NEMLINEÁRIS SZIMBOLIKUS TRANSZFORMÁCIÓK OPTIMALIZÁLÁSI FELADATOKRA[1]

CSENDES TIBOR – DOBJÁNNÉ ANTAL ELVIRA
Szegedi Tudományegyetem – Neumann János Egyetem, Kecskemét

A szimbolikus algebra rendszerek elterjedése kapcsán egyre nyilvánvalóbb az optimalizálási feladatok átírásának, kedvezőbb alakra hozásának lehetősége. Az ilyen egyszerűsítések haszna többrétű lehet. Egyrészt kevesebb művelettel kiértékelhető alakra hozhatjuk az érintett függvényeket. Fölismerhetünk továbbá olyan, a feladat redundanciájára vonatkozó összefüggéseket, amiket általában nincs esélyünk észrevenni. Ezek ismeretében az átalakított feladat megoldásaiból az optimális pontokból álló alteret is meghatározhatjuk. Végül az esetleges dimenziócsökkentés a legtöbb optimalizálási eljárás iterációszámának csökkenését eredményezi. Az automatikus működés miatt eljárásunk érdemi emberi közbeavatkozást nem igényel, és így nagyméretű, bonyolult feladatok is kezelhetővé válhatnak. A jelen cikkben áttekintjük az e téren végzett munkánkat, és új eredményeket mutatunk az intervallum aritmetikán alapuló globális optimalizálási algoritmus hatékonyságának és pontosságának növelése vonatkozásában.

1 Bevezető

Szimbolikus eszközök régóta használatosak optimalizálási feladatok megoldásában. Például a szimbolikus preprocesszálás a lineáris programozásban [16], illetve ilyen átalakítás az AMPL automatikus ,,presolving" mechanizmusa is [9,10]. Egy újabb példa Liberti és munkatársai Reformulation-Optimization Software Engine nevű (vegyes-) egészértékű optimalizálási segédprogramja [14].

Csendes és Rapcsák cikkei [7,19] megmutatták, hogy lehetséges korlátozás nélküli nemlineáris optimalizálási feladatokat úgy átírni automatikus módon szimbolikus eszközökkel, hogy kölcsönösen egyértelmű hozzárendelés létezzen a két probléma szélsőértékei között. Ez a módszer alkalmas a redundáns változók kiküszöbölésére és a feladat más szempontból való egyszerűsítésére is.

Az eredeti (kicsit bonyolult) motiváló feladat a következő légzésmechanikai paraméterbecslési probléma volt [13], az

$$F\left(R_{aw}, I_{aw}, B, \tau\right) = \left[\frac{1}{m}\sum_{i=1}^{m}\left|Z_L(\omega_i) - Z'_L(\omega_i)\right|^2\right]^{1/2}$$

[1]Az elvégzett kutatást részben a Nemzeti Fejlesztési Ügynökség TÁMOP-4.2.2/08/1/2008-0008 pályázata támogatta. Beérkezett: 2017. január 7. E-mail: csendes@inf.szte.hu.

minimalizálása, ahol $Z_L(\omega_i) \in \mathbf{C}$ a mért impedancia érték és $Z'_L(\omega_i)$ az impedancia modellfüggvénye az ω_i frekvencia értékekre ($i = 1, 2, \ldots, m$). A keresett modell paraméterek R_{aw}, I_{aw}, B, és τ. Az eredeti nemlineáris modellfüggvény a fizikai modell paramétereivel:

$$Z'_L(\omega) = R_{aw} + \frac{B\pi}{4.6\omega} - \imath \left(I_{aw}\omega + \frac{B\log(\gamma\tau\omega)}{\omega} \right),$$

ahol $\gamma = 10^{1/4}$ és $\imath$ a képzetes egység.

A szimbolikus algoritmus kidolgozását a következő egyszerűsített modellfüggvény motiválta, ami lineáris a modell paraméterekben:

$$Z'_L(\omega) = R_{aw} + \frac{B\pi}{4.6\omega} - \imath \left(I_{aw}\omega + \frac{A + 0.25B + B\log(\omega)}{\omega} \right).$$

A nehezen fölismerhető ügyes helyettesítés lényegében az $A = B\log(\tau)$. A modell paraméterek száma megegyezik a két alakban. Az átalakítás nagy érdeme, hogy lineáris modellek legkisebb négyzetes illesztése egyszerű, nem is igényel optimalizálást.

A cikk a következő szerkezetet követi: a második szakaszban ismertetjük az ötletet és megadjuk a kapcsolódó elméleti eredményeket, a harmadik szakaszban pedig az eljárással elérhető javulás részleteit adjuk meg több szempont szerint rendezve.

2 Az egyszerűsítési módszer, ötlet és elméleti eredmények

Tekintsük a korlátozás nélküli nemlineáris optimalizálási feladatot a

$$\min_{x \in I\!R^n} \; f(\mathbf{x}), \tag{1}$$

alakban, ahol $f(\mathbf{x}) : I\!R^n \to I\!R$ egy sima függvény, amit képlet formájában is ismerünk. Itt képleten szimbólumoknak (konstans, változó, műveleti jel, függvénynév és zárójel) egy zárt alakban fölírt, szintaktikailag helyes kombinációját értjük. Ezt egy számítógépes algebra rendszerben (mint amilyen például a Mathematica) általában egy irányított körmentes gráfot [20] leíró listával adhatjuk meg. A szimbolikus alak rendelkezésre állása az egyszerűsítés természetes föltétele, habár az optimalizálandó függvény gyakran csak eljárással adott, esetleg közelítés formájában, vagy például szimulációra támaszkodva érhető el.

Az egyszerűsítő eljárás azt dönti el, hogy (1) átalakítható-e olyan ekvivalens alakba, amely kedvezőbb valamely szempontból: az új függvény kevesebb aritmetikai műveletet igényel a kiértékeléshez, a probléma dimenziója alacsonyabb, vagy egyszerűbb megoldani valamely más okból. Az ekvivalens alak itt azt jelenti, hogy kölcsönösen egyértelmű leképezés adható az eredeti és az átalakított feladat szélsőérték pontjai között.

Csendes és Rapcsák [7] megmutatták, hogy egy $g(\mathbf{y})$ célfüggvény ekvivalens $f(\mathbf{x})$-el, ha $g(\mathbf{y})$ előáll a következő transzformációval:

- alkalmazzunk egy helyettesítést $f(\mathbf{x})$-re:

$$y_i := h(\mathbf{x}) \,, \quad 1 \leq i \leq n,$$

  ahol $h(\mathbf{x})$ egy sima függvény, aminek értékkészlete $\mathbb{R}$, és szigorúan monoton legalább egy x_i változójában,

- a megmaradó változókat nevezzük át:

$$y_j := x_j \,, \quad j = 1, \ldots, i-1, i+1, \ldots, n,$$

  és

- hagyjuk el azokat az y_i változókat, amelyek nem szerepelnek a kialakult célfüggvényben.

Alkalmas helyettesítésen olyan $y_i = h(\mathbf{x})$ helyettesítést értünk, amelyre teljesül, hogy

- $h(\mathbf{x})$ sima, monoton legalább egy x_i változóban, és az értékkészlete $\mathbb{R}$,
- $h(\mathbf{x})$ jellemzi legalább egy x_i változó összes előfordulását, azaz x_i teljesen eltávolítható $f(\mathbf{x})$-ből, ha $h(\mathbf{x})$-et y_i-vel helyettesítjük, és
- $y_i = h(\mathbf{x})$ nem egy egyszerű átnevezés, tehát $h(\mathbf{x}) \neq x_i$, $i = 1, \ldots, n$.

Az $y_i = h(\mathbf{x})$ helyettesítés után $\mathbf{y}$ változóinak száma legföljebb annyi, mint az $\mathbf{x}$ dimenziója. A redundáns változók törlődnek, ha $h(\mathbf{x})$ több változó összes előfordulását jellemzi. Más szóval, fel tudjuk ismerni, hogy a modellünket kevesebb ismeretlennel is meg lehet-e fogalmazni. A megadott, a helyettesítésre vonatkozó föltételek elegendőek, de nem föltétlen szükségesek. Ennek megfelelően a tárgyalt eljárás csak egy a lehetségesek között, más utak is elképzelhetők.

Tekintsük például az $f(x_1, x_2) = (x_1 + x_2)^2$ függvény minimalizálását. Ez nyilvánvalóan ekvivalens a $g(y_1) = y_1^2$ szélsőértékének megkeresésével, és az x_1 és x_2 optimális értékeit meghatározhatjuk az $y_1 = x_1 + x_2$ egyenletből, ami egy alkalmas helyettesítést ad. Ezen a módon végtelen sok minimumpontot is tudunk kezelni, ami amúgy nem lenne lehetséges a szokásos numerikus módszerekkel. Ez az egyik fő célja a módszerünknek: szimbolikus helyettesítéssel felismerni és alkalmas helyettesítéssel megszüntetni a redundanciát a függvényünkben.

Az első cikkben [7] igazolt két tétel föltételeket ad az alkalmazott transzformációkra ahhoz, hogy az eredeti és az átírt feladat megoldásai származtathatók legyenek.

1. tétel. *Ha $h(\mathbf{x})$ sima és szigorúan monoton az x_i változóban, akkor az ennek megfelelő transzformáció egyszerűsíti a függvényt abban az értelemben,*

hogy $h(\mathbf{x})$ minden előfordulását egy új változóval helyettesítjük az átalakított $g(\mathbf{y})$ függvényben, míg $f(\mathbf{x})$ minden helyi minimumpontja (maximumpontja) transzformáltja a $g(\mathbf{y})$ függvény minimumpontja (maximumpontja) lesz.

2. tétel. *Ha $h(\mathbf{x})$ sima és szigorúan monoton az x_i változóban, és az értékkészlete egyenlő a valós számokkal ($\mathbb{R}$), akkor a $g(\mathbf{y})$ átalakított függvény minden $\mathbf{y}^*$ helyi minimumpontjához (maximumpontjához) van olyan $\mathbf{x}^*$, hogy $\mathbf{y}^*$ az az $\mathbf{x}^*$ transzformáltja, és $\mathbf{x}^*$ helyi minimumpontja (maximumpontja) $f(\mathbf{x})$-nek.*

Ugyanez a cikk módszert javasolt az alkalmas helyettesítési képletek megkeresésére az 1. és 2. állításban [7], ehhez képezni kell a $\partial f(\mathbf{x})/\partial x_i$ parciális deriváltakat, majd szorzatra bontani ezeket, és a faktorok között megkeresni az alkalmas helyettesítő képleteket.

1. állítás. *Ha az x_i változó az $f(x)$ sima függvény képletében mindenütt a $h(x)$ kifejezés formájában fordul elő, akkor a $\partial f(x)/\partial x_i$ parciális derivált fölírható $(\partial h(x)/\partial x_i)\, p(x)$ alakban, ahol $p(x)$ folytonosan differenciálható.*

2. állítás. *Ha az x_i és x_j változók az $f(\mathbf{x})$ függvényben mindenhol egy $h(\mathbf{x})$ kifejezés formájában jelennek meg, akkor a $\partial f(\mathbf{x})/\partial x_i$ és $\partial f(\mathbf{x})/\partial x_j$ parciális deriváltak szorzatra bonthatók úgy, hogy $(\partial h(\mathbf{x})/\partial x_i)\, p(\mathbf{x})$, illetve $(\partial h(\mathbf{x})/\partial x_j)\, q(\mathbf{x})$, és $p(\mathbf{x}) = q(\mathbf{x})$.*

Ha $\partial f(\mathbf{x})/\partial x_i$ nem bontható szorzatra, akkor csak olyan alkalmas helyettesítést végezhetünk, amelynek képlete az x_i változó lineáris függvénye.

Az említett elméleti eredmények alapján egy olyan számítógépes programot lehet írni, amely korlátozás nélküli optimalizálási feladatok egyszerűsítésére automatikusan képes alkalmas helyettesítést keresni. Az implementációnk a következő lépéseket követi.

1. Határozzuk meg a célfüggvény gradiensét.
2. Faktorizáljuk a parciális deriváltakat.
3. Gyűjtsük össze az x_i-re vonatkozó lehetséges helyettesítési képleteket az l_i listába:
   (a) Inicializáljuk l_i-t az üres halmazzal.
   (b) Ha a faktorizálás eredményes volt $\partial f(\mathbf{x})/\partial x_i$-re nézve, akkor egészítsük ki az l_i listát a faktorok megfelelő integráljaival.
   (c) Bővítsük az l_i listát $f(\mathbf{x})$ olyan részkifejezéseivel, amelyek lineárisak x_i-ben.
   (d) Töröljük l_i azon elemeit, amelyek nem teljesítik az alkalmas helyettesítés föltételeit (az l_i-beli kifejezéseknek monotonnak kell lenniük x_i-re vonatkozóan).
4. Hozzunk létre alkalmas helyettesítéseknek $f(\mathbf{x})$-re való alkalmazásából egy S listát: $S = \bigcup l_i$, $i = 1, \ldots, n$.

5. Válasszuk ki a legkisebb műveletigényű elemet S-ből, ami az egyszerűsített célfüggvény lesz.

6. Oldjuk meg az egyszerűsített célfüggvény minimalizálási feladatát (ha lehetséges).

7. Határozzuk meg az eredeti feladat megoldását az inverz transzformáció alkalmazásával.

Az algoritmus által megkívánt lépések többsége (parciális differenciálás, szorzatra bontás, szimbolikus integrálás és helyettesítés) közvetlenül elérhetők a modern számítógépes algebra rendszerekben. Másrészt a Maple rendszerben írt első implementációnk azt mutatta, hogy még a piacvezető számítógépes algebra rendszerekben is komoly hiányosságok vannak a behelyettesítési képességek és az intervallum aritmetika végtelen intervallumokra való alkalmazása terén [4].

Egy függvény monotonitását intervallum aritmetikával is lehet ellenőrizni. Egy $f : I\!R^n \to I\!R$ függvény akkor szigorúan monoton, ha bármely $\mathbf{x}, \mathbf{y} \in I\!R^n$-re ha $\mathbf{x} < \mathbf{y}$, akkor $f(\mathbf{x}) < f(\mathbf{y})$ ($f(\mathbf{x}) > f(\mathbf{y})$). Ez ellenőrizhető az f deriváltja értékkészletével: amennyiben az nem tartalmazza a nullát, akkor f monoton. Az algoritmusunkban azt kell tisztázni, hogy a $h_i(\mathbf{x})$ helyettesítő képletre igaz-e, hogy $\partial h_i(\mathbf{x})/\partial x_i$ nem veheti föl a nulla értéket. Meg kell említeni, hogy a monotonitás intervallum aritmetikával való ellenőrzését nehezíti az intervallumos számítások gyakori túlbecslése (az, hogy a kapott korlátok sokszor nem élesek).

3 Implementáció és számítógépes eredmények

A [4] cikkben bemutatott Maple implementáció során felmerült – a programkörnyezetből adódó – hiányosságok kiküszöbölésére az eljárást megvalósítottuk Mathematicában is [3]. A korábbival összehasonlítva a Mathematicának több előnye is van. Először is, a mi célunkhoz szükséges helyettesítések sokkal jobban működnek, mivel a Mathematica programozási nyelve term-átíráson alapul [15]. Pontosabban, a Mathematica helyettesítő rutinja reguláris kifejezésekkel megadott term-átírási szabályokkal vezérelhető, ráadásul ezek a felhasználó által definiált szabályok magasabb precedenciát élveznek a rendszerbe építettekkel szemben [11]. A megírt speciális helyettesítő rutin mintegy ötven programsorból áll, ami a Mathematica elegáns, kifejező nyelvét ismerők számára önmagában is árulkodik összetettségéről. Tucatnyi (késleltetett) szabályt vezettünk be, amiket négy különböző módon értékelünk ki, bevonva a képlet Mathematica által kifejtett, egyszerűsített, és faktorizált alakjait is. Valószínűleg ez a rutin a program legfontosabb része, hiszen több fázisban is meghívódik, így kis javításokkal alapvetően megváltoztatható az egyszerűsítési folyamat eredménye.

A Mathematicában található intervallum aritmetika is megbízhatóbb: ez a helyettesítésre szóba jövő kifejezések gyors és megbízható értékkészlet-

becsléséhez különösen fontos. Az értékkészlet behatárolására szolgáló naiv intervallumos befoglalást a beépített intervallum aritmetikával valósítottuk meg.

Az új program támogatja az összes lehetséges helyettesítés felsorolását is, majd ezek közül a legkedvezőbb kiválasztását az algoritmusunk 4.-5. lépésében, futási idő tekintetében mégis felveszi a versenyt az egyszerűbb Maple változattal, amely mohó módon az első megtalált alkalmas helyettesítést szolgáltatta. Köszönhető ez a mindkét számítógépes algebra rendszer által kínált, de a Mathematicában alapvetőbb [12,23] funkcionális programozási paradigma alkalmazásának, és a Mathematica rendszer további kedvező tulajdonságainak, mint amilyen a lista műveletek automatikus párhuzamosítása. Ugyanakkor az összes lehetséges helyettesítésen értelmezett keresési térre vonatkozó korlátozás és szétválasztás jellegű stratégia megalkotása gyorsítana az eljáráson, ez egy eddig kiaknázatlan továbbfejlesztési lehetőség.

Készült egy webes alkalmazás is a bemutatott algoritmus képességeinek demonstrálására, ez a Mathematica programkóddal együtt a következő linken érhető el:

`http://www.inf.u-szeged.hu/~csendes/symbsimp/`

3.1 Előrelépés a Maple változathoz képest

A Mathematica implementáció hatékonyságát először is a saját teszthalmazon, abból is a Maple verzió számára problémás eseteken vizsgáltuk. A korábbi vizsgálatainkban szereplő tesztesetekre a Mathematicában elért eredményt az *1. táblázat* tartalmazza. Az ebben nem szereplő esetekben a két implementáció kimenete azonos volt. Az érdekesebb helyettesítések kiemelése érdekében az egyszerű átnevezéseket ($y_j := x_j$) nem tüntetjük fel. Jelölések: Azon.: a feladat azonosítója, Fel.: a feladat típusa, Ered.: az eredmény jellege. Utóbbi két jellemző tartalma: A feladat típusát A-nak definiáljuk, ha egyszerűsítő átalakítások adhatók a bemutatott elmélettel összhangban. A feladat típusa D, ha nem számítottunk semmilyen hasznos átalakításra. Az eredmény jellege azt mutatja, hogy a programunk korrekt helyettesítést adott (1), vagy semmilyen helyettesítést sem adott (2). A kategorizálás részletesebb volt korábbi közleményeinkben [2,3,4].

Azon.	f függvény	g függvény	Helyettesítések	Fel.	Ered.
Sin2	$2x_3 \cdot \sin(2x_1 + x_2)$	$2x_3 \sin(y_1)$	$y_1 = 2x_1 + x_2$	A	1
Exp1	$e^{x_1+x_2}$	e^{y_1}	$y_1 = x_1 + x_2$	A	1
Exp2	$2e^{x_1+x_2}$	$2e^{y_1}$	$y_1 = x_1 + x_2$	A	1
Sq1	$x_1^2 x_2^2$	semmi	semmi	D	2
Sq2	$(x_1x_2 + x_3)^2$	y_1^2	$y_1 = x_1x_2 + x_3$	A	1
SqCos1	$(x_1x_2 + x_3)^2 - \cos(x_1x_2)$	$y_1^2 - \cos(x_1x_2)$	$y_1 = x_1x_2 + x_3$	A	1
SqExp2	$(x_1 + x_2)^2 + 2e^1 e^{x_1+x_2}$	$y_1^2 + 2e^{1+y_1}$	$y_1 = x_1 + x_2$	A	1
SqExp3	$(x_1 + x_2)^2 + 2e^{1+x_1+x_2}$	$y_1^2 + 2e^{1+y_1}$	$y_1 = x_1 + x_2$	A	1

1. táblázat. A problémás saját tesztfüggvényeken elért eredmények a Mathematica implementációval

A Maple intervallum aritmetikája kapcsán jelentkező anomáliák miatt az eredeti implementációban heurisztikát alkalmaztunk az értékkészlet-becslésre. A felmerülő hibák másik nagy csoportját a gyenge helyettesítő rutin okozta. A következőkben részletesen kifejtjük az eredményekben jelentkező különbségeket. A *Sin2* problémára az új implementáció alkalmas helyettesítést talált, míg a régi egy összetettebb, de nem monoton helyettesítést szolgáltatott.

Exp2-ben $e^{x_1+x_2}$ értékkészlete nem a teljes valós számok halmaza, de a Maple-ben használt heurisztikus értékkészlet-becslés alkalmas helyettesítésként ismerte fel. A Mathematicában készült értékkészlet-becslő rutin jobban teljesített ebben az esetben.

Az *Sq1* esetén a Maple nem ismerte fel $x_1^2x_2^2$-ben a x_1x_2 képletet, habár *Sq2*-nél $x_1x_2+x_3$-at megtalálta a négyzetes alakban. Mivel az utóbbi a legmagasabb szinten szorzat helyett összeadás típusú, a reprezentációja különböző. Mathematicában hasonló az eset, de az egyedi helyettesítési szabályokból építkező speciális helyettesítő rutin jól vizsgázott erre a feladatra. Másrészt x_1x_2 nem monoton sem x_1, sem x_2 függvényeként a teljes keresési tartományban (amit $\mathbb{IR}$-nek feltételezünk), ezért nem volt alkalmas helyettesítésként megjelölve.

Hasonlóképp az *SqCos1* tesztfeladatra az új, Mathematica-alapú eljárás helyesen állapította meg, hogy $y_1 = x_1x_2$ nem monoton, ezért eltávolította a lehetséges helyettesítések listájából.

SqExp2-3 esetén ugyancsak a Maple mintaillesztés terén jelentkező gyengesége figyelhető meg, mivel $x_1 + x_2$-t felismerte $e^1e^{x_1+x_2}$-ben, $e^{1+x_1+x_2}$-ben azonban nem. A Mathematica implementációban nem jelentkezett ez a probléma. Összefoglalásul megállapíthatjuk, hogy a Mathematica alapú implementációnk lényeges javítást jelent a korábbihoz képest, és lényegében az elvárható módon működik.

3.2 Globális optimalizálási tesztfeladatokon elért eredmények

A Mathematica implementáció vizsgálata során a standard és egyéb gyakran használt globális optimalizálási tesztfeladatokat tartalmazó teszthalmazt némileg bővítettük a korábbi publikációhoz képest [4,3]. A legtöbb esetben az új eredmény megegyezik a korábbi megvalósításával. A két eltérést a *Schwefel-227 (Sch227)* és a *Schwefel-32 (Sch32)* függvény kétdimenziós alakjánál tapasztaltuk. A *Schwefel-227* feladatra a Maple változat az $y_1 = x_1^2 + x_2^2 - 2x_1$ helyettesítést adta. Ez a kifejezés jellemzi x_2 minden előfordulását, de nem monoton egyik változójában sem, ezért a Mathematica változat nem javasolta helyettesítésre. A Rosenbrock-feladattal rokon kétdimenziós *Schwefel-32* esetén a Mathematica talált alkalmas helyettesítést, míg a Maple nem.

Ezúttal az átalakítás futási idejét is mértük. Minden átírás a Mathematica 9.0-es verziójával készült, a futási időt fél órában maximáltuk. Azokban az esetekben, amikor a teljes egyszerűsítés nem zajlott le 1800 másodpercen belül, a futást megállítottuk. A numerikus teszteket egy Intel i5-3470 processzorral, 8 GB RAM-mal, és 64-bites operációs rendszerrel felszerelt számí-

tógépen futtattuk.

A teljesítmény profil értékelése alapján kijelenthető, hogy a futási idő nagy részét – nem meglepő módon – a szimbolikus képlet-átalakítások és a kiterjesztett helyettesítő rutin igényelte. Míg az 1. táblázat átalakításainak mindegyike kevesebb, mint 0,2 másodperc alatt lezajlott, a standard tesztfeladatok futási ideje sokkal nagyobb szórást mutatott. A 45-ből 24 teszteset kevesebb, mint egy másodperc alatt futott. További 10 esetben kevesebb, mint egy perc elegendőnek bizonyult. Viszont 7 esetben több, mint fél órát igényelt volna az automatikus egyszerűsítő.

Összesen 45 jól ismert globális optimalizálási tesztfeladatot vizsgáltunk, és ezek közül 8 esetben a Mathematica program talált ekvivalens átírást. Más szavakkal, a módszerünk a teszthalmaz 18%-ára ajánlott valamiféle egyszerűsítést. Tekintve, hogy ezekre a feladatokra nincs más ismert módszer, amely ilyen jellegű automatikus egyszerűsítéseket állítana elő, az eredmény figyelemre méltó. Megvizsgáltuk, hogy a produkált átírások milyen hatást gyakorolnak egy klasszikus numerikus multi-start megoldó teljesítményére. Az összes tesztfeladat átlagában a függvény-kiértékelések számának az átírásnak köszönhető relatív javulása 32,0% volt. Ugyanakkor ez a mutató a saját tesztfeladataink átlagában kedvezőbb (51,8%), a standard problémákon kevésbé jó (14,7%). A futási idők tekintetében az átalakított feladatalak szinte minden esetben egy kicsivel gyorsabb futást tett lehetővé a GLOBAL optimalizáló eljárás számára. Az egész teszthalmazra a futási idő átlagos relatív javulása 31,5%. E mutató a saját tesztfeladataink átlagában 56,9%, a standard problémákra pedig 9,3%.

3.3 Az intervallumos befoglalás javítása

Intervallumos módszerek [1] mind népszerűbbek olyan esetekben, amikor globális optimalizálási feladatok megbízható megoldására van szükség. Az intervallum aritmetikát támogató olyan korszerű számítógépes eszközök, mint a Matlab intervallumos kiegészítő csomagja, az Intlab, vagy magas szintű programozási nyelvek, amelyek mind az intervallumos befoglaló függvényeket, mind a deriváltakat automatikusan tudják generálni (pl. C-XSC) szélesítették a lehetséges alkalmazások körét. Mégis, az intervallumos számítások fő nehézsége, az úgynevezett túlbecslés nagyon meg tudja növelni az egyes feladatok megoldásához szükséges számítási időt.

A túlbecslésre az egyik gyakran használt elrettentő példa az $f(x) = x^2 - x$ függvény értékkészletének intervallumos befoglalása az ún. naiv intervallum aritmetika segítségével. Tekintsük az $X = [0, 1]$ intervallumot. Ezen az $f(x)$ értékkészlete nyilván $f(X) = [-0{,}25, 0]$, hiszen 0 és 1 épp $f(x)$ zérushelyei, és $f(x)$ minimuma a 0,5 pontban van, az értéke $-0{,}25$. Ezzel szemben $f(x)$ egy befoglaló függvénye $F(X) = X \cdot X - X$, amit a [0,1] intervallumon kiértékelve $[0, 1] \cdot [0, 1] - [0, 1] = [0, 1] - [0, 1] = [-1, 1]$-et kapunk. Ez az intervallum nyolcszor szélesebb, mint $f(x)$ értékkészlete a [0,1] intervallumon! Miközben mindössze két műveletet hajtottunk végre, és nyilvánvalóan a bonyolultabb függvények esetén nagyobb túlbecslésre számíthatunk.

A probléma részletes elemzésébe most nem tudunk belemenni, de arra szeretnénk rámutatni, hogy a problémát az okozza, hogy ha matematikai bizonyító erővel bíró befoglaló függvényeket hatékonyan akarunk kiszámítani számítógépen, akkor minden művelet végrehajtása során azt kell föltételeznünk, hogy az argumentum intervallumok függetlenek, azaz az azokban lévő valós számok bármely kombinációja előfordulhat. Ez nyilván nem teljesül a példánkban. Az intervallum aritmetika hüvelykszabálya szerint akkor lehet pontos befoglalást kapni, ha a függvényünk úgynevezett SUE típusú (single use expression), amely tehát minden változót csak egyszer használ a kiértékelés során. Ezt sajnos nem lehet föltételezni általában gyakorlati feladatok esetén, sőt, a nagy méretű modellezési feladatok összeállításában alkalmazott folyamatszintézis rendszerek [8] épp hogy erősen redundáns kifejezéseket produkálnak.

A cikkünkben leírt módszer alapjában véve az optimalizálandó függvényünk kiértékeléséhez szükséges műveletek számát akarja csökkenteni, és amennyiben ez lehetséges, fölismerni az eredeti alakban a redundanciát. Ez nem ugyanaz, mint ami az intervallumos kiértékelés pontosságának növeléséhez kell. A probléma megoldásának első lépéseként mégis azt vizsgáltuk meg, hogy a meglevő nemlineáris függvény egyszerűsítési transzformációnknak mi a hatása globális optimalizálási feladatok intervallumos módszerekkel való megoldására. A tesztelt algoritmus az intervallumos Newton-módszerrel kiegészített korszerű korlátozás és szétválasztás típusú globális optimalizálási algoritmus volt [18]. A megállási feltételbeli konstans legalább két értékes jegynyi relatív pontosságot követelt meg.

Tekintsük először a jól ismert Rosenbrock-feladatot (banánfüggvényként is ismert, a képlete $(1-x)^2+100(x^2-y)^2$, az egyetlen helyi minimumpontja nyilvánvalóan $(1,1)^T$, az optimum értéke nulla). Maga az egyszerűsítés elhanyagolható mennyiségű számítást kellett volna hogy megtakarítson, hiszen a két függvényalak közti eltérés a végzett műveletek számában nem jelentős. Az eredeti függvényalakra a következő eredményt kaptuk:

```
Function name:  ros2
The set of global minimizers is located in the union of the following boxes:
c1:[0.99487304687500, 1.00235210730573] [0.98876953125000, 1.00708007812500]
c2:[1.00271012643331, 1.00488165028086] [1.00097656250000, 1.01318359375000]
The global minimum is enclosed in:
  [0.000000000000000000, 0.000022260135021864]
Statistics:
     Iter   Feval   Geval   Heval   MLL   Time(sec)
      36     251     174      15     10      0.94
```

Megállapíthatjuk, hogy az eredmény helyes, a $c1$ intervallum tartalmazza a megoldást, és a kapott relatív pontosság is jó: 2-3 jegynyi a helyben, és hasonló az optimális célfüggvény értékben. Lássuk az átalakított feladatra kapott outputot:

```
Function name:  ros2v
The set of global minimizers is located in the union of the following boxes:
c1:[-0.00000000000000, 0.00000000000000] [-0.00000000000000, 0.00000000000000]
The global minimum is enclosed in:
  [0.000000000000000000, 0.000000000000000000]
```

```
Statistics:
     Iter   Feval   Geval   Heval   MLL   Time(sec)
      9      42      31      1      1      0.11
```

Ez nyilván az átalakított feladat eredménye, tehát a nulla értékek a helyben és az optimum értékében helyesek. A kapott pontosság olyan jó, hogy igazából össze sem vethető az eredetiével. A Matlab kiíratási rendszere ismeretében azt lehet mondani, hogy a kapott intervallumok eltérnek ugyan a [0,0] intervallumtól, de az eltérések mértéke olyan kicsi, hogy azt még a standardnál hosszabb számábrázolás sem tudja megmutatni. Minden esetre olyan tíz nagyságrenddel pontosabb az eredményünk. Ráadásul ezt a szép eredményt nem növekvő számítási ráfordítással értük el, hiszen minden hatékonysági mutató javult: 36 helyett elég volt 9 iteráció (Iter). A függvényhívások (Feval), a gradienshívások (Geval) és a Hesse mátrix számítások (Heval) száma is csökkent, rendre: 251-ről 42-re, 174-ről 31-re és 15-ről 1-re. Az is nagyon beszédes, hogy a földolgozatlan részintervallumok maximális száma (MLL) is kisebb lett: 10-ről 1-re. Ez azt jelzi, hogy a módszer a lehető legegyszerűbbnek találta az átírt feladatot. A CPU idő csökkenése ugyan összhangban van az előbb mondottakkal, de ennek kisebb a jelentősége, hiszen valódi feladaton inkább az előző mutatók a fontosak. Összességében azt mondhatjuk, hogy a kapott eredmény már-már gyanúsan kedvező.

Lássunk most egy másik standard globális optimalizálási feladatot, a Levy-10 nevűt.

```
Function name:  L10
The set of global minimizers is located in the union of the following boxes:
c1: [0.99999993564181, 1.00000006433416] [0.96591507795486, 1.04004632763272]
    [0.99366780273703, 1.12761120490455] [0.93750000000000, 1.19587646550836]
    [0.89381419541709, 1.00189285440539]
The global minimum is enclosed in:
  [0.000000000000000000, 0.000567456706858098 ]
Statistics:
     Iter   Feval   Geval   Heval   MLL   Time(sec)
      17     127     85      6      17      3.20
```

Az egyszerűsített feladatra kapott eredmény:

```
Function name:  L10v
The set of global minimizers is located in the union of the following boxes:
c1: [0.00000000000000, 0.00077514614094] [-0.01031504005235, 0.02046607537265]
    [-0.04137133829842, 0.01705999539191] [-0.04080756533673, 0.04296875000000]
    [-0.02127534259525, 0.01985113485206]
The global minimum is enclosed in:
  [0.000000000000000000, 0.000000000000000000]
Statistics:
     Iter   Feval   Geval   Heval   MLL   Time(sec)
     258    2130    1651    133     32     51.73
```

Megállapíthatjuk, hogy ez esetben a hatékonysági mutatók mind rosszabbak lettek, az átírt feladatra egy-két nagyságrenddel több számítógépes erőforrást kellett fordítani. Ennek ellenére ez is egy lelkesítő eredmény, mert ugyan a megállási feltétel változatlan volt, mégis ismét sokkal pontosabb eredményt kaptunk. A pontosság javulását is figyelembe véve a függvényhívások stb. száma fajlagosan lényegesen csökkent. Ennek valószínűleg az az

oka, hogy az átírt függvény kedvezőbb volt a Newton-lépés számára. Ezt a megállapítást a többi, itt nem részletezett tesztünk is alátámasztja.

A számítógépes vizsgálatokat megismételtük az olyan gyakran használt globális optimalizálási tesztfeladatokra, amelyen az egyszerűsítési eljárásunk változtatott. Az eredményeket a *2. táblázatban* foglaltuk össze. Tehát a következő lépéseket követtük: lefuttattuk a [18] cikkben leírt intervallumos globális optimalizáló algoritmust minden gyorsító teszttel (monotonitási teszt, konkavitási teszt, intervallumos Newton-lépés) az eredeti és az átírt feladaton. Ezután megmértük a globális optimum helyében és értékében kapott bizonytalanságot. Ezt azzal jellemeztük, hogy a kiindulási több-dimenziós intervallum méretéhez képest mekkora az eredmény intervallumok össztérfogata. A kapott értékeket százalékos formában jelenítettük meg. Az átírt feladatok eredményeit úgy kaptuk, hogy az átalakított kiinduló box térfogatához hasonlítottuk az eredmény térfogatát. Így a táblázatunk összevethető értékeket tartalmaz.

A 2. táblázatban használt jelölések a következők. Min. pont rel. mérete (%): a globális minimum pontot befoglaló boxok össztérfogatának a kiinduló boxhoz viszonyított százalékos aránya. Min. rel. mérete (%): a minimumot befoglaló box térfogatának a kiinduló boxhoz viszonyított százalékos aránya. A futtatási időt másodpercben mértük, és minden kísérletet 10 alkalommal megismételtünk. Maga az optimalizáló eljárás determinisztikus, a megismételt futtatásra csak a futásidő megbízható meghatározása végett volt szükség. Az átírt feladatok azonosítója végére egy "v" betűt írtunk, tehát az eredeti és az egyszerűsített feladatok egymás után következnek a táblázatban a könnyebb összevetés érdekében.

Azon.	Min. pont rel. mérete (%)	Min. rel. mérete (%)	Futási idő (mp)
Br	$6.4114 \cdot 10^{-4}$	$1.4059 \cdot 10^{-11}$	1.5304
Brv	$0.0000 \cdot 10^{0}$	$8.7349 \cdot 10^{-8}$	0.4243
L8	$6.3680 \cdot 10^{-10}$	$6.4578 \cdot 10^{-8}$	1.0499
L8v	$1.2049 \cdot 10^{-5}$	$1.9293 \cdot 10^{-31}$	4.2791
L9	$1.4414 \cdot 10^{-9}$	$5.3049 \cdot 10^{-8}$	1.8486
L9v	$1.3570 \cdot 10^{-7}$	$3.5078 \cdot 10^{-32}$	16.662
L10	$1.1151 \cdot 10^{-15}$	$1.7733 \cdot 10^{-8}$	3.0077
L10v	$9.5443 \cdot 10^{-11}$	$6.3778 \cdot 10^{-33}$	51.2230
L11	$3.7110 \cdot 10^{-30}$	$5.5522 \cdot 10^{-13}$	7.3071
L11v	$4.0440 \cdot 10^{-15}$	$3.8334 \cdot 10^{-35}$	1025.0000
Rb2	$7.3628 \cdot 10^{-7}$	$5.5650 \cdot 10^{-8}$	0.9220
Rb2v	$0.0000 \cdot 10^{0}$	$3.8670 \cdot 10^{-33}$	0.1123
Rb5	$1.1622 \cdot 10^{-11}$	$5.6901 \cdot 10^{-6}$	60.3720
Rb5v	$6.2214 \cdot 10^{-91}$	$2.2637 \cdot 10^{-7}$	36.8400
Sch3.2	$9.3274 \cdot 10^{-3}$	$4.4420 \cdot 10^{-4}$	0.3214
Sch3.2v	$0.0000 \cdot 10^{0}$	$0.0000 \cdot 10^{0}$	0.1014

2. táblázat. Futási eredmények az intervallumos globális optimalizáló eljárással ([18]), 10 futtatás átlagában

A 2. táblázat eredményei kiértékelése során tekintettel kell lennünk arra, hogy az algoritmust az alap beállításokkal futtattuk, változatlan formában.

Így nem is hangoltuk azt az adott helyzetre. Ennek következménye az, hogy a globális minimum helyében, illetve értéke korlátaiban nagyon eltérő pontosságú eredményeket kaptunk. Mindenesetre az megállapítható, hogy az egyszerűsített feladatokra kapott eredmények minden esetben hatalmas pontosságjavulást mutatnak vagy a helyben, vagy az értékben, vagy mindkettőben. Ezzel messze nem arányos a mutatkozó számítási idő igény. Tehát azt jelenthetjük ki, hogy az alkalmazott szimbolikus egyszerűsítési módszer sok nagyságrendnyi pontossági javulást okoz, ezzel össze nem mérhető, sokkal kisebb arányú számítási idő növekedés árán.

Az intervallumos Newton-módszer az intervallumos globális optimalizálási algoritmusok legkritikusabb eleme, jellemző módon más gyorsító technikákkal (például a monotonitási teszt) utolérhetetlen hatékonyságú is lehet, de nagy időpazarlást is okozhat. Tanulmányunk alapján azt az óvatos következtetést tudjuk levonni, hogy a szimbolikus egyszerűsítő transzformációk a megvizsgált standard globális optimalizálási feladatokon nagyon ígéretes hatékonyságjavulást mutattak. A jelen tanulmány csak a kezdetét jelzi a fejlesztési munkának, hiszen a talált rendkívüli pontosságnövelő hatás ügyes kiaknázási módjait még nem ismerjük. A következő vizsgálatok kiterjedt tesztelés után olyan átírási algoritmusokat keresnek majd, amelyek az intervallumos globális optimalizáló algoritmusok pontosságának növelésére, illetve hatékonyságuk javítására törnek. Ezzel együtt az intervallumos globális optimalizálási algoritmust is ennek megfelelően hangolni kell majd, hogy a megállási feltételekkel jól szabályozható legyen a pontosság növekedés nyeresége a számítási idő vonatkozásában.

Irodalom

1. Alefeld, G., Herzberger, J. *Introduction to Interval Computation*. Academic Press, New York, 1983.
2. Antal Elvira: A matematikai modellezés hatása nemlineáris optimalizálási feladatok megoldásának hatékonyságára. PhD értekezés, Szeged, 2017.
3. Antal, E., Csendes, T. Nonlinear Symbolic Transformations for Simplifying Optimization Problems *Acta Cybernetica* 22:715–733, 2016.
4. Antal, E., Csendes, T., Virágh, J. Nonlinear Transformations for the Simplification of Unconstrained Nonlinear Optimization Problems. *Cent. Eur. J. Oper. Res.* 21(4):665–684, 2013.
5. Avanzolini, G., Barbini, P. Comment on "Estimating Respiratory Mechanical Parameters in Parallel Compartment Models". *IEEE Trans. Biomed. Eng.* 29:772–774, 1982.
6. Csendes, T., Pál, L., Sendín, J. O. H., Banga, J. R. The GLOBAL Optimization Method Revisited. *Optimization Letters* 2:445–454, 2008.
7. Csendes, T., Rapcsák, T. Nonlinear Coordinate Transformations for Unconstrained Optimization. I. Basic Transformations. *J. Global Optim.* 3(2):213–221, 1993.

8. Farkas, T., Rév, E., and Lelkes, Z. Process flowsheet superstructures: Structural multiplicity and redundancy: Part I: Basic GDP and MINLP representations. In *Computers & Chemical Engineering*, 29(10): 2180–2197, 2005.
9. Fourer, R., and Gay, D. M. Experience with a Primal Presolve Algorithm. In Hager, W. W., Hearn, D. W. and Pardalos, P. M., editors, *Large Scale Optimization: State of the Art*, pages 135–154. Kluwer Academic Publishers, Dordrecht, 1994.
10. Gay, D. M. Symbolic-Algebraic Computations in a Modeling Language for Mathematical Programming. In Alefeld, G., Rohn, J. and Yamamoto, T., editors. *Symbolic Algebraic Methods and Verification Methods*, pages 99–106. Springer- Verlag, 2001.
11. Gaylord, Richard J., and Kamin C, Samuel N. Wellin, Paul R.: *An Introduction to Programming with Mathematica*. 2. kiad. New York, NY, Springer New York, 1996, 141–143. p. ISBN 978-1-4612-2322-1.
12. Heck, André: Introduction to computer algebra. In Introduction to Maple. New York, NY, Springer New York, 2003, 11. p. ISBN 978-1-4613-0023-6.
13. Hantos, Z., Daróczy, B., Csendes, T., Suki, B., and Nagy, S. Modeling of Low-frequency Pulmonary Impedance in the Dog. *J. of Applied Physiology* 68:849–860, 1990.
14. Liberti, L., Cafieri, S., Savourey, D. The Reformulation-Optimization Software Engine. *Mathematical Software – ICMS 2010, LNCS 6327*, pages 303–314, 2010.
15. Maeder, Roman E.: *The Mathematica Programmer*. Academic Press, 1994, 8. p. ISBN 978-0-12-464990-3.
16. Mészáros, Cs., Suhl, U. H. Advanced preprocessing techniques for linear and quadratic programming. *OR Spectrum* 25(4):575–595, 2003.
17. Pál, L. *Global optimization algorithms for bound constrained problems*. Ph.D. thesis, University of Szeged, 2011.
18. Pál, L. and Csendes, T. INTLAB implementation of an interval global optimization algorithm. *Optimization Methods and Software* 24:749–759, 2009.
19. Rapcsák, T., Csendes, T. Nonlinear Coordinate Transformations for Unconstrained Optimization. II. Theoretical Background. *J. Global Optim.* 3(3):359–375, 1993.
20. Schichl, H., Neumaier, A. Interval Analysis on Directed Acyclic Graphs for Global Optimization. *J. Global Optim.* 33(4):541–562, 2005.
21. Stoutemyer, D. R. Ten commandments for good default expression simplification. *J. Symb. Comput.* 46:859–887, 2011.
22. Tóth, B., Csendes, T. Empirical investigation of the convergence speed of inclusion functions. *Reliable Computing*, 11:253–273, 2005.
23. Wellin, P.: Functional programming. In *Programming with Mathematica: An Introduction*. Cambridge University Press, 2014, 115–188. ISBN 978-1-107-00946-2.
24. Wolfram Mathematica 9 Documentation Center, Mathematica Tutorial: Basic Internal Architecture: https://reference.wolfram.com/mathematica/ tutorial/BasicInternalArchitecture.html
25. Wolfram Mathematica. Software Development: CUDA and OpenCL support. http://www.wolfram.com/mathematica/new-in-8/cuda-and-opencl-support . [2017.01.12.].

NONLINEAR SYMBOLIC TRANSFORMATIONS FOR OPTIMIZATION PROBLEMS

With the advent of symbolic computer algebra systems the possibility has become clear for reforming the optimization problems into a more advantageous form. The advantages of such simplifications are many-folded. At one hand the respective functions can hopefully be evaluated with less operations. Then we can recognize such redundant relations among the variables that usually cannot be noticed. Knowing these allows us to determine the subspace that consists only of optimal points. And finally, due to the possible dimension decrease the iteration numbers of most of the optimization techniques can drop. The procedure does not require human interaction, hence large scale and complex problems can become solvable by applying our simplification tool. The present paper overviews our work done on this field and new results are demonstrated on the application of the novel algorithm to improve the efficiency and precision of interval arithmetic based optimization algorithms.