Markót Mihály Csaba

Garantált megbízhatóságú globális optimalizálási módszerek továbbfejlesztése korlátozásos feladatokra és alkalmazásuk körpakolási feladatok megoldása esetén

Doktori értekezés

Témavezető:
Dr. Csendes Tibor

Szegedi Tudományegyetem, 2003.

Ezúton szeretnék köszönetet mondani témavezetőmnek, Csendes Tibornak, aki nem csak az intervallumos optimalizálás szakterületét, de a kutatói munka rejtelmeit is megismertette velem. Köszönet illeti társszerzőimet, Leocadio González Casadot, Csallner András Eriket, Csendes Tibort, és José Fernández Hernándezt a publikálás és az értekezés írása során adott tanácsaikért. Hálával tartozom továbbá Vinkó Tamásnak a disszertációra tett javító észrevételeiért, illetve Szabó Péter Gábornak a körpakolási feladatok történetének megismertetéséért. Mindemellett köszönöm a Szegedi Tudományegyetem Informatikai Tanszékcsoportja oktatóinak biztatását és támogatását, különös tekintettel Csirik János Tanszékcsoportvezető Úrra, aki lehetővé tette, hogy polgári szolgálatom időtartamát a megszokott egyetemi környezetben kutatómunkával töltsem.

Tartalomjegyzék

Előszó

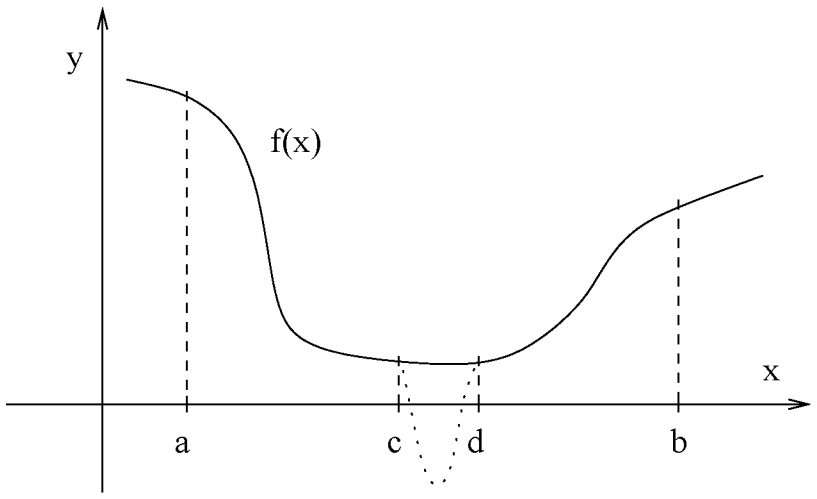
A megbízható numerikus számításokról

A valós számok digitális számítógépeken történő reprezentálásának leginkább elterjedt módja a lebegőpontos számok használata. Ezek a számok az adattípusuk szerint ugyan különböző, de minden esetben véges számú biten vannak eltárolva, így a használt aritmetika alapszámától függetlenül sokszor csupán közelítését adják a reprezentálni kívánt értéknek. Emellett lebegőpontos argumentumokon végrehajtott műveletek eredményeiről nem minden helyzetben várhatjuk, hogy azok lebegőpontos módon pontosan ábrázolhatók legyenek. Figyelembe véve, hogy napjaink szuperszámítógépei másodpercenként 10^{12} lebegőpontos művelet elvégzésére képesek (és egy átlagos asztali PC-be épített processzor is csupán 3–4 nagyságrenddel képes kevesebbre), a keletkező számítási hibák becslése és kezelése fontos feladat.

A felmerülő legfontosabb problémát tehát az alapvető számábrázolási hibák jelentik, melyek mind a be-, és kivitel során, mind pedig a köztes számítások alatt előfordulhatnak. Előbbire a decimális–bináris konverzió, utóbbira egy ún. lebegőpontos akkumulátorban előálló nagy pontosságú eredménynek az output adattípusra történő konverziója mutat példát. Az IEEE 754 bináris lebegőpontos szabványt követő architektúrák esetén a lebegőpontos műveletek eredményei az alapvető aritmetikai operátorokra az ábrázolható legpontosabbak, és ugyanez a helyzet néhány alapvető függvény, pl. a négyzetgyökvonás esetén. Transzcendens függvények esetén azonban a fenti feltétel már nem teljesíthető általánosságban. Számtalan esetben az apró kerekítési hibák is számottevő – olykor meghökkentően nagy – mértékben összegződhetnek, amint azt az alábbi példa ([44] alapján) mutatja:

Az $x_i \in \mathbb{R},\ i = 1, \ldots, n$ számok szórásának egy gyakran alkalmazott kiszámítási módja a következő: $\sigma_n = (\frac{1}{n}(\sum_{i=1}^{n} x_i^2 - \frac{1}{n}(\sum_{i=1}^{n} x_i)^2))^{1/2}$. (A formula előnye az, hogy nem igényli az egyes x_i értékek tárolását.) Egy általánosan elterjedt számítógépes környezetben (Intel Pentium IV processzor, Linux operációs rendszer, GNU C/C++ fordító, duplapontosságú lebegőpontos számábrázolás) a fenti képletet $n = 5$ és $x_1 = \ldots = x_n = d = 100/3$ esetén alkalmazva $\sigma_5 = 2.10312 \cdot 10^{-7}$ adódik. Másrészt pl. a $d = 100/33$ választással σ_5-t már közelítőleg sem tudjuk megadni, mivel a köztes lépésben kiszámított szórásnégyzet értéke negatív lesz. Ugyanakkor a szórás pontos értéke mindkét esetben nyilvánvalóan 0.

Az ellenőrzött számítások, ezen belül is az intervallum aritmetika elterjedésének másik oka a kezelendő adatok pontatlanságából fakadó bizonytalanság kezelése, illetve matematikai állítások numerikus úton történő, ám garantáltan helyes (ellenőrzött) megválaszolása. Ha az input adatok mérésekből származnak, a felhasználó szívesen



1. Ábra: $f(x)$ előjelének vizsgálata $[a, b]$-n.

dolgozna a konkrét mérési adat helyett egy, a mérési hiba segítségével kiszámított halmazzal. Ekkor olyan kérdésekre kereshetjük a választ, mint pl. 'az input halmaz minden elemére az eljárás kielégítő (pl. egy adott értéktartományba eső) megoldást szolgáltat-e?', vagy 'milyen állapotokba juthat el egy dinamikus rendszer, ha a kezdőállapotot csak mérési hibával terhelten tudjuk megállapítani?'. Az ilyen kérdésekre adott válaszok fontos többletinformációt szolgáltathatnak a felhasználónak a numerikus eljárás egyedi kezdőértékeken történő (akár tetszőlegesen sokszori) lefuttatásához képest. A matematikailag korrekt állítások előállítására lássunk egy konkrét – a jelen értekezés témájához is közel álló – példát [22]. A feladat annak megállapítása, hogy egy $f(x)$ folytonos függvénynek létezik-e gyöke egy $[a, b]$ intervallumon. Vegyük észre, hogy ha $f(x)$ a teljes $[a, b]$-n pozitív (1. Ábra), akkor akárhány *pontban* értékeljük is ki a hagyományos módon a célfüggvényt, megbízható választ sohasem kaphatunk, hiszen a függvénynek tetszőlegesen szűk intervallumon lehet az ábrán pontozott vonallal jelölt, negatív helyettesítési értékeket tartalmazó szakasza. A bizonyosság mértéke persze növelhető a függvénykiértékelések számának növelésével (pl. sűrűbb mintavételezéssel véletlen keresés esetében, vagy több iterációs lépés elvégzésével iteratív gyökkereséskor), szigorú matematikai értelemben a kérdést mégsem leszünk képesek megválaszolni. Szélsőséges esetben a $[c, d]$ tartomány két szomszédos lebegőpontos szám *között* is elhelyezkedhet, ekkor a pontokban történő kiértékelés módszere végképp kudarcot vall.

Mint azt látni fogjuk, az intervallum aritmetika alkalmazásával képesek vagyunk $[a, b]$-n a függvény értékkészletét (ami jelen esetben f folytonossága miatt egy zárt intervallum), vagy az értékkészlet valamely befoglalását megadni. Ha az értékkészletre adott befoglalás nem tartalmazza a 0-t, akkor biztosak lehetünk abban, hogy a függvénynek nincs zérushelye $[a, b]$-n. Ellenkező esetben további vizsgálatok szükségesek pl. $[a, b]$ felosztásával, majd a keletkező részintervallumokon végzett értékkészletbecslések segítségével.

Intervallumos műveletvégzés esetén egy állítás teljesülésének eldöntése ehhez hasonló az általános esetben is: az igen-nem típusú válaszok közül legtöbbször csupán

az egyiket tudjuk teljes bizonyossággal megállapítani. Amennyiben ez nem sikerül, az nem fogja automatikusan az ellenkező választ garantálni, csupán bizonytalanságot fejez ki, vagyis tovább kell vizsgálódnunk. (A korábban említett, szórást kiszámító példánál is ez a helyzet: intervallum aritmetika használatával nem leszünk képesek teljes bizonyossággal azt állítani, hogy a bevitt adatok szórása 0, ugyanis mindkét bemutatott esetben egy-egy szűk, 0-t tartalmazó befoglalást kapunk outputként. Ugyanakkor, ha valamely más adathalmazon a szórás befoglalása nem tartalmazza a 0-t, akkor biztosak lehetünk abban, hogy az adatok szórása pozitív.)

A megbízhatóság záloga abban rejlik, hogy adott (valós, vagy intervallum típusú) argumentumok esetén meg tudjuk adni az argumentumokon elvégzett művelet eredményének *garantált befoglalását*. Ezt matematikai oldalról egyszerű aritmetikai, illetve monotonitási tulajdonságok teszik lehetővé, amelyekhez a számítógépes implementáció során a kerekítési hibákat kizáró, ún. direkt kifelé kerekítési műveletek járulnak. A lehetséges kerekítési módok az IEEE 754 szabvány által definiáltak. A legtöbb implementáció esetében ezek a kerekítési műveletek az alkalmazásokat programozó felhasználó számára közvetlenül is elérhetők.

Jelen értekezés témája megbízható globális optimalizálási módszerek új változatainak ismertetése és elemzése. Az 1. Fejezetben az intervallum analízis alapvető definícióit és állításait, majd a keretalgoritmusként használt korlátozás és szétválasztás típusú algoritmus építőelemeit ismertetem. A további fejezetek lényegében a keretalgoritmus egy-egy eleme köré épülnek. A 2. Fejezetben az alapeljárás felosztási lépésének, azaz az aktuális iterációban tekintett keresési résztartomány (box) több részre osztásának (multisection) új módszereit vizsgálom empirikus úton. A 3. Fejezetben egy olyan heurisztikus mutatót ismertetek, amely korlátozásos feladatok esetén az aktuális box adaptív felosztását, illetve a keresés következő iterációjában felosztandó box kiválasztását irányíthatja. A mutató az aktuálisan vizsgált boxnak a lehetséges megoldásokat tartalmazó tartományokhoz képest számított helyzete alapján kerül definiálásra. Az előálló algoritmusok viselkedését részletesen elemzem mind elméleti, mind numerikus úton. Az értekezés 4. Fejezetében egy speciális geometriai optimalizálási feladattal, az egybevágó körök négyzetbe történő legsűrűbb pakolásainak megadásával foglalkozom. A fejezet a kereteljárás azon lépéseinek vizsgálatára helyezi a hangsúlyt, melyek a keresési tér globális optimumpontokat garantáltan nem tartalmazó részeit igyekeznek minél előbb felderíteni (gyorsítótesztek). Jelen esetben ez a körpakolási feladatosztályhoz igazodó speciális gyorsító lépések megadását, majd ezen keresztül egy teljes egészében megbízható számításokon alapuló számítógépes bizonyító eljárás ismertetését jelenti. A fejezet utolsó részében a körpakolási feladatok megoldására javasolt kezdeti eljárás továbbfejlesztéseként egy olyan algoritmust mutatok be, amellyel sikerült néhány korábban megoldatlan feladatpéldány esetén mind az összes optimumhely mind az optimumérték igen nagy pontosságú befoglalását megadnom, illetve a korábban optimálisnak sejtett pakolási struktúra optimalitását igazolnom. A módszer alapötletei nagy jelentőséggel bírhatnak további körpakolási, illetve egyéb geometriai optimalizálási problémák megoldásakor.

Az értekezés tételeinek ismertetésekor az alábbi szabályt követem: a hivatkozott tételek esetén megadom a szerzők nevét és a tétel bizonyításának fellelhetőségét. A

saját, minden esetben bizonyításokkal együtt közreadott eredményeim esetén a tételt és a bizonyítást tartalmazó publikációt tüntetem fel.

1. Fejezet

Bevezetés

1.1 Jelölési konvenciók

A továbbiakban az alábbi jelöléseket követjük: a valós skalárokra a kisbetűs (pl. x, y, z), az intervallumokra pedig a nagybetűs jelölést (pl. X, Y, Z) használjuk. A valós, illetve intervallumvektorokat vastagon szedett szimbólumokkal (pl. $\boldsymbol{x}$, illetve $\boldsymbol{X}$), a vektorok elemeit, illetve pont- és intervallumsorozatok elemeit pedig a szokásos módon alsó indexeléssel jelöljük. Azaz, az $\boldsymbol{X}$ n-dimenziós intervallumvektor komponenseit $(X_1, X_2, \ldots, X_n)$ jelöli, míg $\boldsymbol{X}_1, \boldsymbol{X}_2, \ldots$ n-dimenziós intervallumvektorok sorozatát (illetve annak elemeit) jelenti. A valós függvények jelölésére mindig kisbetűk $(f, g, \ldots)$, ezek intervallumos megfelelőinek (az ún. befoglaló függvényeknek) a jelölésére pedig a megfelelő nagybetűk $(F, G, \ldots)$ szolgálnak.

1.2 Az intervallum analízis alapjai

Az alábbi fejezetben áttekintem az intervallumok és az intervallumokon értelmezett műveletvégzés alapvető tulajdonságait (ezek bővebb ismertetése megtalálható pl. [24, 26, 42, 50, 51]-ben).

1. MEGJEGYZÉS. *A továbbiakban fontos szem előtt tartanunk azt, hogy az ismertetett összefüggések az intervallum analízis elméleti részét képezik. Az intervallumos műveletvégzés számítógépes implementációi esetén – az Előszóban említett kifelé kerekítés miatt – a bemutatott összefüggésektől kismértékben eltérő, de a számítások megbízhatóságát minden esetben garantáló eredményeket kaphatunk.*

Jelölje $\mathbb{R}$ a valós számok halmazát, $\mathbb{R}^+$ pedig a nemnegatív valós számokat. *Valós intervallumon* (vagy egyszerűen *intervallumon*) az $\mathbb{R}$ elemeinek azon nemüres, zárt és korlátos részhalmazát értjük, melyre az alábbi teljesül:

$$X = [\underline{X}, \overline{X}] = \{x \in \mathbb{R} \mid \underline{X} \leq x \leq \overline{X}\},$$

ahol $\underline{X}$, illetve $\overline{X}$ jelöli az X intervallum *alsó*, illetve *felső korlát*ját. Az $\underline{X} = \overline{X}$ esetben X-t *pontszerű* ('thin') intervallumnak hívjuk.

A valós intervallumok halmazának jelölésére a továbbiakban az $\mathbb{I}$ szimbólumot, az n-dimenziós *intervallumvektorok* (vagy röviden *box*ok) halmazának jelölésére pedig az

$\mathbb{I}^n$ jelet használjuk. (Ha nem okoz félreértést, az angol szóhasználatot követve gyakran boxok esetében is az intervallum elnevezést használom.) Egy $D \subseteq \mathbb{R}^n$ halmaz esetén $\mathbb{I}(D)$-vel jelöljük azon $\boldsymbol{X}$ n-dimenziós boxok halmazát, melyekre $\boldsymbol{X} \subseteq D$.

Egy intervallum *szélesség*ét, *sugar*át és *középpont*ját rendre a következőképpen definiálhatjuk:

$$w(X) := \overline{X} - \underline{X},$$
$$r(X) := (\overline{X} - \underline{X})/2,$$
$$m(X) := (\overline{X} + \underline{X})/2.$$

Egy intervallum *legkisebb* és *legnagyobb abszolút érték*ét pedig az alábbiak szerint:

$$\langle X \rangle := \min\{|x| \mid x \in X\}, \text{ illetve}$$

$$|X| := \max\{|x| \mid x \in X\} = \max\{|\underline{X}|, |\overline{X}|\}.$$

Tegyük, fel, hogy számításaink során egy ismeretlen értékű x valós szám X intervallumos befoglalását használjuk (azaz egy olyan, két gépi számmal reprezentált intervallumot, mely *biztosan* tartalmazza x-et). A befoglalás minőségének jellemzéseként az X intervallum *relatív szélesség*ét tekinthetjük:

$$w_{rel}(X) := \begin{cases} w(X)/\langle X \rangle & \text{ha } 0 \notin X, \\ w(X) & \text{különben.} \end{cases}$$

A relatív szélesség bevezetésének indoka nyilván az, hogy adott lebegőpontos típus esetén a gépi számok az origótól távolodva egyre 'ritkábban' helyezkednek el.

Intervallumvektorok sugara, középpontja és abszolút értékei a komponensenként számított értékek *vektor*aként, intervallumvektorok szélessége (relatív szélessége) pedig a komponensek szélességeinek (relatív szélességeinek) maximumaként, azaz *skalár*ként értelmezett.

Tekintsünk két intervallumot: $X = [a, b]$-t, és $Y = [c, d]$-t $(a, b, c, d \in \mathbb{R})$. A két intervallum *távolsága* az alábbi definícióval adott:

$$|X, Y| := \max\{|a - c|, |b - d|\},$$

két n-dimenziós intervallum, $\boldsymbol{X} = (X_1, \ldots, X_n)$ illetve $\boldsymbol{Y} = (Y_1, \ldots, Y_n)$ távolsága pedig az

$$|\boldsymbol{X}, \boldsymbol{Y}| := \max\{|X_i, Y_i| \colon i = 1, \ldots, n\}$$

formulával definiált.

Könnyen igazolható, hogy tetszőleges $\boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{Z} \in \mathbb{I}^n$ esetén $|\boldsymbol{X}, \boldsymbol{Y}|$ nemnegatív, és $|\boldsymbol{X}, \boldsymbol{Y}| = 0$ akkor és csak akkor teljesül, ha $\boldsymbol{X} = \boldsymbol{Y}$. Emellett $|\boldsymbol{X}, \boldsymbol{Y}| = |\boldsymbol{Y}, \boldsymbol{X}|$, valamint $|\boldsymbol{X}, \boldsymbol{Y}| + |\boldsymbol{Y}, \boldsymbol{Z}| \geq |\boldsymbol{X}, \boldsymbol{Z}|$. Azaz $|.,.| : \mathbb{I}^n \times \mathbb{I}^n \to \mathbb{R}^+$ egy metrika $\mathbb{I}^n$-en.

A valós számokon értelmezett *elemi művelet*ek intervallumokra vonatkozó kiterjesztései az alábbi kézenfekvő formula által adottak:

$$X \circ Y := \{x \circ y \mid x \in X,\ y \in Y\} \in \mathbb{I}, \text{ ahol } \circ \in \{+, -, \cdot, /\}. \tag{1}$$

Az intervallumos művelet eredménye tehát a valós számok azon halmaza, mely az X és Y által tartalmazott összes valós számpáron elvégzett művelet eredményeiből áll. Az intervallumos osztás esetében természetesen ki kell kötnünk, hogy $0 \notin Y$. Intervallumvektorokon végzett elemi műveletek ez esetben is komponensenként definiálhatók. A tekintett valós operátorok folytonosságát felhasználva egyszerűen igazolható ([50]), hogy (1) jobboldalán minden összefüggés esetén valóban intervallum áll. A valós műveletek monotonitási tulajdonságaiból az egyes intervallumos műveletekre az alábbi formulákat kapjuk:

$$X + Y = [\underline{X} + \underline{Y}, \overline{X} + \overline{Y}],$$

$$X - Y = [\underline{X} - \overline{Y}, \overline{X} - \underline{Y}],$$

$$X \cdot Y = [\min\{\underline{X}\,\underline{Y}, \underline{X}\,\overline{Y}, \overline{X}\,\underline{Y}, \overline{X}\,\overline{Y}\}, \max\{\underline{X}\,\underline{Y}, \underline{X}\,\overline{Y}, \overline{X}\,\underline{Y}, \overline{X}\,\overline{Y}\}],$$

$$X/Y = X \cdot [1/\overline{Y}, 1/\underline{Y}], \quad \text{ha } 0 \notin Y.$$

2. MEGJEGYZÉS. *Az úgynevezett* kiterjesztett intervallum aritmetika *segítségével – melynek lényege, hogy* $\mathbb{R}$*-et kiterjesztjük a* $\{-\infty, \infty\}$ *halmazzal – a nullát tartalmazó intervallumokkal való osztás is értelmezhetővé válik [22]. Ez a lehetőség a bemutatásra kerülő algoritmusok forráskódjaiban is rendelkezésre áll.*

Az intervallumos alapműveletek legfontosabb tulajdonságai a következők:

1. A négy alapművelet elvégzése (két adott intervallumra) pontos eredményt szolgáltat abban az értelemben, hogy az előálló intervallum pontosan az (1) definícióval adott halmaz (azaz a tekintett művelet értékkészlete) lesz.

2. Az összeadás és a szorzás asszociatív és kommutatív: $X \circ Y = Y \circ X$, illetve $X \circ (Y \circ Z) = (X \circ Y) \circ Z$, $\forall X, Y, Z \in \mathbb{I}$, ha $\circ \in \{+, \cdot\}$.

3. Az egyetlen zéruselem a $[0, 0]$, az egyetlen egységelem az $[1, 1]$ intervallum.

4. A $+$ és $-$ műveletek nem inverzei egymásnak, ugyanakkor $Y \subseteq X+(Y-X)$, $\forall X, Y \in \mathbb{I}$: pl. $[2, 5] + ([3, 4] - [2, 5]) = [2, 5] + [-2, 2] = [0, 7] \supset [3, 4]$.

5. A $\cdot$ és / műveletek sem inverzei egymásnak, viszont igaz, hogy $Y \subseteq X \cdot (Y/X)$, $\forall\, 0 \notin X$, $X, Y \in \mathbb{I}$: pl. $[1, 3] \cdot ([3, 6]/[1, 3]) = [1, 3] \cdot [1, 6] = [1, 18] \supset [3, 6]$.

6. Általában nem teljesül, hogy $\cdot$ disztributív a $+$ műveletre nézve, ehelyett az ún. *szubdisztributivitási* tulajdonság igaz: $X \cdot (Y + Z) \subseteq (X \cdot Y) + (X \cdot Z)$, $\forall X, Y, Z \in \mathbb{I}$. Például: $[1, 3] \cdot ([2, 3] + [-4, -1]) = [-6, 6]$, ugyanakkor $([1, 3] \cdot [2, 3]) + ([1, 3] \cdot [-4, -1]) = [2, 9] + [-12, -1] = [-10, 8]$.

Legyen $\varphi : D \subseteq \mathbb{R} \to \mathbb{R}$ *egy valós, standard matematikai függvény*, mely folytonos minden D-beli zárt intervallumon. (Ilyen függvények pl. az $\mathrm{abs}(x)$, e^x, $\arctan(x)$.) Ekkor φ intervallumos kiterjesztése a

$$\Phi(X) := \{\varphi(x) \mid x \in X\}$$

formulával adott minden $X \in \mathbb{I}(D)$ esetén. A standard függvények monotonitási tulajdonságait kihasználva az egyes intervallumos megfelelők egyszerűen megkonstruálhatók, a fent említett konkrét függvényekre pl. az alábbi módon:

$$\operatorname{abs}(X) = [\langle X \rangle, |X|],$$

$$e^X = [e^{\underline{X}}, e^{\overline{X}}],$$

$$\arctan(X) = [\arctan(\underline{X}), \arctan(\overline{X})].$$

A tekintett valós függvények folytonossága és az argumentum intervallumok kompaktsága miatt a kapott intervallumos függvények minden esetben intervallumot szolgáltatnak eredményül. Továbbá, hasonlóan az alapműveletekhez, az egyes intervallumos függvényszámítások eredményei megegyeznek a valós függvénynek az argumentum intervallumon felvett értékkészletével.

1.3 Befoglaló függvények

A szakasz tárgya az összetett – a négy alapműveletből, illetve standard függvényekből felépített – többváltozós *folytonos* függvények intervallumos kiszámítása. A legfontosabb elvárásunk az, hogy az eredmény intervallum minden argumentum box esetén tartalmazza a valós függvénynek az illető boxon számított értékkészletét:

1. DEFINÍCIÓ. *Az $F : \mathbb{I}^n \to \mathbb{I}$ intervallumos függvény az $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ függvény* befoglaló függvénye *$\boldsymbol{X} \in \mathbb{I}(D)$-n, ha minden $\boldsymbol{Y} \subseteq \boldsymbol{X}$ esetén $\boldsymbol{y} \in \boldsymbol{Y}$-ból $f(\boldsymbol{y}) \in F(\boldsymbol{Y})$ következik.*

Más szavakkal, F befoglaló függvénye f-nek az $\boldsymbol{X}$ boxon, ha minden $\boldsymbol{Y} \subseteq \boldsymbol{X}$-re $f(\boldsymbol{Y}) = \{z : z \in \mathbb{R},\, z = f(\boldsymbol{y}),\, \boldsymbol{y} \in \boldsymbol{Y}\} \subseteq F(\boldsymbol{Y})$ teljesül, ahol tehát $f(\boldsymbol{Y})$ az f értékkészletét jelenti $\boldsymbol{Y}$-on.

Az intervallumos számításokkal kapcsolatos vizsgálatok körében központi szerepet kap az a tény, hogy bár mind az alapműveletek, mind a standard függvények *egyenkénti* végrehajtásának eredménye pontosan tartalmazza a tekintett művelet, illetve függvény értékkészletét, összetett függvények esetén a kiértékelés eredménye sokszor a valódi értékkészletnél bővebb, ún. *túlbecslés*t eredményező intervallum lesz. (Nyilvánvalóan, amennyiben az intervallumos függvénykiszámítás eredménye mindig pontosan az értékkészlet lenne, az olyan alapvető fontosságú problémák, mint a függvények szélsőértékeinek meghatározása egyetlen intervallumos kiértékeléssel megoldhatóvá válnának.) Fontos feladat tehát a minél kisebb túlbecslést adó befoglaló függvények megadása.

2. DEFINÍCIÓ. *Tekintsük az $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ függvényt, mint egy, az $x_1, \ldots, x_n$ változókból, az aritmetikai alapműveletekből, és a standard függvényekből felépülő kifejezést. Ekkor f* természetes intervallum-kiterjesztése *$\boldsymbol{X} \in \mathbb{I}(D)$-n az az $F : \mathbb{I}^n \to \mathbb{I}$ függvény, melyet úgy kapunk, hogy az f-et megadó kifejezésben*

1. *az x_i változó előfordulásait X_i-re cseréljük ($i = 1, \ldots, n$),*

2. *a valós alapműveleteket és standard függvényeket pedig a megfelelő intervallumos változataikkal helyettesítjük.*

3. MEGJEGYZÉS. *Mivel minden vizsgált függvényhez végtelen sok vele ekvivalens kifejezés írható fel, az azokhoz tartozó befoglaló függvénykifejezések is eltérőek lesznek (és ezek különböző függvényértékeket szolgáltathatnak).*

A számítások során keletkező túlbecslések érzékeltetésére tekintsünk egy rövid példát:

PÉLDA. Számítsuk ki az $f(x) = x \cdot x - 2 \cdot x$ függvény értékét az $X = [-1, 5]$ intervallumon az előző fejezetben bevezetett összefüggések segítségével! A függvény értékkészlete $f(X) = [-1, 15]$. $f(x)$ kiszámításra többféle lehetőségünk is van:

- $F_1(X) := X \cdot X - 2 \cdot X = [-1, 5] \cdot [-1, 5] - 2 \cdot [-1, 5] = [-5, 25] - [-2, 10] = [-15, 27]$.
- Az $X \cdot X$ kifejezést X^2-re alakítva az $x^2 \geq 0$ összefüggés is felhasználhatóvá válik (ezt egyébként a számítógépes implementációk erősen ki is használják): $F_2(X) := X^2 - 2 \cdot X = [0, 25] - [-2, 10] = [-10, 27]$.
- A szubdisztributivitás tulajdonságát felhasználva az első esetben adódó befoglalást javíthatjuk: $F_3(X) := X \cdot (X - 2) = [-1, 5] \cdot ([-1, 5] - 2) = [-1, 5] \cdot [-3, 3] = [-15, 15]$.

A kapott befoglalások mindegyike eredményez valamennyi túlbecslést a valódi értékkészlethez képest.

A példa alapján azt a fontos következtetést vonhatjuk le, hogy ha a kiszámítandó kifejezés explicit formulaként rendelkezésünkre áll, akkor ésszerűen elvégzett átalakításokkal (pl. az alapműveletek 4., 5. és 6. tulajdonságai alapján) nem csupán a műveletvégzések számát, de az esetleges túlbecslés nagyságát is csökkenthetjük.

4. MEGJEGYZÉS. *A természetes intervallum-kiterjesztést az irodalomban gyakran 'naív intervallum aritmetikával adott kifejezésnek' is nevezik.*

1. TÉTEL. *Moore [42]: A természetes intervallum-kiterjesztés teljesíti a befoglaló függvény definícióját. Továbbá, ha a valós függvényt megadó kifejezésben minden változó pontosan egyszer fordul elő (SUE: single use expression), akkor a természetes intervallumkiterjesztés minden esetben az argumentumon vett értékkészlet pontos befoglalását adja.*

Folytonosan differenciálható függvények esetén a természetes intervallum-kiterjesztéssel kapott befoglalás egyik alternatívája az ún. középérték-formula (mean-value form) használata. A középérték-formula a függvény deriváltja alapján ad becslést az értékkészlet befoglalására [42]:

3. DEFINÍCIÓ. *Legyen $f : D \subseteq \mathbb{R}^n \to \mathbb{R}$ differenciálható D-n. Ekkor az f* középérték formulája *az*

$$F(\boldsymbol{X}) = f(\boldsymbol{c}) + \boldsymbol{\nabla} F(\boldsymbol{X}) \cdot (\boldsymbol{X} - \boldsymbol{c})$$

definícióval adott minden $\boldsymbol{X} \in \mathbb{I}(D)$-re, ahol $\boldsymbol{\nabla} F$ a ∇f gradiens befoglaló függvényét jelöli, $\boldsymbol{c}$ pedig egy $\boldsymbol{X}$-beli pont, az ún. kifejtési pont.

A kifejtési pont megadása tetszőleges lehet, leggyakoribb választása $\boldsymbol{c} = m(\boldsymbol{X})$ (a később ismertetendő tulajdonságok miatt).

2. TÉTEL. *Ratschek és Rokne [50]: A fent definiált középérték formula kielégíti a befoglaló függvény definícióját.*

PÉLDA. Az iménti példa folytatásaként számítsuk ki $f(x) = x \cdot x - 2 \cdot x$ értékét középérték formulával az $X = [-1, 5]$ intervallumon!

$f(x)$-t szimbolikusan deriválva $f'(x) = 2 \cdot x - 2$, ennek egy befoglaló függvénye (a természetes intervallum-kiterjesztést használva) $F'(X) = 2 \cdot X - 2$. Legyen $c = m(X) = 2$, így

$$F_4(X) = f(c) + F'(X) \cdot (X - c) = 0 + (2 \cdot [-1, 5] - 2) \cdot ([-1, 5] - 2) = [-24, 24],$$

ami jelen esetben jóval tágabb befoglalást ad, mint a naiv intervallum aritmetika. $w(X)$ csökkenésével azonban a középérték formula általában sokkal pontosabb becslést eredményez az értékkészletre, mint a természetes intervallum-kiterjesztéssel kapott befoglalások.

A bemutatott középérték formula, illetve ennek magasabbrendű deriváltakat használó változatai egy általános alakú, az irodalomban középponti formulák néven tárgyalt befoglaló függvényosztály elemei [50].

További gyakran alkalmazott (ám jelen értekezés algoritmusaiban nem használt) befoglaló függvények az ugyancsak az általános középponti formulák osztályába tartozó különböző rendű *standard középponti formulák* és az ún. *intervallumos lejtő (slope) formulák*. Ez utóbbiak a befoglaló függvények iteratív eljárással történő gyors kiszámítását teszik lehetővé [32].

Nyilvánvalóan, amennyiben az időkorlát, illetve a számítási kapacitás lehetővé teszi, egy-egy esetben többféle módszerrel is számíthatunk befoglaló függvényértékeket. A befoglaló függvény definíciója miatt ezen befoglalások metszete is korrekt befoglalást ad, így az egyes befoglaló függvények előnyei egyesíthetők. A fenti példát folytatva például $F_5(X) = F_1(X) \cap F_2(X) \cap F_3(X) \cap F_4(X) = [-10, 15]$ az eddigi legpontosabb (bár változatlanul némi túlbecslést eredményező) befoglalás lesz.

1.4 A befoglaló függvények legfontosabb tulajdonságai

A befoglaló függvényekkel kapcsolatos legfontosabb elvárásunk az, hogy szűkebb intervallumon történő kiértékelés esetén szűkebb intervallumot kapjunk eredményül (ez főleg az iteratív módszereknél fontos):

4. DEFINÍCIÓ. *Az* $F : \mathbb{I}^n \to \mathbb{I}$ *befoglaló függvény* izoton *tulajdonságú az* $\boldsymbol{X} \in \mathbb{I}^n$ *boxon, ha minden* $\boldsymbol{Y}, \boldsymbol{Z} \subseteq \boldsymbol{X}$ *és* $\boldsymbol{Y}, \boldsymbol{Z} \in \mathbb{I}^n$ *esetén* $\boldsymbol{Y} \subseteq \boldsymbol{Z}$*-ből* $F(\boldsymbol{Y}) \subseteq F(\boldsymbol{Z})$ *következik.*

Egyszerűen igazolható, hogy az intervallumos alapműveletek, illetve standard matematikai függvények intervallumos változatai izoton tulajdonságúak, amiből indukcióval következik a természetes intervallum-kiterjesztés izotonitása. Ugyanakkor a középérték formula csak abban az esetben izoton, ha kifejtési pontként az argumentum box középpontját tekintjük [22].

A következő tulajdonsággal azt tudjuk jellemezni, hogy a számított befoglalás, $F(\boldsymbol{Y})$ az $\boldsymbol{Y}$ szélességének csökkenésével milyen gyorsan közelíti az értékkészletet, $f(\boldsymbol{Y})$-t:

5. DEFINÍCIÓ. *Az* $F : \mathbb{I}^n \to \mathbb{I}$ *befoglaló függvény* konvergencia rendje $\alpha > 0$ *(vagy másképp fogalmazva,* F α-konvergens*) az* $\boldsymbol{X} \in \mathbb{I}^n$ *boxon, ha létezik olyan* $c \in \mathbb{R}$ *pozitív konstans, hogy bármely* $\boldsymbol{Y} \subseteq \boldsymbol{X}$, $\boldsymbol{Y} \in \mathbb{I}^n$ *esetén*

$$w(F(\boldsymbol{Y})) - w(f(\boldsymbol{Y})) \leq c(w(\boldsymbol{Y}))^{\alpha}$$

teljesül, ahol $f(\boldsymbol{Y})$ *az* f *(valós) függvény értékkészletét jelöli* $\boldsymbol{Y}$*-on.*

Az α-konvergencia speciális eseteként lineárisan, illetve kvadratikusan konvergens befoglaló függvényekről beszélünk akkor, ha az α-konvergencia $\alpha = 1$-gyel, illetve $\alpha = 2$-vel teljesül.

3. TÉTEL. *Ratschek és Rokne [50]: A természetes intervallum-kiterjesztés lineárisan konvergens befoglaló függvényeket eredményez.*

6. DEFINÍCIÓ. *Az* $F : \mathbb{I}^n \to \mathbb{I}$ *függvény* Lipschitz-folytonos *(vagy egyszerűen* Lipschitz*)* $\boldsymbol{X} \in \mathbb{I}^n$*-n, ha létezik olyan* k *valós szám – az ún. Lipschitz konstans –, hogy*

$$w(F(\boldsymbol{Y})) \leq k w(\boldsymbol{Y})$$

teljesül minden $\boldsymbol{Y} \subseteq \boldsymbol{X}$, $\boldsymbol{Y} \in \mathbb{I}^n$ *esetén.*

Egyszerűen igazolható, hogy az intervallumos alapműveletekből és a folytonos standard matematikai függvényekből a természetes intervallum-kiterjesztéssel felépített befoglaló függvények Lipschitz-folytonosak.

4. TÉTEL. *Ratschek és Rokne [50]: A középérték formula kvadratikusan konvergens az* $\boldsymbol{X} \in \mathbb{I}^n$ *boxon, ha* $\boldsymbol{c} = m(\boldsymbol{X})$*, valamint a gradiens komponenseinek befoglaló függvényei, azaz* $\boldsymbol{\nabla}_i F$, $i = 1, \ldots, n$ *Lipschitz-folytonosak* $\boldsymbol{X}$*-en.*

[50] alapján a kvadratikus konvergencia a korábban említett középponti formulák teljes osztályára igazolható.

A befoglaló függvények konvergencia rendjei szolgáltatják tehát az elméleti magyarázatot arra, hogy szűkebb argumentumokon miért ad pontosabb becslést a természetes intervallum-kiterjesztésnél a középérték formula (és általában a középponti formulák).

7. DEFINÍCIÓ. *Az $F : \mathbb{I}^n \to \mathbb{I}$ befoglaló függvény teljesíti a* zéró-konvergencia tulajdonság*ot $\boldsymbol{X} \in \mathbb{I}^n$-en, ha $w(F(\boldsymbol{Z}_i)) \to 0$ teljesül minden olyan $\{\boldsymbol{Z}_i\}$, $\boldsymbol{Z}_i \in \mathbb{I}^n$, $i = 1, 2, \ldots$ intervallumsorozat esetén, melyre $\boldsymbol{Z}_i \subseteq \boldsymbol{X}$, valamint $w(\boldsymbol{Z}_i) \to 0$.*

Az α-konvergencia definíciójából a rendőr-elv egyszerű alkalmazásával adódik, hogy minden α-konvergens befoglaló függvény (így a természetes intervallum-kiterjesztéssel és a középérték formulával a 4. Tétel alapján adott befoglalás) egyben zérókonvergens is.

5. MEGJEGYZÉS. *Ha az izotonitás, a Lipschitz-folytonosság, az α-konvergencia, illetve a zéró-konvergencia használatakor nem adjuk meg az $\boldsymbol{X}$ boxot, akkor úgy tekintjük, hogy a függvényre annak teljes értelmezési tartományán (illetve optimalizálási feladatok esetén a teljes keresési tartományon) teljesül az adott tulajdonság.*

1.5 A vizsgált optimalizálási feladatok

Legyenek f és g_j $(j = 1, \ldots, r)$ az $\boldsymbol{X}_0$ n-dimenziós intervallumvektoron értelmezett, és a teljes $\boldsymbol{X}_0$-on folytonos valós függvények. A vizsgált első feladatosztály az alábbi:

$$\min_{\boldsymbol{x} \in \boldsymbol{X}_0} f(\boldsymbol{x}), \tag{$*$}$$

azaz a zárt $\boldsymbol{X}_0$ tartományon, a keresési tartományon keressük f *összes globális minimumhely*ét, és a *minimum érték*ét, f^*-ot. Speciális esetekben, mint pl. a körpakolási feladatok esetén f^*-ot (vagy annak elegendően jó közelítését) ismerjük, ekkor célunk az összes globális minimumhely meghatározása lehet. A $(*)$ feladat elnevezése *a keresési tartomány korlátait tartalmazó optimalizálási probléma* (bound-constrained optimization problem), hiszen az $\boldsymbol{x} \in \boldsymbol{X}_0$ feltétel $2n$ db egyenlőtlenséggel, a lehetséges megoldások komponenseinek alsó és felső korlátaival írható le. Ezek az összefüggések azonban implicit módon megjelennek a feladat megoldásában, nevezetesen a keresési tér mint induló tartomány megadásakor. Emiatt (illetve a lent ismertetendő feladattípustól való megkülönböztetés okán) a $(*)$ alakú feladat neve a továbbiakban *korlátozó feltétel nélküli feladat.* Az irodalomban ez az elnevezés gyakran a teljes $\mathbb{R}^n$-en való optimalizálást jelöli, ilyen feladatokkal azonban jelen értekezésben nem foglalkozom, így az elnevezés nem lesz félreértelmezhető.

A tekintett második típusú globális optimalizálási feladat az alábbi alakú:

$$\begin{aligned} \min \quad & f(\boldsymbol{x}), \\ \text{s.t.} \quad & g_j(\boldsymbol{x}) \leq 0, \ j = 1, \ldots, r, \\ & \boldsymbol{x} \in \boldsymbol{X}_0, \end{aligned} \tag{$**$}$$

vagyis itt a keresési tartomány specifikálása mellett további, egyenlőtlenség alakú korlátozó feltételek is megjelennek. A $(**)$ típusú feladat elnevezése *egyenlőtlenség alakú korlátozó feltételekkel ellátott (feltételes) optimalizálási probléma* (inequality constrained global optimization problem). Szűkebb értelemben korlátozó feltételek alatt a továbbiakban csak a $g_j(\boldsymbol{x}) \leq 0$ feltételeket értem, hiszen az $\boldsymbol{x} \in \boldsymbol{X}_0$ feltétel ismét a keresési tartomány megadásakor realizálódik.

1.6 A korlátozás és szétválasztás típusú algoritmus

A $(*)$ illetve $(**)$ alakú feladatok megoldására az intervallum analízisen alapuló korlátozás és szétválasztás (branch–and–bound, B&B) típusú módszert használom kereteljárásként. Az eljárás alapelemeit Moore [42], illetve Skelboe [58] dolgozta ki, az utóbbi évtizedekben számos módosításuk látott napvilágot. A módszer előnye, hogy az intervallumos számítások használatával globális, megbízható és garantált pontosságú (azaz a lokális optimumhelyeket kiszűrő, és számítási hibáktól mentes) megoldásokat szolgáltat a hagyományos eljárásokkal nem, vagy csak nehezen megoldható feladatok esetére is. Az előnyöket szépen mutatja pl. [7], ahol egy vegyipari alkalmazást vizsgálva az intervallumos módszer derített fényt bizonyos alakú feladatok optimális megoldásaira, megcáfolva ezzel egy alapvető, a szakterületen korábban elfogadottnak tartott feltételezést az optimális megoldások szerkezetére vonatkozóan.

Az eljárás prototípusát az 1. Algoritmus mutatja. Az algoritmus egy *iterációján* a 3. és 20. lépések közötti fő ciklus egyszeri végrehajtását értjük. A további fejezetekben az egyes speciális jellegű problémák megoldása esetén megadom az algoritmus pontos specifikációját. Ez a befoglaló függvények és a deriváltak megadását, a felosztási irány megadását, az intervallum-felosztási és intervallum-kiválasztási szabályt, az alkalmazott gyorsítóteszteket, valamint a megállási feltételt foglalja magában. A következőkben megvizsgálom az alkalmazott lehetőségeket, előbb azonban érdemes áttekinteni az eljárás során keletkező boxsorozatokra vonatkozó néhány fontos definíciót.

1.6.1 Boxsorozatok konvergenciája

Az 1. Algoritmus működését (konvergencia-tulajdonságait) vizsgáló módszerek kulcsa az egyes iterációs lépésekben felosztásra kijelölt boxok (leading box) sorozatának elemzése. Az aszimptotikus viselkedés leírásához az algoritmus megállási feltételeit kikapcsoljuk, így elegendő $\mathcal{L}_{\mathcal{W}}$ elemeinek vizsgálatára szorítkoznunk. A szóban forgó sorozat ekkor az $\{\boldsymbol{Y}_i\}_{i=1}^{\infty}$ módon jelölhető, ahol $\boldsymbol{Y}_i$ az i-edik iterációs lépésben felosztott, az 1. Algoritmusban $\boldsymbol{Y}$-nal jelölt box.

8. Definíció. *Egy $\{\boldsymbol{Z}_i\}_{i=1}^{\infty}, \boldsymbol{Z}_i \in \mathbb{I}^n$ boxsorozat akkor és csak akkor konvergál egy $\boldsymbol{Z} \in \mathbb{I}^n$ boxhoz, ha $\boldsymbol{Z}_i$ j-edik komponensei alsó, illetve felső korlátjának sorozatai $\boldsymbol{Z}$ j-edik komponensének alsó, illetve felső korlátjához tartanak minden $j = 1, \ldots, n$ esetén.*

A tárgyalt esetekben az $\{\boldsymbol{Y}_i\}_{i=1}^{\infty}$ sorozat egymásba ágyazott boxokból álló részsorozatait vizsgáljuk. Ezen részsorozatok között lehetnek végesek, amennyiben egy boxot valamely iterációs lépés után nem választunk ki felosztásra, pl. mert kedvezőtlen értéket szolgáltat az intervallum-kiválasztási szabály számára, vagy mert teljes egészében töröljük egy gyorsítóteszttel. Az algoritmus konvergenciája szempontjából számunkra nyilván a végtelen ilyen részsorozatok a lényegesek.

Jegyezzük meg, hogy $\{\boldsymbol{Y}_i\}_{i=1}^{\infty}$ minden konvergens $\{\boldsymbol{Y}_{i_l}\}_{l=1}^{\infty}$ részsorozata azonosítható egy olyan $\{\boldsymbol{Y}_i\}$-beli, megegyező határértékű részsorozattal, melynek alsó korlátai monoton növő, felső korlátai pedig monoton csökkenő sorozatokat alkotnak minden

1. Algoritmus B&B globális optimalizálási algoritmus – általános alak

Input: – f: a célfüggvény,
– $\boldsymbol{X}_0$: a keresési tartomány.
Output: – *Minimum*: a globális minimum befoglalása,
– $\mathcal{L}_\mathcal{S}$ (Eredménylista): az összes globális minimumhely befoglalásait tartalmazó lista.

1: $\boldsymbol{Y} := \boldsymbol{X}_0$; inicializáljuk az $\mathcal{L}_\mathcal{W}$ Munkalistát: $\mathcal{L}_\mathcal{W} := \{\}$;
2: inicializáljuk a minimum aktuálisan ismert legjobb felső korlátját: $\tilde{f} := \overline{F}(\boldsymbol{X}_0)$.
3: **do**
4: *felosztási irány(ok) választása* $\boldsymbol{Y}$-ra;
5: *intervallum-felosztás*: vágjuk $\boldsymbol{Y}$-t a kiválasztott irányok szerint s darab boxra, $\boldsymbol{Y}^1, \ldots, \boldsymbol{Y}^s$-re;
6: **for** $i := 1$ **to** s **do**
7: $\tilde{f}$ *aktualizálása* (csökkentése), ha lehetséges;
8: *gyorsítótesztek* végrehajtása $\boldsymbol{Y}^i$-re;
9: **if** ($\boldsymbol{Y}^i$ teljes egészében törölhető) **then** következő i;
10: jelölje $\hat{\boldsymbol{Y}}^i \subseteq \boldsymbol{Y}^i$ az $\boldsymbol{Y}^i$-ből nem törölt tartományokat befoglaló boxot;
11: **if** (a *megállási feltétel* teljesül $\hat{\boldsymbol{Y}}^i$-re) **then** tegyük $(\hat{\boldsymbol{Y}}^i, \underline{F}(\hat{\boldsymbol{Y}}^i))$-t az $\mathcal{L}_\mathcal{S}$-be;
12: **else** tegyük $(\hat{\boldsymbol{Y}}^i, \underline{F}(\hat{\boldsymbol{Y}}^i))$-t az $\mathcal{L}_\mathcal{W}$-be;
13: **end for**
14: *tovább* := 0;
15: **if** ($\mathcal{L}_\mathcal{W}$ nemüres) **then**
16: *tovább* := 1;
17: *intervallum-kiválasztás*: válasszunk egy $(\boldsymbol{Y}, \underline{F}(\boldsymbol{Y}))$-t $\mathcal{L}_\mathcal{W}$-ből;
18: töröljük $(\boldsymbol{Y}, \underline{F}(\boldsymbol{Y}))$-t $\mathcal{L}_\mathcal{W}$-ből;
19: **end if**
20: **while** (*tovább*);
21: $Minimum := [\min\{\underline{F}(\boldsymbol{Y}) \mid (\boldsymbol{Y}, \underline{F}(\boldsymbol{Y})) \in \mathcal{L}_\mathcal{S}\}, \tilde{f}]$; **return** *Minimum*, $\mathcal{L}_\mathcal{S}$;

komponens esetén. Más szóval az utóbbi részsorozat egymásba ágyazott boxokból áll. Egy ilyen részsorozat pl. a $\{\boldsymbol{Z}_k\}_{k=1}^\infty$, $\boldsymbol{Z}_k := E(\{\boldsymbol{Y}_{i_l}\}_{l=k}^\infty)$ módon lehet adott, ahol $E()$ az argumentum sorozat elemeit befoglaló legszűkebb $\{\boldsymbol{Y}_i\}$-beli boxot jelöli. Az egymásba ágyazottság feltételezése emiatt nem jelent megkötést, ugyanakkor több esetben egyszerűbb vizsgálódást tesz lehetővé.

Egy $\boldsymbol{z} \in \mathbb{R}^n$ pont a $\{\boldsymbol{Z}_i\}_{i=1}^\infty, \boldsymbol{Z}_i \in \mathbb{I}^n$ sorozat torlódási pontja, ha létezik olyan $\{\eta_i\}_{i=1}^\infty$, $\eta_i \in \boldsymbol{Z}_i$ pontsorozat, melynek $\boldsymbol{z}$ torlódási pontja. Mivel $\boldsymbol{Y}_i \subseteq \boldsymbol{X}_0$ minden i-re, továbbá $\boldsymbol{X}_0$ korlátos, ezért $\{\boldsymbol{Y}_i\}_{i=1}^\infty$-nek van legalább egy torlódási pontja.

9. DEFINÍCIÓ. *Tekintsük az* $\{\boldsymbol{Y}_i\}_{i=1}^\infty$ *sorozat (egymásba ágyazott boxokból álló) konvergens részsorozatainak hatérértékeit. Legyen ezen határértékek halmazának uniója* $A \subseteq \boldsymbol{X}_0$. *Ekkor a vizsgált algoritmus változatra azt mondjuk, hogy az az* A *halmazhoz konvergál.*

1.6.2 Befoglaló függvények és deriváltak

A befoglaló függvények megadása az értekezés legtöbb numerikus vizsgálatában természetes intervallum-kiterjesztéssel történik. Amennyiben a célfüggvény legalább egyszer, illetve legalább kétszer folytonosan differenciálható, akkor a deriváltból, illetve a Hesse-mátrixból származó információt is felhasználhatjuk a számítások során. Ez esetben a befoglaló függvények értékeit a középérték formula alapján pontosítjuk. A deriváltak befoglaló függvényértékeit, tekintettel arra, hogy a célfüggvény az implementációban egy függvényhívás (pontosabban egy kifejezés) képében explicit módon a rendelkezésünkre áll, az *automatikus differenciálás* módszerével [22, 49] számíthatjuk ki. Ez a módszer a szimbolikus, illetve numerikus differenciálási módszerek mellett nyújt újabb lehetőséget azzal, hogy a deriváltértékek szubrutinokon keresztül, közvetlenül a kívánt boxra vonatkozólag kerülnek kiszámításra a deriváltfüggvény-kifejezések meghatározása nélkül. A használt adattípus lehetőséget biztosít arra, hogy a tekintett boxon a célfüggvény, a gradiens, és a Hesse-mátrix befoglalásának kiszámítása egy lépésben, egyszerre történjen meg, azaz szubrutinhívások egy sorozatával az összes kívánt befoglalás egyidőben megkapható. Az értekezésben alapul vett C–XSC Toolbox intervallumos csomag az ún. előrefelé történő (forward) automatikus differenciálást bocsátja rendelkezésre. Az elnevezés arra utal, hogy a tekintett kifejezés deriváltjainak kiértékelési műveletei ugyanabban a sorrendben hajtódnak végre, mint ahogy magának a kifejezésnek a kiértékelése történik. (A visszafelé történő automatikus differenciálás gyorsabb, de nehezebben implementálható alternatíváját adja a forward módszernek.)

1.6.3 Az intervallum-felosztási irány választása

Az 1. Algoritmus 4. lépése az aktuálisan a Munkalistából kivett boxra (leading box), $\boldsymbol{Y}$-ra határozza meg azt a koordináta irányt (vagy a továbbfejlesztett módszerek esetén *irányok*at), amelyre merőlegesen a box felosztásra kerül [15, 54]. Általánosságban, az $\boldsymbol{Y}$ n-dimenziós box felosztásához meghatározzuk a

$$k := \min \left\{ j \mid j \in \{1, 2, \ldots, n\},\ D(j) = \max_{i=1}^{n} D(i) \right\} \tag{2}$$

értéket, ahol $D(i)$-t az alkalmazott stratégia definiálja:

'A'–stratégia: A klasszikus [42, 51, 58] eljárás szerint

$$D(i) := w(Y_i),$$

azaz k azon irányok egyike lesz, amely mentén $\boldsymbol{Y}$ a legszélesebb. Ez a stratégia azon a meggondoláson alapul, hogy az induló $\boldsymbol{X}_0$ boxot ily módon felosztva tart a részboxok szélessége a leggyorsabban a 0-hoz. Az 'A' szabály előnye, hogy nem igényel $\boldsymbol{Y}$-ról sok és nehezen meghatározható információt.

'B'–stratégia: A Hansen és Walster [24] által leírt felosztó módszer lényege a következő: válasszuk azt a koordináta irányt, melyre a

$$W_i \quad = \quad \max_{t \in Y_i} f\left(m(Y_1), \ldots, m(Y_{i-1}), t, m(Y_{i+1}), \ldots, m(Y_n)\right)$$

$$-\min_{t\in Y_i} f\left(m(Y_1),\ldots,m(Y_{i-1}),t,m(Y_{i+1}),\ldots,m(Y_n)\right)$$

kifejezés értéke maximális. A W_i érték annak jellemzésére szolgál, mennyit változik $t\in\mathbb{R}$ Y_i-n való változtatásával f értéke. W_i-t az algoritmusban a $w(\nabla_i F(\boldsymbol{Y}))\cdot w(Y_i)$ kifejezéssel közelítjük, a 'B'-stratégiára tehát

$$D(i) := w(\nabla_i F(\boldsymbol{Y}))\cdot w(Y_i).$$

A 'B' jelű szabálynál így az $\boldsymbol{Y}$-on számított gradiensre is szükség van. Ennek minden felosztás előtti kiszámítása jelentősen megnövelné az algoritmus időigényét. Vegyük azonban észre, hogy minden $\boldsymbol{Y}\neq\boldsymbol{X}_0$-ra valamely korábbi iteráció 8. lépésében már végrehajtottunk bizonyos gyorsító eljárásokat (lásd később). Ha ezekhez az eljárásokhoz kiszámítottuk a gradiens befoglalását, akkor érdemes már $\boldsymbol{Y}$ *Munkalistába tételekor* meghatározni $D(i)$-t, és azt $\boldsymbol{Y}$-nal (és más információkkal) együtt a Munkalistában eltárolni.

'C'–stratégia: A következő módszer Kearfott-tól és Ratz-tól [52] származik. A cél ezúttal az aktuális boxon vett célfüggvény-befoglalás szélességének minimalizálása:

$$w(F(\boldsymbol{Y})) = w(F(\boldsymbol{Y}) - f(m(\boldsymbol{Y}))) \approx w(\nabla F(\boldsymbol{Y})(\boldsymbol{Y}-m(\boldsymbol{Y}))) =$$

$$w\left(\sum_{i=1}^{n}\nabla_i F(\boldsymbol{Y})(Y_i-m(Y_i))\right) = \sum_{i=1}^{n} w\left(\nabla_i F(\boldsymbol{Y})(Y_i-m(Y_i))\right).$$

Ezek alapján tehát azt az irányt kell választanunk, melyre $w(\nabla_i F(\boldsymbol{Y})\cdot(Y_i-m(Y_i)))$ maximális, így

$$D(i) := w(\nabla_i F(\boldsymbol{Y})\cdot(Y_i-m(Y_i))).$$

A gradiens számítása miatt a $D(i)$ meghatározására természetesen most is fennállnak a 'B'-stratégiánál elmondottak.

'D'–stratégia: Ez a stratégia sem igényel gradiens számítást, csakúgy, mint az 'A', és sokban hasonlít is hozzá. A különbség mindössze annyi, hogy míg az 'A'-stratégiánál a box komponensenkénti szélességeit vizsgáltuk, addig most a relatív szélességeket vesszük figyelembe. Azaz

$$D(i) := w_{rel}(Y_i).$$

Tekintsünk egy olyan felosztásra váró boxot, melynek komponensenkénti szélességei hasonlóak, de az egyes komponensek abszolút értékei lényegesen különböznek. Mivel a lebegőpontos számábrázolás miatt az origótól távolabbi komponensekben 'ritkábban' helyezkednek el a gépi számok, a befoglaló függvény túlbecslésének csökkentése miatt érdemesebb lehet a kis abszolútértékű komponensek menti felosztást választanunk.

'E'–stratégia: Ugyancsak Ratz-tól származik, és a 'C'-stratégia továbbfejlesztett változatának tekinthető [55]. Most $w(F(\boldsymbol{Y}))$-t másodrendű deriváltak segítségével közelítjük:

$$w(F(\boldsymbol{Y})) = w(F(\boldsymbol{Y}) - f(m(\boldsymbol{Y}))) \approx$$

$$w\left((\boldsymbol{Y}-m(\boldsymbol{Y}))\cdot(\nabla F(m(\boldsymbol{Y})) + \frac{1}{2}\nabla^2 F(\boldsymbol{Y})\cdot(\boldsymbol{Y}-m(\boldsymbol{Y}))\right) =$$

$$w\left(\sum_{i=1}^{n}(Y_i - m(Y_i)) \cdot (\nabla_i F(m(\boldsymbol{Y})) + \frac{1}{2}\sum_{j=1}^{n} \nabla_{ij}^2 F(\boldsymbol{Y}) \cdot (Y_j - m(Y_j)))\right) =$$

$$\sum_{i=1}^{n} w\left((Y_i - m(Y_i)) \cdot (\nabla_i F(m(\boldsymbol{Y})) + \frac{1}{2}\sum_{j=1}^{n} \nabla_{ij}^2 F(\boldsymbol{Y}) \cdot (Y_j - m(Y_j)))\right).$$

Az 'E'-szabály szerint tehát

$$D(i) := w((Y_i - m(Y_i)) \cdot (\nabla F(m(\boldsymbol{Y})) + \frac{1}{2}\nabla^2 F(\boldsymbol{Y}) \cdot (Y_i - m(Y_i)))).$$

A stratégia hátránya, hogy Hesse-mátrixok kiszámítását igényli, ezért olyan algoritmusokban célszerű használni, amelyek gyorsító eljárásként az ún. *intervallumos Newton-lépést* (1.6.5 szakasz) is tartalmazzák. Emellett $D(i)$-hez szükség van az $\boldsymbol{Y}$ box középpontjában számított gradiens értékére (ami tulajdonképpen egy intervallumos befoglalásként lesz adott). Ezt viszont nem használjuk máshol az algoritmusban, így az 'E'-stratégiák esetén külön ki kell számolnunk. Amint azt a tapasztalatok mutatják, ez a tény jelentősen rányomja bélyegét a végrehajtáshoz szükséges gradiens hívások számára, és ezzel a futási idő nagyságára.

A különböző intervallum-felosztási szabályokat használó algoritmusok viselkedését az 'A'–'D' szabályokra Csendes és Ratz [15, 54], míg az 'E' szabály esetén Ratz [55] tárgyalja. A [15]-beli modelleljárás izoton és zéró-konvergens befoglaló függvényeket vizsgál az alap algoritmus megállási feltételének elhagyásával, gyorsító eljárásként az 1.6.5 szakaszban ismertetett 'kivágási', illetve 'monotonitási' tesztet alkalmazza, míg a felosztandó intervallumot kiválasztó szabály (1.6.6 szakasz) az ún. Moore–Skelboe stratégia.

10. DEFINÍCIÓ. *Egy (∗) alakú feladat egy $\boldsymbol{x}'$ globális minimumpontja* rejtett, *ha létezik olyan pozitív mértékű $\boldsymbol{X}' \subseteq \boldsymbol{X}_0$ box, melyre $\boldsymbol{x}' \in \boldsymbol{X}'$ és $\underline{F}(\boldsymbol{X}') = f^*$, ugyanakkor a tekintett feladatnak létezik olyan $\boldsymbol{x}''$ globális minimumpontja, melyre $\underline{F}(\boldsymbol{X}'') < f^*$ minden olyan pozitív mértékű $\boldsymbol{X}'' \subseteq \boldsymbol{X}_0$ box esetén, mely tartalmazza $\boldsymbol{x}''$-t.*

5. TÉTEL. *Csendes és Ratz [15, 54] alapján: Jelölje a modellalgoritmus által felosztásra kijelölt boxok sorozatát $\{\boldsymbol{Y}_i\}_{i=1}^{\infty}$, és tegyük fel, hogy a tekintett feladat minimumhelyei stacionárius pontok.*
(i) Az 'A', 'D' felosztási stratégiák egyikét alkalmazó modellalgoritmus az összes nem rejtett globális minimumhely halmazához tart, $\lim_{i\to\infty} F(\boldsymbol{Y}_i) = f^*$, *továbbá* $\lim_{i\to\infty} w(\boldsymbol{Y}_i) = 0$ *is teljesül.*
(ii) A 'B' és 'C' stratégiák egyikét használó algoritmus vagy az összes nem rejtett globális minimumpont halmazához tart, vagy pedig olyan $\boldsymbol{X} \subseteq \boldsymbol{X}_0$ boxok halmazához, melyekre $w(\boldsymbol{X}) > 0$, továbbá $\boldsymbol{X}$ kizárólag nem rejtett globális minimumpontokat tartalmaz.

1.6.4 Intervallum-felosztási szabályok

A prototípus algoritmus 5. lépésében az aktuális $\boldsymbol{Y}$ box felosztása történik meg.

A hagyományos felező (bisection) stratégiák esetén az aktuális boxot egyszerűen elfelezzük a $D(i)$ alapján meghatározott k irányra merőlegesen. A kérdés, melyet először Berner [1], majd Csallner, Csendes és Markót [9] vetettek fel, az, hogy vajon hatékonyabban megtalálhatók-e a feladat globális minimumhelyei akkor, ha egy lépésben nem két, hanem több új boxot hozunk létre. Egyrészt, kettőnél több részre vágás esetén kisebb intervallumokat kapunk, így gyorsabban eljuthatunk a 11. lépés megállási feltételét teljesítő boxokhoz. Emellett kisebb boxokon a befoglaló függvények alapvető tulajdonságainak fennállása esetén pontosabbak a célfüggvény, gradiens, és Hesse-mátrix befoglalások, ezáltal pedig jobban működhetnek a gyorsítótesztek. Másrészt viszont, ha egy iterációs lépésben több boxunk van, a gyorsító eljárások alkalmazásához, illetve a felosztási irány(ok) kiszámításához többször kell elvégeznünk az esetleg szükséges célfüggvény, gradiens, illetve Hesse-mátrix kiszámításokat, ami jelentősen megnövelheti a számítási igényt. Az értekezés 2. Fejezetében tulajdonképpen ezt a kérdést tisztázom empirikus úton $(*)$ alakú feladatok esetén.

A több részre történő felosztó eljárások továbbfejlesztett változatai az *adaptív intervallum-felosztási szabályok*, melyek lényege, hogy az algoritmus futása során különböző számú részboxot eredményező felosztó stratégiák közül választunk. Az értekezés 3. Fejezetében $(**)$ alakú feladatokra vizsgálom az ilyen adaptív szabályokat.

1.6.5 Gyorsító lépések

Az algoritmus 8. lépésében történik meg a korábbiakban már többször említett gyorsítótesztek (accelerating devices) végrehajtása a felosztás után keletkezett boxok mindegyikére. A tesztek lényege, hogy törlik az aktuálisan vizsgált box azon részeit (esetleg a teljes boxot), amelyek *biztosan* nem tartalmaznak globális minimumhelyet. A bemutatásra kerülő algoritmusokban a felhasznált tesztek halmaza a megoldandó feladat típusától és a deriváltak befoglalásainak elérhetőségétől függően többféle. Az értekezésben használt tesztek az alábbiak:

- Lehetséges megoldások tartalmazását vizsgáló teszt (vagy röviden lehetséges megoldások tesztje, (feasibility test), $(**)$ alakú feladatok esetén: ha egy $\boldsymbol{Y}$ box biztosan nem tartalmaz lehetséges megoldásokat, azaz $\underline{G_j}(\boldsymbol{Y}) > 0$ valamely $j \in \{1, \ldots, r\}$-re, akkor $\boldsymbol{Y}$ teljes egészében törölhető.

- Középponti teszt (midpoint test), $(*)$ és $(**)$ alakú feladatok esetén: ha a feldolgozásra kerülő $\boldsymbol{Y}$ box $\boldsymbol{c}$ középpontja (illetve a középpontot befoglaló intervallumvektor minden pontja) biztosan lehetséges megoldás, akkor $\overline{F}(\boldsymbol{c})$-t kiszámítva aktualizálhatjuk a globális minimum befoglalásának felső korlátját: $\tilde{f} := \min\{\tilde{f}, \overline{F}(\boldsymbol{c})\}$. Továbbá, ha egy $\boldsymbol{Y}$ boxról megállapítható, hogy legalább egy lehetséges megoldást tartalmaz, akkor a frissítés az $\tilde{f} := \min\{\tilde{f}, \overline{F}(\boldsymbol{Y})\}$ szabály szerint is elvégezhető. Emiatt minden olyan $\boldsymbol{Y}$ box törölhető a Munkalistából, melyre $\underline{F}(\boldsymbol{Y}) > \tilde{f}$ áll, illetve egy $\boldsymbol{Y}$ boxot csak akkor kell $\mathcal{L}_{\mathcal{W}}$-be tennünk, ha rá $\underline{F}(\boldsymbol{Y}) \leq \tilde{f}$ teljesül. A tesztet a Munkalista átvizsgálása miatt gyakran kivágási tesztnek (cut-off test) is nevezik.

- Monotonitási teszt (monotonicity test), (∗) és (∗∗) alakú feladatok esetén: ha a vizsgált $\boldsymbol{Y}$ garantáltan csak lehetséges megoldásokat tartalmaz, valamint a célfüggvény szigorúan monoton valamely változójában $\boldsymbol{Y}$-on, akkor $\boldsymbol{Y}$ biztosan nem tartalmazhat globális minimumpontot a *belsejében*. Formálisan, ha $\boldsymbol{\nabla} F = (\boldsymbol{\nabla}_1 F, \ldots, \boldsymbol{\nabla}_n F)^T$ a célfüggvény ∇f gradiensének befoglaló függvénye, és $0 \notin \boldsymbol{\nabla}_i F(\boldsymbol{Y})$ valamely $i = 1, \ldots, n$ esetén, akkor $\boldsymbol{Y}$ a megfelelő alsó vagy felső határaira redukálható, stacionárius optimumhelyek keresése esetén pedig teljes egészében törölhető.

- Konkavitási teszt (concavity test), (∗) és (∗∗) alakú feladatok esetén: tételezzük fel, hogy a feladat optimális megoldásai egyben *stacionárius* minimumpontok. Így, ha az aktuális $\boldsymbol{Y}$ boxon az f függvény nem konvex, akkor $\boldsymbol{Y}$ nem tartalmazhat globális mininumhelyet. Ha f konvex $\boldsymbol{Y}$-on, akkor Hesse-mátrixa pozitív szemidefinit. Ennek egy szükséges feltétele az, hogy a Hesse-mátrix összes diagonális eleme nemnegatív legyen. Azaz ha $\boldsymbol{H} = \boldsymbol{\nabla}^2 F(\boldsymbol{Y}) \in \mathbb{I}^{n \times n}$ és

$$\overline{H_{ii}} < 0 \text{ valamely } i = 1, \ldots, n \text{ esetén,}$$

  akkor $H_{ii} < 0$. Így F befoglaló tulajdonsága miatt $h_{ii} < 0$ áll fenn minden $\boldsymbol{y} \in \boldsymbol{Y}$-re, ahol $\boldsymbol{h} = \nabla^2 f(\boldsymbol{y})$. Ezért f nem lehet konvex $\boldsymbol{Y}$-on, tehát az $\boldsymbol{Y}$ box teljes egészében törölhető.

  A konkavitási teszt a Hesse-mátrix számítása miatt jóval költségesebb, mint a monotonitási teszt, ráadásul a teljes keresési téren konvex függvények esetén teljesen hatástalan. Alkalmazása mégis célszerűnek tűnik azokban az algoritmusokban, ahol *intervallumos Newton-módszer*t is alkalmaznak, ez utóbbihoz ugyanis a Hesse-mátrix kiszámítása mindenképpen szükséges.

- Intervallumos Newton-lépés, (∗) és (∗∗) alakú feladatok esetén: amennyiben ismét csak stacionárius minimumhelyeket kell keresnünk, akkor használhatjuk a kiterjesztett intervallum aritmetikát (lásd a 2. Megjegyzést) használó Newton-Gauss-Seidel-féle iterációs eljárást a

$$\nabla f(\boldsymbol{y}) = \boldsymbol{0}, \qquad \boldsymbol{y} \in \boldsymbol{Y}$$

  nemlineáris egyenletrendszer megoldására (azaz stacionárius pont keresésére) $\boldsymbol{Y}$-on [25, 52].

  Az aktuális $\boldsymbol{Y}$ boxra végrehajtott iterációs lépés eredménye háromféle lehet. Először is, megállapítjuk, hogy $\boldsymbol{Y}$ nem tartalmaz stacionárius pontot, és ekkor $\boldsymbol{Y}$-t törölhetjük. Másodszor, sikerül $\boldsymbol{Y}$-t valamely komponensei mentén 'összenyomni', így $F(\boldsymbol{Y})$-t pontosabban határozhatjuk meg, hatékonyabban működhet a többi gyorsító eljárás. Harmadszor, $\boldsymbol{Y}$-ból több kisebb boxot állítunk elő az eredeti box egyes részeinek elhagyásával (a több részre darabolódás egyébként az intervallumokon értelmezett kiterjesztett aritmetikai osztás műveletnek tudható be). Ha $\boldsymbol{Y}$ n-dimenziós, a teszt során legfeljebb $n + 1$ új boxot kaphatunk (az [52]-beli implementációt alkalmazva). Az iterációnak csak egy lépését használjuk, mert ez a teszt viszonylag időigényes, és nem képes megállapítani, valóban globális, vagy csak egy lokális minimumpontot közelítünk-e.

6. MEGJEGYZÉS. *A gyorsítóteszteknek az algoritmusba történő beillesztésekor mindig figyelemmel kell lennünk az egyes tesztek számítási igényére. Pl. egy Hesse-mátrixot használó teszthez csak akkor számítsuk ki a másodrendű derivált befoglalását, ha a tekintett boxról már megállapítottuk, hogy azt nem tudjuk teljes egészében redukálni az 'olcsóbb' eljárásokkal.*

Jelen értekezés 4. Fejezetében olyan speciális gyorsító eljárásokat ismertetek, melyek – az általános módszerek nehézkes használata, illetve a hatékonyság megnövelése miatt – a tekintett feladatosztály speciális tulajdonságait használják ki. További, esetünkben nem alkalmazott gyorsítótesztek találhatók még [19, 23, 63]-ban.

1.6.6 Intervallum-kiválasztási szabályok

Az algoritmus 17. lépése tartalmazza azt a szabályt, amely a következő iterációs lépésben felosztandó boxot határozza meg (amennyiben az illető boxra nem teljesül a megállási feltétel). Az algoritmusban a Munkalistabeli elemeket ezen szabály szerint érdemes rendezni (azaz az elemeket $\mathcal{L}_{\mathcal{W}}$-be beilleszteni), így az intervallum-kiválasztás során egyszerűen a Munkalista első eleme kerül kiválasztásra.

7. MEGJEGYZÉS. *A Munkalista reprezentálása – [22] alapján – egyszeresen láncolt listával történt a 2. Fejezet algoritmusában. A 3. és 4. Fejezetekben (Casado javaslata alapján) fejlettebb, és az ottani algoritmusokban a listaműveletek szempontjából hatékonyabb kiegyensúlyozott bináris keresőfákat, nevezetesen AVL-fákat használtam. Ennek ellenére a 'Munkalista' elnevezést a keresőfás reprezentációknál is megtartottam.*

A tradicionálisan használt két intervallum-kiválasztási szabály az alábbi:

'Moore–Skelboe-szabály: Válasszuk a következő lépésben felosztásra azt az $\boldsymbol{Y}$ Munkalistabeli boxot, melyre $\underline{F}(\boldsymbol{Y})$ minimális [42, 51, 58]. Ezen intervallum-kiválasztási szabállyal az algoritmus egyfajta 'először a legjobbat' (best-first) keresést végez.

'Hansen-szabály: Válasszuk felosztásra azt a boxot, amely a legrégebben szerepel a Munkalistán (ezzel 'szélességi keresést' implementálva) [24]. Ha feltételezzük, hogy a gyorsító lépések csak teljes egészében törölhetnek boxokat, akkor a Hansen-szabállyal a legnagyobb térfogatú boxot egyikét választjuk ki felosztásra.

A kiválasztás szabályai jelentősen befolyásolják az algoritmus viselkedését. A Moore–Skelboe-, illetve Hansen-szabályt használó algoritmusok aszimptotikus viselkedését úgy tanulmányozhatjuk, hogy feltételezzük, hogy az 1. Algoritmus 11. lépésében a megállási kritérium sohasem teljesülhet. Az algoritmus gyorsító lépéseinek elhagyása esetén, valamint a klasszikus 'A' felosztási irányválasztás és felező intervallum-felosztás mellett ekkor az alábbi tulajdonság igazolható:

6. TÉTEL. *Ratschek és Rokne [51]: Ha a célfüggvény befoglaló függvénye zéró-konvergens, akkor mind a Moore–Skelboe-, mind a Hansen-szabályt használó algoritmusváltozatok esetén a globális minimum alsó becslése a minimum értékéhez tart, azaz* $\min_{\boldsymbol{Y}\in\mathcal{L}_{\mathcal{W}}}\underline{F}(\boldsymbol{Y}) \to f^*$.

A fenti algoritmusváltozatok további tulajdonságairól a 2. Fejezetben lesz szó.

A gyakorlati tapasztalatok szerint a két klasszikus módszer közül a Moore–Skelboe változat általában gyorsabb (ezt a 3. Fejezetbeli vizsgálatok egy 'mellékhatásaként' jelen értekezésben is alátámasztom).

Az utóbbi években napvilágot látott egy érdekes intervallum-kiválasztási szabály, mely a globális minimum becsült értéke és a feldolgozásra váró boxokon számított célfüggvényérték alapján választ. A szabály alapjául szolgáló mutatót korlátozás nélküli feladatokra dolgozták ki. A módszertan továbbfejlesztését korlátozó feltételeket tartalmazó, azaz a (**) alakú problémák esetén részletesen vizsgálom a 3. Fejezetben, így az említett mutatóról abban a részben lesz bővebben szó.

1.6.7 Megállási feltételek

Az algoritmus 11. lépésében azt vizsgáljuk, hogy a Munkalistából kiválasztott $\boldsymbol{Y}$ box teljesít-e egy olyan feltételt, amely alapján joggal feltételezhetjük róla, hogy globális minimumhely(ek)et tartalmaz. Ha ez a feltétel teljesül, akkor a box az ún. Eredménylistára kerül. A bemutatott algoritmusokban feltételként a $w(F(\boldsymbol{Y})) < \varepsilon$ összefüggést használtam. Azaz, ha a boxot egyetlen gyorsító eljárás sem tudta törölni, de a rajta számított függvénybefoglalás szélessége már elég kicsi, akkor az Eredménylistára kerülhet. Az ε értéke az értekezésben 10^{-2} és 10^{-12} között változik a konkrét numerikus vizsgálat céljától, illetve a vizsgált feladatok nehézségétől függően.

Az irodalomban számos helyen a vizsgált box a szélességére vonatkozó összefüggést használnak (esetleg a fenti feltétellel együtt). Ezt a feltételt vizsgálataimban nem használtam, aminek az az oka, hogy a tesztfüggvények között számos olyan szerepel, amely 'lapos', ám ezzel együtt nagy befoglaló túlbecslést ad a minimumhely(ek) közelében. Ezen feladatokon a box szélességére vonatkozó feltevés a minimumhelyek körüli tartomány feleslegesen sok részre történő feldarabolását eredményezné. A konkrét alkalmazásokhoz a jelenleg beépített megállási feltételek esetleges módosítása szükség esetén könnyen elvégezhető.

2. Fejezet

Multisection felosztási szabályok

Ebben a fejezetben a klasszikus intervallum-felosztási (bisection) eljárások azon alternatíváiról lesz szó, melyekben az aktuálisan tekintett $\boldsymbol{Y} \in \mathbb{I}^n, \boldsymbol{Y} \subseteq \boldsymbol{X}_0$ boxot egy lépésben kettőnél több részre osztjuk. A fejezet első részében áttekintem a felezést alkalmazó algoritmusváltozatok viselkedésére vonatkozó korábbi eredményeket. Ezután az ún. *szeletelő* (multisplitting, egy irány mentén több részre osztó) felosztási szabályokra Csallner elméleti eredményeit [9] ismertetem. Végül néhány, több koordináta irány szerint felosztó (multisection) stratégiát, és az ezekre vonatkozó empirikus vizsgálataimat mutatom be [39] alapján.

2.1 A felezést használó algoritmusok konvergencia-sebessége

Az elméleti konvergencia-vizsgálatok során most is feltesszük, hogy az 1. Algoritmus megállási feltétele sohasem teljesül. Gyorsítótesztek alkalmazása nélkül az 'A' irányválasztással kombinált klasszikus felezés esetén a következő állítások igazolhatók (vö. 6. Tétel):

7. Tétel. *Ratschek és Rokne [51]: Ha a célfüggvény befoglaló függvénye nem izoton, akkor a Moore–Skelboe-szabállyal dolgozó algoritmus esetén a globális minimum alsó becslése tetszőlegesen lassan konvergálhat a minimumhoz.*

Csallner és Csendes [8]-ban legrosszabb eset analízis során a minimum és annak aktuális alsó becslésének eltérésére adott felső korlátot a Hansen-féle algoritmusban:

8. Tétel. *Csallner és Csendes [8]: Ha a célfüggvény befoglaló függvénye α-konvergens, akkor a Hansen-szabályt használó algoritmusra*

$$f^* - \min_{\mathbf{Z} \in \mathcal{L}_{\mathcal{W}} \cup \mathbf{Y}} \underline{F}(\boldsymbol{Z}) \leq c(2w(\boldsymbol{X}_0))^{\alpha}(k+1)^{-\alpha/n}, \tag{3}$$

ahol c a legkisebb pozitív konstans, amelyre az α-konvergencia teljesül, $\boldsymbol{Y}$ a k-adik iterációs lépésben felosztott box, $\mathcal{L}_{\mathcal{W}}$ pedig a Munkalista a k-adik lépés elején.

Továbbá, ha f befoglaló függvénye izoton, akkor ugyanez a felső korlát a Moore–Skelboe-szabállyal dolgozó algoritmusra is áll.

Jegyezzük meg, hogy $\mathcal{L}_{\mathcal{W}} \cup \boldsymbol{Y}$ az összes box halmazát adja a k-adik iterációs ciklus *kezdetén*, amikor a felosztásra kiválasztott $\boldsymbol{Y}$ box már nincs $\mathcal{L}_{\mathcal{W}}$-ben. Továbbá, a Moore–Skelboe felosztási szabály esetén $\min_{\mathbf{Z} \in \mathcal{L}_{\mathcal{W}} \cup \mathbf{Y}} \underline{F}(\boldsymbol{Z}) = \underline{F}(\boldsymbol{Y})$.

2.2 Szeletelést alkalmazó algoritmusok viselkedése

A több részre osztó eljárások felezéssel szembeni lehetséges előnyeiről, illetve hátrányairól a Bevezetésben már esett szó. Eszerint az $\boldsymbol{Y}$ több részre osztása esetén egyrészt több a részboxokból kinyerhető információ, melyekkel a minimum korlátai hatékonyabban javíthatók, másrészt pedig a kisebb részboxok eredményesebben vizsgálhatók a gyorsító eljárásokkal. Mindezek ára a felosztás irányainak kiszámítása, és a részboxokra meghívott eljárások összköltségének növekedése lehet (esetlegesen több boxot egyenként törlünk, pedig óvatosabb felosztási stratégiával egyben is törölhetnénk a kérdéses tartományt). A multisplitting és a multisection közötti döntő különbség az, hogy multisplitting alkalmazásával az elsőként említett hátrányt kiküszöbölhetjük az előnyök megtartása mellett: egy irány menti darabolásnál elég – a felezéshez hasonlóan – a legkedvezőbbnek tűnő irányt meghatároznunk. Csallner és munkatársai [9] Hansen kiválasztási szabályát vizsgálják, $\boldsymbol{Y}$-t pedig a klasszikus 'A' szabály által adott irányra merőlegesen $s > 1$ egybevágó részboxra szeletelik. A kezdő intervallumról feltételezhetjük, hogy ún. s-kvázi hiperkocka, azaz minden komponense kisebb, mint a többi komponensének s-szerese. Így az algoritmus futása az alábbi definíció alapján *szint*ekre bontható:

11. DEFINÍCIÓ. *A k-adik iterációs lépés az l ($l \geq 0$) szinten hajtódik végre, ha a k-adik ciklus felosztó lépése előtt az $\mathcal{L}_{\mathcal{W}}$-ben található boxok $w_{\max}$ maximális szélességére $s^{-(l+1)}w(\boldsymbol{X}_0) < w_{\max} \leq s^{-l}w(\boldsymbol{X}_0)$ teljesül.*

Tekintsük elsőként a gyorsítótesztek nélküli algoritmusváltozatot. Az alábbi két egyszerű tétel az l-edik szinten végrehajtott iterációs lépésekről szolgáltat információt:

9. TÉTEL. *Csallner, Csendes és Markót [9]: Az l-edik iterációs szinthez tartozó iterációs lépések száma*

$$k_l = \frac{s^{nl}(s^n - 1)}{s - 1}.$$

10. TÉTEL. *Csallner, Csendes és Markót [9]: A k-adik iterációs lépés pontosan akkor hajtódik végre az l-edik szinten, ha*

$$\frac{s^{nl} - 1}{s - 1} + 1 \leq k \leq \frac{s^{n(l+1)} - 1}{s - 1}. \tag{4}$$

Az algoritmus konvergencia-sebességét a 8. Tételhez hasonlóan f^* és annak legjobb alsó becslésének eltérésével jellemezhetjük. [9] alapján egyszerűen látható, hogy a legrosszabb esetben k egy adott szinthez tartozó utolsó iterációs lépés (azaz k a (4)-beli felső korláttal egyezik meg). Ekkor az alábbi igaz:

11. TÉTEL. *Csallner, Csendes és Markót [9]: Ha F α-konvergens befoglaló függvénye f-nek az $\boldsymbol{X}_0$ boxon, akkor a legrosszabb esetben*

$$f^* - \min_{\mathbf{Z} \in \mathcal{L}_{\mathcal{W}} \cup \mathbf{Y}} \underline{F}(\boldsymbol{Z}) \leq cw^{\alpha}(\boldsymbol{X}_0)s^{\alpha}(k(s-1)+1)^{-\alpha/n}, \tag{5}$$

ahol c a legkisebb pozitív konstans, amelyre az α-konvergencia teljesül, $\boldsymbol{Y}$ a k-adik lépésben felosztandó box, $\mathcal{L}_{\mathcal{W}}$ pedig a Munkalista a k-adik lépés elején.

A tétel eredményét a felezést használó algoritmus viselkedésével összevetve azt kapjuk, hogy a legrosszabb esetben a (3)-beli $\mathcal{O}(k^{-\alpha/n})$ rendű becslés s részre szeletelés esetén $\mathcal{O}(s^{\alpha(n-1)/n}k^{-\alpha/n})$-re módosul. Mindez $s \geq 2$, illetve $\alpha > 0$ miatt a multisplitting esetén lassabb konvergenciára utal. Érdemes továbbá megjegyezni, hogy egydimenziós feladatok esetén ($n = 1$) a konvergencia-sebességet nem befolyásolja a több részre szeletelés.

A fenti algoritmusváltozat esetében a legjobb becslést akkor kaphatjuk, ha a k-adik iteráció egy adott szinten végrehajtott első iterációs lépés (vagyis k a (4)-beli alsó korláttal egyezik meg). Ekkor az alábbi bizonyítható:

12. TÉTEL. *Csallner, Csendes és Markót [9]: Ha F az f α-konvergens befoglaló függvénye az $\boldsymbol{X}_0$ boxon, k pedig valamely l szint első iterációs lépésének indexe, akkor*

$$f^* - \min_{\mathbf{Z} \in \mathcal{L}_{\mathcal{W}} \cup \mathbf{Y}} \underline{F}(\boldsymbol{Z}) \leq cw^{\alpha}(\boldsymbol{X}_0)((k-1)(s-1)+1)^{-\alpha/n}, \tag{6}$$

ahol c a legkisebb pozitív konstans, amelyre az α-konvergencia teljesül, $\boldsymbol{Y}$ a k-adik lépésben felosztandó box, $\mathcal{L}_{\mathcal{W}}$ pedig a Munkalista a k-adik lépés elején.

Az optimumra adott alsó becslés pontossága ekkor $\mathcal{O}((ks)^{-\alpha/n})$ rendű, ami s növelésével csökken, azaz rögzített k-ra több részre osztás esetén pontosabb becslést kaphatunk.

A következő lépésben a gyorsítóteszteket használó algoritmus variánsokat tekintjük. A gyorsító lépések hatékonyságának jellemzésére bevezetjük az η-gyorsított algoritmus fogalmát, mely esetén azzal a feltételezéssel élünk, hogy a gyorsítótesztek minden iterációs *szinten* a végigvizsgált boxok *legalább* $1 - \eta$ $(0 < \eta \leq 1)$ részét teljes egészében eliminálják. A feltevés létjogosultságát empirikus vizsgálatok során igazoltuk ugyancsak [9]-ben, egy általánosan használt nagy tesztfeladat-halmazon a monotonitási, konkavitási és kivágási tesztek alkalmazása esetén.

A k és l értékek kapcsolatának meghatározásához szükségünk van az η-feltevés mélyebb vizsgálatára. Vegyük észre, hogy az l-edik szint iterációs lépéseinek száma függ attól, hogy az illető szint mely lépéseiben törlünk boxokat. Könnyen látható, hogy az l-edik szinten akkor hajtjuk végre a legkevesebb iterációt, ha az összes törlendő boxot a szinthez tartozó első iterációs lépésekben töröljük (legjobb eset), a legtöbbet pedig akkor, ha csak a szint utolsó lépéseiben törlünk boxokat (legrosszabb eset). [9] a két esetre az alábbi korlátokat adja:

13. TÉTEL. *Csallner, Csendes és Markót [9]: Az η-gyorsított algoritmus l-edik szintjéhez tartozó iterációs lépések k_l számára a legjobb esetben*

$$k_l \leq \frac{s^n - 1}{s - 1} \lceil \eta \lfloor (\eta s^n)^l N \rfloor \rceil,$$

míg a legrosszabb esetben

$$k_l \leq \frac{s^n - 1}{s - 1} \lfloor (\eta s^n)^l N \rfloor,$$

ahol $\lceil \cdot \rceil$, illetve $\lfloor \cdot \rfloor$ az alsó, illetve felső egészrészt jelenti, N pedig az első szint befejezésének pillanatában a megmaradt (nem törölt) boxok száma.

A következő állításban a 13. Tétel legrosszabb eset vizsgálata alapján egy szükséges feltételét adjuk annak, hogy a k-adik iteráció az l-edik szinthez tartozik:

14. TÉTEL. *Csallner, Csendes és Markót [9]: Ha a k indexű iterációs lépést az l-edik szinten hajtjuk végre, akkor (i) $\lceil \eta s^n \rceil > 1$ esetén*

$$k \leq \frac{s^n - 1}{s - 1} \frac{\lceil \eta s^n \rceil^{l+1} - 1}{\lceil \eta s^n \rceil - 1} N,$$

míg (ii) $\lceil \eta s^n \rceil = 1$ esetén

$$k \leq \frac{s^n - 1}{s - 1}(l + 1)N$$

teljesül.

(A $\lceil \eta s^n \rceil = 1$ eset meglehetősen speciális, ennek vizsgálatát a bizonyítás technikája indokolja.) Érdemes megemlíteni, hogy a gyorsítóteszteket használó algoritmus esetén a 10. Tétel analógiájára nem tudunk alsó becslést adni k-ra, hiszen az η-feltevés definíciója miatt a szintenként megmaradó boxok számára csupán egyirányú (felső) becslésünk lesz. A fenti tételek alapján az η-gyorsított algoritmus elméleti konvergencia-sebességére a legrosszabb esetben a következő bizonyítható:

15. TÉTEL. *Csallner, Csendes és Markót [9]: Legyen F az f α-konvergens befoglaló függvénye az $\boldsymbol{X}_0$ boxon. Tegyük fel továbbá, hogy a k-adik iterációt az l-edik szinten hajtjuk végre. Ekkor az η-gyorsított algoritmusváltozatra*
(i) $\lceil \eta s^n \rceil > 1$ esetén

$$f^* - \min_{\mathbf{Z} \in \mathcal{L}_{\mathcal{W}} \cup \mathbf{Y}} \underline{F}(\boldsymbol{Z}) = \mathcal{O}(\eta^{\alpha l/n} s^{\alpha(n-1)/n} k^{-\alpha/n}), \tag{7}$$

míg (ii) $\lceil \eta s^n \rceil = 1$ esetén

$$f^* - \min_{\mathbf{Z} \in \mathcal{L}_{\mathcal{W}} \cup \mathbf{Y}} \underline{F}(\boldsymbol{Z}) = \mathcal{O}(s^{-\alpha l/n} k^{-\alpha l/n} l^{\alpha l/n}) = \mathcal{O}\Big(\Big(\frac{l}{sk}\Big)^{\alpha l/n}\Big). \tag{8}$$

A kapott eredmények az alábbiakban foglalhatók össze: (7)-ben az s és k paraméterek nagyságrendjei megegyeznek a gyorsítótesztek nélküli algoritmus esetén kapottakkal (vö. 11. Tétel). A gyorsítótesztek hatása (7) $\eta^{\alpha l/n}$ tényezőjében jut érvényre. Az η növelésével (azaz a gyorsítótesztek kevésbé hatékony működésekor) a többi paraméter változatlansága mellett az optimum becslésének pontossága csökken; az $\eta = 1$ eset természetesen az (5)-beli konvergencia-sebességet adja. Továbbá, rögzített α és n esetén a befoglalás pontossága a szintek emelkedésével η-ban polinomiálisan javul: minél magasabb iterációs szinten vagyunk, annál gyorsabb az optimum becslésének konvergenciája.

2.3 A multisection szabályok numerikus vizsgálata

A következőkben néhány, a gyakorlatban ígéretesnek tűnő multisection szabály numerikus vizsgálatát ismertetem [39] alapján. Vizsgálataimban az 'A', 'B', 'C' és 'D' irányválasztási szabályok (1.6.3 szakasz) mindegyikét különböző intervallum-felosztási stratégiákkal kombináltam. (A különböző irányválasztási szabályok viselkedésére, illetve az ezeket használó algoritmusok konvergenciájára vonatkozóan lásd az 5. Tételt.) Csendes és Ratz ([15, 54]) korábban numerikus vizsgálatokkal is elemezte a klasszikus felezéssel kombinált felosztási irányválasztó szabályokat. Ezek eredményeit és a belőlük levont következtetéseket jelen vizsgálat részeként – némiképp eltérő algoritmus beállítások mellett – megerősítjük.

Hardver-szoftver környezet. A numerikus teszteket több különböző platformon is elvégeztem, a végleges változatban használt számítógép egy 133 MHz-es Pentium processzorral és 64 MB RAM-mal rendelkező PC volt. A futtatás Linux operációs rendszer alatt történt, a B&B algoritmus alapelemeit a Numerical Toolbox for Verified Computing [22] moduljai alapján állítottam össze, míg az intervallumos műveleteket és adattípusokat a PROFIL/BIAS [31] csomag biztosította. (Érdekes adalék, hogy a Toolbox könyvtár eredetileg az intervallum aritmetikát támogató C–XSC nyelven [29] készült. A tesztek futtatásának időpontjában azonban a C–XSC-nek csupán a DOS alatti, Borland C fordító alá készült változata állt rendelkezésünkre, ez azonban nem biztosított megfelelő lehetőségeket. Egy tesztsorozat erejéig a karlsruhei egyetem HP 9000-730 munkaállomását is igénybe vehettük a C–XSC használatával együtt, végül azonban a PROFIL/BIAS használata mellett döntöttem. Így a szükséges Toolbox modulokat át kellett írnom ez utóbbi intervallumos környezetbe. Jelen értekezés legtöbb numerikus vizsgálata is ezt a 'vegyes' alapot használja. Mára a helyzet megváltozott, a C–XSC-t a legújabb igényekhez – és fordítókhoz – igazodva folyamatosan fejlesztik, így szóba kerülhet a C–XSC alá történő visszatérés. Érdemes megemlíteni, hogy a C–XSC és a PROFIL/BIAS által adott függvénybefoglalások túlbecslései az előbbi csomag esetén számottevően kisebbek. A PROFIL/BIAS használatával készült programok viszont tapasztalataim alapján a standard feladatokon kb. 5–10-szer gyorsabbak voltak.)

A különböző hardver–szoftver környezetek összehasonlítására szolgáló Standard Időegység (Standard Time Unit, STU [16]), azaz a Shekel–5 tesztfüggvény [62] 1000-szeri kiértékelésének időigénye a $(4,4,4,4)^T$ pontban 0.0125 másodperc volt valós adattípusok (double) használata esetén.

Algoritmus beállítások. A branch–and–bound algoritmust az alábbiak szerint specifikáltam: a célfüggvény befoglaló függvényének megadása a természetes intervallum-kiterjesztéssel illetve középérték formulák használatával történt, a derivált és Hesse-mátrix értékeket automatikus differenciálással határoztam meg.

Gyorsítótesztként a kivágási és monotonitási teszteket szerepeltettem, emellett a Newton-lépés végrehajtását (és így a Hesse-mátrix kiszámítását) csak egy első ránézésre teljesen meglepő feltétel fennállása esetén végeztem el: Nevezetesen, ha a felosztáskor keletkező s darab box közül csak egy boxot nem lehetett teljes egészében törölni az 'olcsóbb' tesztek valamelyikével, akkor az egyetlen megmaradt box nem törölt területeit befoglaló boxra kiszámítottam a Newton-iteráció egy lépését. (Ezután

a már rendelkezésre álló Hesse-mátrix alapján érdemes volt elvégezni az egyszerűen kivitelezhető konkavitási tesztet is.) Így a Newton-lépés végrehajtása azokban az esetekben történt meg, amikor *a többi teszt* működött jól. A módosítás lényege az, hogy a felosztás után kapott részboxok számától függetlenül maximálni lehet (egy iteráción belül) a Hesse-mátrix befoglalások kiszámításának költségét. Ennek az ötletnek a következtében az alkalmazott algoritmus hatékonysága meglepő módon megnőtt a Newton-lépést mindig elvégző, vagy egyáltalán nem használó változatokhoz képest.

A gyorsítótesztek elvégzése után megmaradó $\hat{Y}^i$ box a $w(F(\hat{Y}^i)) < 0.01$ megállási feltétel teljesülése esetén került $\mathcal{L}_S$-be. Emellett $\mathcal{L}_S$ méretét 100-ban korlátoztam (ennek elérésekor az algoritmus megállt), mivel néhány tesztfeladat esetén a globális minimumhelyek száma igen magas, így az algoritmusok futásideje a fenti megállási feltétellel aránytalanul nagy lett volna. (Emlékezzünk rá, az algoritmus futása alatt nem törlünk minimumhelyet tartalmazó tartományt, így a terminálláskor $\mathcal{L}_W$-t is figyelembe véve megadhatók az összes minimumpont, illetve a minimum garantált befoglalásai.) Az algoritmus megállása után az $\mathcal{L}_S$-beli elemekre lefuttattam egy ugyancsak az intervallumos Newton-iteráción alapuló ellenőrző lépést [52, 53], melynek célja a tekintett boxban található minimumpont unicitásának és stacionárius mivoltának vizsgálata. A legtöbb feladat eredmény boxai esetén ez az ellenőrzés a kívánalmaknak megfelelően sikeres volt, és a Newton-iteráció sokszor több nagyságrenddel csökkentette a minimumhely befoglalásának pontosságát.

Az algoritmus intervallum-kiválasztási szabálya a Moore–Skelboe stratégia volt, jelen vizsgálat fő tárgya, a boxok felosztása pedig az alábbi szabályok szerint történt:

- /2 szabály: a klasszikus felezés módszere ($s = 2$);
- /3 szabály: $\boldsymbol{Y}$ 3 részre osztása történik meg ($s = 3$); először az irányválasztási szabály szerinti legígéretesebb irányban felezünk, majd az egyik kapott boxot tovább felezzük a második legígéretesebb irányban;
- /4 szabály: ez esetben a két legjobbnak vélt irányban egyidejűleg felezünk, ezzel $s = 4$ részboxot létrehozva.

Érdemes hangsúlyozni, hogy a 2.2 szakaszban tárgyalt *szeletelő* eljárások $s = 3$-ra, illetve $s = 4$-re adódó változatait is vizsgáltam numerikus úton. A több irányt használó felosztó eljárások ez utóbbiaknál azonban számottevően jobbnak bizonyultak, így az értekezésben a fenti szabályok bemutatására szorítkozom. ($s = 2$ esetén a természetesen a szeletelés megegyezik a klasszikus felező /2 szabállyal.)

A numerikus teszteket egy, a széles szakmai kör által tanulmányozott és elfogadott feladatcsoporton hajtottam végre. A feladatok [15, 24, 54, 62]-ből származnak, és jórészt megegyeznek az irányválasztási szabályokat vizsgáló [15]-ben használtakkal: ez utóbbi publikáció feladatcsoportjából kivettünk néhány nem kétszer folytonosan differenciálható problémát, ugyanakkor bevontunk néhány Ratz [54] által definiált nehezebb feladatot. A használt, összesen 37 feladat a következő, zárójelben a feladat dimenziószámával: Shekel-5, 7, 10 (4), Hartman-3, 6 (3, ill. 6), Goldstein–Price (2), Six-Hump-Camel-Back (2), Branin RCOS (2), Rosenbrock-2, 5 (2, ill. 5), Three-Hump-Camel-Back (2), Levy-3 (2), Levy-5 (2), Levy-8–12 (3–10), Levy-13–18 (2–7),

Schwefel-2.1 = Beale (2), Schwefel-3.1 (3), Schwefel-3.1p (3), Schwefel-2.5 = Booth (2), Schwefel-2.18 = Matyas (2), Shwefel-3.2 (3), Schwefel-3.7/5, 3.7/10 (5 ill. 10), EX1 (2), Griewank-5, 7 (5, ill. 7), Ratz-4 (2), Ratz-5, 6 (3, ill. 5).

Numerikus eredmények. A különböző intervallum-felosztási szabályokat az 'A'–'D' szabályok mindegyikével együtt használtam, vagyis minden feladatot összesen 12-szer oldottam meg. A kapott egyedi algoritmus változatokra az irányválasztási, illetve felosztási szabály rövidítésének összekapcsolásával hivatkozom, tehát pl. B/2 a 'B' szabállyal kombinált klasszikus felezést jelenti. Az 1. Táblázat három hatékonysági mutató értékeit tartalmazza: az egyes oszlopcsoportok az összesített időigényt (CPU, másodpercben – Sum), a tárigény jellemzéséhez a Munkalisták feladatonkénti maximális hosszainak átlagát (MLL – Av.), valamint az intervallumos célfüggvény kiértékelések összes számát (NFE – Sum) mutatják. Mindhárom mutató esetén a táblázat tartalmaz két viszonyszámot is; az AoP (average of percentages) oszlop az A/2 módszerhez képest az egyes feladatokon számított százalékos arányok átlagát, a /(A/2) oszlop pedig az illető mutató aggregált értékeinek az A/2 szabályhoz viszonyított arányát mutatja. Az AoP mutató esetén tehát minden tesztfeladatot ugyanakkora súllyal veszünk figyelembe, míg a /(A/2) értékek inkább azt jellemzik, hogyan viselkednek az egyes algoritmus változatok nehéz feladatokon. Így ez utóbbiak fontosabbnak tekinthetők az elemzés során.

A legfontosabb numerikus jellemző a futás CPU ideje, mely legtöbb esetben a célfüggvény, gradiens, illetve Hesse-mátrix kiértékelések számával arányos. Kivételt a nagy memóriaigényű feladatok jelenthetnek, melyeknél a listakezelés műveletei is számottevő időt vehetnek igénybe. Az 1. Táblázat alapján megállapíthatjuk, hogy a 'B' és 'C' irányválasztási szabályok minden felosztó stratégia esetén kedvezőbbek a másik két szabálynál. (A felezés esetén kapott értékek megerősítik a [15, 54]-beli eredményeket a 'B' és 'C' szabályok általános előnyeire vonatkozóan.) A futásidők /(A/2) oszlopa alapján az 'A', 'B' és 'C' szabályokra az $s = 3$ választás tűnik a legjobbnak, míg a 'D' (legkevésbé javasolható) irányválasztási módszer esetén a felezés a kedvezőbb. A két legkevesebb időt a C/3 és C/4 változatok esetén mértem, ezek egyaránt 88%-kal voltak gyorsabbak az alapul vett A/2 variánsnál. Fontos megjegyezni, hogy a két multisection felosztási szabály használatával a felezéshez képest a 'C' szabály esetén 22%, míg a 'B' esetén 17%, illetve 18% százalékos javulást tapasztalhatunk. Az AoP mutató szerint a futásidő tekintetében a C/3, valamint C/2 variánsokat találtam a legkedvezőbbnek 80, illetve 81 százalékkal. Az AoP értékek a 'B' és 'C' szabály minden változata esetén lényegesen nagyobbak a /(A/2)-nél, ami ezen stratégiák nehezen megoldható feladatokon mutatkozó előnyét jelenti.

A Munkalisták maximális hosszai (MLL) az egyes algoritmus-változatok tárigényét jellemzik. Hasonlóan a CPU időkhöz, a listahosszak átlagát, illetve ezeknek az A/2 változathoz képest számított arányait döntően a nehezebb feladatok határozzák meg. Az AoP, Av., és /(A/2) mutatók szerint a multisectiont használó algoritmusok egy kivételével minden esetben némiképp több tárat igényeltek a felező szabályokhoz képest. Ez a többlet tárigény azonban az alacsony listaméretek miatt nem jelent számottevő hátrányt. (Érdemes megjegyezni, hogy a maximális listahosszak jelen tesztsorozatban kb. 50%-kal alacsonyabbak, mint a valamivel egyszerűbb, kevesebb gyorsítótesztet alkalmazó [15]-beli algoritmusok esetén.)

Szabály		CPU			MLL			NFE		
		AoP	Sum	/(A/2)	AoP	Av.	/(A/2)	AoP	Sum	/(A/2)
	/2		4 180			228			502 360	
A	/3	101%	4 134	99%	125%	276	121%	103%	411 439	82%
	/4	110%	5 613	134%	147%	360	158%	113%	434 574	87%
	/2	82%	675	16%	89%	71	31%	83%	131 822	26%
B	/3	84%	551	13%	104%	78	34%	86%	107 046	21%
	/4	100%	562	13%	130%	92	40%	104%	105 036	21%
	/2	81%	666	16%	88%	71	31%	82%	132 395	26%
C	/3	80%	518	12%	103%	73	32%	84%	102 674	20%
	/4	97%	521	12%	128%	84	37%	101%	98 950	20%
	/2	136%	5 369	128%	142%	299	131%	129%	664 474	132%
D	/3	112%	8 830	211%	141%	409	179%	112%	741 433	146%
	/4	110%	12 871	308%	148%	579	254%	111%	801 293	160%

1. Táblázat: 37 tesztfeladat megoldásának eredménye különböző irányválasztási, illetve felosztási szabályok esetén. A táblázat Szabály oszlopa az alkalmazott stratégiákat, míg CPU a futási időre, MLL a munkalista maximális hosszára, NFE a célfüggvény kiértékelések számára vonatkozó adatokat mutatja: ezeken belül Sum az egyes feladatokon kapott adatok összegét, míg Av. az átlagát jelenti. Az AoP oszlop az A/2 módszerhez képest az egyes feladatokon számított arányszámok átlagát, a /(A/2) oszlop pedig a mutató aggregált értékeinek az A/2 szabályhoz viszonyított arányát jelöli.

A célfüggvény-számítások alakulása nagyrészt megegyezik az időigényeknél tapasztaltakkal. Itt is a 'C' és 'B' irányválasztási szabályok voltak a leghatékonyabbak, a Sum és /(A/2) mutatók alapján pedig megállapíthatjuk, hogy nehéz problémák megoldásakor a több részre osztás lényegesen csökkenti a függvényhívások számát. A C/2 és C/4 változatokat összevetve ez a csökkenés több mint 25 százaléknyi. Az összes algoritmus variánst tekintve az A/2 szabályhoz képest a C/3, C/4 változatok jelentették a legnagyobb, mintegy 80%-os csökkenést. Másrészt az 'A', 'B', 'C' irányválasztás esetén a multisection az AoP értékek tekintetében (azaz a könnyebb és nehezebb feladatoknak egyforma súlyt adva) nem jelent előnyt. (A felhasználó számára nyilván a nehéz feladatok megoldásakor jelentkező javulások a fontosabbak.)

A 2. Táblázat a gradiens, illetve Hesse-mátrix számítások, valamint az iterációs lépések számának alakulását foglalja össze az 1. Táblázat felépítéséhez hasonlóan. Figyeljük meg, hogy az NFE, NGE és NHE Sum értékek különbözőek az egyes algoritmus változatokon belül is. Ennek az az oka, hogy – a korábban említettek alapján – minden vizsgált boxon először mindig a csak célfüggvény-számítást igénylő kivágási tesztet hajtottam végre. Ha a box nem volt törölhető, akkor a további gyorsító lépések hívásakor került sor (esetleg ismételt célfüggvény kiértékelés mellett) a magasabbrendű deriváltak kiszámítására. Figyeljük meg, hogy a Sum mutatók esetén az NGE/NFE, illetve NHE/NFE arányok meglehetősen stabilan alakulnak: előbbi 56–65%, utóbbi 3–8% közötti az egyes változatokon. Ez ismételten a 2.2 szakaszbeli η-feltevés empirikus bizonyítékának tekinthető (jóllehet az arányokat esetünkben

Szabály		NGE			NHE			IT		
		AoP	Sum	/(A/2)	AoP	Sum	/(A/2)	AoP	Sum	/(A/2)
A	/2		289 016			39 243			79 422	
	/3	108%	254 050	88%	96%	24 809	63%	88%	57 964	73%
	/4	123%	283 938	98%	86%	15 852	40%	88%	58 545	74%
B	/2	84%	76 234	26%	82%	10 185	26%	81%	19 151	24%
	/3	92%	66 296	23%	76%	7 071	18%	71%	13 511	17%
	/4	114%	69 046	24%	75%	5 344	14%	80%	12 501	16%
C	/2	84%	76 794	27%	82%	10 500	27%	81%	19 053	24%
	/3	89%	63 629	22%	74%	6 860	17%	70%	12 863	16%
	/4	110%	64 918	22%	70%	5 123	13%	78%	11 628	15%
D	/2	126%	373 084	129%	119%	44 789	114%	137%	112 829	142%
	/3	118%	440 675	152%	101%	41 943	107%	98%	104 188	131%
	/4	121%	522 237	181%	78%	21 592	55%	87%	114 472	144%

2. Táblázat: 37 tesztfeladat megoldásának eredménye különböző irányválasztási, illetve felosztási szabályok esetén. A táblázat Szabály oszlopa az alkalmazott stratégiákat, míg NGE a gradiens, NHE a Hesse-mátrix kiértékelések számára, IT pedig a végrehajtott iterációs lépések számára vonatkozó adatokat mutatja: ezeken belül Sum jelenti az egyes feladatokon kapott adatok összegét. Az AoP oszlop az A/2 módszerhez képest az egyes feladatokon számított arányszámok átlagát, a /(A/2) oszlop pedig a mutató aggregált értékeinek az A/2 szabályhoz viszonyított arányát jelöli.

több más tényező, pl. a Newton-lépés bekapcsolásának stratégiája is befolyásolja).

A gradiensszámítások összes száma sok hasonlóságot mutat a célfüggvényhívásokéval: e mutató tekintetében is a 'C' és 'B' irányválasztások a legkedvezőbbek, a két legjobb variáns pedig a C/3 és C/4 változat (22% A/2-höz képest). A 'C' szabály esetén az $s = 3$ választás (C/3) 17%-os relatív javulást eredményezett a felezéshez (C/2) képest. A /(A/2) és az AoP oszlop adatait összevetve azt látjuk, hogy a gradienshívások számának csökkenése a nehezebb feladatokon számottevő.

A Hesse-mátrix számítások alakulása némileg eltérő tendenciát mutat: összességében most is a 'B' és 'C' szabályokat javasolhatjuk, viszont a multisection szabályok használatának előnye itt már nem csak a /(A/2), hanem az AoP arányokat tekintve is kimutatható mind a négy irányválasztási módszer esetén. A két legkisebb Sum értéket a C/4, illetve B/4 variánsoknál találhatjuk, továbbá a legjobb C/4 mintegy 51% javulást mutat az felező C/2 változattal összehasonlítva. Az NHE értékek ilyen alakulása azért is biztató, mert magasabb dimenziós feladatokon az erőforrásigény java részét a Hesse-mátrix számítások teszik ki. [1] vizsgálatai szerint a Hesse-mátrix számítások száma s növelésével sok esetben még tovább csökkenthető.

Az iterációs lépések száma közvetlenül ugyan nem befolyásolja az algoritmus komplexitását, mégis sok esetben értékes lehet a felhasználó számára (a 4. Fejezet egy módszerében pl. a végrehajtott iterációs lépések számában korlátozom az algoritmus futását). Az 'A', 'B', 'C' szabályokra $s = 3$ használata $s = 2$-höz képest kevesebb iterációs lépést eredményez. Ez egybevág azzal az intuitív meggondolással, hogy a

keresési tér kellően finom felosztásához kevesebb lépés szükségeltetik, amennyiben egy lépésben több részre osztunk. Ugyanakkor az iterációs lépések számának csökkenése az $s = 3$-ról $s = 4$-re való áttéréskor már nem figyelhető meg ilyen egyértelműen.

Az eredmények összegzéseként megállapíthatjuk, hogy a bemutatott multisection szabályok nehezen megoldható feladatok esetén számottevő mértékben javíthatnak az alapalgoritmus hatékonyságán. Összességében a C/3, illetve C/4 stratégiák tekinthetők a legkedvezőbbnek. A multisection előnye ugyanakkor mind a 'B', mind a 'C' szabályok esetén fennáll, azaz a tradicionális A/2 módszerhez képest az új típusú irányválasztási szabályok, illetve a több részre osztó szabályok erősítik egymás hatását. Ez különösen figyelemreméltó annak fényében, hogy az algoritmusokban kifinomult gyorsítóteszteket alkalmaztam. Nehéz feladatokon a multisection használata a legjobb felező szabályokhoz képest kb. 22, illetve 25 százalékkal csökkentette az időigényt, illetve a célfüggvényhívások számát. Emellett a multisectiont használó módszerek tárigénye nem nőtt jelentős mértékben a bisection változatokhoz képest.

Befejezésül érdemes néhány szót szólni az 1.6.3 szakaszbeli 'E' irányválasztással kombinált multisection módszerekről. A bemutatott tesztsorozat nem tartalmazta az 'E' szabályt, mert az minden felosztandó box esetén igényelné a Hesse-mátrix kiértékelését. Egy korábbi vizsgálatomban, ahol mindig kiszámításra került a Hesse-mátrix, az E/3, illetve E/4 szabályok viselkedése lényegében megegyezett a 'B' és 'C' stratégiával kombinált multisectionnel. A most bemutatott algoritmus-változatok hatékonysága azonban lényegesen jobbnak bizonyult e korai algoritmusokénál.

3. Fejezet

Egy új heurisztikus mutató vizsgálata

A fejezetben az egyenlőtlenség alakú korlátokat tartalmazó (∗∗) alakú feladatok megoldásával foglalkozom. Az áttekinthetőség kedvéért álljon itt újra a vizsgált feladat általános felírása:

$$\begin{array}{ll} \min & f(\boldsymbol{x}) \\ \text{s.t.} & g_j(\boldsymbol{x}) \leq 0,\ j = 1 \ldots, r, \\ & \boldsymbol{x} \in \boldsymbol{X}_0. \end{array} \qquad (**)$$

Egy $\boldsymbol{x} \in \boldsymbol{X}_0$ pont *lehetséges megoldás*, ha $g_j(\boldsymbol{x}) \leq 0, \forall j = 1, \ldots, r$, illetve *szigorúan lehetséges megoldás*, ha $g_j(\boldsymbol{x}) < 0, \forall j = 1, \ldots, r$. Jelölje $G_j : \mathbb{I}^n \to \mathbb{I}$ a g_j befoglaló függvényét $\boldsymbol{X}_0$-on $(j = 1, \ldots, r)$. Egy $\boldsymbol{Y} \subseteq \boldsymbol{X}_0$ boxról azt mondjuk, *biztosan kielégíti* a $g_j(x) \leq 0$ korlátot, ha $\overline{G_j}(\boldsymbol{Y}) \leq 0$, illetve *biztosan nem elégíti ki* a szóban forgó korlátot, ha $\underline{G_j}(\boldsymbol{Y}) > 0$. Egy $\boldsymbol{Y} \subseteq \boldsymbol{X}_0$ box *biztosan csak lehetséges megoldásokat tartalmaz*, ha biztosan kielégíti az összes korlátot, *biztosan nem tartalmaz lehetséges megoldást*, ha legalább egy korlátot biztosan nem elégít ki, illetve *meghatározatlan a lehetséges megoldásokra nézve* minden egyéb esetben. (A fenti definíciók bevezetését a G_j függvények esetleges túlbecslése indokolja.)

3.1 Az alapul vett mutató

Az utóbbi években Casado, Csendes, García és Martínez korlátozás nélküli – (∗) alakú – feladatok megoldása esetére javasolta a

$$pf^*(\boldsymbol{X}) = p(f^*, \boldsymbol{X}) = \frac{f^* - \underline{F}(\boldsymbol{X})}{w(F(\boldsymbol{X}))}$$

paraméter használatát, ahol f^* a globális minimumot jelenti. A pf^* mutató (illetve becslései) branch–and–bound algoritmusok intervallum-kiválasztási [6, 11] illetve felosztási [3] szabályaiban, memóriakorlátozott B&B eljárásokban [4], továbbá párhuzamos intervallumos optimalizáló algoritmusok vezérlésében [2] kapott szerepet.

A fenti mennyiség a globális minimum relatív elhelyezkedést mutatja a tekintett boxon vett célfüggvény-befoglalás értékén belül. Így, ha azzal a feltételezéssel élünk, hogy az F befoglaló függvény hozzávetőleg ugyanakkora túlbecslést ad az alsó és felső korlátjában, akkor pf^* egy heurisztikus becslést szolgáltat arra nézve, hogy a

vizsgált box mennyire ígéretes a globális minimumhely tartalmazása szempontjából. Kisebb pf^* értékkel rendelkező boxok kevésbé ígéretesek, míg magasabb pf^* értéket adó boxok esélyesebbek.

8. MEGJEGYZÉS. *A pf^* mutató nevezője a vizsgált boxon vett célfüggvényérték befoglalásának szélességét tartalmazza. A $w(F(\boldsymbol{X})) \neq 0$ összefüggés teljesülése – az intervallumos számítások kifelé kerekítései miatt – általában még pontszerű (0 szélességű) boxok esetében is igaz. Másrészt, a $w(F(\boldsymbol{X})) = 0$ esetben $\boldsymbol{X}$ vagy kizárólag szuboptimális, vagy kizárólag optimális célfüggvényértékű pontokat tartalmaz (f^* és $F(\boldsymbol{X})$ összevetéséből), így $pf^*(\boldsymbol{X})$ ennek megfelelően alkalmas módon definiálható.*

Jegyezzük meg, hogy a gyakorlatban a globális minimum értéke, f^* általában nem ismert. Ekkor a $pf^*(\boldsymbol{X})$ mutató helyett annak egy közelítése használható f^* valamely $\hat{f}$ becslése segítségével:

$$p(\hat{f}, \boldsymbol{X}) = \frac{\hat{f} - \underline{F}(\boldsymbol{X})}{w(F(\boldsymbol{X}))},.$$

Egy lehetséges közelítés az $\hat{f} := \tilde{f}$ módon adható, ahol $\tilde{f}$ a globális minimum aktuális legjobb felső becslése, azaz a kivágási teszt során használt érték. Amint azt látni fogjuk, az $\hat{f}$ értékek megválasztása döntő fontosságú lesz az előálló algoritmusok konvergenciájának szempontjából. Könnyen látható, hogy korlátozás nélküli feladatok esetén minden, az $\mathcal{L}_{\mathcal{W}}$-be beillesztett box esetén $p(\tilde{f}, \boldsymbol{X}) \in [0, 1]$ áll: $p(\tilde{f}, \boldsymbol{X}) < 0$ esetén ugyanis a kivágási teszt törli $\boldsymbol{X}$-et, a $p(\tilde{f}, \boldsymbol{X}) > 1$ eset pedig $\tilde{f}$ aktualizálása miatt zárható ki.

Érdekes módon korlátozásos problémáknál előfordulhat, hogy a $p(\tilde{f}, \boldsymbol{X})$ érték bizonyos, a lehetséges megoldások tartalmazására nézve meghatározatlan $\boldsymbol{X}$ boxokon nagyobb lesz, mint 1. Ez akkor állhat fenn, ha nem állíthatjuk teljes biztonsággal $\boldsymbol{X}$-ről azt, hogy legalább egy lehetséges megoldást tartalmaz, azaz, amikor a $\tilde{f}$ mutató aktualizálása nem lehetséges.

Csendes [11] az $\hat{f}$ értékeket az f^* becslésének sokféle lehetősége miatt egy általános $\{f_k\}_{k=1}^{\infty}$ sorozat aktuális tagjaiként tekintette, ahol a k-adik iterációs lépésben egy f_k érték alapján számítjuk az aktuális $p(\hat{f}, \boldsymbol{X}) = p(f_k, \boldsymbol{X})$ mennyiségeket. Az $\{f_k\}$ sorozat előállhat futási időben a B&B algoritmus 'belsejében' (mondjuk $\tilde{f}$ alapján), vagy lehet a felhasználó által kívülről megadott is.

Az 1. Algoritmus 17. lépésének, azaz a következő iterációban felosztandó box kiválasztásának (mely jelen fejezet egyik tárgya) két klasszikus stratégiája a Bevezetésben már ismertetésre került:

C1: Moore–Skelboe-szabály: válasszuk $\mathcal{L}_{\mathcal{W}}$ elemei közül a minimális $\underline{F}(\boldsymbol{X})$ értékkel rendelkezőt.

C2: Hansen intervallum-kiválasztási szabálya: válasszuk $\mathcal{L}_{\mathcal{W}}$ elemei közül a listába legkorábban beillesztett boxot.

A $p(f_k, \boldsymbol{X})$ mutatót használva az alábbi új szabályok konstruálhatók:

C3: [11] alapján a Munkalista elemei közül a maximális $p(f_k, \boldsymbol{X})$ értékkel rendelkező box legyen kiválasztva.

C4: [6] szerzői egy hibrid szabály javasoltak: Legyen N_m egy rögzített pozitív konstans, és minden iterációban válasszuk ki $\mathcal{L}_{\mathcal{W}}$ elemei közül a legnagyobb $p(f_k, \boldsymbol{X})$ értékkel rendelkező N_m darab boxot. (Ha $\mathcal{L}_{\mathcal{W}}$ hossza kisebb, mint N_m, akkor válasszuk ki annak összes elemét.) A kiválasztott boxok alkotják az aktualizált Munkalistát, míg a többi box egy másodlagos listára kerül, és az algoritmus végén, egy második fázisban lesz feldolgozva a hagyományos **C1** szabály alkalmazásával. A másodlagos lista mérete a kivágási teszt jóvoltából természetesen már az első fázisban is redukálható.

Csendes [11] a C3 szabályt részletesen vizsgálja $(*)$ alakú feladatokon. A legfontosabb elméleti következtetések az alábbi tételben foglalhatók össze:

16. TÉTEL. *Csendes [11] alapján: Jelölje F a f célfüggvény befoglaló függvényét. Tekintsük azt az intervallumos B&B algoritmust, mely a C3 stratégiát használja intervallum-felosztási szabályként.*

- *Tegyük fel, hogy a célfüggvény izoton és zéró-konvergens. Ha $f_k \to \hat{f} > f^*$, és létezik olyan $\boldsymbol{x} \in \boldsymbol{X}_0$ pont, melyre $f(\boldsymbol{x}) = \hat{f}$, akkor a gyorsítótesztek nélküli algoritmusváltozat egy nemoptimális megoldáshoz konvergálhat.*

- *Tegyük fel, hogy a célfüggvény zéró-konvergens. Ha $f_k \to \hat{f} < f^*$, akkor a gyorsítótesztek nélküli algoritmus az $\boldsymbol{X}_0$ keresési tartomány összes pontjához konvergál.*

- *Tegyük fel, hogy a célfüggvény izoton és zéró-konvergens, továbbá az algoritmus a kivágási, monotonitási, intervallumos Newton-, illetve konkavitási teszteket használja gyorsító eljárásokként. Annak szükséges és elegendő feltétele, hogy az algoritmus globális minimumpontok egy halmazához konvergál, az, hogy az $\{f_k\}$ sorozat a globális minimumhoz, f^*-hoz tart, valamint $\{f_k\}$ véges sok eleme kisebb, mint f^*.*

Fentiekre és további [11]-beli eredményekre a korlátozásos feladatokra javasolt algoritmus-változatok vizsgálatában többször hivatkozom.

A kiválasztott box felosztását illetően (1. Algoritmus 4. és 5. lépése), amint arra a Bevezetésben rámutattam, a legelterjedtebb megközelítés a box két részre osztása (bisection) annak legszélesebb komponensére merőlegesen. Korlátozás nélküli problémák esetén azonban sokszor nem ez a leghatékonyabb megközelítés. A felosztási irány megadására Csendes és Ratz [14, 15, 54, 55] által, az s érték megválasztására pedig Berner, Ratz, Csallner, Csendes és Markót [1, 9, 39, 55] által ismertetett új, eredményesebb stratégiák leírásával és diszkussziójával az értekezés Bevezetésében és a 2. Fejezetben részletesen foglalkoztam.

A korlátozásos feladatok jelenlegi vizsgálatában az irányválasztási szabályok közül a hagyományos, a box legszélesebb oldalára (oldalaira) vonatkozó szabályt ('A'-stratégia) alkalmazom. Az s értékek és a különböző számú felosztási irány alapján a kiindulásul tekintett intervallum-felosztási módszerek az alábbiak:

S1_/2: klasszikus bisection a legszélesebb komponensre merőlegesen;

S2_/4: felezés a két legszélesebb komponensre merőlegesen, azaz 4 részboxra osztás (/4 multisection);

S3_/9: három egyforma részre darabolás a két legszélesebb komponensre merőlegesen, tehát 9 részboxra történő felosztás (/9 multisection).

Az S1-S3 szabályok mindegyike ún. statikus szabály, azaz az algoritmus futása alatt a felosztás szabálya változatlan marad. Ahogy azonban korábban arra már utaltam, számos esetben ígéretes lehet az, ha a keresési tér egyes részein (tipikusan a minimumhelyektől távol) 'óvatosabb' felosztási szabályt alkalmazunk kevesebb részre osztással. Ekkor ugyanis abban bízhatunk, hogy a keletkező nagyobb részboxokat is eliminálni tudjuk az eredeti box szükségtelenül sok részre osztása nélkül. Másrészt, a minimumhelyekhez közelebb érdemes lehet több részre osztani, hogy minél előbb juthassunk olyan boxokhoz, amiket az Eredménylistára tehetünk.

Az ismertetett $p(\hat{f}, \boldsymbol{X})$ mutató lehetőséget biztosít arra, hogy heurisztikusan jellemezhessük azt, melyik esetben milyen felosztást érdemes alkalmazni. A Casado és munkatársai [6] által ismertetett *adaptív intervallum-felosztási* szabály alkalmas $0 \leq P_1 < P_2 \leq 1$ küszöbértékek megadásával a következő alakot ölti:

S4_pf_/2,4,9:

(a) ha $p(f_k, \boldsymbol{X}) < P_1$, akkor $\boldsymbol{X}$-et vágjuk két részre a legszélesebb komponens szerint (klasszikus bisection);

(b) ha $P_1 \leq p(f_k, \boldsymbol{X}) \leq P_2$, akkor $\boldsymbol{X}$-et vágjuk négy részre a két legszélesebb komponensre merőlegesen felezve (/4 multisection);

(c) ha $p(f_k, \boldsymbol{X}) > P_2$, akkor $\boldsymbol{X}$-et vágjuk kilenc részre a két legszélesebb komponensre merőlegesen harmadolva (/9 multisection).

3.2 Az új heurisztikus mutató

A következőkben a J. Fernández Hernández által javasolt heurisztikus mennyiséget bemutatom be. Ez képezi az alapját a korlátozásos feladatok esetén használatos új módszereknek. A mutató becslést ad arra, hogy a tekintett box milyen mértékben tartalmaz lehetséges megoldásokat, ily módon a korlátok elhelyezkedéséből adódó többletinformációt is kihasználhatjuk az algoritmusokban.

Kiindulásul tekintsünk egy korlátozó feltételt, $g_j(x) \leq 0$-t. Számítsuk ki a

$$pu_{G_j}(\boldsymbol{X}) = \min\left\{\frac{-\underline{G_j}(\boldsymbol{X})}{w(G_j(\boldsymbol{X}))}, 1\right\} \tag{9}$$

mennyiséget, ahol $G(\boldsymbol{X})$ a $g(\boldsymbol{x})$ függvény befoglaló függvénye. A továbbiakban $G(\boldsymbol{X})$-t a korlátozó feltétel (vagy korlát) befoglaló függvényének nevezem. (Ez az elnevezés némiképp pontatlan ugyan, de a feltételek általános alakja miatt nem

félrevezető.) A $w(G_j(\boldsymbol{X})) = 0$ eset kezelésére a 8. Megjegyzésben F-re leírtak igazak az alábbi kiegészítéssel: ha a nevező 0 értéket venne fel, akkor $\boldsymbol{X}$-en a g_j függvény konstans. Ekkor, ha a $g_j(\boldsymbol{X}) \in \mathbb{R}$ értékkészlet pozitív, akkor $\boldsymbol{X}$ törölhető a lehetséges megoldások tesztjével, ellenkező esetben legyen $pu_{G_j}(\boldsymbol{X}) = 1$.

Vegyük észre, hogy ha $pu_{G_j}(\boldsymbol{X}) < 0$, akkor $\boldsymbol{X}$ biztosan nem tartalmaz egyetlen lehetséges megoldást sem, amennyiben pedig $pu_{G_j}(\boldsymbol{X}) = 1$, akkor $\boldsymbol{X}$ minden pontja lehetséges megoldás. (Az implementáció során az első esetben a $\underline{G_j}(\boldsymbol{X}) > 0$, míg a másodikban a $-\underline{G_j}(\boldsymbol{X}) = w(G_j(\boldsymbol{X}))$ feltételt kell vizsgálnunk, hogy a (9)-beli nem intervallumos típusú osztás esetleges pontatlanságait elkerülve megbízható következtetésekre jussunk!)

Tekintsük most a $(**)$ lehetséges megoldásainak halmazát leíró korlátok halmazát: $g_1(x) \leq 0, \ldots, g_r(x) \leq 0$. Azon boxok, melyekről megállapítható, hogy biztosan nem tartalmaznak egyetlen lehetséges megoldást sem, azonnal törölhetők, így ezekre nem definiáljuk a lenti indexet. A többi boxra legyen a lehetséges megoldások tartalmazását leíró heurisztikus index a

$$pu(\boldsymbol{X}) = \prod_{j=1}^{r} pu_{G_j}(\boldsymbol{X}).$$

formulával adott. A definíció alapján könnyen ellenőrizhető, hogy

- $pu(\boldsymbol{X}) = 1 \Leftrightarrow \boldsymbol{X}$ kizárólag lehetséges megoldásokat tartalmaz, valamint
- $pu(\boldsymbol{X}) \in [0, 1) \Leftrightarrow \boldsymbol{X}$ meghatározatlan a lehetséges megoldások tartalmazására nézve.

A $pu(\boldsymbol{X})$ index használatával a $p(\hat{f}, \boldsymbol{X})$ mennyiség alábbi módosítása javasolható:

$$pup(\hat{f}, \boldsymbol{X}) = pu(\boldsymbol{X}) \cdot p(\hat{f}, \boldsymbol{X}),$$

ahol $\hat{f}$-t ismét valamely f_k értékek sorozata által tekintjük adottnak. Az új mutató tehát úgy működik, hogy a korábbi $p(\hat{f}, \boldsymbol{X})$ értéket korrigálja a lehetséges megoldások tartalmazására vonatkozó faktorral: egy box minél nagyobb részéről sejtjük, hogy nem tartalmaz lehetséges megoldásokat, a hozzá tartozó $pup(\hat{f}, \boldsymbol{X})$ érték annál kisebb lesz. Fentiek alapján a felosztásra kerülő box kiválasztására a következő szabályok konstruálhatók:

C5: Válasszuk felosztásra a legnagyobb $pup(f_k, \boldsymbol{X})$ értékkel rendelkező boxot.

C6: Alkalmazzunk egy a C4-hez hasonló hibrid szabályt, de ezúttal $p(f_k, \boldsymbol{X})$ helyett $pup(f_k, \boldsymbol{X})$-t használva.

Végezetül, a Moore–Skelboe-szabály és az új szabály kombinációjaként választhatjuk felosztásra az egyszerre alacsony $\underline{F}(\boldsymbol{X})$ értékkel és magas $pup(f_k, \boldsymbol{X})$ értékkel rendelkező boxot. Ehhez definiáljuk a

$$pupb(f_k, \boldsymbol{X}) = \begin{cases} \underline{F}(\boldsymbol{X})/pup(f_k, \boldsymbol{X}) & \text{ha } pup(f_k, \boldsymbol{X}) \neq 0, \\ M & \text{egyébként} \end{cases}$$

mennyiséget, ahol M egy előre beállított nagy pozitív szám.

C7: Válasszuk a *legkisebb* $pupb(f_k, \boldsymbol{X})$-vel rendelkező boxot felosztásra.

C8: Használjuk a C4 hibrid szabályt, de ezúttal a *minimális* $pupb(f_k, \boldsymbol{X})$ értékkel rendelkező boxot választva.

A $pup(\hat{f}, \boldsymbol{X})$ index alkalmas új típusú adaptív multisection szabályok előállítására is. Ez azon a feltételezésen alapul, hogy minél kisebb mértékben tartalmaz egy box lehetséges megoldásokat, annál kevésbé lehet szükség arra, hogy nagyobb s értéket válasszunk, azaz több részre osszuk a boxot (bízva abban, hogy a box lehetséges megoldásokat nem tartalmazó részintervallumai így kevesebb munka árán lesznek eldobhatók). A [6]-beli, esetünkben S4-gyel jelölt szabály alapján az ottani $p(f_k, \boldsymbol{X})$ érték helyett $pup(\hat{f}, \boldsymbol{X})$-t tekintve, alkalmas $0 \leq P_1 < P_2 \leq 1$ küszöbértékek megadásával a következő intervallum-felosztási szabály javasolható:

S5_pup_/2,4,9:

(a) ha $pup(f_k, \boldsymbol{X}) < P_1$, akkor $\boldsymbol{X}$-et vágjuk két részre a legszélesebb komponens szerint (klasszikus bisection);

(b) ha $P_1 \leq pup(f_k, \boldsymbol{X}) \leq P_2$, akkor $\boldsymbol{X}$-et vágjuk négy részre a két legszélesebb komponensre merőlegesen felezve (/4 multisection);

(c) ha $pup(f_k, \boldsymbol{X}) > P_2$, akkor $\boldsymbol{X}$-et vágjuk kilenc részre a két legszélesebb komponensre merőlegesen harmadolva (/9 multisection).

Az előző szakaszban rámutattam, hogy $p(\tilde{f}, \boldsymbol{X})$ bizonyos boxok esetén 1-nél nagyobb értéket is felvehet. Ez az intervallum-kiválasztási szabályok használatakor a kívántnál korábbi kiválasztást, az adaptív multisection szabálynál pedig a kelleténél több részre vágást eredményezhet. A problémát elkerülendő, minden olyan boxra, melyre egyébként $p(\tilde{f}, \boldsymbol{X}) > 1$ teljesül, a $p(\tilde{f}, \boldsymbol{X}) := 1$ módosítást alkalmazom. (Azaz az új index ez esetben a $pup(\tilde{f}, \boldsymbol{X}) = pu(\boldsymbol{X})$ formulával lesz adott.)

3.3 Konvergencia vizsgálatok

3.3.1 Az új típusú intervallum-kiválasztási szabályok konvergencia-tulajdonságai

A tradicionálisan használt C1, illetve C2 szabályok konvergenciáját [8, 13, 51]-ben vizsgálták, míg a C3 tulajdonságainak vizsgálata [11]-ben található meg. Alábbiakban a C5 és C7 szabályokat használó algoritmusok konvergencia-jellemzőit mutatom be [40] alapján. A bizonyítások alapjául több helyütt Csendes [11] módszereit alkalmaztam és fejlesztettem tovább a korlátozásos esetre, illetve az új heurisztikákra. Az algoritmusokról a konvergencia-vizsgálat során feltételezzük, hogy a megállási feltételeknek egy box sem tehet eleget. Az egyes módszereket gyorsítótesztek nélkül vizsgálom, kivéve azokat az eseteket, ahol az alkalmazott tesztek használatát külön kiemelem, illetve a lehetséges megoldások tesztjét, amelyet minden esetben alkalmazok.

Jelöljük f_k-val az f^* közelítését az algoritmus k-adik iterációjában, ekkor tehát egy vizsgált $\boldsymbol{X}$ boxhoz a $p(f_k, \boldsymbol{X})$ paramétert rendelhetjük hozzá. Jegyezzük meg, hogy az f_k értékek megadási módjára semmilyen kikötést nem teszünk, azok számíthatók akár az algoritmus működése során (pl. az $f_k := \tilde{f}$ esetben), vagy lehetnek a felhasználó által az algortimuson kívülről adottak.

Az elsőként ismertetett tétel megadja, miképp viselkedik a C5 szabályt használó algoritmus, ha az $\{f_k\}$ sorozat a globális minimumnál nagyobb számhoz konvergál:

17. TÉTEL. *[40] Tegyük fel, hogy a célfüggvény befoglaló függvénye izoton, zéró-konvergens, a korlátok befoglaló függvényei ugyancsak izoton tulajdonságúak, a $p(f_k, \boldsymbol{Z})$ paraméterek pedig egy $\{f_k\}_{k=1}^{\infty}, f_k \to \hat{f} > f^*$ sorozat segítségével adottak, ahol $\hat{f} = f(\hat{\boldsymbol{x}})$ valamely $\hat{\boldsymbol{x}} \in \boldsymbol{X}_0$ lehetséges megoldás esetén. Akkor az intervallumos B&B algoritmus, mely minden lépésben a legnagyobb $pup(f_k, \boldsymbol{Z})$ értékkel rendelkező intervallumot választja ki felosztásra (azaz a C5 szabályt használja)* konvergálhat *egy olyan $\hat{\boldsymbol{x}} \in \boldsymbol{X}_0$ lehetséges megoldáshoz, melyre $f(\hat{\boldsymbol{x}}) > f^*$ áll, azaz, amely nem globális minimumhelye a megoldandó feladatnak.*

BIZONYÍTÁS. [11] 1. Tétele hasonló állítást bizonyít korlátozó feltételek nélküli feladatok, valamint a C3 kiválasztási szabály esetére. Az említett tétel a célfüggvény befoglaló függvényének egy alkalmas konstrukcióján alapul: ehhez definiáljuk a $d(\boldsymbol{X}) = \max(\overline{f}(\boldsymbol{X}) - \hat{f}, \hat{f} - \underline{f}(\boldsymbol{X}))$ mennyiséget minden $\boldsymbol{X} \subseteq \boldsymbol{X}_0$ boxra. Legyen a kiinduló boxra a célfüggvény befoglalása az $F(\boldsymbol{X}_0) = [\hat{f} - 9d(\boldsymbol{X}_0), \hat{f} + d(\boldsymbol{X}_0)]$ formulával adott, valamint minden olyan $\boldsymbol{X}_i$ boxra, mely tartalmazza $\hat{\boldsymbol{x}}$-et, legyen $F(\boldsymbol{X}_i) = [\hat{f} - 9d(\boldsymbol{X}_i), \hat{f} + d(\boldsymbol{X}_i)]$. Minden más $\boldsymbol{Y}$ boxra a befoglalás az $F(\boldsymbol{Y}) = [\hat{f} - d(\boldsymbol{Y}), \hat{f} + d(\boldsymbol{Y})]$ formulával legyen adott. Az így definiált függvény teljesíti az intervallumos befoglaló függvény tulajdonságát, izoton, és (minden $\hat{\boldsymbol{x}}$-ot tartalmazó végtelen boxsorozat esetén) zéró-konvergens. Továbbá, mivel f_k az $\hat{f}$-hoz tart, ezért egy bizonyos iterációs lépésszám elérése után a Munkalistán lévő, $\hat{\boldsymbol{x}}$-et tartalmazó $\boldsymbol{X}$ boxra a $p(f_k, \boldsymbol{X})$ érték elegendően közel lesz 0.9-hez, valamint a Munkalistán lévő összes többi $\boldsymbol{Y}$ boxra a $p(f_k, \boldsymbol{X})$ értékek elegendően közel lesznek 0.5-höz ahhoz, hogy a C3 szabályt alkalmazó algoritmus $\boldsymbol{X}$-et válassza ki felosztásra. Azaz, egy megfelelő iterációs lépéstől kezdve az algoritmus minden alkalommal a $\hat{\boldsymbol{x}}$-et tartalmazó boxot választja ki, vagyis az algoritmus $\hat{\boldsymbol{x}}$-hoz konvergál.

Korlátozásos feladatokra és a C5 szabályt alkalmazó algoritmusra a fenti konstrukció egyszerűen kiterjeszthető: alkalmazzuk a célfüggvény befoglalásának megadására a fenti összefüggéseket, és definiáljuk a korlátozó feltételeket úgy, hogy az összes $\boldsymbol{X}_0$-beli pont szigorúan lehetséges megoldás legyen. Ezután képezzük a korlátok befoglaló függvényeit a *természetes intervallum-kiterjesztés*sel (1.3 szakasz), így azok a 1.4 szakaszban leírtak szerint izoton tulajdonságúak lesznek, és tegyük fel, hogy $pu(\boldsymbol{X}_0) = 1$. Ez utóbbi feltétel az esetleges intervallumos túlbecslés ellenére is biztosítható. Ekkor $pup(f_k, \boldsymbol{Z}) = p(f_k, \boldsymbol{Z})$ teljesül minden $\boldsymbol{Z} \subseteq \boldsymbol{X}_0$ boxra, hiszen a korlátok befoglaló függvényeinek izotonitásából $pu(\boldsymbol{Z}) = 1$ következik. Azaz, a kiválasztott részboxok sorozata megegyezik azzal, amit a C3-at használó algoritmus generálna. □

A következő tételben megmutatom, hogy az algoritmus akkor is konvergálhat nemoptimális pontokhoz, ha a korlátok befoglaló függvényei α-konvergensek:

18. TÉTEL. *[40] Tegyük fel, hogy a célfüggvény befoglaló függvénye izoton, zérókonvergens, a korlátok befoglaló függvényei α-konvergensek (és akár nem is izoton tulajdonságúak) a $p(f_k, \boldsymbol{Z})$ paraméterek pedig egy $\{f_k\}_{k=1}^{\infty}, f_k \to \hat{f} > f^*$ sorozat segítségével adottak, ahol $\hat{f} = f(\hat{\boldsymbol{x}})$ valamely $\hat{\boldsymbol{x}} \in \boldsymbol{X}_0$ lehetséges megoldás esetén. Akkor az intervallumos B&B algoritmus, mely minden lépésben a legnagyobb $pup(f_k, \boldsymbol{Z})$ értékkel rendelkező intervallumot választja ki felosztásra (azaz a C5 szabályt használja)* konvergálhat *egy olyan $\hat{\boldsymbol{x}} \in \boldsymbol{X}_0$ lehetséges megoldáshoz, melyre $f(\hat{\boldsymbol{x}}) > f^*$ áll, azaz, amely nem globális minimumhelye a megoldandó feladatnak.*

BIZONYÍTÁS. A bizonyítás az előző tételhez hasonlóan most is egy alkalmas konstrukción alapul. Az egyszerűség kedvéért tekintsünk egy olyan korlátozásos globális optimalizálási feladatot, amely mindössze egy korlátot, a $g(\boldsymbol{x}) \leq 0$ feltételt tartalmazza. Tegyük fel, hogy $g(\boldsymbol{x})$ racionális és a teljes $\boldsymbol{X}_0$-on értelmezett. Állítsuk elő a célfüggvény F befoglaló függvényét az előző tételben megadottak szerint, a korlátozó feltétel befoglaló függvénye pedig legyen az alábbi eljárás által meghatározott: képezzük először a G befoglaló függvényt természetes intervallum-kiterjesztéssel. A kapott függvény izoton és α-konvergens $\alpha = 1$-gyel (ld. 3. Tétel), azaz $w(G(\boldsymbol{X})) - w(g(\boldsymbol{X})) \leq cw(\boldsymbol{X})$ egy alkalmas c pozitív konstanssal. $G(\boldsymbol{X})$-ből előállítjuk a korlát egy olyan befoglaló függvényét (K-t) minden $\boldsymbol{X} \subseteq \boldsymbol{X}_0$ boxra, amely megfelelő lesz a tétel bizonyításához:

(1) Ha $\underline{G}(\boldsymbol{X}) > 0$, akkor legyen $K(\boldsymbol{X}) = G(\boldsymbol{X})$. (Amennyiben az algoritmus futása során előáll egy ilyen tulajdonságú $\boldsymbol{X}$ box, az azonnal törölve lesz a lehetséges megoldások tesztjével.)

(2) Ha $\underline{G}(\boldsymbol{X}) \leq 0$ és $\overline{G}(\boldsymbol{X}) > 0$, akkor $G(\boldsymbol{X})$ legnagyobb abszolút értéke (1.2 szakasz) alapján legyen

$K(\boldsymbol{X}) = [-9|G(\boldsymbol{X})|, |G(\boldsymbol{X})|]$, amennyiben $\hat{\boldsymbol{x}} \in \boldsymbol{X}$ (azaz ekkor $pu(\boldsymbol{X}) = 0.9$ teljesül), illetve

$K(\boldsymbol{X}) = [-|G(\boldsymbol{X})|, |G(\boldsymbol{X})|]$, ha $\boldsymbol{X}$ nem tartalmazza $\hat{\boldsymbol{x}}$-t (ekkor pedig $pu(\boldsymbol{X}) = 0.5$ teljesül).

(3) Ha $\overline{G}(\boldsymbol{X}) \leq 0$, akkor legyen a befoglalás a $K(\boldsymbol{X}) = G(\boldsymbol{X})$ módon adott (nyilván minden ilyen $\boldsymbol{X}$-re $pu(\boldsymbol{X}) = 1$).

Mindhárom ismertetett esetben a definiált befoglalásra $G(\boldsymbol{X}) \subseteq K(\boldsymbol{X})$, azaz K valóban befoglaló függvénye a korlátozó feltételnek. Továbbá, K α-konvergens ($\alpha = 1$-gyel) az alábbiak miatt: mivel a korlátot racionálisnak és a teljes $\boldsymbol{X}_0$-on értelmezettnek választottuk, ezért a természetes intervallum-kiterjesztés definíciójának egyszerű következményeként (lásd pl. [50]) G Lipschitz-folytonos egy m pozitív konstanssal: $w(G(\boldsymbol{X})) \leq mw(\boldsymbol{X}), \forall \boldsymbol{X} \subseteq \boldsymbol{X}_0, \boldsymbol{X} \in \mathbb{I}^n$.

Mivel minden $\boldsymbol{X} \subseteq \boldsymbol{X}_0$-ra $w(K(\boldsymbol{X})) \leq 10w(G(\boldsymbol{X}))$ teljesül, így G α-konvergenciája miatt

$$w(K(\boldsymbol{X})) - w(g(\boldsymbol{X})) \leq w(G(\boldsymbol{X})) - w(g(\boldsymbol{X})) + 9w(G(\boldsymbol{X})) \leq$$

$$\leq cw(\boldsymbol{X}) + 9w(G(\boldsymbol{X})) \leq (c + 9m)w(\boldsymbol{X}),$$

azaz, K α-konvergens $\alpha = 1$-gyel.

Jegyezzük meg, hogy a K függvény egyszerűen módosítható úgy, hogy ne legyen izoton tulajdonságú: kiválasztva véges sok $\boldsymbol{X} \subseteq \boldsymbol{X}_0$-t, melyre K definíciójának (2) pontja teljesül, majd az ezekre adott befoglaló intervallumokat alkalmas számokkal megszorozva elérhető a módosított befoglaló függvény izotonitásának sérülése. A fenti módosítás ugyanakkor nem lesz hatással a K α-konvergenciájára: a $w(K(\boldsymbol{X})) - w(g(\boldsymbol{X}))$ kifejezés tudniillik csak a kiválasztott véges sok boxra változik; ezeket újra kiértékelve az α-konvergenciára imént kapott konstans könnyen módosítható.

A bizonyítás hátralévő része az előző tételben megfogalmazottakhoz hasonlóan történik: a célfüggvényre és a korlátozó feltételre definiált befoglaló függvényeket használva azt állíthatjuk, hogy egy megfelelő iterációs szám elérésétől kezdve a $\hat{\boldsymbol{x}}$-t tartalmazó $\boldsymbol{X}$ boxon a $pup(f_k, \boldsymbol{X}) = p(f_k, \boldsymbol{X}) \cdot pu(\boldsymbol{X})$ érték elegendően közel lesz $0.9 \cdot 0.9 = 0.81$-hoz, *vagy* $0.9 \cdot 1 = 0.9$-hez ahhoz, hogy az algoritmus az $\boldsymbol{X}$-t válassza ki felosztásra szemben a Munkalistán lévő összes többi $\boldsymbol{Y}$ box-szal, melyekhez $0.5 \cdot 0.5 = 0.25$-hoz, vagy $0.5 \cdot 1 = 0.5$-hez közeli $pup(f_k, \boldsymbol{Y})$ értékek tartoznak. Azaz az algoritmus $\hat{\boldsymbol{x}}$-hez fog konvergálni. □

9. MEGJEGYZÉS. *A kivágási teszt és a célfüggvény befoglaló függvényének zéró-konvergenciája elegendő lehet az algoritmus globális minimumpontokhoz történő konvergenciájához még abban az esetben is, amikor f_k nem tart f^*-hoz: a felosztásra kijelölt boxok tetszőleges, egy f^*-nál nagyobb értékhez tartó $\boldsymbol{X}_0, \boldsymbol{X}_1, \ldots$ részsorozata esetén a zéró-konvergenciából $\underline{F}(\boldsymbol{X}_j) > f^*$ adódik valamely j-re, azaz, ha $\tilde{f}$ oly módon közelíti f^*-ot, hogy $\underline{F}(\boldsymbol{X}_j) > \tilde{f}$ ugyancsak fennáll, akkor az ilyen boxok a kivágási teszttel törölhetők.*

Az alábbi lemma karakterizációját adja egymásba ágyazott boxok egy olyan sorozatának (azaz a felosztásra kijelölt boxok valamely részsorozatának), amely speciális esetként jelentkezik a konvergencia-vizsgálatokban.

1. LEMMA. *[40] Tételezzük fel, hogy a korlátok befoglaló függvényei izoton tulajdonságúak. Legyen $\{\boldsymbol{X}_k\}_{k=1}^{\infty}$ az algoritmus által generált egymásba ágyazott boxok egy olyan sorozata, mely az $\boldsymbol{x}$ lehetséges megoldáshoz konvergál. Akkor az alábbi három állítás ekvivalens:*

1. *Egy alkalmas i indexre $pu(\boldsymbol{X}_i) = 0$.*

2. *Egy alkalmas i indexre $pu(\boldsymbol{X}_k) = 0$ $\forall k \geq i$.*

3. *Egy alkalmas i index és $\forall k \geq i$ esetén $\boldsymbol{X}_k$ nem tartalmaz szigorúan lehetséges megoldásokat, továbbá létezik olyan s indexű korlátozó feltétel, melyre $g_s(\boldsymbol{x}) = 0$, valamint a G_s befoglaló függvény az alsó korlátjában pontos minden $\boldsymbol{X}_k$ esetén.*

BIZONYÍTÁS. 1. $\Rightarrow$ 3. : Amennyiben a lemma 1. állítása teljesül, akkor $pu(\boldsymbol{X})$ definíciója miatt $\underline{G_s}(\boldsymbol{X}_i) = 0$ teljesül valamely s $(1 \leq s \leq r)$ index esetén. Mivel az $\{\boldsymbol{X}_k\}$ sorozat összes eleme tartalmaz legalább egy lehetséges megoldást ($\boldsymbol{x}$-t), valamint G_s izoton, ezért szükségképpen $\underline{G_s}(\boldsymbol{X}_k) = 0$ és így $G_s(\boldsymbol{X}_k) \geq 0$ teljesül minden $k \geq i$ esetén is. Ez épp azt jelenti, hogy az i-edik iterációtól kezdve a boxsorozat elemei nem tartalmaznak szigorúan lehetséges megoldásokat. Másrészt, azon boxokra, melyek tartalmaznak legalább egy lehetséges megoldást, de nem tartalmaznak egyetlen szigorúan lehetséges megoldást sem, $\underline{g_s}(\boldsymbol{X}_k) = 0$ is teljesül minden $k \geq i$-re. Ebből az alapvető tartalmazási tulajdonságok miatt $g_s(\boldsymbol{x}) = 0$ következik, valamint az, hogy G_s minden k esetén pontos (túlbecsléstől mentes) az alsó korlátjában.

3. $\Rightarrow$ 2. : Tegyük fel, hogy a lemma 3. állítása teljesül. Ha G_s-nek nincs túlbecslése az alsó korlátban olyan boxokon, melyek nem tartalmaznak szigorúan lehetséges megoldást, de tartalmazzák $\boldsymbol{x}$-t, akkor $\underline{G_s}(\boldsymbol{X}_k) = 0$, és ezért $pu(\boldsymbol{X}_k) = 0$ áll $\forall k \geq i$ esetén.

2. $\Rightarrow$ 1. : Ez az implikáció nyilvánvalóan teljesül. □

A további tételek pontos megfogalmazásához az alábbi fogalmakat és jelöléseket használjuk: Az algoritmus által felosztásra kiválasztott boxok konvergens $\{\boldsymbol{X}_l\}_{l=1}^{\infty}$ részsorozataival foglalkozunk. (Az 1.6.1 szakasz értelmében ezen részsorozatokról az általánosság megszorítása nélkül feltehető, hogy egymásba ágyazott boxokból állnak. Ezt a tulajdonságot bizonyos esetekben felhasználjuk.) Egy ilyen részsorozatról feltételezhetjük, hogy az elemei rendre a $k_1, k_2, \ldots$ iterációs lépésekben lettek felosztásra kiválasztva, azaz a hozzájuk tartozó $p(f_k, \boldsymbol{Y})$ és $pup(f_k, \boldsymbol{Y})$ értékek a $p(f_{k_l}, \boldsymbol{X}_l)$, illetve $pup(f_{k_l}, \boldsymbol{X}_l)$ formulák által meghatározottak $l = 1, 2, \ldots$ esetén.

A következő tétel azt az esetet vizsgálja, amikor az $\{f_k\}$ sorozat a globális minimumnál kisebb értékhez tart:

19. TÉTEL. *[40] Tegyük fel, hogy a célfüggvény és a korlátok befoglaló függvényei zéró-konvergensek, továbbá nem létezik olyan* $\{\boldsymbol{X}_k\}_{k=1}^{\infty}$, $\boldsymbol{X}_k \subseteq \boldsymbol{X}_0$ *boxsorozat, melynek valamely konvergens* $\{\boldsymbol{X}_{k_l}\}$ *részsorozatára* $\lim_{l\to\infty} pu(\boldsymbol{X}_{k_l}) = 0$ *teljesül, valamint az* f_k *értékek sorozata egy* $\hat{f} < f^*$ *számhoz konvergál. Akkor az intervallumos B&B algoritmus, mely minden lépésben a legnagyobb* $pup(f_k, \boldsymbol{Z})$ *értékkel rendelkező* $\boldsymbol{Z}$ *intervallumot választja ki felosztásra, a maximális* $pup(f_k, \boldsymbol{Z})$*-vel rendelkező boxok közül pedig a maximális* $p(f_k, \boldsymbol{Z})$ *értékűt választja, a felosztásra kijelölt boxok olyan részsorozatait állítja elő, melyek az* $\boldsymbol{X}_0$ *keresési tartomány minden lehetséges megoldásához konvergálnak, függetlenül az azokban számított célfüggvényértékektől.*

BIZONYÍTÁS. Először [11] alapján megmutatjuk, hogy az algoritmus által felosztásra kiválasztott boxok tetszőleges, valamely $\boldsymbol{x} \in \boldsymbol{X}_0$ ponthoz tartó $\{\boldsymbol{X}_l\}$ részsorozatára a $\{p(f_{k_l}, \boldsymbol{X}_l)\}$ sorozat mínusz végtelenhez tart. F zéró-konvergenciája, valamint $f_k \to \hat{f} < f^*$ és $f^* \leq f(\boldsymbol{x})$ miatt $f_{k_l} - \underline{F}(\boldsymbol{X}_l) \to \tilde{f} - f(\boldsymbol{x}) < 0$. Másrészt, ugyancsak F zéró-konvergenciája miatt $w(F(\boldsymbol{X}_l)) \to 0$, azaz $p(f_{k_l}, \boldsymbol{X}_l) \to -\infty$.

Vizsgáljuk most a $pup(f_{k_l}, \boldsymbol{X}_l)$ értékeket a fenti boxsorozatra. Azt állítjuk, hogy $\lim_{l\to\infty} pup(f_{k_l}, \boldsymbol{X}_l) = -\infty$. Ennek igazolásához tekintsük az alábbi két esetet:

1. Ha $\{\boldsymbol{X}_l\}$ a szigorúan lehetséges $\boldsymbol{x}$ megoldáshoz tart, akkor $-\underline{G_j}(\boldsymbol{X}_l) \to -g_j(\boldsymbol{x}) > 0$ minden j-re ($j = 1, \ldots, r$) a G_j függvények zéró-konvergenciája miatt, amiből $pu_{G_j}(\boldsymbol{X}_l) \to 1, \forall j$, és így $lim_{l\to\infty} pu(\boldsymbol{X}_l) = 1$ adódik. Következésképpen $\lim_{l\to\infty} pup(f_{k_l}, \boldsymbol{X}_l) = \lim_{l\to\infty} p(f_{k_l}, \boldsymbol{X}_l) = -\infty$.

2. Tegyük most fel, hogy $\{\boldsymbol{X}_l\}$ egy olyan lehetséges megoldáshoz tart, melyre az előző pont nem teljesül, azaz létezik a korlátok indexeinek egy olyan nemüres J halmaza, melyre $g_j(x) = 0$, $\forall j \in J$. Mivel feltevésünk szerint $\lim_{l\to\infty} pu(\boldsymbol{X}_l) \neq 0$, ezért $pup(f_{k_l}, \boldsymbol{X}_l)$ definíciója és $0 \leq pu(\boldsymbol{X}_l) \leq 1$ miatt $\lim_{l\to\infty} pup(f_{k_l}, \boldsymbol{X}_l) = -\infty$.

(Jegyezzük meg, hogy az imént tárgyalt 1. illetve 2. pont minden számunkra releváns lehetőséget lefed: az $\{\boldsymbol{X}_l\}$ részsorozat ugyanis nem tarthat egy nem lehetséges megoldáshoz a korlátok zéró-konvergenciája, illetve a lehetséges megoldások tesztjének alkalmazása miatt.)

A fentiek következményeként minden, a Munkalistán tartózkodó tetszőleges box valamely későbbi iterációs lépésben biztosan felosztásra kerül, azaz az algoritmus a keresési tartomány összes lehetséges megoldásához tart. Ezzel igazoltuk a tétel állítását. □

A 19. Tétel következményei az alábbiakban foglalhatók össze: ha az $\{f_k\}$ sorozat egy, az optimumnál kisebb értékhez tart, akkor az algoritmus (a) vagy minden lehetséges megoldáshoz generál az adott ponthoz tartó intervallum részsorozatot, vagy pedig (b) néhány olyan konvergens $\boldsymbol{X}_l$ részsorozatot állít elő, melyek mindegyike egy-egy nem szigorúan lehetséges megoldáshoz tart: az (a) esetben a $pup(f_{k_l}, \boldsymbol{X}_l)$ értékek mínusz végtelenhez tartanak minden konvergens $\{\boldsymbol{X}_l\}$ esetén, míg a (b) esetben létezik a felosztásra kijelölt intervallumoknak néhány olyan $\{\boldsymbol{X}_l\}$ részsorozata, melyre $\lim_{l\to\infty} pu(\boldsymbol{X}_{k_l}) = 0$, valamint $\lim_{l\to\infty} pup(f_{k_l}, \boldsymbol{X}_l)$ véges. Ez utóbbi eset jellemzi például – izoton és zéró-konvergens G_j-k esetén – az olyan boxsorozatokat is, melyekre az 1. Lemma valamely állítása teljesül (hiszen ekkor $pup(f_{k_l}, \boldsymbol{X}_l) = pu(\boldsymbol{X}_l) = 0$ véges sok lépés eltelte után).

Meglepő következményei vannak annak, hogy az $f_k \to \hat{f} < f^*$ esetben az algoritmus konvergenciájára a fenti (a), illetve (b) pont is teljesülhet. Az 1. Lemma segítségével például olyan feladatok konstruálhatók, melyek esetén a keresési tartomány, vagy az intervallum-felosztási szabály módosítása (a feladat, illetve az algoritmus többi részének változatlanul hagyása mellett) az algoritmus konvergenciájának megváltozását vonja maga után.

A következő állításban egy elegendő feltételt mutatok be (ez esetben már gyorsítóteszteket is feltételezve) az algoritmus globális konvergenciájára vonatkozóan. A feltétel az $\{f_k\}$ sorozat alkalmas megválasztásán alapul:

20. TÉTEL. *[40] Tegyük fel, hogy mind a célfüggvény, mind a korlátok befoglaló függvényei izoton tulajdonságúak és zéró-konvergensek. Tekintsük azt az intervallumos B&B algoritmust, amely a lehetséges megoldások tesztje mellett esetlegesen a kivágási, illetve a monotonitási tesztet is tartalmazza, és amely minden lépésben a legnagyobb $pup(f_k, \boldsymbol{Z})$ értékkel rendelkező boxot választja ki felosztásra, a maximális*

$pup(f_k, \boldsymbol{Z})$-vel rendelkező boxok közül pedig a legnagyobb $p(f_k, \boldsymbol{Z})$ értékűt választja. Ha az $\{f_k\}$ sorozat a globális minimumhoz, f^*-hoz tart, továbbá $\{f_k\}$ véges sok eleme kisebb, mint f^*, akkor az algoritmus globális minimumpontok egy halmazához tart.

BIZONYÍTÁS. Tegyük fel, hogy bár az $\{f_k\}$ sorozatra tett feltételek fennállnak, mégis, az algoritmus által felosztásra kijelölt boxok sorozatának van olyan $\{\boldsymbol{X}_l\}$ részsorozata, mely egy nemoptimális $\boldsymbol{x}'$ ponthoz tart ($f(\boldsymbol{x}') > f^*$). Vegyük észre, hogy $\boldsymbol{x}'$-ről feltehető, hogy lehetséges megoldás, ellenkező esetben ugyanis $G_j(\boldsymbol{X}_l) > 0$ fog teljesülni valamely j-re, illetve l-re a G_j zéró-konvergenciája miatt, azaz $\boldsymbol{X}_l$ törölve lesz a lehetséges megoldások tesztjével.

Vizsgáljuk először azt az esetet, amikor az 1. Lemma nem teljesül $\{\boldsymbol{X}_l\}$-re, azaz $pu(\boldsymbol{X}_l) > 0$ áll fenn minden $l \geq 1$-re. Az F zéró-konvergenciájának következményeként $\underline{F}(\boldsymbol{X}_l) \to f(\boldsymbol{x}')$ teljesül. Mivel $f_k \to f^*$ és F izoton, ezért létezik olyan i pozitív egész, melyre $\underline{F}(\boldsymbol{X}_l) > f_{k_l}$ teljesül, valahányszor $l \geq i$. Mindez azt jelenti, hogy valamely i' iterációs lépéstől kezdve az $\{\boldsymbol{X}_l\}$ részsorozat azon elemére, mely aktuálisan a Munkalistán van, mindig $p(f_{k_l}, \boldsymbol{X}_l) < 0$ áll. Ezt $pu(\boldsymbol{X}_l) > 0$-val egybevetve $pup(f_{k_l}, \boldsymbol{X}_l) < 0$ is teljesül az i'-edik lépéstől kezdődően. (Jegyezzük meg, hogy $i = i'$ nem mindig igaz: i' az algoritmus egy iterációs indexét jelenti, i pedig azt mutatja, hogy az $\{\boldsymbol{X}_l\}$ sorozat hányadik eleme tartózkodik épp a Munkalistán.)

Másrészt, könnyen belátható, hogy a Munkalistán bármely időpontban rajta kell lennie egy olyan $\boldsymbol{Y}$ boxnak, amely tartalmaz egy $\boldsymbol{x}^*$ optimális megoldást. Ez abból a tényből adódik, hogy az alkalmazott gyorsítótesztek definícióik értelmében nem törölhetnek egyetlen optimális megoldást tartalmazó intervallumvektort sem. Megvizsgálva a megfelelő $pup(f_k, \boldsymbol{Y})$ értékeket, F befoglaló tulajdonsága miatt $f^* - \underline{F}(\boldsymbol{Y}) \geq 0$-t kapjuk. Emiatt $f_k - \underline{F}(\boldsymbol{Y}) \geq 0$, ha $f_k \geq f^*$, azaz $p(f_k, \boldsymbol{Y})$ nemnegatívvá válik valamely i^* iterációs indextől kezdve. Azaz i^*-tól kiindulva $pu(\boldsymbol{Y}) \geq 0$-t figyelembe véve $pup(f_k, \boldsymbol{Y}) \geq 0$ teljesül.

Fentieket összefoglalva, a $\max(i', i^*)$ indexű iterációtól kezdve minden lépésben az aktuális $\boldsymbol{X}_l$ box (mely $\boldsymbol{x}'$-t tartalmazza, és negatív *pup* értékkel rendelkezik) lesz összehasonlítva (többek között) egy olyan $\boldsymbol{Y}$ box-szal, amely $\boldsymbol{x}^*$-ot tartalmazza és amelyhez nemnegatív *pup* érték tartozik. Az elsődleges kiválasztási kritérium miatt az algoritmus nyilvánvalóan ezután nem fogja $\boldsymbol{X}_l$-t kiválasztani felosztásra, ami ellentmond a kezdeti indirekt feltevésünknek.

Másodikként vizsgáljuk azt az esetet, amikor a feltételezett nemoptimális $\boldsymbol{x}'$-höz tartó $\{\boldsymbol{X}_l\}$ részsorozatra teljesül az 1. Lemma valamely állítása. Akkor valamely i' iterációs lépéstől kezdve $pu(\boldsymbol{X}_l) = pup(f_{k_l}, \boldsymbol{X}_l) = 0$, illetve $p(f_{k_l}, \boldsymbol{X}_l) < 0$ teljesül a Munkalistán aktuálisan jelenlévő $\boldsymbol{X}_l$ boxra.

Hasonlóan az első esetben tárgyaltakhoz, a Munkalistán ezúttal is jelen kell legyen egy optimális $\boldsymbol{x}^*$ megoldást tartalmazó, nemnegatív $p(f_k, \boldsymbol{Y})$, illetve $pup(f_k, \boldsymbol{Y})$ értékekkel rendelkező $\boldsymbol{Y}$ box, amennyiben az iterációs számláló elér valamely i^* értéket. Ekkor $\max(i', i^*)$ lépés után a $pup(f_{k_l}, \boldsymbol{X}_l) = 0$ értékkel rendelkező $\boldsymbol{X}_l$ box lesz összehasonlítva egyebek között egy nemnegatív *pup*-vel rendelkező, $\boldsymbol{x}^*$-ot tartalmazó $\boldsymbol{Y}$ box-szal. Könnyen látható, hogy $\boldsymbol{X}_l$ ez esetben sem lesz preferálva az algoritmus kiválasztó lépése által: ha $pup(f_{k_l}, \boldsymbol{Y}) > 0$, akkor az elsődleges kiválasztási szabály jobbnak találja $\boldsymbol{Y}$-t $\boldsymbol{X}_l$-nél, másrészt pedig, ha $pup(f_{k_l}, \boldsymbol{Y}) = pup(f_{k_l}, \boldsymbol{X}_l) = 0$, akkor

a másodlagos kritérium (azaz a C3 szabály) értelmében $p(f_k, \boldsymbol{Y}) \geq 0 > p(f_{k_l}, \boldsymbol{X}_l)$ miatt $\boldsymbol{Y}$ ugyancsak kedvezőbb választás az algoritmus számára, mint $\boldsymbol{X}_l$. Ez esetben is ellentmondásra jutottunk azzal a kezdeti feltevéssel, hogy az algoritmus $\{\boldsymbol{X}_l\}$-en keresztül egy nemoptimális ponthoz konvergálhat. Azaz, ha az f_k értékek sorozata f^*-hoz konvergál, és csak véges sok f_k érték kisebb, mint f^*, akkor a tárgyalt algoritmus kizárólag globális minimumpontok egy halmazához tart. □

Az előző tétel feltételei mellett a felosztásra kiválasztott boxok sorozatának torlódási pontjai a feladat globális minimumhelyeinek egy részhalmazát alkotják. A megtalált és a valódi minimumhelyek halmazának egyenlősége az esetleges rejtett minimumhelyek miatt (10. Definíció) az általánosságban nem teljesül. A 20. Tételben f^* legáltalánosabban használt közelítését, a globális minimum aktualizált legjobb felső korlátját ($\tilde{f}$-t) használva az alábbi korollárіumhoz jutunk:

1. KOROLLÁRIUM. *[40] Tekintsük azt az algoritmust, amely a lehetséges megoldások tesztje mellett esetlegesen a kivágási, illetve a monotonitási tesztet is tartalmazza, és amely minden lépésben a legnagyobb $pup(\tilde{f}, \boldsymbol{Z})$ értékkel rendelkező boxot választja ki felosztásra, a maximális $pup(\tilde{f}, \boldsymbol{Z})$-vel rendelkező boxok közül pedig a maximális $p(\tilde{f}, \boldsymbol{Z})$ értékűt választja. Ha biztosak lehetünk abban, hogy az $\tilde{f}$ értékek sorozata f^*-hoz tart, akkor az algoritmus által felosztásra kijelölt intervallumok konvergens részsorozatai a globális minimumpontok egy halmazához tartanak.*

[11]-t követve megvizsgálhatók azok a módszerek, melyekben egy heurisztikus intervallum-kiválasztási szabályt (jelen esetben C5-t) a C1 szabállyal kombináljuk egy $u : \mathbb{R}^2 \to \mathbb{R}$ függvény segítségével, mely azután a maximális $u(pup(f_k, \boldsymbol{Z}), -\underline{F}(\boldsymbol{Z}))$ értékkel rendelkező boxot választja ki felosztásra.

21. TÉTEL. *[40] Tegyük fel, hogy mind a célfüggvény, mind a korlátok befoglaló függvényei izoton tulajdonságúak és zéró-konvergensek. Tekintsük azt az intervallumos B&B algoritmust, amely a lehetséges megoldások tesztje mellett esetlegesen a kivágási, illetve a monotonitási tesztet is tartalmazza, és amely minden lépésben a legnagyobb $u(pup(f_k, \boldsymbol{Z}), -\underline{F}(\boldsymbol{Z}))$ értékkel rendelkező boxot választja ki felosztásra, a maximális $u(x, y)$ mennyiséggel rendelkező boxok közül pedig a legnagyobb $p(f_k, \boldsymbol{Z})$ értékűt választja.*

(a) *Ha az $\{f_k\}$ sorozat a globális minimumhoz, f^*-hoz tart, ezen sorozat véges sok eleme kisebb, mint f^*, valamint u szigorúan monoton növő mindkét változójában, akkor az algoritmus a globális minimumpontok egy halmazához tart.*

(b) *Másrészről, ha $f_k > f^* + \delta$ valamely $\delta > 0$-ra, akkor az algoritmus még az (a)-beli tulajdonsággal rendelkező u függvények esetén is konvergálhat egy nemoptimális ponthoz.*

BIZONYÍTÁS.

(a) Tegyük fel, hogy bár $f_k \to f^*$ és véges sok f_k érték kisebb f^*-nál, az algoritmus által felosztásra kijelölt boxoknak mégis létezik olyan $\{\boldsymbol{X}_l\}$ részsorozata, amely

egy nemoptimális $\boldsymbol{x}'$ ponthoz tart. A bizonyításban felhasználhatunk néhány, az előző tételben ugyanilyen feltételek fennállása esetén már igazolt állítást. Ezek az alábbiak:

– $\boldsymbol{x}'$-ről feltehető, hogy lehetséges megoldás;

– a Munkalistán minden lépésben létezik egy globális minimumpontot tartalmazó $\boldsymbol{Y}$ box, melyre $pup(f_k, \boldsymbol{Y}) \geq 0$ teljesül véges sok iterációs lépés eltelte után;

– véges sok lépés eltelte után $pup(f_{k_l}, \boldsymbol{X}_l) \leq 0$, azaz a vizsgált részsorozat aktuálisan $\mathcal{L}_{\mathcal{W}}$-ben található eleméhez nempozitív pup érték tartozik. Pontosabban, $pup(f_{k_l}, \boldsymbol{X}_l) < 0$, ha az $\{\boldsymbol{X}_l\}$ sorozatra nem teljesül az 1. Lemma, illetve $pup(f_{k_l}, \boldsymbol{X}_l) = 0$ egyébként – a 20. Tétel bizonyításának két esete alapján.

Továbbá, ugyancsak egy megfelelő iterációs lépéstől kezdve az $\boldsymbol{x}^*$-ot tartalmazó aktuális $\boldsymbol{Y}$ és az $\boldsymbol{x}'$-t tartalmazó aktuális $\boldsymbol{X}_l$ boxok között az $\underline{F}(\boldsymbol{Y}) < \underline{F}(\boldsymbol{X}_l)$ összefüggés áll fenn (F zéró-konvergenciája, valamint $\boldsymbol{x}'$ és $\boldsymbol{x}^*$ viszonya miatt), amiből

$$u(pup(f_{k_l}, \boldsymbol{Y}), -\underline{F}(\boldsymbol{Y})) > u(pup(f_{k_l}, \boldsymbol{X}_l), -\underline{F}(\boldsymbol{X}_l)) \tag{10}$$

következik u szigorú monotonitása miatt. (Vegyük észre, hogy amennyiben mind $pup(f_{k_l}, \boldsymbol{X}_l)$, mind $pup(f_{k_l}, \boldsymbol{Y})$ 0-val egyezik meg, akkor u második argumentumai garantálják a szigorú egyenlőtlenséget (10)-ben.) Azaz, véges sok lépés után az algoritmus nem fogja többé felosztásra kijelölni az $\{\boldsymbol{X}_l\}$ sorozatnak a Munkalistán lévő elemét, $\{\boldsymbol{X}_l\}$ tehát nem tarthat $\boldsymbol{x}'$-höz. A kapott ellentmondás igazolja a tétel (a) állítását.

(b) Először megmutatjuk az állítás teljesülését korlátozó feltételek nélküli feladatok és $u(p(f_k, \boldsymbol{X}), -\underline{F}(\boldsymbol{X}))$ típusú függvények esetén. Könnyen igazolható, hogy létezik olyan optimalizálási feladat, amelyen az algoritmus egy $\boldsymbol{x}'$ nemoptimális ponthoz tart, ahol $f(\boldsymbol{x}') = f^* + \delta$ (ld. [11]). Egy ilyen konstrukció – hasonlóan a 17. Tételben tárgyalthoz – az F befoglaló függvény alkalmas átdefiniálásán alapul (az izotonitás és zéró-konvergencia megtartása mellett) oly módon, hogy a $p(f_k, \boldsymbol{Z})$ értékeket megnöveljük azon $\boldsymbol{Z} \subseteq \boldsymbol{X}_0$ boxokra, melyek tartalmazzák $\boldsymbol{x}'$-t. Ha most egy olyan $u(p(f_k, \boldsymbol{X}), -\underline{F}(\boldsymbol{X}))$ függvényt tekintünk, amelyre tetszőlegesen nagy függvényérték állítható elő az első argumentum változtatásával, akkor elérhetjük azt, hogy az algoritmus elegendően sok iterációs lépés eltelte után minden lépésben az $\boldsymbol{x}'$-t tartalmazó boxot válassza ki felosztásra.

A fenti eljárás az értekezés 17. Tételében alkalmazott módszer alapján könnyen kiterjeszthető korlátozásos feladatok, valamint $u(pup(f_k, \boldsymbol{X}), -\underline{F}(\boldsymbol{X}))$ alakú kiválasztási függvények esetére. Vegyük az iménti konstrukció célfüggvényét és annak befoglaló függvényét, majd ebből állítsunk elő egy olyan korlátozásos feladatot, melynek minden pontja szigorúan lehetséges megoldás, valamint $pu(\boldsymbol{X}_0) = 1$. A korlátok befoglaló függvényeinek izotonitásából ezért $p(f_k, \boldsymbol{X}) = pup(f_k, \boldsymbol{X})$, és így $u(p(f_k, \boldsymbol{X}), -\underline{F}(\boldsymbol{X})) = u(pup(f_k, \boldsymbol{X}), -\underline{F}(\boldsymbol{X}))$, $\forall \boldsymbol{X} \subseteq \boldsymbol{X}_0$, $\boldsymbol{X} \in \mathbb{I}^n$ adódik, azaz az algoritmus előállítja ugyanazt a nemoptimális ponthoz

konvergáló boxsorozatot, mint amelyet a korlátozó feltételek nélküli feladatra kaptunk. Ezzel a tétel állításának (b) részét is igazoltuk. □

Térjünk most át a C7 szabály vizsgálatára. Vegyük észre, hogy bár formailag a C7 alapjául szolgáló $\underline{F}(\boldsymbol{X})/pup(\hat{f}, \boldsymbol{X})$ mennyiség az $u(x, y)$ formulával adott (azaz $u = -pupb(f_k, \boldsymbol{X})$ maximuma alapján történhet a kiválasztás), rá a 21. Tétel nem használható, mert $\underline{F}(\boldsymbol{X})/pup(\hat{f}, \boldsymbol{X})$ általában nem szigorúan monoton növő mindkét változójában: Ha feltesszük, hogy $\underline{F}(\boldsymbol{X})$ csak negatív értékeket vehet fel, akkor pozitív $pup(f_k, \boldsymbol{X})$ értékek esetén az

$$u(pup(f_k, \boldsymbol{X}), -\underline{F}(\boldsymbol{X})) = -pupb(f_k, \boldsymbol{X}) = \frac{-\underline{F}(\boldsymbol{X})}{pup(f_k, \boldsymbol{X})}$$

függvény a második változójában szigorúan monoton csökkenő lesz.

10. MEGJEGYZÉS. *Vegyük észre, hogy a 20. Tétel sem alkalmazható változatlan formában a C7 stratégiára; ez egyrészt a tétel bizonyításából fakadó tényezőkre vezethető vissza (hiszen* $pupb(f_k, \boldsymbol{X},)$ *előjelét* $\underline{F}(\boldsymbol{X})$ *előjele is befolyásolja), másrészt pedig egy egyszerű példával is megvilágítható: Konstruáljunk egy feladatot egyetlen optimális* $\boldsymbol{x}^*$ *megoldással, és legyen* $\boldsymbol{x}'$ *egy nemoptimális lehetséges megoldás. Tegyük fel, hogy mondjuk az első intervallum-felosztás során* $\boldsymbol{x}'$ *szeparálódik* $\boldsymbol{x}^*$*-tól, és a kapott részboxokra az alábbiak állnak: egyrészt legyen* $pup(f_k, \boldsymbol{Y}) = pu(\boldsymbol{Y}) = 0$*, és így* $pupb(f_k, \boldsymbol{Y}) = M$ *az* $\boldsymbol{x}^*$*-ot tartalmazó boxra (az 1. Lemma alapján ez a konstrukció könnyen kivitelezhető). Másrészt legyen* $pupf(f_k, \boldsymbol{Y}) < 0$ *az* $\boldsymbol{x}'$*-t tartalmazó boxra, és annak minden,* $\boldsymbol{x}'$*-t befoglaló részboxára (a 20. Tétel alapján). Amennyiben* $\underline{F}(\boldsymbol{X}) > 0$ *minden* $\boldsymbol{X} \subseteq \boldsymbol{X}_0$*,* $\boldsymbol{X} \in \mathbb{I}^n$ *esetén, akkor* $pupb(f_k, \boldsymbol{Y}) < 0$ *minden, az aktuálisan* $\mathcal{L}_{\mathcal{W}}$*-ben lévő, és* $\boldsymbol{x}'$*-t tartalmazó boxra. Így nyilván a* $\boldsymbol{x}^*$*-ot tartalmazó box sohasem kerül kiválasztásra.*

Amennyiben az algoritmusban előírjuk a kivágási teszt használatát az $\tilde{f}$ értékek alapján (a numerikus vizsgálatokban alkalmazott algoritmusainkban is ez volt a helyzet), akkor egy általános, az összes ismertetett kiválasztási szabályra érvényes konvergencia-feltétel fogalmazható meg (vö. 9. Megjegyzés):

22. TÉTEL. *[40] Tegyük fel, hogy mind a célfüggvény, mind a korlátok befoglaló függvényei zéró-konvergensek. Tekintsük azt az intervallumos B&B algoritmust, mely a lehetséges megoldások tesztje mellett a kivágási tesztet (illetve esetlegesen a monotonitási tesztet) is tartalmazza, és tetszőleges intervallum-kiválasztási szabályt használ (pl. a C1,...,C8 szabályok valamelyikét). Ha az* f_k *értékek sorozata* f^**-hoz tart, valamint véges sok iterációs lépés kivételével minden lépésben* $f_k = f(\boldsymbol{x}_k)$ *teljesül valamely* $\boldsymbol{x}_k \in \boldsymbol{X}_0$ *lehetséges megoldásra, akkor az algoritmus által felosztásra kijelölt intervallumok konvergens részsorozatai a globális minimumpontok egy halmazához tartanak.*

BIZONYÍTÁS. A konvergenciának az állításban megfogalmazott második feltétele annyit jelent, hogy (véges sok esettől eltekintve) a kivágási teszthez használt $\tilde{f}$ aktuális

értéke az f_k-nál nem nagyobb értékre állítható a k-adik lépésben, azaz, minden olyan $\boldsymbol{Y}$ box törlése lehetséges a kivágási teszt segítségével, melyre $\underline{F}(\boldsymbol{Y}) > f_k$.

Tegyük fel, hogy bár $f_k \to f^*$ és az i-edik lépéstől kezdve a kivágási teszthez használt küszöbértékek a fentiek szerint alakulnak, mégis, az algoritmus által felosztásra kiválasztott boxok sorozatának létezik olyan $\{\boldsymbol{X}_l\}$ részsorozata, mely a nemoptimális $\boldsymbol{x}'$ ponthoz ($f(\boldsymbol{x}') = f' > f^*$) konvergál. A korlátok zéró-konvergenciája alapján $\boldsymbol{x}'$ lehetséges megoldás, a célfüggvény zéró-konvergenciája miatt pedig az $\{\boldsymbol{X}_l\}$-en vett befoglaló függvényértékek sorozatára $\underline{F}(\boldsymbol{X}_l) \to f'$ teljesül. Ekkor viszont valamely i', $i' > i$ lépésben $\underline{F}(\boldsymbol{X}_l) > f_{i'}$ teljesül, azaz az $\{\boldsymbol{X}_l\}$ sorozatnak a Munkalistán tartózkodó elemét törli a kivágási teszt. Az algoritmus tehát nem tarthat $\boldsymbol{x}'$-höz, a kapott ellentmondás pedig a tétel igazolását jelenti. □

11. MEGJEGYZÉS. *A 22. Tételben a konvergencia elegendő feltétele valójában a 20. Tétel elegendő feltételének egy erősebb megfogalmazása: a 22. Tételbeli $f_k = f(\boldsymbol{x}_k)$ feltételt kielégítő $\boldsymbol{x}_k$ sorozat létezéséből ugyanis $f_k \geq f^*$ következik véges sok lépés kivételével, azaz, az f_k sorozatnak legfeljebb véges sok tagja kisebb f^*-nál. A 22. Tétel ezen erősebb feltétele teszi lehetővé a konvergencia bizonyítását általános intervallum-kiválasztási szabályok esetére.*

A globális minimum mindenkori legjobb felső korlátjának (azaz az aktualizált $\tilde{f}$ értékeknek) a sorozatára nyilvánvalóan teljesül a 22. Tétel f_k sorozatára tett második feltétel. Azaz ha biztosak lehetünk abban, hogy az $\tilde{f}$ értékek sorozata f^*-hoz tart, akkor a 22. Tétel feltételei mellett a C1–C8 szabályokat használó algoritmusváltozatok a globális minimumpontok egy halmazához tartanak. Jegyezzük meg, hogy az $\tilde{f} \to f^*$ előfeltevés ugyan elég erősnek tűnik, mégsem feltétlenül ekvivalens az optimalizálási feladat megoldásával. Számos esetben pl. ismert optimumérték mellett kell megkeresnünk a globális minimumhelyek halmazát.

Amint azt az ismertetett tételekben láttuk, a tárgyalt algoritmus globális minimumpontokhoz történő konvergenciájához elengedhetetlen egy, a globális minimum értékéhez tartó sorozat megadása. A kérdés az, hogyan állíthatók elő ilyen sorozatok. A globális minimum egy hagyományosan használt becslése az

$$\tilde{f}_k := \min\{\tilde{f}_{k-1}, \overline{F}(\boldsymbol{Y}^1), \ldots, \overline{F}(\boldsymbol{Y}^s)\} \tag{11}$$

formulával adott, ahol $\boldsymbol{Y}^1, \ldots, \boldsymbol{Y}^s$ a k-adik iterációs lépés intervallum-felosztása során előálló boxokat jelölik, valamint $\tilde{f}_0 = \overline{F}(\boldsymbol{X}_0)$. A globális minimum egy másik szokásos becslése a felosztásra kiválaszott boxokon számított célfüggvényértékek alsó korlátjával, azaz az

$$f_k := \underline{F}(\boldsymbol{X}_k) \tag{12}$$

módon adott, ahol $\boldsymbol{X}_k$ a k-adik iterációs lépésben felosztandó box.

A 6. Tétel szerint a $\{\underline{F}(\boldsymbol{X}_k)\}$ sorozat a C1 szabály esetén alulról tart f^*-hoz. A C1 szabályra egyszerűen igazolható ([51]) továbbá az, hogy a $\{\tilde{f}_k\}$ sorozat pedig felülről tart f^*-hoz. Csendes [11] ugyanakkor igazolta, hogy a C3 szabály esetén a (11) és (12)-beli becslések nem feltétlenül tartanak f^*-hoz. Hasonló – negatív – eredmény bizonyítható a korlátozásos feladatok esetén bevezetett intervallum-kiválasztási szabályokra:

2. Korollárium. *[40] Tegyük fel, hogy mind a célfüggvény, mind a korlátok befoglaló függvényei izoton tulajdonságúak és zéró-konvergensek. Tekintsük azt az intervallumos B&B algoritmust, amely a lehetséges megoldások tesztje mellett esetlegesen a kivágási, illetve a monotonitási tesztet is tartalmazza, valamint az alábbi szabályok egyike alapján választja ki a felosztandó intervallumot:*

- *maximális* $pup(f_k, \boldsymbol{X})$ *érték (C5 szabály),*
- *maximális* $u(pup(f_k, \boldsymbol{X}), -\underline{F}(\boldsymbol{X}))$ *érték,*
- *minimális* $pupb(f_k, \boldsymbol{X})$ *érték (C7 szabály).*

Ekkor mind a (11) által adott $\{\tilde{f}_k\}$*-ra, mind pedig a (12) által definiált* $\{\underline{F}(\boldsymbol{X}_k)\}$*-ra igaz az, hogy az illető sorozat nem feltétlenül tart* f^**-hoz.*

Bizonyítás. A C5 szabályra a 17. Tételből, az $u(x, y)$-t használó szabály esetén a 21. Tételből, a C7 szabályra pedig a 10. Megjegyzésből adódik, hogy az adott szabályt használó algoritmus konvergálhat *kizárólag* egy vagy több nemoptimális ponthoz (pl. valamely $\boldsymbol{x}'$-höz, ahol $f(\boldsymbol{x}') = f' > f^*$). Mindhárom szabály esetében az $\tilde{f}_k$ értékek sorozata vagy valamely f'-höz tart vagy pedig véges sok lépés után valamely f' alatti konstans értéket vesz fel. (Az utóbbi helyzet akkor állhat elő, ha az algoritmus nem használja a kivágási tesztet és az első iterációs lépések során sikerül néhány f' alatti értéket kinyernünk a felosztáskor kapott részintervallumokból – ezt azonban az algoritmus nemoptimális pont(ok)hoz való konvergenciája miatt nem tudjuk tovább javítani.) Azaz $\tilde{f}_k \to f^*$ nem feltétlenül teljesül.

Másrészt ha az algoritmus nemoptimális pont(ok)hoz konvergál, akkor F zérókonvergenciája miatt az $\{\underline{F}(\boldsymbol{X}_k)\}$ sorozat minden konvergens részsorozata valamely nemoptimális függvényértékhez tart. Azaz $\underline{F}(\boldsymbol{X}_k) \to f^*$ sem teljesül minden esetben. □

A tárgyalt becslések mellett természetesen számos más eljárás is kidolgozható az optimumnak az algoritmus futása közben történő közelítésére (pl. lokális keresők alkalmazásával, vagy az előálló boxokból számított más mennyiségekkel). Az új szabályok esetén is használható garantáltan konvergens pontsorozatok megadásának módszerei a kapcsolódó kutatások egyik fontos irányát képezik (lásd pl. [12]).

3.3.2 Az intervallum-felosztási szabályok konvergencia-tulajdonságai

Az adaptív multisection szabályokat Casado [3] és munkatársai ismertették a $p(\tilde{f}, \boldsymbol{X})$ értéken alapuló döntések esetére. Jelen szakaszban az újonnan javasolt $pup(\tilde{f}, \boldsymbol{X})$ mutató által irányított adaptív intervallum-felosztási szabályt (lásd 3.2. szakasz) vizsgálom. Az $\tilde{f}$ becsléseinek sorozatát a (11) összefüggés alapján állítjuk elő.

23. Tétel. *[40] Tekintsük azt az intervallumos B&B algoritmust, amely a Munkalistából minden iterációs lépésben a legkisebb* $\underline{F}(\boldsymbol{X})$ *értékkel rendelkező boxot választja*

ki felosztásra (azaz a C1 szabályt alkalmazza). Ekkor léteznek olyan (**) alakú optimalizálási feladatok, melyekre a célfüggvény és a korlátok befoglaló függvényei izotonok és α-konvergensek, és amelyekre a következő állítások teljesülnek:

1. *Tetszőlegesen nagy rögzített $N > 0$-ra az algoritmus által N db egymás utáni lépésben felosztásra kiválasztott boxok egyike sem tartalmaz globális minimumhelyet, és minden ilyen boxon vett $pup(\tilde{f}, \boldsymbol{X})$ érték nagyobb, mint egy előre rögzített $P_2 < 1$ szám.*

2. *Megadható az algoritmus által felosztásra kijelölt boxok sorozatának egy olyan, globális minimumhelyhez tartó részsorozata, melynek minden elemére $pup(\tilde{f}, \boldsymbol{X}) < P_1$, egy előre rögzített $0 < P_1$ esetén.*

BIZONYÍTÁS. A tétel bizonyítása alkalmas konstrukciók megadásával történik. Ehhez először tekintsünk egy olyan (**) alakú feladatot, melynek két különböző globális minimumhelye van: $\boldsymbol{x}^* \in \boldsymbol{X}_0$ és $\boldsymbol{x}' \in \boldsymbol{X}_0$, ahol $f(\boldsymbol{x}^*) = f^* = f(\boldsymbol{x}')$, $\boldsymbol{x}^* \neq \boldsymbol{x}'$, valamint a lehetséges megoldások halmaza tartalmaz egy $\hat{\boldsymbol{x}} \in \boldsymbol{X}_0$ nemoptimális megoldást: $f(\hat{\boldsymbol{x}}) > f(\boldsymbol{x}^*)$. Tegyük fel továbbá, hogy az $\boldsymbol{x}^*, \boldsymbol{x}', \hat{\boldsymbol{x}}$ pontok mindegyike *szigorúan lehetséges megoldás.*

1. A bizonyítás során a célfüggvény befoglaló függvényének megadására a [3]-ban definiált függvényt használom, majd ezt kiegészítem a korlátozó feltételekre megadott alkalmas befoglaló függvényekkel.

Jelölje $\{\boldsymbol{X}_i\}_{i=0}^{N_0+N-1}$ az algoritmus által felosztásra kiválasztott boxok azon halmazát, melyre $\hat{\boldsymbol{x}} \in \boldsymbol{X}_i$, N_0 pedig egy olyan konstans, hogy legfeljebb N_0 lépés elteltével $\hat{\boldsymbol{x}}$ a Munkalistán szeparálódik mindkét globális minimumponttól, azaz a Munkalistán különböző elemek tartalmazzák $\hat{\boldsymbol{x}}$-et, illetve $\boldsymbol{x}^*$-ot és $\boldsymbol{x}'$-t. (Jegyezzük meg, hogy a lentebb megadott konstrukció révén $\{\boldsymbol{X}_i\}_{i=0}^{N_0+N-1}$ egyértelműen meghatározottá válik.)

Az f befoglaló függvénye legyen az alábbi módon meghatározott: először képezzük az F függvényt természetes intervallum-kiterjesztés segítségével minden $\boldsymbol{X} \subseteq \boldsymbol{X}_0$ boxra, majd az $\{\boldsymbol{X}_i\}_{i=0}^{N_0+N-1}$ halmaz elemeire definiáljuk át F értékét a

$$K(\boldsymbol{Z}) = [\underline{F}(\boldsymbol{Z}) - dw^{\alpha}(\boldsymbol{Z}), \hat{f}],$$

formula szerint, ahol $\hat{f} \geq \overline{F}(\boldsymbol{X}_0)$, d pedig egy alkalmas nagy érték. A fenti befoglaló függvény α-konvergens (α = 1-gyel) és izoton [3], valamint, $K(\boldsymbol{Z})$ alsó korlátainak megadása miatt az algoritmus által kiválaszott $N_0 + N$ box éppen $\{\boldsymbol{X}_i\}_{i=0}^{N_0+N-1}$ lesz. Másrészt, a definiált felső korlátok miatt $\hat{f} = \tilde{f}$, így az ezen boxokon számított $p(\tilde{f}, \boldsymbol{Z})$ érték mindenhol 1-gyel lesz egyenlő.

Konstruáljuk meg a korlátozó feltételek befoglaló függvényeit a természetes intervallum-kiterjesztéssel. Ez implikálja a G_j-k izotonitását és α-konvergenciáját α = 1-gyel, ezért, mivel $\boldsymbol{x}'$ feltevésünk szerint szigorúan lehetséges megoldás, $pu(\boldsymbol{Z}) = pup(\hat{f}, \boldsymbol{Z}) = 1$ teljesül $\{\boldsymbol{X}_i\}_{i=N_0+1}^{N_0+N-1}$ minden elemére egy alkalmasan rögzített N_0 esetén (N_0 megadásakor tehát $\hat{\boldsymbol{x}}$-nak a minimumpontoktól való szeparálódását, illetve a G_j-k viselkedését kell figyelembe vennünk).

Összefoglalva a kapott eredményeket, az algoritmus által az első $N_0 + N$ lépésben kiválasztott box $\{\boldsymbol{X}_i\}_{i=0}^{N_0+N-1}$ lesz, melyek közül az utolsó N darab box nem tartalmaz globális minimumhelyet, de a rájuk számított *pup* érték minden esetben 1. Ezzel igazoltuk a tétel állításának 1. részét.

2. A második állítás bizonyításához az előző rész konstrukcióját folytatjuk: a célfüggvény befoglaló függvénye legyen most

$$K(\boldsymbol{Z}) = [\underline{f}(\boldsymbol{Z}) - cw^{\alpha}(\boldsymbol{Z}), \overline{F}(\boldsymbol{Z})],$$

ha $\boldsymbol{x}' \in \boldsymbol{Z}$, de $\hat{\boldsymbol{x}} \notin \boldsymbol{Z}$, ahol $f(\boldsymbol{Z})$ a korábbiakhoz hasonlóan f értékkészletét jelöli $\boldsymbol{Z}$-n (és így $\underline{f}(\boldsymbol{Z}) = f^*$), a c és α valós konstansok pedig az F α-konvergenciájából adódó értékek.

Másrészt azon boxokra, melyek tartalmazzák $\boldsymbol{x}^*$-ot, de nem tartalmazzák sem $\boldsymbol{x}'$-t, sem $\hat{\boldsymbol{x}}$-ot, az újradefiniált befoglalás legyen

$$K(\boldsymbol{Z}) = [\underline{f}(\boldsymbol{Z}) - cw^{\alpha}(\boldsymbol{Z}), E],$$

ahol $E = \underline{K}(\boldsymbol{Z}) + \delta(\tilde{f} - \underline{K}(\boldsymbol{Z}))/P_1$ valamely alkalmas $\delta > 1$-re. Ez utóbbi összefüggésből egyszerű számolás után $p(\tilde{f}, \boldsymbol{Z}) < P_1$ adódik. A fenti K izoton és α-konvergens [3], továbbá K alsó korlátja és az α-konvergencia miatt elegendően sok iterációs lépés után az algoritmus váltakozva választja ki felosztásra az $\boldsymbol{x}'$-t, illetve $\boldsymbol{x}^*$-t aktuálisan tartalmazó boxokat.

Ami a korlátozó feltételek befoglalásainak megadását illeti, vegyük észre, hogy az 1. pontban alkalmazott természetes intervallum-kiterjesztés használata elegendő a bizonyításhoz. Ekkor ugyanis a G_j-k izotonitása, α-konvergenciája és $\boldsymbol{x}'$ valamint $\boldsymbol{x}^*$ szigorúan lehetséges megoldás mivolta miatt véges számú iterációs lépés elérése után $pu(\boldsymbol{Z}) = 1$, azaz $pup(\tilde{f}, \boldsymbol{Z}) = p(\tilde{f}, \boldsymbol{Z})$ áll minden, az algoritmus által felosztásra kiválaszott boxra. Következésképpen, a felosztásra kijelölt boxoknak létezik olyan részsorozata, melynek elemei tartalmazzák $\boldsymbol{x}^*$-ot, és amelyekre $pup(\tilde{f}, \boldsymbol{Z}) = p(\tilde{f}, \boldsymbol{Z}) < P_1$ teljesül. Ezzel igazoltuk a tétel állításának 2. részét is. □

A tétel következménye az, hogy még a befoglaló függvények tulajdonságaira tett erős kikötések ellenére is előfordulhatnak olyan esetek, amikor a kiválasztott boxok tetszőlegesen sok (N számú) iterációs lépésen keresztül nem tartalmaznak globális minimumhelyet, és amelyekre mégis a legtöbb darabra történő felosztást alkalmazzuk. Jegyezzük meg azonban, hogy ez csupán a legrosszabb esetet jelenti; a gyakorlatban, amint azt látni fogjuk, ennél lényegesen kedvezőbb a helyzet.

3.4 Számítógépes vizsgálatok

3.4.1 Elhelyezési feladatok

A J. Fernández Hernández által a numerikus vizsgálatokhoz javasolt feladatosztály, a 'taszításon alapuló elhelyezési problémák' (obnoxious facility location problems) [17, 18, 19, 34] lényege az alábbi: egy földrajzi területen kell elhelyeznünk egy olyan

objektumot, mely az adott régió lakosaiból valamiféle ellenszenvet vált ki, azaz szeretnék azt a saját településüktől minél messzebb elhelyezni. A cél tehát az objektum egy olyan pozíciójának meghatározása, amely minimalizálja a régióban élők aggregált ellenszenvét. Ugyanakkor az elhelyezésnél a terület geográfiai jellemzőit is figyelembe kell venni, azaz az elhelyezés bizonyos területeken tiltott.

A probléma az alábbi módon modellezhető: m darab várost tekintve, ha az objektumot az $\boldsymbol{x} \in \mathbb{R}^2$ pontban helyezzük el, akkor az $\boldsymbol{a}_i \in \mathbb{R}^2$ pontban lévő várost jellemző ellenszenvérték az

$$\mathrm{rp}(\boldsymbol{a}_i, \boldsymbol{x}) = \frac{1}{1 + e^{\alpha_i + \beta_i \cdot d_i(\boldsymbol{a}_i, \boldsymbol{x})}}$$

módon adott, ahol $\alpha_i \in \mathbb{R}$ és $\beta_i \in \mathbb{R}$, $\beta_i > 0$ előre rögzített paraméterek, $d_i(\boldsymbol{a}_i, \boldsymbol{x})$ pedig a város és az objektum távolsága. Célunk ezek alapján az

$$f(x) = \sum_{i=1}^{m} \omega_i \cdot \mathrm{rp}(\boldsymbol{a}_i, \boldsymbol{x}),$$

függvény minimalizálása, ahol $\omega_i \in \mathbb{R}$ az i-edik város súlyát (pl. lakosainak számát) jelenti. A feladat korlátozó feltételeit a tiltott területek adják meg, ezek egyrészt a városok körüli, ω_i-vel arányos sugarú körök, másfelől pedig az ún. védett területek, valamint a földrajzi régió határait leíró feltételek. A vizsgált modellben a védett területek és a határok egyenesek, körök, ellipszisek, illetve parabolák által adottak.

A javasolt feladathalmaz összesen 300 problémát tartalmazott: a keresési tartomány minden esetben ugyanakkora volt, a tiltott területek pedig ezen belül véletlenszerűen helyezkedtek el. A tekintett városok száma 5 és 45 között, az összes korlátozó feltétel száma pedig 10 és 105 között változott.

Inkább csak érdekessége miatt, illetve a feladat nehézségének demonstrálására ismertetek egy példát, ami az elhelyezési feladatok gyakorlati hasznát mutatja. Egy egyszerű modellben Magyarország területén a legnépesebb 20 város figyelembe vételével próbáltam elhelyezni egy ellenszenvet kiváltó objektumot. Az országhatárokat 48 lineáris korláttal közelítettem. Mivel a kapott poligon nem konvex, így a korlátok kezelése sem az általánosan tárgyalt módszerrel történt, azokat nem lehet ugyanis logikai *ÉS* kapcsolatban figyelembe venni, mint a (∗∗) formulában. A határokat megadó szakaszokat – a poligonok konvex burka meghatározásának jól ismert módszeréhez hasonlóan – meredekségük monotonitása alapján osztályokba sorolhatjuk, és ezeket egy-egy osztályon belül *VAGY* kapcsolatban kezelhetjük. Így a lehetséges megoldások halmazát azon pontok jelölik ki, melyek minden korlátozó feltételcsoport szerinti lehetséges megoldások (azaz az egyes csoportokat szimbolikusan az *ÉS* operátorral köthetjük össze). A határokon belül tiltott régióként a Budapest középpontja körüli 30 km-es, míg a többi város középpontja körüli 15 km-es kör alakú területeket tekintettem. A célfüggvényben szereplő súlyok a figyelembe vett városok népességével voltak arányosak. Az optimalizáló módszer a Moore–Skelboe-féle intervallum-kiválasztási szabályon és a statikus felező felosztáson alapult, a cél pedig az *összes* globális minimumpont meghatározása volt. Az eredményt a 2. Ábra szemlélteti, amelyen az algoritmus által generált felosztásokat tüntettem fel (a városok körüli tiltott régiók

2. Ábra: Az elhelyezési feladat egy egyszerű modelljének megoldása.

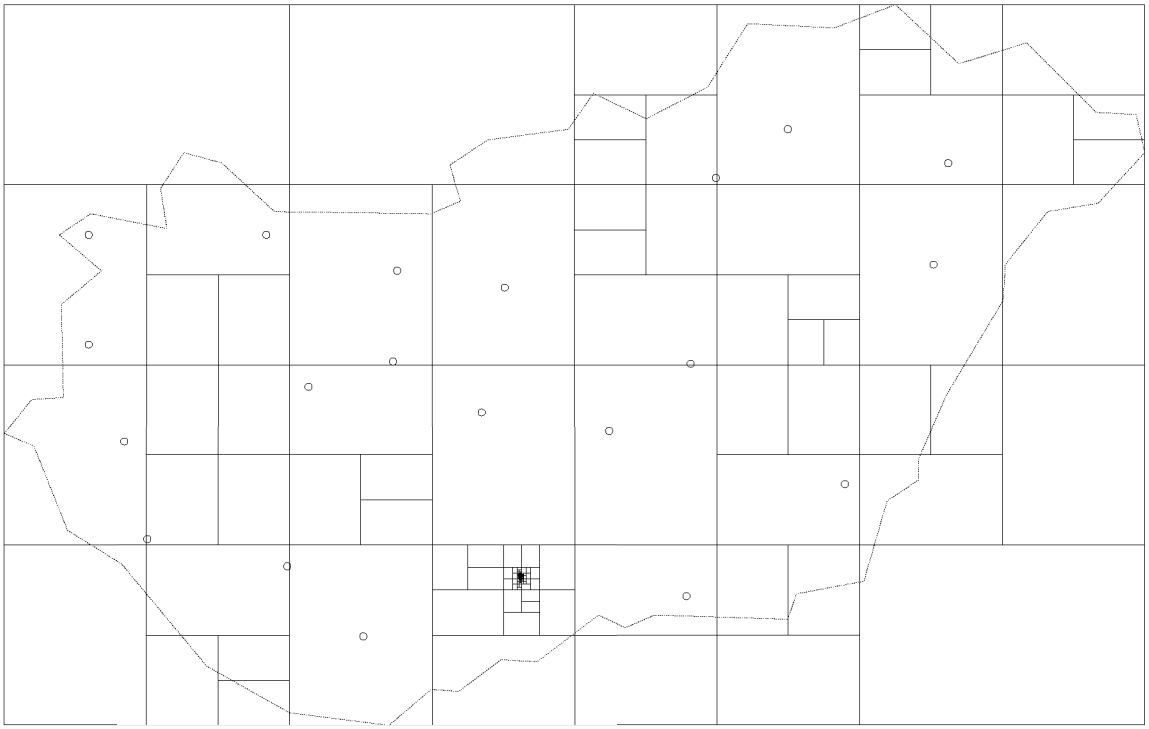
jelölését a jobb áttekintés miatt mellőztem, csupán a városok középpontját jelöltem). Amint megfigyelhető, a módszer rendkívül hatékonyan működött, néhány tizedmásodperc futási idő alatt megtalálta az ország északkeleti határán található globális minimumhelyet. A minimumhely befoglalásának pontossága a valóságban körülbelül 10 méternek felel meg, ami a gyakorlati alkalmazásokban megfelelő.

Elkerülendő az országhatáron történő elhelyezést (pl. a környező országok igényei miatt), a következő modellben azt is kikötöttem, hogy az objektumot a határtól legalább 30 km-re kell elhelyezni. A kapott eredményt a 3. Ábra mutatja. A feladat lényegében nem vált nehezebbé, a minimumhely (esetlegesen minimumhelyek) ezúttal a déli régióban, a módosított lehetséges tartomány határán lévő boxban található(k).

A fenti modell nyilván sok tekintetben tovább finomítható, de már a bemutatott egyszerű megközelítések is jól mutatják a jelen vizsgálatokhoz generált feladatok legfontosabb tulajdonságait.

3.4.2 Standard globális optimalizálási feladatok

A tesztfeladatok halmazának kibővítése egy általánosan használt (pl. [15, 39, 54]) korlátozó feltétel nélküli standard feladatcsoport alapján történt. Ezen feladatokból kiválasztottam a legnehezebbeket a különböző típusok szerint, majd ezeket véletlen módon generált korlátokkal láttam el. A feladathalmaz választásának egy további fontos indoka az volt, hogy Csendes [11]-ben ugyanezeket a feltétel nélküli feladatokat vizsgálta a jelen módszerek egyik kiindulópontjának tekintett pf-típusú eljárások esetén. A kiválasztott 6 feladat az alábbi volt: Shekel–10 (4-dimenziós), Hartman–



3. Ábra: Az elhelyezési feladat egy másik modelljének megoldása.

6 (6-dim.), Goldstein–Price (2-dim.), Levy–3 (2-dim.), Ratz–4 (2-dim.) és EX–2 (5-dim.). A feladatok definíciói és a javasolt keresési tartományok megtalálhatók [62]-ben, illetve [15]-ben. Érdemes megemlíteni, hogy az alapul vett standard feladathalmaz tartalmaz néhány további nehéz feladatot is (pl. Levy–8, ..., Levy–12, Schwefel–2.7, Griewank–5). Ezekről azonban egyszerűen látható, hogy a kivágási teszt és $f_k = f^*$ használatakor $p(f^*, \boldsymbol{X}) = 0$ és így $pup(f^*, \boldsymbol{X}) = 0$ teljesül a keresési tartomány összes $\boldsymbol{X}$ boxára. Azaz ezen feladatokon a C3–C8 mutatók mindegyike minden felosztásra kiválasztott boxra ugyanolyan értéket ad, így a nekik megfelelő intervallum-kiválasztási szabályok között nincs különbség. Mivel az implementált algoritmusokban másodlagos, illetve harmadlagos kiválasztási szabályként az alapvető C1-t alkalmazom, a C3–C8 stratégiák minden futási mutatója a klasszikus Moore–Skelboe stratégia eredményét szolgáltatja.

A standard feladatokhoz generált korlátozó feltételek lineáris, illetve kvadratikus korlátok voltak, dimenziószámuk minden esetben a tekintett célfüggvény dimenziójával egyezett meg. A korlátok száma $r = 15, 30$ és 45 volt, az egyes r értékekhez pedig 5-5 különböző feltételhalmazt (azaz minden egyes kiválasztott célfüggvényhez 15 különböző korlátozásos feladatot) állítottam elő. A korlátokat minden esetben oly módon generáltam, hogy az eredeti (korlátozás nélküli) feladat globális minimuma megegyezzen a kapott korlátozásos feladat minimumával (ezzel azt kívántam elérni, hogy a célfüggvény tulajdonságai – és így a feladatok nehézsége – megmaradjanak, legalábbis a minimumhelyek környezetében).

3.4.3 Az intervallum-kiválasztási szabályok vizsgálata

A numerikus teszteket egy PC-n (Intel Pentium IV. 1400 MHz processzor, 1 Gbyte RAM) hajtottam végre Linux operációs rendszer alatt. A 2. Fejezet programjaihoz hasonlóan az intervallum aritmetika, az intervallumos adatszerkezetek és a standard függvények megvalósítása Knüppel [31] PROFIL/BIAS könyvtárából származott, a B&B algoritmusok elemei pedig a C–XSC Toolbox könyvtár alapján [22, 29] készültek. A különböző hardver és szoftver környezetek összehasonlítására szolgáló Standard Időegység (2.3 szakasz) valós adattípusok használata esetén (`'real'`, `'double'`) 0.00076 másodperc, természetes intervallum-kiterjesztés és intervallumos adattípusok (`'interval'`) használatakor pedig 0.00674 másodperc volt.

Befoglaló függvények konstruálásához a természetes intervallum-kiterjesztést alkalmaztam (a középérték formulákra tehát ezúttal nem támaszkodtam), gyorsítóesztként pedig a lehetséges megoldásokat vizsgáló, a középponti (kivágási) és monotonitási teszteket építettem be az algoritmusba. Mindez azt jelenti, hogy a célfüggvény és a korlátok befoglaló függvényei mellett bizonyos boxokra a célfüggvény gradiensének befoglalását is számítottam. A tesztek alkalmazásakor természetesen ezúttal is figyelembe kellett venni azt, hogy a gyorsító lépések a fenti sorrendben igényelnek egyre több és több számítást; tehát ha valamely teszttel a teljes vizsgált box törölhető, akkor a további vizsgálatokhoz szükséges számítások elkerülhetők. Így például amennyiben a tekintett boxban garantáltan nincsenek lehetséges megoldások, akkor a célfüggvény és a gradiens befoglaló értékeit nem kell kiszámítani.

A Munkalistát egyfajta kiegyensúlyozott bináris keresőfával (AVL-fával) implementáltam: az egyes csomópontokhoz az elsődleges kiválasztási szabály szerinti egyforma értékekkel rendelkező boxok voltak hozzárendelve, melyeket minden csomópont esetén egy-egy egyszeresen láncolt listában tároltam (a másodlagos, harmadlagos döntési szabályok szerint rendezve). A fa csomópontjai a hozzájuk tartozó kiválasztási érték szerint voltak rendezve, megkönnyítve ezzel a használt listaműveletek (beszúrás és törlés) végrehajtását.

A vizsgálatok során megállási feltételként az aktuális boxra teljesülő $w(F(\boldsymbol{X})) < \varepsilon$ összefüggést alkalmaztam az $\varepsilon = 10^{-4}$ választással. [11]-at követve az $\{f_k\}$ sorozat választása $f_k = f^*,\ k = 0, 1, \ldots$ volt, azaz a globális minimum értékét előre adottnak tekintettem. (Erre amiatt volt szükség, mert – amint arról az előző szakaszban szó volt – a vizsgálatok időpontjában az új módszerek esetén nem volt ismert olyan, futási időben előállítható pontsorozat, amely f^*-hoz tart (vö. [12]). Másrészt a gyakorlatban is találhatunk olyan feladatokat, amelyekre a globális minimum értéke (vagy annak jó közelítése) ismert, és a minimumhelyeket kell meghatároznunk: lényegében ilyenek a 4. Fejezetben ismertetendő körpakolási feladatok is.) Könnyen látható, hogy ismert f^* esetén, ha az összes optimumhely meghatározása a cél (azaz az algoritmust $\mathcal{L}_{\mathcal{W}}$ kiürüléséig futtatjuk), akkor a felosztott boxok *halmaza* megegyezik az összes intervallum-kiválasztási szabály esetén. Így az egyes eljárások minden fontos futási jellemzője is megegyezik. Emiatt jelen Fejezet numerikus vizsgálataiban az algoritmusokat az első jelölt $\mathcal{L}_{\mathcal{S}}$-be tételekor megállítottam (hasonlóan [11]-hoz).

A C4, C6 és C8 hibrid szabályok esetén az N_m értékeket egyformán 100-ra választottam. Casado és munkatársai [6] részletes vizsgálatot végeztek a korlátozás nélküli

Szabály	CPU	rel.	MLL	rel.	NFE	rel.	NGE	rel.
C1	289.15		5 962		56 639		21 743	
C2	322.57	112%	5 057	85%	64 488	114%	24 947	115%
C3 (pf)	75.27	26%	2 794	47%	17 412	31%	3 780	17%
C4 (pf)	295.38	102%	5 430	91%	60 311	106%	22 572	104%
C5 (pup)	110.31	38%	3 496	59%	22 956	41%	6 383	29%
C6 (pup)	297.28	103%	5 455	91%	60 651	107%	22 739	105%
C7 (pupb)	105.14	36%	3 969	67%	21 480	38%	5 381	25%
C8 (pupb)	300.05	104%	5 478	92%	60 664	107%	22 736	105%

3. Táblázat: 300 elhelyezési (facility location) feladat megoldásának eredményei a különböző intervallum-kiválasztási szabályokat használó algoritmusok esetén. A táblázat a futások CPU idejének összegét (CPU), a maximális Munkalistahosszak összegét (MLL), valamint a célfüggvény, illetve gradiens kiértékelések számának összegét (NFE, NGE) mutatja. Minden oszlop a C1 módszer eredményéhez viszonyított relatív értékeket is tartalmazza, százalékos formában.

feladatokon alkalmazott C4 hibrid szabály esetén a megfelelő N_m értékek megadásához; ez alapján $N_m = 100$ megfelelőnek tűnt a jelen feladatok esetére is.

Érdemes hangsúlyozni, hogy az egyes intervallum-kiválasztási szabályok az algoritmusokon belül különböző *intervallum-felosztási* szabályokkal kombinálhatók. Mivel a kiválasztás és felosztás hatását külön kell választanunk, ezért az intervallum-kiválasztási szabályok vizsgálata során egységes felosztási szabályt kell használnunk. A numerikus teszteket próbaképpen elvégeztem a 3.1 szakaszban ismertetett statikus felosztási szabályok mindegyikével kombinálva, és azt tapasztaltam, hogy a feladatok döntő többségében a negyedelő módszer (felezés a két legszélesebb komponensre merőlegesen) a legkedvezőbb az összes kiválasztási módszer esetén. Az alábbiakban ezért a statikus négy részre osztással kombinált intervallum-kiválasztási szabályokra kapott eredményeket ismertetem.

Az elsőként bemutatott tesztsorozatban az elhelyezési feladatok 300 generált problémapéldányát oldottam meg a C1–C8 intervallum-kiválasztási szabályok mindegyikével. A továbbiakban egy adott kiválasztási szabállyal működő algoritmusra egyszerűen az illető szabály jelölésével hivatkozom. Az eredményeket a 3. Táblázat tartalmazza. A táblázatban a futások CPU idejének összegét (CPU), a Munkalisták feladatonkénti maximális hosszainak összegét (MLL), valamint a célfüggvény, illetve gradiens kiértékelések számának összegét (NFE, illetve NGE) adtam meg. Minden stratégiánál a kapott mutatók relatív értékeit is feltüntettem a C1-gyel jelzett Moore–Skelboe-féle kiválasztási szabállyal kapott adatokhoz viszonyítva.

A 3. Táblázat eredményeit a futási idő szempontjából értékelve megállapíthatjuk, hogy a C3, C5, C7-tel jelzett szabályok lényegesen gyorsabbnak bizonyultak az alapul vett C1 módszernél. Hozzá kell azonban tennünk, hogy a leggyorsabb a korábbi $p(f_k, \boldsymbol{Z})$ mutatón alapuló C3 szabállyal működő eljárás volt: ez az algoritmus 26%-át igényelte a C1 által felhasznált CPU időnek, míg a $pupb(f_k, \boldsymbol{Z})$-n alapuló C7 és a $pup(f_k, \boldsymbol{Z})$-n alapuló C5 módszerre ugyanez az érték 36%, illetve 38%. Másrészt

a vizsgált hibrid szabályok (C4, C6, C8) és az alapmódszer közötti eltérés kevésbé jelentős, rendre 2, 3, illetve 4 százalékos többlet időfelhasználás C1-hez képest.

A tárigényt jellemző MLL mutatókat megfigyelve azt találjuk, hogy a C2–C8 algoritmusok mindegyike kevesebb memóriát használt fel, mint az alapmódszer. Közülük is C3 tűnik a legjobb választásnak, a C1-hez képest kapott 43%-os tárfelhasználással, majd a további nem-hibrid szabályokkal vezérelt algoritmusok következnek, 59 (C5), illetve 67 százalékkal (C7). A hibrid szabályokkal dolgozó módszerek ugyanakkor csupán 8-9%-kal használtak kevesebb memóriát, mint C1.

Az összes függvény- és gradienshívások relatív értékei az időigényhez nagyon hasonló mintát mutatnak: ez azt jelenti, hogy a futásidőt döntően az NFE és NGE értékek határozták meg, a listahosszak mérete miatt a listakezelésre fordított idő nem volt jelentős. Ezért ismét a C3 szabály a legjobb választás (31% az NFE és 17% az NGE mutató esetén). A C5 és C7 szabályok ennél némileg többet számoltak (41% és 38% a függvényhívásokra valamint 29% és 25% a gradiensszámításokra az alapmódszerhez képest). A hibrid szabályok a Moore–Skelboe-szabályhoz képest 6-7%-kal többször számították ki a célfüggvény befoglaló függvényének értékét, és 4-5%-kal többször a célfüggvény gradiensének befoglaló függvényértékét.

Összefoglalva a kapott eredményeket, a vizsgált elhelyezési feladatokon a korábbi C3 típusú szabály volt a leghatékonyabb, de ettől nem sokkal maradtak el az új típusú (nem-hibrid) C5 és C7 eljárások sem. A feladatosztály nehézsége ugyanakkor jól jellemezhető a kapott abszolút mutatószámokkal: a 300 feladatot a Moore–Skelboe módszert használó algoritmus is átlagosan 189 függvény- és 72 gradiensszámítással oldotta meg. Amint azt látni fogjuk, a standard tesztfeladatok módosított változatai között ennél lényegesen nehezebben megoldható feladatok is előfordultak.

Tekintsük most a Shekel–10 függvény korlátozásos változatain kapott eredmények összefoglalását (4. Táblázat). A futásidőket megvizsgálva legfontosabb megállapításunk az, hogy az összes heurisztikus algoritmus változat (azaz a C3–C8 szabályok) gyorsabban dolgozott, mint a C1 szabály, igaz, a relatív javulás minden esetben csupán néhány százalékos; a két leghatékonyabb módszer (a pf-típusú C3, illetve a *pup*-típusú C5 szabályt használó eljárás) az alapul vett C1-hez képest egyaránt 4 százaléknyi javulást eredményezett. A listahosszak mutatóit elemezve szembeötlő, hogy az MLL értékek minden esetben igen alacsonyak: valójában minden egyes problémapéldányt az összes algoritmusváltozat meg tudott oldani egyenként maximálisan 2 vagy 3 tárolási egység használatával. Ez arra utal, hogy a bemutatott algoritmusok rendkívül kifinomultak, bizonyos feladatokon szinte nem is hajtanak végre a Munkalista méretét megnövelő szükségtelen intervallum-felosztásokat.

A függvény- és gradiens kiértékelések relatív számának alakulása közelítőleg a futásidők adatait követi: az NFE értékekben a C3 és C5 bizonyult a legjobbnak (96%-kal), a gradiens befoglalását pedig minden módszer nagyjából ugyanannyiszor számította ki a 15 feladaton.

Az 5. Táblázat a korlátozásos Hartman–6 feladatokon elvégzett számítások eredményét mutatja. Amint a futtatások outputjaiból és a korlátozás nélküli Hartman–6 feladatra kapott korábbi eredményekből (pl. [11, 39]) kiderül, a hozzáadott korlátok sokkal nehezebbé tették a tekintett 15 feladat nagy részét. A két leghatékonyabb módszer egyértelműen a pf-típusú hibrid C4, illetve az újonnan javasolt hibrid C6

Szabály	CPU	rel.	MLL	rel.	NFE	rel.	NGE	rel.
C1	3.20		31		2 618		1 534	
C2	3.20	100%	31	100%	2 618	100%	1 534	100%
C3 (pf)	3.07	96%	33	106%	2 523	96%	1 529	100%
C4 (pf)	3.11	97%	31	100%	2 618	100%	1 534	100%
C5 (pup)	3.06	96%	33	106%	2 523	96%	1 529	100%
C6 (pup)	3.11	97%	31	100%	2 618	100%	1 534	100%
C7 (pupb)	3.13	98%	31	100%	2 618	100%	1 534	100%
C8 (pupb)	3.12	98%	31	100%	2 618	100%	1 534	100%

4. Táblázat: 15 korlátozásos Shekel–10 feladat megoldásának eredményei a különböző intervallum-kiválasztási szabályokat használó algoritmusok esetén.

Szabály	CPU	rel.	MLL	rel.	NFE	rel.	NGE	rel.
C1	35 022.20		2 559 703		25 293 057		3 761 838	
C2	31 683.30	90%	1 436 597	56%	26 102 840	103%	3 911 177	104%
C3 (pf)	2 866.24	8%	712 566	28%	2 138 638	8%	272 999	7%
C4 (pf)	195.01	1%	31 402	1%	149 647	1%	32 913	1%
C5 (pup)	3 280.22	9%	225 776	9%	2 515 313	10%	368 425	10%
C6 (pup)	191.50	1%	22 453	1%	144 329	1%	39 133	1%
C7 (pupb)	32 986.40	94%	6 636	0%	26 102 840	103%	3 911 177	104%
C8 (pupb)	46 728.90	133%	2 558 641	100%	25 293 094	100%	3 761 838	100%

5. Táblázat: 15 korlátozásos Hartman–6 feladat megoldásának eredményei a különböző intervallum-kiválasztási szabályokat használó algoritmusok esetén.

stratégiát használó eljárás, melyek egyaránt mintegy 99%-os (két nagyságrendnyi) javulást adtak C1-hez képest az összes vizsgált mutató tekintetében. A C4 és C6 sorok abszolút mutatószámait összevetve azt kapjuk, hogy az új *pup*-típusú szabály 2%-kal kevesebb CPU-időt használt C4-nél, illetve 4%-kal kevesebb célfüggvény kiértékelést végzett, ugyanakkor mintegy 19%-kal többször számított gradienst, mint a C4 eljárás.

A korlátozó feltételeket is figyelembe vevő szabályok esetén a gradiens-kiértékelések számának megnövekedése más tesztfeladatokon is megfigyelhető. A jelenség oka az, hogy a gradiens befoglalásának kiszámítása (a monotonitási teszt alkalmazásakor) csak olyan boxokra történik meg, melyek minden pontja garantáltan lehetséges megoldás. Tekintsünk két, a Munkalistán jelenlévő boxot azonos (vagy körülbelül azonos) $p(f_k, \boldsymbol{Z})$ értékkel úgy, hogy az egyik box minden pontja biztosan lehetséges megoldás, míg a másik boxra ez nem teljesül. Akkor a $pup(f_k, \boldsymbol{Z})$ definíciója miatt a C5, C6, C7, C8 szabályok esetén az előbbi box lesz preferálva a kiválasztásra, majd felosztásra – ellentétben a C3 és C4 szabályokkal, amikor a két box közti választás nem függ a lehetséges megoldások tartalmazásától.

Visszatérve a Hartman–6 feladatokhoz, fontos megemlíteni, hogy a legnagyobb eltérés a C6 és C4 módszer között a tárigény tekintetében jelentkezett: az új C6

Szabály	CPU	rel.	MLL	rel.	NFE	rel.	NGE	rel.
C1	79.51		24 188		423 612		323 000	
C2	75.91	95%	30 979	128%	423 717	100%	323 105	100%
C3 (pf)	1.37	2%	2 519	10%	7 833	2%	4 055	1%
C4 (pf)	18.31	23%	14 035	58%	100 896	24%	67 079	21%
C5 (pup)	1.13	1%	1 976	8%	6 415	2%	3 423	1%
C6 (pup)	18.14	23%	13 442	56%	99 698	24%	67 927	21%
C7 (pupb)	76.73	97%	18 984	78%	414 548	98%	319 840	99%
C8 (pupb)	78.03	98%	23 145	96%	423 612	100%	323 000	100%

6. Táblázat: 15 korlátozásos Goldstein–Price feladat megoldásának eredményei a különböző intervallum-kiválasztási szabályokat használó algoritmusok esetén.

összességében 28 százalékkal kevesebb memóriát igényelt a 15 feladaton, mint a C4 eljárás, ami igen biztató a *pup* mutatót használó szabályokra nézve.

Megvizsgálva a további szabályok működését, az ugyancsak párba állítható nemhibrid C3 és C5 módszerek is igen hatékonynak (bár a C6-hoz képest lényegesen rosszabbnak) bizonyultak, 90-93% közötti javulást adva C1-hez képest a CPU, NFE, NGE mutatókban. Ezekben az értékekben a C3 mutat némi előnyt C5-tel szemben. A tárfelhasználásban azonban már nagyobb különbséget, 68%-os relatív javulást figyelhetünk meg C5-ben C3-hoz képest. Még egy érdekes adatra érdemes felhívni a figyelmet, ez pedig az ugyancsak újonnan javasolt C7 szabály memóriaigénye, ami messze a legkedvezőbb az összes eljárás között. Tehát a Hartman–6-hoz hasonló típusú feladatok esetén, ha a feladat nehéznek tűnik, de a rendelkezésre álló memória kevés, a C7 szabály alkalmazása lehet a célszerű választás.

A Goldstein–Price probléma korlátozásos változatain kapott numerikus eredmények (6. Táblázat) értékelésekor elsőként a C3 és C5 szabályoknak a C1 alapmódszerhez képest kimutatható jelentős előnyét kell hangsúlyoznunk: 1-2% az alapul vett 100%-hoz képest a futási időkben, illetve az NFE és NGE mutatókban, valamint 8-10% a tárfelhasználás tekintetében. Fontos látnunk, hogy az új C5 szabály ezen belül jóval kedvezőbb a C3-nál: a C5 algoritmus összességében 18%-kal kevesebb processzoridőt (bár az abszolút futási időkben csak kis különbséget), átlagosan 22%-kal kisebb Munkalistákat és 18%-kal, illetve 16%-kal kevesebb célfüggvény- illetve gradiensszámítást tud felmutatni, mint a C3. A megfelelő hibrid intervallum-kiválasztási szabályokkal rendelkező módszerek, C4 és C6 nagyjából egyformán viselkedtek: a Moore–Skelboe-szabályt használó alapalgoritmushoz képest 23% CPU időt és 56-58% memóriát használtak. A további algoritmusváltozatok, nevezetesen C2, C7 és C8 többségükben nem túl jelentős, 2% és 5% közötti javulást mutattak C1-hez képest a vizsgált mennyiségek tekintetében. A kedvező kivételt a C7 módszer memóriaigénye jelentette: bár C7 futási ideje a C1-gyel közel azonos, mégis, C7 az alapmódszer tárfelhasználásához képest csupán 78 százaléknyi memóriát igényelt.

A Levy–3 probléma korlátozásos változatain kapott eredményeket a 7. Táblázat mutatja. A felhasznált CPU-idő, a célfüggvény- illetve a gradiensszámítások tekintetében az egyes algoritmus variánsokra kapott eredmények hasonlóak: mind a

Szabály	CPU	rel.	MLL	rel.	NFE	rel.	NGE	rel.
C1	2.11		276		5 006		2 964	
C2	2.08	99%	277	100%	5 003	100%	2 961	100%
C3 (pf)	0.96	45%	232	84%	2 315	46%	1 380	47%
C4 (pf)	2.10	100%	300	109%	5 006	100%	2 964	100%
C5 (pup)	0.96	45%	156	57%	2 303	46%	1 467	49%
C6 (pup)	2.10	100%	297	108%	5 006	100%	2 964	100%
C7 (pupb)	2.12	100%	110	40%	5 004	100%	2 962	100%
C8 (pupb)	2.12	100%	268	97%	5 004	100%	2 962	100%

7. Táblázat: 15 korlátozásos Levy–3 feladat megoldásának eredményei a különböző intervallum-kiválasztási szabályokat használó algoritmusok esetén.

Szabály	CPU	rel.	MLL	rel.	NFE	rel.	NGE	rel.
C1	1.41		790		12 036		6998	
C2	1.48	105%	618	78%	12 672	105%	7 163	102%
C3 (pf)	0.23	16%	389	49%	1 866	16%	853	12%
C4 (pf)	1.50	106%	718	91%	12 582	105%	7 073	101%
C5 (pup)	0.21	15%	366	46%	1 982	16%	1 097	16%
C6 (pup)	1.53	109%	678	86%	12 582	105%	7 073	101%
C7 (pupb)	1.51	107%	206	26%	12 920	107%	7 367	105%
C8 (pupb)	1.56	111%	554	70%	12 920	107%	7 367	105%

8. Táblázat: 15 korlátozásos Ratz–4 feladat megoldásának eredményei a különböző intervallum-kiválasztási szabályokat használó algoritmusok esetén.

C3, mind a C5 szabállyal dolgozó algoritmus 45%-át használta a C1-re kapott CPU időnek, illetve 46-49%-ot eredményezett az utóbbi két mutatóban. A CPU, NFE és NGE mennyiségekben C1 és a többi eljárásváltozat között nem mutatható ki számottevő különbség. A tárigény tekintetében ellenben más a helyzet: az új C7 változat használta a legkevesebb memóriát (40% az alapmódszerhez képest), a második legjobb az ugyancsak új C5 eljárás (57%) volt. A leggyorsabb C3 és C5 közül tehát az utóbbi lényegesen kisebb, C3-hoz képest 33%-kal kevesebb tárral dolgozott.

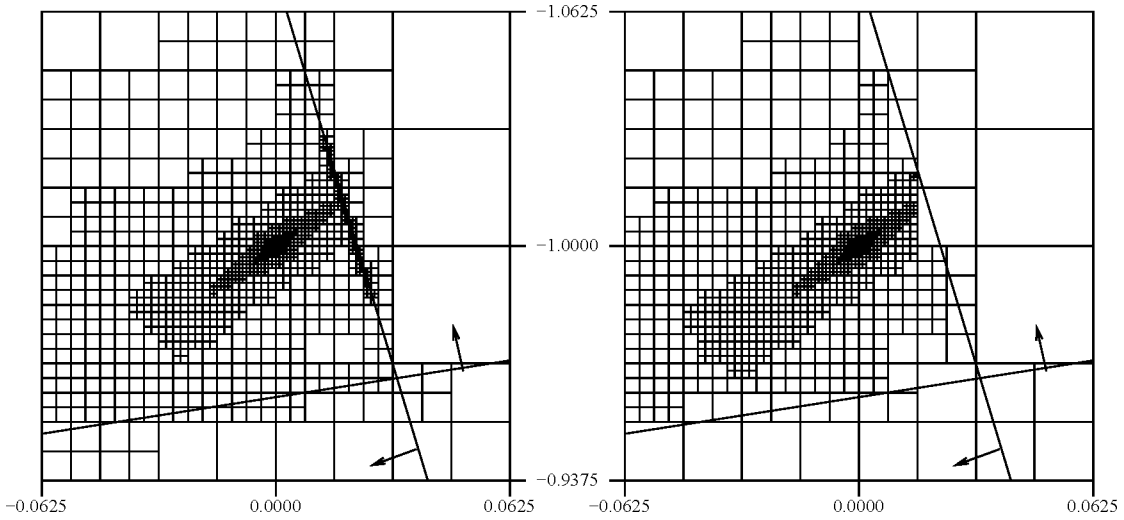
A Ratz–4 feladatokra a futási eredmények (8. Táblázat) sok tekintetben megegyeznek a Levy–3 feladatéval. Ismét az újonnan javasolt (nem-hibrid) C5 eljárás bizonyult a leghatékonyabbak, bár nem sokkal maradt el tőle a C3 változat sem: 15% és 16% az időigényben, 16-16%, illetve 16% és 12% az NFE illetve NGE mennyiségekben. E mutatók tekintetében a további változatok némileg gyengébbnek bizonyultak az alapmódszernél, a CPU időkben 5–11 százalékkal, a célfüggvény, illetve gradiens kiértékelések számában pedig 5–7, illetve 1–5 százalékkal. A tárfelhasználásban azonban a C2–C8 változatok mindegyike hatékonyabb volt C1-nél: ismét magasan a C7 eljárás volt a leggazdaságosabb (26%), ezt követte a C5-tel, majd a C3-mal jelzett módszer (46% és 49%).

Szabály	Sol.	CPU	rel.	MLL	rel.	NFE	rel.	NGE	rel.
C1	2	9 058.98		225 595		10 873 257		7 909 806	
C2	3	12 971.40	143%	589 317	261%	19 580 523	180%	8 677 571	110%
C3 (pf)	6	866.43	10%	81 422	36%	1 129 690	10%	318 501	4%
C4 (pf)	4	4 615.08	51%	115 571	51%	5 447 346	50%	3 960 092	50%
C5 (pup)	10	1 360.74	15%	60 383	27%	1 840 579	17%	909 877	12%
C6 (pup)	4	4 462.79	49%	115 372	51%	5 448 385	50%	3 961 576	50%
C7 (pupb)	9	2 857.86	32%	532 712	236%	4 361 676	40%	2 033 400	26%
C8 (pupb)	2	8 905.02	98%	225 595	100%	10 873 257	100%	7 909 806	100%

9. Táblázat: 15 korlátozásos EX–2 feladat megoldásának eredményei a különböző kiválasztási szabályok esetére. A táblázat a megoldott problémapéldányok számát (Sol.), a futási idők átlagát (CPU, másodpercben), a Munkalisták maximális hosszainak átlagát (MLL), valamint a célfüggvény- és gradienshívások átlagos számát (NFE, NGE) tartalmazza, a megoldott feladatokra vonatkoztatva. Az utolsó négy mutató esetén a C1 eljáráshoz viszonyított relatív mutatószámok (rel.) is fel lettek tüntetve.

Az alapul vett standard tesztfeladatok korlátozásos változatai között a legnehezebbek kétségtelenül az EX–2 feladat példányai voltak. A választott futási limittel a feladatokat számos esetben nem sikerült megoldani (a Munkalisták maximális hosszát 2 000 000-ban határoztam meg – ez a használt tárolási reprezentációval kb. 300 Mbyte helyfelhasználást jelent, míg időkorlátot nem állítottam fel). Az eredményeket a 9. Táblázat mutatja, melynek felépítése némiképp eltér a korábbiaktól: külön jeleztem a megoldott problémapéldányok számát (Sol.), illetve a megoldott különböző számú feladat miatt most minden mutatóra egy *átlagos* értéket adtam meg. Jegyezzük meg, hogy az így adódó összehasonlítás nem teljesen precíz: mindössze két feladatot oldott meg minden módszer, így a bemutatott értékek esetenként különböző (az aktuális módszerrel sikeresen megoldott) feladatokból származnak. A legfontosabb mutató mindenképpen a megoldott feladatok száma, melyben két új típusú intervallum-kiválasztási szabály, a C5 illetve C7 magasan a két legjobbnak bizonyult 10, illetve 9 megoldott feladattal. Őket a C3 szabállyal vezérelt algoritmusváltozat követi 6 feladattal, a további eljárások pedig 2–4 feladatot oldottak meg. A futási outputok áttanulmányozásakor kiderült, hogy 4 feladattal (ezek mindegyike 45 korlátozó feltételt tartalmazott) egyik módszer sem tudott a megadott memóriakorláton belül megbirkózni. Emellett mindössze egy esetben fordult elő az, hogy a javasolt C5 algoritmus sikertelen volt, míg valamely másik algoritmus variáns sikeresen megoldotta az illető feladatot.

Fentiek fényében a bemutatott további futási jellemzők kisebb hangsúlyt kapnak: ezekben mindenütt a C3 és C5 kiemelt hatékonyságát figyelhetjük meg. A CPU, NFE és NGE mutatókban a C3 5–8 százalékponttal volt jobb, ami ez esetben annyit tesz, hogy a megoldott feladatokon átlagosan ennyivel működött eredményesebben a pf típusú szabály. Nyilvánvalóan, volt azonban néhány (egészen pontosan 4) nehezebb feladat, amit C3 nem oldott meg, míg C5 igen, és ez utóbbiak bekerültek az összevetésbe. Az MLL mutatók azonban még így is C5 előnyét jelzik C3-mal szemben.



4. Ábra: A C4 (balra) és C6 (jobbra) szabályok által vezérelt algoritmusok működése a Goldstein–Price feladat egyik korlátozásos változatán, a globális minimumhely környezetében.

A további eljárásváltozatok közül C7-et kell még kiemelnünk, ami ugyan sok feladatot oldott meg, de – a reálisabb összevetést adó C5-tel szemben – jóval több memóriát használt. Emellett érdemes megfigyelni, hogy a C8 eljárás tulajdonképpen teljesen azonosan dolgozott az alapmódszerrel.

Az intervallum-kiválasztási szabályok numerikus vizsgálatának összefoglalásaként megállapíthatjuk, hogy a C5, C6, C7, C8-cal jelzett újonnan javasolt szabályok a korlátokból fakadó információ felhasználásával jelentősen megnövelhetik az algoritmus hatékonyságát. Ez a javulás a nehezen megoldható feladatokon, a klasszikus módszerekhez képest már az önmagukban igen hatékony $p(f_k, \boldsymbol{Z})$-típusú szabályokhoz viszonyítva is megfigyelhető (ilyenek a bemutatott korlátozásos Goldstein–Price, illetve Hartman–6 feladatok). Sőt, ahogyan azt az EX–2 feladat több problémapéldánya mutatja, számos esetben kizárólag a korlátokat figyelembe vevő kiválasztási szabályokkal rendelkező algoritmusokkal vált lehetővé a feladatok megoldása az egyébként igen magasra választott memóriakorláton belül.

Az új módszerekre kapott pozitív eredmények egy szemléletes példáját a 4. Ábra mutatja: az egyik korlátozásos Goldstein–Price problémára láthatjuk a C4 (balra) és C6 (jobbra) módszereket használó algoritmusok által generált intervallum-felosztásokat a globális minimumhely, $(0, -1)$ közelében. A kinagyított tartományban elhelyezkedő korlátok ugyancsak szerepelnek az ábrákon. Amint a bal oldali ábra mutatja, a korlátozó feltételeket a kiválasztás során figyelmen kívül hagyó $p(f_k, \boldsymbol{Z})$-típusú szabály rengeteg szükségtelen felosztást hajtott végre az egyik lineáris feltétel *határán* elhelyezkedő boxokra. Mindez amiatt történhetett meg, mert a C3 szabály ebben a környezetben (de már valahol a nem lehetséges megoldások tartományában) alacsony célfüggvényértéket 'sejtett', így a fenti boxokhoz magas $p(f_k, \boldsymbol{Z})$ értéket rendelt.

Ezek a szükségtelen felosztások elkerülhetővé váltak a $pup(f_k, \boldsymbol{Z})$ szabály alkalmazásával (jobb oldali ábra): ekkor a határon lévő boxokra ($pu(\boldsymbol{Z}) < 1$ miatt) a

pup értékek lecsökkentek $p(f_k, \boldsymbol{Z})$-hez képest, így mintegy eltaszítva az algoritmus által generált felosztásokat a feltétel határától a $(0, -1)$ körüli, szigorúan lehetséges megoldásokat tartalmazó tartományt preferálva.

A korlátozó feltételekből adódó információ felhasználásának előnye a vizsgált feladatokon egyrészt a felhasznált CPU idő (és az ezzel szoros kapcsolatban álló célfüggvény-hívásszám) tekintetében, főként pedig a lecsökkent tárigényben figyelhető meg. A Hartman–6, Goldstein–Price, Levy–3, Ratz–4 és EX–2 feladatok esetén (azaz a nehezebb problémáknál) a $pup(f_k, \boldsymbol{Z})$-t, vagy $pupb(f_k, \boldsymbol{Z})$-t használó módszerek e tekintetben jobbak voltak a $p(f_k, \boldsymbol{Z})$-típusú szabályoknál; ezen belül három feladatcsoport esetén a C7 szabály memória-felhasználása volt a legkedvezőbb.

Érdemes szem előtt tartanunk, hogy bár a vizsgált feladathalmaz korlátozott méretű, a kapott eredmények még így is nagy változatosságot mutatnak. Például az elhelyezési, Goldstein–Price, Levy–3, Ratz–4 és EX–2 feladatokon a nem-hibrid C3, C5 változatok hatékonyabbnak bizonyultak, mint a megfelelő hibrid C4, C6 variánsok, de a Hartman–6 feladatváltozatok esetén épp fordított volt a helyzet. Ez valószínűleg annak a jelenségnek tudható be, hogy a hibrid szabályok igen érzékenyek az N_m paraméter megadására. Az N_m értékek tanulmányozása korlátozásos feladatok esetén ezért egy fontos jövőbeli feladat lehet.

Ugyancsak érdemes a tekintett algoritmusok megállási feltételére (vagyis az első, optimumhely(ek) tartalmazásának szempontjából ígéretes box megtalálása utáni terminálásra) is felhívni a figyelmet. Ez a feltétel feltehetőleg az inkább mélységi jellegű keresést megvalósító heurisztikus szabályok számára kedvező. A garantáltan a minimumhoz tartó $\{f_k\}$ sorozatok [12] szerinti előállításával a jövőben fontos lehet egy erősebb megállási feltétel mellett is megvizsgálni a javasolt intervallum-kiválasztási szabályokat.

3.4.4 Az adaptív intervallum-felosztási szabályok vizsgálata

Néhány javasolt felosztási szabályt is megvizsgáltam mind az elhelyezési, mind a korlátozásos standard tesztfeladatok megoldása esetén. Minden egyes feladatot ötször oldottam meg a korábban ismertetett S1–S5 felosztási stratégiák (3.1 és 3.2 szakasz) használatával, azaz az alábbi szabályokkal:

- S1_/2: klasszikus bisection a leghosszabb irányra merőlegesen;
- S2_/4: felezés két koordináta irány mentén (/4 multisection);
- S3_/9: három egyforma részre darabolás két irányban (/9 multisection);
- S4_pf_/2,4,9: adaptív multisection a $p(\tilde{f}, \boldsymbol{X})$ érték alapján, a /2, /4 és /9 szabályok használatával ([6] alapján);
- S5_pup_/2,4,9: adaptív multisection, mint S4-ben, de ezúttal a $pup(\tilde{f}, \boldsymbol{X})$ értékek alapján.

Mind az öt esetben a klasszikus Moore–Skelboe intervallum-*kiválasztási* szabályt alkalmaztam (így a most S2_/4-gyel jelölt algoritmus az előző szakaszban C1-gyel

Feladat	S4 (pf)		S5 (pup)	
	P_1	P_2	P_1	P_2
fac. loc.	0.000000	0.278475	0.000000	0.199556
Shekel–10	0.001669	0.522521	0.000561	0.600309
Hartman–6	0.040979	0.576613	0.015952	0.998564
Goldstein–Price	0.138963	0.304043	0.002206	0.982160
Levy–3	0.019173	1.000000	0.000000	0.984546
Ratz–4	0.000000	0.438942	0.000000	0.411167

10. Táblázat: Az S4, illetve S5 intervallum-felosztási szabályokhoz javasolt P_1 és P_2 paraméterek.

jelölt módszerrel egyezik meg). A numerikus tesztek hardver-szoftver környezete, az alkalmazott adatszerkezetek és gyorsító lépések, valamint az algoritmus megállási feltétele ugyanaz volt, mint az intervallum-kiválasztási szabályok vizsgálatakor.

Az adaptív módszerekhez szükséges P_1 és P_2 értékeket a [3]-ban ismertetett eljáráshoz hasonló módon adtam meg: A 300 elhelyezési feladat közül 30-at, a korlátozásos standard feladatok közül minden típus esetén a 15 feladatból ötöt-ötöt választottam ki véletlenszerűen. P_1-et és P_2-t az SASS [59] sztochasztikus kereső segítségével határoztam meg, a kiválasztott feladatokat az S4 és S5 módszerekkel különböző P_1 és P_2 paraméterek használatával megoldva, és a célfüggvényhívások számát minimalizálva. A kapott eredményeket (melyeket a 10. Táblázat tartalmazza) használtam azután a megfelelő feladattípus összes problémájának megoldásakor.

A P_1 és P_2 ily módon történő előzetes meghatározása nyilvánvalóan nem alkalmazható akkor, amikor egy adott típusú feladatból csak egy példányt kell megoldanunk. Azonban pl. az ismertetett elhelyezési feladatok esetében, amiket az alkalmazások során esetleg sokszor oldunk meg ugyanolyan jellegű célfüggvénnyel, de különböző korlátokkal, a korábban megoldott feladatok tapasztalatai alapján javasolhatunk megfelelő P_1 és P_2 értékeket. Nehézségekbe ütközik a megfelelő paraméterek előállítása akkor is, ha a tekintett feladattípus egyes elemei túl nehezek ahhoz, hogy az SASS futtatása során akár több tízszer vagy százszor meg tudjunk őket oldani. Jelen esetben az EX–2 feladat esetén is ez volt a helyzet (lásd a C1 felosztási szabállyal kapott eredményeket az előző szakaszban). Emiatt az EX–2 változatokon nem vizsgáltam az egyes intervallum-felosztási szabályokat.

A futási eredményeket a 11. Táblázat foglalja össze. A táblázat felépítve megegyezik az intervallum-kiválasztási szabályoknál bemutatottakkal. A relatív mutatószámokat ezúttal a klasszikus statikusan felező (S1_/2) eljáráshoz viszonyítva képeztem.

A kapott eredmények szerint a CPU idő és a célfüggvény-számítások számában minden feladatcsoportnál a pf mutatót használó adaptív, S4-gyel jelzett módszer bizonyult a legjobbnak. Ez a vizsgált algoritmus beállítások esetében is alátámasztja a [3]-ban az S4 stratégiára kapott eredményeket. A *pup* mutatót használó S5 a

futásidők tekintetében 4–11 százalékponttal, a függvényhívások számában pedig 2–10 százalékponttal maradt el S4-től. Az egyetlen mutatószám, ahol S5 előnyét figyelhetjük meg S4-gyel szemben, az elhelyezési feladatokon kapott gradienshívások száma (113%, illetve 116%).

A *pup* típusú intervallum-felosztási szabályra kapott zömében negatív eredmények oka nagy valószínűséggel a P_1 és P_2 értékek beállítása. Elképzelhető például, hogy az SASS eljárás a pf szabályt alkalmazó módszerre ugyanakkora erőforrás-felhasználással megbízhatóbb eredményt ad, mint a *pup* stratégia esetén (az SASS algoritmus beállításai megegyeztek mindkét paraméterpár számításakor). Egy másik, az előzővel részben összefüggő ok az lehet, hogy a korlátok bevonásával kapott $pup(f_k, \boldsymbol{Z})$ mennyiségek feladatról feladatra változatosabb értékeket mutatnak a $p(f_k, \boldsymbol{Z})$ mennyiségeknél, így nehezebb általánosságban is jónak bizonyuló P_1, P_2 paramétereket előállítani. Ennek ellenére a korlátozó feltételek elhelyezkedését kiaknázó intervallum-felosztási szabályokat a jövőben érdemes lesz még részletesebben vizsgálni.

Szabály	CPU	rel.	MLL	rel.	NFE	rel.	NGE	rel.
facility location								
S1_/2	394.65		6 643		77 981		21 854	
S2_/4	289.15	73%	5 962	90%	56 639	73%	21 743	99%
S3_/9	321.08	81%	5 445	82%	62 748	80%	32 738	150%
S4_pf_/2,4,9	277.25	70%	5 728	86%	54 606	70%	25 385	116%
S5_pup_/2,4,9	291.79	74%	5 773	87%	55 892	72%	24 738	113%
Shekel–10								
S1_/2	3.69		35		3 108		1 519	
S2_/4	3.15	85%	31	89%	2 618	84%	1 534	101%
S3_/9	4.05	110%	22	63%	3 356	108%	2 336	154%
S4_pf_/2,4,9	3.00	81%	21	60%	2 561	82%	1 527	101%
S5_pup_/2,4,9	3.14	85%	31	89%	2 603	84%	1 536	101%
Hartman–6								
S1_/2	57 456.6		3 997 926		37 880 483		2 588 570	
S2_/4	35 022.2	61%	2 559 703	64%	25 293 057	67%	3 761 838	145%
S3_/9	69 117.6	120%	3 046 386	76%	49 158 875	130%	12 523 206	484%
S4_pf_/2,4,9	30 242.9	53%	2 376 171	59%	20 594 716	54%	2 713 321	105%
S5_pup_/2,4,9	33 652.3	59%	2 634 512	66%	24 103 404	64%	3 467 623	134%
Goldstein–Price								
S1_/2	108.26		28 983		578 040		362 759	
S2_/4	80.61	74%	24 188	83%	423 612	73%	323 000	89%
S3_/9	93.17	86%	18 584	64%	486 023	84%	421 350	116%
S4_pf_/2,4,9	76.30	70%	22 176	77%	397 732	69%	327 142	90%
S5_pup_/2,4,9	88.17	81%	25 114	87%	446 097	77%	340 832	94%
Levy–3								
S1_/2	2.47		235		6 252		2 827	
S2_/4	2.11	85%	276	117%	5 006	80%	2 964	105%
S3_/9	2.48	100%	263	112%	5 820	93%	4 151	147%
S4_pf_/2,4,9	2.05	83%	276	117%	4 890	78%	2 856	101%
S5_pup_/2,4,9	2.11	85%	276	117%	5 006	80%	2 964	105%
Ratz–4								
S1_/2	2.04		878		17 349		7 525	
S2_/4	1.41	69%	790	90%	12 036	69%	6 998	93%
S3_/9	1.57	77%	679	77%	12 733	73%	8 777	117%
S4_pf_/2,4,9	1.37	67%	671	76%	11 228	65%	7 237	96%
S5_pup_/2,4,9	1.55	76%	732	83%	12 190	70%	7 887	105%

11. Táblázat: Az intervallum-felosztási szabályok eredményei a vizsgált feladatok szerint csoportosítva. A relatív értékek ezúttal az S1_/2 módszerhez vannak viszonyítva.

4. Fejezet

Körpakolási feladatok megoldása

A geometriai ihletésű optimalizálási feladatok gyakorlati szempontból is fontos csoportját teszik ki az úgynevezett pakolási problémák. Ezek közös jellemzője, hogy geometriai alakzatokat kell elhelyeznünk a síkon (illetve a 3-dimenziós feladatok esetén a térben), vagy annak valamely részhalmazán úgy, hogy a kitöltés valamilyen szempontból optimális legyen. Ez az optimalitás vonatkozhat például az alakzatok számára, vagy az előálló kitöltés sűrűségére (azaz a kitöltött és ki nem töltött terület vagy térfogat nagyságának arányára). A probléma alkalmazási lehetőségeinek száma igen nagy, a legkézenfekvőbb alkalmazások a szabás, illetve raktározás területéről származnak, de pl. a 3. Fejezet 'facility location' feladatai is kapcsolatba hozhatók pakolási feladatokkal.

A pakolási feladatok egy történetileg is érdekes tagja az egybevágó körök zárt egységnégyzetbe történő elhelyezésének problémája a körök átmérőjének maximalizálásával. Az utóbbi 40 évben sokan és sokféleképpen próbálkoztak ezen feladatok megoldásával, az igazi áttörést azonban az jelentette, amikor a hagyományos "kézi" úton történő megoldási kísérletek helyét átvették a számítógépes megoldó algoritmusok ('computer-aided proofs'). Csak egy példa a módszerek fejlődésének érzékeltetésére: 10 kör optimális pakolásának problémája 1990-ig megoldatlan volt, ekkor született rá az első számítógépes módszeren alapuló megoldás, a hagyományos úton történő igazolás azonban máig nyitott probléma. Mára már több száz kör esetére is léteznek közelítő (de még nem bizonyítottan optimális), számítógéppel előállított megoldások. Ezzel megnő a szerepe a kapott eredmények numerikus megbízhatóságát vizsgáló, illetve önmagukban is megbízhatóan dolgozó optimalizáló eljárásoknak.

Ebben a fejezetben a feladatosztály alkalmas megfogalmazásának segítségével felépített garantált megbízhatóságú – intervallumos B&B típusú – optimalizálási algoritmusokat ismertetek. Az első algoritmus alkalmas a legtöbb már ismert optimális (vagy optimálisnak feltüntetett) megoldások nagy pontosságú lokális, illetve globális ellenőrzésére. Az algoritmus továbbfejlesztett változatának segítségével sikerült megoldanom az eddig nyitott, 28, 29 és 30 kör optimális pakolására vonatkozó feladatokat.

4.1 A körpakolási feladatok megfogalmazásai

Ebben a szakaszban a tekintett feladat szóbajöhető megfogalmazásait és a hozzájuk tartozó optimalizálási feladatok felírásait tekintem át $n \geq 2$ kör (pont) esetén. A

legkézenfekvőbb megfogalmazás az alábbi:

1. *Helyezzünk el n számú egybevágó kört az egységnégyzetben átfedés nélkül úgy, hogy a körök sugara maximális legyen.*

Ez a megközelítés az alábbi módon formalizálható: Legyen az egységnégyzet a $[0,1]^2$ halmaz, és jelöljük a körök középpontjait $((x_1,y_1),\dots,(x_n,y_n))$-nel (a továbbiakban a középpontok halmazát a tömörebb $(\boldsymbol{x},\boldsymbol{y})\in[0,1]^{2n}$ formában adom meg), a körök sugarát pedig r-rel. Az (x_i,y_i) és (x_j,y_j) pontok *távolságnégyzetét* jelöljük d_{ij}-vel, azaz $d_{ij}:=(x_i-x_j)^2+(y_i-y_j)^2$, $1\le i\ne j\le n$. (A $d_{ij}=d_{ji}$ azonosság miatt a gyakorlatban mindenütt elegendő az $1\le i<j\le n$ feltétel alkalmazása. Az $1\le i\ne j\le n$ feltétel egy később ismertetendő vizsgálat technikája miatt indokolt.) Ekkor a feladat a következő:

$$\begin{array}{lll} \max & r, & \\ \text{s.t.} & d_{ij}\ge(2r)^2, & 1\le i\ne j\le n;\\ & r\le x_i,y_i\le 1-r, & i=1,2,\dots,n;\\ & 0\le r. & \end{array} \tag{13}$$

Az első feltételcsoport a körök átfedésmentességét, a második pedig a köröknek az egységnégyzetben történő elhelyezését fejezi ki, ahogyan az az 5.a) Ábrán látható.

2. *Helyezzünk el n számú pontot az egységnégyzetben úgy, hogy a pontpárok közötti távolságok minimuma maximális legyen.*

A fenti jelölésekkel a 2. megfogalmazáshoz tartozó feladat az 5.b) Ábrán látható módon szemléltethető, maga az optimalizálási feladat pedig a következő alakot ölti:

$$\begin{array}{lll} \max & \min_{1\le i\ne j\le n}\sqrt{d_{ij}}, & \\ \text{s.t.} & 0\le x_i,y_i\le 1, & i=1,2,\dots,n. \end{array} \tag{14}$$

A gyakorlat alkalmazás szempontjából azok a megfogalmazások tekinthetők gyakoribbnak, amelyekben az elhelyezendő objektumok száma mellett azok mérete is adott, és a feladat a minél kisebb helyet igénylő pakolás megadása:

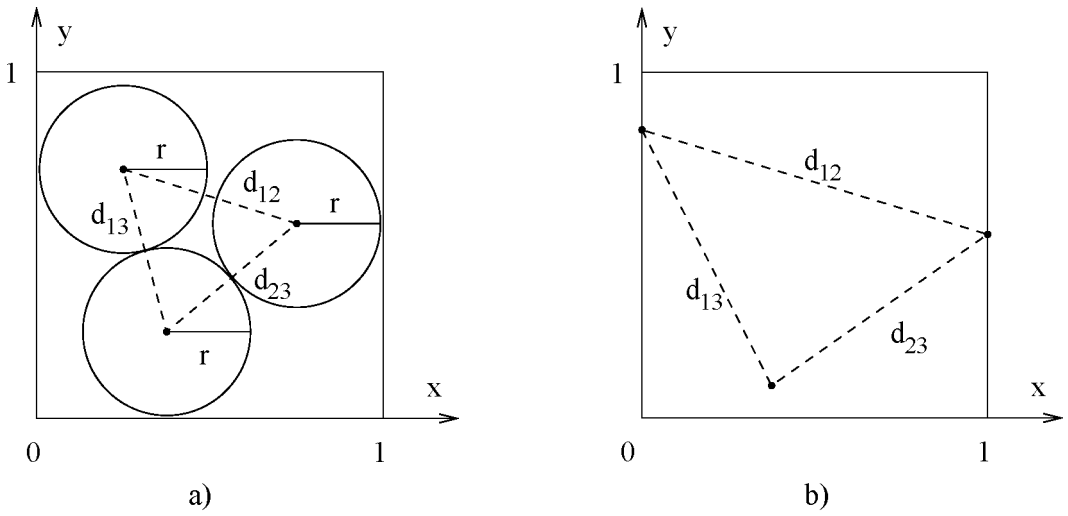
3. *Adjuk meg azt a legkisebb oldalhosszúságú négyzetet, amelyikben el tudunk helyezni n darab egységnyi sugarú kört átfedés nélkül.*

Az s oldalhosszúságú négyzet egyik csúcsát az origóba téve, magát a négyzetet pedig a pozitív síknegyedbe elhelyezve:

$$\begin{array}{lll} \min & s, & \\ \text{s.t.} & d_{ij}\ge(2r)^2=4, & 1\le i\ne j\le n;\\ & 1\le x_i,y_i\le s-1, & i=1,2,\dots,n;\\ & 0\le s. & \end{array} \tag{15}$$

4. *Adjuk meg azt a legkisebb oldalhosszúságú négyzetet, amelyikben el tudunk helyezni n pontot úgy, hogy bármely két pont távolsága legalább két egységnyi legyen.*

Az ehhez a feladathoz tartozó optimalizálási probléma:



5. Ábra: A körpakolási feladat 1. (a), illetve 2. (b) megfogalmazásának ábrázolása.

$$\begin{aligned} \min \quad & s, \\ \text{s.t.} \quad & d_{ij} \geq 2^2 = 4, \quad 1 \leq i \neq j \leq n; \\ & 0 \leq x_i, y_i \leq s, \quad i = 1, 2, \ldots, n; \\ & 0 \leq s. \end{aligned} \tag{16}$$

A bemutatott 1. és 2. megfogalmazásra egyszerűen megmutatható pl. [61] alapján a következő: a két feladat ekvivalens abban az értelemben, hogy az 1. feladat és 2. feladat optimális megoldásai kölcsönösen egyértelműen megfeleltethetők egymásnak. Továbbá, ha egy rögzített n esetén r_1^* jelöli az 1. feladat optimumát, akkor a megfelelő 2. megfogalmazáshoz tartozó feladat optimuma $d_2^* = 2r_1^*/(1 - 2r_1^*)$.

Mivel az 1. és 3., illetve 2. és 4. megfogalmazások nyilvánvalóan ugyanazt a problémát modellezik, a bemutatott négy különböző felírás tehát egymást helyettesítve használható. Vegyük észre, hogy egyrészt a pontok kezelése egyszerűbb a körök reprezentálásánál, másrészt, a 2. megfogalmazás kivételével minden probléma a változók korlátain kívül további, egyenlőtlenség alakú korlátozó feltételeket is tartalmaz. A legkönnyebben kezelhető megfogalmazás így a másodikként ismertetett. A továbbiakban tehát egységnégyzetbe történő pontpakolásokkal foglalkozom (azonban a szemléletesség kedvéért a feladatot gyakran továbbra is 'körpakolási feladat' néven említem).

(14) célfüggvényének kiszámítása a páronkénti távolságok miatt négyzetgyökvonásokat igényel. Mivel azonban az alkalmazások során rendkívül fontos a célfüggvény befoglalásának minél gyorsabb kiértékelése, a gyakorlatban a gyökvonások kiszámítását – a négyzetgyökfüggvény monotonitását kihasználva – megtakaríthatjuk:

$$\begin{aligned} \max \quad & \min_{1 \leq i \neq j \leq n} d_{ij}, \\ \text{s.t.} \quad & 0 \leq x_i, y_i \leq 1, \quad i = 1, 2, \ldots, n. \end{aligned} \tag{17}$$

A (14) és (17) feladatok optimális megoldásainak halmazai megegyeznek, az előbbi feladat optimuma pedig az utóbbi négyzetgyöke. Így a négyzetgyökvonást csak speciális helyzetekben (amikor valóban távolságértékekre van szükség) kell elvégeznünk. A

2. Algoritmus B&B globális optimalizálási algoritmus körpakolási feladatokhoz

Input:
- f: a célfüggvény,
- $\boldsymbol{Z}_0$: a keresési tartomány,
- ε: tolerancia érték a megállási feltételhez.

Output:
- *Maximum*: a globális maximum befoglalása,
- $\mathcal{L}_\mathcal{S}$ (Eredménylista): az összes globális maximumhely befoglalásait tartalmazó lista.

1: $\boldsymbol{Z} := \boldsymbol{Z}_0$; $\mathcal{L}_\mathcal{W} := \{(\boldsymbol{Z}, \overline{F}(\boldsymbol{Z}))\}$; inicializáljuk az $\tilde{f}$ kivágási értéket;
2: **while** ($\mathcal{L}_\mathcal{W}$ nemüres) **do**
3: válasszunk egy $(\boldsymbol{Z}, \overline{F}(\boldsymbol{Z}))$-t $\mathcal{L}_\mathcal{W}$-ből;
4: töröljük $(\boldsymbol{Z}, \overline{F}(\boldsymbol{Z}))$-t $\mathcal{L}_\mathcal{W}$-ből;
5: osszuk fel $\boldsymbol{Z}$-t az $\boldsymbol{U}^1$ és $\boldsymbol{U}^2$ boxokra;
6: **for** $i := 1$ **to** 2 **do**
7: aktualizáljuk (növeljük) $\tilde{f}$-t, ha lehetséges;
8: gyorsítótesztek végrehajtása $\boldsymbol{U}^i$-re;
9: **if** ($\boldsymbol{U}^i$ teljes egészében törölhető) **then** következő i;
10: jelölje $\hat{\boldsymbol{U}}^i \subseteq \boldsymbol{U}^i$ az $\boldsymbol{U}^i$-ből nem törölt tartományokat befoglaló boxot;
11: **if** $(w(F(\hat{\boldsymbol{U}}^i)) < \varepsilon)$ **then** tegyük $(\hat{\boldsymbol{U}}^i, \overline{F}(\hat{\boldsymbol{U}}^i))$-t az $\mathcal{L}_\mathcal{S}$-be;
12: **else** tegyük $(\hat{\boldsymbol{U}}^i, \overline{F}(\hat{\boldsymbol{U}}^i))$-t az $\mathcal{L}_\mathcal{W}$-be;
13: **end for**
14: **end while**
15: $Maximum := [\tilde{f}, \max\{\overline{F}(\boldsymbol{Z}) \mid (\boldsymbol{Z}, \overline{F}(\boldsymbol{Z})) \in \mathcal{L}_\mathcal{S}\}]$; **Return** *Maximum*, $\mathcal{L}_\mathcal{S}$;

vizsgált pontpakolási feladat teljes megoldása tehát (17) *összes* optimumhelyének és az optimum értékének megadásával ekvivalens.

12. MEGJEGYZÉS. *A probléma vizsgálatának kezdeti fázisában a negyedik, ugyancsak pontpakolási modellre épített eljárást implementáltam. Az eredmények jelentősen elmaradtak a jelenlegi algoritmusok eredményességétől; ennek oka épp a korlátozó feltételek kezelésének nehézkessége volt. Elképzelhető azonban, hogy a 3. Fejezetben ismertetett eredmények ígéretessége miatt érdemes lesz egy vizsgálat erejéig a 4. megfogalmazásra visszatérni.*

Az alkalmazott branch–and–bound eljárás prototípusát a 2. Algoritmus mutatja. Az eljárás lényegében a Bevezetésben ismertetett 1. Algoritmus maximalizálási feladatok megoldására alkalmas változata. (Az 1. Algoritmushoz képest a fő ciklus elejére került az intervallum-kiválasztási lépés; jelen fejezetben kényelmi szempontok miatt érdemes azt feltételeznünk, hogy minden iterációs ciklus végén az összes még nem törölt box vagy $\mathcal{L}_\mathcal{W}$-ben, vagy $\mathcal{L}_\mathcal{S}$-ben található.)

4.2 A négyzetbe történő körpakolás eddigi eredményei

A körpakolási problémák kutatásának kezdeti irányait az elméleti vizsgálatok jelentették: ez egyrészt adott n esetén n kör optimális pakolásainak ismertetését és

az optimalitás igazolását, illetve nehezebb esetekben az optimumértékre vonatkozó alsó és felső korlátok megadását jelenti. A számítógépes módszerek céljai hasonlóak, csupán az eljárás más. Ezen módszerek bizonyító erejének sarkalatos pontja – az alkalmazott algoritmusok korrektségének vizsgálata mellett – a pontosság és megbízhatóság kérdése. Az eddigi kutatások legfontosabb eredményei kronológiai sorrendben a következők:

- A feladatosztály felvetése – L. Moser (1960) [43].

- Optimalitás $n = 2, 3, 4, 5, 8, 9$-re elméleti úton – J. Schaer, A. Meir (1965) [56, 57].

- Optimalitás $n = 14, 16, 25, 36$-ra elméleti úton (a terület-eliminációs módszerek megjelenése) – K. Kirchner, G. Wengerodt (1983, 1987) [28, 64, 65, 66].

- Optimalitás 20 körig számítógépes eljárással (a terület-elimináció algoritmikus implementálása, csempézés) – R. Peikert, D. Würtz, M. Monagan, C. de Groot (1990, 1992) [21, 48].

- Optimalitás $n = 6$ esetén elméleti úton – H. Melissen (1994) [41].

- "Jó pakolások" számítógépes megadása (az optimalitás igazolása nélkül):

  $n = 30$-ig – nemlineáris optimalizáló eljárásokkal, C.D. Maranas, C.A. Floudas, P.M. Pardalos (1995) [35];

  $n = 52$-ig és $n = 54, 56, 60, 61, 62, 72, 78$-re – a 'billiárd-szimuláció' nevű eljárással és ismételt mintakitöltéssel, R.L. Graham, B.D. Lubachevsky (1996) [20];

  $n = 50$-ig – energiafüggvény-minimalizálással, K. Nurmela, P.R.J. Östergård (1997) [45].

- Optimalitás 27 körig számítógépes eljárással, Peikert és társai módszere alapján – K. Nurmela, P.R.J. Östergård (1999) [46, 47].

- "Jó pakolások" $n = 100$-ig a TAMSASS-PECS véletlenkereső algoritmussal – L.G. Casado, I. García, P.G. Szabó, T. Csendes (1999) [5].

- Az utóbbi időszakban: további megközelítések az ismert legjobb pakolások javítására és az optimalitás igazolására $n > 27$ esetén, pl.

  ε-optimális megoldások: az ismert legjobb pakolási értékek optimalitása az $\varepsilon = 10^{-5}$ tolerancia értéken belül (az összes optimális megoldás felderítése nélkül) $n = 28, \ldots, 35, 38, 39$-re, valamint az eddigi legjobb pakolás megadása $n = 37$-re – Locatelli és Raber (2002) [33].

Jegyezzük meg, hogy a számítógépes optimalitási bizonyítások mindegyike alapvetően hagyományos gépi aritmetikát használt. A megbízható eredmények előállítását célzó módszerek ezekben az algoritmusokban nehézkesek és az alkalmazott gépi aritmetika pontosságától függőek voltak, sőt, véleményem szerint helyenként nem biztosítottak minden szempontból kielégítő megoldást a kerekítési problémákra. Ezen

módszerekkel szemben célom egy teljes egészében intervallumos számításokon alapuló eljárás kidolgozása volt.

4.3 Intervallumos befoglaló függvény megadása

Az intervallumos eljárás felépítéséhez a legfontosabb teendő egy alkalmas befoglaló függvény konstruálása a körpakolási (illetve ez esetben pontpakolási) feladat célfüggvényére. Az (17) feladathoz tartozó célfüggvény tehát az alábbi:

$$f_n(\boldsymbol{x},\boldsymbol{y}) = \min_{1\le i\neq j\le n} (x_i - x_j)^2 + (y_i - y_j)^2 = \min_{1\le i\neq j\le n} d_{ij}, \tag{18}$$

amelyet a $[0,1]^{2n}$ kiinduló keresési tartományon kell *maximalizálnunk.*

Legyen adott egy, a számítások során előálló $2n$-dimenziós $(\boldsymbol{X},\boldsymbol{Y})$ box; ehhez keresünk olyan $F_n : \mathbb{I}^{2n} \to \mathbb{I}$ függvényt, hogy bármely $(\boldsymbol{x},\boldsymbol{y}) \in (\boldsymbol{X},\boldsymbol{Y})$-ra $f_n(\boldsymbol{x},\boldsymbol{y}) \in F_n(\boldsymbol{X},\boldsymbol{Y})$ teljesüljön, azaz F_n az f_n befoglaló függvénye legyen az $(\boldsymbol{X},\boldsymbol{Y})$ boxon. Az alábbi tétel egy ilyen tulajdonságokkal rendelkező és könnyen kiszámítható intervallumos függvényt ad meg.

24. TÉTEL. *[36] Legyen* $(\boldsymbol{X},\boldsymbol{Y}) \subseteq [0,1]^{2n}$, *és legyen*

$$D_{ij} = (X_i - X_j)^2 + (Y_i - Y_j)^2, \qquad \text{minden } 1 \le i \neq j \le n \text{ -re.}$$

Ekkor az $a := \min_{1\le i\neq j\le n} \underline{D_{ij}}$, $a \in \mathbb{R}$, *valamint a* $b := \min_{1\le i\neq j\le n} \overline{D_{ij}}$, $b \in \mathbb{R}$ *értékekkel definiált* $F_n(\boldsymbol{X},\boldsymbol{Y}) := [\,a,b\,]$ *intervallum az* $f_n(\boldsymbol{x},\boldsymbol{y})$ *célfüggvény értékkészletének befoglalását adja az* $(\boldsymbol{X},\boldsymbol{Y})$ $2n$*-dimenziós intervallumvektoron.*

BIZONYÍTÁS. Legyen $(x_1,\ldots,x_n,y_1,\ldots,y_n) = (\boldsymbol{x},\boldsymbol{y}) \in (\boldsymbol{X},\boldsymbol{Y}) \subseteq [0,1]^{2n}$ tetszőleges. Az intervallumokra kiterjesztett aritmetikai műveletek és elemi függvények befoglaló tulajdonságai (ld. 1. Fejezet) miatt $d_{ij} = d_{ij}(x_i,x_j,y_i,y_j) \in D_{ij}$ áll fenn minden $1 \le i \neq j \le n$ esetén, azaz

$$\underline{D_{ij}} \le d_{ij} \le \overline{D_{ij}}\ , \qquad \forall (1 \le i \neq j \le n). \tag{19}$$

Jelölje $(\tilde{i},\tilde{j})$ azt az indexpárt, melyre d_{ij} minimális. (Ha több ilyen pár van, akkor legyen közülük $(\tilde{i},\tilde{j})$ az egyik.) Ekkor a definíciója és (19) alapján a következő összefüggésekhez jutunk:

$$a = \min_{1\le i\neq j\le n} \underline{D_{ij}} \ \le\ \underline{D_{\tilde{i}\tilde{j}}} \ \le\ d_{\tilde{i}\tilde{j}} \ = \min_{1\le i\neq j\le n} d_{ij}. \tag{20}$$

Most jelölje $(\hat{i},\hat{j})$ azt az indexpárt, melyre $\overline{D_{ij}}$ minimális. (Ha több ilyen pár van, akkor legyen közülük $(\hat{i},\hat{j})$ az egyik.) Most (19)-t, majd b definícióját felhasználva kapjuk:

$$\min_{1\le i\neq j\le n} d_{ij} \ \le\ d_{\hat{i}\hat{j}} \ \le\ \overline{D_{\hat{i}\hat{j}}} \ = \min_{1\le i\neq j\le n} \overline{D_{ij}} = b. \tag{21}$$

Innen (20) és (21) két-két oldalát összevonva adódik, hogy

$$a \ \le \min_{1\le i\neq j\le n} d_{ij} \ \le\ b,$$

ami épp a befoglaló függvény helyességét igazoló összefüggés. □

A definiált befoglaló függvény legfontosabb tulajdonságai az alábbiakban foglalhatók össze:

1. *Az $F_n(\boldsymbol{X},\boldsymbol{Y})$ függvény kiszámítása (hasonlóan $f_n(\boldsymbol{x},\boldsymbol{y})$ kiértékeléséhez) $\Theta(n^2)$ műveletigényű* (amennyiben nincs előzetes információnk az egyes $((X_i,Y_i),(X_j,Y_j))$ komponenspárok relatív elhelyezkedéséről): a D_{ij} értékeket ugyanis *minden* párra ki kell számítanunk. Az egyes komponenspárok esetén az intervallumos műveletvégzések száma konstansszorosa a valós típusú műveletvégzésnek (az intervallumos műveletek és standard függvények definíciói alapján), az a és b értékek képzése pedig $\Theta(n)$ összehasonlítással elvégezhető.

2. *$F_n(\boldsymbol{X},\boldsymbol{Y})$ az $n=2$ esetben pontos* (az intervallumos műveletvégzés kifelé kerekítéseitől eltekintve), hiszen ekkor egy D_{ij} értéket kell kiszámítanunk ($D_{12}=D_{21}$ felhasználásával), azaz $[a,b]=D_{12}=(X_1-X_2)^2+(Y_1-Y_2)^2$, melynek értéke az 1. Tétel értelmében nem eredményez túlbecslést.

$n\geq 3$ esetén viszont $F_n(\boldsymbol{X},\boldsymbol{Y})$ csupán az alsó korlátjában pontos, a felső korlátban túlbecslést eredményezhet: $\underline{F_n}(\boldsymbol{X},\boldsymbol{Y})=\min_{1\leq i\neq j\leq n}\underline{D_{ij}}$ valóban minden esetben az (i,j) párokhoz tartozó d_{ij} távolságnégyzetek minimuma. Ami F_n felső korlátját illeti, tekintsük az $n=3$ esetben az $(X_1,Y_1)=([0,0.5],[0,0.5])$, $(X_2,Y_2)=([0,0.5],[0.5,1])$, $(X_3,Y_3)=([0.5,1],[0,0.5])$ boxot, ami tehát a $[0,1]^2$-ben elhelyezett 3 téglalapot jelent. Egyszerű számítással $\overline{F_3}(\boldsymbol{X},\boldsymbol{Y})=1.25$, ugyanakkor az $f_3(\boldsymbol{X},\boldsymbol{Y})$ értékkészlet felső korlátja – ami nem más, mint a 3 pont optimális pakolásához tartozó, a (17) feladat szerinti optimumérték – ≈ 1.072. A feladat nehézségének tükrében nyilván nem meglepő a felső korlátban kapott túlbecslés: ha $\overline{f_n}(\boldsymbol{X},\boldsymbol{Y})$-t $\overline{F_n}(\boldsymbol{X},\boldsymbol{Y})$ kiszámításával pontosan meg tudnánk adni, a feladat egyetlen intervallumos függvénykiszámítással megoldhatóvá válna.

3. *$F_n(\boldsymbol{X},\boldsymbol{Y})$ teljesíti a befoglaló függvényektől elvárt egyik jó tulajdonságot, az izotonitást.* Tekintsük ugyanis az $(\boldsymbol{X}^{(1)},\boldsymbol{Y}^{(1)}),(\boldsymbol{X}^{(2)},\boldsymbol{Y}^{(2)})\subseteq[0,1]^{2n}$ boxokat, és legyen $(\boldsymbol{X}^{(2)},\boldsymbol{Y}^{(2)})\subseteq(\boldsymbol{X}^{(1)},\boldsymbol{Y}^{(1)})$. A D_{ij} kifejezéseket természetes intervallumkiterjesztéssel kapjuk, így azok az 1.4 szakaszban leírtak miatt izotonok, vagyis a két boxra számított intervallumos távolságnégyzetekre $D_{ij}^{(2)}\subseteq D_{ij}^{(1)}$ minden $1\leq i\neq j\leq n$-re. Emiatt $\underline{D_{ij}^{(1)}}\leq\underline{D_{ij}^{(2)}}$ minden $1\leq i\neq j\leq n$-re és így

$$\underline{F_n}(\boldsymbol{X}^{(1)},\boldsymbol{Y}^{(1)})=\min_{1\leq i\neq j\leq n}\underline{D_{ij}^{(1)}}\leq\min_{1\leq i\neq j\leq n}\underline{D_{ij}^{(2)}}=\underline{F_n}(\boldsymbol{X}^{(2)},\boldsymbol{Y}^{(2)}).$$

Hasonló számítással a felső korlátokra $\overline{F_n}(\boldsymbol{X}^{(1)},\boldsymbol{Y}^{(1)})\geq\overline{F_n}(\boldsymbol{X}^{(2)},\boldsymbol{Y}^{(2)})$ adódik, azaz $F_n(\boldsymbol{X}^{(2)},\boldsymbol{Y}^{(2)})\subseteq F_n(\boldsymbol{X}^{(1)},\boldsymbol{Y}^{(1)})$.

4. *$F_n(\boldsymbol{X},\boldsymbol{Y})$ teljesíti a befoglaló függvényektől elvárt másik jó tulajdonságot, a zérókonvergenciát.* Tekintsük az $(\boldsymbol{X},\boldsymbol{Y})\subseteq[0,1]^{2n}$ boxok olyan sorozatát, melyre $w(\boldsymbol{X},\boldsymbol{Y})\to 0$. Ez azt jelenti, hogy minden komponensre $w(X_i)\to 0$, illetve $w(Y_i)\to 0$ teljesül. A D_{ij} függvények zéró-konvergenciája (1.4 szakasz) miatt

$w(D_{ij}) \to 0$, $\forall 1 \leq i \neq j \leq n$, azaz a $\{D_{ij}\}$ intervallumsorozatok mindegyike egy-egy d^*_{ij} ponthoz tart. Emiatt pedig

$$\min_{1\leq i\neq j\leq n} \underline{D_{ij}} \to \min_{1\leq i\neq j\leq n} d^*_{ij}, \quad \text{valamint} \quad \min_{1\leq i\neq j\leq n} \overline{D_{ij}} \to \min_{1\leq i\neq j\leq n} d^*_{ij},$$

azaz a vizsgált boxsorozatra $w(F_n(\boldsymbol{X}, \boldsymbol{Y})) \to 0$ teljesül.

4.4 A kifejlesztett algoritmus első változata

Az alábbi szakaszokban áttekintem az optimalizáló kereteljárás további részeit. Elsőként monotonitási tulajdonságokon alapuló gyorsító eljárásokat, majd egy könnyen implementálható, ún. *terület-eliminációs* gyorsítótesztet ismertetek. Végül a keresés globális fázisának kérdéseit veszem szemügyre.

4.4.1 A célfüggvény monotonitási tulajdonságai

Az intervallumos optimalizáló eljárások talán leghatékonyabb gyorsítótesztje az ún. monotonitási vizsgálat. Amint azt a Bevezetésben ismertettem, folytonosan differenciálható célfüggvények esetén a gradiensek befoglalásának számításai (pl. [9, 22, 27, 39]) jó módszert kínálnak ennek a tesztnek a végrehajtására. A körpakolási feladatra felírt $f_n(\boldsymbol{x}, \boldsymbol{y})$ célfüggvényről azonban könnyen látható, hogy nem differenciálható (így nem is folytonosan differenciálható) mindenütt a $[0,1]^{2n}$-n; a min() függvény ugyanis nem bír ezzel a tulajdonsággal. (A célfüggvény számításakor használt függvények, így a min() függvény folytonossága miatt a folytonosság tulajdonsága viszont fennáll a teljes keresési térben.) Mivel a célfüggvény nyilvánvalóan monoton bizonyos tartományokban, így a monotonitást valamilyen eszköztár segítségével mindenképpen érdemes lenne kihasználnunk. Ezen eszközök megtalálásához jó kiindulópontot kínálnak a feladat geometriai jellemzői. Az alábbi tétel egy új és igen egyszerűen elvégezhető monotonitási tesztet ismertet, mely nem igényel gradiensszámítást:

25. TÉTEL. *[36] Legyen $(\boldsymbol{X}, \boldsymbol{Y}) \subseteq [0,1]^{2n}$ tetszőleges $2n$-dimenziós box. Ha valamely k indexre ($k \in \{1, 2, \ldots, n\}$) fennáll, hogy minden $j(\neq k)$ esetén vagy*

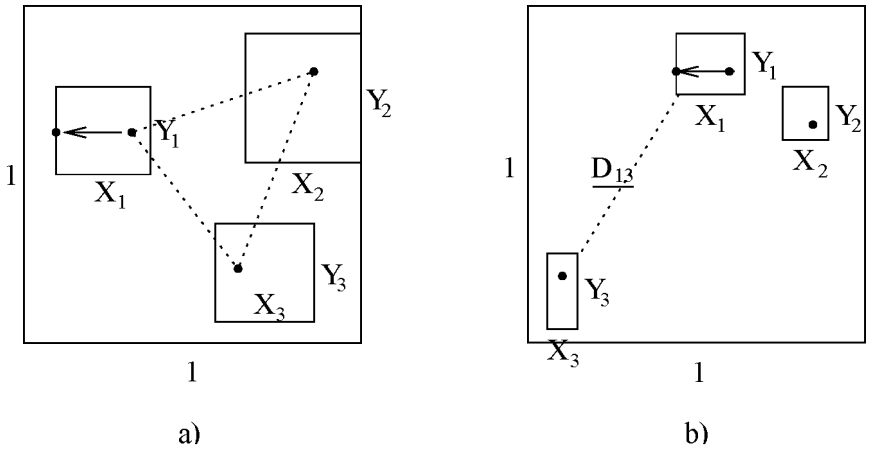
$$\overline{X_k} \leq \underline{X_j}, \quad \textit{vagy pedig} \tag{22}$$

$$\underline{D_{kj}} > \overline{F_n}(\boldsymbol{X}, \boldsymbol{Y}) \tag{23}$$

teljesül, akkor f_n monoton csökkenő $(\boldsymbol{X}, \boldsymbol{Y})$-n az x_k változójában. Ekkor az X_k-t $[\underline{X_k}, \underline{X_k}]$-ra módosíthatjuk.

BIZONYÍTÁS. Tekintsük a 6. Ábra két részét, mely egy-egy tipikus esetet mutat az $n = 3$-ra vonatkozó probléma megoldása során, a $k = 1$ választás mellett. A boxok itt a korábbiakat követve az X_i és Y_i intervallumokkal vannak megadva; mindhárom téglalapból egy-egy pontot kiválasztva jutunk egy lehetséges megoldáshoz.

A következőkben megmutatjuk, hogy a tétel feltételeinek teljesülése esetén az $f_n(\boldsymbol{x}, \boldsymbol{y})$ célfüggvény értéke nem csökkenhet x_k X_k-ban történő csökkentésével.



6. Ábra: Monotonitási vizsgálatok.

(i) Először vegyük észre, hogy minden olyan j, $j \in \{1, 2, \ldots, n\}$, $j \neq k$ indexre, melyre (22) teljesül, az (x_j, y_j) és (x_k, y_k) pontok közötti távolság nem csökken, ha x_k-t csökkentjük X_k-n belül. A 6. Ábra a) részén ez a helyzet $j = 2, 3$, a b) részen pedig $j = 2$ esetén.

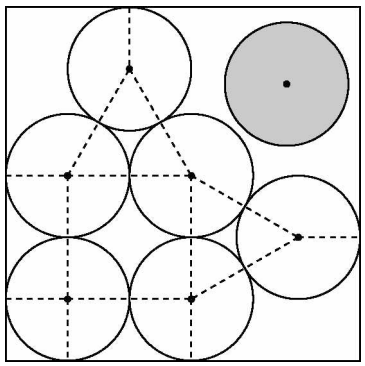
(ii) Azon j indexek esetén, melyek nem teljesítik (22)-t, az (x_j, y_j) és (x_k, y_k) közötti távolság csökkenhet $x_k \in X_k$ csökkenésével; lásd $j = 3$-t a 6. Ábra b) részén. Az új távolság négyzete azonban mindenképpen legalább $\underline{D_{kj}}$ kell legyen, ami viszont (23) miatt nagyobb, mint a boxon számított célfüggvényérték. Más szavakkal, az ilyen j és k indexű téglalapokból választott pontok távolságnégyzetei kívül esnek a célfüggvény befoglalásán, azaz a d_{kj} távolságnégyzetek változásának nem lesz hatása az $f_n(\boldsymbol{x}, \boldsymbol{y})$ célfüggvény értékére.

(iii) Azon pontpárok távolsága, melyek egyik tagja sem a k indexű téglalapból választott, nyilván változatlan marad x_k módosításával. □

Ha tehát a fentiek alapján egy boxon a célfüggvény monoton valamely változójában, akkor a kérdéses változót tartalmazó intervallumot egy ponttá, pontosabban egy 0 szélességű intervallummá szűkíthetjük le, a 6. Ábrán látható esetekben például az X_1-t $\underline{X_1}$-re.

13. MEGJEGYZÉS. *Mivel az értelmezési tartományok határai, a* 0 *és az* 1 *lebegőpontos módon pontosan ábrázolhatók, valamint az algoritmus futása során előálló összes intervallum határa is gépi szám, ezért ez a módszer a gyakorlatban is pontosan, és az intervallumos algoritmusokra jellemző megbízhatósággal fog működni.*

14. MEGJEGYZÉS. *Az ismertetett módszer valójában a célfüggvény monotonitását, és nem* szigorú *monotonitását teszteli, így használatakor nem feltétlenül kapjuk meg az összes optimumhelyet, ha a célfüggvény az optimumát egy végtelen halmazon veszi fel (lásd a következő szakaszt); helyette ebből a halmazból pusztán egy vagy több reprezentánst szolgáltat a módszer. Fontos megjegyezni, hogy legalább egyet viszont mindenképpen fog, azaz a fenti gyorsító módszer az optimumérték megadását illetően helyesen viselkedik.*



7. Ábra: 7 kör optimális elhelyezése.

A célfüggvény monoton növekedésének vizsgálata hasonlóképpen történik:

26. TÉTEL. *[36] Legyen* $(\boldsymbol{X},\boldsymbol{Y}) \subseteq [0,1]^{2n}$ *tetszőleges* $2n$*-dimenziós box. Ha valamely* k *indexre* $(k \in \{1,2,\ldots,n\})$ *fennáll, hogy minden* $j(\neq k)$ *esetén vagy*

$$\underline{X_k} \geq \overline{X_j}, \quad \textit{vagy pedig} \tag{24}$$

$$\underline{D_{kj}} > \overline{F_n}(\boldsymbol{X},\boldsymbol{Y}) \tag{25}$$

teljesül, akkor f_n *monoton növekvő* $(\boldsymbol{X},\boldsymbol{Y})$*-n az* x_k *változóban. Ekkor* X_k*-t* $[\overline{X_k},\overline{X_k}]$*-ra módosíthatjuk.*

Természetesen $\boldsymbol{X}$ és $\boldsymbol{Y}$ szerepének felcserélésével az y_k változókra is igazak a fenti tételek, ezért az algoritmusban a monotonitási vizsgálatot az összes változóra elvégezhetjük, mind a monoton növekedés, mind a monoton csökkenés tesztelésével.

A bemutatott monotonitási teszt egyszerű és gyors végrehajtása mellett érdemes kiemelni annak hatékonyságát is. Az optimális körpakolások esetén több kör érinti az egységnégyzet oldalát (illetve pontpakolások esetén több pont az oldalakon helyezkedik el). Tapasztalataim szerint az intervallumos monotonitási teszt alkalmazásával ezek az esetek gyorsan felderíthetők, ezzel a keresési tartomány mérete már a kezdeti iterációs lépésekben jelentősen redukálható.

4.4.2 A "szabad körök" kezelésének módszere

Tekintsük a 7. Ábrát, mely 7 kör optimális elhelyezését mutatja ([46, 61]). Az ábrán megfigyelhető, hogy mely körök érintik egymást, illetve az egységnégyzet oldalát. Vegyük észre, hogy bár a megoldás optimális, csak hat kör helye rögzített pontosan, a hetedik szabadon mozoghat (a megegyező struktúrájú optimális pontpakolásban is ugyanez a helyzet: hat pont helye rögzített, egy pedig "szabad"). Az optimális pontpakolásokban található "szabad" pontokat az alábbi módon definiálhatjuk:

12. DEFINÍCIÓ. *Tekintsük az (17) által adott pontpakolási feladatot valamely rögzített* $n \geq 2$ *esetén. Tekintsük továbbá az* $(\boldsymbol{x},\boldsymbol{y}) = (x_1,\ldots,x_n,y_1,\ldots,y_n)$ optimális *pontpakolást, amelyre tehát* f_n *maximális. A* $p_k = (x_k,y_k)$, $k \in \{1,\ldots,n\}$ *pontot az*

$(\boldsymbol{x}, \boldsymbol{y})$ pakolás szabad pontjának *nevezzük, ha létezik olyan p_k végpontú H félegyenes, és létezik valamely pozitív ε valós szám, hogy*

$$f_n(\boldsymbol{x}, \boldsymbol{y}) = f_n(x_1, \ldots, x'_k, \ldots, x_n, y_1, \ldots, y'_k, \ldots, y_n)$$

teljesül minden $(x'_k, y'_k) \in H \cap \Delta_\varepsilon(p_k)$ esetén, ahol $\Delta_\varepsilon(p_k)$ jelöli a p_k pont ε sugarú környezetét.

A kör- és pontpakolási feladatok ekvivalenciájából könnyen látható, hogy egy optimális pontpakolásban található szabad pontok a megfelelő optimális körpakolásban olyan körök középpontjait adják meg, amelyek valamely irányban elmozdulhatnak a körök átfedésmentességének sérülése nélkül. Mivel a szabad pontok egy-egy végtelen (és általában pozitív mértékű) halmaz tetszőleges elemei lehetnek, ésszerű gondolat az ilyen halmazokat egy-egy pontjukkal reprezentálni, és így kontinuum sok, tulajdonképpen ekvivalens optimális megoldást egyszerűbben kezelni.

Egyszerű észrevétel, hogy a szabad körök kezelése a monotonitás egy speciális esete, hiszen a p_k szabad pontot a $H \cap \Delta_\varepsilon(p_k)$ halmazon mozgatva a célfüggvény konstans. Amennyiben p_k-t az aktuálisan vizsgált $(\boldsymbol{X}, \boldsymbol{Y})$ boxból származó (X_k, Y_k) téglalapon mozgatva is konstans a célfüggvény, akkor egy a monotonitási teszthez hasonló vizsgálatot alkalmazhatunk.

Eddigiek összefoglalásaként megfogalmazható az alábbi tétel, mely a szabad pontokat tartalmazó boxok létezésére vonatkozóan ad meg egy elegendő feltételt:

27. TÉTEL. *[36] alapján. Legyen $(\boldsymbol{X}, \boldsymbol{Y}) \subseteq [0,1]^{2n}$ tetszőleges $2n$-dimenziós box. Ha valamely k indexre $(k \in \{1, 2, \ldots, n\})$ fennáll, hogy minden $j(\neq k)$ esetén*

$$\underline{D_{kj}} > \overline{F_n}(\boldsymbol{X}, \boldsymbol{Y}),$$

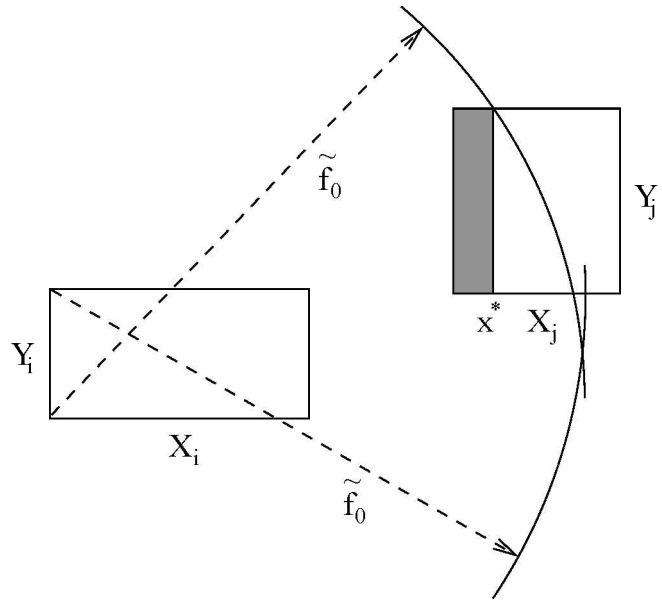
akkor az $(\boldsymbol{X}, \boldsymbol{Y})$-ben esetlegesen létező összes optimális $(\boldsymbol{x}, \boldsymbol{y})$ pakolás esetén a pakolás (x_k, y_k) pontja szabad, és ez a pont szabadon mozoghat (X_k, Y_k)-ban. Ezért a box további vizsgálatához elegendő az (X_k, Y_k) egy pontjának, valamint az '(X_k, Y_k) kizárólag szabad pontokat tartalmaz' információnak a tárolása.

BIZONYÍTÁS. Legyen $(\boldsymbol{x}, \boldsymbol{y}) \in (\boldsymbol{X}, \boldsymbol{Y})$ egy optimális pontpakolás. Ekkor $f_n(\boldsymbol{x}, \boldsymbol{y}) \leq \overline{F_n}(\boldsymbol{X}, \boldsymbol{Y}) < \underline{D_{kj}}$. Az (x_k, y_k) elhelyezkedése (X_k, Y_k)-n belül tehát nem befolyásolja a pakoláshoz tartozó célfüggvényértéket, vagyis

$$f_n(\boldsymbol{x}, \boldsymbol{y}) = f_n(x_1, \ldots, x'_k, \ldots, x_n, y_1, \ldots, y'_k, \ldots, y_n)$$

minden $(x'_k, y'_k) \in (X_k, Y_k)$ esetén. □

A monotonitási teszthez hasonlóan a szabad körök felderítése is az eljárás minél korábbi fázisában szükséges. A probléma azonban az, hogy azok a területek, melyeken valamely pont szabadon mozoghat a célfüggvény értékének megváltozása nélkül, sokszor nem reprezentálhatók intervallumokkal. Ilyen például a 7. Ábra alapján 7 pont optimális pakolásakor a szabad pont mozgási tartománya, mely a négyzet két oldala, illetve a szabad pont 'szomszédságában' lévő pontokból írt körívek által meghatározott zárt, konkáv síkidom. A probléma egy lehetséges megoldása az algoritmus egyes lépéseiben a még megvizsgálandó (tehát nem törölt) tartományok speciális kezelését vetíti előre, amely a következő szakasz tárgya. A szabad pontok kezelésének egy kifinomultabb módszerére a 4.5.1 szakaszban még visszatérünk.



8. Ábra: Az aktív pontok módszere.

4.4.3 Az aktív pontok módszere

Szintén a feladat geometriai interpretációján alapul az ebben a szakaszban ismertetett módszer, melyre a továbbiakban az 'aktív pontok módszere' néven hivatkozom. Az eljárás először Kirchner és Wengerodt [28, 64, 65, 66] elméleti jellegű bizonyításaiban jelent meg, és ezt alkalmazták az eddigi meghatározó számítógépes megoldó eljárásokban is [46, 47, 48]. A módszer alapján kidolgozott garantált megbízhatóságú eljárások bizonyultak a kifejlesztett algoritmusok leghatékonyabb gyorsítótesztjének. A módszer alapelve a következő:

1. Tegyük fel, hogy a (14) feladat maximumára ismert egy $\tilde{f}_0$ alsó korlát (pl. egy optimumhoz közeli, ún. 'jó' pakoláshoz tartozó verifikált érték). Az aktuálisan vizsgált keresési tartományt – ez az intervallumos módszerekben egy $(\boldsymbol{X}, \boldsymbol{Y}) \subseteq [0,1]^{2n}$ box – jelölje $(A_1, A_2, \ldots, A_n)$, ahol az i-edik pakolandó pontot az A_i tartományból választhatjuk. Jelen algoritmusban tehát A_i egy $(X_i, Y_i) \subseteq [0,1]^2$ téglalappal egyezik meg.

2. Minden A_i tartományból töröljük azokat a pontokat, amelyek valamely másik, A_j $(j \neq i)$ halmaz *összes* pontjától vett távolsága kisebb, mint $\tilde{f}_0$.

3. A nem törölt, aktuálisan még aktív területeket tovább használhatjuk a többi tartományból való törlésre.

4. Ha valamelyik tartomány aktív területe üressé válik, az azt jelenti, hogy a kiinduló $(\boldsymbol{X}, \boldsymbol{Y})$ box kizárólag olyan $(\boldsymbol{x}, \boldsymbol{y})$ pontpakolásokat tartalmaz, amelyre $f_n(\boldsymbol{x}, \boldsymbol{y}) < \tilde{f}_0$. Ekkor a B&B algoritmus törölheti $(\boldsymbol{X}, \boldsymbol{Y})$-t.

Ahhoz, hogy a módszert az intervallumos korlátozás és szétválasztás típusú algoritmusba integrálni tudjuk, szükséges tehát a redukálás során az aktív területeknek egy

speciális adatszerkezetben történő tárolása. A redukáló lépést minden (i,j), $1 \leq i \neq j \leq n$ párra általában többször is érdemes elvégezni. A redukálások után megmaradó aktív területeket (ha egy ponthoz tartozó terület sem ürült ki) a terület legszűkebb befoglalását adó téglalapokká kell visszaalakítani, a gyorsító eljárás kimenete ugyanis esetünkben valamilyen *box* kell legyen. Egy intervallumokkal dolgozó algoritmus számára az aktív pontok módszerének magjához tehát az alábbi feladatot kell megoldanunk megbízható módszerekkel (8. Ábra):

Tekintsük az (X_i, Y_i), $(X_j, Y_j) \subseteq [0,1]^2$ *kétdimenziós boxokat. Tegyük fel, hogy (i)* (X_i, Y_i)*-nek és* (X_j, Y_j)*-nek nincs közös belső pontja, továbbá (ii)* $\overline{X_i} \leq \underline{X_j}$*. Keressük azon* legnagyobb x^* *valós számot, melyre az alábbi áll:*

$$(X_i - [\underline{X_j}, x^*])^2 + (Y_i - Y_j)^2 \leq \tilde{f}_0^2, \tag{26}$$

azaz bármely $([\underline{X_j}, x^*], Y_j)$ *és* (X_i, Y_i)*-beli pontpár távolsága legfeljebb* $\tilde{f}_0$*. Ekkor az* $([\underline{X_j}, x^*], Y_j)$ *törölhető terület lesz.* (A 8. Ábrán ez a satírozott rész.)

Jegyezzük meg, hogy a (26) egyenlőtlenségben megengedett az egyenlőség is, hiszen a törölt terület x komponensének felső korlátja megegyezik a megmaradó aktív terület x komponensének alsó korlátjával, azaz mind a törölt, mind az aktív terület zárt. Ugyanakkor fontos látnunk azt, hogy a megoldandó feladat két feltétele nem jelent semmilyen újabb megszorítást: az (i) feltétel elérését a 4.4.5 szakasz 'csempéző' eljárása fogja garantálni. Ekkor pedig (ii) is biztosítható, esetlegesen a koordináta irányok felcserélésével. Az utóbbi feltétel a két téglalap relatív pozíciójának rögzítéséhez szükséges.

A 8. Ábráról az is egyszerűen leolvasható, hogy általános helyzetben a törölhető területet az $(\underline{X_i}, \underline{Y_i})$, illetve $(\underline{X_i}, \overline{Y_i})$ pontokból húzott köríveknek az $y = \overline{Y_j}$, illetve az $y = \underline{Y_j}$ egyenesekkel vett metszéspontjai határozzák meg (amennyiben léteznek ilyen metszéspontok). Ez a tény az alábbi, az irodalomban gyakran említett egyszerű állításon alapul:

2. LEMMA. *Nurmela és Östergård [47]: Ha egy pont kisebb, mint* $\tilde{f}_0$ *távolságra van egy téglalap (általános esetben egy poligon) minden csúcspontjától, akkor az* $\tilde{f}_0$*-nál kisebb távolságra van a téglalap (poligon)* összes *pontjától.*

Esetünkben tehát az

$$(\underline{X_i} - x^*)^2 + (\underline{Y_i} - \overline{Y_j})^2 = \tilde{f}_0^2, \text{ illetve az} \tag{27}$$

$$(\underline{X_i} - x^*)^2 + (\overline{Y_i} - \underline{Y_j})^2 = \tilde{f}_0^2 \tag{28}$$

egyenlet $x^* \geq \underline{X_i}$ feltételt kielégítő megoldásai (melyet jelöljünk x_1^*-gal az első és x_2^*-gal a második egyenlet esetén) közül keressük a kisebbet.

Pontos számításokat feltételezve x_1^* az alábbiak szerint számítható ki: (27)-ben, ha $\tilde{f}_0^2 - (\underline{Y_i} - \overline{Y_j})^2 < 0$, akkor x_1^* nem létezik, ellenkező esetben $x_1^* \geq \underline{X_i}$ miatt

$$x_1^* = \sqrt{\tilde{f}_0^2 - (\underline{Y_i} - \overline{Y_j})^2} + \underline{X_i}.$$

Hasonlóan, ha x_2^* létezik, akkor

$$x_2^* = \sqrt{\tilde{f}_0^2 - (\overline{Y_i} - \underline{Y_j})^2} + \underline{X_i}.$$

Azaz, ha x_1^* és x_2^* közül egyik sem létezik, akkor a *teljes* (X_j, Y_j) törölhető, ha pontosan egy x_k^* létezik, akkor x^* ezzel lesz egyenlő, egyébként pedig $x^* := \min(x_1^*, x_2^*)$.

Ha az algoritmusban megbízható módon szeretnénk x^*-ot megadni, akkor a fenti számítást intervallumokkal kell elvégeznünk. Tehát valamely $X_1^* \in \mathbb{I}$, $X_2^* \in \mathbb{I}$ befoglalásokat kell kiszámítanunk a (27) és (28) egyenletekből intervallumos változókkal és műveletekkel. A célunk a pontos x^* garantált alsó becslésének megadása, azaz annak elérése, hogy csak garantáltan inaktív pontokat töröljünk az eljárás végrehajtásával.

Jegyezzük meg, hogy az intervallumos algoritmus működéséből fakadóan X_i, X_j, Y_i és Y_j korlátai, valamint $\tilde{f}_0$ is gépi szám, így intervallumos megfelelőjük 0 szélességű lesz. (27) alapján tehát, ha a $Z_1 = \tilde{f}_0^2 - (\underline{Y_i} - \overline{Y_j})^2 \in \mathbb{I}$ befoglalás felső korlátja kisebb, mint 0, akkor X_1^* nem létezik, ellenkező esetben a gyökvonás elvégzése miatt $Z_1 := (\max\{0, \underline{Z_1}\}, \overline{Z_1})$ -t kell képeznünk, és így x_1^* befoglalására

$$X_1^* = \sqrt{Z_1} + \underline{X_i}$$

adódik. Hasonló gondolatmenettel x_2^* befoglalására annak létezése esetén

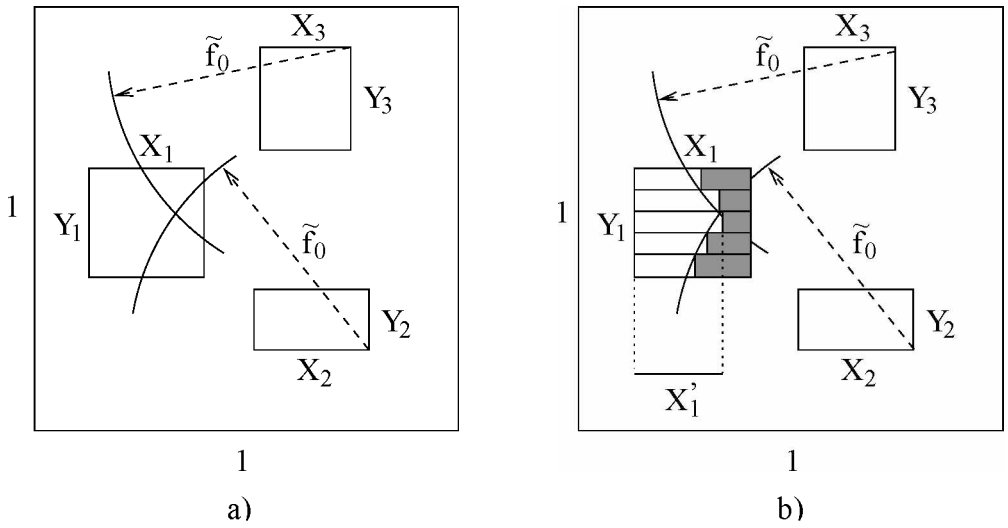
$$X_2^* = \sqrt{Z_2} + \underline{X_i}$$

teljesül. Azaz, ha X_1^* és X_2^* közül egyik sem létezik, akkor a *teljes* (X_j, Y_j) törölhető, ha pontosan egy X_k^* létezik, akkor $x^* := \underline{X_k^*}$, egyébként pedig $x^* := \min(\underline{X_1^*}, \underline{X_2^*})$.

Ezek után a megmaradó aktív terület számítása az alábbiak szerint történik: ha $x^* > \overline{X_j}$, akkor a teljes (X_j, Y_j) eliminálható, $x^* < \underline{X_j}$ esetén nem tudunk eliminálni, a többi esetben pedig a $([\underline{X_j}, x^*], Y_j)$ téglalap eliminálható. Az intervallumos befoglalások miatt a pontos számításokkal előálló maradék aktív terület minden pontját tartalmazza az intervallumos műveletekkel kapott terület, így a kerekítési és műveletvégzési hibák miatt nem veszítünk el egyetlen aktív pontot sem.

A vázolt eljárás önmagában már alkalmas az aktuálisan aktív területpárokon végzett eliminációra, ahhoz azonban, hogy magát a teljes aktív pontos eljárást hatékonnyá tegyük, érdemes még néhány dolgot figyelembe vennünk. Tekintsük a 9. Ábra a) részét, ahol az (X_1, Y_1) terület pontjait próbáljuk törölni az (X_2, Y_2) és (X_3, Y_3) téglalapok pontjai alapján. Az ábra azt a gyakorlatban sokszor előforduló esetet mutatja, amikor bár mind a 2, mind a 3 indexű téglalappal tudnánk bizonyos régiókat törölni (X_1, Y_1)-ből, az a törölhető terület téglalapokkal történő becslése miatt egyik esetben sem lehetséges. Ugyanakkor az (X_2, Y_2), illetve (X_3, Y_3) alapján elvégzett *egyidejű* elimináció már eredményezne egy téglalap alakú törölhető régiót. A további célkitűzéseink ezek alapján következők lehetnek:

1. Az aktív terület minél pontosabb közelítése (összetett, de megbízható műveletvégzést lehetővé tevő adatszerkezetek megvalósításával).



9. Ábra: Terület-elimináció cellasorok segítségével.

2. Egy-egy aktív területnek a többi aktív terület *együttes figyelembe vételével* történő redukciója.

Az 1. cél egy lehetséges megvalósításának egy lehetőségét [21, 48]-ban vetették fel, később ugyanezt a megközelítést alkalmazták [46] eljárásaiban is. A módszer az aktív területet minden esetben téglalapokkal közelíti, így annak megbízható változata különösebb nehézségek áthidalása nélkül alkalmas a jelen intervallumos módszerbe történő beillesztésre. A közelítés elve a következő: minden, kiinduláskor tekintett téglalapot osszunk fel mindkét koordináta irányban több kisebb részre (cellára), majd a távolságaik alapján releváns cellapárokra hajtsuk végre az alap eliminációs eljárást. Az eljárás időigényessége miatt az intervallumos algoritmusban kihasználtam, hogy a redukálandó és redukáló terület helyzetétől függően elég csak a cellák sorait vagy oszlopait figyelembe venni (azaz egyes cellacsoportok együtt kezelhetők). Ez alapján egy redukálandó-redukáló területpár esetén a redukálandó terület minden cellasorából (oszlopából) egy lépésben pontosan az a terület eliminálható, melynek minden pontja $\tilde{f}_0$-nél kisebb távolságra van a redukáló terület minden cellasorának (oszlopának) teljes, aktuálisan aktív részétől. A cellasorok, illetve oszlopok implementálása lehetővé teszi az aktív, illetve törlendő terület pontosabb közelítését, valamint ez a közelítés felhasználható akkor, amikor a redukálandó területből a további, még aktív redukáló területek pontjai alapján kívánunk eliminálni. A 9. b) Ábrán a cellasorokkal történő redukció eredménye látható az (X_1, Y_1) párra. Az eliminációs lépés végén az X_1 komponens helyett az X_1' intervallum lesz eltárolva az aktuális boxban.

15. MEGJEGYZÉS. *A cellázás módszerének fontos paramétere a használt cellák száma. Figyelembe véve a növekvő cellaszám esetén elérhető pontosabb területközelítést, ám a nagyobb számítási igényt is, a tapasztalataim azt mutatták, hogy* 20×20 *cellára osztás megfelelő a jelenlegi B&B algoritmus esetén. Ugyanakkor a jövőben elképzelhető a 3. Fejezetben ismertetett adaptív felosztási módszerekhez hasonló megközelítés vizsgálata,*

azaz a globális optimumhelyek közelében több, attól távolabb pedig kevesebb cellára történő darabolás alkalmazása.

4.4.4 A lokális ellenőrzés numerikus eredményei

A futtatások során a korábbról ismert optimális megoldások, azaz az $n = 2, \ldots, 27, 36$-ra vonatkozó eredmények intervallumos, garantált megbízhatóságú ellenőrzését tűztem ki célul. A felhasznált eredmények egy részét tehát a hagyományos valós aritmetikát alkalmazó módszerekkel érték el. (Az elméleti úton talált optimális megoldások esetén is érdemes lefuttatni a konstruált számítógépes algoritmust a hatékonyság tesztelése céljából.) A megoldás során a közölt, általában 16 tizedesjegyre adott optimumértékből indultam ki, ezt használva a (14) feladat maximumának legjobb ismert alsó korlátjaként (azaz $\tilde{f}_0$-ként). Ebből az (17) feladat alsó korlátja, a kivágási tesztben is használt $\tilde{f}$ érték egy intervallumos négyzetre emelés segítségével, az $F = \tilde{f}_0^2 \in \mathbb{I}$, $\tilde{f} := \underline{F}$ módon adódik. Az algoritmus induló keresési tartományának komponenseit a közölt optimális megoldás megfelelő komponensét tartalmazó 0.01 szélességű intervallummal adtam meg, vagyis az induló box minden esetben tartalmazta a közölt optimális megoldást. A $[0, 1]$ határán fekvő komponensek köré értelemszerűen a fenti módon, de a $[0, 1]$-gyel való metszetet véve adtam meg induló intervallumot.

Az algoritmus elemeinek specifikációja a Bevezetésben említett pontok alapján a következő: a befoglaló függvények megadása a 4.3 szakasz alapján került kiszámításra, az intervallum-felosztási irány a felosztandó box szélességén alapult ('A'-stratégia), a felosztási módszer pedig a klasszikus felezés volt. A felosztandó intervallum kiválasztására a Moore–Skelboe-szabályt alkalmaztam. Gyorsítótesztként a kivágási-teszt, illetve a feladatosztályra kidolgozott új monotonitási teszt (4.4.1 szakasz), a szabad köröket felderítő teszt (4.4.2 szakasz), valamint a cellázáson alapuló aktív pontos teszt (4.4.3 szakasz) szerepelt az algoritmusban. Egy vizsgált $\boldsymbol{X}$ box abban az esetben került az Eredménylistára, ha rá $w(F(\boldsymbol{X})) < 10^{-12}$ teljesült, azaz igen szűk, a duplapontosságú aritmetika ábrázolási pontosságához közeli intervallumos megoldás, illetve célfüggvényérték megadása volt a cél. Az algoritmust a Munkalista ($\mathcal{L}_{\mathcal{W}}$) kiürüléséig futtattam, tehát a teljes keresési tartomány feldolgozásra került.

A numerikus vizsgálatokat egy 333 MHz-es Pentium Celeron processzorral rendelkező, 128 MB RAM-ot tartalmazó PC-n végeztem Linux operációs rendszer alatt. A megoldó algoritmus kódolása a korábbi fejezetek numerikus tesztjeihez hasonlóan a C–XSC Toolbox [22, 29], illetve a PROFIL/BIAS [31] numerikus könyvtárak alapján történt. Az egyes futások időkorlátját 2 órában szabtam meg.

A lokális ellenőrző eljárás lehetséges eredményei az alábbiak lehetnek (az értelmezés megegyezik a (14), illetve (17) alakú feladattípusok esetén):

- Megerősítés: a feladat maximumára kapott befoglalás tartalmazza a közölt valós aritmetikával kapott maximumértéket, illetve $\mathcal{L}_{\mathcal{S}}$ egy eleme tartalmazza az előre megadott optimális megoldást. Ekkor a feladatnak nem létezik a maximum F^* befoglalásának felső korlátjánál jobb célfüggvényértékű megoldása a keresési tartományon belül, továbbá a publikált optimum legfeljebb $w(F^*) < 10^{-12}$-vel tér el a pontos optimumértéktől.

- Elvetés: mind $\mathcal{L}_{\mathcal{W}}$, mind $\mathcal{L}_{\mathcal{S}}$ üres az algoritmus terminálásakor. Ekkor a feladatnak nem létezik a publikált maximumértékkel rendelkező megoldása a keresési tartományon belül.

A megoldott feladatokon kapott eredményeket a 12. Táblázat tartalmazza, az egyes oszlopokban rendre a körök számát, a feladat dimenzióját, a futás másodpercben mért CPU idejét, a Munkalista maximális méretét, a célfüggvényhívások számát, a terület-eliminációs módszer alkalmazásainak számát és az iterációszámot feltüntetve.

Amint az a táblázatból kiolvasható, a bemutatott eljárás néhány kivételtől eltekintve az összes jelenleg ismert optimális megoldás ellenőrzését elvégezte. A feladatsor mintegy próbaként tartalmazta a '21 (-)'-szal jelölt feladatot, amiben 21 kör esetére a [35]-ben hibásan megadott pakolási értéket kellett megvizsgálni. Ezt az értéket az algoritmus a megadott futási adatok mellett elvetette – helyesen. A többi feltüntetett feladat esetén az ellenőrzés megerősítette az optimálisként publikált eredményt. A feladatcsokor tartalmazta a 21 körre közölt korrigált optimummal induló feladatot is, azonban ezt – a 22, 26 és 27 kör optimális pakolására vonatkozó feladatokkal együtt – nem sikerült a rendelkezésre álló idő alatt megoldani.

A futási mutatókkal kapcsolatban a legfontosabb észrevételeink az alábbiak: a dimenziószám növekedése nem eredményezi egyértelműen a feladat nehézségének növekedését, az ellenőrzés azokban az esetekben volt igazán nehéz, amelyekben az optimális pakolás nem valamilyen szabályos mintát követett. A futási időt döntő mértékben a legköltségesebb, ugyanakkor leghatékonyabbnak bizonyuló aktív pontos módszer alkalmazásainak száma határozta meg, míg a tárigény meglepően alacsonynak bizonyult, legalábbis a megoldott esetekben.

A bemutatott módszer hatékonyságának egy jellemzését megadhatjuk a futtatás előtti és utáni bizonytalansági tartomány összehasonlításával, azaz a kiinduló tartomány térfogatának és az Eredménylista elemei össztérfogatának összevetésével. Mivel a megoldások minden komponensét a kezdeti, kb. 0.01-ről minden esetben kb. 10^{-12}-re redukáltuk, a nehezebb $n \geq 10$ problémákon is legalább 200 *nagyságrendnyi* térfogatcsökkenést értünk el a megbízható numerikus eljárással.

4.4.5 A globális ellenőrzés numerikus eredményei

A körpakolási feladatosztály egyes példányainak megoldásához, illetve a publikált valós típusú optimális megoldások globális ellenőrzéséhez (a globalitáson ez esetben a teljes egységnégyzeten történő keresést értem) egy újabb eszközzel kell kiegészítenünk az előző szakaszokban ismertetett eljárást. Felmerül ugyanis egy további nehézség, nevezetesen a geometriailag ekvivalens, ugyanakkor az optimalizálási modellekben különböző megoldások hatékony kezelése. Ilyen jellegű problémát jelentenek pl. a szimmetria-transzformációkkal kapott optimális megoldások, illetve a körök (pontok) indexelése miatt az algoritmusokban megkülönböztetett, de geometriai értelemben azonos pakolások. Vegyük észre, hogy a lokális ellenőrzés során megkerülhettük ezt a problémát, hiszen mindvégig feltételezhettük azt, hogy a vizsgált $(\boldsymbol{X}, \boldsymbol{Y}) \subseteq [0,1]^{2n}$ box (X_i, Y_i) komponense az i-edik pakolandó pontot tartalmazza, és ezen komponensek a specifikált keresési tartományon belül nem, vagy csak kevéssé átfedőek.

n	$dim = 2n$	CPU	MLL	NFE	NMAA	IT
2	4	0.01	1	4	1	1
3	6	0.11	1	36	12	8
4	8	0.05	1	24	8	8
5	10	0.09	4	126	27	24
6	12	0.69	4	142	56	28
7	14	0.74	6	116	46	23
8	16	0.41	4	114	44	23
9	18	1.19	16	116	78	39
10	20	3.98	8	317	126	63
11	22	2.52	6	270	106	54
12	24	19.96	15	1 023	407	204
13	26	778.10	726	44 969	17 641	8 908
14	28	2.71	4	143	56	29
15	30	2.37	6	146	58	29
16	32	0.88	2	42	28	14
17	34	33.04	24	1 659	662	331
18	36	50.35	20	1 627	648	324
19	38	3 637.25	1 095	82 862	32 945	16 538
20	40	4.65	4	165	66	33
21 (-)	42	1 269.68	1 902	28 755	11 496	5 750
23	46	337.09	92	7 039	2 796	1 401
24	48	5.37	4	91	60	30
25	50	3.16	2	67	40	20
36	72	16.87	8	141	96	43

12. Táblázat: Körpakolási feladatok megoldásainak lokális ellenőrzése. A táblázat egyes oszlopai a körök számát (n), a feladat dimenzióját (dim), a futási időt (CPU, másodpercben), a Munkalista maximális méretét (MLL), a célfüggvényhívások számát (NFE), a területeliminációs módszer alkalmazásainak számát (NMAA) és az iterációszámot (IT) tartalmazzák.

Az ekvivalens megoldásoknak a B&B eljárás során történő kiszűrése legegyszerűbben a lehetséges megoldások lexikografikus rendezésével oldható meg: ez alapján csak olyan lehetséges $(\boldsymbol{x}, \boldsymbol{y}) \in [0,1]^{2n}$ megoldások vizsgálatára szorítkozunk, amelyekre

$$(x_i, y_i) \preceq (x_j, y_j) \quad \forall\, 1 \leq i \leq j \leq n \text{ esetén}, \tag{29}$$

ahol $\preceq \subseteq \mathbb{R}^2 \times \mathbb{R}^2$ a 'lexikografikusan kisebb, vagy egyenlő' reláció (azaz tetszőleges $a, b, c, d \in \mathbb{R}$ esetén $(a, b) \preceq (c, d)$ akkor és csak akkor teljesül, ha vagy $a < c$, vagy pedig $a = c$ és ezzel együtt $b \leq d$). Jegyezzük meg, hogy $\preceq$ tranzitivitása miatt adott *pont n-esre* a (29) feltétel legfeljebb $n - 1$ lexikografikus összehasonlítással ellenőrizhető.

A fenti megközelítés az algoritmusban vizsgált boxokra egyszerűen alkalmazható. Vezessük be a $\preceq_{\mathbb{I}} \subseteq \mathbb{I}^2 \times \mathbb{I}^2$ relációt az alábbiak szerint: tetszőleges $(A, B), (C, D) \in \mathbb{I}^2$

esetén $(A,B) \preceq_{\mathbb{I}} (C,D)$ akkor és csak akkor teljesüljön, ha léteznek olyan $(a,b) \in (A,B)$ és $(c,d) \in (C,D)$ pontok, melyekre $(a,b) \preceq (c,d)$. Ezen definíció alapján, ha az $(\boldsymbol{X},\boldsymbol{Y}) \subseteq [0,1]^{2n}$ boxra

$$(X_i,Y_i) \preceq_{\mathbb{I}} (X_j,Y_j) \quad \forall\, 1 \leq i \leq j \leq n \text{ esetén,} \tag{30}$$

akkor $(\boldsymbol{X},\boldsymbol{Y})$ tartalmazhat olyan pont n-eseket, amelyekre (29) áll. Ha $(\boldsymbol{X},\boldsymbol{Y})$-ra nem teljesül (30), akkor a box a további vizsgálatokból kiszűrhető. A gyakorlatban (30) tesztelése $\preceq_{\mathbb{I}}$ komplementere alapján történhet: Legyen $(X_i,Y_i) \succ_{\mathbb{I}} (X_j,Y_j) \Leftrightarrow (X_i,Y_i) \not\preceq_{\mathbb{I}} (X_j,Y_j)$, azaz $(X_i,Y_i) \succ_{\mathbb{I}} (X_j,Y_j)$ akkor és csak akkor teljesül, ha

$$\overline{X_j} < \underline{X_i}, \text{ vagy}$$

$$\overline{X_j} = \underline{X_i} \text{ és } \overline{Y_j} < \underline{Y_i}.$$

A (30) tulajdonság ellenőrzésének műveletigényét vizsgálva egyszerűen belátható, hogy – ellentétben a pontokra alkalmazott $\preceq$ relációval – $\preceq_{\mathbb{I}}$ nem tranzitív: pl. $X_1 = [0.5,1.0], X_2 = [0.0,0.5], X_3 = [0.0,0.25], Y_1 = Y_2 = Y_3 = [0.0,1.0]$ esetén $(X_1,Y_1) \preceq_{\mathbb{I}} (X_2,Y_2)$ és $(X_2,Y_2) \preceq_{\mathbb{I}} (X_3,Y_3)$, ugyanakkor $(X_1,Y_1) \not\preceq_{\mathbb{I}} (X_3,Y_3)$. Hasonló konstrukcióval tetszőleges n esetén megadható továbbá olyan $(\boldsymbol{X},\boldsymbol{Y}) \subseteq [0,1]^{2n}$ box, melyre (30) teljesül, ugyanakkor az (X_n,Y_n) komponens megváltoztatható oly módon, hogy a kapott boxra (30) csupán az $(n-1,n)$ indexpárban nem áll. Mindez azt jelenti, hogy a (30) tulajdonságot a komponenspárok összehasonlításával ellenőrizve a legrosszabb esetben $n(n-1)/2$ összehasonlítást kell végeznünk.

Számítógépes vizsgálataim azt mutatják, hogy – a $[0,1]^{2n}$ vektort megadva induló keresési tartományként – a fenti lexikografikus módszer csak kb. $n = 5$-ig szolgáltat elfogadható időn belül megoldást. A lexikografikus rendezésnek létezik azonban egy olyan alternatívája, melyet a körpakolási feladatok korábbi megoldó módszereihez fejlesztettek ki [21, 48], és amely a jelen algoritmussal együtt is alkalmazható. Az eljárásra a szakirodalom 'csempézés' ('tiling') néven hivatkozik.

A csempéző eljárás lényege az alábbi: tegyük fel, hogy (valamely rögzített n esetén) a (14) feladat maximumának valamely $\tilde{f}_0$ alsó korlátja ismert. Osszuk fel az egységnégyzetet zárt síkidomokra, ún. csempékre úgy, hogy semelyik két különböző csempének nincs közös belső pontja, továbbá bármely csempében bármely két pont közötti távolság kisebb, mint $\tilde{f}_0$. Ekkor minden olyan lehetséges megoldásra, melyhez tartozó célfüggvényérték legalább $\tilde{f}_0$, igaz az, hogy minden egyes csempe a megoldásnak legfeljebb egy pontját tartalmazza. Az optimális pakolások felderítése így az összes lehetséges n elemű csempe-*kombináció* átvizsgálásával végezhető el. Esetünkben ez annyit jelent, hogy egyenként minden egyes csempe-kombinációra (azaz az általuk definiált induló tartományokra) lefuttatjuk a 2. Algoritmust.

Érdemes megemlíteni, hogy a csempézés azon esetekben is használható, amikor nem célunk, vagy a feladat nehézsége miatt nem lehetséges az optimális pakoláshoz tartozó célfüggvényértéknek, illetve annak befoglalásának pontos meghatározása, elegendő csupán annak megállapítása, hogy a feladat optimuma *kisebb* egy adott $\hat{f}_0$ értéknél. Amennyiben az összes csempe-kombináció minden pontja törölhető a 2. Algoritmussal (a gyorsítótesztekben $\hat{f}_0$-t használva), akkor a tekintett problémapéldány optimuma bizonyosan kisebb, mint $\hat{f}_0$.

Önmagában is érdekes feladat a rögzített n-hez és a pillanatnyilag ismert legjobb $\tilde{f}_0$-hez tartozó minimális számú csempét eredményező felosztás megtalálása. Az intervallumos adatszerkezetek miatt azonban esetünkben célszerű az egyes csempéket téglalap alakúra választani. Bár a csempék számában való optimalitása nem alátámasztott, mégis, egyszerűsége miatt a csempézés leggyakrabban használt módszere az egységnégyzet $k \times l$ (uniform) részre történő felosztása (ezt használja a legtöbb eddigi, az irodalomban fellelhető módszer is). A $k \times l$-es felosztás esetén az átvizsgálandó csempe-kombinációk minimális száma így

$$\min\left\{ \binom{k \cdot l}{n} \mid k, l \geq 1 \text{ egészek, } (1/k^2 + 1/l^2)^{1/2} < \tilde{f}_0 \right\}.$$

Bár a csempe-kombinációk egy részét szimmetria okok miatt figyelmen kívül hagyhatjuk, az $n > 27$ esetek globális megoldásának elsődleges akadálya a kiinduló csempe-kombinációk igen magas száma: $n = 27$ esetén a $k = l = 6$ választás $\binom{36}{27} \approx 9.4 \cdot 10^7$ kiinduló csempe-kombinációt eredményez. [47] nem prezentál a futtatás időigényére vonatkozó számítási részleteket, annyi azonban kiderül, hogy a 27 kör optimális pakolását megadó eljárás kb. egyhavi CPU időt vett igénybe (pusztán lebegőpontos számításokat alkalmazva a 4-szer–35-ször lassabb intervallum aritmetika helyett). 28 kör esetén viszont legalább 7×6-os csempézésre van szükség, azaz $\binom{42}{28} \approx 5.3 \cdot 10^{10}$ kombinációt kellene végigvizsgálni.

A globális ellenőrzés numerikus vizsgálatainak hardver-szoftver környezete megegyezett a 4.4.4 szakaszban bemutatottakkal, azzal az eltéréssel, hogy a futási időt ezúttal 4 órában korlátoztam. Az ellenőrizendő optimumok és optimális megoldások ismét az $n = 2, \ldots, 27, 36$ esetén ismertek voltak. A B&B algoritmus beállításai megegyeztek a lokális vizsgálatokban tárgyaltakkal, ám a gyorsítótesztek alkalmazásában végrehajtottam egy módosítást, nevezetesen, a 2. Algoritmusban a kiinduló $\boldsymbol{Z}_0$ boxra még annak felosztása előtt elvégeztem az aktív pontos gyorsítótesztet.

A globális ellenőrzés a csempézés miatt tehát a B&B algoritmus többszöri lefuttatásából áll, azaz az eredmények értékelésekor mind a futási hatékonysági mutatókat, mind pedig az egyes Eredménylisták tartalmát összesíteni kell. A megoldott feladatok esetén kapott eredményeket a 13. Táblázat mutatja, az egyes oszlopokban rendre a körök számát, a feladat dimenziószámát, a felosztás formáját $k \times l$ alakban, a megvizsgált csempe-kombinációk számát, a futás idejét (másodpercben), a Munkalisták maximális méretét, a függvényhívások számát, a terület-eliminációs módszer alkalmazásainak számát és az iterációszámot feltüntetve. A futásidőt, valamint az utolsó 3 mutatószámot az egyes csempe-kombinációk vizsgálata során kapott értékek összegeként, a tárigényt jellemző MLL számot pedig ez egyedi futási adatok maximumaként képeztem.

Az eredmények összegzéseként megállapítható, hogy a publikált optimális pakolások globális ellenőrzésére a csempézési módszer lényegesen hatékonyabb a lexikografikus rendezésnél. Az időkorláton belül az $n = 2, \ldots, 20$ értékekre 3 kivétellel minden esetben sikerült ellenőrizni (megerősíteni) a megadott optimális megoldások és optimumértékek helyességét. A 'megerősítés' eredményének értelmezése a 4.4.4 szakaszban leírtak szerinti. A meg nem oldott esetek mindegyikében egy-egy egyedi kombináció vizsgálata ütközött nehézségbe, ezt $n \geq 21$ esetén csak tetézte a csempe-

n	$dim = 2n$	Comb.	Tiles_nr	CPU	MLL	NFE	NMAA	IT
2	4	2x2	6	0.02	1	8	8	2
3	6	2x2	4	0.61	1	170	66	36
4	8	2x2	1	0.09	1	27	10	9
5	10	3x2	6	0.14	1	68	26	16
6	12	3x3	84	11.65	2	1 897	813	386
7	14	3x3	36	15.99	8	1 588	660	312
8	16	3x3	9	0.84	1	75	41	16
9	18	3x3	1	0.51	1	30	18	10
10	20	4x3	66	127.67	11	4 923	2 231	1 099
11	22	5x3	1 365	625.72	48	17 395	9 414	4 065
12	24	4x4	1 820	153.34	14	3 665	3 281	736
14	28	5x4	38 760	1 088.92	14	4 246	40 698	986
15	30	5x4	15 504	454.14	6	668	15 792	147
16	32	5x4	4 845	196.85	1	33	4 866	11
19	38	5x5	177 100	13 415.15	326	48 762	197 757	10 562
20	40	5x5	53 130	3 914.12	2	316	53 311	97

13. Táblázat: Globális ellenőrzés csempézéssel. A táblázat a körök számát (n), a feladat dimenzióját (dim), a csempézést adó felosztást (Comb.), a csempe-kombinációk számát (Tiles_nr), a futások összidejét (CPU, másodpercben), a Munkalisták maximális méretét (MLL), az összes függvényhívások számát (NFE), a terület-eliminációs módszer alkalmazásainak összes számát (NMAA) és az összes iterációszámot (IT) tartalmazza.

kombinációk nagy száma. Jónéhány esetben (pl. 16 és 20 pont pakolásakor) viszont az aggregált MLL, NFE és IT mutatók nagyon alacsonyak, ami az aktív pontos módszer erejét jelzi az optimumhelyeket nem tartalmazó kombinációk eliminálásakor. Jegyezzük meg, hogy a táblázat több sorában az IT érték kisebb, mint a csempe-kombinációk száma. Ez a $\boldsymbol{Z}_0$ boxokra azonnal elvégzett terület-eliminációs gyorsítóteszt eredménye: ha ugyanis $\boldsymbol{Z}_0$-t azonnal törölni tudjuk, akkor nem kell az intervallum-felosztó lépést végrehajtanunk. Ekkor az algoritmus futása még az első iterációs ciklusba való belépés előtt befejeződik.

4.5 A továbbfejlesztett algoritmus

További kutatásaim [37, 38] eredményeként a 4.4 szakaszban ismertetett algoritmus továbbfejlesztésként egy olyan módszer jött létre, amely már nem csupán a korábban publikált számítógépes eljárások eredményének ellenőrzésére, hanem az eddig megoldatlan $n \geq 28$, $n \neq 36$ esetek közül néhány újnak a megoldására is alkalmas, kizárólag megbízható számítások alkalmazásával. Az alábbiakban a szabad körök kezelésének, majd a 4.4.3 szakasz terület-eliminációs módszerének új változatát ismertetem, végül bemutatom annak az eljárásnak az elvét, amely a 4.4.5 szakaszban említett szekvenciális csempekombináció-vizsgálat alternatívája. Jelen szakaszban az *összes* optimális pakolás megadása a cél, így a (nem szigorú) monotonitáson alapuló

módszerek nem, vagy csak módosított formában lesznek alkalmazhatók.

4.5.1 A szabad körök módszerének kifinomultabb változata

A 4.4.2 szakaszban szó esett a *szabad pont*okat tartalmazó optimális pakolások egyfajta kezeléséről. Amint arra korábban már kitértem, az eljárás hátránya az, hogy csak olyan téglalap alakú területeket redukál egy pontra, melyek minden pontja biztosan szabad pont. Ugyanakkor a szabad pont elhelyezkedésének tartománya általában néhány körív (és esetlegesen az egységnégyzet oldalai) által kijelölt konkáv síkidom. A szabad pontokat tartalmazó tartományok a gyakorlatban az optimális megoldás többi komponensét befoglaló téglalapoknál lényegesen nagyobbak és így esetleg szükségtelenül sokszor kerülnek felosztásra. Másrészt a szabad pontokhoz tartozó komponensek felosztásával nem nyerhető ki számottevő többletinformáció pl. az aktív pontos módszer számára, hiszen a célfüggvény értéke a szabad pont mozgatásával változatlan marad. Az alábbi eljárás lényege az, hogy ha egy esetlegesen szabad pontot tartalmazó komponensben találunk olyan pontot, amely 'elegendően távol' helyezkedik el a pakolás többi pontjától, akkor a szóbanforgó komponenst ideiglenesen a talált ponttal helyettesíthetjük:

1. Legyen az $(X_1, \ldots, X_n, Y_1, \ldots, Y_n) = (\boldsymbol{X}, \boldsymbol{Y}) \in \mathbb{I}^{2n}$ box a Munkalistában és az Eredménylistában tárolt összes box befoglalása a 2. Algoritmus futtatásakor tetszőleges számú iterációs lépés elvégzése után. (Tegyük fel, hogy vagy az $\mathcal{L}_{\mathcal{W}}$ Munkalista, vagy az $\mathcal{L}_{\mathcal{S}}$ Eredménylista nemüres az adott pillanatban.) Legyen $\tilde{f}$ az aktuálisan használt kivágási érték.

2. Tegyük fel, hogy léteznek olyan $p_{k_1}, \ldots, p_{k_t}$, $p_{k_s} \in (X_{k_s}, Y_{k_s})$, $s \in \{1, \ldots, t\}$ *gépi pontok* az $(\boldsymbol{X}, \boldsymbol{Y})$ box t különböző tartományában úgy, hogy

$$\underline{D}(p_{k_s}, (X_j, Y_j)) > \overline{F}(\boldsymbol{X}, \boldsymbol{Y}) \geq \tilde{f} \tag{31}$$

minden $s \in \{1, \ldots, t\}$, és minden $j \neq k_s$, $j \in \{1, \ldots, n\}$ esetén. Jelölje K a $\{k_1, \ldots k_t\}$ indexhalmazt.

3. Cseréljük az (X_i, Y_i) komponenseket a p_i pontszerű intervallumokra minden $i \in K$-ra. Futtassuk az így előálló $(\boldsymbol{X}', \boldsymbol{Y}')$ boxon a 2. Algoritmust, kihagyva az $\tilde{f}$-t javító lépést (azaz a korábban talált $\tilde{f}$-t használva), és állítsuk meg az algoritmus futását valamely iterációs lépés elvégzése után.

4. Tartalmazza $(\boldsymbol{X}'', \boldsymbol{Y}'') \in \mathbb{I}^{2n}$ a megállás pillanatában megmaradt összes boxot. Az eljárás output boxa az alábbi módon adott: (X_i, Y_i), ha $i \in K$ és (X_j'', Y_j''), ha $j \notin K$. Az utóbbi komponensek bizonyos területeit tehát $(\boldsymbol{X}, \boldsymbol{Y})$-ban törölhetjük.

A javasolt eljárás mögött az alábbi ötlet áll: tegyük fel, hogy létezik optimális pontpakolás $(\boldsymbol{X}, \boldsymbol{Y})$-ban. Akkor $\overline{F}(\boldsymbol{X}, \boldsymbol{Y}) > f^*$ áll az (17) feladat f^* globális maximumára. Tekintsük a $\{p_j \,|\, p_j \in (X_j, Y_j),\ j \notin K\}$ halmazt úgy, hogy a pontpárok közötti távolság legalább f^*. Vegyük észre, hogy (31) miatt a fenti ponthalmaz

kiegészíthető $\{p_i,\ i \in K\}$-val úgy, hogy egy optimális pakolást kapjunk. Továbbá, ha $d : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^+$ jelöli az euklideszi távolságnégyzet függvényt a síkon, akkor $d(p_i, p_j) > \overline{F}(\boldsymbol{X}, \boldsymbol{Y}) > f^*$ teljesül minden $i \in K,\ j \in \{1, \ldots, n\},\ i \neq j$ esetén. Tehát létezik olyan $\varepsilon > 0$, hogy $d(p_i', p_j) > f^*,\ \forall p_i' \in \Delta_\varepsilon(p_i)$. Azaz, a 12. Definíció értelmében p_i a pakolás szabad pontja.

28. Tétel. *[38] A fenti eljárás korrekt, azaz az előálló output box az $(\boldsymbol{X}, \boldsymbol{Y})$-ban elhelyezkedő összes optimális megoldást tartalmazza.*

Bizonyítás. Vezessük be az alábbi jelöléseket: $\boldsymbol{Z} := (\boldsymbol{X}, \boldsymbol{Y}),\ \boldsymbol{Z}' := (\boldsymbol{X}', \boldsymbol{Y}')$. Tekintsünk egy tetszőleges $\boldsymbol{W}' \subseteq \boldsymbol{Z}'$ boxot, amelyet a 2. Algoritmus futtatása során töröltünk (a gyorsító lépésekben $\tilde{f}$-t használva) a fenti eljárás 3. lépésében. Emlékeztetünk rá, hogy $W_i' = p_i, \forall i \in K$. Legyen $\boldsymbol{W} \subseteq \boldsymbol{Z}$ az alábbi módon adott: $W_j := W_j'$, ha $j \notin K$, valamint $W_i := Z_i$ a többi komponens esetén. A tétel igazolásához elegendő azt megmutatnunk, hogy $\boldsymbol{W}$ is eliminálható a 2. Algoritmussal $\tilde{f}$-t használva. (A tárgyalt továbbfejlesztett algoritmus csak az aktív pontos eljárást, valamint a kivágási tesztet tartalmazza gyorsítótesztként, így elegendő ezeket sorra vennünk.)

Először tegyük fel, hogy $\boldsymbol{W}'$-t az aktív pontos módszerrel töröltük. Vegyük észre, hogy (31) miatt terület-eliminálás csak a $j_1, j_2 \notin K$ indexű komponensek között lehetséges. Így nyilván, ha egy W_s' téglalap teljes egészében redukálható ily módon, akkor az teljes egészében törölhető $\boldsymbol{W}$-ben is, hiszen $\boldsymbol{W}$-ben az összes szóba jöhető redukáló komponens jelen van.

Másodszor, tekintsük azt az esetet, amikor $\boldsymbol{W}'$-t a kivágási teszt használatával töröltük. A bizonyításhoz szükségünk van (31) alábbi két egyszerű következményére:

$$\underline{D}(p_i, Z_j) > \tilde{f} \ \Rightarrow\ \overline{D}(p_i, W_j') = \overline{D}(W_i', W_j') > \tilde{f}, \tag{32}$$

$$\underline{D}(p_i, Z_j) > \tilde{f} \ \Rightarrow\ \overline{D}(Z_i, W_j) = \overline{D}(W_i, W_j) > \tilde{f} \tag{33}$$

minden $i \in K,\ j \neq i,\ j \in \{1, \ldots, n\},\ W_j' \subseteq Z_j$ és $W_j \subseteq Z_j$ esetén.

Mindkét állítás egyszerűen igazolható azt kihasználva, hogy D a d távolságnégyzet függvény befoglaló függvénye. Ha feltesszük ugyanis, hogy a két állítás konklúzióinak ellentettje teljesül, akkor a $d(p_i, (x_j, y_j)) \leq \tilde{f}$ állításhoz jutunk, ahol (x_j, y_j) tetszőlegesen választható W_j'-ből (32) esetén, illetve W_j-ből (33) esetén. Ez viszont ellentmond mind (32), mind (33) premisszájának.

Felhasználva, hogy $\boldsymbol{W}'$-t a kivágási teszttel töröltük, az alábbi összefüggésekhez jutunk:

$$\tilde{f} > \overline{F}(\boldsymbol{W}') = \min_{1 \leq i \neq j \leq n} \overline{D}(W_i', W_j') = \min_{\substack{1 \leq i \neq j \leq n \\ i \notin K, j \notin K}} \overline{D}(W_i', W_j') = \min_{\substack{1 \leq i \neq j \leq n \\ i \notin K, j \notin K}} \overline{D}(W_i, W_j) =$$

$$= \min_{1 \leq i \neq j \leq n} \overline{D}(W_i, W_j) = \overline{F}(\boldsymbol{W}).$$

Itt a (32) és (33) állítások konklúzióit a második, illetve negyedik egyenlőség során használtuk, míg a harmadik egyenlőség $\boldsymbol{W}$ konstrukciójából következik. A lánc első

és utolsó kifejezéséből $\overline{F}(\boldsymbol{W}) < \tilde{f}$ adódik, azaz $\boldsymbol{W}$ az $\tilde{f}$ kivágási értéket használva törölhető. Ez pedig épp a bizonyítani kívánt állítás. □

A javasolt eljárást a gyakorlatban az alábbi módon alkalmaztam: minden (X_i, Y_i), $i \in \{1, \ldots, n\}$ komponens esetén véletlen kereséssel meghatároztam egy olyan $p_i \in (X_i, Y_i)$ gépi pontot (vagyis két gépi számból álló vektort), amelyre

$$g(i) = \min_{1 \le j \le n, j \neq i} \underline{D}(p_i, (X_j, Y_j))$$

a lehető legnagyobb. Jelen algoritmusban keresési eljárásként a GLOBAL sztochasztikus optimalizálót [10] használtam. Amennyiben $g(i) > \overline{F}(\boldsymbol{X}, \boldsymbol{Y}) \geq \tilde{f}$ teljesül a legjobb talált $g(i)$ értékre és a hozzá tartozó p_i pontra, akkor biztosak lehetünk benne, hogy (31) ugyancsak teljesül p_i-re. Jegyezzük meg, hogy a kapott p_i pont megbízható, hiszen $g(i)$-t minden esetben intervallumos műveletvégzéssel számítjuk. A szabad körök kezelésére vonatkozó új módszer több nagyságrendnyi javulást eredményezett az újonnan megoldott pakolási feladatok optimumhelyei befoglalásának pontosságában.

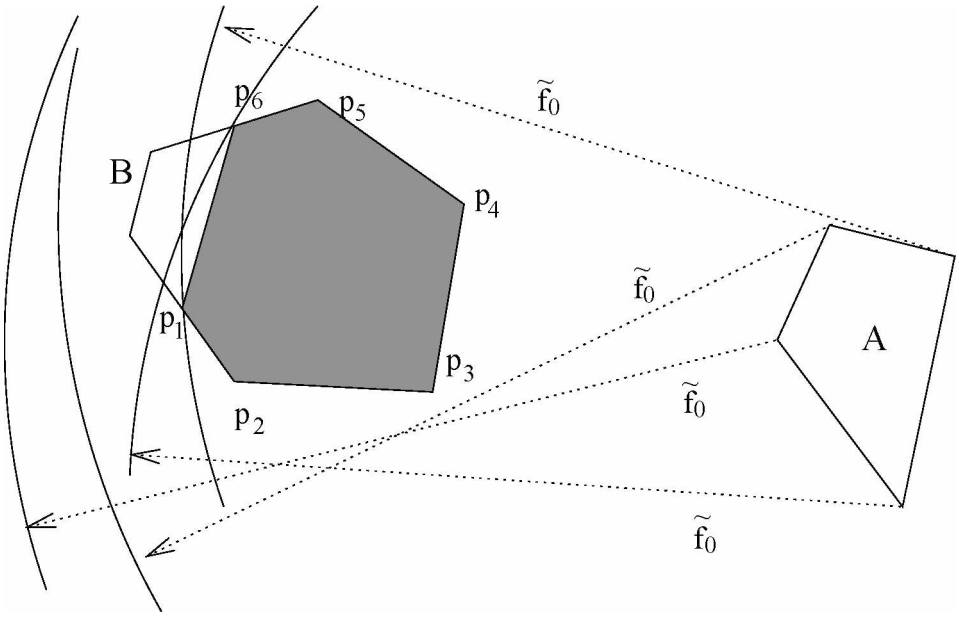
4.5.2 Az aktív pontos módszer továbbfejlesztése: terület-eliminálás poligonokkal

Bár a javasolt intervallumos algoritmus kezdeti változatának leghatékonyabb eleme a cellázást használó terület-eliminációs eljárás volt, vizsgálataim során felvetődött az aktív területek közelítésének egy további módja. A valós számábrázolást használó számítógépes "bizonyító" eljárások eddigi leghatékonyabb változatában Nurmela és Östergård az aktív területeket *poligonok*kal közelítve adott optimális pakolásokat $n =$ 27-ig [47]. Mindez arra utalhat, hogy a jelen algoritmusokban is eredményesebb lehet ez az ötlet, nem könnyű azonban egy olyan intervallumos adattípusokon alapuló összetett adatszerkezetet konstruálni, amely poligonokkal való közelítést tesz lehetővé a korábbi cellasorok és cellaoszlopok helyett.

A poligon-közelítéses eljárás eredeti változata a 2. Lemmán, illetve az alábbi Tételen alapul:

29. TÉTEL. *Nurmela és Östergård [47]: Tekintsük az A és B poligonokat a síkon. Tegyük fel, hogy $p_1, p_2, \ldots, p_k$ a B poligon határán fekvő különböző pontok úgy, hogy $p_i p_{i+1}$ $(2 \leq i \leq k-2)$ a B élei, a $p_1 p_2$, illetve $p_{k-1} p_k$ szakaszok pedig B-nek a felsoroltaktól különböző élein fekszenek. Ha a p_i $(1 \leq i \leq k)$ pontok mindegyike kisebb, mint $\tilde{f}_0$ távolságra fekszik A összes csúcsától, akkor a p_i pontok által meghatározott poligon pontjai eliminálhatók B-ből.*

A Tétel alkalmazását a 10. Ábra szemlélteti. A feladat tehát a redukálandó poligon olyan egymás utáni $p_2, \ldots, p_{k-1}$ csúcsainak megtalálása, melyek mindegyike kisebb, mint $\tilde{f}_0$ távolságra van a redukáló A poligon összes csúcsától. A redukáló poligon összes csúcsától kisebb, mint $\tilde{f}_0$ távolságra elhelyezkedő csúcsok nem feltétlenül B egymást követő csúcsai [47]. Ekkor a több lehetséges csúcssorozat egyike kerül kiválasztásra. Ha a B poligon összes csúcsára igaz a fenti feltétel, akkor a teljes B törölhető, ellenkező esetben a hátralévő feladat a p_1 és p_k pontok kiszámítása: jelölje



10. Ábra: Terület-eliminálás poligonokkal ($k = 6$).

p_0, illetve p_{k+1} a kiválasztott csúcssorozatot megelőző, illetve követő csúcsot. A p_0p_2, illetve $p_{k-1}p_{k+1}$ szakaszokon keresünk egy-egy (nyilván, az A-tól minél 'távolabbi') pontot, mely az A összes csúcsától kevesebb, mint $\tilde{f}_0$ távolságra van. A kapott két pont lesz p_1, illetve p_k. Az aktív pontos módszer egy elemi törlő lépése után megmaradó terület nyilván a p_1, p_k pontok és a B poligon $p_2, \ldots, p_{k-1}$ csúcsaitól különböző csúcsok által meghatározott.

A vázolt eljárás több helyen is kritikus a számítógépes számábrázolás szempontjából. Magában [47]-ben is felvetődik ennek kérdése, a szerzők az elvégzett számítások kerekítési hibáira adott felső korlátokkal számolnak. Ez a módszer azonban a rögzített hibakorlátok miatt nem teszi lehetővé tetszőleges pontosságú megoldások megadását, függ a használt lebegőpontos aritmetika típusától, és nehezen ellenőrizhetővé teszi az alkalmazott algoritmus korrektségét. További problémát okoz, hogy bár egzakt számítások feltételezése esetén konvex aktív területekből kiindulva az aktív pontos eljárás minden lépésében a megmaradó területek konvexek [47], a gyakorlatban épp a számábrázolási pontatlanságok miatt konkáv, sőt, akár önmagukat metsző poligonok is előállhatnak; az ilyen helyzetek elemzésére [47] – véleményem szerint – nem fordít elegendő figyelmet.)

A fenti problémák megoldására javasolt megbízható számításokon alapuló eljárás az alábbiakban foglalható össze (a módszer részletes algoritmikus leírása és helyességigazolása a jelenleg publikálás alatt álló [38]-ban található):

- Az intervallumos aktív pontos eljárás során előálló minden poligon összes csúcspontja gépi szám. A kiinduló aktív területekre, az (X_i, Y_i)-kre ez nyilvánvalóan teljesül. Az eljárás outputja vagy a teljes vizsgált box törlése (amennyiben az egyik aktív területből minden pontot törlünk), vagy az aktuális megmaradó poligonok legszűkebb befoglaló téglalapjai által definiált box.

- Az eljárás során minden poligonpárra iteratívan többször végrehajtjuk a 29. Tételen alapuló elemi törlő eljárás intervallumos változatát.

- A B redukálandó és A redukáló poligonok közötti elemi törlő eljárás során egy alkalmas $p_2, \ldots, p_{k-1}$ sorozatot intervallumos távolságszámításokkal határozunk meg. Az eredeti eljárás p_1, illetve p_k pontjai helyett azok $P_1, P_k \in \mathbb{I}^2$ befoglalásait számítjuk ki intervallumos műveletekkel. A redukció során kapott B poligonra igaz az, hogy az tartalmazza az *összes olyan megmaradó poligont, amelyet egzakt műveletvégzéssel kaphatnánk* az összes lehetséges $p_1 \in P_1$, illetve $p_k \in P_k$ pontok segítségével. Ily módon garantáltan nem veszítünk egyetlen aktív pontot sem.

- A poligonok fent említett, esetlegesen rendkívül nehezen kezelhető alakjainak elkerülése érdekében minden köztes lépésben előálló poligon vagy egy pont, vagy egy szakasz, vagy egy ún. egyszerű (önmagát nem metsző, és nem érintő) poligon lehet. Ha azt találjuk, hogy a P_1 és P_k alapján kapott megmaradó poligonra ez nem teljesül biztosan, akkor az ún. biztonsági outputként az eredeti redukálandó poligonnal térünk vissza. Ez a megközelítés jelentősen megkönnyíti a program kódjának olvasását és ellenőrzését (ne feledjük, hogy számítógépes bizonyító eljárásról van szó!), és meggyorsítja az elemi redukciós lépések végrehajtását.

4.5.3 Rész-csempekombinációk vizsgálata

Amint arra a 4.4.5 szakaszban rámutattam, az $n > 27$, $n \neq 36$ esetek megoldásának legnagyobb akadálya a végigvizsgálandó csempe-kombinációk nagy száma. A szekvenciális módszer mintegy 3 nagyságrenddel több processzoridőt (összességében több évtizedet) igényelne $n = 28$-ra a 27 kör pakolási problémájához képest.

A csempe-kombinációk szekvenciális feldolgozása helyett javasolt eljárás lényege az alábbi: a teljes (n elemű) csempe-kombinációk helyett tekintsük azok bizonyos részhalmazait. Ha valamely rész-csempekombináció törölhető az optimalizáló eljárással (azaz nem választható ki a vizsgált részhalmazban olyan pontpakolás, amelyben a páronkénti távolságnégyzetek minimuma egy aktuális $\tilde{f}$ értéknél nem kisebb), akkor az *összes* olyan n elemű csempe-kombinációt törölhetjük, amelyek tartalmazzák a tekintett részkombinációt. A csempék elhelyezkedéséből adódó lokális információt használjuk tehát kombináció-csoportok egyidejű eliminálására. Amennyiben a részkombináció csak részben törölhető, akkor az egyes csempékből törölt régiókat hagyhatjuk figyelmen kívül az eredeti, teljes kombináció vizsgálatakor. A továbbiakban jelöljük a pontpakolási probléma egyfajta általánosításának egy példányát $P(n, X_1, \ldots, X_n, Y_1, \ldots, Y_n)$-nel, ahol n a pakolandó pontok száma, $(X_i, Y_i) \in \mathbb{I}^2$, $i = 1, \ldots, n$ a keresési tartomány komponensei, a feladat célfüggvénye pedig (18) által adott. Az alábbi tétel azt mutatja, hogyan lehet egy $2m$-dimenziós pakolási feladaton kapott eredményt egy $2n$-dimenziós feladatra alkalmazni $n \geq m \geq 2$ esetén:

30. TÉTEL. *[38] Legyenek $n \geq m \geq 2$ egészek, és tekintsük a*

$$P_m = P(m, Z_1, \ldots, Z_m, W_1, \ldots, W_m) = P(m, (\boldsymbol{Z}, \boldsymbol{W})), \text{ illetve}$$

$$P_n = P(n, X_1, \ldots, X_n, Y_1, \ldots, Y_n) = P(n, (\boldsymbol{X}, \boldsymbol{Y}))$$

pontpakolási feladatokat ($X_i, Y_i, Z_i, W_i \in \mathbb{I}$; $X_i, Y_i, Z_i, W_i \subseteq [0,1]$). *Futtassuk* P_m*-re a 2. Algoritmust egy* $\tilde{f}$ *kivágási értéket használva a gyorsítótesztekben, de kihagyva az* $\tilde{f}$*-t aktualizáló lépést. Állítsuk meg az algoritmus futását tetszőleges számú iterációs lépés elvégzése után. Jelölje* $(Z'_1, \ldots, Z'_m, W'_1, \ldots, W'_m) = (\boldsymbol{Z}', \boldsymbol{W}')$ *az* $\mathcal{L}_{\mathcal{W}}$*-ben és* $\mathcal{L}_{\mathcal{S}}$*-ben található boxokat befoglaló intervallumvektort. Tegyük fel, hogy létezik egy* φ *invertálható távolságtartó geometriai transzformáció, amelyre* $\varphi(Z_i) = X_i$, $\varphi(W_i) = Y_i$, $\forall i = 1, \ldots, m$. *Ekkor minden* $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^{2n}$ *pontpakolásra, melyre* $(\boldsymbol{x}, \boldsymbol{y}) \in (\boldsymbol{X}, \boldsymbol{Y})$ *és* $f_n(\boldsymbol{x}, \boldsymbol{y}) \geq \tilde{f}$ *áll, egyidejűleg az alábbi összefüggés is teljesül:*

$$(\boldsymbol{x}, \boldsymbol{y}) \in (\varphi(Z'_1), \ldots, \varphi(Z'_m), \ldots, X_n, \varphi(W'_1), \ldots, \varphi(W'_m), \ldots, Y_n) := (\boldsymbol{X}', \boldsymbol{Y}').$$

BIZONYÍTÁS. Indirekt bizonyítást alkalmazunk: tegyük fel, hogy P_n egy $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^{2n}$, $f_n(\boldsymbol{x}, \boldsymbol{y}) \geq \tilde{f}$ lehetséges megoldását 'elveszítjük' a P_n keresési tartományának módosításakor, azaz, $(\boldsymbol{x}, \boldsymbol{y}) \in (\boldsymbol{X}, \boldsymbol{Y})$, de $(\boldsymbol{x}, \boldsymbol{y}) \notin (\boldsymbol{X}', \boldsymbol{Y}')$. Legyen $(\boldsymbol{z}, \boldsymbol{w}) = (\varphi^{-1}(x_1), \ldots, \varphi^{-1}(x_m), \varphi^{-1}(y_1), \ldots, \varphi^{-1}(y_m)) \in \mathbb{R}^{2m}$. Vegyük észre, hogy $(\boldsymbol{x}, \boldsymbol{y}) \notin (\boldsymbol{X}', \boldsymbol{Y}')$-ből $(\boldsymbol{z}, \boldsymbol{w}) \notin (\boldsymbol{Z}', \boldsymbol{W}')$ következik az indirekt feltevés miatt, valamint

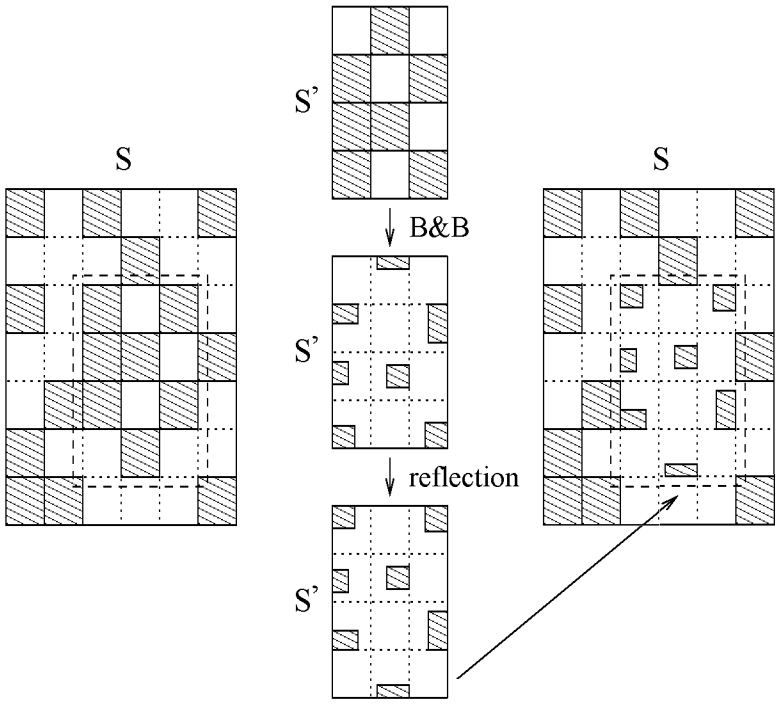
$$\tilde{f} \leq f_n(\boldsymbol{x}, \boldsymbol{y}) \leq f_m(x_1, \ldots, x_m, y_1, \ldots y_m) = f_m(\boldsymbol{z}, \boldsymbol{w}). \tag{34}$$

A (34) második egyenlőtlensége a célfüggvény (18) definíciójából, míg az egyenlőség φ távolságtartó tulajdonságából következik. (34) alapján tehát a 2. Algoritmus törölte a P_m azon $(\boldsymbol{z}, \boldsymbol{w})$ lehetséges megoldását, melyre $\tilde{f} \leq f_m(\boldsymbol{z}, \boldsymbol{w})$ teljesül. De ez ellentmondás, hiszen az 2. Algoritmus csak azon pontpakolásokat törli, amelyekhez tartozó célfüggvényérték kisebb, mint $\tilde{f}$. □

A 30. Tétel jelentése az alábbi: tegyük fel, hogy képesek vagyunk egy S' csempehalmaz valamely tartományait törölni. Amikor egy másik feladatot oldunk meg egy olyan S csempehalmazon, amelyre $|S| \geq |S'|$, továbbá $\varphi(S') \subseteq S$ (azaz S tartalmazza az S' φ melletti képét), akkor S-ben elegendő a $\varphi(S')$ komponensek esetén az S'-ben megmaradt területek φ melletti képeit vizsgálni. A tétel alkalmazását a 11. Ábra szemlélteti. Az ábrán a φ transzformáció egy tengelyes tükrözéssel egyezik meg.

A 30. Tétel következményeként, ha bebizonyosodik, hogy S' kizárólag $\tilde{f}$-nél kisebb célfüggvényértékű pontpakolásokat tartalmaz, akkor azonnal eliminálhatjuk (változatlan $\tilde{f}$ használatával) az összes olyan n elemű S csempe-kombinációt, melyre $S' \subseteq S$ fennáll. Ez utóbbi tény az alábbi módon formalizálható (a 30. Tétel jelöléseit használva):

3. KOROLLÁRIUM. *[38] Legyen* φ *az identitás transzformáció és tegyük fel, hogy a 2. Algoritmus üres Munkalistával és üres Eredménylistával terminál, azaz a teljes* $(\boldsymbol{Z}, \boldsymbol{W}) = (Z_1, \ldots, Z_m, W_1, \ldots, W_m) = (X_1, \ldots, X_m, Y_1, \ldots, Y_m)$ *keresési tartomány eliminálható az* $\tilde{f}$ *érték használatával. Ekkor* $(\boldsymbol{X}, \boldsymbol{Y})$ *nem tartalmaz olyan* $(\boldsymbol{x}, \boldsymbol{y}) \in \mathbb{R}^{2n}$ *vektort, amelyre* $f_n(\boldsymbol{x}, \boldsymbol{y}) \geq \tilde{f}$ *teljesül.*



11. Ábra: A 30. Tétel alkalmazása.

16. MEGJEGYZÉS. *Ahhoz, hogy a 30. Tételt numerikusan megbízható formában tudjuk használni, az alábbi megfigyeléseket kell tennünk:*

(i) *A $w(Z_i) = w(X_i)$ és $w(W_i) = w(Y_i)$, $\forall i = 1, \ldots, m$ összefüggéseket gépi aritmetika esetén is biztosítanunk kell, más szóval egybevágó csempéket kell előállítanunk lebegőpontos számábrázolás mellett. A megoldás az egységnégyzet felnagyítása oly módon, hogy a csempék határai gépi számok (pl. kis egészek) legyenek.*

(ii) *A φ transzformációként az identitás mellett pl. eltolásokat, forgatásokat és tengelyes tükrözéseket alkalmazhatunk. Ezen egyszerű műveleteket is intervallumos számításokkal kell azonban implementálni ahhoz, hogy a transzformációk eredményeinek garantált befoglalásait kapjuk.*

4.5.4 Az optimalitási bizonyításokban használt eljárások

A bemutatásra kerülő optimalitási bizonyítások elve a 4.5.3 szakasz 30. Tétele alapján a következő: induljunk ki viszonylag alacsony dimenziójú, könnyen feldolgozható részcsempekombinációkból, és lépésről lépésre térjünk át egyre nagyobb csempehalmazokra. Az eljárás utolsó lépésében az összes n elemű kombináció helyett csupán a megmaradt (még aktív pontokkal rendelkező) kombinációk korábban nem eliminált tartományait kell megvizsgálnunk. Maga a teljes eljárás tehát egymásra épülő fázisokból áll. (Ez a megvalósításban nem okoz nehézséget, az egyes fázisokban végrehajtandó műveletek kisméretű parancsállományokkal vezéreltek.)

A csempehalmazok 'növesztésének' természetesen több módszere is lehet. Az értekezésben egy viszonylag egyszerű eljárást mutatok be, amelynek során a csempehalmazokat balról jobbra haladva, mindig egy vagy több újonnan bevont oszlopba tartozó csempehalmazzal bővítjük. A minél kisebb erőforrásigényű köztes csempevizsgálatokból felépített többfázisú eljárások kutatása a jövő egyik fontos feladata.

Vezessük be a következő rövidítéseket és jelöléseket: n pont pakolásával és az egységnégyzet $k \times l$-es uniform felosztásával (k sorral és l oszloppal) foglalkozunk. Jelen bizonyításokban $n = 28, 29$, illetve 30, $k = 7$, $l = 6$. Az oszlopokat balról jobbra haladva sorszámozzuk. Minden fázis kezdetén meghatározunk bizonyos csempekombinációkat (és azok még aktív területeit) a korábbi fázisok eredményeiből, illetve teljes csempék alapján. Ezen csempe-kombinációk *halmazát* az $S^m_{s..f}$ módon jelöljük, ahol $s \leq f$, $s, f \in \{1, \ldots, l\}$, $m \in \{0, 1, \ldots, n\}$ azt szimbolizálva, hogy m darab csempe aktív területeit tekintjük, ahol a csempék az s, $s+1, \ldots, f$ oszlopokból származnak. Továbbá, egy $M = \{m_1, \ldots, m_k\} \subseteq \{0, 1, \ldots, n\}$ halmazra legyen $S^M_{s..f} = \cup_{i=1}^k S^{m_i}_{s..f}$. Az $S^m_{s..f}$ halmazok elemeit ezután szekvenciálisan feldolgozzuk a branch–and–bound algoritmussal. Ennek eredményeképp bizonyos csempekombinációkat részben vagy egészben eliminálunk: a kapott halmazokat – felülvonással megkülönböztetve a kiinduló halmaztól – $\bar{S}^m_{s..f}$-vel jelöljük. (Fontos megjegyezni, hogy a köztes fázisokban felesleges az algoritmus eredeti megállási feltételét használni; ehelyett az utolsó fázis kivételével az algoritmust valamely rögzített iterációszámig futtattam.) Az $\bar{S}^m_{s..f}$ halmazokból származtatjuk a további fázisokban feldolgozandó kombinációkat. A teljes algoritmus magyarázatához egy további jelölés bevezetése szükséges: ${}^{m_1}_{s_1..f_1}S^m_{s..f}$, $1 \leq s \leq s_1 \leq f_1 \leq f \leq l$, $0 \leq m_1 \leq m \leq n$ jelöli $S^m_{s..f}$ azon elemeinek halmazát, amelyek pontosan m_1 tartományt tartalmaznak az $s_1, \ldots, f_1$ oszlopokból. Jegyezzük meg, hogy a bizonyító eljárások során előálló $S^m_{s..f}$, $\bar{S}^m_{s..f}$, ${}^{m_1}_{s_1..f_1}S^m_{s..f}$, ${}^{m_1}_{s_1..f_1}\bar{S}^m_{s..f}$, halmazok természetesen a megfelelő *egzakt* megmaradó tartományok befoglalásait adják.

A felhasznált két elemi algoritmus összefoglalása a következő (az algoritmusok pszeudokódjait és diszkusszióját [38] tartalmazza):

- **Grow()** (új oszlop hozzáadása): az algoritmus inputja $\bar{S}^{m'}_{1..c}$ valamely $1 \leq m' \leq ck$ esetén, azaz, az összes olyan még nem eliminált csempe-kombináció halmaza (az egyes csempe-kombinációkhoz tartozó aktuálisan aktív területek megadásával együtt), ahol pontosan m' csempe származik az első c oszlopból. Grow() outputja ${}^{m'}_{1..c}S^{m''}_{1..(c+1)}$ egy adott m'' esetén, ahol $m_1 \leq m'' \leq \min(m'+k, n)$ egy alkalmas m_1 alsó korláttal (m_1 megadásához ld. a 17. Megjegyzést). Vagyis $\bar{S}^{m'}_{1..c}$ minden eleméhez az összes lehetséges módon hozzáadunk $m''-m'$ csempét a $c+1$. oszlopból. Mivel a csempekombinációkat balról jobbra haladva 'növesztjük', a Grow() eljárás hívásakor már ismerni fogjuk az összes $\bar{S}^m_{1..c}$, $0 \leq m \leq ck$ halmazt. Ezek elemeire a 30. Tétel alapján egy eltolást alkalmazva megkapjuk $\bar{S}^m_{2..c+1}$-t, amelyet azután $\bar{S}^m_{1..c}$-vel együtt az $1, \ldots, c+1$ oszlopokból választható aktív területek meghatározására használunk.

17. MEGJEGYZÉS. *Ami az m_1 korlát megadását illeti, $m_1 := m'$ alkalmas választás, azonban, ha a $c+2, \ldots, l$ oszlopokban elhelyezhető csempék számáról van valamilyen információnk, ennél magasabb m_1 értékkel is dolgozhatunk. Nevezetesen, ha igazolni tudjuk, hogy $S^{t_0}_{c+2..l} = \emptyset$, vagy $\bar{S}^{t_0}_{c+2..l} = \emptyset$ minden $t_0 > t$ esetén (azaz minden $(\boldsymbol{x}, \boldsymbol{y}) \in$*

$\mathbb{R}^{2n}, f_n(\boldsymbol{x},\boldsymbol{y}) \geq \tilde{f}$ *pontpakolásban legfeljebb* t *pont lehet a* $c+2,\ldots,l$ *oszlopokban), akkor* $m_1 := n-t$ *korrekt választás.*

A gyakorlatban Grow()-t az összes szóbajöhető m', és minden egyes m' esetén az összes m'', $m_1 \leq m'' \leq \min(m'+k, n)$ érték esetén alkalmazzuk.

- **Join()** (két csempehalmaz összeillesztése): tegyük fel, hogy ismertek az ${}^{m'}_{1..c}\bar{S}^{m_1}_{1..(c+1)}$ halmazok egy rögzített m', $0 \leq m' \leq ck$, és az m'-höz tartozó összes lehetséges m_1, $m' \leq m_1 \leq n$ esetén (pl. a Grow() eredményeképp). Hasonlóképpen, tételezzük fel, hogy a ${}^{n-m'}_{(c+1)..l}\bar{S}^{m_2}_{c..l}$ halmazok is adottak minden $n-m' \leq m_2 \leq n$-re. A gyakorlatban az utóbbi halmazokat ${}^{n-m'}_{1..(l-c)}\bar{S}^{m_2}_{1..(l-c+1)}$-ből kaphatjuk meg, minden elemet a teljes keresési tartomány középpontja körül 180 fokkal elforgatva. A két halmazrendszerből származó csempe-kombinációk páronként összeilleszthetők, az aktív területeket a c, illetve $c+1$ oszlopok csempéin tesztelve. Join() végrehajtásának eredménye az ${}^{m'}_{1..c}S^{n}_{1..l}$ halmaz lesz.

A Join() eljárás általánosítható $i = 0,\ldots,l-1$ közös oszloppal rendelkező kombinációk összekapcsolására. Jelen esetben azonban az $i=2$ választás már elegendőnek bizonyult a vizsgált problémapéldányok megoldásához.

4.5.5 Numerikus eredmények: 28, 29 és 30 kör optimális pakolása

Hardver-szoftver környezet. Az optimalitást igazoló eljárásokat egy Intel Pentium IV, 1400 MHz-es processzorral és 1 Gbyte RAM-mal rendelkező PC-n futtattam, a 4.4.4 és 4.4.5 szakaszok vizsgálataihoz hasonlóan Linux operációs rendszer alatt. Az algoritmus megvalósítása most is a PROFIL/BIAS [31] és a C–XSC Toolbox [22] intervallumos könyvtárakon alapult. Emellett felhasználtam a PROFIL/BIAS többszörös pontosságú intervallumos műveletvégzést biztosító csomagját [30] a kezdeti $\tilde{f}$ értékek megadásához, és ugyanez a csomag biztosította a kifelé kerekítést használó I/O rutinokat az eredmények decimális megjelenítéséhez. (Az eljárás köztes fázisaiban a csempe-kombinációkat bináris formában reprezentáltam, az ezekre meghívott be- és kiviteli eljárásokban természetesen nem volt szükség kifelé kerekítésre.)

A teljes eljárás. A megoldó eljárás egyik fontos eleme a kivágási tesztben, illetve az aktív pontos gyorsítótesztben használt $\tilde{f}$ érték minél jobb (nagyobb) megválasztása. A vizsgált esetekhez tartozó eddigi legjobb pakolásokat [20] ($n=28$-ra) illetve [45] ($n=29$, illetve $n=30$ esetén) ismertette. A megoldások koordinátái, illetve az ismert legjobb célfüggvényértékek [60]-ban találhatók 12 tizedesjegy pontossággal megadva. A nagyobb pontosságú közelítő megoldások előállítását a pakolások struktúrája alapján (azaz figyelembe véve, hogy mely pontok helyezkednek el a négyzet valamely oldalán, illetve mely pontpárok között minimális a távolság) végeztem. Minden pakolás esetén a fenti struktúra formálisan egy nemlineáris egyenletrendszerrel adott. Ezen egyenletrendszer egzakt megoldása 28 és 29 pont esetén nem ismert, így a Maple szimbolikus algebrarendszerrel 20 jegy pontosságú közelítő megoldást állítottam elő. A kapott közelítő megoldáshoz tartozó célfüggvényérték garantált befoglalását a 4.3

szakaszban tárgyalt befoglaló függvény alapján (négyszeres pontosságú intervallumos számításokkal) kaptam meg.

A 30 pontra ismert eddigi legjobb pakolás a fenti két esettel szemben szabályos struktúrát követ, a pontok koordinátái a $(qd+\mathrm{par}(p+1)(1-4d), pa)$ módon adottak, ahol $d = (20-\sqrt{10})/75$, $a = 1/5$, par a paritásfüggvény, valamint $q \in \{0,\ldots,4\}$, $p \in \{0,\ldots,5\}$. Ezen pakoláshoz tartozó célfüggvényérték pontosan d, aminek befoglalását a d-t megadó kifejezés intervallumos kiszámításával kaphatjuk. A (14) feladat optimumára adott alsó korlát kezdeti értéke tehát mindhárom feladat esetén a számított célfüggvényérték-befoglalás alsó korlátja volt:

$$\begin{aligned}\tilde{f}_{0,28} &= 0.2305354936426673,\\ \tilde{f}_{0,29} &= 0.2268829007442089,\\ \tilde{f}_{0,30} &= 0.2245029645310881,\end{aligned}$$

mely értékekhez a 4.4.5 szakaszban leírtak alapján a keresési tartomány 7×6-os csempézését rendelhetjük.

A 16. Megjegyzés (i) pontjának értelmében az egységnégyzetet a $[0,42] \times [0,42]$ négyzetté nagyítottam, így a csempék korlátai gépi számok lettek. A $\tilde{f}_0$ értékek transzformálása és a (17) feladatra való áttérés után kapott $\tilde{f}$ korlátok a következők:

$$\begin{aligned}\tilde{f}_{28} &= 93.7506267944766227,\\ \tilde{f}_{29} &= 90.8034005467881010,\\ \tilde{f}_{30} &= 88.9083890308478496.\end{aligned}$$

A branch–and–bound algoritmus elemeinek specifikációja nagyrészt megegyezett a 4.4.5 szakaszban használt beállításokkal: a befoglaló függvény továbbra is a 4.3 szakaszban tárgyalt, az intervallum-felosztási irány a klasszikus 'A'-stratégia, a felosztási módszer pedig a klasszikus felezés volt. A felosztandó intervallum kiválasztása a Moore–Skelboe-szabály alapján történt. Gyorsítótesztként a kivágási-teszt, illetve a poligonközelítésen alapuló aktív pontos módszer (4.5.2 szakasz) szerepelt az algoritmusban. A 4.4.5 szakaszhoz hasonlóan, a B&B algoritmus kezdetén a teljes aktuális keresési tartományra még annak felosztása előtt végrehajtottam az aktív pontos eljárást. Az algoritmus megállási feltételeként – ahogy arról korábban már volt szó – a végső fázis kivételével egy rögzített iterációszám elérését állítottam be (az algoritmus output boxa ekkor az $\mathcal{L}_{\mathcal{W}} \cup \mathcal{L}_{\mathcal{S}}$ elemeit tartalmazó legszűkebb box). Ez az iterációszám az első három fázisban 3, míg a negyedik fázisban (amikor a cél a megmaradt kombinációk maradék területeinek finomítása volt) 10 000 lépés volt. Az utolsó fázis megállási feltételeként a $w(F(\hat{\boldsymbol{U}}^i)) < 1e-10$ összefüggést használtam.

Mivel a bizonyító eljárás egyes fázisai igen hasonlóak mindhárom tekintett pakolási problémapéldányra, annak működését [37] alapján az $n = 28$ esetre mutatom be. Mindhárom feladat esetén ismertetem azonban a számítások és az erőforrásigények összefoglalását a 14., 15. és 16. Táblázatokban.

1. fázis: Az eljárás első lépésében $\bar{S}^m_{1..3}$-t számítottam ki néhány m esetén. Ehhez a kiinduló $S^m_{1..3}$ kombinációkat teljes csempékből kell felépítenünk, hiszen ekkor még nincs előzetes információnk a releváns aktív területekről. Vegyük észre, hogy az $S^m_{1..3}$ halmazokat nem érdemes kiszámítani 'kis' m-ekre, mivel ekkor várhatóan nem érünk

fázis	S	$\lvert S\rvert$	CPU	NFE	NMAA	$\bar{S}$	$\lvert\bar{S}\rvert$
1.	$S^{14}_{1..3}$	116 280	958	54 846	60 249	$\bar{S}^{14}_{1..3}$	17 799
	$S^{15}_{1..3}$	54 264	139	4 401	16 658	$\bar{S}^{15}_{1..3}$	1 082
	$S^{16}_{1..3}$	20 349	24	27	5 253	$\bar{S}^{16}_{1..3}$	0
	$S^{13}_{1..3}$	203 490	2 526	232 268	174 995	$\bar{S}^{13}_{1..3}$	77 852
2.	${}^{13}_{1..3}S^{17}_{1..4}$	588 703	57 072	4 641 519	3 004 399	${}^{13}_{1..3}\bar{S}^{17}_{1..4}$	390 402
	${}^{13}_{1..3}S^{18}_{1..4}$	632 289	47 620	2 843 863	2 214 934	${}^{13}_{1..3}\bar{S}^{18}_{1..4}$	226 564
	${}^{13}_{1..3}S^{19}_{1..4}$	43 729	1 174	56 620	81 812	${}^{13}_{1..3}\bar{S}^{19}_{1..4}$	3 044
	${}^{13}_{1..3}S^{20}_{1..4}$	0	-	-	-	${}^{13}_{1..3}\bar{S}^{20}_{1..4}$	0
	${}^{14}_{1..3}S^{17}_{1..4}$	438 207	35 565	2 734 521	1 894 823	${}^{14}_{1..3}\bar{S}^{17}_{1..4}$	231 820
	${}^{14}_{1..3}S^{18}_{1..4}$	354 167	22 466	1 312 522	1 090 923	${}^{14}_{1..3}\bar{S}^{18}_{1..4}$	103 582
	${}^{14}_{1..3}S^{19}_{1..4}$	88 508	2 909	143 263	177 416	${}^{14}_{1..3}\bar{S}^{19}_{1..4}$	9 478
	${}^{14}_{1..3}S^{20}_{1..4}$	2 540	13	12	2 550	${}^{14}_{1..3}\bar{S}^{20}_{1..4}$	0
	${}^{14}_{1..3}S^{21}_{1..4}$	0	-	-	-	${}^{14}_{1..3}\bar{S}^{21}_{1..4}$	0
	${}^{15}_{1..3}S^{17}_{1..4}$	21 844	1 276	95 333	74 306	${}^{15}_{1..3}\bar{S}^{17}_{1..4}$	7 816
	${}^{15}_{1..3}S^{18}_{1..4}$	26 904	1 230	66 376	65 169	${}^{15}_{1..3}\bar{S}^{18}_{1..4}$	4 962
	${}^{15}_{1..3}S^{19}_{1..4}$	14 134	312	13 370	22 543	${}^{15}_{1..3}\bar{S}^{19}_{1..4}$	853
	${}^{15}_{1..3}S^{20}_{1..4}$	2 870	24	530	3 217	${}^{15}_{1..3}\bar{S}^{20}_{1..4}$	31
	${}^{15}_{1..3}S^{21}_{1..4}$	0	-	-	-	${}^{15}_{1..3}\bar{S}^{21}_{1..4}$	0
	${}^{15}_{1..3}S^{22}_{1..4}$	0	-	-	-	${}^{15}_{1..3}\bar{S}^{22}_{1..4}$	0
3.	${}^{13}_{1..3}S^{28}_{1..6}$	243 762	2 147	6 910	249 347	${}^{13}_{1..3}\bar{S}^{28}_{1..6}$	56
	${}^{14}_{1..3}S^{28}_{1..6}$	998 204	9 107	38 614	1 028 665	${}^{14}_{1..3}\bar{S}^{28}_{1..6}$	506
4.	$S^{28}_{1..6}$	562	4 248	213 692	125 743	$\bar{S}^{28}_{1..6}$	6
5.	$S^{28}_{1..6}$	1	1 854	139 334	72 982	$\bar{S}^{28}_{1..6}$	1
	$\sum$	3 850 807	190 664	12 598 021	10 365 984		1 075 854

14. Táblázat: A 28 pontra vonatkozó pakolási probléma megoldásának részletei. A bal szélső oszlop az eljárás fázisait, a következő két oszlop az adott fázisban a B&B algoritmussal feldolgozott halmazok nevét és elemszámát mutatja. A következő oszlopcsoport az input halmazok szekvenciális feldolgozásához szükséges összes CPU időt (CPU, másodpercben), célfüggvényhívások számát (NFE), illetve az aktív pontos módszer hívásainak számát (NMAA) tartalmazza. A jobboldali két oszlop a feldolgozás után kapott halmaz nevét és méretét mutatja.

el jelentős redukciót a csempékben. Másrészt, $\bar{S}^{m_0}_{1..3} = \emptyset$-ból a 3. Korollárium miatt $\bar{S}^{m}_{1..3} = \emptyset\ \forall m \geq m_0$ következik. Ez utóbbi két megfigyelés alapján az alábbi stratégiát alkalmazhatjuk: számítsuk ki az $\bar{S}^{\lceil n/2\rceil}_{1..3}, \bar{S}^{\lceil n/2\rceil+1}_{1..3}, \ldots$ sorozat elemeit addig, amíg $\bar{S}^{\lceil n/2\rceil+t}_{1..3} = \emptyset$-t kapunk. Ezután számítsuk ki az $\bar{S}^{\lfloor n/2\rfloor}_{1..3}$ (ha n páratlan), $\bar{S}^{\lfloor n/2\rfloor-1}_{1..3}, \ldots, \bar{S}^{\lfloor n/2\rfloor-t+1}_{1..3}$ sorozatot. Az $n = 28$ esetben $\lceil n/2\rceil = \lfloor n/2\rfloor = 14$, így csak $\bar{S}^{M_1}_{1..3}, M_1 = \{14, 15, 16, 13\}$ lett kiszámítva, ugyanis $S^{16}_{1..3}$ minden elemét teljes egészében eliminálni lehetett. A további fázisokban az $\bar{S}^{m}_{1..3}$, $m \leq 12$ halmazokat teljes csempékből álló kombinációk halmazainak tekintettem.

Gyorsítások az 1. fázisban. Világos, hogy a $\bar{S}^{m}_{1..3}$ feldolgozásának erőforrásigénye jelentősen csökkenthető, ha $S^{m}_{1..3}$-ben az egymáshoz képest szimmetrikus eseteket csak egyszer vizsgáljuk. Ehhez az alábbi szimmetria-transzformációkat alkalmaztam:

fázis	S	$\lvert S\rvert$	CPU	NFE	NMAA	$\bar{S}$	$\lvert\bar{S}\rvert$
1.	$S^{15}_{1..3}$	54 264	305	14 928	22 758	$\bar{S}^{15}_{1..3}$	4 194
	$S^{16}_{1..3}$	20 349	50	395	5 518	$\bar{S}^{16}_{1..3}$	40
	$S^{17}_{1..3}$	5 985	8	0	1 574	$\bar{S}^{17}_{1..3}$	0
	$S^{14}_{1..3}$	116 280	1 618	104 620	86 973	$\bar{S}^{14}_{1..3}$	33 794
	$S^{13}_{1..3}$	203 490	3 454	338 771	230 121	$\bar{S}^{13}_{1..3}$	112 824
2.	${}^{13}_{1..3}S^{18}_{1..4}$	318 042	42 945	2 660 219	1 712 120	${}^{13}_{1..3}\bar{S}^{18}_{1..4}$	221 087
	${}^{13}_{1..3}S^{19}_{1..4}$	111 682	8 323	475 178	391 266	${}^{13}_{1..3}\bar{S}^{19}_{1..4}$	35 833
	${}^{13}_{1..3}S^{20}_{1..4}$	0	-	-	-	${}^{13}_{1..3}\bar{S}^{20}_{1..4}$	0
	${}^{14}_{1..3}S^{18}_{1..4}$	691 320	77 284	4 638 570	3 182 171	${}^{14}_{1..3}\bar{S}^{18}_{1..4}$	383 148
	${}^{14}_{1..3}S^{19}_{1..4}$	248 188	18 399	1 049 402	848 878	${}^{14}_{1..3}\bar{S}^{19}_{1..4}$	80 553
	${}^{14}_{1..3}S^{20}_{1..4}$	20 685	492	15 298	31 861	${}^{14}_{1..3}\bar{S}^{20}_{1..4}$	520
	${}^{14}_{1..3}S^{21}_{1..4}$	0	-	-	-	${}^{14}_{1..3}\bar{S}^{21}_{1..4}$	0
	${}^{15}_{1..3}S^{18}_{1..4}$	113 761	10 142	605 305	448 297	${}^{15}_{1..3}\bar{S}^{18}_{1..4}$	49 428
	${}^{15}_{1..3}S^{19}_{1..4}$	73 971	4 357	241 322	216 327	${}^{15}_{1..3}\bar{S}^{19}_{1..4}$	18 064
	${}^{15}_{1..3}S^{20}_{1..4}$	20 303	605	21 069	34 290	${}^{15}_{1..3}\bar{S}^{20}_{1..4}$	1 137
	${}^{15}_{1..3}S^{21}_{1..4}$	652	3	2	654	${}^{15}_{1..3}\bar{S}^{21}_{1..4}$	0
	${}^{15}_{1..3}S^{22}_{1..4}$	0	-	-	-	${}^{15}_{1..3}\bar{S}^{22}_{1..4}$	0
	${}^{16}_{1..3}S^{18}_{1..4}$	689	33	1 839	1 742	${}^{16}_{1..3}\bar{S}^{18}_{1..4}$	130
	${}^{16}_{1..3}S^{19}_{1..4}$	855	20	999	1 471	${}^{16}_{1..3}\bar{S}^{19}_{1..4}$	60
	${}^{16}_{1..3}S^{20}_{1..4}$	557	4	55	601	${}^{16}_{1..3}\bar{S}^{20}_{1..4}$	0
	${}^{16}_{1..3}S^{21}_{1..4}$	114	0	0	114	${}^{16}_{1..3}\bar{S}^{21}_{1..4}$	0
	${}^{16}_{1..3}S^{22}_{1..4}$	0	-	-	-	${}^{16}_{1..3}\bar{S}^{22}_{1..4}$	0
	${}^{16}_{1..3}S^{23}_{1..4}$	0	-	-	-	${}^{16}_{1..3}\bar{S}^{23}_{1..4}$	0
3.	${}^{13}_{1..3}S^{29}_{1..6}$	2 349	17	67	2 400	${}^{13}_{1..3}\bar{S}^{29}_{1..6}$	1
	${}^{14}_{1..3}S^{29}_{1..6}$	860 709	6 875	21 410	878 061	${}^{14}_{1..3}\bar{S}^{29}_{1..6}$	180
4.	$S^{29}_{1..6}$	181	3 785	157 168	80 731	$\bar{S}^{29}_{1..6}$	4
5.	$S^{29}_{1..6}$	1	3	166	107	$\bar{S}^{29}_{1..6}$	1
	$\sum$	2 864 427	178 722	10 346 783	8 178 035		940 998

15. Táblázat: A 29 pontra vonatkozó pakolási probléma megoldásának részletei. A táblázat felépítése megegyezik a 14. Táblázatéval.

tengelyes tükrözés az $x = 10.5$, illetve $y = 21$ tengelyekre és középpontos tükrözés a $(10.5, 21)$ pontra. Mivel minden, teljes csempékből álló kombináció egy bináris sztringgel azonosítható, így a szimmetrikus esetek kiszűrése ezen sztringeken végzett egyszerű műveletekkel végrehajtható. Természetesen a megmaradt kombinációk aktív területeire a megfelelő inverz transzformációkat végre kell hajtanunk ahhoz, hogy az $\bar{S}^m_{1..3}$ output halmazok összes elemét megkapjuk.

2. fázis: Az 1. fázis eredményeként rendelkezésünkre állt $\bar{S}^m_{1..3}$ minden $0 \leq m \leq 21$ esetén. Ebben a fázisban ${}^{m'}_{1..3}\bar{S}^{m''}_{1..4}$-t számítottam ki két lépésben: először az ${}^{m'}_{1..3}S^{m''}_{1..4}$ halmazokat kellett meghatározni a Grow() eljárással, majd ezen halmazok elemeit a B&B algoritmussal végigvizsgálva kaptam az ${}^{m'}_{1..3}\bar{S}^{m''}_{1..4}$ kombinációkat. Az m' és m'' paraméterek meghatározása a következő módon történt: Nyilván elegendő az $m' \in M_2 = \{13, 14, 15\}$ értékekkel dolgoznunk, ahol M_2 az 1. fázis eredményeiből

fázis	S	$\lvert S\rvert$	CPU	NFE	NMAA	$\bar{S}$	$\lvert\bar{S}\rvert$
1.	$S^{15}_{1..3}$	54 264	499	27 566	30 132	$\bar{S}^{15}_{1..3}$	7 974
	$S^{16}_{1..3}$	20 349	91	1 561	6 318	$\bar{S}^{16}_{1..3}$	255
	$S^{17}_{1..3}$	5 985	13	20	1 590	$\bar{S}^{17}_{1..3}$	0
	$S^{14}_{1..3}$	116 280	2 248	146 498	109 752	$\bar{S}^{14}_{1..3}$	47 003
2.	${}^{14}_{1..3}S^{18}_{1..4}$	0	-	-	-	${}^{14}_{1..3}\bar{S}^{18}_{1..4}$	0
	${}^{14}_{1..3}S^{19}_{1..4}$	414 511	40 089	2 298 205	1 685 461	${}^{14}_{1..3}\bar{S}^{19}_{1..4}$	182 441
	${}^{14}_{1..3}S^{20}_{1..4}$	53 774	3 110	117 194	129 588	${}^{14}_{1..3}\bar{S}^{20}_{1..4}$	7 146
	${}^{14}_{1..3}S^{21}_{1..4}$	0	-	-	-	${}^{14}_{1..3}\bar{S}^{21}_{1..4}$	0
	${}^{15}_{1..3}S^{18}_{1..4}$	53 490	7 289	432 161	281 500	${}^{15}_{1..3}\bar{S}^{18}_{1..4}$	37 167
	${}^{15}_{1..3}S^{19}_{1..4}$	191 328	14 536	814 390	653 867	${}^{15}_{1..3}\bar{S}^{19}_{1..4}$	63 213
	${}^{15}_{1..3}S^{20}_{1..4}$	58 757	3 602	133 421	141 579	${}^{15}_{1..3}\bar{S}^{20}_{1..4}$	8 812
	${}^{15}_{1..3}S^{21}_{1..4}$	3 899	45	564	4 357	${}^{15}_{1..3}\bar{S}^{21}_{1..4}$	7
	${}^{15}_{1..3}S^{22}_{1..4}$	0	-	-	-	${}^{15}_{1..3}\bar{S}^{22}_{1..4}$	0
	${}^{16}_{1..3}S^{18}_{1..4}$	3 055	311	17 255	12 708	${}^{16}_{1..3}\bar{S}^{18}_{1..4}$	1 359
	${}^{16}_{1..3}S^{19}_{1..4}$	7 170	368	19 367	18 519	${}^{16}_{1..3}\bar{S}^{19}_{1..4}$	1 359
	${}^{16}_{1..3}S^{20}_{1..4}$	4 697	153	4 878	7 983	${}^{16}_{1..3}\bar{S}^{20}_{1..4}$	263
	${}^{16}_{1..3}S^{21}_{1..4}$	1 186	13	206	1 342	${}^{16}_{1..3}\bar{S}^{21}_{1..4}$	7
	${}^{16}_{1..3}S^{22}_{1..4}$	20	0	0	20	${}^{16}_{1..3}\bar{S}^{22}_{1..4}$	0
	${}^{16}_{1..3}S^{23}_{1..4}$	0	-	-	-	${}^{16}_{1..3}\bar{S}^{23}_{1..4}$	0
3.	${}^{14}_{1..3}S^{30}_{1..6}$	16 799	104	10	16 807	${}^{14}_{1..3}\bar{S}^{30}_{1..6}$	0
	${}^{15}_{1..3}S^{30}_{1..6}$	239 430	2 068	3 197	242 002	${}^{15}_{1..3}\bar{S}^{30}_{1..6}$	36
4.	$S^{30}_{1..6}$	36	1 391	79 916	39 935	$\bar{S}^{30}_{1..6}$	2
5.	$S^{30}_{1..6}$	1	0	18	7	$\bar{S}^{30}_{1..6}$	1
	$\sum$	1 245 031	75 930	4 096 427	3 383 467		357 045

16. Táblázat: A 30 pontra vonatkozó pakolási probléma megoldásának részletei. A táblázat felépítése megegyezik a 14. Táblázatéval.

adódik. A 17. Megjegyzés értelmében az m'' paraméter m_1 alsó korlátját az 1. fázis vizsgálatához hasonló módszerrel határoztam meg: $\bar{S}^{12}_{5..6}$ üresnek bizonyult (viszont $\bar{S}^{11}_{5..6}$ nem), amivel igazolást nyert, hogy minden legalább $\tilde{f}_{28}$ célfüggvényértékű pontpakolásoknak legalább $m_1 = 17$ pontot kell az első 4 oszlopból tartalmaznia. Az m'' felső korlátja minden m' esetén $m' + 7$ volt.

3. fázis: Látható, hogy az eljárást tovább lehetett volna folytatni újabb oszlopok hozzáadásával. Tapasztalataim azonban azt mutatták, hogy a tekintett 7×6-os csempézés esetén a 2. fázis végén már elegendő lokális információ áll rendelkezésre az aktív területek lehetséges elhelyezkedéséről ahhoz, hogy a Join() eljárás végrehajtása után már csak kis számú teljes ($2n$-dimenziós) kombinációt kelljen vizsgálnunk.

A 2. fázis eredményeként ismertek az ${}^{i}_{1..3}\bar{S}^{m_i}_{1..4}$ halmazok minden $17 \leq m_i \leq 28$, $i \in M_2$ esetén. (Emlékezzünk rá, hogy a 2. fázis vizsgálata alapján nem kell az $i \leq m_i < 17$ eseteket vizsgálnunk.) Mivel a fenti halmazok csak $m_{13} \in M_{13} = \{17, 18, 19\}$, $m_{14} \in M_{14} = \{17, 18, 19\}$, valamint $m_{15} \in M_{15} = \{17, 18, 19, 20\}$ esetén nemüresek, elegendő ezen m_i értékeket tekintenünk Join() futtatásakor. Ahogy arra a Join() eljárás ismertetésekor utaltam, a $3, \ldots, 6$ oszlopokból vett csempe-kombinációk

maradék területeit (vagyis az ${}^{i}_{4..6}\bar{S}^{M_i}_{3..6}$, $i \in M_2$ halmazokat) ${}^{i}_{1..3}\bar{S}^{M_i}_{1..4}$-ből kaphatjuk, minden kombináció maradék területeit a $(21, 21)$ középpont körül 180 fokkal elforgatva.

A 3. fázis során tehát egyrészt össze kellett kapcsolni ${}^{13}_{1..3}\bar{S}^{M_{13}}_{1..4}$ minden elemét ${}^{15}_{4..6}\bar{S}^{M_{15}}_{3..6}$ elemeivel (ezen eljárást a továbbiakban Join(13,15)-tel jelöljük), majd az ${}^{14}_{1..3}\bar{S}^{M_{14}}_{1..4}$ és ${}^{14}_{4..6}\bar{S}^{M_{14}}_{3..6}$, illetve az ${}^{15}_{1..3}\bar{S}^{M_{15}}_{1..4}$ és ${}^{13}_{4..6}\bar{S}^{M_{13}}_{3..6}$ halmazokra kellett hasonló műveletet elvégezni (Join(14,14), illetve Join(15,13)). A kapott ${}^{13}_{1..3}S^{28}_{1..6}$, ${}^{14}_{1..3}S^{28}_{1..6}$ és ${}^{15}_{1..3}S^{28}_{1..6}$ halmazok elemeit a szokásos módon az optimalizáló algoritmussal egyenként dolgoztam fel. Mivel az 1. fázis során igazolást nyert, hogy az összes, legalább $\tilde{f}_{28}$ függvényértékű pontpakolás 13, 14 vagy 15 pontot tartalmazhat a négyzet mindkét felében, a ${}^{13}_{1..3}\bar{S}^{28}_{1..6} \cup {}^{14}_{1..3}\bar{S}^{28}_{1..6} \cup {}^{15}_{1..3}\bar{S}^{28}_{1..6}$ halmaz $S^{28}_{1..6}$-ként tekinthető.

Gyorsítások a 3. fázisban. Ismét lehetőség nyílt arra, hogy szimmetria tulajdonságok figyelembe vételével csökkentsük a szükséges számítások mennyiségét. Egyrészt, az általánosság megszorítása nélkül feltehetjük, hogy minden optimális pakolásban a négyzet bal fele legalább annyi pontot tartalmaz, mint a jobb (a négyzet függőleges középvonalán esetlegesen elhelyezkedő pontokat mindkét félhez tartozónak tekinthetjük). Ekkor Join(13,15) nem szükséges, ha Join(15,13)-t végrehajtottuk. A szimmetria kihasználásának másik módja páros n esetén lehetséges: Join(14,14) során feltételezhetjük, hogy minden optimális pakolásban a négyzet 3. oszlopa legalább annyi pontot tartalmaz, mint a 4. oszlop (az oszlopokat természetesen most is zárt tartományokként tekinthetjük). Azaz, Join(14,14)-ben elegendő ${}^{14}_{1..3}\bar{S}^{i}_{1..4}$ minden elemét ${}^{14}_{4..6}\bar{S}^{j}_{3..6}$ minden elemével összekapcsolnunk, ahol $i, j \in M_{14}$, és $j \geq i$. Az $n = 30$ esetben hasonló megközelítést alkalmaztam Join(15,15)-ben. (Az ismertetett két módszer előnye, hogy viszonylag könnyen implementálható. Rajtuk kívül természetesen további, kifinomultabb eljárások is kidolgozhatók.) A fenti szimmetria tulajdonságok felhasználásával kapott $S^{28}_{1..6}$-ra igaz, hogy szimmetrikus esetektől eltekintve tartalmazza a 28 pont összes optimális pakolását.

4. fázis: Az előző megállapítások alapján $S^{28}_{1..6} = {}^{13}_{1..3}\bar{S}^{28}_{1..6} \cup {}^{14}_{1..3}\bar{S}^{28}_{1..6}$. Mivel a megmaradt kombinációk még mindig sok geometriailag ekvivalens megoldást tartalmazhatnak, ezért érdemes azok aktív területeit tovább finomítani az egy-egy kombinációra végrehajtott iterációs lépések számának 3-ról 10 000-re növelésével. Ennek eredményeként sikerült a tovább vizsgálandó kombinációk számát $n = 28$ esetén hatra, $n = 29$ esetén négyre, $n = 30$ esetén pedig kettőre redukálnom.

5. fázis: Most már lehetőségünk volt a megmaradt kombinációk egyenkénti átvizsgálására. (Ezt a lépést egyelőre kézzel végeztem; mivel azonban más esetekben a fentieknél jóval több kombináció maradhat a 4. fázis végén – pl. ha sok geometriailag különböző optimális pakolás létezik – a jövőben érdemes ezt a fázist is automatizálni.) A kombinációk eredeti csempéi elhelyezkedésének elemzésével, illetve a megmaradt területekre 7×6-os csempézés helyett 6×7-est illesztve (hogy a ± 90 fokos elforgatásokból fakadó ekvivalens megoldásokat is ki lehessen szűrni) mindhárom feladat esetén sikerült igazolni, hogy azok geometriai értelemben ugyanazon optimális pakolásokat tartalmazzák. Ezzel megkaptuk a vizsgált feladatokra az első fontos eredményt: *Legyen $n \in \{28, 29, 30\}$. Ha eltekintünk a szimmetrikus esetektől, akkor egy kezdeti csempe-kombináció (illetve annak megmaradt aktív területe) tartalmazza*

i	X_i	Y_i
1.	[0.0000000000000000, 0.0000000000000142]	[0.0000000000000000, 0.0000000000000182]
2.	[0.0000000000000000, 0.0000000000000723]	[0.2833357019511142, 0.2833357019512377]
3.	[0.0000000000000000, 0.0000000000000617]	[0.5138711955937813, 0.5138711955939050]
4.	[0.0000000000000000, 0.0000000000000689]	[0.7444066892364485, 0.7444066892365722]
5.	[0.0000000000000000, 0.0000000000000094]	[0.9999999999999859, 1.0000000000000000]
6.	[0.1818703764471228, 0.1818703764471709]	[0.1416678509755570, 0.1416678509756246]
7.	[0.1996495939685054, 0.1996495939687475]	[0.3986034487723758, 0.3986034487725656]
8.	[0.1996495939685047, 0.1996495939686531]	[0.6291389424150430, 0.6291389424152327]
9.	[0.1918713858351473, 0.1918713858351900]	[0.8722033446182193, 0.8722033446182866]
10.	[0.3637407528942488, 0.3637407528964056]	[0.0000000000000000, 0.0000000000015377]
11.	[0.3815199704156590, 0.3815199704157629]	[0.2569355977967746, 0.2569355977969572]
12.	[0.3992990052060252, 0.4023815131388759]	[0.5089492733445356, 0.5138712298523256]
13.	[0.3915209798036835, 0.3915209798037213]	[0.7569355977968880, 0.7569355977969508]
14.	[0.3837427716702937, 0.3837427716703746]	[0.9999999999999370, 1.0000000000000000]
15.	[0.5633903468627852, 0.5633903468641770]	[0.1152677468208971, 0.1152677468228100]
16.	[0.5839312817244451, 0.5839312817244956]	[0.3672817571470099, 0.3672817571470896]
17.	[0.5911705737722300, 0.5911705737722556]	[0.6416678509755755, 0.6416678509756131]
18.	[0.5833923656388268, 0.5833923656389046]	[0.8847322531786048, 0.8847322531787316]
19.	[0.7630399408313210, 0.7630399408318696]	[0.0000000000000000, 0.0000000000001518]
20.	[0.7694645063572478, 0.7694645063573329]	[0.2304459563257084, 0.2304459563258553]
21.	[0.7727203431310999, 0.7727203431311068]	[0.4995893688792053, 0.4995893688792224]
22.	[0.7830419596073901, 0.7830419596074357]	[0.7694645063572883, 0.7694645063573327]
23.	[0.7830419596073660, 0.7830419596074357]	[0.9999999999999551, 1.0000000000000000]
24.	[0.9935754344739882, 0.9935754344745461]	[0.0000000000000000, 0.0000000000000352]
25.	[0.9999999999999152, 1.0000000000000000]	[0.2304459563257084, 0.2304459563257429]
26.	[0.9999999999999947, 1.0000000000000000]	[0.4609814499683757, 0.4609814499684068]
27.	[0.9999999999999551, 1.0000000000000000]	[0.6915169436110431, 0.6915169436111594]
28.	[0.9999999999999316, 1.0000000000000000]	[0.9220524372537104, 0.9220524372539002]

17. Táblázat: 28 pont optimális pakolásának befoglalása.

az n pont összes optimális pakolását. Ezt az eredményt rövidesen tovább finomítom, megmutatva, hogy a megmaradt aktív terület rendkívül kisméretűre zsugorítható.

Mindhárom pakolási feladat esetén az egyik megmaradt kombinációra végrehajtottam a szabad pontok kezelésének 4.5.1 szakaszban ismertetett módszerét. Ennek eredményeként $n = 28$ esetén a vizsgált kombináció 12. komponensét, míg $n = 29$-re az 5. komponenst ideiglenesen egy ponttal helyettesíthettem. Az $n = 30$ esetben az eljárás nem talált 'összezsugorítható' komponenst. (Ezek az eredmények egybevágnak az ismert legjobb pakolások szerkezetével, melyekben épp a fenti komponensekhez tartozó pontok mozoghatnak szabadon.) Ezután a módosított keresési tartományokra lefuttattam az optimalizáló algoritmust a $w(F(\hat{\boldsymbol{U}}^i)) < 10^{-10}$ megállási feltétellel. Ahogy azt a 4.4.4 szakasz vizsgálatai alapján sejthettük, ezen lokális jellegű keresés erőforrásigénye messze elmaradt a korábbi fázisok tár- és időigényétől. Az $\mathcal{L}_S$ elemeit befoglaló boxot képezve, majd az esetlegesen szabad pontokat tartalmazó komponenseket visszahelyettesítve kaptam az egyes pakolási feladatok maximumhelyének befoglalásait, $(\boldsymbol{X}, \boldsymbol{Y})_n^*$-t, melyeket a 17., 18., és 19. Táblázatok mutatnak. A táblázatokban aláhúzással jelöltem az egyes koordináták befoglalásának pontosságát. A három feladat maximumainak garantált befoglalásai a következők:

i	X_i	Y_i
1.	$[0.0000000000000000, 0.0000000000000246]$	$[0.0000000000000000, 0.0000000000000393]$
2.	$[0.0000000000000000, 0.0000000000000217]$	$[0.2268829007442089, 0.2268829007442435]$
3.	$[0.0000000000000000, 0.0000000000000199]$	$[0.4537658014884179, 0.4537658014884444]$
4.	$[0.0000000000000000, 0.0000000000000136]$	$[0.6806487022326268, 0.6806487022326528]$
5.	$[0.0000000000000000, 0.0532481705058822]$	$[0.9073423366158249, 1.0000000000000000]$
6.	$[0.2009548756284472, 0.2009548756284715]$	$[0.1053232576939058, 0.1053232576939296]$
7.	$[0.2009548756284483, 0.2009548756284689]$	$[0.3322061584381170, 0.3322061584381365]$
8.	$[0.2009548756284484, 0.2009548756284683]$	$[0.5590890591823258, 0.5590890591823453]$
9.	$[0.2009548756284486, 0.2009548756284655]$	$[0.7859719599265345, 0.7859719599265537]$
10.	$[0.2762399917698153, 0.2762399917698536]$	$[0.9999999999999868, 1.0000000000000000]$
11.	$[0.4019097512568943, 0.4019097512569189]$	$[0.0000000000000000, 0.0000000000000215]$
12.	$[0.4019097512568967, 0.4019097512569177]$	$[0.2268829007442089, 0.2268829007442285]$
13.	$[0.4019097512568968, 0.4019097512569169]$	$[0.4537658014884179, 0.4537658014884354]$
14.	$[0.4019097512568973, 0.4019097512569210]$	$[0.6806487022326268, 0.6806487022326512]$
15.	$[0.5983961069856828, 0.5983961069857054]$	$[0.1134414503720853, 0.1134414503721164]$
16.	$[0.5983961069856840, 0.5983961069857049]$	$[0.3403243511162964, 0.3403243511163233]$
17.	$[0.5983961069856844, 0.5983961069857049]$	$[0.5672072518605076, 0.5672072518605315]$
18.	$[0.5983961069856861, 0.5983961069857088]$	$[0.7940901526047168, 0.7940901526047401]$
19.	$[0.5031228925140242, 0.5031228925140623]$	$[0.9999999999999830, 1.0000000000000000]$
20.	$[0.7948824627144688, 0.7948824627144926]$	$[0.0000000000000000, 0.0000000000000197]$
21.	$[0.7948824627144706, 0.7948824627144921]$	$[0.2268829007442089, 0.2268829007442256]$
22.	$[0.7948824627144710, 0.7948824627144921]$	$[0.4537658014884178, 0.4537658014884334]$
23.	$[0.7948824627144749, 0.7948824627144919]$	$[0.6806487022326267, 0.6806487022326419]$
24.	$[0.7290826584835373, 0.7290826584837260]$	$[0.9795541003348881, 0.9795541003356186]$
25.	$[0.9999999999999812, 1.0000000000000000]$	$[0.0969672447170968, 0.0969672447171463]$
26.	$[0.9999999999999812, 1.0000000000000000]$	$[0.3238501454613097, 0.3238501454613550]$
27.	$[0.9999999999999820, 1.0000000000000000]$	$[0.5507330462055206, 0.5507330462055635]$
28.	$[0.9999999999999859, 1.0000000000000000]$	$[0.7776159469497363, 0.7776159469497720]$
29.	$[0.9550424244530482, 0.9550424244532232]$	$[0.9999999999999683, 1.0000000000000000]$

18. Táblázat: 29 pont optimális pakolásának befoglalása.

$$F_{28}^* = [0.2305354936426673, 0.2305354936426743], \quad w(F_{28}^*) \approx 7 \cdot 10^{-15},$$
$$F_{29}^* = [0.2268829007442089, 0.2268829007442240], \quad w(F_{29}^*) \approx 2 \cdot 10^{-14},$$
$$F_{30}^* = [0.2245029645310881, 0.2245029645310903], \quad w(F_{30}^*) \approx 2 \cdot 10^{-15}.$$

A következtetéseink az alábbiak: legyen $n \in \{28, 29.30\}$. *Szimmetrikus esetektől eltekintve, n pont optimális pakolásának összes globális maximumhelye az ismertetett boxokban található. A boxok komponensei (az esetleges szabad pontokat befoglaló tartományoktól eltekintve) igen szűkek. Ez alapján megállapíthatjuk, hogy a jelenleg ismert legjobb pakolások célfüggvényértékei a $w(F_n^*)$ pontosságon belül közelítik az egzakt optimumokat. Más szavakkal, a pontos maximumhelyeket tartalmazzák az $(\boldsymbol{X}, \boldsymbol{Y})_{28}^*$, $(\boldsymbol{X}, \boldsymbol{Y})_{29}^*$, $(\boldsymbol{X}, \boldsymbol{Y})_{30}^*$ boxok, az egyes maximumok pontos értéke pedig legfeljebb $w(F_n^*)$-gal térhet el az ismert legjobb célfüggvényértékektől.*

A 4.4.4 szakaszhoz hasonlóan ismét meghatározhatjuk, mekkora növekedést értünk el az optimális megoldás elhelyezkedésének pontosságában. Jelen esetben ezt a

$$\log(w([0,1])^{2n} / \prod_{i=1}^{n} w(X_n^*) w(Y_n^*))$$

i	X_i	Y_i
1.	[0.1019881418756426, 0.1019881418756476]	[0.0000000000000000, 0.0000000000000026]
2.	[0.0000000000000000, 0.0000000000000026]	[0.1999999999999995, 0.2000000000000022]
3.	[0.1019881418756451, 0.1019881418756477]	[0.3999999999999989, 0.4000000000000017]
4.	[0.0000000000000000, 0.0000000000000025]	[0.5999999999999984, 0.6000000000000011]
5.	[0.1019881418756451, 0.1019881418756478]	[0.7999999999999978, 0.8000000000000006]
6.	[0.0000000000000000, 0.0000000000000045]	[0.9999999999999972, 1.0000000000000000]
7.	[0.3264911064067307, 0.3264911064067357]	[0.0000000000000000, 0.0000000000000028]
8.	[0.2245029645310881, 0.2245029645310907]	[0.1999999999999995, 0.2000000000000022]
9.	[0.3264911064067332, 0.3264911064067358]	[0.3999999999999989, 0.4000000000000017]
10.	[0.2245029645310881, 0.2245029645310905]	[0.5999999999999984, 0.6000000000000011]
11.	[0.3264911064067332, 0.3264911064067359]	[0.7999999999999978, 0.8000000000000006]
12.	[0.2245029645310881, 0.2245029645310926]	[0.9999999999999972, 1.0000000000000000]
13.	[0.4490059290621762, 0.4490059290621788]	[0.1999999999999995, 0.2000000000000021]
14.	[0.4490059290621762, 0.4490059290621786]	[0.5999999999999984, 0.6000000000000011]
15.	[0.4490059290621762, 0.4490059290621807]	[0.9999999999999972, 1.0000000000000000]
16.	[0.5509940709378189, 0.5509940709378239]	[0.0000000000000000, 0.0000000000000025]
17.	[0.5509940709378213, 0.5509940709378240]	[0.3999999999999989, 0.4000000000000016]
18.	[0.5509940709378213, 0.5509940709378240]	[0.7999999999999978, 0.8000000000000006]
19.	[0.7754970354689073, 0.7754970354689120]	[0.0000000000000000, 0.0000000000000024]
20.	[0.6735088935932643, 0.6735088935932668]	[0.1999999999999995, 0.2000000000000020]
21.	[0.7754970354689095, 0.7754970354689120]	[0.3999999999999990, 0.4000000000000015]
22.	[0.6735088935932643, 0.6735088935932666]	[0.5999999999999986, 0.6000000000000011]
23.	[0.7754970354689095, 0.7754970354689120]	[0.7999999999999981, 0.8000000000000006]
24.	[0.6735088935932643, 0.6735088935932688]	[0.9999999999999976, 1.0000000000000000]
25.	[0.9999999999999955, 1.0000000000000000]	[0.0000000000000000, 0.0000000000000023]
26.	[0.8980118581243525, 0.8980118581243549]	[0.1999999999999995, 0.2000000000000019]
27.	[0.9999999999999976, 1.0000000000000000]	[0.3999999999999991, 0.4000000000000015]
28.	[0.8980118581243525, 0.8980118581243548]	[0.5999999999999987, 0.6000000000000011]
29.	[0.9999999999999977, 1.0000000000000000]	[0.7999999999999981, 0.8000000000000006]
30.	[0.8980118581243525, 0.8980118581243570]	[0.9999999999999976, 1.0000000000000000]

19. Táblázat: 30 pont optimális pakolásának befoglalása.

összefüggéssel számíthatjuk, amely $n = 28, 29, 30$-ra rendre több mint 711, 764, illetve 872 nagyságrendnyi különbséget mutat a kezdeti és az eredmény boxok térfogata között.

Amint a 14., 15., és 16. Táblázatokból leolvasható, a teljes bizonyítási eljárás időigénye rendre kb. 53, 50, illetve 20 óra volt. Ez lényegesen kevesebb, mint a korábban ismert eljárások végrehajtásának több évtizednyire becsült ideje. (Jegyezzük meg, hogy a bizonyítás táblázatokban nem jelölt részeinek, így a 2. fázisban m_1 meghatározásának, illetve a Grow() és a Join() eljárásoknak az összideje a feltüntetettekhez képest nem számottevő, a három feladaton egyenként nagyjából egypercnyi volt.) A módszer memória-felhasználása az eljárás legnagyobb részében igen kicsi volt, hiszen az egyes rész-csempekombinációk B&B eljárással történő feldolgozása nagyon hatékonyan elvégezhető. Az egyszerre sok kombináció kezelését végző Grow() és Join() eljárások esetenként nagy tárat igényeltek, azonban a rendelkezésre álló RAM mindig elég volt az összes aktuálisan vizsgált kombináció memóriában történő egyidejű tárolásához és feldolgozásához.

4.5.6 A korábbról ismert legjobb pakolásokhoz tartozó struktúrák optimalitása

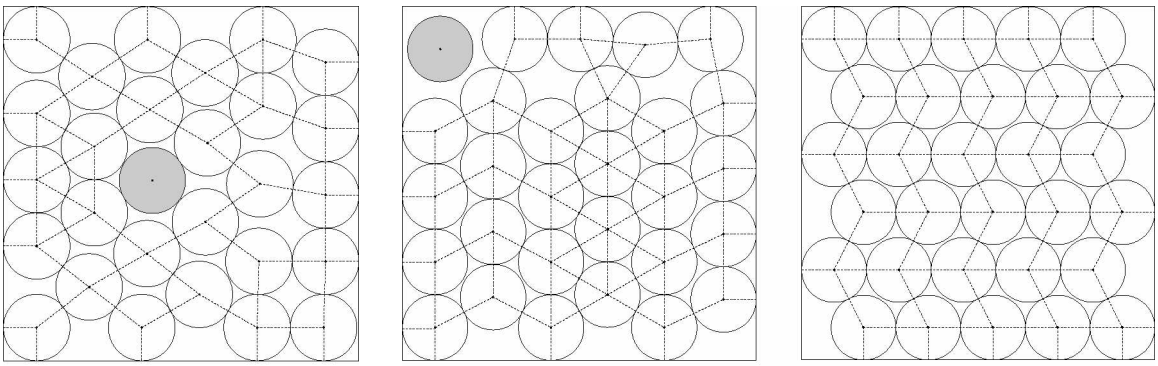
Az előző fejezetben numerikus értelemben megoldottam a 28, 29 illetve 30 pont (kör) optimális pakolására vonatkozó feladatokat: mindhárom esetben megadtam mind az optimumérték, mind pedig – szimmetrikus esetektől eltekintve – az összes optimális megoldás igen szűk befoglalását (F_n^*-ot, illetve $(\boldsymbol{X},\boldsymbol{Y})_n^*$-ot). Tekintettel arra, hogy maga a probléma geometriai jellegű, geometriai szempontból a fenti megoldás nem feltétlenül kielégítő. Ebben a szakaszban teljessé és geometriai értelemben is elfogadhatóvá teszem az optimalitási bizonyításokat: megmutatom, hogy a 4.5.5 szakaszban már említett, [20, 45]-beli pakolási struktúrák alapján adódó pakolások optimálisak, továbbá a szimmetrikus esetektől és a szabad kör mozgásától ($n = 28, 29$ esetén) eltekintve ezek az egyedüli optimális pakolások. Az említett struktúrákhoz tartozó pont-, illetve ekvivalens körpakolásokat a 12. Ábra mutatja. (Egy pakolási struktúra tehát megadja, mely pontok helyezkednek el a négyzet oldalán, mely pontpárok között minimális a távolság, illetve melyek a szabadon mozgó pontok.) A bizonyítás során $(\boldsymbol{X},\boldsymbol{Y})_n^*$ és F_n^* mellett felhasználtam a 4.5.5 szakaszban az illető struktúra alapján számított nagy pontosságú $(\hat{\boldsymbol{x}},\hat{\boldsymbol{y}})_n \in [0,1]^{2n}, n = 28, 29, 30$ közelítő megoldásokat. A bizonyítás két fő lépésre osztható:

1. lépés. Először is tisztázni kellett azt, hogy a 12. Ábrán jelölt struktúrák mennyiben feleltethetők meg egy (vagy több) *pontpakolás*nak. A továbbiakban megkülönböztetjük a teljes pakolási struktúrát, illetve annak merev (vagy legalábbis merevnek gondolt) részét, azaz a rögzített, nem mozgó pontok halmazát. A merev pontok halmazát leíró vektorokat, azok közelítését, illetve az intervallumos befoglalások merev pontokhoz tartozó komponenseiből álló vektorokat az r felső index-szel jelöljük. A merev pontokhoz tartozó koordináták száma tehát $n = 28, 29, 30$ esetén rendre 54, 56, illetve 60. Az 1. lépésben az alábbi állításokat igazoltam:

(i) *Az n ponthoz tartozó pontpakolási struktúra merev részhalmazát leíró egyenletrendszernek pontosan egy $(\boldsymbol{x},\boldsymbol{y})_n^r$ megoldása létezik $(\boldsymbol{X},\boldsymbol{Y})_n^{*,r}$-ben.*

(ii) *az $(\boldsymbol{x},\boldsymbol{y})_n^r$ pontpakolás $n = 28, 29$-re valóban kiegészíthető egy-egy szabadon mozgó ponttal úgy, hogy az illető pont $(\boldsymbol{X},\boldsymbol{Y})_n^*$ megfelelő komponensében található.*

A korábbiakban már említettük, hogy a pakolási struktúrát megadó egyenletrendszerek egzakt megoldása nem ismert $n = 28, 29$-re. Ezekben az esetekben az (i) állítást a C–XSC Toolbox könyvtárának [22] nemlineáris egyenletrendszereket megoldó intervallumos programjával igazoltam. Az említett program az intervallumos Newton-iteráción alapul és a megadott keresési tartományban elhelyezkedő összes megoldás megtalálására, valamint azok lokális unicitásának ellenőrzésére alkalmas [25, 52]. (A program egyébként az $(\boldsymbol{X},\boldsymbol{Y})_n^{*,r}$ boxnál jóval nagyobb boxon is képes volt igazolni a megoldás egzisztenciáját és unicitását.) Az eljárás az egyenletrendszer megoldásaként $(\boldsymbol{x},\boldsymbol{y})_n^r$ megbízható és igen pontos $(\boldsymbol{X},\boldsymbol{Y})_n^r$ befoglalását szolgáltatta. Az $n = 30$-hoz tartozó egyenletrendszer megoldható szimbolikusan – akár kézzel – is, a tartalmazás és a megoldás egyértelműsége itt könnyen ellenőrizhető.



12. Ábra: Az ismert legjobb pakolások struktúrája, $n = 28$, $n = 29$, valamint $n = 30$ esetén. A középpontok közötti minimális távolságot, illetve a négyzet oldalának érintését pontozott vonalak jelölik. A szabad köröket satírozással jelöltem.

A (ii) állítás a $(\boldsymbol{X}, \boldsymbol{Y})_n^r$ befoglalások alapján nyert bizonyítást. A 4.5.1 szakaszbeli módszerhez hasonlóan most azt kellett megmutatni, hogy létezik olyan, az $(\boldsymbol{X}, \boldsymbol{Y})_n^*$ j-edik komponensében található $(x, y) \in \mathbb{R}^2$ gépi pont, amelyre $(\boldsymbol{X}, \boldsymbol{Y})_n^r$ minden $i \neq j$ komponense esetén

$$\underline{D}((x, y), (X_i, Y_i)_n^r) > \overline{F}_n^* \tag{35}$$

teljesül ($n = 28$, illetve 29 esetén $j = 12$, illetve $j = 5$.) Alkalmas z keresésére ezúttal már nem volt szükség, mivel mindkét vizsgált n esetén egyszerűen ellenőrizni lehetett azt, hogy a 4.5.5 szakasz 5. fázisában talált gépi pont – melyekkel a globális keresés során a megmaradt box megfelelő komponensét helyettesítettem – kielégíti (35)-t is.

2. lépés. Ebben a lépésben megmutattam, hogy $(\boldsymbol{x}, \boldsymbol{y})_n^r$ *az egyetlen optimális pontpakolás* $(\boldsymbol{X}, \boldsymbol{Y})_n^{*,r}$*-ben.* Ehhez Nurmela és Östergård fixponttételekhez hasonló tételét használtam. A tétel elődjét a korábbi, kézzel történő bizonyításokban már többször alkalmazták. A tételben d az egyszerűség kedvéért ezúttal nem a távolságnégyzet, hanem a távolságfüggvényt jelöl.

31. Tétel. *Nurmela és Östergård [46]: Tegyük fel, hogy az* $(\boldsymbol{x}, \boldsymbol{y})_n^r$ *megoldásnak ismert egy olyan* $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})_n^r$ *közelítése, melyre*

$$d((\hat{x}_i, \hat{y}_i)_n^r, (x_i, y_i)_n^r) < d'_n \quad \textit{minden } i \textit{ komponensre}, \tag{36}$$

valamely alkalmas d'_n *esetén. Rajzoljunk minden* $(\hat{x}_i, \hat{y}_i)_n^r$ *pont köré egy-egy* $(\hat{X}_i, \hat{Y}_i)_n^r \in \mathbb{I}^2$ *hibanégyzetet a következők szerint: a hibanégyzet teljes egészében az egységnégyzetben található és tartalmazza* $(X_i, Y_i)_n^{*,r}$ *minden pontját, továbbá a fenti két tulajdonság akkor is teljesül, ha egy ugyanekkora hibanégyzetet* $(x_i, y_i)_n^r$ *körül veszünk fel. Válasszunk egy olyan* $\hat{f}_n$ *kivágási értéket, melyre*

$$\hat{f}_n + 2d'_n < f_n^* \tag{37}$$

teljesül. Futtassuk a terület-eliminációs (illetve esetünkben a teljes B&B) eljárást az $(\hat{\boldsymbol{X}}, \hat{\boldsymbol{Y}})_n^r$ *keresési tartományon véges sok iterációs lépésen át, mindvégig az* $\hat{f}_n$

kivágási értéket használva. Amennyiben a megmaradt tartomány minden komponense befoglalható egy-egy $(\hat{x}_i, \hat{y}_i)_n^r$ *köré írt* $(\hat{X}_i', \hat{Y}_i')_n^r$ *négyzetbe úgy, hogy*

$$(\hat{X}_i', \hat{Y}_i')_n^r \subset (\hat{X}_i, \hat{Y}_i)_n^r,$$

akkor $(\boldsymbol{x}, \boldsymbol{y})_n^r$ *az egyetlen optimális pontpakolás a kiinduló hibanégyzetek által definiált tartományban.*

A tétel alkalmazásakor a 4.5.5 fejezetben kiszámított nagy pontosságú közelítő megoldásokat használtam $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})_n^r$-ként. A (36) feltételt kielégítő d_n' érték $D_n' \in \mathbb{I}$ befoglalását az $(\boldsymbol{x}, \boldsymbol{y})_n^r$-re az 1. lépésben számított $(\boldsymbol{X}, \boldsymbol{Y})_n^r$ befoglalás, illetve $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})_n^r$ befoglalásának komponensenkénti intervallumos távolságai alapján kaptam. (Jegyezzük meg, hogy $(\hat{\boldsymbol{x}}, \hat{\boldsymbol{y}})_n^r$ minden komponensének befoglalását tartalmazta $(\boldsymbol{X}, \boldsymbol{Y})_n^r$ megfelelő komponense.) Ugyancsak D_n'-ből, valamint az f_n^* globális maximum F_n^* befoglalásából ezután megkonstruálható volt a (37)-beli $\hat{f}_n$ kivágási érték, melynek az $\hat{f}_n < \underline{F_n^* - 2D_n'}$ összefüggést kellett teljesítenie. A kezdeti hibanégyzeteket $2 \cdot 10^{-5}$ oldalhosszúságúnak választottam (szükség esetén az egységnégyzettel való metszetüket képezve), ezekre a tétel további feltételei egyszerűen ellenőrizhetők voltak. Hála a maximum nagy pontossággal ismert befoglalásának (és így a magas $\hat{f}_n$ kivágási értékeknek), 100 iterációs lépés mindhárom feladat esetén elegendő volt a kezdeti és megmaradt hibatartományok közötti szigorú tartalmazási reláció igazolásához. Ezzel megkaptuk a pakolási feladatok struktúrájára vonatkozó eredményt: *mindhárom pakolási feladat esetén az optimálisnak sejtett struktúra (szimmetrikus esetektől és* $n = 28, 29$*-re a szabad pont (kör) mozgásától eltekintve) az* $(\boldsymbol{X}, \boldsymbol{Y})_n^*$*-beli egyetlen optimális pont(kör-)pakolást definiálja.*

Összefoglalás

Az értekezés tárgya új típusú intervallumos globális optimalizálási eljárások elméleti és empirikus vizsgálata. Az alkalmazott intervallum aritmetika segítségével numerikus szempontból megbízható, garantált pontosságú befoglalásokat adhatunk mind az összes optimumhely, mind az optimum értékére. Az 1. Fejezetben részletes áttekintést adtam az intervallum analízis legfontosabb tulajdonságairól, a vizsgált globális optimalizálási problémák típusáról, valamint az alkalmazott korlátozás és szétválasztás típusú keretalgoritmus elemeinek általánosan használt változatairól.

A 2. Fejezetben speciális intervallum-felosztási módszereket, az ún. multisection szabályokat vizsgáltam empirikus úton. A multisection szabályokban – a klasszikus felező felosztással szemben – az aktuálisan vizsgált boxot kettőnél több részboxra osztjuk. Az összehasonlító vizsgálatokat standard optimalizálási feladatok széles és szakmailag elfogadott halmazán végeztem. Az egyes algoritmusváltozatokban különböző, 2, 3, illetve 4 részre osztó szabályokat kombináltam egyrészt a klasszikus, másrészt néhány, a közelmúltban ismertetett felosztási irányválasztó stratégiával. Az eredmények a multisection szabályok előnyeit mutatják a nehezen megoldható feladatok esetén. A legtöbb futási mutató alapján az újabb irányválasztási szabályokkal kombinált multisection változatok bizonyultak a leghatékonyabbnak. A multisection használatakor ugyanakkor csak alig nagyobb memória-felhasználásra számíthatunk, mint a felezés esetén. A vizsgálat részeként megerősítést nyertek az új irányválasztási szabályok előnyeire vonatkozó korábbi eredmények, továbbá megmutattam, hogy ezen előnyöket a multisection használatával még tovább növelhetjük: nehéz feladatok megoldásakor a legjobb multisection algoritmusok esetén várható relatív hatékonyságnövekedés – a legjobb felező módszerekhez képest – kb. 22% a futásidő, illetve 25% a célfüggvényhívások számának tekintetében.

A 3. Fejezetben egy korlátozásos feladatok esetén alkalmazható új heurisztikus mennyiség bemutatására került sor. Az új mutató arra ad becslést, hogy egy box milyen mértékben tartalmaz lehetséges megoldásokat. Ezzel lehetőség nyílik a korlátozó feltételekből fakadó többletinformáció felhasználására az algoritmus intervallum-kiválasztási, illetve intervallum-felosztási lépése során. A 3.3 szakaszbeli elméleti konvergencia-vizsgálatok legfontosabb eredménye az, hogy ha rendelkezésünkre áll f_k paraméterek egy olyan sorozata, amely a globális optimum f^* értékéhez tart, továbbá, a sorozat véges sok eleme kisebb, mint f^*, akkor a legalapvetőbb új intervallum-kiválasztási szabályt használó algoritmus a globális minimumhelyek egy halmazához tart. A javasolt kiválasztási stratégia egy alkalmas hasznossági függvény segítségével összekapcsolható a klasszikus Moore–Skelboe-szabállyal. Ha ez a függvény szigorúan monoton növő mindkét változójában, akkor az $\{f_k\}$-ra tett fenti feltételek teljesülése

garantálja a globális minimumpontok egy halmazához történő konvergenciát. Vizsgálataimban ugyanakkor azt is kimutattam, hogy a globális optimum leggyakrabban használt becslései az új szabályok alkalmazása esetén nem feltétlenül tartanak f^*-hoz.

A 3.3 szakasz második részében az új heurisztikus mutatón alapuló egyfajta adaptív intervallum-felosztási szabályt tanulmányoztam. A szabály viselkedését elemezve megállapítottam, hogy legrosszabb esetként – a befoglaló függvények tulajdonságaira tett erős kikötések ellenére is – előállhat az a helyzet, hogy az algoritmus által kiválasztott boxok tetszőlegesen sok iterációs lépésen keresztül nem tartalmaznak globális minimumhelyet, de ezen boxokon a felosztási szabály mégis a legtöbb részboxra történő felosztást javasolja.

A 3.4 szakasz átfogó numerikus tesztjeiből kiderült, hogy a korlátozó feltételeket használó kiválasztási szabályok előnyös tulajdonságokkal rendelkeznek, különösen nehéz feladatok megoldása, illetve korlátozott memória-felhasználás esetén. Bár a következtetések részben a taszításon alapuló elhelyezési feladatokon, részben nehéz standard tesztfeladaton elvégzett kísérletek eredményein alapulnak, mégis, azok érvényesek lehetnek olyan más korlátozásos feladatokon is, amelyekben az optimumpontok a lehetséges megoldások halmazának határához közel találhatók.

Az értekezés 4. Fejezetében egy, a diszkrét geometria területéről származó optimalizálási feladatot, nevezetesen az egybevágó körök négyzetbe történő optimális pakolásainak megadását vizsgáltam. Az utóbbi időszakban napvilágot látott számítógépes megoldó eljárásokkal szemben az ismertetett algoritmusok kizárólag megbízható, intervallumos számításokra támaszkodnak. A fejezet legnagyobb részében a keretalgoritmus problémaspecifikus részeinek kidolgozásával foglalkoztam. A 4.3 szakaszban egy nem-triviális intervallumos befoglaló függvényt és annak legfontosabb tulajdonságait elemeztem. A 4.4 szakaszban új típusú gyorsítólépéseket ismertettem: rámutattam, hogyan hasznosíthatók a célfüggvény monotonitási tulajdonságai, hogyan kezelhető a kontinuum sok ekvivalens optimális megoldást eredményező 'szabad körök' esete, és legfőképp, hogyan építhető fel egy korábbról ismert terület-eliminációs eljárás megbízható változata. Az első algoritmus alkalmazhatóságát a már publikált valós típusú optimális megoldások nagy pontosságú lokális és globális ellenőrzésével demonstráltam.

A 4.5 szakasz a továbbfejlesztett algoritmus leírását tartalmazza. Az új eljárásban egyrészt a szabad körök kezelésének megújított stratégiáját alkalmaztam, másrészt egy igen hatékonynak bizonyuló terület-eliminációs módszer intervallumos változatát implementáltam. Az utóbbi módszer a keresési tartomány még vizsgálandó részeit poligonokkal közelíti (szemben a 4.4 szakaszban használt cellás közelítéssel). Az eljárás talán legfontosabb része a globális megoldhatóság kérdéséhez kötődik: jelenleg az összes optimális pakolás felderítésének legígéretesebb módja az eredeti feladat részfeladatokra osztása (a keresési tér alkalmas feldarabolásával). A megvizsgálandó részfeladatok számának növekedése miatt azonban az eddigi számítógépes eljárások alkalmatlanok voltak a 27-nél több körre vonatkozó pakolási feladatok megoldására. Az értekezésben bemutattam egy gondosan felépített technikát, amely egyes részfeladatcsoportok egyidejű kezelésével áthidalja a fenti nehézséget.

A továbbfejlesztett algoritmus erejének demonstrálására a korábban megoldat-

lan 28, 29, illetve 30 kör optimális pakolására vonatkozó feladatokat oldottam meg. Mindhárom feladatpéldány esetén megadtam az összes globális optimumhely, illetve az optimumérték nagy pontosságú befoglalását. Végül megmutattam, hogy mindhárom esetben a korábban optimálisnak sejtett pakolási struktúra valóban egy optimális pakolásnak felel meg, továbbá, hogy ez a pakolás szimmetriától (illetve 28 és 29 kör esetén a szabad kör mozgásától) eltekintve az egyetlen optimális megoldás.

Summary

The subject of the dissertation is the theoretical and empirical investigation of new interval methods for global optimization. Interval arithmetic provides a powerful tool to produce numerically reliable enclosures of all the global optimizers and the optimum value with guaranteed accuracy. In Chapter 1 I gave a detailed overview of interval analysis, the global optimization problem types we are dealing with, and some of the commonly used variants of the basic elements of the branch–and–bound frame algorithm.

In Chapter 2 several interval subdivision methods, the so-called multisection rules were discussed in an empirical way. In contrast to the classical bisection, a multisection rule divides the leading box into many subboxes. The comparative study was performed on a wide set of standard test problems. I combined different subdivision rules producing 2, 3, and 4 subboxes with the traditional and some recently investigated subdivision direction selection strategies. From the results we can conclude that multisection is advantageous in solving hard problems. According to the majority of indicators, the multisection algorithm variants used together with the newer direction selection rules were the most efficient. On the other hand, multisection usually means only slightly larger memory complexity than bisection. As a part of the study, the results of earlier numerical tests showing the advantages of the newer direction selection rules were confirmed, and moreover, it was shown that multisection may additionally improve these advantages. For hard problems, the expected relative improvements of the best multisection algorithm variants are about 22% in the running time and about 25% in the number of objective function evaluations – compared to the best bisection methods.

In Chapter 3 a new heuristic quantity for inequality constrained problems was introduced. The quantity approximates the feasibility degree of a box, thus, it is suitable to utilize the constraint information both in interval selection and in subdivision rules. The main result of the theoretical investigations (Section 3.3) is that the proposed new interval selection rule enables the branch-and-bound interval optimization algorithm to converge to a set of global optimizer points if we have a proper sequence of f_k parameter values that converges to the global optimum f^* and that has at most finite number of values smaller than f^*. The new interval selection rule can be combined with the classical Moore-Skelboe selection rule in a suitable utility function. If this function is strictly monotonically increasing in both of its arguments, then the above criteria for $\{f_k\}$ guarantee the convergence to a set of global optimizers. However, I also pointed out that for algorithms using the new selection rules the commonly used estimators of the optimum value may fail to converge to f^*.

The behaviour of an adaptive interval multisection rule based on the feasibility index was also studied. As a worst case, it turned out that – even under strict restrictions of the inclusion functions – the algorithm may select subboxes not containing global optimizers for an arbitrary long time, and at the same time the leading boxes are splitted into the largest number of subboxes proposed by the multisection rule.

Through an extensive computational study (Section 3.4), it has been shown that the constraint-based interval selection rules have several advantageous features, especially for harder problems and for computations with memory limitations. Although these conclusions were in part obtained for a particular obnoxious facility location model and for standard global optimization test problems, they may be valid for similar constrained problems for which the optimizers are close to the border of the feasible set.

Chapter 4 deals with an optimization problem arising from the discrete geometry, namely, the problem of finding the optimal packings of congruent circles in a square. In contrast to the recent computer-assisted solving methods, the presented algorithms are fully based on interval computations. In the bulk of the chapter I discussed the construction of some elements of the algorithmic frame. These tools were designed specifically for the considered problem class. In Section 4.3 a non-trivial interval inclusion function was presented and its properties were investigated. In Section 4.4 some new interval accelerating devices were presented: I showed how to utilize the monotonicity properties of the objective function, how to handle a continuum number of equivalent optimizers in the case of 'free circles', and mainly, how to build a reliable variant of an earlier known area reduction method. The applicability of the first version of the developed algorithm was shown by the local and global validation of the previously published real-type solutions.

Section 4.5 contains the description of an advanced algorithm. On the one hand, I improved the method of handling free circles, and on the other hand, I developed the interval version of a very efficient area reduction method representing the remaining parts of the search regions by polygons (instead of the rectangular representation discussed in Section 4.4). Probably the most important part of the algorithm is connecting to the question of obtaining global optimizers: currently, the most promising strategy of finding optimal circle packing configurations is the partitioning of the original problem into subproblems. Still as a result of the highly increasing number of subproblems, earlier computer-aided methods were not able to solve problem instances where the number of circles was greater than 27. In the present dissertation I introduced a carefully developed technique resolving this difficulty by eliminating large groups of subproblems together.

As a demonstration of the capabilities of the new algorithm the previously unsolved problems of packing 28, 29, and 30 circles were solved. High-precision enclosures of all the global optimizers and the optimum value were given for all the three problems instances. Finally, it was shown that for all the three cases the conjectured optimal packing structure really corresponds to an optimal packing. Moreover, apart from symmetry (and from the movement of a free circle for $n = 28, 29$) this optimal packing is unique.

Irodalomjegyzék

[1] S. Berner (1996), *New Results on Verified Global Optimization*, Computing 57, 323–343.

[2] L.G. Casado and I. García (1998), *New load balancing criterion for parallel interval global optimization algorithm*, Proceedings of the 16th IASTED International Conference, 321–323, Garmisch-Partenkirchen, Germany.

[3] L.G. Casado, I. García, and T. Csendes (2000), *A new multisection technique in interval methods for global optimization*, Computing 65, 263–269.

[4] L.G. Casado, I. García, and T. Csendes (2001), *A heuristic rejection criterion in interval global optimization algorithms*, BIT 41, 683–692.

[5] L.G. Casado, I. García, P.G. Szabó, and T. Csendes (2001), *Equal Circles Packing in a Square II. New Results for up to 100 Circles Using the TAMSASS-PECS algorithm.* In: F. Giannessi, P. Pardalos, and T. Rapcsak (eds.): Optimization Theory - Recent Developments from Mátraháza, 207–224, Kluwer, Dordrecht.

[6] L.G. Casado, J.A. Martínez, and I. García (2001), *Experimenting with a new selection criterion in a fast interval optimization algorithm*, J. Global Optimization 19, 247–264.

[7] A.E. Csallner (1993), *Global optimization in separation network synthesis*, Hungarian Journal of Industrial Chemistry 21, 303–308.

[8] A.E. Csallner and T. Csendes (1996), *On the convergence speed of interval methods for global optimization*, Computers, Mathematics and Applications 31, 173–178.

[9] A.E. Csallner, T. Csendes, and M.Cs. Markót (2000), *Multisection in interval branch–and–bound methods for global optimization. I. Theoretical results*, J. Global Optimization 16, 371–392.

[10] T. Csendes (1988), *Nonlinear parameter estimation by global optimization – efficiency and reliability*, Acta Cybernetica 8, 361–370.

[11] T. Csendes (2001), *New subinterval selection criteria for interval global optimization*, J. Global Optimization 19, 307–327.

[12] T. Csendes, *Generalized subinterval selection criteria for interval global optimization.* Publikálásra elfogadva a Numerical Algorithms folyóiratban.

[13] T. Csendes and J. Pintér (1993), *The impact of accelerating tools on the interval subdivision algorithm for global optimization*, European Journal of Operational Research 65, 314–320.

[14] T. Csendes and D. Ratz (1996), *A review of subdivision direction selection in interval methods for global optimization*, ZAMM 76, 319–322.

[15] T. Csendes and D. Ratz (1997), *Subdivision direction selection in interval methods for global optimization*, SIAM Journal on Numerical Analysis 34, 922–938.

[16] L.C.W. Dixon and G.P. Szegö (1978), *The Global Optimization Problem: An Introduction.* In: L.C.W. Dixon and G.P. Szegö (eds.): Towards Global Optimization 2, 1-15, North-Holland, Amsterdam.

[17] Z. Drezner (1995), *Facility Location: a Survey of Applications and Methods*, Springer-Verlag, Berlin.

[18] J. Fernández, P. Fernández, and B. Pelegrín (2000), *A continuous location model for siting a non-noxious undesirable facility within a geographical region*, European Journal of Operational Research 121, 259–274.

[19] J. Fernández and B. Pelegrín (2001), *Using interval analysis for solving planar single-facility location problems: new discarding tests*, J. Global Optimization 19, 61–81.

[20] R.L. Graham and B.D. Lubachevsky (1996), *Repeated patterns of dense packings of equal disks in a square*, The Electronic J. of Combinatorics 3, 1–16.

[21] C. de Groot, R. Peikert, and D. Würtz (1990), *The optimal packing of ten equal circles in a square*, IPS Research Report, ETH Zürich No. 90-12, August, 1990.

[22] R. Hammer, M. Hocks, U. Kulisch, and D. Ratz (1995), *C++ Toolbox for Verified Computing I.*, Springer-Verlag, Berlin.

[23] E. Hansen (1980), *Global optimization using interval analysis – the multidimensional case*, Numerische Mathematik 34, 247–270.

[24] E. Hansen (1992), *Global Optimization Using Interval Analysis*, Marcel Dekker, New York.

[25] E. Hansen and S. Sengupta (1981), *Bounding Solutions of System of Equations Using Interval Analysis*, BIT 21, 203–211.

[26] R.B. Kearfott (1996), *Rigorous Global Search: Continuous Problems*, Kluwer, Dordrecht.

[27] R.B. Kearfott (1996), *Test results for an interval branch and bound algorithm for equality-constrained optimization.* In: State of the art in global optimization, Kluwer, Dordrecht, 181–199.

[28] K. Kirchner and G. Wengerodt (1987), *Die dichteste Packung von 36 Kreisen in einem Quadrat*, Beiträge zur Algebra und Geometrie 25, 147–159.

[29] R. Klatte, U. Kulisch, C. Lawo, M. Rauch, and A. Wiethoff (1993), *C–XSC — A C++ Class Library for Extended Scientific Computing*, Springer-Verlag, Heidelberg.

[30] O. Knüppel (1993), *A Multiple Precision Arithmetic for PROFIL*, Bericht 93.6., Technische Universität Hamburg-Harburg.

[31] O. Knüppel (1994), *PROFIL/BIAS — a fast interval library*, Computing 53, 277–287.

[32] R. Krawczyk (1983), *Intervallsteigungen für rationale Funktionen und zugeordnete zentrische Formen*, Freiburger Intervall-Berichte, 83/2, Freiburg.

[33] M. Locatelli and U. Raber (2002), *Packing equal circles in a square: a deterministic global optimization approach*, Discrete Applied Mathematics 122, 139-166.

[34] R.F. Love, J.G. Morris, and G.O. Wesolowsky (1988), *Facilities Location: Models and Methods*, North-Holland, New York.

[35] C.D. Maranas, C.A. Floudas, and P.M. Pardalos (1995), *New results in the packing of equal circles in a square*, Discrete Mathematics 142, 287–293.

[36] M.Cs. Markót (2000), *An interval method to validate optimal solutions of the "packing circles in a unit square" problems*, Central European Journal of Operations Research 8, 63–78.

[37] M.Cs. Markót, *Optimal Packing of 28 Equal Circles in a Unit Square – the First Reliable Solution.* Publikálásra elfogadva a Numerical Algorithms folyóiratban. Elérhető: `http://www.inf.u-szeged.hu/~markot/opt28.ps.gz`.

[38] M.Cs. Markót and T. Csendes, *A New Verified Optimization Technique for the "Packing Circles in a Unit Square" Problems.* Publikálásra benyújtva a SIAM J. Optimization folyóirathoz. Elérhető:
`http://www.inf.u-szeged.hu/~markot/impcirc.ps.gz`.

[39] M.Cs. Markót, T. Csendes, and A.E. Csallner (2000), *Multisection in interval branch–and–bound methods for global optimization. II. Numerical tests*, Journal of Global Optimization 16, 219–228.

[40] M.Cs. Markót, J. Fernández, L.G. Casado, and T. Csendes, *New interval methods for constrained global optimization.* Feltételesen elfogadva a Mathematical Programming folyóiratban. Elérhető: `http://www.inf.u-szeged.hu/~markot/mathprog.ps.gz`.

[41] H. Melissen (1994), *Densest packing of six equal circles in a square*, Elemente der Mathematik 49, 27–31.

[42] R.E. Moore (1966), *Interval Analysis*, Prentice-Hall, Englewood Cliffs, N.J.

[43] L. Moser (1960), *Problem 24 (corrected)*, Canadian Mathematical Bulletin 3, 78.

[44] A. Neumaier (2001), *Introduction to Numerical Analysis*, Cambridge University Press, Cambridge.

[45] K.J. Nurmela and P.R.J. Östergård (1997), *Packing up to 50 equal circles in a square*, Discrete & Computational Geometry 18, 111–120.

[46] K.J. Nurmela and P.R.J. Östergård (1999), *Optimal packings of equal circles in a square.* In: Y. Alavi, D.R. Lick, and A. Schwenk (eds.): Combinatorics, Graph Theory, and Algorithms. Proc. 8th Quadrennial International Conference on Graph Theory, Combinatorics, Algorithms, and Applications, 671–680.

[47] K.J. Nurmela and P.R.J. Östergård (1999) *More optimal packings of equal circles in a square*, Discrete & Computational Geometry, 22 (1999), 439–457.

[48] R. Peikert, D. Würtz, M. Monagan, and C. de Groot (1992), *Packing circles in a square: a review and new results.* In: P. Kall (ed.): System Modelling and Optimization, Lecture Notes in Control and Information Sciences 180, 45–54, Springer-Verlag, Berlin.

[49] L.B. Rall (1981), *Automatic Differentiation: Techniques and Applications*, Lecture Notes in Computer Science 120.

[50] H. Ratschek and J. Rokne (1984), *Computer Methods for the Range of Functions*, Ellis Horwood, Chichester.

[51] H. Ratschek and J. Rokne (1988), *New Computer Methods for Global Optimization*, Ellis Horwood, Chichester.

[52] D. Ratz (1992), *Automatische Ergebnisverifikation bei globalen Optimierungsproblemen*, Dissertation, Universität Karlsruhe.

[53] D. Ratz (1994), *Box-splitting strategies for the interval Gauss-Seidel step in a global optimization method*, Computing 53, 337–353.

[54] D. Ratz and T. Csendes (1995), *On the Selection of Subdivision Directions in Interval Branch–and–bound Methods for Global Optimization*, J. Global Optimization 7, 183–207.

[55] D. Ratz (1996), *On Branching Rules in Second-Order Branch–and–bound Methods for Global Optimization.* In: G. Alefeld, A. Frommer, and B. Lang (eds.): Scientific Computing and Validated Numerics, 221–227, Akademie-Verlag, Berlin.

[56] J. Schaer (1965), *The densest packing of nine circles in a square*, Canadian Mathematical Bulletin 8, 273–277.

[57] J. Schaer and A. Meir (1965), *On a geometric extremum problem*, Canadian Mathematical Bulletin 8, 21–27.

[58] S. Skelboe (1974), *Computation of rational interval functions*, BIT 14, 87–95.

[59] F.J. Solis and J.B. Wets (1981), *Minimization by random search techniques*, Mathematics of Operations Research 6(1), 19–50.

[60] E. Specht, www.packomania.com.

[61] P.G. Szabó (2000), *Some New Structures for the "Equal Circles Packing in a Square" Problem*, Central European Journal of Operations Research 8, 79–91.

[62] A. Törn and A. Žilinskas (1987), *Global Optimization*, Springer-Verlag, Berlin.

[63] R. Vaidyanathan and M. El-Halwagi (1994), *Global optimization of nonconvex nonlinear programs via interval analysis*, Computers & Chemical Engineering 18, 889–897.

[64] G. Wengerodt (1983), *Die dichteste Packung von 16 Kreisen in einem Quadrat*, Beiträge zur Algebra und Geometrie 16, 173–190.

[65] G. Wengerodt (1987), *Die dichteste Packung von 14 Kreisen in einem Quadrat*, Beiträge zur Algebra und Geometrie 25, 25–46.

[66] G. Wengerodt (1987), *Die dichteste Packung von 25 Kreisen in einem Quadrat*, Ann. Univ. Sci. Budapest Eötvös Sect. Math 30, 3–15.