváló futási eredményekkel bíró klaszterező sztochasztikus algoritmust, a CRS-t (Controlled Random Search). Ennek néhány változata ma is használt, főleg nagy számításigényű feladatoknál. Egy párhuzamos változata, a PCRS (Parallel CRS) – amit García és munkatársai adtak meg 1995-ben – hatékonyan alkalmazható képfeldolgozási területről származó globális optimalizálási problémák megoldására.

A 2. fejezet a PCRS egy új, buffert használó változatának leírása és elemzése. Ezen aszinkron párhuzamos megvalósítás, az RPCRS – amint a teszteredmények azt mutatják – sikeresen alkalmazható olyan feladatok esetén, amelyeknél a célfüggvény kiértékelése nagy számításigényű.

Az RPCRS módszer lényege, hogy a szolgaprocesszorok egy buffert használnak a kiértékelendő pontok tárolására, és – mint ezt numerikusan megmutattuk – ez a stratégia azt eredményezi, hogy a szolgaprocesszorok várakozási ideje csökkenthető (akár eliminálható), hiszen a kommunikációs pluszköltség (az „overhead") csökken, mivel a kommunikáció és az effektív számítás ideje átlapolódik az aszinkronitásnak köszönhetően.

Az RPCRS abban a speciális esetben, amikor a buffer mérete 1, akkor visszaadja a PCRS-t, és amikor pedig még az is teljesül, hogy a processzorok száma 1, akkor pedig a CRS-t. Az RPCRS által elért sebességnövelés ("speed up") lehetőségeinek tanulmányozására az RPCRS szekvenciális algoritmusát – ekkor a processzorszám 1, a bufferméretre pedig nincs megkötés – azaz az RPCRS(b,1)-et is teszteltük.

Mindkét verzióra, a párhuzamos RPCRS(b,p)-re, és a szekvenciális RPCRS(b,1)-re gépfüggetlen implementáció adott és széles körből választott 22 tesztfüggvényen tesztelt: ezen belül az előbbi egy CRAY T3D szuperszámítógépen, 16 processzorig.

Az implementáció sikerességét igazolja, hogy a CRS-sel összehasonlítva az RPCRS(b,p) párhuzamos verzióval elért sebességnövelés a lineárisnál is jobb $(b = 16,\ 64)$. A sebességnövelés mértéke az egyprocesszoros RPCRS(b,1) implementációval összehasonlítva is majdnem lineáris, viszont ebben az esetben már nem szuperlineáris.

Fázisstabilitási problémák

Az utóbbi időben nagy érdeklődés övezi új (mind elméleti mind algoritmikus) megoldási módszerek kifejlesztését vegyipari műveletek tervezési és optimalizálási feladataira. Ennek az utóbbi évtizedben intenzíven kutatott részterülete,

– számos publikációval – a fázisstabilitás elemzésének problémája.

Ennek szokásos megoldási módszere az érintősík-távolsági függvény vagy a szabadenergia minimalizálása. Mindkettő megbízható numerikus technikákat igényel a globális megoldás meghatározásához. Lényegében ez a két megközelítés használható annak igazolására, hogy egy egyensúlyi megoldás megfelel a szabadenergia minimumának; azaz közvetlenül a szabadenergia minimalizálása, vagy az érintősík-távolsági függvény minimalizálása (érintősík-kritérium). A második megközelítés nagyon hatékony lehet, valódi egyensúlyi megoldás megtalálási esélyének növelésében, mint ahogyan azt McDonald és Floudas találta. Ők voltak az elsők, akik 1994-95-ös munkáikban átfogalmazták az alapproblémát a szabadenergia minimalizálására és az érintősík-kritériumra, a fázis- és kémiai reakcióegyensúlyt egy globális optimalizálási problémaként fölfogva.

Mi egy új megközelítést alkalmaztunk a fázisstabilitási probléma megoldására. Egy, Stateva és Tsvetkov által bevezetett új célfüggvény, az érintősík-távolsági függvény egy módosításának optimalizálását használtuk fel erre, és egy hatékonynak bizonyult globális optimalizálási módszert vizsgáltunk.

A leglényegesebb lépés technikánkban a módosított érintősík-távolsági függvény, mint célfüggvény összes zérushelyének meghatározása. A függvény olyan, hogy ennek zérushelyei pontosan a minimumhelyei is egyben. Lényeges az összes ilyen pont meghatározása, hiszen csak ebben az esetben adódik korrekt válasz a fázis stabil, illetve nem stabil voltára. Optimalizálási szempontból ez azt jelenti, hogy olyan módszert kell használni, amely erre képes ilyen, multimodális függvényeknél. Erre C.G.E. Boender és szerzőtársai klaszterező, sztochasztikus algoritmusának a hasonló, ismert módszerek között legjobbnak talált, Csendes Tibor nevéhez fűződő implementációját használtuk, illetve tettük megbízhatóbbá bizonyos beállítási hangolásokkal.

A dolgozatban először a célfüggvényt és a módszert írtuk le röviden; majd alkalmazott globális optimalizálási eljárásunk robusztusságát és hatékonyságát demonstráltuk konkrét rendszerek termodinamikai stabilitás-elemzésének példáján keresztül; a kénhidrogén + metán rendszer illetve a nitrogén + metán + etán rendszereken.

A megbízhatóság mértéke, azaz a sikerességi ráta magas foka tűnik ki az eredményekből. Megadtuk a nyert eredmények elemzését. Ezek más, eltérő módszert használo szerzők eredményeivel összehasonlítva ugyan kedvezőnek tűnnek, explicit összehasonlítást azonban nem adtunk meg, éppen a módszerek eltérő jellege miatt.

Korlátok szemi-on-line ládapakolási feladatokhoz

A 4. fejezetben alsó és felső korlátokat adtunk meg egy speciális, úgynevezett „szemi-on-line" ládapakolási feladatra. Vizsgálatainkban aszimptotikus legrosszabb eset vizsgálatra szorítkoztunk, a szakirodalomban szokásos módon definiált aszimptotikus legrosszabb eset hányadossal (= aszimptotikus versenyképességi hányados). A tárgyalt feladat annyiban tér el a klasszikus on-line, egydimenziós ládapakolási feladattól, hogy minden egyes lépésben (egy lépésnek az input lista egy új elemének érkezése számít) egy előre rögzített c konstanssal korlátozott számú, korábban elpakolt elem átpakolása is megengedett.

A problémához először alsó korlátot adtunk a 4.2. szakaszban. Konstrukciónk egy szép nemlineáris optimalizálási problémához vezetett. Egy, $1,3871$-es alsó korlátot kaptunk bármely ilyen típusú algoritmus aszimptotikus legrosszabb eset hányadosára. Az alsó korlátot egy szélsőértékfeladat megoldásán keresztül a Lambert-féle W függvény egy értékével fejeztük ki. Eredményünk javítja a problémára az Ivkovič-Lloyd szerzőpáros által 1996-ban megadott $\frac{4}{3}$-os alsó korlátot. Ezt ők az úgynevezett teljesen dinamikus ládapakolási feladat esetére bizonyították, de könnyen átvihető a lépésenként konstans számú elem átpakolását megengedő szemi-on-line esetre is. Megmutattuk, hogy az általunk adott alsó korlát szintén érvényes a teljesen dinamikus ládapakolási feladat megfelelő esetére. Bizonyítottuk, hogy ugyanez a korlát minden d ($d \geq 2$) dimenzióban is érvényes.

Ha rögzítjük a megengedett különböző elemméretek maximális számát, legyen ez p, akkor a konkrét p értékekre is megoldhatjuk a problémát, globális optimalizálási eszközöket felhasználva az így nyert nemlineáris optimalizálási probléma megoldására. Az ennek megoldásával adódó alsó korlátok megválaszolnak egy Gutin és szerzőtársai által 2005-ös cikkükben felvetett kérdést, hogy az úgynevezett „batched" ládapakolási feladatra, két batch esetén (2-BBP probléma) az általuk megadott korlátok $p \geq 3$ esetén optimálisak-e. Bizonyítottuk, hogy nem. A bizonyítás lényege, hogy konstrukciónk általánosítja az általuk alkalmazottat, mert a mi listáinknál alkalmazott elemméretekre nincs előírva, hogy egy intervallum ekvidisztáns beosztásából származzanak. Az itt megadott alsó korlátok szintén érvényesek a másik két tárgyalt szemi-on-line ládapakolási feladat hasonló, legfeljebb p számú különböző elemméretet megengedő speciális esetére.

A 4.4. alfejezetben felső korlátokat adtunk a c-átpakolásos szemi-on-line problémára, egy dimenzióban. Míg az alsó korlátok bármely, pozitív egész c értékre érvényesek, addig itt minden c-re egy-egy, a c paraméterhez tartozó, HFR-c nevű algoritmus definiált. Az így megkonstruált algoritmussorozatban az algoritmusok aszimptotikus versenyképessége is függ a c értéktől, és bizonyítottuk, hogy az ezekből álló sorozat monoton csökkenve tart $1,5$-hez, ha c tart végtelenbe. Az algoritmusok közül kiemelendőek a $c = 2$ és a $c = 4$ esetre megadott algoritmusok. A HFR-2 és HFR-4 algoritmusok az általunk megadott algoritmussorozatban az első olyanok, amelyek aszimptotikus versenyképessége jobb, mint az on-line algoritmusok aktuálisan legjobb felső, illetve alsó korlátja. Ez azt jelenti, hogy már kis c értékek esetén is jól ki lehet használni az on-line feladat meglazítása miatt a rendelkezésünkre álló plusz információt.

Summary

The present work contains some results in the field of applications of global optimization and some on semi on-line bin packing. The four sections of this dissertation can be summarized in the following.

In Section 1 a computational approach of global optimization on Stiefel–manifolds is investigated. The optimization on Stiefel manifolds was discussed by Rapcsák in earlier papers. Here, some methods of global optimization are dealt with and tested on Stiefel manifolds. The structure of the optimizer points of a quadratic problem is studied theoretically and numerically for the lowest interesting dimensional case, as well as the criterion for the finiteness of the number of optimizer points. Then possible reduction tricks are examined together with a numerical study. In the computational investigation we also focused on a special case of the problem, namely when the coefficient matrices in the objective function are diagonal. As the main result of the section test functions are given with known global optimum points and their optimal function values, and a restriction, which leads to a discretization of the problem, which results in a problem equivalent to the well-known assignment problem.

Section 2 dealt with a stochastic global optimization algorithm, called CRS (Controlled Random Search), which originally was devised as a sequential algorithm. Our work was intended to analyze the degree of parallelism that can be introduced into CRS and to propose a new refined parallel CRS algorithm (RPCRS). As a first stage, evaluations of RPCRS were carried out by simulating parallel implementations. The degree of parallelism of RPCRS is controlled by a user given parameter whose value must be tuned to the size of the parallel computer system. It is shown that the greater the degree of parallelism the better the performance of the sequential and parallel executions.

In Section 3 we applied a stochastic clustering optimization method, called GLOBAL to phase stability analysis. This is fundamental importance in various chemical engineering applications, such as azeotropic and three-phase distillation, supercritical extraction, petroleum and reservoir engineering, etc. Phase stability is often tested using the tangent plane criterion, and a practical implementation of this criterion is to minimise the tangent plane distance function (TPDF), defined as the vertical distance between the molar Gibbs energy surface and the tangent plane for the given phase composition. In the present work we used a modified TPDF and an equation of state as the thermodynamic model. We tested the efficiency and reliability of the advocated stochastic sampling and clustering method on some benchmark problems of the related literature, tuning some parameters of

the original method for this. The method is user-friendly and not computationally demanding regarding the number of function-evaluations, and CPU time.

In Section 4 new lower and upper bounds for semi-on-line bin packing problems are given. Semi-on-line algorithms for the bin-packing problem allow, in contrast to pure on-line algorithms, to execute one of certain types of additional operations in each step as repacking, reordering or buffering some elements before packing them. In 1996 Ivkovič and Lloyd gave the lower bound $\frac{4}{3}$ on the asymptotic worst-case ratio for the so-called fully dynamic bin packing algorithms, where the number of repackable items in each step is restricted by a constant. We improved this result to about 1.3871. We presented our proof for a semi-on-line case of the classical bin packing, but it works for fully dynamic bin packing as well. We proved the lower bound by analyzing and solving a specific optimization problem. The bound can be expressed exactly using Lambert's W function. Some lower bound results for the special case of the problem are given as well. After this, some upper bounds is given; we defined and analyzed a "c-repacking" semi-on-line algorithm for this semi-on-line case of the classical bin packing problem, where in each step it is allowed to repack at most c elements, for some positive integer c. We proved that the asymptotic competitive ratio for a given c is not larger than $\frac{3}{2} + \frac{b_k}{1-b_c}$ where b_c is in the interval $(0, \frac{1}{6c})$.

Irodalomjegyzék

[1] Baker, L.E., A.C. Pierce, and K.D. Luks, Gibbs energy analysis of phase equilibria, *Soc. Petrol. Eng. J.* **22**(1982):731.

[2] Balogh, J., Global optimization on Stiefel manifolds – some particular problem instances, *Proceedings of the 6th International Conference on Applied Informatics*, Eger, 2004, 259–268.

[3] Balogh, J., J. Békési, G. Galambos, and M.Cs. Markót, Improved lower bounds for semi-on-line bin packing problems. Közlésre benyújtva, 2007. Elérhető: `http://www.jgytf.u-szeged.hu/~balogh/babegama.ps`.

[4] Balogh, J., J. Békési, G. Galambos and G. Reinelt, Lower bound for the on-line bin packing problem with restricted repacking. *SIAM Journal on Computing*, megjelenés alatt, 2007. Elérhető: `http://www.jgytf.u-szeged.hu/~balogh/crestbinp.pdf`.

[5] Balogh, J., R.J.B. Craven, and R.P. Stateva, The area method for phase stability analysis revisited: further developments. Formulation in terms of the convex envelope of thermodynamic surfaces, *Ind. Eng. Chem. Res.*, **46**(2007):1611–1631.

[6] Balogh, J., T. Csendes, and T. Rapcsák, Some global optimization problems on Stiefel manifolds, *Journal of Global Optimization*, **30**(2004):91–101.

[7] Balogh J., Csendes T. és Rapcsák T., Globális optimizálás Stiefel-sokaságokon – egy érdekes diszkretizálási eredmény, *Alkalmazott Matematikai Lapok*, **22**(2005):163–176.

[8] Balogh J., Csendes T. és R.P. Stateva, A fázisstabilitás elemzése egy új célfüggvény és egy globális optimalizálási módszer fölhasználásával, *Magyar Kémiai Folyóirat*, **107**(2001):82–90.

[9] Balogh, J., T. Csendes, and R.P. Stateva, Application of a stochastic method to the solution of the phase stability problem: cubic equations of state, *Fluid Phase Equilibria*, **212**(2003):257–267.

[10] Balogh J. és Galambos G., Átpakolást használó szemi-on-line ládapakolási algoritmusok, *Alkalmazott Matematikai Lapok*, **24**(2007):117–130.

[11] Balogh, J. and B. Tóth, Global optimization on Stiefel manifolds: a computational approach, *Central European Journal of Operations Research*, **13**(2005):213–232.

[12] Bánkövi, Gy., J. Veliczky, and M. Ziermann, Dynamic models for prediction of the development of national economies, in *Models and decision making in national economies*, eds.: Janssen, J.M.L., L.F. Pau, and A. Straszak, North-Holland, Amsterdam, 1979, 257–267.

[13] Bánkövi, Gy., J. Veliczky, and M. Ziermann, Dynamic factor analysis, *Karl Marx University of Economics*, Budapest, 1982, 81.

[14] Boender, C.G.E., A.H.G. Rinnooy Kan, G.T. Timmer, and L. Stougie: A stochastic method for global optimization, *Mathematical Programming*, **22**(1982):125–140.

[15] Bolla, M., G. Michaletzky, G. Tusnády, and M. Ziermann, Extrema of sums of heterogeneous quadratic forms, *Linear Algebra and its Applications*, **269**(1998):331–365.

[16] Casado, L.G., I. García, and T. Csendes, A new multisection technique in interval methods for global optimization, *Computing*, **65**(2000):263–269.

[17] Casado, L.G., I. García, and T. Csendes, A heuristic rejection criterion in interval global optimization algorithms, *BIT*, **41**(2001):683–692.

[18] Coffman, E.G., G. Galambos, S. Martello, and D. Vigo, Bin packing approximation algorithms: Combinatorial analysis, in *Handbook of Combinatorial Optimization Supplement Volume*, eds.: Du, D.-Z. and P.M. Pardalos, Kluwer, Dordrecht, 1999, 151–208.

[19] Coffmann, E.G., M.R. Garey, and D.S. Johnson, Dynamic bin packing, *SIAM Journal on Computing*, **12**(2):227–260, 1983.

[20] Corless, R.M., G.H. Gonnet, D.E.G. Hare, D.J. Jeffrey, and D.E. Knuth, On the Lambert W function, *Advances in Computational Mathematics*, **5**(1996):329–359.

[21] Csallner A.E., T. Csendes, and M.Cs. Markót, Multisection in interval branch-and-bound methods for global optimization I. Theoretical results, *Journal of Global Optimization*, **16**(4):371–392, 2000.

[22] Csendes, T., A GLOBAL szubrutin felhasználói leírása. *A Kalmár László Kibernetikai Laboratórium Közleményei*, 1986/9., JATE, Szeged.

[23] Csendes, T., Nonlinear parameter estimation by global optimization – efficiency and reliability, *Acta Cybernetica*, **8**(1988):361–370.

[24] Csendes, T., B.M. Garay, and B. Bánhelyi, A verified optimization technique to locate chaotic regions of a Hénon system, *Journal of Global Optimization*, **35**:145–160, 2006.

[25] Csirik, J. and G.J. Woeginger, On-line packing and covering problems, in: *On-line algorithms, Lecture Notes in Computer Science*, eds.: Fiat, A. and G. Woeginger, Vol. 1442, Springer-Verlag, Berlin, 1998, 147–177.

[26] Dixon L.C.W. and G.P. Szegő, *Towards Global Optimisation*, North-Holland, Amsterdam, 1975.

[27] Duckbury, P.G., *Parallel Array Processing*. Chichester: Ellis Horward, 1986.

[28] Edelman, A., T.A. Arias, and S.T. Smith, The geometry of algorithms with orthogonality constraints, *SIAM Journal on Matrix Analysis and Applications*, **20**(1998):303–353.

[29] Fernandez de la Vega, W. and G.S. Lueker, Bin packing can be solved within $1 + \varepsilon$ in linear time, *Combinatorica*, **1**(1981):349–355.

[30] Floudas, C.A., Recent advances in global optimization for process synthesis, design and control: enclosure of all solutions, *Comput. Chem. Engng., Supplement* S 963-S 973, 1999.

[31] Floudas, C.A. and P.M. Pardalos, *A Collection of Test Problems for Constrained Global Optimization Algorithms*, Lecture Notes in Computer Science 455, Springer-Verlag, Berlin, 1990.

[32] Floudas, C.A., P.M. Pardalos, C.S. Adjiman, W.R. Esposito, Z.H. Gumus, S.T. Harding, J.L. Klepeis, C.A. Meyer, and C.A. Schweiger, *Handbook of Test Problems for Local and Global Optimization*, Kluwer, Dordrecht, 1999.

[33] Galambos, G., *A new heuristic for the classical bin packing problem*, Technical Report 82, Institute für Mathematik, Universitäet Augsburg, 1985.

[34] Galambos, G. and G.J. Woeginger, Repacking helps in bounded space on-line bin packing, *Computing*, **49**(1993):329–338.

[35] Gambosi, G., A. Postiglione, and M. Talamo, New algorithms for on-line bin packing, in *Algorithms and Complexity, Proceedings of the First Italian Conference*, eds.: Petreschi R., G. Aussiello, D.P. Bovet, World Scientific, Singapore, 1990, 44–59.

[36] Gambosi, G., A. Postiglione, and M. Talamo, Algorithms for the relaxed online bin-packing model, *SIAM Journal on Computing*, **30**(5):1532–1551, 2000.

[37] García, I. and G.T. Herman, Global optimization by parallel constrained biased random search, in *State of Art in Global Optimization: Computational Methods and Applications*, eds.: Floudas, C.A. and P.M. Pardalos, Kluwer, Dordrecht, 1996, 433–455.

[38] García, I., P.M. Ortigosa, L.G. Casado, G.T. Herman, and S. Matej, A parallel implementation of the controlled random search algorithm to optimize an algorithm for reconstruction from projections, *IIIrd Workshop on Global Optimization*, (Szeged, Hungary), December 1995, 28–32.

[39] García, I., P.M. Ortigosa, L.G. Casado, G.T. Herman, and S. Matej, Multidimensional optimization in image reconstruction from projections, in *Developments in Global Optimization*, eds.: Bomze L.M., T. Csendes, R. Horst, and P.M. Pardalos, Kluwer, Dordrecht, 1997, 289–300.

[40] Garey, M.R. and D.S. Johnson, *Computers and Intractability (A Guide to the theory of NP-Completeness)*. W.H. Freeman and Company, San Francisco, 1979.

[41] Geweke, J.F., The dynamic factor analysis of economic time series models, in *Latent variables in socio-economic models*, eds.: Aigner, D.J. and A.S. Goldberger, North-Holland, Amsterdam, 1977, 365–382.

[42] Gill, P.E., W. Murray, and M.H. Wright, *Practical optimization*, Academic Press, London, 1981.

[43] Gould, N. and P.L. Toint, Preprocessing for quadratic programming, *Mathematical Programming*, **100**(2004):95–132.

[44] Grove, E.F., On-line bin packing with lookahead, *Proceedings of the Sixth Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, ACM, New York, SIAM, Philadelphia, 1995, 430–436.

[45] Gutin, G., T. Jensen, and A. Yeo, Batched bin packing, *Discrete Optimization*, **2**(1): 71–82, 2005.

[46] Gutin, G., T. Jensen, and A. Yeo, Optimal on-line bin packing with two item sizes, *Algorithmic Oper. Research*, **1**(2006): 72–78.

[47] Hammer, R., M. Hocks, U. Kulisch, and D. Ratz, *Numerical Toolbox for Verified Computing I.*. Springer-Verlag, Berlin, 1993.

[48] Hansen, E., *Global Optimization Using Interval Analysis*. Marcel Dekker, New York, 1992.

[49] Harding, S.T. and C.A. Floudas, Phase stability with cubic equations of state: Global optimization approach, *AIChE Journal*, **46**(7):1422–1440, 2000.

[50] Helmke, U. and J.B. Moore, *Optimization and Dynamical Systems*, Springer-Verlag, Berlin, 1994.

[51] Hendrix, E.M.T., *Global Optimization at Work*, PhD. Dissertation, Wageningen Agricultural University, 1998.

[52] Higham D.J. and N.J. Higham, *MATLAB Guide*, SIAM, Philadelphia, 2000.

[53] Horst, R. and P.M. Pardalos, *Handbook of Global Optimization*, Kluwer, Dordrecht, 1995.

[54] Hua, J.Z., J.F. Brennecke, and M.A. Stadtherr, Reliable prediction of phase stability using an interval-Newton method, *Fluid Phase Equilibria*, **116**(1996):52–59.

[55] Hua, J.Z., J.F. Brennecke, and M.A. Stadtherr, Reliable computation of phase stability using interval analysis: cubic equation of state models, *Comput. Chem. Engng.*, **20**(1996) 1207–1214.

[56] Hua, J.Z., J.F. Brennecke, M.A. Stadtherr, Enhanced interval analysis for phase stability: Cubic equation of state models, *Fluid Phase Equilibria*, **158-160**(1999):607–615.

[57] Hua, J.Z., R.W. Maier, St.R. Tessier and J.F. Brennecke, and M.A. Stadtherr, Interval analysis for thermodynamic calculations in process design: a novel and completely reliable approach, *Ind. Eng. Chem. Res.*, **37**(1998):1519–1527.

[58] Ivkovič, Z. and E.L. Lloyd, A fundamental restriction on fully dynamic maintenance of bin packing, *Information Processing Letters*, **59**(4):229–232, 1996.

[59] Ivkovič, Z. and E.L. Lloyd, Fully dynamic algorithms for bin packing: being (mostly) myopic helps, *SIAM Journal on Computing*, **28**(2): 574-611, 1998.

[60] James, J.M., *The Topology of Stiefel Manifolds*, London Mathematical Society, Lecture Notes Series 24, Cambridge University Press, Cambridge, 1976.

[61] Järvi, T., A random search optimizer with an application to a maxmin problem, *Publications of the Inst. of Appl. Math.,* Univ. of Turku, No. 3 (1973).

[62] Karmarkar, N. and R. Karp, An efficient approximation scheme for the one-dimensional bin packing problem, *Proc. 23rd Annual Symposium Foundation Computer Science (FOCS)*, IEEE Computer Society, Los Alamitos, CA, 1982, 312–320.

[63] Kearfott, R.B., *Rigorous Global Search: Continuous Problems*. Kluwer, Dordrecht, 1996.

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

[65] Knüppel, O., *PROFIL – Programmer's Runtime Optimized Fast Interval Library*. Bericht 93.4., Technische Universität Hamburg-Harburg, 1993.

[66] Kuhn, H.W., The Hungarian method for the assignment problem, *Naval Research Logistics Quarterly*, **2**(1955):83–97.

[67] Lewis, R.M., V. Torczon, and M.W. Trosset, *Why pattern search works*, OPTIMA, **59**(1998):1–7.

[68] Markót, M.Cs. and T. Csendes, A new verified optimization technique for the "packing circles in a unit square" problems, *SIAM Journal on Optimization*, **16**(2005):193–219.

[69] Markót, M.Cs., T. Csendes, and A.E. Csallner, Multisection in interval branch-and-bound methods for global optimization. II. Numerical tests. *Journal of Global Optimization*, **16**(3):219–228, 2000.

[70] Markót, M.Cs., J. Fernandez, L.G. Casado, and T. Csendes, New interval methods for constrained global optimization, *Mathematical Programming*, **106**(2006):287–318.

[71] Márkus, L., O. Berke, J. Kovács, and W. Urfer, Spatial prediction of the intensity of latent effects governing hydrogeological phenomena, *Environmetrics*, **10**(1999):633–654.

[72] McDonald, C.M. and C.A. Floudas, Decomposition based branch and bound global optimization approaches for the phase equilibrium problem, *J. Global Optimization*, **5**(1994):205–251.

[73] McDonald, C.M. and C.A. Floudas, Global optimization for the phase and chemical equilibrium problem: application to the NRTL equation, *Comput. Chem. Engng.*, **19**(1995):1111–1141.

[74] McDonald, C.M. and C.A. Floudas, Global optimization for the phase stability problem, *AIChE J.*, **41**(1995):1798–1814.

[75] McKeown, J.J., Aspects of parallel computations in numerical optimization, in *Numerical Techniques for Stochastic Systems*, eds.: Arcetti, F. and M. Cugiani, 1980, 297–327.

[76] Michelsen, M.L., The isothermal flash problem. Part I: Stability. *Fluid Phase Equilibria* **9**(1982):1–21.

[77] Moore, R. E., *Interval Analysis*. Prentice–Hall, Englewood Cliffs, 1966.

[78] Nelder, J.A. and R. Mead, A simplex method for function minimization, *The Computer Journal*, 1965, 308–313.

[79] Neumaier, A., A comparison of stochastic global optimization programs, Elérhető: `http://solon.cma.univie.ac.at/~neum/glopt/test_results.html`.

[80] Ortigosa, P.M., J. Balogh, and I. García, A parallelized random search global optimization algorithm, *Acta Cybernetica* **14**(1999):403–418.

[81] Pardalos, P.M., F. Rendl, and H. Wolkowicz, *The quadratic assignment problem: A survey and recent developments*, Proceedings of the DIMACS Workshop on Quadratic Assignment Problems, AMS, 1994, 1–41.

[82] Peng, D.Y. and D.B. Robinson, A New Two-Constant Equation of State. *Industrial and Engineering Chemistry: Fundamentals*, **15**(1976): 59–64.

[83] Price, W. L., A controlled random search procedure for global optimization, in *Towards Global Optimization 2*, eds.: Dixon, L.C.W. and G.P. Szegő, North-Holland, Amsterdam, 1978, 71–84.

[84] Price, W. L., A controlled random search procedure for global optimization, *The Computer Journal*, **20**(1979):367-370.

[85] Price, W.L., Global optimization algorithms by controlled random search, *Journal of Optimization Theory and Applications*, **40**(1983):333–348,.

[86] Price, W.L., Global optimization algorithms for a CAD workstation, *Journal of Optimization Theory and Applications*, **55**(1987):133–146.

[87] Ramanan, P., D.J. Brown, C.C. Lee, and D.T. Lee, On-line bin packing in linear time, *Journal of Algorithms*, **10**(3):305–326, 1989.

[88] Rapcsák, T., On minimization of sums of heterogeneous quadratic functions on Stiefel manifolds, in *From local to global optimization*, eds.: Pardalos, P.M., A. Migdalas, and P. Varbrand, Kluwer, Dordrecht, 2001, 277–290.

[89] Rapcsák, T., On minimization on Stiefel manifolds, *European Journal of Operational Research*, **143**(2002):365–376.

[90] Rapcsák, T., Optimization problems in statistics, *Journal of Global Optimization*, **28**(2004):217–228.

[91] Ratschek, H. and J. Rokne, *New Computer Methods for Global Optimization*. Ellis Horwood, Chichester, 1988.

[92] Ratz, D., *Automatische Ergebnisverifikation bei globalen Optimierungsproblemen*. Dissertation, Universität Karlsruhe, 1992.

[93] Redlich, O. and J.N.S. Kwong, On the thermodynamics of solutions V: an equation of state. Fugacities of gaseous solutions, *Chemical Reviews*, **44**(1949):233–244.

[94] Sahni, S. and T. Gonzalez, P-complete approximation problems, *Journal of the ACM*, **23**(1976):555–565.

[95] Seiden, S.S., On the on-line bin packing problem, *Journal of the ACM*, **49**(5):640-671, 2002.

[96] Sokal, R.R. and F.J. Rohlf, *Biometry*. New York: W. H. Freeman and company, 1981.

[97] Soave, G., Equilibrium constants from a modified Redlich-Kwong equation of state, *Chemical Engineering Science*, **27**(5):1197–1203, 1972.

[98] Stateva, R.P. and St.G. Tsvetkov, A new method for thermodynamic stability analysis in multicomponent systems, *Hung. J. Ind. Chem.*, **19**(1991):179–188.

[99] Stateva R.P. and St.G. Tsvetkov, A rigorous approach to stability analysis as a first step when solving the isothermal multiphase flash problem, *Technology Today*, **4**(1991):223–240.

[100] Stateva, R.P. and St.G. Tsvetkov, A diverse approach for the solution of the isothermal multiphase flash problem. Application to vapor-liquid-liquid systems, *Can. J. Chem. Eng.*, **72**(1994):722–734.

[101] Stateva, R.P. and W.A. Wakeham, Phase equilibrium calculations for chemically reacting systems, *Ind. Eng. Chem. Res.*, **36**(1997):5474–5482.

[102] Stiefel, E., Richtungsfelder und Fernparallelismus in n-dimensionalen Mannigfaltigkeiten, *Commentarii Math. Helvetici*, **8**(1935-36):305–353.

[103] Sun, A.C. and W.D. Seider, Homotopy-continuation method for stability analysis in the global minimization of the Gibbs free energy, *Fluid Phase Equilibria*, **103**(1995):213–249.

[104] Sutti, C., Local and global optimization by parallel algorithms for MIMD systems, *Annals of Operations Research*, **1**(2):151–164, 1984.

[105] Tóth B. és Vinkó T., Egy hatékony számítógépes eszköz matematikai problémák megoldására, *Polygon*, **11**(2002):19-42.

[106] Törn, A., A search-clustering approach to global optimization, in: *Towards Global Optimization 2.*, eds.: Dixon, L.C.W. and G.P. Szegő, North-Holland, Amsterdam, 1978, 49–72.

[107] Törn, A. and A. Žilinskas, *Global Optimization. Lecture Notes in Computer Science 350*. Springer-Verlag, Berlin, 1989.

[108] Ullman, J.D., *The performance of a memory allocation algorithm*, Technical Report 100, Princeton University, Princeton, NJ, 1971.

[109] van Vliet, A., An improved lower bound for on-line bin packing algorithms, *Information Processing Letters*, **43**(5): 277–284, 1992.

[110] van der Waals, J. D., *On the Continuity of the Gaseous and Liquid States (doctoral dissertation).* Universiteit Leiden, 1873.

[111] Wakeham, W.A. and R.P. Stateva, Numerical solution of the isothermal, isobaric phase equilibrium problem, *Reviews in Chemical Engineering*, **20**(1–2):1–56, 2004.

[112] Woodhams, F.W.D. and W.L. Price, Optimizing accelerator for CAD workstation, *IEE Proceedings Part E*, **135**(4):214–221, 1988.

[113] Yushan, Z. and Zhihong, X., Lipschitz optimization for phase stability analysis: application to Soave-Redlich-Kwong equation of state, *Fluid Phase Equilibria,* **162**(1999):19–29.