banhelyi.balazs_224_24

MTA DOKTORI ÉRTEKEZÉS

Számítógéppel segített bizonyítások és optimalizálási modellek dinamikai és fizikai feladatokra

BÁNHELYI BALÁZS

SZEGEDI TUDOMÁNYEGYETEM
TERMÉSZETTUDOMÁNYI ÉS INFORMATIKAI KAR
INFORMATIKAI INTÉZET

Szeged
2024

banhelyi.balazs_224_24

Tartalomjegyzék

Előszó

Az értekezés két témakörben elért eredményeket tárgyal. Ezek a témakörök a megbízható számítások, illetve az optimalizálási eljárások fejlesztése és alkalmazása. Már a doktori képzésem alatt is ezekkel a szakterületekkel foglalkoztam, sőt ez a két terület sok esetben össze is ért, mivel mind a kettő szükséges volt bizonyos eredmények eléréséhez. A PhD képzés során értem el az első komolyabb eredményeimet a káosz keresésben az optimalizálás alkalmazásával. Ezeket a témákat Garay Barna és Csendes Tibor ismertette és szerettette meg velem. Az első eredményeket a klasszikus Hénon-leképezés kaotikus régióinak vizsgálatával értem el [9, 28]. Ezekben a munkákban az optimalizálást kaotikus régiók bizonyításához szükséges geometriai alakzatok keresésére használtam.

A doktori képzés alatt a megbízható számítások területét tovább erősítette a Hatvani László által megismertetett probléma, mely a kényszerrezgéses fékezett inga kaotikus viselkedésének a bizonyítása volt. Az eredeti cikkben John Hubbard csak egy sejtést fogalmazott meg, matematikailag elfogadható teljes bizonyítást nem tudott adni [47]. Ennek a problémának a megoldása [12, 27] olyan élmény volt számomra, hogy a mai napig keresem az igazán érdekes, mások által felvetett megoldatlan problémákat. Ezért első ránézésre a doktori utáni témaválasztásaim kaotikusnak tűnhetnek, de tulajdonképpen mindegyik egy-egy érdekes probléma, melyeken az általam ismert eszközök jól alkalmazhatók és velük magasabb szintű eredményeket lehet elérni.

Az első ilyen kiemelkedő munkámnak a Wright-sejtés [93] igazolásához vezető eredményünk rám eső részét tartom. A problémát Krisztin Tibor által ismertem meg. Ezen a területen már a doktori képzés alatt is értem el kisebb eredményeket, melyek akkor még nem igazán tűntek ígéretesnek [6]. Az áttörés 2010-ben történt, mikor már látszódott, hogy a sejtés egy jó részét tudjuk bizonyítani, jóval tovább menve a Wright által feltételezhetően megoldott tartományon. A probléma nehézségét és számítási igényét jelzi, hogy a teljes munkánk közel 5 évbe telt [14]. A megoldás értékét jól illusztrálja a cikkel elnyert, az intervallum analízis alkalmazásáért odaítélt nagy presztízsű nemzetközi Moore-díj, melyet a társszerzőimmel 2016-ban vehettünk át. Az ezzel kapcsolatban végzett munkámat az 1. fejezetben mutatom be.

A PhD dolgozatomban szerepelt a kényszerrezgéses fékezett inga kaotikus régiója létezésének megbízható bizonyítása, de több kérdés nyitva maradt ezzel kapcsolatban. Ilyen például a régió kaotikus trajektóriáinak megkeresése, illetve a rendszer lehetséges kontrolljai. Az előbbit megoldó eljárást doktorandusz hallgatómmal, Lévai Balázzsal közösen publikáltuk [55], melyhez úgyszintén a megbízható számítások és az optimalizálás kellett. Az utóbbit Csendes Tibor és Hatvani László társszerzőimmel publikáltuk [13], az ehhez kapcsolódó eredményemet a 2. fejezetben mutatom be. Lévai Balázzsal és Palati-

nus Endrével közös munkánk az optimális körfedés területén elért eredményünk is [17], melyet a 3. fejezetben mutatok majd be. Ebben ismét az optimalizálásnak és a megbízható számításoknak van fontos szerepe.

A következő, negyedik fejezetben a jelenleg is aktívan kutatott területemet mutatom be, ami a mesterséges neuronhálók verifikálása. Első hallásra ez a probléma csak távolról csatlakozik a korábbi kutatási területeimhez, de a verifikálás során több megbízható eljárást, illetve a gyorsításhoz optimalizálást alkalmaznak, így a korábbi tapasztalataimnak nagy hasznát veszem. Az első nagyobb eredményünk ezen a területen kimondottan a tapasztalataimon alapult, miszerint az MIT kutatói által megbízhatónak mondott verifikáló [80] sebezhetőségére hívtuk fel a szakmai közösség figyelmét [96]. Ennek a hiányosságnak a teljes értékű kimutatásában nagy szerepe volt doktorandusz hallgatómnak, Zombori Dánielnek. Az ezzel kapcsolatos eredményeket a 4. fejezetben fogom bemutatni.

A kutatások során először még a - mások által fejlesztett - GLOBAL algoritmust használtam, mely C nyelven íródott. Már ekkor láttam, hogy ez az algoritmus mely területeken lehet hatékony, de észrevettem azt is, hogy a kor szoftverfejlesztési szellemének ez már nem tesz eleget. Az algoritmus implementációja elavult volt, az egyedi igényekre való hangolása pedig sok esetben nehézkes, vagy inkább lehetetlen volt. Az évek során több esetben is nagy számításigényű feladattal találkoztam, így a párhuzamosításra is egyre nagyobb hangsúlyt fektettem, mellyel kapcsolatban jelentek meg publikációim [4, 7, 18]. Az eredeti GLOBAL optimalizáló algoritmus egy erősen szekvenciális eljárás, de a problémákon elért eredmények miatt általam rengeteget használt eljárás lett. Ennek a párhuzamosítására több megoldást is elkészítettünk Zombori Dániel doktorandusz hallgatómmal [16, 95], melyek eredményeként az 5. fejezetben tárgyalt algoritmus bizonyult a leghatékonyabbnak az általam vizsgált problémákra. A fejezet végén több olyan eredményt is bemutatok, melyet a GLOBAL-lal értem el, illetve pár olyan értékes eredményt is, amelyek véleményem szerint a GLOBAL párhuzamos változata nélkül nem jöttek volna létre azok extrém nagy számítási igénye miatt. Ezeknek a feladatoknak a nagy részét fizikus társszerzőim vetették fel, melyekhez sok esetben a párhuzamosításon kívül további módosítások is szükségesek voltak a GLOBAL-ban. Ezek a változtatások ebben az új elméleteken alapuló implementációban könnyedén megvalósíthatók voltak.

1. fejezet

A Wright-sejtés bizonyításával kapcsolatos eredmények

Jelen részben egy bő fél évszázada felvetett [93], egyszerűnek tűnő problémát fogunk megvizsgálni. Feladatunk azt eldönteni, hogy egy, a differenciálegyenletek jobb oldalán szereplő paraméter függvényében hogyan viselkedik a differenciálegyenlet megoldása, trajektóriája. Hasonló, de páratlan függvénnyel adott jobboldalú esetekre már léteznek eredmények [52, 53], de jelen nem páratlan esetekkel kapcsolatban több még bizonyítandó feltételezés volt.

Ez a probléma különösen felkeltette az érdeklődésemet, mellyel még a doktori képzés alatt találkoztam. Kicsit szkeptikusan álltam hozzá ehhez a feladathoz, hogy van-e az eszköztáramban bármi, ami előrébb vihet a megoldásban. A kényszerrezgéses inga kaotikusságában szerzett tapasztalataimon felbuzdulva nem teljesen vetettem el a problémát.

Az évek során több lehetséges utat is kipróbáltunk Krisztin Tibor és főként Arnold Neumaier ötletei alapján. Sajnos ezek az ötletek egyike sem hozta a várt eredményt, sőt még a Wrigth által állított, nem kidolgozott, csak sejtett értéket sem tudtuk elérni a paraméterben. De minden egyes kísérlet egyre közelebb vitt ahhoz a gondolathoz, hogy a trajektóriák zérushelyeinek lokális vizsgálata nem elegendő az érdemi előre lépéshez.

Sajnos a késleltetett differenciálegyenletek megoldására nem volt ismert olyan numerikus eljárás, mely matematikai bizonyításokban is használható. Hagyományos differenciálegyenletekre voltak megbízható eljárások, úgy mint például a VNODE [62, 63], COSY [57] vagy a régebbi AWA és hasonló eljárások [35, 56]. Az ezekben az algoritmusokban alkalmazott módszer kiterjeszthető bizonyos megkötésekkel a jelenlegi késleltetett differenciálegyenletünk megoldására is.

A Krisztin Tibor által felvázolt lehetséges részeredmények kiterjesztése során több kísérletet is tettünk a bizonyítás elérésére. Ezek során kifejlesztettem a jelen esetünkre egy megbízható eljárást [6] a késleltetett differenciálegyenletünk megoldásának befoglalására. Ezzel a módszerrel, illetve a jelenlegi fejezetben bemutatásra kerülő zérushelyek összekapcsolására vonatkozó ötletem során kapott jobb korlátozó függvényekkel sikerült olyan eljárást kapnunk, melyet sikeresen alkalmaztunk az adott problémára.

1.1. Wright-sejtés és eredete

Sir Edward Maitland Wright (1906–2005) angol matematikus a számelmélet egyik kiemelkedő alakja volt. Legismertebb munkája a G.H. Hardy-val közös Bevezetés a számelméletbe (An Introduction to the Theory of Numbers) című könyve, mely az 1938-as első kiadása óta hat új kiadásban jelent meg [44]. Az 1940-es években sok matematikus foglalkozott a prímszámtétel elemi bizonyításának lehetőségével. Hadamard és de la Vallé-Poussin francia matematikusok 1896-ban már megmutatták, hogy $\pi(n)$, ami a prímszámok számát jelenti n-ig, aszimptotikusan $n/\log(n)$, azaz a hányadosuk egyhez tart, ha n tart végtelenbe:

$$\lim_{n\to\inf} \frac{\pi(n)}{\frac{n}{\log(n)}} = 1.$$

A bizonyítás a komplex analízis és a Riemann-féle zéta-függvény eszközein alapult, és nem volt nyilvánvaló kapcsolata a prímszámok hétköznapi értelemben ismert tulajdonságaival.

Elemi bizonyítást Erdős [36] és Selberg [73] adtak 1949-ben, de az egyik legbiztatóbb elképzelés egy valószínűségszámítási megközelítés volt. Lord Cherwell felvetését követve Wright feltételezte a prímszámok eloszlásának bizonyos véletlenszerű tulajdonságát, és belátta, hogy ekkor a prímszámtétel éppen azt jelentené, hogy a

$$z'(t) = -\alpha z(t-1)\left(1+z(t)\right),\ \alpha \in \mathbb{R}^+$$

differenciálegyenlet -1-nél nagyobb megoldásai mind nullához konvergálnak az $\alpha = \log(2)$ esetben. Wrightnak annyira megtetszett ez a differenciálegyenlet, hogy egy egész cikksorozatot szentelt neki, amiben 1946 és 1960 között számos érdekes tulajdonságot bizonyított be az egyenlettel kapcsolatban. Ez az egyenlet később a Wright-féle késleltetett differenciálegyenlet néven terjedt el. Bár ezt az egyenletet korábban Hutchinson vezette be és késleltetett logisztikus egyenletnek hívták. Ez az egyenlet nagyon egyszerű, nemlineáris differenciálegyenlet időkésleltetéssel. Az egyenlet egy végtelen dimenziós dinamikai rendszer, melynél a fázistér egy függvénytér, a $[-1,0]$-án folytonos függvények Banach-tere. A népszerűségét talán annak is köszönheti, hogy látszólagos egyszerűsége ellenére nagyon bonyolult a viselkedése és megoldásainak jellemzése nehézkes.

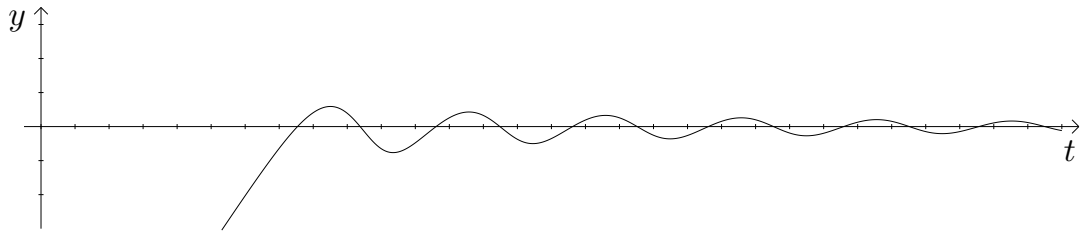
A megfelelő átalakításokkal a Wright-féle késleltetett differenciálegyenlet egy másik, vele ekvivalens, de a jobb oldalon már csak késleltetést tartalmazó alakra hozható. Ha $z(t) \geq -1$ (számunkra csak ez az eset érdekes), akkor tekintsük a $z(t) = e^{y(t)} - 1$ helyettesítést. Ezzel $z'(t) = e^{y(t)}y'(t)$ és $z(t-1) = e^{y(t-1)} - 1$. Így az alábbi egyenletet kapjuk:

$$e^{y(t)}y'(t) = -\alpha\left(e^{y(t-1)} - 1\right)\left(1 + e^{y(t)} - 1\right),$$

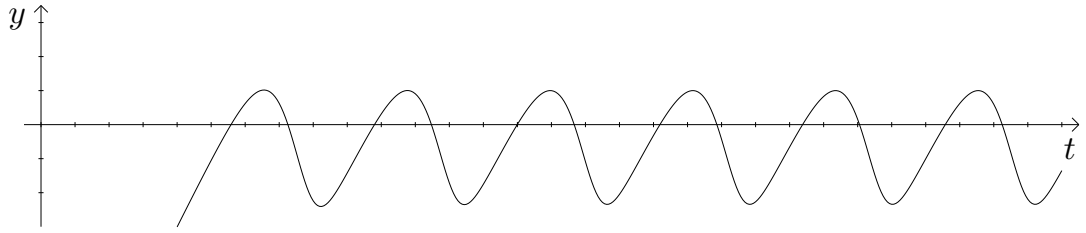
melyből egyszerűsítés után:

$$y'(t) = -\alpha\left(e^{y(t-1)} - 1\right). \tag{1.1}$$

Tetszőleges indulófüggvény és $\alpha \leq 1.5$ esetén ismert [93], hogy a trajektória oszcillálva konvergál a nullához (lásd az 1.1(a) ábrát az $\alpha = 1.5$ és $\phi(s) \equiv -11$ indulófüggvény esetére).



(a) Az $\alpha = 1.5$ eset.



(b) Az $\alpha = 2.0$ eset.

1.1. ábra. Közelítő ábrák az (1.1) egyenlet trajektóriájára.

Wright azt is állította, hogy a módszerét tovább folytatva $\alpha \leq \frac{37}{24}$ is elérhető, és esetleg még $\alpha \leq 1.567$ is. Ma már azt is tudjuk, hogy $\alpha = \frac{\pi}{2}$-ben a nulla megoldás elveszti a stabilitását (úgynevezett bifurkáció történik), és megjelennek periodikus megoldások, melyekhez tartanak a megoldások (lásd az 1.1(b) ábrát az $\alpha = 2.0$ és $\phi(s) \equiv -11$ esetre). Az 1955-ös eredeti Wright-sejtés azt mondja ki, hogy az összes megoldás nullához konvergál az $\alpha \leq \frac{\pi}{2}$ értékre, mely a korábbi numerikus eredmények alapján sejthető volt, hogy igaz is.

Bár maga az egyenlet látszólag egyszerű, de a késleltetéses $z(t-1)$-es tagja miatt, valamint a nemlinearitás miatt a sejtés továbbra is nyitott volt. Annak ellenére, hogy az ilyen differenciálegyenletekkel foglalkozó kutatók mindegyike előtt nagyon jól ismert volt a probléma, még a Wright által állított $\alpha \in \left[\frac{3}{2}, \frac{37}{24}\right]$ esetet sem sikerült igazolni. Sőt a 80-as években még hibás bizonyítások is megjelentek.

A Wright-sejtés bizonyításának módszere az úgynevezett globális attraktorok megadásán és teljes jellemzésén alapult. A $\frac{\pi}{2}$-nél nagyobb paraméterekre a globális attraktorokról sokat tudunk, de továbbra sem ismert a teljes jellemzésük. A társszerzőim egyik elméleti eredménye az, hogy a Wright-sejtés igazolásához elegendő a lassan oszcilláló periodikus megoldások létezését kizárni. Ez egyébként korábban nem volt ismert, Wright sem tudott róla. A lassan oszcilláló periodikus megoldások egyszerű megoldások, amelyeknek a periódusát három egymás utáni zéróhely határozza meg, és a zérushelyek távolsága 1-nél nagyobb. De a periodikus megoldás nemlétezésének igazolása nehéz, még közönséges differenciálegyenletekre is. Késleltetett differenciálegyenletekre a nemlineáris függvény bizonyos tulajdonsága mellett volt már ilyen bizonyítás, de nem gyakori a szakirodalomban. A bizonyításunk során a Wright-féle végtelen dimenziós dinamikai rendszernek a periodikus megoldásainak nemlétezését sikerült véges dimenziós feladattá redukálni. A nemlétezés problémáját bizonyos paramétertartományra intervallumaritme-

tikai eszközökkel sikerült igazolni. Melyet a fejezet további részében mutatok be.

1.2. Wright módszerén alapuló bizonyítás az $\alpha \leq 1.0$ esetre

Tekintsük a Wright-féle késleltetett differenciálegyenletet az alábbi formában:

$$y'(t) = -\alpha \left(e^{y(t-1)} - 1\right). \tag{1.2}$$

Bizonyítani fogjuk, hogy nem létezik a konstans nullán kívül más periodikus pálya. Abban az esetben, ha nem létezik ilyen, a konstans 0-tól eltérő periodikus pálya, akkor egyéb eredményekből tudjuk, hogy a számunkra érdekes esetekben nullához fog tartani a megoldás. A nem azonosan nulla periodikus pályák nem létezését indirekten fogjuk bizonyítani.

Kezdetben vegyük észre a differenciálegyenlet trajektóriájának néhány hasznos tulajdonságát. Tételezzük fel, hogy létezik periodikus pálya és tekintsünk egy ilyet. Ha $y(t-1) < 0$ és $\alpha > 0$, akkor az (1.2) képletből következik, hogy $y'(t) > 0$. Pozitív derivált esetén kétfajta trajektória lehetséges: a függvény nem marad végtelen hosszú ideig a negatív tartományban, vagy alulról konvergál egy nem pozitív számhoz. Mivel egy periodikus megoldást tekintünk, így csak az első eset lehetséges. Ha $y(t-1) > 0$ és $\alpha > 0$, akkor az $y'(t) < 0$. Az előzőhöz hasonló gondolatmenettel igazolható, hogy a pozitív tartományban sem maradhat végtelen hosszú ideig egy periodikus megoldás. Az utóbb két észrevételből következik, hogy a periodikus pályáknak a nulla pont körül kell oszcillálniuk. Feltehetjük, hogy a periodikus pálya szélsőértékei M, illetve $(-m)$ $(M, m > 0)$, azaz a függvény minden t időpontjára igaz, hogy

$$(-m) \leq y(t) \leq M. \tag{1.3}$$

Mivel a függvény a nulla körül oszcillál, így létezik olyan zéruspontja, melynek környezetében $y(t)$ előjelet vált. Legyen t_0 egy ilyen időpillanat, azaz $y(t_0) = 0$ és legyen a függvény t_0 előtt negatív, míg t_0 után pedig pozitív. Az (1.2) képletből következik, hogy a $(t_0 + 1)$ pontban az $y'(t)$ előjelet vált. Ezen tulajdonság miatt a $(t_0 + 1)$ pont egy maximumpont, azaz $y(t_0 + 1) = M$. Az előző gondolatmenethez hasonlóan látható, hogy ha az $y(t_0) = 0$ és a t_0 környezetében $y'(t) < 0$, akkor $y(t_0 + 1) = -m$, mely egy minimumpont.

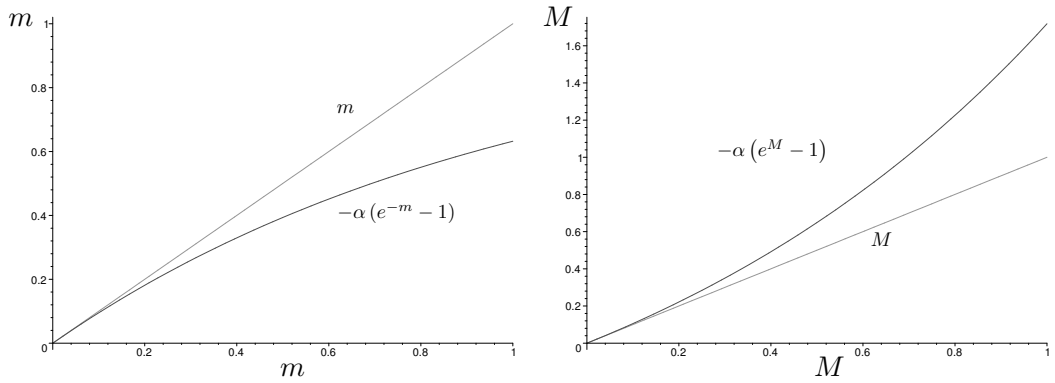
Először ismerjük meg Wright bizonyításának [93] lényegét az $\alpha \leq 1.0$ esetben. Vizsgáljuk meg, milyen feltételeket kaphatunk a periodikus pálya szélsőértékeire.

1.1. Állítás. *A periodikus pályák szélsőértékeire az alábbiak igazak:*

$$\textbf{1.} \quad M \leq -\alpha \left(e^{-m} - 1\right); \tag{1.4}$$

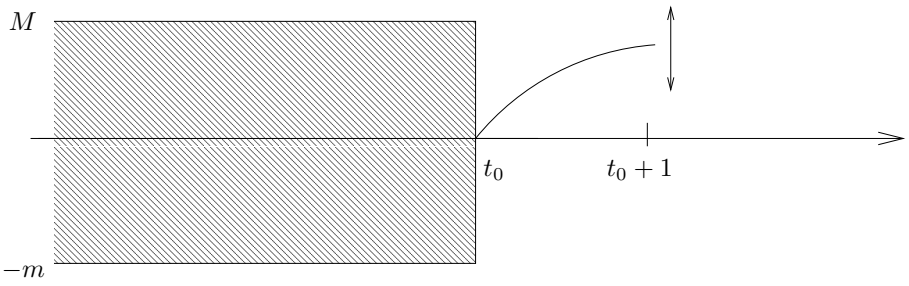
$$\textbf{2.} \quad m \leq \alpha \left(e^{M} - 1\right). \tag{1.5}$$

Az 1.2. ábrán láthatjuk az állítás összefüggéseit az $\alpha = 1.0$ esetben. Az 1.3. ábra a függvény alakulását mutatja a $(t_0 + 1)$ időpontig. A függvény a t_0 pont előtt a satírozott



(a) m eset. (b) M eset.

1.2. ábra. Korlátok az $\alpha = 1.0$ esetben.



1.3. ábra. A bizonyítás lényege az $\alpha \leq 1.0$ esetben.

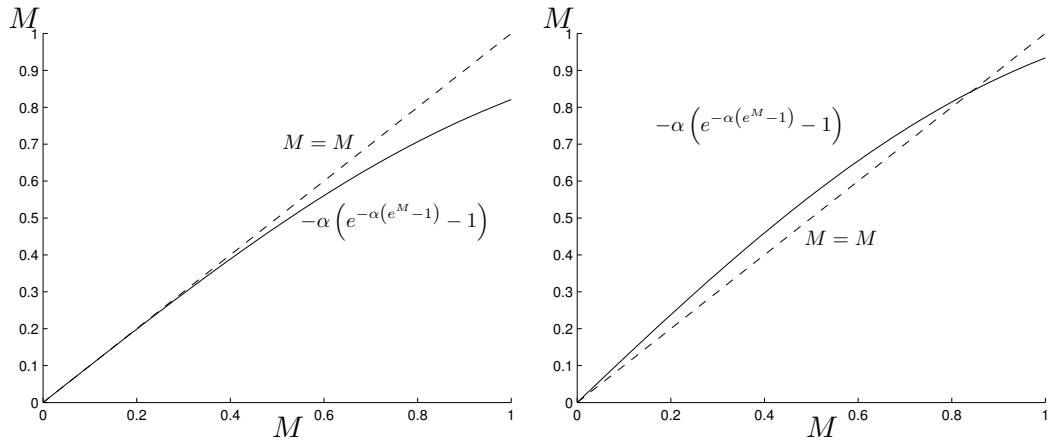
területben kell, hogy legyen. A t_0 időpillanatban a függvény értéke nulla. Ha a t_0 előtti szakaszt $(-m)$-mel korlátozzuk alulról, akkor az integráláskor a t_0 utáni résznek egy felső korlátját kapjuk. A feltétel szerint a (t_0+1) időpillanatban ezen felső becslésnek legalább M-nek kell lennie.

Mint látható, a M értékére felső korlátot ad m, és fordítva. Az (1.4) feltételbe behelyettesítve az (1.5) feltételt egy felső korlátot kapunk M-re M függvényében:

$$M \leq -\alpha \left(e^{-\alpha\left(e^M - 1\right)} - 1 \right). \tag{1.6}$$

Ezt a M-et korlátozó függvényt az 1.4. ábrán láthatjuk. (A megfelelő behelyettesítéssel egy hasonló összefüggést kaphatunk m-re.)

Ez a feltétel bármilyen periodikus pályára igaz, azaz olyan M-et kell találni, mely esetén a jobb oldal nem ad kisebbet, mint M. Az 1.4(a) ábrán látható, hogy az $\alpha \leq 1.0$ esetben csak a $M = 0$ elégíti ki ezt a feltételt, amiből következik az is, hogy $m = 0$. Az 1.4(b) ábrából sejthető, hogy az $\alpha > 1.0$ esetben léteznek olyan $M, m > 0$ számok is, melyekre igazak a megfelelő feltételek. Ezeket az észrevételeket Wright elméleti eszközökkel bizonyította.



(a) $\alpha = 1.0$ eset. (b) $\alpha = 1.1$ eset.

1.4. ábra. Korlátok M-re különböző α értékek mellett.

1.3. Wright módszerén alapuló bizonyítás az $\alpha \leq 1.5$ esetre

Wright erősebb feltételeket is adott a periodikus pálya szélsőértékeire. Ezek segítségével a sejtést bizonyította az $\alpha \leq 1.5$ esetre is. A továbbiakban ezen, erősebb feltételek levezetését ismertetjük.

1.2. Állítás. *A periodikus pályák szélsőértékeire az alábbiak igazak:*

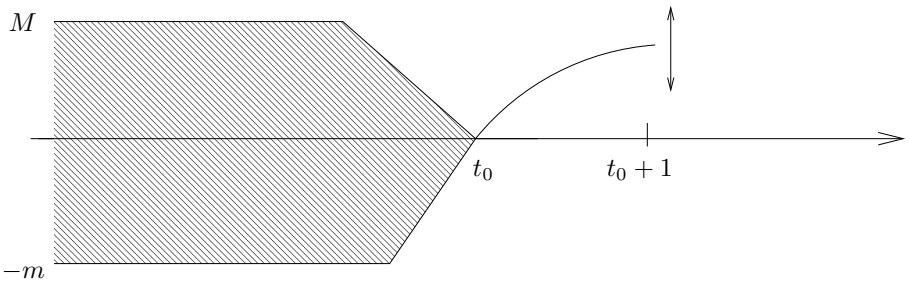
1. ha $-m \geq \alpha\left(e^{-m}-1\right)$, akkor $M \leq -\alpha\left(e^{-m}-1\right)+(-m)\frac{e^{-m}}{e^{-m}-1}-1$; (1.7)

2. $$M \leq \alpha - \frac{1-e^{\alpha\left(e^{-m}-1\right)}}{e^{-m}-1}; \tag{1.8}$$

3. $$m \leq \alpha\left(e^{M}-1\right) - M\frac{e^{M}}{e^{M}-1}+1. \tag{1.9}$$

Az 1.3. ábrához hasonlóan a jelen bizonyítás lényegét az 1.5. ábra illusztrálja. A M esetén két különböző korlátot kaptunk ((1.7), (1.8)). A derivált segítségével a periodikus megoldásra kapott új korlát, és az eredeti $y(t) \geq -m$-es korlát metszéspontjának elhelyezkedése alapján kaptuk a tárgyalt két esetet. Kis m-ek esetén e metszéspont a (t_0-1) után van, ekkor kaptuk az (1.7) esetet. Ha a metszéspont a (t_0-1)-nél korábban van, akkor a deriváltból jövő korlátot használtuk, így kaptuk az (1.8) korlátot. A m becslése esetén a t_0 pont előtti függvényre adott korlátok mindig a (t_0-1) pont után metszik egymást, ezért csak az első eset alkalmazható.

Az 1.4. ábrán látható, hogy csak kicsi M és kicsi m esetén nem működött az $\alpha \leq 1.0$ esetben használt módszer. Ezért vizsgáljuk meg külön ezt az esetet. Kicsi m esetén a $-m \geq \alpha\left(e^{-m}-1\right)$ igaz, és az (1.7) korlátot használhatjuk. Azaz alkalmazzuk egymás után az (1.7) és az (1.9) korlátokat. Ezzel egy korlátot kapunk a M értékére a M függvényében. Az 1.6. ábrából sejthető, hogy ez a korlát csak az $\alpha = 1.5$-ig igazolja a sejtést. Ha nagyobb α-ra is bizonyítani akarjuk a sejtést, akkor még erősebb korlátokat kell kidolgoznunk.



1.5. ábra. A bizonyítás lényege az $\alpha \leq 1.5$ esetben.

1.4. A periodikus pályák további számítható korlátai

Tekintsünk egy tetszőleges $y(t)$ periodikus megoldást. Vegyük a görbe t_0 zéruspontjai és a $(t_0 + 1)$ maximumpontok közötti részeit. Az ilyen egységnyi hosszú, monoton növő részeket nevezzük „$(inc, 1)$"-nek. Ezután tekintsük a függvény csökkenő részét a következő zéruspontig, melyek hosszát nem tudjuk pontosan meghatározni. Ezeknek a szakaszoknak a neve legyen „(dec, n)". Az előzőekhez hasonlóan az egy hosszúságú csökkenő részeket „$(dec, 1)$"-nek és az utána következő növekvő részeket „(inc, n)"-nek nevezzük.

Definiáljuk az alábbi korlátfüggvényeket az $y(t)$-re:

$y^{(upper)}_{(dec,n)}(t)$: felső korlát a $t'_0 - 1 \leq t \leq t'_0$ esetben,

$y^{(lower)}_{(inc,n)}(t)$: alsó korlát a $t''_0 - 1 \leq t \leq t''_0$ esetben,

$y^{(upper)}_{(inc,1)}(t)$: egy felső korlát a $t_0 \leq t \leq t_0 + 1$ esetben,

$y^{(lower)}_{(dec,1)}(t)$: alsó korlát a $t'_0 \leq t \leq t'_0 + 1$ esetben.

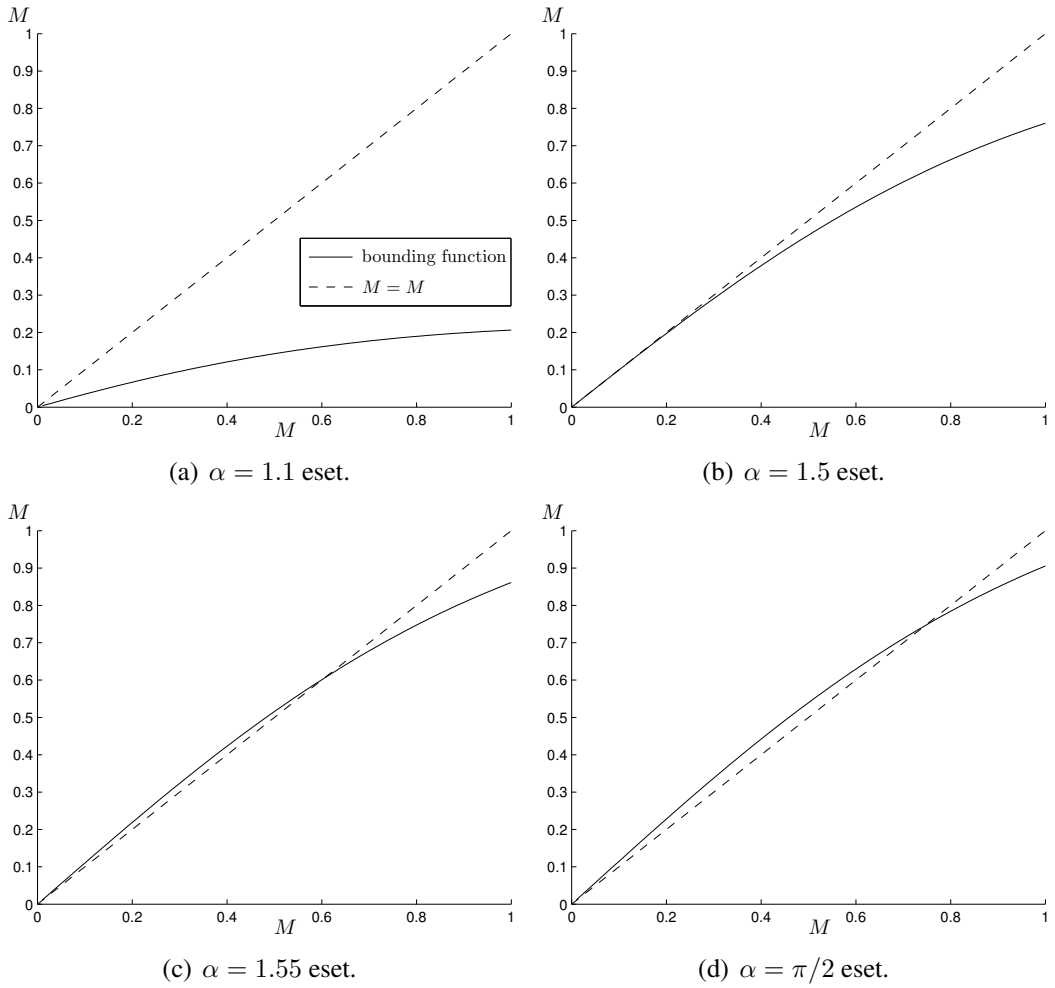
$y^{(lower)}_{(inc,1)}(t)$: alsó korlát a $t_0 \leq t \leq t_0 + 1$ esetben,

$y^{(upper)}_{(dec,1)}(t)$: felső korlát a $t'_0 \leq t \leq t'_0 + 1$ esetben.

A definíció szerint ezen y függvények korlátot adnak a megfelelő részen a periodikus megoldásra. A t_0, t'_0 és t''_0 legyenek rendre egymást követő zéruspontjai a periodikus megoldásnak. Mivel a vizsgált megoldás periodikus, így t_0-t tetszőleges olyan zéruspontnak tekinthetjük, amelyben a függvény monoton nő.

Tudjuk, hogy a periodikus megoldás $(dec, 1)$ és $(inc, 1)$ szakaszai egységnyi hosszúak, míg a (dec, n) és az (inc, n) szakaszról nem ismert hasonló információ. Ezeknek a szakaszoknak a hossza legyen rendre p_M, illetve p_m. Ekkor $(1 + p_M)$ és $(1 + p_m)$ a két zéruspont közötti távolság, azaz a $t'_0 = (t_0 + 1 + p_M)$ és a $t''_0 = (t'_0 + 1 + p_m)$.

Mivel a (dec, n) és (inc, n) hosszát nem ismerjük, így a $(p_M - 1)$, illetve a $(p_m - 1)$ előjelét sem ismerjük. Több felső korlát is ismert rájuk, de ezek számunkra nem használhatók. Ezen p értékekre jelen esetünkben csak az 1 körüli értékek lesznek ideálisak,



(a) $\alpha = 1.1$ eset.

(b) $\alpha = 1.5$ eset.

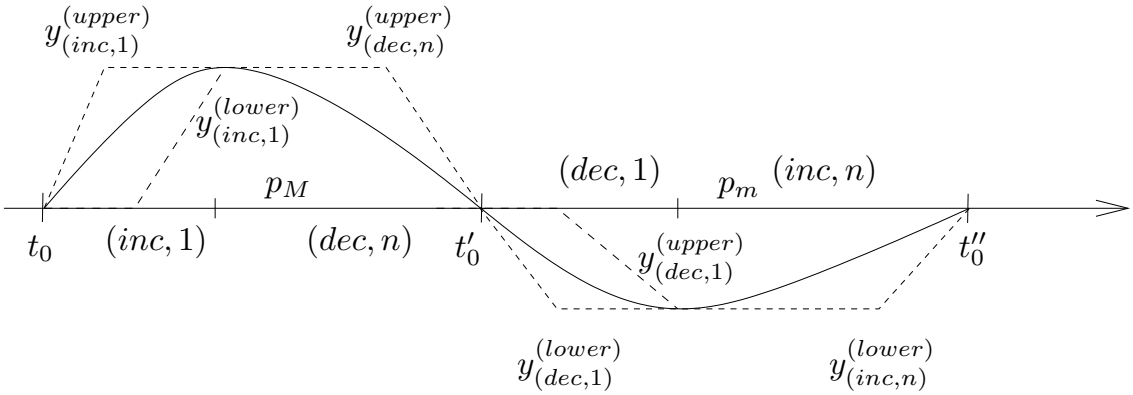
(c) $\alpha = 1.55$ eset.

(d) $\alpha = \pi/2$ eset.

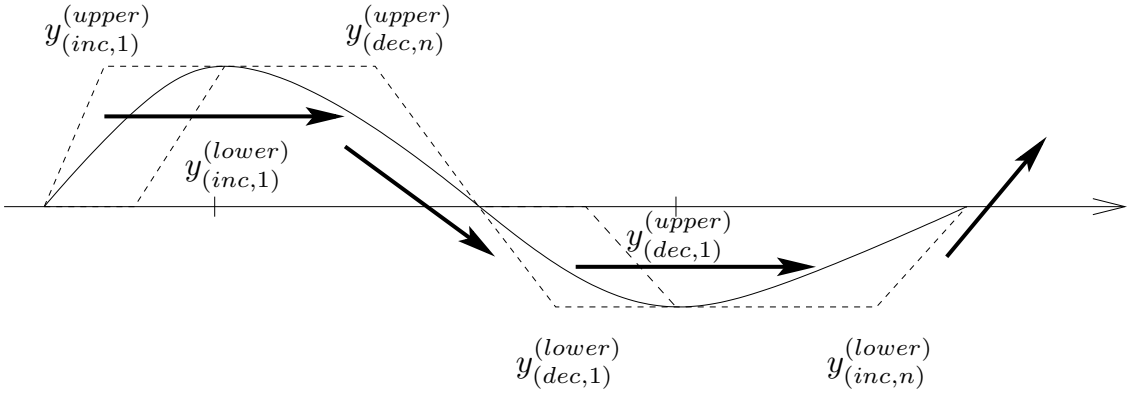
1.6. ábra. Erősebb korlátok M-re különböző α értékek mellett.

így felmerülhet a kérdés, hogy egyáltalán létezhetnek-e számunkra jól használható értékek erre a két paraméterre. Várható, hogy igen, mivel a $\pi/2$-nél egy 4 hosszú periódus jelenik meg. Egy teljes periódusról pedig tudjuk, hogy $(1 + p_m + 1 + p_M)$ hosszú, így a $(p_m + p_M)$-re egy 2-höz közeli értéket kaphatunk. Tehát a $(p_m$ és a $p_M)$ értékek várhatóan 1 körüliek.

Ezen hat korlátfüggvényből levezetünk hasonló, de erősebb, szűkebb halmazt megengedő korlátfüggvényeket. Az újonnan kapott korlátfüggvényeket az eredetiekkel összevetve erősebb korlátfüggvényt kaphatunk. Az $y^{(upper)}_{(inc,1)}$-ből és az $y^{(lower)}_{(inc,1)}$-ből levezetünk egy új helyes $y^{(upper)}_{(dec,n)}$ korlátot, valamint az $y^{(upper)}_{(dec,n)}$-ből egy-egy korlátot az $y^{(lower)}_{(dec,1)}$-re és az $y^{(upper)}_{(dec,1)}$-re. Ezután levezetünk hasonló módszerrel korlátokat az $y^{(lower)}_{(inc,n)}$-re az $y^{(lower)}_{(dec,1)}$-ből



1.7. ábra. A periodikus megoldást korlátozó függvények (szaggatott vonallal jelölve).



1.8. ábra. A korlátok összefüggései.

és az $y^{(upper)}_{(dec,1)}$-ből, valamint az $y^{(upper)}_{(inc,1)}$-re és az $y^{(lower)}_{(inc,1)}$-re az $y^{(lower)}_{(inc,n)}$-ből. Összefoglalva:

$$\left\{ \begin{array}{c} y^{(upper)}_{(inc,1)} \\ \\ y^{(lower)}_{(inc,1)} \end{array} \right\} \Rightarrow y^{(upper)}_{(dec,n)} \Rightarrow \left\{ \begin{array}{c} y^{(lower)}_{(dec,1)} \\ \\ y^{(upper)}_{(dec,1)} \end{array} \right\} \Rightarrow y^{(lower)}_{(inc,n)} \Rightarrow \left\{ \begin{array}{c} y^{(upper)}_{(inc,1)} \\ \\ y^{(lower)}_{(inc,1)} \end{array} \right\}.$$

Az előző korlát javítási sorrendet mutatja az 1.8. ábra. Ezeket az összefüggéseket sorban egymás után alkalmazzuk, és mivel periodikus megoldást vizsgálunk, így az eredményül kapott korlátokat használhatjuk az iterációban indulási korlátokként is. Mutatunk helyes, pontosabb korlátfüggvényeket, majd ezeket alkalmazva a létrejött korlátsorozatról belátjuk, hogy annak bármely eleme érvényes korlát.

Kezdetben vehetjük az $y^{(upper)}_{(dec,1)}$-et és a $y^{(lower)}_{(inc,1)}$-et azonosan 0-nak, míg a további felső korlátokat azonosan M-nek és az alsó korlátokat azonosan $(-m)$-nek. Az $(inc,1)$ részen az $y(t_0+1)$ pontban a függvény értékéről azt tudjuk, hogy az a M értéket veszi fel, így az $y^{(upper)}_{(inc,1)}$ függvényértéknek a (t_0+1) pontban M-nél nagyobbnak vagy egyenlőnek kell lennie $\left(y^{(upper)}_{(inc,1)}(t_0+1) \geq M\right)$. Hasonló gondolatmenettel kapjuk, hogy $y^{(lower)}_{(dec,1)}(t'_0+$

$1) \leq -m$-nek is teljesülnie kell. Abban az esetben, ha ezen egyenlőtlenségek valamelyike nem teljesül a megoldás függvényre, akkor nem létezhet ilyen szélsőértékekkel periodikus pálya.

Látható majd, hogy a (dec, n) szakaszon nincs szükség alsó, míg az (inc, n) szakaszon felső korlátok kiszámítására. Köztes számítások során kapni fogunk ilyen korlátokat, de ezeket nem használjuk ki, így nem térünk ki ezek vizsgálatára.

1.5. A megoldás erősebb korlátjai az 1 hosszú szakaszokon

Kezdetben adjunk meg korrekt $y^{(upper)}_{(inc,1)}$ és $y^{(lower)}_{(dec,1)}$ korlátozó függvényeket a t_0 és t'_0 előtti értékekből. Ez az elv Wright korábban említett módszerén alapul.

1.3. Állítás. *Az alábbi módon definiált korlátfüggvények helyesek:*

$$y^{(upper)}_{(inc,1)}(t) = \min \left\{ \begin{array}{c} y^{(upper)}_{(inc,1)}(t) \\ \\ -\alpha \int\limits_{t_0-1}^{t-1} e^{y^{(lower)}_{(inc,n)}(x)} - 1 \; dx \end{array} \right\}, \text{ ha } t \in [t_0, t_0+1], \tag{1.10}$$

$$\bar{y}^{(lower)}_{(dec,1)}(t) = \max \left\{ \begin{array}{c} y^{(lower)}_{(dec,1)}(t) \\ \\ -\alpha \int\limits_{t'_0-1}^{t-1} e^{y^{(upper)}_{(dec,n)}(x)} - 1 \; dx \end{array} \right\}, \text{ ha } t \in [t'_0, t'_0+1]. \tag{1.11}$$

Bizonyítás. Először megmutatjuk, hogy az $y^{(lower)}_{(inc,n)}(t)$ függvényből hogyan kaphatunk a periodikus pályára egy felső korlátot az $(inc, 1)$ intervallumon. Mivel az $y^{(lower)}_{(inc,n)}$ egy alsó korlátot adó függvény, így teljesül, hogy

$$y^{(lower)}_{(inc,n)}(t) \leq y(t), \text{ minden } t \leq t_0\text{-ra.} \tag{1.12}$$

Integráljuk az y'-t t_0-tól t-ig ($t_0 \leq t \leq t_0 + 1$). Ekkor a Riemann-integrált felhasználva azt kapjuk, hogy:

$$y(t) = y(t) - y(t_0) = -\alpha \int\limits_{t_0}^{t} e^{y(x-1)} - 1 \; dx = -\alpha \int\limits_{t_0-1}^{t-1} e^{y(x)} - 1 \; dx.$$

A $y(x)$ becsléséhez használjuk fel az (1.12) becslést:

$$y(t) = -\alpha \int\limits_{t_0-1}^{t-1} e^{y(x)} - 1 \; dx \leq -\alpha \int\limits_{t_0-1}^{t-1} e^{y^{(lower)}_{(inc,n)}(x)} - 1 \; dx.$$

Az eredeti és az újonnan kapott korlátokat összevetve kapjuk az állításban szereplő újabb, erősebb $y^{(upper)}_{(inc,1)}$ korlátot a $t \geq t_0$ esetben.

Hasonlóan kaphatunk egy jobb korlátot az $y^{(lower)}_{(dec,1)}$-re, mellyel az állítást igazoltuk. ■

Most vizsgáljuk meg, hogy milyen módon kaphatunk egy helyes alsó korlátot az $(inc,1)$-en, illetve egy felső korlátot a $(dec,1)$-en.

1.4. Állítás. *Az alábbi módon definiált korlátfüggvények helyesek:*

$$\bar{y}^{(lower)}_{(inc,1)}(t) = \max \left\{ \begin{array}{c} y^{(lower)}_{(inc,1)}(t) \\ \\ M + \alpha \int\limits_{t-1}^{t_0} e^{y^{(lower)}_{(inc,n)}(x)} - 1 \; dx \end{array} \right\}, \text{ ha } t \in [t_0, t_0+1], \tag{1.13}$$

$$\bar{y}^{(upper)}_{(dec,1)}(t) = \min \left\{ \begin{array}{c} y^{(upper)}_{(dec,1)}(t) \\ \\ -m + \alpha \int\limits_{t-1}^{t'_0} e^{y^{(upper)}_{(dec,n)}(x)} - 1 \; dx \end{array} \right\}, \text{ ha } t \in [t'_0, t'_0+1]. \tag{1.14}$$

Bizonyítás. Integráljuk az y' függvényt t-től (t_0+1)-ig ($t \in [t_0, t_0+1]$):

$$\begin{aligned} y(t_0+1) - y(t) &= -\alpha \int\limits_{t}^{t_0+1} e^{y(x-1)} - 1 \; dx = -\alpha \int\limits_{t-1}^{t_0} e^{y(x)} - 1 \; dx \leq \\ &\leq -\alpha \int\limits_{t-1}^{t_0} e^{y^{(lower)}_{(inc,n)}(x)} - 1 \; dx. \end{aligned}$$

Mivel $y(t_0+1) = M$, így az állításban szereplő jobb korlátot kapjuk az $y^{(lower)}_{(inc,1)}$-re.

Hasonló gondolatmenettel kaphatjuk a jobb felső korlátot az $y^{(upper)}_{(dec,1)}$-re is, mellyel az állítást igazoltuk. ■

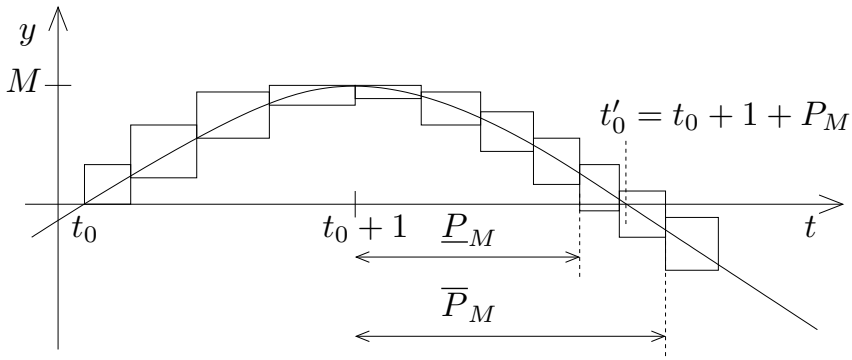
Tehát az $y^{(upper)}_{(dec,n)}$-ből kaphatunk egy helyes erősebb $y^{(upper)}_{(dec,1)}$, illetve $y^{(lower)}_{(dec,1)}$ korlátfüggvényt, míg az $y^{(lower)}_{(inc,n)}$-ből kaphatunk egy erősebb $y^{(upper)}_{(inc,1)}$, illetve $y^{(lower)}_{(inc,1)}$ korlátfüggvényt.

1.6. A periodikus pályák hossza

A vizsgálatainkban fontos szerepe van a (dec,n) és az (inc,n) nem 1 hosszú szakaszok hosszának, azaz a p_m, illetve p_M értékeknek. Ezek nagyságára adunk egy-egy garantált befoglalást. Ezen eljárás eredményeként kapunk a maximumhely és minimumhely utáni szakaszra a megoldásra egy felső, illetve alsó korlátot.

A p_m, illetve p_M értékek befoglalását a trajektória megbízható követésével határozzuk meg. Tudjuk, hogy a Lagrange-féle maradéktaggal ellátott Taylor-polinom az alábbi formában alkalmazható a trajektória követésére ($n=1$ eset, Euler-módszer):

$$Y(x) = Y(x_0) + Y^{(1)}([x_0, x])(x - x_0),$$



1.9. ábra. A trajektória követése a következő zérushely megtalálása céljából.

$$Y([x_0,x]) = Y(x_0) + Y^{(1)}([x_0,x])([0,x-x_0]).$$

Vegyük észre, hogy a trajektória követéséhez szükségünk van 1 hosszan a trajektória befoglalására, valamint a végpontban a függvényérték befoglalására. Ezekre az információkra támaszkodva már megbízhatóan követni tudjuk a trajektóriát.

Mivel az $(inc,1)$ szakaszon ismert egy alsó és egy felső korlátja a periodikus megoldásnak, így az $(inc,1)$ szakaszokra ismert egy garantált befoglalása ezen megoldásnak. A (t_0+1) pontban a függvény értéke M. Hasonlóan a $(dec,1)$-re is megkaphatjuk ezen szükséges információkat. Ezek az adatok elegendőek a trajektória megbízható követéséhez.

Most nézzük meg, hogy hogyan kaphatunk meg egy befoglalást a p_m és a p_M értékekre a trajektória követése során. Ezen követés a (t_0+1), illetve $(t_0'+1)$ pontból indul. A trajektória befoglalása valamelyik lépésben tartalmazni fogja a nullát. Ekkor az eddigi trajektóriakövetés hossza a p_M, illetve a p_m értékek alsó korlátja lesz. Az várható, hogy a trajektóriát tovább követve, a befoglalás egy idő múlva nem fogja tartalmazni a nullát. Az eddig eltelt idő egy felső korlátot ad a p_m, illetve a p_M értékekre. Ezt a gondolatmenetet láthatjuk megjelenítve az 1.9. ábrán.

Elméleti eredményekből tudjuk, hogy ezen értékek nem lehetnek nagyobbak 2-nél, így ha a követés a (t_0+1) ponttól hosszabb lenne mint 2, akkor befejezhetjük a követést, és a megfelelő p érték felső korlátja 2. Ebből az is következik, hogy egy megbízható eljárás az alsó korlátra sem adhat 2-nél nagyobb értéket.

Jelöljük a p_m-re adott befoglalást P_m-mel, illetve p_M-ét P_M-mel, a P_m intervallum alsó és felső végpontjait $\underline{P}_m$-mel és $\overline{P}_m$-mel, valamint a P_M intervallum alsó és felső végpontjait $\underline{P}_M$-mel, illetve $\overline{P}_M$-mel.

1.7. A megoldás erősebb korlátjai a nem 1 hosszú szakaszokon

Tekintsük első megközelítésben a (dec,n) szakaszt. Korábbi ismereteink alapján van egy felső korlátunk az $(inc,1)$ szakaszon. A trajektóriakövetés eredményéből pedig ka-

punk egy felső korlátot a t_0 utáni szakaszra, melyet az 1 hosszú szakaszok becsléseiből és a trajektória követéséből kapunk. Definíció szerint azonban az $y^{(upper)}_{(dec,n)}$ a következő zéruspont (t'_0) előtti részére ad felső korlátot. Mivel nem tudjuk a p_M pontos értékét, így e követést nem tudjuk közvetlenül felhasználni. Ezért azt szükséges megvizsgálni, hogy a t_0 utáni részt hol használhatjuk a t'_0 előtti részen. Továbbá az 1 hosszú szakaszok erősebb korlátainak számítása során csak a $[t'_0 - 1, t'_0]$ intervallumot használjuk, azaz a t'_0 előtt lévő szakaszon a periodikus megoldás korlátjait csak a $[t'_0 - 1, t'_0]$ intervallumon kell ismerni. Ismertetünk egy a Wright ötletén alapuló lehetőséget a távolabbi felső korlátok kihasználására a hasznos intervallumon.

1.5. Állítás. *A P_M számítása során keletkezett trajektória befoglalás $[t_0, t_0 + 1]$ részét használhatjuk a*

$$\left[t'_0 - \left(\overline{P}_M + 1\right), t'_0 - \overline{P}_M\right]$$

intervallumon, míg a $[t_0 + 1, t_0 + 1 + \underline{P}_M]$ részét a

$$[t'_0 - \underline{P}_M, t'_0]$$

intervallumon egy t'_0 előtti korlátfüggvény meghatározásához, ahol t'_0 a következő zéruspont, azaz a $(t_0 + 1 + p_M)$ időpont.

Bizonyítás. Tekintsük a trajektória befoglalását a $[t_0, t_0 + 1 + \overline{P}_M]$ szakaszokon. Ehhez kell a trajektóriakövetés kezdőpontja előtti 1 hosszú szakaszon is a befoglalás. A trajektória követés eredményét jelöljük $Y(t)$-vel, a felső határát pedig $\overline{Y}(t)$-vel.

Egy tetszőleges monoton növő függvény esetében tudjuk, hogy

$$y(t) \geq y(t - \Delta t), \text{ ha } \Delta t \geq 0.$$

Hasonlóan, egy monoton csökkenő függvényre az teljesül, hogy:

$$y(t) \geq y(t - \Delta t), \text{ ha } \Delta t \leq 0.$$

A trajektóriáról tudjuk, hogy az $(inc, 1)$ és az (inc, n) intervallumokon szigorúan monoton növő, míg a $(dec, 1)$ és a (dec, n) intervallumokon szigorúan monoton csökkenő.

Vegyük azt az $(inc, 1)$ szakaszt, amelyen a megoldás szigorúan monoton növő. Itt a trajektóriakövetés során használt kezdeti $y^{(upper)}_{(inc,1)}$-ből kapunk egy felső korlátot, azaz az $\overline{Y}(t)$ egy felső korlát lesz a periodikus pályára a $[t_0, t_0 + 1]$ intervallumon. Mivel $p_M \leq \overline{P}_M$, így a

$$\Delta t = \left(t_0 + 1 + \overline{P}_M\right) - t'_0 \geq 0.$$

Az eddigi információkat összegezve:

$$\overline{Y}(t) \geq y(t) \geq y(t - \Delta t) = y\left(t - \left(\left(t_0 + 1 + \overline{P}_M\right) - t'_0\right)\right).$$

Ebből a relációból látszik, hogy az $\overline{Y}(t)$ egy felső korlátja az $y(t)$-nek a

$$\begin{aligned}\left[t_0 - \left(\left(t_0 + 1 + \overline{P}_M\right) - t'_0\right), t_0 + 1 - \left(\left(t_0 + 1 + \overline{P}_M\right) - t'_0\right)\right] = \\ = \left[t'_0 - \left(\overline{P}_M + 1\right), t'_0 - \overline{P}_M\right]\end{aligned}$$

intervallumon. Mivel a t_0' zéruspont nem lehet nagyobb, mint $t_0+1+\overline{P}_M$, ezért a $[t_0, t_0+1]$ intervallumot felhasználhatjuk egy jobb $y^{(upper)}_{(dec,n)}$ korláthoz a

$$\left[t_0' - \left(\overline{P}_M + 1\right), t_0' - \overline{P}_M\right]$$

intervallumon.

Vegyük a trajektóriakövetés $(t_0 + 1)$ kezdőpont utáni részét. Ebből kapunk egy felső korlátot az $y(t)$-re a $[t_0 + 1, t_0 + 1 + \underline{P}_M]$ időintervallumon. Mivel itt szigorúan monoton csökkenő a függvény, és $\underline{P}_M \leq p_M$, ezért

$$\overline{Y}(t) \geq y(t) \geq y(t - \Delta t) = y\left(t - \left((t_0 + 1 + \underline{P}_M) - t_0'\right)\right)$$

teljesül.

Ebből szintén látható, hogy felső korlát lesz a

$$\begin{aligned}\left[t_0 + 1 - \left((t_0 + 1 + \underline{P}_M) - t_0'\right), t_0 + 1 + \underline{P}_M - \left((t_0 + 1 + \underline{P}_M) - t_0'\right)\right] = \\ = \left[t_0' - \underline{P}_M, t_0'\right]\end{aligned}$$

intervallumon is. ■

Ezen eredményt a következő módon is magyarázhatjuk. Az első esetben a periodikus megoldást eltoljuk úgy, hogy az eredeti t_0' zéruspont a $(t_0 + 1 + \overline{P}_M)$ pontba kerüljön. Ekkor a monoton növő $(inc, 1)$ szakaszon az eltolt függvény az eredeti értéknél nem vesz fel nagyobb értéket, azaz az eredeti periodikus megoldás felső korlátja felső korlátja marad az eltolt megoldásnak is. Ezt az ötletet illusztráljuk az 1.10(a) ábrán.

A második esetben a periodikus megoldást úgy toljuk el, hogy az eredeti t_0' zéruspont a $(t_0 + 1 + \underline{P}_M)$ pontba kerüljön. Így a monoton csökkenő (dec, n) szakaszon az eltolt függvény az eredeti értéknél nem vesz fel nagyobb értéket. Ezért az eredeti periodikus megoldás felső korlátja felső korlátja marad a periodikus megoldásnak a t_0' pontot megelőzően is. Ezt láthatjuk az 1.10(b) ábrán.

Az előző két ötletből és az $y^{(upper)}_{(dec,n)}$-ből kaphatunk egy felső korlátot a zéruspont előtti részre. Hasonlóan kaphatunk egy új alsó korlátot a t_0'' előtti részre is.

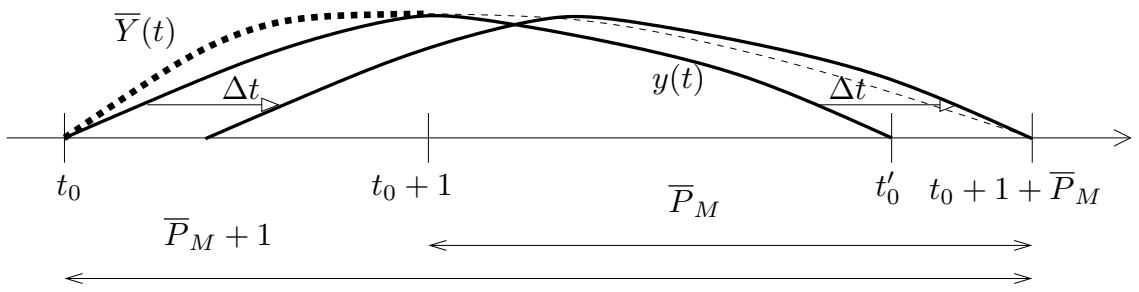
Most nézzük meg, hogyan kaphatunk egy erősebb korlátot a (dec, n) és az (inc, n) szakaszok $[t_0' - 1, t_0']$ intervallumára az előző korlátok $(t_0' - 1)$ előtti részeiből.

1.6. Állítás. *Az alábbi módon definiált korlátfüggvények helyesek:*

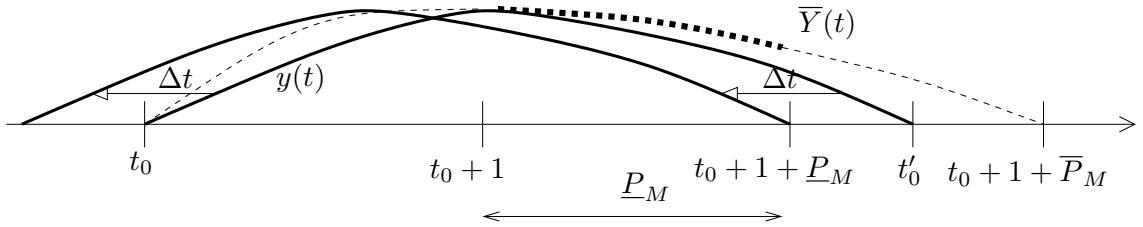
$$\bar{y}^{(upper)}_{(dec,n)}(t) = \min\left\{\begin{array}{c} y^{(upper)}_{(dec,n)}(t) \\ \\ \alpha \int\limits_{t-1}^{t_0'-1} e^{y^{(upper)}_{(dec,n)}(x)} - 1 \; dx \end{array}\right\}, \text{ ha } t \in [t_0' - 1, t_0'], \tag{1.15}$$

és

$$\bar{y}^{(lower)}_{(inc,n)}(t) = \max\left\{\begin{array}{c} y^{(lower)}_{(inc,n)}(t) \\ \\ \alpha \int\limits_{t-1}^{t_0''-1} e^{y^{(lower)}_{(inc,n)}(x)} - 1 \; dx \end{array}\right\}, \text{ ha } t \in [t_0'' - 1, t_0'']. \tag{1.16}$$



(a) A $\overline{P}_M$ eset.



(b) A $\underline{P}_M$ eset.

1.10. ábra. A trajektória követésből kapott felső korlátok.

Bizonyítás. Először tekintsük a (dec, n) esetet. Ekkor tudjuk, hogy

$$y^{(upper)}_{(dec,n)} \geq y(t).$$

Integráljuk az y'-t t-től t_0'-ig ($t_0' - 1 \leq t \leq t_0'$):

$$-y(t) = y(t_0') - y(t) = -\alpha \int_t^{t_0'} e^{y(x-1)} - 1 \, dx = -\alpha \int_{t-1}^{t_0'-1} e^{y(x)} - 1 \, dx.$$

Ebből

$$y(t) \leq \alpha \int_{t-1}^{t_0'-1} e^{y^{(upper)}_{(dec,n)}(x)} - 1 \, dx.$$

Ezzel megkaphatjuk az állításban szereplő felső korlátot.

Hasonlóan az $y^{(lower)}_{(inc,n)}$ esetén:

$$y(t) \geq \alpha \int_{t-1}^{t_0''-1} e^{y^{(lower)}_{(inc,n)}} - 1 \, dx,$$

melyből következik az állításban szereplő második állítás. ■

Összefoglalva, az $y^{(upper)}_{(inc,1)}$, illetve az $y^{(lower)}_{(inc,1)}$ korlátfüggvényekből kaphatunk egy garantált trajektóriakövetést, melyből meghatározhatunk egy helyes felső korlátfüggvényt.

E korlátfüggvény távolabbi értékeit felhasználhatjuk a következő zéruspont előtti 1 hosszú szakaszon a periodikus megoldás erősebb korlátainak meghatározásához. Ezen 1 hosszú szakasz korlátja szükséges az $y^{(upper)}_{(dec,1)}$, illetve az $y^{(lower)}_{(dec,1)}$ pontosabb korlátfüggvények kiszámításához. Hasonló elv működik az $y^{(lower)}_{(inc,n)}$ esetén is.

1.8. A periodikus korlátokat számító eljárás

Az $y^{(lower)}_{(inc,1)}$, $y^{(upper)}_{(dec,1)}$, $y^{(upper)}_{(inc,1)}$ és az $y^{(lower)}_{(dec,1)}$ korlátfüggvények csak 1 hosszan értelmezettek a $[t_0, t_0+1]$, illetve a $[t'_0, t'_0+1]$ időintervallumokon. Az $y^{(lower)}_{(inc,n)}$ és az $y^{(upper)}_{(dec,n)}$ függvények számítása során azonban (t'_0-1)-nél, illetve a (t''_0-1)-nél kisebb értékekre is szükségünk van, így ezeket hosszabban tároljuk. E rész túl hosszú tárolása szintén nem hasznos, így e részt 2 hosszan fogjuk tárolni, amely a korábbiak miatt nem okozhat gondot.

Mind a hat korlátfüggvény esetében az egységnyi hosszú szakaszt egyenlő részekre osztjuk fel. A számítógép pontosságát kihasználva minden esetben egy 2^n alakú számot választunk ezen részek számára, ahol $n \in \mathbb{N}$. Jelöljük az így kapott időszeleteket t_i-vel, ahol $i \in (1, \ldots, 2^n)$, illetve (dec, n) és (inc, n) esetén $i \in (1, \ldots, 2^{n+1})$ a zérusponttól vett távolságuk szerint sorszámozva. Azaz a (dec, n), illetve az (inc, n) részekben visszafelé, míg az $(inc, 1)$ és a $(dec, 1)$ részekben előrefelé számozzuk meg az időszeleteket. Egy ilyen szakasz hossza 2^{-n}, mely számítógépen pontosan ábrázolható, nem túl nagy n-ekre. Ezeken az időintervallumokon minden esetben egy konstans értéket vesznek fel a korlátfüggvényeink. Ilyen konstrukcióval egy lépcsős függvényt kapunk, mely minden pontban megfelelően becsli a periodikus függvényünket. Ezen lépcsős függvényt jelöljük Y-nal.

A lépcsős függvényen az integrálok numerikus számítására már alkalmazhatjuk a téglalap módszert. Az eljárás korrektségének megtartása érdekében egy t_j időszakaszban a függvényérték meghatározásához a $(t_j - 1)$ időszakaszra eső téglalap területét is be kell számítanunk a téglalap módszerbe.

Így az $Y^{(lower)}_{(inc,n)}$ függvényből az alábbi módon lehet kiszámolni az $\bar{Y}^{(upper)}_{(inc,1)}(t_i)$ $(i = 1, \ldots, 2^n)$ értéket az (1.10) képlet alapján:

$$\bar{Y}^{(upper)}_{(inc,1)}(t_i) = \min\left\{-\alpha \sum_{j=1}^{i}\left(e^{Y^{(lower)}_{(inc,n)}(t_{2^n-j+1})} - 1\right)/2^n;\ Y^{(upper)}_{(inc,1)}(t_i)\right\}. \qquad (1.17)$$

Az (1.11), az (1.13) és az (1.14) esetén is hasonlóan kapjuk, hogy

$$\bar{Y}^{(lower)}_{(dec,1)}(t_i) = \max\left\{-\alpha \sum_{j=1}^{i}\left(e^{Y^{(upper)}_{(dec,n)}(t_{2^n-j+1})} - 1\right)/2^n\ ;\ Y^{(lower)}_{(dec,1)}(t_i)\right\}, \qquad (1.18)$$

$$\bar{Y}^{(lower)}_{(inc,1)}(t_i) = \max\left\{M + \alpha \sum_{j=i}^{2^n}\left(e^{Y^{(lower)}_{(inc,n)}(t_{2^n-j+1})} - 1\right)/2^n;\ Y^{(lower)}_{(inc,1)}(t_i)\right\}, \qquad (1.19)$$

1. Algoritmus. A periódushossz számítása.

Input:
- s: M vagy $(-m)$, mely a periodikus pálya szélsőértéke,
- α: a vizsgált differenciálegyenlet paramétere,
- 2^n: az egységnyi szakasz felosztásának darabszáma,
- L, U: az 1 hosszú szakasz alsó és felső korlátja.

Output:
- A nem 1 hosszú szakasz hosszának befoglalása,
- a trajektória befoglalása a kezdő zérusponttól.

1. lépés: Foglaljuk be az $Y(t_i)$-be ($i = 1, \ldots, 2^n$) az 1 hosszú szakaszon a periodikus megoldást az U és az L függvények segítségével.
2. lépés: Legyen $j = 2^n + 1$ és legyen $Y_{last} = s$.
3. lépés: Foglaljuk be az $Y(t_j)$-t a $\left(Y_{last} + \left(-\alpha\left(e^{Y(t_{j-n})} - 1\right)\right) \cdot [0, 1/2^n]\right)$ képlettel.
4. lépés: Legyen $Y_{last} = Y_{last} + \left(-\alpha\left(e^{Y(t_{j-n})} - 1\right)\right)/2^n$.
5. lépés: Ha az $Y(t_{j-1})$ befoglalás nem tartalmazza a 0-át és az $Y(t_j)$ igen, akkor a nem 1 hosszú szakasz hosszára vonatkozó alsó korlát a $\frac{j}{2^n} - 1$.
6. lépés: Ha az $Y(t_{j-1})$ befoglalás tartalmazza a 0-át és az $Y(t_j)$ nem, akkor a nem 1 hosszú szakasz hosszára vonatkozó felső korlát a $\frac{j}{2^n} - 1$, és STOP.
7. lépés: Legyen $j = j + 1$.
8. lépés: Ha $j < 3 \cdot 2^n$, akkor folytassuk a 3. lépéssel, egyébként a felső korlát legyen $\frac{j}{2^n} - 1$ és STOP.

és

$$\bar{Y}^{(upper)}_{(dec,1)}(t_i) = \min\left\{-m + \alpha \sum_{j=i}^{2^n} \left(e^{Y^{(upper)}_{(dec,n)}(t_{2^n-j+1})} - 1\right)/2^n;\ Y^{(upper)}_{(dec,1)}(t_i)\right\}. \quad (1.20)$$

A fenti formulákkal megadunk egy befoglalást az $(inc, 1)$, illetve a $(dec, 1)$ szakaszokon a periodikus megoldásra. Tudjuk, hogy ezen szakaszok végeinél a periodikus megoldás felveszi a szélsőértékeket. Ezután egy egyszerű eljárással követhető a trajektória (lásd az 1. Algoritmust), mely során azt kell ellenőrizni a p_m és a p_M értékek befoglalásának megadásához, hogy a trajektória jelenlegi és előző befoglalása tartalmazza-e a 0 értéket. Abban az esetben, ha az egy lépéssel korábbi befoglalás még nem tartalmazta a 0-át, de a jelenlegi már tartalmazza, akkor a jelenlegi időintervallumon a periodikus megoldás elérhette már a zéruspontot. Hasonlóan, ha az előző befoglalás tartalmazta a 0-át, a jelenlegi pedig nem, akkor a trajektória biztos, hogy már áthaladt a zérusponton. E két tulajdonság együtt egy garantált befoglalást ad a maximumpont és a zéruspont közötti távolságra. Továbbá az eljárásnak eredménye (a távolságon felül) egy-egy befoglalása a trajektóriának a $\left[t_0, t_0 + 1 + \overline{P}_M\right]$, illetve a $\left[t'_0, t'_0 + 1 + \overline{P}_m\right]$ intervallumon.

Vizsgáljuk meg, hogy hogyan alkalmazhatjuk a trajektória követésénél kapott befoglalás felső értékeit az $\tilde{Y}^{(upper)}_{(dec,n)}$ függvény javítására, mely utóbbi egy 3 hosszú közbülső számításokhoz használt korlátfüggvény. Először tekintsük az $(inc, 1)$ szakasz felső korlátját, mely az $Y(t_i)$ trajektória követés felső korlátja az $i \in (1, \ldots, 2^n)$ részen. Az 1.5. Állítás alapján ezt a részt kell felhasználni az $Y^{(upper)}_{(dec,n)}$ függvényben a t_i időintervallu-

mokon, ahol $i \in \left(\left\lceil 2^n \overline{P}_M \right\rceil + 2^n, \ldots, \left\lceil 2^n \overline{P}_M \right\rceil + 1\right)$. Az alábbi képlet adja a beillesztés módját:

$$\tilde{Y}^{(upper)}_{(dec,n)}(t_i) = \min\left\{\overline{Y}(t_{\left\lceil 2^n \overline{P}_M \right\rceil + 2^n + 1 - i});\ Y^{(upper)}_{(dec,n)}(t_i)\right\}, \quad \text{ahol } i \in \left(\left\lceil 2^n \overline{P}_M \right\rceil + 1, \left\lceil 2^n \overline{P}_M \right\rceil + 2^n\right). \tag{1.21}$$

Most nézzük meg, hogy hogyan használható a trajektória követésből a maximum pont utáni szakaszra kapott eredmény. Az $Y(t_i)$ függvény tartalmazza ezen használható korlátokat az $i \in (2^n + 1, 2^n + \lfloor 2^n \underline{P}_M \rfloor)$ szakaszon. Ezeket kell felhasználni az $\tilde{Y}^{(upper)}_{(dec,n)}$ függvényben a t_i időintervallumokon, ahol $i \in (\lfloor 2^n \underline{P}_M \rfloor, \ldots, 1)$. Az 1.5. Állítás alapján az alábbi képlet adja a beillesztés módját:

$$\tilde{Y}^{(upper)}_{(dec,n)}(t_i) = \min\left\{\overline{Y}(t_{\lfloor \underline{P}_M * 2^n \rfloor + 2^n + 1 - i});\ Y^{(upper)}_{(dec,n)}(t_i)\right\}, \quad \text{ahol } i \in (1, \ldots, \lfloor 2^n \underline{P}_M \rfloor). \tag{1.22}$$

Ha $\overline{P}_M < 1$, akkor a 3 hosszú intervallumon tárolt $\tilde{Y}^{(upper)}_{(dec,n)}$ függvény t'_0-tól $\overline{P}_M + 1$-nél távolabb eső pontjaiba 0-át is írhatunk felső korlátnak, azaz

$$\tilde{Y}^{(upper)}_{(dec,n)}(t_i) = 0, \text{ ahol } \left\lceil 2^n \overline{P}_M \right\rceil + 2^n + 1 \leq i \leq 3 \cdot 2^n. \tag{1.23}$$

Az előző gondolatmenetek alapján az $Y^{(lower)}_{(inc,n)}$ függvényre a következőt kapjuk:

$$\tilde{Y}^{(lower)}_{(inc,n)}(t_i) = \min\left\{\underline{Y}(t_{\left\lceil 2^n \overline{P}_m \right\rceil + 2^n + 1 - i});\ Y^{(lower)}_{(inc,n)}(t_i)\right\}, \quad \text{ahol } i \in \left(\left\lceil 2^n \overline{P}_m \right\rceil + 1, \left\lceil 2^n \overline{P}_m \right\rceil + 2^n\right), \tag{1.24}$$

$$\tilde{Y}^{(lower)}_{(inc,n)}(t_i) = \min\left\{\underline{Y}(t_{\left\lceil 2^n \overline{P}_m \right\rceil + 2^n + 1 - i});\ Y^{(lower)}_{(inc,n)}(t_i)\right\}, \quad \text{ahol } i \in \left(\left\lceil 2^n \overline{P}_m \right\rceil + 1, \left\lceil 2^n \overline{P}_m \right\rceil + 2^n\right), \tag{1.25}$$

$$\tilde{Y}^{(lower)}_{(inc,n)}(t_i) = \underline{Y}(t_{\lfloor \underline{P}_m * 2^n \rfloor + 2^n + 1 - i}), \text{ ahol } i \in (1, \ldots, \lfloor 2^n \underline{P}_m \rfloor). \tag{1.26}$$

Miután meghatároztuk az erősebb $\tilde{Y}^{(upper)}_{(dec,n)}$ és $\tilde{Y}^{(lower)}_{(inc,n)}$ függvényeket, nézzük meg, hogyan számíthatók a „távoli" korlátokból pontosabb $\bar{Y}^{(upper)}_{(dec,n)}(t_i)$ és $\bar{Y}^{(lower)}_{(inc,n)}(t_i)$ korlátozófüggvények, ahol $i \in (1, \ldots, 2^n)$. Ezeket a függvényeket az előzőekhez hasonlóan megkaphatjuk az (1.15) és az (1.16) képletekből:

$$\bar{Y}^{(upper)}_{(dec,n)}(t_i) = \min\left\{\alpha \sum_{j=i}^{2^n} \left(e^{\tilde{Y}^{(upper)}_{(dec,n)}(t_{j-2^n})} - 1\right)/2^n;\ \tilde{Y}^{(upper)}_{(dec,n)}(t_i)\right\}, \tag{1.27}$$

illetve

$$\bar{Y}^{(lower)}_{(inc,n)}(t_i) = \max\left\{\alpha\sum_{j=i}^{2^n}\left(e^{\tilde{Y}^{(lower)}_{(inc,n)}(t_{j-2^n})}-1\right)/2^n;\ \tilde{Y}^{(lower)}_{(inc,n)}(t_i)\right\}. \tag{1.28}$$

Ezzel egy számítási módszert adtunk a korlátsorozatok előállítására. Tudjuk, hogy a periodikus megoldás értéke az $y(t_0+1)$-ben az $(inc,1)$ szakaszon M, illetve a $(dec,1)$ szakaszon $(-m)$. Így a korlátfüggvényre az alábbiaknak kell teljesülnie:

$$Y^{(upper)}_{(inc,1)}(t_{2^n}) \geq M,$$

illetve

$$Y^{(lower)}_{(dec,1)}(t_{2^n}) \leq -m.$$

A következőkben ismertetjük a 2. Algoritmust, mely egy adott α-ra, M-re, illetve m-re eldönti, hogy létezhet-e periodikus megoldás M, illetve $(-m)$ szélsőértékekkel. Az eljárás során használhatjuk az $\alpha \leq 1.5$ esetben levezetett összefüggéseket is.

Ezen eljárás működik akkor is, hogy ha α, M és m intervallumok. Ebben az esetben a korlátozó függvények az intervallumok minden értékére korlátozzák a periodikus pályákat.

1.9. A sejtés bizonyítása további α értékekre

Az egyszerűbb számítások végett az (1.4) és az (1.5) feltételeken fogjuk bemutatni a teljes bizonyítás vázlatát. Az erősebb (1.7), (1.8) és (1.9) feltételekkel is hasonlóan számíthatók a következőkben leírt állítások.

A korábbi számítógépes eljárások egyik problémája az, hogy csak véges nagy intervallumokra működnek. Lássuk be az alábbi állítást, melyben felülről korlátozzuk a lehetséges M-eket és m-eket.

1.7. Állítás. *Egy periodikus megoldás esetén a lehetséges M értékek nem lehetnek nagyobbak $\pi/2$-nél, míg az m értékek 6-nál.*

Bizonyítás. Tekintsük az (1.4) feltételt. Mivel egy hatványfüggvény sosem lehet negatív, így igaz a következő összefüggés:

$$M \leq -\alpha\left(e^{-m}-1\right) \leq -\alpha(0-1) = \alpha.$$

Mivel minket csak az $\alpha \leq \frac{\pi}{2}$ eset érdekel, így bizonyítottuk az állítást az M-re. Most tekintsük az (1.5) feltételt. Mivel tudjuk, hogy $M \leq \frac{\pi}{2}$ és $\alpha \leq \frac{\pi}{2}$, így

$$m \leq \alpha\left(e^{M}-1\right) \leq \alpha(e^{\frac{\pi}{2}}-1) \leq 6,$$

amivel bizonyítottuk az állítást. ■

Krisztin Tibor eredményét az alábbi tételben foglalhatjuk össze:

2. Algoritmus. A periodikus pálya létezésének ellenőrzése.

Input:
- M és $(-m)$: a periodikus pálya szélsőértékei,
- α: a vizsgált differenciálegyenlet paramétere,
- 2^n: az egységnyi szakasz felosztásának darabszáma,
- $cikl_{max}$: a korlátozó sorozat maximális hossza.

Output:
- A megállapítás, miszerint létezhet-e periodikus pálya a megadott szélsőértékekkel.

0. lépés: Ellenőrizzük adott m-re, illetve M-re az (1.7), (1.8) és (1.9) feltételeket. Ha bármelyik feltétel hamis, akkor nem létezik periodikus megoldás, és STOP.

1. lépés: Legyen $c = 0$ és minden $i = 1, \ldots, 2^n$-re:

$$\text{az } Y^{(upper)}_{(inc,1)}(t_i) = M, \text{ az } Y^{(lower)}_{(dec,1)}(t_i) = -m,$$

$$\text{az } Y^{(lower)}_{(inc,1)}(t_i) = 0 \text{ és az } Y^{(upper)}_{(dec,1)}(t_i) = 0,$$

valamint minden $i = 1, \ldots, 2^{n+1}$-re:

$$\text{az } Y^{(upper)}_{(dec,n)}(t_i) = M, \text{ és } Y^{(lower)}_{(dec,n)}(t_i) = -m.$$

2. lépés: Számoljunk ki egy-egy erősebb korlátfüggvényt az $Y^{(upper)}_{(inc,1)}$-re és az $Y^{(lower)}_{(inc,1)}$-re az $Y^{(lower)}_{(inc,n)}$-ből az (1.17) és az (1.19) képletek felhasználásával.

3. lépés: Ha az $y^{(upper)}_{(inc,1)}(t_{2^n}) < M$, akkor nem létezik periodikus pálya és STOP.

4. lépés: Számoljunk ki a trajektória követő eljárással egy-egy befoglalást a trajektóriára.

5. lépés: Készítsünk a trajektóriakövetésből kapott eredményből egy erősebb felső korlátot az $Y^{(upper)}_{(dec,n)}$-re az (1.21-1.23) és az (1.27) képletekkel.

6. lépés: Javítsuk az $Y^{(lower)}_{(dec,1)}$ és az $Y^{(upper)}_{(dec,1)}$ korlát függvényeket az $Y^{(upper)}_{(dec,n)}$-ből az (1.18) és az (1.20) képletek felhasználásával.

7. lépés: Ha az $y^{(lower)}_{(dec,1)}(t_{2^n}) > -m$, akkor nem létezik periodikus pálya és STOP.

8. lépés: Készítsünk a trajektóriakövetésből kapott eredményből egy erősebb alsó korlátot az $Y^{(lower)}_{(inc,n)}$-re az (1.24-1.26) és az (1.28) képletekkel.

9. lépés: Ha $c \geq cikl_{max}$, akkor létezhet periodikus megoldás és STOP.

10. lépés: Legyen $c = c + 1$, és folytassuk a 2. lépéssel.

1.8. Tétel. *Léteznek olyan $K_M(\alpha)$ és $K_m(\alpha)$, függvények, melyekhez nem létezik periodikus pálya egyik $M \in [0, K_M(\alpha)]$ és $m \in [0, K_m(\alpha)]$ szélsőértékekkel sem.*

Ezen eredmény elegendő lenne a bizonyításhoz, ha minden α-ra $K_M(\alpha)$ és $K_m(\alpha)$ elegendő nagyok lennének, de sajnos ezen függvényekre

$$\lim_{\alpha \to \frac{\pi}{2}} K_M(\alpha) = 0 \text{ és } \lim_{\alpha \to \frac{\pi}{2}} K_m(\alpha) = 0.$$

A bizonyítás csak akkor lehet sikeres, ha az ellenőrizendő M és m intervallumok egyike sem tartalmazza a nullát. Így lássuk be a következő állítást, mely következik az eddigi elméleti eredményekből.

1.9. Állítás. *Léteznek olyan $K'_M(\alpha)$ és $K'_m(\alpha)$ függvények, melyek esetében nem létezik periodikus pálya egyetlen $M \notin [K'_M(\alpha), \pi/2]$ és $m \notin [K'_m(\alpha), 6]$ szélsőértékekkel sem ($m, M \geq 0$).*

Bizonyítás. Legyen

$$K'_M(\alpha) = \min \left\{ K_M(\alpha), \log \left(\frac{K_m(\alpha)}{\pi/2} + 1 \right) \right\}$$

és

$$K'_m(\alpha) = \min \left\{ K_m(\alpha), -\log \left(\frac{K_M(\alpha)}{-\pi/2} + 1 \right) \right\}.$$

Belátjuk, hogy a fent definiált konstansokra az eddigi elméleti eredményekből következik az állítás. Ha $M > \pi/2$, vagy $m > 6$, akkor az 1.7. Állításból következik, hogy nincs periodikus megoldás ilyen szélsőértékekkel. Ha $M < K'_M(\alpha)$ és $m < K'_m(\alpha)$, akkor a minimumképzés miatt $M < K_M(\alpha)$ és $m < K_m(\alpha)$, és ebben az esetben az 1.8. Állításból tudjuk, hogy nincs periodikus megoldás.

Már csak az alábbi intervallumokra kell belátni, hogy nem tartalmaznak lehetséges szélsőértékeket:

- $M \in (K'_M(\alpha), \pi/2)$ és $m \in (0, K'_m(\alpha))$,
- $M \in (0, K'_M(\alpha))$ és $m \in (K'_m(\alpha), 6)$.

A továbbiakban belátjuk az első állítást. Vegyük az (1.4) korlátot:

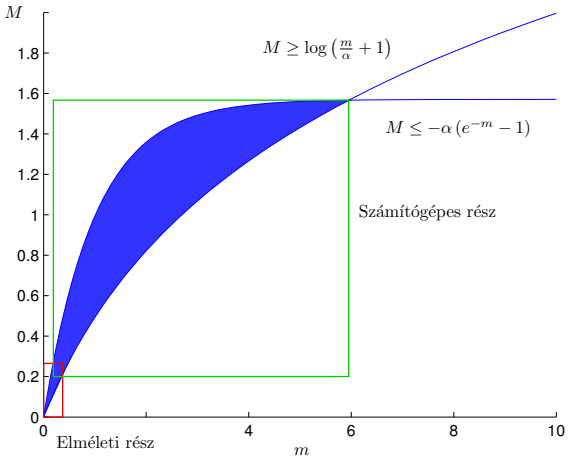
$$M \leq -\alpha \left(e^{-m} - 1 \right),$$

majd rendezzük át az egyenlőtlenséget és nézzük meg, hogy m-re milyen korlátot kapunk:

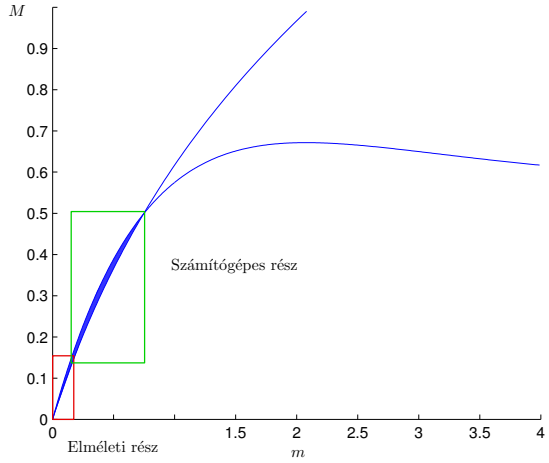
$$m \geq -\log \left(\frac{M}{-\alpha} + 1 \right).$$

A vizsgált rész miatt $M \geq K'_M(\alpha)$, továbbá a minimumképzés miatt $M \geq K_M(\alpha)$, tehát az előbbi feltétel alapján:

$$m \geq -\log \left(\frac{K_M(\alpha)}{-\alpha} + 1 \right),$$



(a) Az $\alpha \leq 1.0$ esetben használt feltételek esetében.



(b) Az $\alpha \leq 1.5$ esetben használt feltételek esetében.

1.11. ábra. Elméleti eredmények és a számítógépes rész kapcsolata (a részek nagysága csak illusztráció).

aminek eleget tevő pont nincs a vizsgált m tartományban. A második állítás igazolását hasonlóan végezhetjük az (1.5) korlátból. ■

Az előzőekből következően elegendő az alábbi intervallumot vizsgálni számítógéppel (lásd az 1.11(a). ábrát):

$$M \in [K'_M(\alpha), \pi/2] \text{ és } m \in [K'_m(\alpha), 6].$$

Ahogy a szakasz elején is említettük, ezeket a számításokat a gyengébb feltételekkel kaptuk. Azonban az erősebb (1.7), (1.8) és (1.9) feltételekkel is számolhatunk, mely esetben sokkal kisebb számítógéppel ellenőrizendő intervallumot kapunk (lásd az 1.11(b). ábrát). De sajnos ha $\alpha \to \pi/2$, akkor ezen számítógéppel ellenőrizendő tartomány nő. Továbbá a számítógépes eljárás számítási igénye egyre nagyobb, ha a tartományok alsó határa a nullához közelít.

A korábbi számítási kapacitásainkkal a számítógépes eljárásunk a $1.5 \leq \alpha \leq 1.5706$ értékekre bizonyította az alábbi állítást:

1.10. Állítás. *Az $\alpha \in [1.5, 1.5706]$ értékeire nem létezik periódikus megoldás*

$$M \in [K'_M(\alpha), \pi/2] \text{ és } m \in [K'_m(\alpha), 6]$$

szélsőértékekkel.

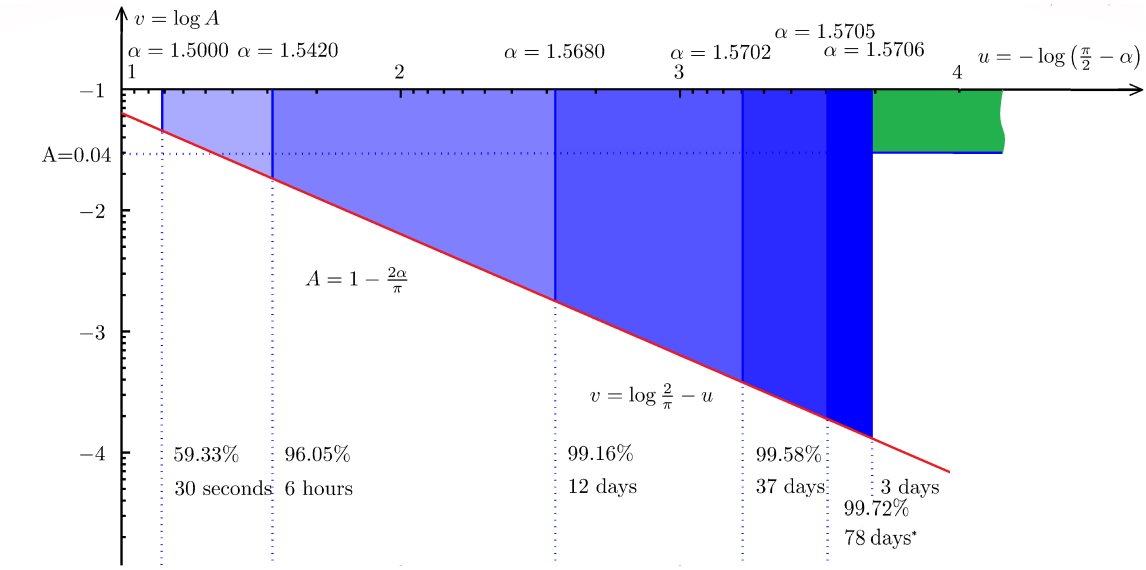
Ezzel sikerül a Wright-sejtést az $\alpha \in [1.5, 1.5706]$ alábbi intervallumra kiterjeszteni [14]:

1.11. Tétel. *Az $\alpha \in [1.5, 1.5706]$ paramétertartományban a*

$$y'(t) = -\alpha \left(e^{y(t-1)} - 1\right)$$

Wright-féle késleltetett differenciálegyenletnek csak a konstans nulla globálisan vonzó megoldása létezik, és minden megoldása tart a 0-ba, ha $t \to \infty$.

Ezen eredményekről az 1.12. ábra ad egy összefoglalót. Később a $[1.5706, \pi/2]$ tartományon is próbáltunk elérni numerikus eredményeket, melyek az 1.1. táblázatban láthatók [15]. Ezen eredmények számolása egy 12 magos, 24 szálat alkalmazó szerveren történt. Ezekből a numerikus eredményekből az látható, hogy bár elvi szinten határértékben a sejtés a teljes tartományon bizonyítható lehetne, a számítási igények drasztikus növekedése miatt a $\pi/2 = 1.5707963267\ldots$ értéke kivárható időben a módszerünkkel elérhetetlen.



1.12. ábra. Az elért eredmények összefoglalása.

A tisztán számítógépes eredmények segítségével az is bizonyítható, hogy nincs periodikus megoldás egyetlen $\alpha \leq \pi/2$ értéknél sem, ha

$$m \geq 0.04 \text{ és } M \geq 0.04.$$

Intervallum	CPU idő (óra)
$[1.57060, 1.57061]$	56.9
$[1.57061, 1.57062]$	64.9
$[1.57062, 1.57063]$	83.7
$[1.57063, 1.57064]$	119.4
$[1.57064, 1.57065]$	141.2

1.1. táblázat. A szükséges CPU idő különböző α intervallumok bizonyításához.

1.12. Állítás. *Az $\alpha \in [1.5706, \pi/2]$ értékeire nem létezik periódikus megoldás*

$$M \in [0.04, \pi/2] \text{ és } m \in [0.04, 6]$$

szélsőértékekkel.

Természetesen ez a 0.04-es érték csökkenthető a számítási igény növekedésével és elvi szinten a 0 is tetszőlegesen megközelíthető lehet. Azaz bizonyításunk egyetlen nyitott része a $\pi/2$ közeli és 0 körüli szélsőértékekkel rendelkező periodikus megoldások nem létezésének a bizonyítása.

Megjegyzendő, hogy pont ezt a részt és csak ezt a részt tudta Jan Bouwe és Jonathan Jaquette [85] tisztán matematikai eszközökkel bizonyítani. Ezt a bizonyítást ki tudták terjeszteni a 0.04-es m, M értékig, így az általunk számítógéppel igazolt esetek, és az általuk megalkotott matematikai bizonyítás kellett a Wright-sejtés teljes bizonyításához. Továbbá a számítógépes részben megtalálható módszert felhasználták a Jones-sejtés bizonyítása során is [49].

1.10. Összefoglalás

A Wright-sejtéssel kapcsolatban a megjelenését követő 50 évben több eredmény is született, de ezek egyike sem tudott a sejtéssel kapcsolatban teljes értékű érdemi előrehaladást felmutatni. Azonban ezen eredmények közül több is kellett a fejezetben tárgyalt eredmények eléréséhez, illetve a probléma mélyebb megismeréséhez. Ilyenek például, a Krisztin Tibor által bizonyított 0 közeli szélsőértékes eredmények, mivel ezt a részt a számítógépes algoritmusunk nem képes bizonyítani. Továbbá a bizonyítás számítógépes része is használ a Wright eredeti bizonyításában szereplő elméleti eredményeket is a hatékonyság érdekében.

A jelenlegi fejezet fő eredménye egy megbízható numerikus algoritmus kidolgozása a Wright-sejtés véges dimenziós változatának a megoldására. Az algoritmus két kulcsfontosságú, általam kifejlesztett, részét emelem ki. Az egyik a korábban publikált trajektória követő eljárásom [6], mely szükséges volt, hogy újabb korlátokat tudjunk mondani a trajektóriára vonatkozóan. Ebben még nem használtam ki a zérus pontokat, és az ezeken alapuló ciklikusságot. A másik eredmény az az új ötletem volt, hogy a zéruspont környékén lévő korlátokból induló trajektória követés eredményét felhasználtam más zéruspontok környékén lévő trajektória korlátozására. Ez az ötletem szükséges volt az érdemi

előrelépéshez a sejtés bizonyítása során. Itt még egyszer megjegyezném, hogy a teljes kidolgozásban és a hatékony megvalósításban az elméleti matematikai eredmények is szükségesek voltak. Ezen elméleti matematikai részeket ebben a fejezetben nem mutattam be részletesen.

Ezzel a javasolt megbízható eljárással sikerült a sejtést bizonyítani az $\alpha \leq 1.5706$ [14], párhuzamosítással az $\alpha \leq 1.57065$ értékekre [15]. Az eljárásunknak ez nem az elméleti határa, de a szükséges CPU idő nagymértékű növekedése miatt ez a módszer nem alkalmazható hatékonyan a bizonyítás teljes egészére. Pontosabban, a cikkünkben publikált matematikai eredményekhez szükséges informatikai eljárás emberi léptékben mérhető időn belül nem képes bizonyítani a teljes sejtést. A kapcsolódó cikkben közöltünk további részeredményeket, melyet az általam kifejlesztett algoritmussal el lehetett érni. Az egyik ilyen részeredmény felhasználásával, a matematikai részt kiegészítve, Jan Bouwe van den Berg és Jonathan Jaquette teljes egészében bizonyította a Wright-sejtést. Továbbá az általam kifejlesztett megbízható eljárást az utóbbi szerzők alkalmazták a Jones-sejtés bizonyítása során is.

2. fejezet

Kaotikus mozgások vizsgálata az inga kényszerrezgése során

J. H. Hubbard 1999-ben publikálta cikkét [47], amelyben behatóan tanulmányozta a fékezett inga kényszerrezgését, mint egyszerűnek tekinthető dinamikai rendszert. Vizsgálatai egy sejtés megfogalmazására sarkalták, miszerint az általa bemutatott inga periodikus mozgáspályái mellett kaotikusak is vannak. Bár az írás megjelenése után több publikáció is született a témában, azonban az úgynevezett kaotikus trajektóriák létezésének formális bizonyítása [12] csak jóval később, majd tíz évvel a sejtés közlése után került publikálásra. Ebben a cikkünkben már több, korábban publikált technikát kellett ötvöznünk a differenciálegyenletek trajektóriájának megbízható befoglalására szolgáló módszerrel.

A káosz létezése igazolására szolgáló megbízható technikát már több, az Hénon leképezés kaotikus régióival kapcsolatos állítás bizonyítására alkalmaztuk. De míg az Hénon-leképezés egy egyszerűnek mondható kétdimenziós leképezés, addig a fékezett inga kényszerrezgése egy differenciálegyenlettel adott mechanikai rendszer, azaz az eljárásunkat erre az esetre bővítenünk kellett az állítások bizonyításához. Továbbá Hubbard állítása nem egészen illeszkedett a Hénon-leképezésnél megszokott eljárásához. A Hubbard-féle kaotikus viselkedés megfogalmazásában az inga speciális viselkedései szerepeltek. Ezért szükség volt a klasszikusnak mondható káosz bizonyítás, és a Hubbard-féle definíció ekvivalenciájának a bizonyítására is. Ennek bizonyítására szintén egy megbízható eljárást valósítottam meg.

Bár társszerzőimmel beláttuk e nehezen megfogható tulajdonság egzisztenciáját, magukról a kaotikus trajektóriák kezdőpontjairól, vagy azok megtalálásának lehetséges módszereiről nem esett benne szó. Ezzel kapcsoltban publikáltuk az eredményeinket Lévai Balázs doktoranduszommal [55]. Ebben a cikkünkben bemutattunk egy olyan megbízható eljárást, mely optimalizálás segítségével képes volt az adott véges sorozatokhoz megfelelő kezdő pontot keresni.

Hubbard az eredeti cikkében a kaotikus viselkedésen felül azt is tárgyalta, hogy érdekes lenne egy kaotikus rendszer kontrollját megkonstruálni. Ingával kapcsolatos kontrollok már korábban is léteztek, de jelen rendszerre nem létezett ilyen módszer. Ebben a fejezetben az általam megkonstruált kontroll rendszert fogunk bemutatni a kaotikus féke-

zett inga kényszerrezgéseire.

2.1. A fékezett inga kényszerrezgéseinek matematikai modellje és a káosz definíciója

Jelen fejezetben a fékezett inga kényszerrezgéseit vizsgáljuk, mely egy 1 szabadsági fokkal rendelkező mechanikai rendszer. Egy m tömegű pontszerű test lóg egy l hosszú súlytalan, merev rúdon (lásd a 2.1. ábrán). Ez azt jelenti, hogy a pontszerű test egy kör mentén mozoghat, melynek sugara l. A vizsgált ingára három erő hat. Az egyik a gravitáció, amelynek nagysága g és függőlegesen lefelé hat. A másik a légellenállás, amelynek nagysága a test sebességétől függ, és iránya ellentétes a pontszerű test mozgásának irányával. A forgási sebesség és ezen erő hányadosa legyen konstans, és jelöljük ezt $(-\gamma)$-val, ahol $\gamma > 0$. A harmadik egy periodikus külső erő, amely hat az inga sebességére az inga minden pozíciójában. Ez a kényszererő legyen $A\cos t$, ahol A egy konstans és t az idő. Ekkor a rendszer felírható egy másodrendű differenciálegyenlettel:

$$mlx''(t) = -mg\sin x(t) - \gamma l x'(t) + A\cos t,$$

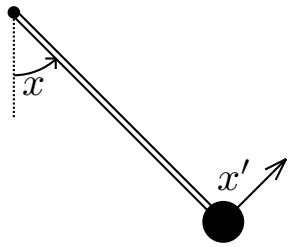
ahol x az inga függőlegessel bezárt szöge, és x' az inga forgási sebessége. A tömeget egységnyinek választva és az inga hosszával (l) leosztva az előző egyenletet, az alábbi egyenlethez jutunk:

$$x''(t) = -\frac{g}{l}\sin x(t) - \gamma x'(t) + \frac{A}{l}\cos t.$$

A további szabad paraméterek megválaszthatók úgy ($g = l$ és $A = l$), hogy az alábbi egyenletet kapjuk:

$$x''(t) = -\sin x(t) - 0.1x'(t) + \cos t.$$

Ez az egyenlet fogja vizsgálataink tárgyát képezni. Korábbi, a kapcsolódó irodalombeli eredményekből tudható, hogy az ilyen rendszerek numerikus megoldásai nagyon érzékenyek néhány pont közelében [19].



2.1. ábra. A fékezett inga kényszerrezgése.

A fenti egyenletet felírhatjuk az alábbi formában is:

$$x_1'(t) = x_2(t),$$

$$x_2'(t) = -0.1x_2(t) - \sin x_1(t) + \cos t,$$

ahol x_1 az inga szöge, míg x_2 az inga szögsebessége.

A káosz bizonyításához az egyik eszköz a Poincaré-leképezés. Mivel a kényszererő 2π periodikus, így a Poincaré-képeket a $t = 2n\pi$ időpillanatokban vesszük, ahol n egy egész szám. Az ezen időpillanatok között kialakult leképezések azonosak. Így a trajektória követéséhez elegendő a Poincaré-leképezést ($\mathcal{P} : \mathbb{R}^2 \mapsto \mathbb{R}^2$) iterálni, melyet az alábbi módon definiálhatunk:

$$\mathcal{P} : (x(0), x'(0)) \mapsto (x(2\pi), x'(2\pi)) .$$

Megjegyzendő, hogy ez a Poincaré-leképezés x-ben 2π periodikus, mivel az inga szöge is 2π periodikus, és egyik erő sem függ a körbefordulások számától. De amíg a hagyományos inga szimmetrikus x-ben is, a mi rendszerünk nem az, mivel a kényszererő nem szimmetrikusan hat az inga szimmetrikus szögeiben.

Hubbardnak a tekintett inga kaotikusságára vonatkozó tételének ismertetése előtt vezessük be az alábbi jelöléseket:

- Jelöljük a $[2k\pi, 2(k+1)\pi]$ időintervallumot I_k-val.
- $\epsilon_{k\in\mathbb{Z}}$ egy szimbólum, melynek értékkészlete $\{\ominus, \otimes, \oplus\}$.
- – Ha az I_k időintervallumban az inga *az óra járásával megegyező irányban pontosan egyszer* haladt át az alsó ponton, akkor azt mondjuk, hogy az inga az $\epsilon_k = \ominus$ szerinti mozgást végezte.
  – Ha az I_k időintervallumban az inga *nem* haladt át az alsó ponton, akkor azt mondjuk, hogy az inga az $\epsilon_k = \otimes$ szerinti mozgást végezte.
  – Ha az I_k időintervallumban az inga *az óra járásával ellentétes irányban pontosan egyszer* haladt át az alsó ponton, akkor azt mondjuk, hogy az inga az $\epsilon_k = \oplus$ szerinti mozgást végezte.

Megjegyzendő, hogy az alsó pont alatt azt értjük, hogy az inga szöge egyenlő $2k\pi$-vel, valamilyen egész k-ra. Ekkor az inga nincs egyensúlyban, mint ahogy ez a korábbi eredményeinkből látható. Azt is érdemes megjegyezni, hogy az ϵ_k szerinti mozgáson kívül léteznek egyéb típusú mozgások is. Például amikor az I_k időintervallum alatt az inga többször is átmegy az alsó ponton. Most az ilyen típusú mozgásokat figyelmen kívül hagyjuk, a kaotikus viselkedést csak a három definiált típust tartalmazó trajektóriákra mondjuk ki.

2.1. Tétel. *[Hubbard] Az összes, mindkét irányban végtelen hosszú $\epsilon_k \in \{\ominus, \otimes, \oplus\}$-val megadott sorozathoz létezik olyan $(x(0), x'(0))$ kezdőérték, amelyre az inga az I_k időintervallumok alatt az ϵ_k szerinti mozgást végzi.*

Ez azt jelenti, hogy tetszőlegesen előírhatjuk azt, hogy az inga az egymás utáni 2π hosszú időintervallumok alatt melyik irányba haladjon át az alsó ponton, vagy azt, hogy ne haladjon át az adott idő alatt ezen a ponton. Például előírhatjuk azt a sorozatot, mely szerint a $[0, 2\pi]$ idő alatt balra, míg a $[2\pi, 4\pi]$ idő alatt jobbra, majd a $[4\pi, 6\pi]$ idő alatt ne haladjon át, és végül a $[8\pi, 10\pi]$ idő alatt megint jobbra haladjon át az alsó ponton. A tétel

szerint ezt a sorozatot tetszőlegesen folytatva is létezik az ingának olyan kezdeti szöge és sebessége, melyből elindítva ezt az előírt sorozatot írja le.

Hubbard a cikkében [47] kimondta a tételt, de egzakt matematikai bizonyítást nem tudott adni rá, habár a bizonyítás egy lehetséges menetét leírta. Ezen alapult a korábbi, [12] cikkben szereplő bizonyítás.

A kaotikusságot bizonyító geometriai tulajdonságokat mutató régiók keresése csak a Poincaré-metszeteken alkalmazható. Jelen esetben ezen Poincaré-metszetek az inga szöge és sebessége terében vannak. A viselkedést mutató régiókat rendre L, M és R szimbólumokkal fogjuk jelölni, és ezt a teret szimbolikus térnek fogjuk nevezni. Kaotikus viselkedésének a létét az alábbi tétel segítségével bizonyíthatjuk.

2.2. Tétel. *Ha az (x, x') szimbolikus térben léteznek olyan L, M és R halmazok, amelyekre igaz a káosz létezésének a feltétele, azaz tetszőleges $\{Q_{\gamma_k}\}_{k\in\mathbb{Z}}$ (ahol $\gamma_k \in \{-1, 0, 1\}$ és $Q_{-1} = L$, $Q_0 = M$, $Q_1 = R$) sorozathoz mutathatunk egy pontot a szimbolikus térben, melynek a trajektóriája épp az adott sorozatot járja be, akkor a rendszer rendelkezik kaotikus régióval.*

A Poincaré-metszeteken megbízható számítógépes módszerekkel sikerült bizonyítani a fenti tételt, így már tudtuk, hogy a rendszer kaotikus, de a 2.1. Tétel ennél erősebbet állít. A tételben szereplő események a két Poincaré-metszet között bekövetkezett ingamozgásokat írják elő. A következő lépésben megmutattuk, hogy a kaotikus pontok ilyen típusú mozgásokat fognak végrehajtani a 2π hosszú idő alatt. Azaz a bizonyításunk következő részében az alábbi állítást bizonyítottuk:

2.3. Tétel. *Igaz, hogy az (x, x') szimbolikus térben kimutatott káosz ekvivalens az ingán definiált káosszal, azaz az L, M, R szimbólumok rendre ekvivalensek az $\epsilon_k = \ominus$, $\epsilon_k = \otimes$, $\epsilon_k = \oplus$ típusú mozgással.*

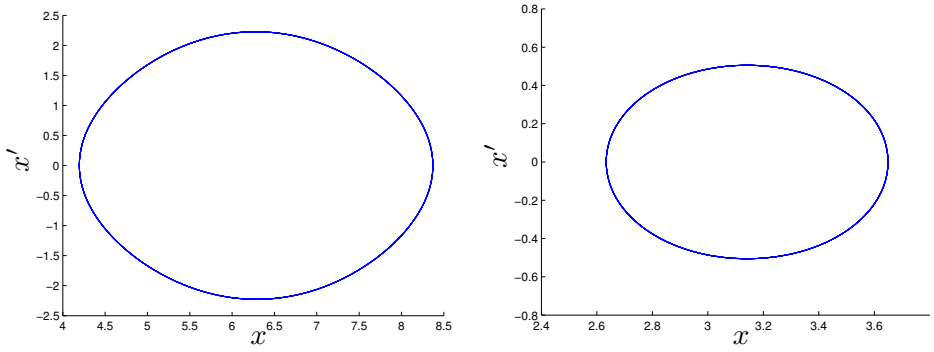
Ezen két tétel bizonyításával sikerült a Hubbard eredeti sejtését teljes egészében bizonyítani.

2.2. A periodikus pontok

A kaotikus viselkedésben fontos szerepet játszanak a periodikus pontok. Ezek helyének és stabilitásának vizsgálata szükséges a káosz bizonyításához. Korábbi eredményekből tudható [58], hogy ilyen rendszereknek legalább egy periodikus megoldása van, de ennél többre lesz szükségünk.

Jelen részben csak a Poincaré-leképezés fixpontjaival foglalkozunk, ugyanis léteznek olyan periodikus pontjai is az ingának, melyek 2π időközönként ugyanazt az állapotot veszik fel, de közben legalább egyszer körbefordulnak. Sejthető, hogy egy bizonyos értéknél nagyobb sebességgel mozgó inga nem lehet periodikus. Durva számításokkal az alábbi felső korlátot kapjuk a kaotikus inga sebességére:

$$|x_2| < \int_0^{2\pi} (1 - \cos t) e^{0.1t} dt < 10.1.$$



(a) Az alsó periodikus pálya. (b) A felső periodikus pálya.

2.2. ábra. A periodikus pályák.

A periodikus pontok kereséséhez egy egyszerű megbízható korlátozás és szétválasztás (B&B) eljárást alkalmaztunk. A keresési terület az

$$(x, x') \in [0, 2\pi] \times [-10.1, 10.1]$$

kezdő intervallum volt. A B&B eljárásunk olyan 2 dimenziós intervallumokat (I_i) generál a kezdő intervallumból, melyekre igaz az alábbi állítások valamelyike:

1. az I_i intervallumnak nincs közös pontja a $P(I_i)$ és $P^{-1}(I_i)$ legalább egyikével, vagy

2. az I_i intervallum kicsi (a felhasználó által beállított méretű), és van közös pontja a $P(I_i)$-vel és a $P^{-1}(I_i)$-vel is.

A periodikus pontok csak a második állításnak megfelelő halmazban lehetnek. Következő lépésként az ezen csoportban lévő közös ponttal rendelkező intervallumokat összevonjuk egy nagyobb intervallumba, amely tartalmazza mindkét kisebb intervallumot. Ezt mindaddig csináljuk, míg van ilyen intervallum a csoportban. Ezzel feltehetően pontosan annyi intervallumot kapunk, mint ahány periodikus pontja van a rendszernek, és mindegyiknek egy-egy garantált befoglalását nyerjük. Sajnos ez a technika egyelőre nem zárja ki, hogy egy-egy „dobozban” több vagy akár egy periodikus pont se szerepeljen.

Esetünkben ezen 2 dimenziós intervallumok az alábbiaknak adódtak:

$$(x, x')_1 = ([2.634272, 2.634274], [0.02604294, 0.02604485]) ,$$

$$(x, x')_2 = ([4.236893, 4.236894], [0.3926964, 0.3926973]) .$$

Megvizsgálva a trajektóriákat, azt sejtettük, hogy az első egy felső egyensúlyi pályát, míg a második egy alsó egyensúlyi pályát feltételez (lásd a 2.2. ábrát).

A létezés és az egyértelműség igazolásához a karakterisztikus multiplikátort fogjuk használni. Ezt az úgynevezett variációs egyenlet módszerrel határozzuk meg. Ehhez szükségünk van a jobboldalak x_1 és x_2 szerinti deriváltjaira, melyek a következők:

$$\begin{pmatrix} 0 & 1 \\ -\cos(x_1) & -0.1 \end{pmatrix} .$$

Vegyük észre, hogy az x_2'-nek az x_1 szerinti deriváltja tartalmazza az x_1-et, így szükségünk van a megoldásra is, melyet csak numerikusan befoglalva tudunk megadni. Így az alábbi differenciálegyenlet-rendszert vizsgáljuk:

$$\begin{aligned} z_1'(t) &= z_2(t), \\ z_2'(t) &= -0.1z_2(t) - \sin z_1(t) + \cos t, \\ z_3'(t) &= z_4(t), \\ z_4'(t) &= -0.1z_4(t) - z_3(t)\cos z_1(t). \end{aligned}$$

A z_1 és z_2 teljesen azonos az x_1-gyel, illetve x_2-vel, és ezek fogják a megoldást minden időpillanatra kiszámolni, míg a z_3 és z_4 a multiplikátorokat tartalmazzák.

Az eljárás során a differenciálegyenlet z_1, z_2 kezdőértékei legyenek az első fixpont koordinátái, valamint $z_3(0) := 1$ és $z_4(0) := 0$. Ezekkel az értékekkel kiszámolt Poincaré-leképezés z_3 és z_4 értékei legyenek egy A_1 2×2-es mátrix első oszlopa. Az A_1 mátrix második oszlopát hasonlóan számoljuk ki, csak a $z_3(0) := 0$ és $z_4(0) := 1$. Az így adódó mátrix:

$$A_1 = \begin{pmatrix} [169.6369, 169.6370] & [168.7925, 168.7926] \\ [152.9595, 152.9597] & [152.2012, 152.2014] \end{pmatrix},$$

melynek sajátértékei befoglalása:

$$\alpha_1^1 = [321.8363, 321.8368],$$

$$\alpha_1^2 = [0.001421, 0.001894].$$

Mivel abszolút értékben az egyik nagyobb, mint 1, a másik pedig kisebb, ezért az $(x, x')_1$ intervallum pontosan egy instabil fixpontot tartalmazhat.

Hasonlóan a másik intervallumra:

$$A_2 = \begin{pmatrix} [-0.7426217, -0.7426218] & [0.09101517, 0.09101522] \\ [-0.04890921, -0.04890920] & [-0.7123905, -0.7123904] \end{pmatrix},$$

melynek sajátértékei

$$\alpha_2^1 = [-0.7275062, -0.7275061] + \sqrt{[-0.01689209, -0.01689190]},$$

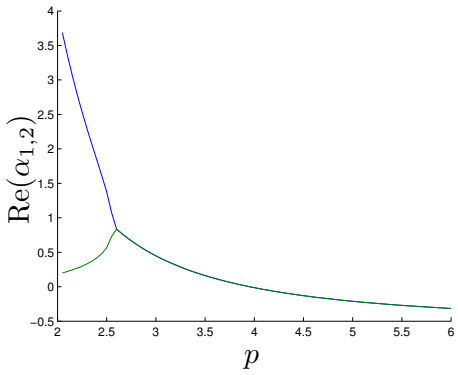
$$\alpha_2^2 = [-0.7275062, -0.7275061] - \sqrt{[-0.01689209, -0.01689190]}.$$

Mivel a sajátértékek abszolút értéke mindkét esetben kisebb mint 1, így ez egy stabil fixpontot tartalmaz.

2.3. Az egyszerű inga stabilizálása

Ismert az a tény, hogy az egyszerű inga felső egyensúlyi állapota instabil. Bizonyított [3], hogy a felfüggesztés megfelelő mozgatásával ezt stabillá lehet tenni. Egy ilyen eredményt érhetünk el a felfüggesztési pont adott periódusú és nagyságú függőleges irányú oszcilláló mozgatásával. Ennek a mozgásnak a gyorsulása legyen

$$\frac{8ap^2}{l}\sin(pt),$$



2.3. ábra. A sajátértékek valós részének alakulása.

ahol a a kitérés amplitúdója, l az inga hossza, és p a kitérések száma 2π idő alatt. Ekkor a hagyományos inga differenciálegyenlete az alábbi formában írható fel:

$$x''(t) = \left(-\frac{g}{l} + \frac{8ap^2}{l}\sin(pt)\right)\sin x - \gamma x'.$$

A stabilitás vizsgálatához alkalmazhatjuk a korábban is használt variációs egyenlet módszerét. Jelen esetben ismert, hogy az ingának két fixpontja van, melyek közül az egyik stabil (az alsó egyensúlyi állapot), a másik pedig instabil (a felső egyensúlyi állapot). Vizsgálataink most a felső egyensúlyi állapotra koncentrálódnak és ezt szeretnénk stabillá tenni. A fenti egyenlet periódusideje $\frac{2\pi}{p}$, azaz ilyen hosszan fogjuk vizsgálni a variációs egyenletrendszert. Ez idő alatt a stabilizálandó periodikus pálya ismert (az inga szöge konstans π, szögsebessége 0), így nincs szükségünk a pálya kiszámítására. Ezen észrevételekkel az egyenletrendszer az alábbi módon adható meg:

$$z_3'(t) = z_4(t),$$
$$z_4'(t) = \left(-\tfrac{g}{l} + \tfrac{8ap^2}{l}\sin(pt)\right) z_3(t) - \gamma z_4(t).$$

További vizsgálatainkban a kaotikus fékezett inga kényszerrezgéséhez legjobban hasonlító rendszert vizsgáltuk, azaz $\gamma = 1$ és $l = g = 9.81$. Megvizsgáltuk, hogy milyen gyorsan kell mozgatni a felfüggesztési pontot, ha azt szeretnénk, hogy a felső egyensúlyi állapot stabilizálódjon. Ehhez az kell, hogy teljesüljön az, hogy a fent említett differenciálegyenlet-rendszerrel kapott mátrix sajátértékeinek abszolút értékei 1-nél kisebbek legyenek. Konstans $a = 1$ és különböző p paraméterek mellett a 2.3. ábra jeleníti meg ezen értékeket.

2.4. A fékezett inga kényszerrezgésének stabilizálása

Az előző módszer analógiájára, a kaotikus inga stabilizálásával próbálkozunk. Legyen a fékezett inga kényszerrezgésének a felső instabil periodikus megoldásának a szöge az idő függvényében $\hat{x}(t)$. Az előző eredmények alapján sejthető, hogy egy $\hat{x}(t)$ irányú és

$\frac{8ap^2}{l}\sin(pt)$ értékkel gyorsuló felfüggesztési ponttal rendelkező fékezett inga kényszerrezgésének stabilizálódik a felső instabil egyensúlyi pályája valamely a és p paraméterek mellett. Ebben az esetben a középpont függőleges irányú gyorsulása:

$$\cos(\hat{x}(t))\frac{8ap^2}{l}\sin(pt) = \omega_f,$$

a vízszintes irányú gyorsulása pedig:

$$\sin(\hat{x}(t))\frac{8ap^2}{l}\sin(pt) = \omega_v.$$

Ekkor az alábbi formában írható fel a differenciálegyenlet:

$$x''(t) = (-1-\omega_f)\sin(x) + \omega_v\cos(x(t)) - 0.1x' + \cos(t).$$

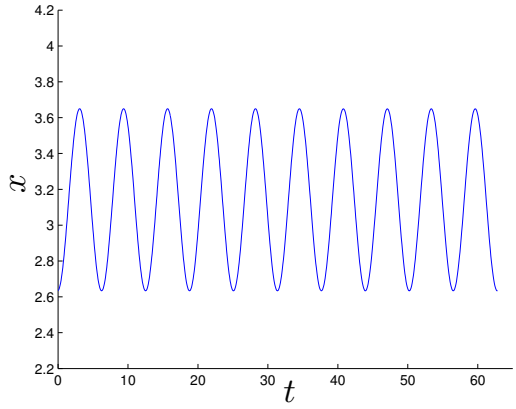
Az $a = 0.5$ és $p = 4$ paraméterekkel következett be egy stabilizálódás. Ezt mutatja a 2.4. ábra. A 2.4(a) ábrán az inga szögének alakulását láthatjuk az idő függvényében a felső instabil állapotban. A kontrollal ezt az állapotot szeretnénk stabilizálni. Egy közeli állapotból indított inga szögének alakulását láthatjuk a 2.4(b) ábrán. Majd ugyanezen állapotból a kontrollal ellátott rendszer alakulását mutatja a 2.4(c) ábra. Sejthető, hogy a kontrollált inga szöge tart a felső pályához és ez a kontroll stabilizálja a tekintett pályát.

A bizonyításhoz a Poincaré-metszeteken alkalmazott multiplikátor módszert alkalmazzuk. Ezen vizsgálathoz a kényszererő periódusának, és a középpont gyorsulásának periódusának összemérhetőnek kell lennie. Ez jelen esetben igaz, és a minimális közös periódus 2π.

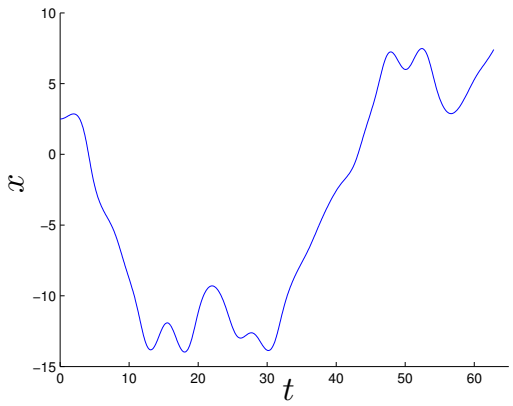
A sajátértékek meghatározásához az alábbi rendszert használtuk:

$$\begin{aligned}
z_1'(t) &= z_2'(t),\\
z_2'(t) &= -0.1z_2(t) - \sin\left(z_1(t)\right) + \cos(t),\\
z_3'(t) &= z_4'(t),\\
z_4'(t) &= \left(-1 - \cos\left(z_1(t)\right)\frac{8ap^2}{l}\sin(pt)\right)\sin(z_3(t)) +\\
&\quad + \left(\sin\left(z_1(t)\right)\frac{8ap^2}{l}\sin(pt)\right)\cos(z_3(t)) - 0.1z_4(t) + \cos(t),\\
z_5'(t) &= z_6'(t),\\
z_6'(t) &= \left(\left(-1 - \cos\left(z_1(t)\right)\frac{8ap^2}{l}\sin(pt)\right)\cos(z_3(t)) -\right.\\
&\quad \left. - \left(\sin(z_1(t))\frac{8ap^2}{l}\sin(pt)\sin(pt)\right)\sin(z_3(t))\right) z_5(t) - bz_6(t).
\end{aligned}$$

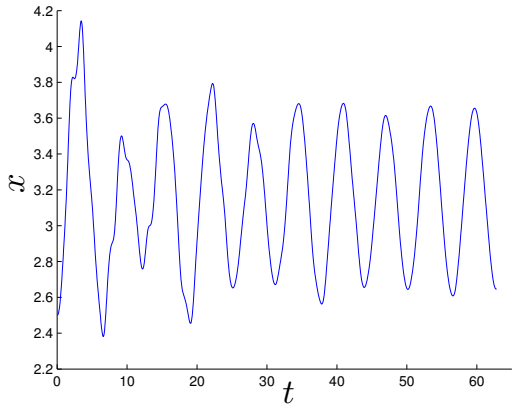
Az 1-2. egyenlet a felső instabil megoldás követéséhez szükséges. Itt a kezdő értékre a korábbi fixpontos módszert alkalmazzuk. A kapott értékek tartalmazzák az instabil megoldást. A 3-4. egyenlet a stabilizált trajektóriát számolja ki. Az itt kapott eredmények az



(a) Felső instabil pálya.



(b) Kontroll nélküli kaotikus mozgás.



(c) Kontrollált mozgás.

2.4. ábra. Az inga kényszerrezgésének stabilizálódása.

instabil megoldással stabilizált megoldást foglalják be. Ezt is a fixpontos kezdő dobozzal indítjuk. A 5-6. egyenlet pedig a mátrix elemeit számolják.

Ekkor a két sajátérték:

$$\lambda_1 = [-0.41408955, -0.41383926] + [0.60060509, 0.60292339]\ \mathrm{i},$$

és

$$\lambda_2 = [-0.41408955, -0.41383926] - [0.60060509, 0.60292339]\ \mathrm{i},$$

azaz ez a pálya a kontrollal stabil.

Kétféle stabilizálási eljárást ismertet a szakirodalom. Az egyik az, amikor a dinamikai rendszer állapotától függően befolyásoljuk a rendszert. Ezt vagy folytonosan, vagy diszkrét időközönként tehetjük meg. Ez a típusú kontroll, melyet „visszacsatolásos technikának” is neveznek, a kezdő állapotok sokkal szélesebb körére stabilizálja a kívánt állapotot. A másik eljárás az, amikor a dinamikai rendszer éppen aktuális állapotától nem függ a rendszerre ható erő. Ebben az esetben a kezdő állapotok sokkal kisebb halmazára tapasztalható a stabilizálódás.

Vegyük észre, hogy az ω_f és az ω_v gyorsulások nem függnek az éppen aktuális inga állapotaitól – sem a sebességétől, sem a szögétől. Ezen függvényekben csak a felső instabil pálya megoldása szerepel, melyet akár előre kiszámíthatunk és eltárolhatunk. Tehát a jelen problémán alkalmazott kontroll a nem visszacsatolásos technikák körébe tartozik.

2.5. Összefoglalás

Hubbard cikke alapján várható volt a fejezetben tárgyalt fékezett inga kényszerrezgésének kaotikussága [47], de a sejtés egzakt bizonyítása ismeretlen volt. Korábban adtunk erre a sejtésre egy megbízható módszerekkel segített bizonyítást [12], valamint a kaotikus trajektóriák keresésére is készítettünk egy megbízható eljárást [55]. Ezen felül Hubbard az eredeti cikkben felvetette a kaotikus rendszerek kontrolljának problémáját is, de erre nem tudott megoldást javasolni.

A jelen fejezet fő eredménye egy olyan megkonstruált kontroll, mely stabilizálja a kaotikus inga instabil megoldását. A rendszer általam konstruált kontrollja a Kapica-inga ötletén alapszik. Ez a kontroll az inga aktuális állapotától teljesen független, az inga felfüggesztési pontjának periodikus mozgatásán alapszik, mely előre megmatározható. A fejezet elején a technika alapötletének a hátterét és a periodikus pályákat mutattam be. A kiindulási ötlet egy egyszerű fékezett inga stabilizálása volt. Ezen a rendszeren bemutattuk a matematikus társszerzőimmel kifejlesztett megbízható technikánkat, melyre támaszkodva képesek vagyunk matematikai értelemben is bizonyítani az eljárás helyességét. A későbbiekben bemutattuk ennek a technikának a kiterjesztését a vizsgált rendszerre, illetve azokat a numerikus eredményeket, melyek igazolják, hogy a rendszer instabil megoldása stabilizálódik [13].

3. fejezet

Optimális körfedés megtalálása sokszögeken

A pakolási problémákat régóta nagy figyelem övezi, ezeket még napjainkban is sokat vizsgálják. Egyik népszerű feladat ebben a témakörben a körpakolás [76]. Ennek célja adott számú egybevágó, páronként diszjunkt körnek az egységnégyzetbe való legsűrűbb elhelyezésének megadása.

A feladat duálisának tekinthető probléma a körfedés. Ez a probléma sokkal nehezebbnek bizonyul, melyet a témában elért eredmények mennyisége is tükröz. Ebben az esetben adott számú egybevágó körrel az egységnégyzet legritkább lefedését keressük, ahol legritkább alatt azt értjük, hogy a lehető legkisebb sugarú körökkel fedjük le a négyzetet.

A tekintett témákban kétféle eredményt szoktak publikálni. Az egyik, amikor megpróbálnak minél jobb fedést/pakolást elérni, de a valódi optimumról nem mondanak semmit. A másik eset, amikor ténylegesen próbálják bizonyítani egy elhelyezési struktúrára, hogy az a tekintett körök számában optimális. A körfedési probléma eddig ismert legjobb eredményei megtalálhatók a [37] weblapon. További numerikus eredményeket taglal a [65] publikáció, bár megjegyzendő, hogy az ebben közölt eredmények azon felül, hogy optimalitást nem bizonyítanak, nem tekinthetők megbízhatónak sem, mivel a közölt pontok a számítógép pontatlanságát nem veszik figyelembe.

A megbízhatóság érdekében mi az intervallum-aritmetikát használjuk fel annak eldöntésére, hogy a körök egy adott elrendezése teljesen lefedi-e a tekintett sokszöget. A kérdés eldöntésére egy korlátozás és szétválasztás alapú eljárást fejlesztettünk ki.

Az eljárásunk hatékonyságát egy telekommunikációs feladaton mutatjuk be. A fő rádióadó-tornyok pozícióinak ismeretében határozzuk meg Magyarország rádióadásokkal való optimális lefedését. Az egyes rádióadó-tornyok által kibocsájtott rádióhullámok jó közelítéssel kör alakú területeket fednek le, és ezen területek összegét szeretnénk minimalizálni, mely arányos a rádióadó-torony sugárzási energiájának nagyságával. Ennél a problémánál a körök sugarai különbözőek is lehetnek. Az optimális sugárméretek meghatározására egy korlátozás és szétválasztás alapú megbízható optimalizáló eljárást alkalmazunk. Hasonló problémákat oldottak meg a [32]-ben, azonban nem megbízható számításokat alkalmaztak.

3.1. Az ellenőrző eljárás elméleti háttere

Jelen fejezetben egy körfedési problémát vizsgálunk. A tekintett feladatban a körök középpontjait adottnak vesszük. A köröket jelöljük $O_i = (x_i, y_i, r_i)$-vel, melyek középpontja x_i, y_i és a sugara r_i. A köröket tartalmazó halmazt jelölje $\mathcal{O} = \{O_i\}$ $(i = 1 \dots n)$. A vizsgálataink során csak a sugarak nagyságára koncentrálunk, melyek nem feltétlenül azonosak minden körre. Vezessük be az alábbi jelölést.

3.1. Definíció. *Jelölje $r = (r_1, r_2, \dots, r_n)$ a körök sugárhosszait tartalmazó vektort. Ezt a továbbiakban konfigurációnak fogjuk nevezni.*

Az eljárásunk alkalmas lesz minden zárt, véges és összefüggő területfedési probléma megoldására. Ezért a későbbiekben ezen terület pontjainak halmazát jelöljük $\mathbf{T}$-vel. Ez a terület a leggyakrabban vizsgált esetekben négyzet, de előfordulnak téglalapok ([46]), körök ([45]), szabályos háromszögek ([64]) és egyéb alakzatok is. Most definiáljuk, hogy mikor mondjuk azt, hogy a körök lefedik a tekintett $\mathbf{T}$ síkidomot.

3.2. Definíció. *Egy r konfigurációt megfelelőnek nevezünk, ha az általa reprezentált körök lefedik a tekintett területet ($\mathbf{T}$), azaz minden $(x, y) \in \mathbf{T}$ ponthoz létezik olyan $O_i \in \mathcal{O}$, melyre $d\left((x, y), (x_i, y_i)\right) = \sqrt{(x - x_i)^2 + (y - y_i)^2} < r_i$. Ellenkező esetben a konfigurációt nem megfelelőnek nevezzük.*

A megbízhatóság elérése érdekében a számítógéppel segített bizonyítás során intervallum-aritmetikát fogunk használni. Egy intervallumon két valós számot és azok között lévő valós számok halmazát értjük:

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

ahol $\underline{X}$ az intervallum alsó, míg az $\overline{X}$ a felső végpontját jelenti. Egy n dimenziós intervallumon az $X = (X_1, X_2, \dots, X_n)$ vektort értjük, ahol X_i $(i = 1, \dots, n)$ egy-egy egydimenziós intervallum.

Az intervallumok halmazát $\mathbb{I}$-vel, míg az n-dimenziós intervallumok halmazát $\mathbb{I}^n$-nel jelöljük.

Az egydimenziós X intervallum szélességén a

$$\mathrm{wid}(X) = \overline{X} - \underline{X},$$

míg egy n-dimenziós intervallum szélességén a

$$\mathrm{wid}(X) = \max_{i=1,\dots,n} \left(\overline{X}_i - \underline{X}_i\right)$$

értéket értjük. Egy n-dimenziós intervallum középpontját a

$$\mathrm{mid}(X) = \frac{1}{2}(\underline{X} + \overline{X})$$

kifejezéssel határozhatjuk meg, ahol $\underline{X} = (\underline{X}_1, \dots, \underline{X}_n)$ és $\overline{X} = \left(\overline{X}_1, \dots, \overline{X}_n\right)$.

Az intervallumos számítások esetén valós számok helyett intervallumokat használunk, melyekre így definiálni kell a műveleteket is. A valós számokon értelmezett elemi műveletek (Ω) intervallumos kiterjesztését az alábbi alakban adjuk meg:

3. Algoritmus. Az ellenőrző eljárás

Input:
- ϵ: a részintervallumok felhasználó által beállított minimális mérete,
- $\mathbf{T}$: a vizsgált halmaz.

Output:
- A geometriai feltétel teljesül az adott $\mathbf{T}$ halmaz minden pontjára, vagy
- mutat egy olyan ϵ-nál kisebb méretű intervallumot, mely megsérti az adott feltételt.

1. lépés Határozzuk meg a kezdő intervallumot, mely befoglalja $\mathbf{T}$-t.
2. lépés Tegyük be ezt az intervallumot a verembe.
3. lépés Vegyünk ki egy $v = (X, Y)$ intervallumot a veremből.
4. lépés Határozzuk meg a v intervallum legszélesebb oldalát.
5. lépés Ha a $\mathbf{T} \cap v \neq \emptyset$, és a $\mathcal{T}$ tulajdonság v-re nem teljesül, akkor

ha a v intervallum szélesebbik oldala kisebb, mint a felhasználó által adott érték, akkor

- kiírjuk v-t és STOP,

egyébként

- felezzük el v-t a szélesebbik oldala mentén, rakjuk be a két részintervallumot a verembe, és folytassuk a 3. lépéssel.

6. lépés Ha a verem üres, akkor írjuk ki, hogy a bizonyítás sikerült és STOP. Egyébként folytassuk a 3. lépéssel.

3.3. Definíció. *Az elemi műveletek intervallumos kiterjesztése:*

$$A \circ B = \{a \circ b \mid a \in A \text{ és } b \in B\}, \text{ ahol } A, B \in \mathbb{I}, \circ \in \Omega.$$

A fenti definíció alapján két nem nulla szélességű intervallumon végzett elemi művelet elvégzéséhez végtelen sok valós számon elvégzett számítás szükséges. Könnyen látható, hogy bizonyos műveletek esetében – a monotonitás miatt – a definíció szerinti eredmény véges számú valós művelettel is kiszámítható. Az intervallum-aritmetikáról további információk olvashatóak az [1, 2] közleményekben. A számítógépes megvalósítás során a C-XSC könyvtárat használjuk [25].

Az ellenőrzést egy intervallumos korlátozás és szétválasztás alapú eljárással végezzük. Az ellenőrizendő feltétel az, hogy minden $p = (x, y) \in \mathbf{T}$ pont valamely O_i körben benne van, azaz $d\left((x, y), (x_i, y_i)\right) < r_i$. Ezt a tulajdonságot jelöljük a későbbiekben $\mathcal{T}$-vel. Ezen tulajdonság intervallumos kiterjesztésén azt értjük, hogy $P = (X, Y)$ kétdimenziós intervallum minden egyes pontjára teljesül a $\mathcal{T}$ tulajdonság.

Az eljárás helyességét és végességére vonatkozó tulajdonságait a következő tételekben foglaljuk össze.

3.4. Tétel. *Tegyük fel, hogy rendelkezésre áll a $\mathcal{T}$ tulajdonság intervallumos kiterjesztésére egy ellenőrző eljárás, továbbá az ellenőrző rutin azt az eredményt adja, hogy a $\mathbf{T}$ minden pontjára teljesül a $\mathcal{T}$ tulajdonság. Ekkor az ellenőrző rutin az ellenőrizendő tarto-*

mány köré írt (I) kiindulási intervallumnak egy olyan felosztását állítja elő, hogy minden részintervallumra teljesül, hogy

1. *vagy ennek a részintervallumnak nincs közös pontja a vizsgált* $\mathbf{T}$ *tartománnyal,*

2. *vagy ennek a részintervallumnak minden pontja benne van valamely inputként kapott körben.*

Ez a tétel azt jelenti, hogy a 3. Algoritmus helyes. Ezek után nézzük meg, hogy az algoritmus valóban képes-e eldönteni, hogy a $\mathbf{T}$ tartomány rendelkezik-e a teljes lefedési tulajdonsággal.

3.5. Tétel. *Tegyük fel, hogy rendelkezésre áll* $\mathcal{T}$ *tulajdonság egy intervallumos kiterjesztése, mely minden pontra és annak valamely kis méretű környezetére képes eldönteni, hogy teljesül-e a* $\mathcal{T}$ *tulajdonság, továbbá az algoritmus* ϵ *paramétere nulla, és a* $\mathcal{T}$ *tulajdonság fennáll. Ekkor az ellenőrző rutin véges sok iterációs lépés után megáll pozitív eredménnyel.*

Vizsgáljuk meg, hogy abban az esetben, ha nem teljesül az adott halmazra a geometriai feltétel, akkor a 3. Algoritmussal milyen eredményre számíthatunk.

3.6. Tétel. *Tegyük fel, hogy rendelkezésre áll a* $\mathcal{T}$ *tulajdonság egy intervallumos kiterjesztése, továbbá az algoritmus* ϵ *paramétere nulla, és van olyan* $p \in \mathbf{T}$ *pont, hogy a* $\mathcal{T}$ *tulajdonság nem teljesül rá. Ekkor az ellenőrző rutin nem tud véges lépésben megállni, és nem ad eredményt arra vonatkozóan, hogy a tulajdonság teljesül-e.*

A fenti tételek bizonyítása hasonlóan történik, mint a [28] cikkben megtalálható hasonló tételek bizonyítása.

3.2. A fedést ellenőrző eljárás megvalósítása

A fent említett eljárás működéséhez fontos egy olyan algoritmus megvalósítása, mely képes megbízhatóan eldönteni egy v kétdimenziós intervallumról, hogy

- van-e közös pontja a $\mathbf{T}$ tartománnyal, és
- teljesül-e minden pontjára a $\mathcal{T}$ tulajdonság.

Az első kérdés eldöntése egyszerű azokban az esetekben, amikor az indulási intervallum minden egyes pontja benne van a tekintett tartományban, azaz az olyan 2 dimenziós intervallumok esetén, amiknek az oldalai párhuzamosak a tengelyekkel. Ekkor minden, az algoritmus által generált v kétdimenziós intervallumnak van közös pontja a $\mathbf{T}$ tartománnyal. Általános esetben szükségünk van egy olyan eljárásra, mely képes eldönteni egy tetszőleges zárt, véges és összefüggő területről, hogy van-e közös pontja a v intervallummal. Ez gyakorlatilag a "pont a poligonban" algoritmus intervallumos kiterjesztése. A megvalósítás során (lásd a 4. Algoritmust) azt vizsgáljuk, hogy a tekintett kétdimenziós intervallumtól balra lévő sávon mennyi további él megy át. Ha ez páratlan, akkor

4. Algoritmus. Intervallum a poligonban

Input: – P: poligon, mely a határpontjaival adott,
– v: a vizsgált kétdimenziós intervallum.
Output: – A v tartománynak van-e közös pontja a poligonnal.

1. lépés Ha a P poligon bármely oldalának van közös pontja a v tartománnyal, akkor a válasz igen és vége az eljárásnak.

2. lépés Az m számláló legyen 0.

3. lépés Vizsgáljuk meg a P poligon minden élét.

Ha a v intervallumból balra a végtelenbe indított sávba a poligon egy éle "alulról lép be", és a rákövetkező további élek "felül lép ki", vagy fordítva, akkor az m számlálót növeljük 1-gyel.

4. lépés Ha az m számláló páros, akkor nincs közös pontja az intervallumnak és a poligonnak (kívül van), egyébként igen (belül van).

5. Algoritmus. Intervallum valamely körben

Input: – $(r_i, x_i, y_i)(i = 1 \ldots, n)$: az aktuális körök sugarának nagysága és középpontjai,
– v: a vizsgált kétdimenziós intervallum.
Output: – A v tartomány benne van-e valamely körben.

1. lépés Vizsgáljuk meg az összes körre az alábbi tulajdonságot.

Ha a v intervallum és a vizsgált kör középpont távolságainak maximuma kisebb, mint a kör sugara, akkor igen (belül van).

2. lépés Ha egyik körre sem volt igaz az előző állítás, akkor nem tudtuk eldönteni, hogy a v intervallum minden pontja benne van-e valamelyik körben.

benne van, egyébként nincs közös pontja a poligonnal. Az intervallumos kiterjesztés következménye lehet, hogy a vizsgált kétdimenziós intervallumnak vannak belső, illetve külső pontjai is. Ez akkor következik be, mikor a poligon valamely éle metszi a vizsgált intervallumot.

A $\mathcal{T}$ tulajdonságot, azaz, hogy a körök uniója lefedi-e a vizsgált v tartományt, úgy tudjuk eldönteni, hogy megvizsgáljuk, hogy a kétdimenziós intervallum minden pontja benne van-e valamely körben, azaz minden pont távolsága valamely kör középponttól kisebb, mint az adott kör sugara. Matematikailag megfogalmazva, létezik-e olyan $O_i \in \mathcal{O}$, amelyre minden $(x, y) \in v$ pontra igaz, hogy $\sqrt{(x - x_i)^2 + (y - y_i)^2)} < r_i$.

Mint látható, az 5. Algoritmus olyan eset eldöntésére nem alkalmas, mikor a vizsgált tartomány nem egyetlen kör által van lefedve, hanem több kör fedi le együttesen. Továbbá olyan összetett kérdéseket sem tudunk eldönteni azonnal, hogy a vizsgált kétdimenziós intervallum poligonba eső része le van-e fedve körökkel. A 4. Algoritmus is csak azt képes eldönteni, hogy van-e olyan pont a vizsgált intervallumban, melyet fedési szempontból

vizsgálni kell. De azt, hogy melyek ezek a pontok, azt nem határozza meg az algoritmus. Azonban a 3. Algoritmus azokat az intervallumokat fogja tovább darabolni, melyekre már igaz lehet a megfelelő tulajdonság. A köröket nyitottnak tekintettük, azaz a határvonaluk nem számít lefedettnek. Így igaz az, hogy bármely pontra és annak kis környezetére tudjuk igazolni a lefedettséget. Megjegyzendő, hogy zárt körökre nem lenne igaz, mert ekkor a körvonal határpontjaira nem lenne teljesül, hogy kis környezetére is tudjuk bizonyítani a megfelelő tulajdonságokat.

Mivel a 3.5. Tétel feltételei teljesülnek, így igaz az alábbi tétel.

3.7. Tétel. *Ha a tekintett p poligon le van fedve körökkel, akkor a 3., 4. és 5. Algoritmusok $\epsilon = 0$ paraméterrel véges lépésben megállnak pozitív eredménnyel.*

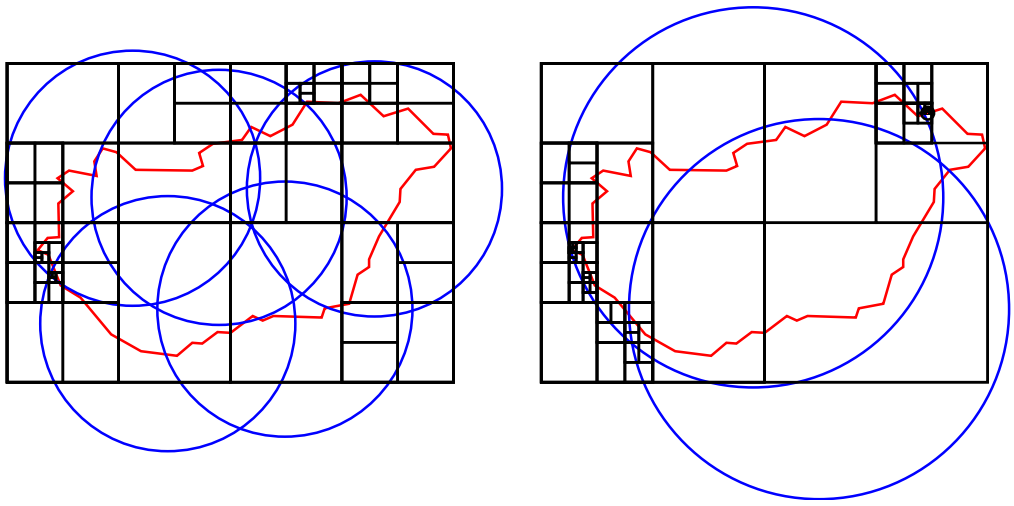
Az algoritmusok csak a pozitív eredmény igazolására alkalmasak. Azaz, ha nem fedik a körök a tekintett síkidomot, akkor az algoritmus nem áll meg véges lépésben, ahogy azt a 3.6. Tétel kimondja. Természetesen, ha az $\epsilon \neq 0$ és elegendően kicsi a pozitív válaszhoz, akkor véges lépésben megáll az algoritmus. Ha az ϵ paraméter pozitív értéket vesz fel, akkor a futás eredménye lehet sikertelen is, de ez nem minden esetben jelenti azt, hogy matematikai értelemben nem teljesül a geometriai feltétel.

3.3. Az ellenőrző eljárás futása

Most vizsgáljuk meg, hogyan alakulhatnak a számítógépes eredmények. A 3.1. ábrán látható két futás eredménye. Mind a két ábrán Magyarország határainak a körvonala látható, mely a példánkban a lefedendő terület. Mindkét esetben a nagy befoglaló téglalap az indulási kétdimenziós intervallum, mely tartalmazza a lefedendő poligon minden egyes pontját. Ezt darabolja az ellenőrző eljárás mindaddig, míg van közös pontja a lefedendő poligonnal, vagy a részintervallum nincs teljes egészében lefedve egyik kör által sem. Például a generált intervallumok közül a bal felsők olyanok, melyeknek nincs közös pontja a poligonnal.

A 3.1(a). ábrán egy olyan eset látható, amikor az algoritmus sikeres eredménnyel állt meg. Ekkor az ábrán látható darabolást kaptuk, melyre igaz a vizsgált feltétel. Látható az is, hogy a keletkezett, a lefedendő poligonnal közös ponttal rendelkező intervallumok olyanok, melyek legalább egy kör által teljesen le vannak fedve. Azaz igaz, hogy a vizsgált poligon teljesen le van fedve. A 3.1(b). ábrán látható egy olyan futás képe, mikor az algoritmus sikertelen eredménnyel állt meg. Ekkor az algoritmus egy, a paraméterül kapott ϵ értéknél kisebb méretű intervallumot adott vissza, mely nincs teljes egészében lefedve, de van közös pontja a lefedendő területtel. Ezt az intervallumot az ábrán egy kis körrel jelöltük a jobb felső sarokban. Mint látható az algoritmus nem azonnal találta meg ezt az intervallumot.

A futási idő a fenti esetekben csupán pár másodperc. Továbbá az ellenőrző algoritmus párhuzamos változatát is megvalósítottuk, mellyel közel lineáris gyorsulást tudtunk elérni a magok számától függően [17]. Ez már elegendően gyors ahhoz, hogy egy összetettebb algoritmusba illeszthessük. Például egy optimalizáló eljárást helyezzünk az ellenőrző eljárás fölé, mellyel képesek leszünk optimális lefedést találni különböző problémákra.



(a) Egy pozitív eredmény képe. (b) Egy negatív eredmény képe.

3.1. ábra. Az ellenőrző eljárás által generált intervallumok.

3.4. Az optimalizáló eljárás

Több probléma esetén az alkalmazott eszközök erőforrásigénye az általuk lefedett terület sugarának négyzetével arányosan alakul. Ilyen tulajdonsággal rendelkeznek az általunk vizsgált szenzorok és sugárzók is. Így a felhasznált sugárzási energia négyzetesen nő a lefedett terület sugarával. A célunk az eszközökhöz rendelt sugarak négyzetösszegét minimalizálni:

$$f\left((r_1, r_2, \ldots, r_n)\right) = \sum_{i=1}^{n} r_i^2 \to \min \tag{3.1}$$

A feladatunk olyan megfelelő konfiguráció megtalálása, amelyre a (3.1) célfüggvény értéke minimális.

Most bebizonyítunk néhány, a konfigurációkhoz kapcsolódó egyszerű állítást, melyeket az optimalizálás során felhasználunk.

3.8. Állítás. *Ha egy $r = (r_1, r_2, \ldots, r_n)$ konfiguráció megfelelő, akkor bármely $r' = (r'_1, r'_2, \ldots, r'_n)$ konfiguráció is megfelelő, ha $r'_i \geq r_i$ minden i-re (lásd a 3.2(a) ábra).*

Bizonyítás. Mivel az r konfiguráció megfelelő, így az általa reprezentált körök lefedik a vizsgált poligont. Ha ezen körök közül bármelyiknek megnöveljük a sugarát, akkor a kapott körök szintén lefedik azt. Az utóbbi gondolatot ismételve minden i-re látható, hogy r' is megfelelő konfiguráció. ■

Hasonlót mond ki a nem megfelelő konfigurációkra is a következő állítás.

3.9. Állítás. *Ha egy $r = (r_1, r_2, \ldots, r_n)$ konfiguráció nem megfelelő, akkor bármely $r' = (r'_1, r'_2, \ldots, r'_n)$ konfiguráció is nem megfelelő, ha $r'_i \leq r_i$ minden i-re (lásd a 3.2(b) ábra).*

Bizonyítás. Hasonlóan az előző állítás bizonyításához. ■

A fenti állításokban látható, hogy a kiemelt konfigurációk jellemzik az összes ezeknél nagyobb, illetve kisebb sugarakkal rendelkező konfigurációkat. Ezért a konfiguráció halmazokat érdemes egységesen kezelni.

3.10. Definíció. *Jelölje* $C = ([\underline{c_1}, \overline{c_1}], \ldots, [\underline{c_n}, \overline{c_n}])$ *azon konfigurációk halmazát, amelyek* i*-edik komponense a* $[\underline{c_i}, \overline{c_i}]$ *intervallumba esik. A* C *halmaz két kitüntetett szerepű konfigurációja a következő:* $\underline{C} = (\underline{c_1}, \ldots, \underline{c_n})$ *és* $\overline{C} = (\overline{c_1}, \ldots, \overline{c_n})$ *(lásd a 3.2(c) ábrát).*

Most vizsgáljuk meg, hogy ezen konfiguráció halmazból, mely konfigurációk lehetnek optimálisak.

3.11. Állítás. *Ha egy* C *halmaz* $\underline{C}$ *és* $\overline{C}$ *kitüntetett konfigurációi nem megfelelőek, akkor az optimális megoldás nem lehet a* C *halmazban.*

Bizonyítás. Mivel a $\overline{C}$ konfiguráció nem megfelelő, a 3.9 Állításból következik hogy a $C' = ([0, \overline{c_1}], \ldots, [0, \overline{c_n}])$ halmaz összes konfigurációja is nem megfelelő, mely tartalmazza C összes elemét. Azaz a C halmaz minden eleme nem megfelelő konfiguráció. Az optimális megoldás azonban egy megfelelő konfiguráció, így nem lehet eleme a C halmaznak. ■

Hasonló állítást mondhatunk ki a megfelelő konfigurációkra is.

3.12. Állítás. *Ha egy* C *halmaz* $\underline{C}$ *és* $\overline{C}$ *kitüntetett konfigurációi megfelelőek, akkor az optimális megoldás nem lehet a* C *halmazban.*

Bizonyítás. A C halmazon a (3.1) célfüggvény a $\underline{C}$ pontban veszi fel a minimumát, mivel $\underline{C}$ egy megfelelő konfiguráció. Ebből következik, hogy az optimális megoldás nem lehet a megadott halmazban. Nyitott körök esetén a $\underline{C}$ konfiguráció sem optimális, ugyanis létezik olyan kis $\epsilon > 0$, hogy $C_\epsilon = (\underline{c_1} - \epsilon, \ldots, \underline{c_n} - \epsilon)$ is megfelelő konfiguráció és $f(C_\epsilon) < f(\underline{C})$. ■

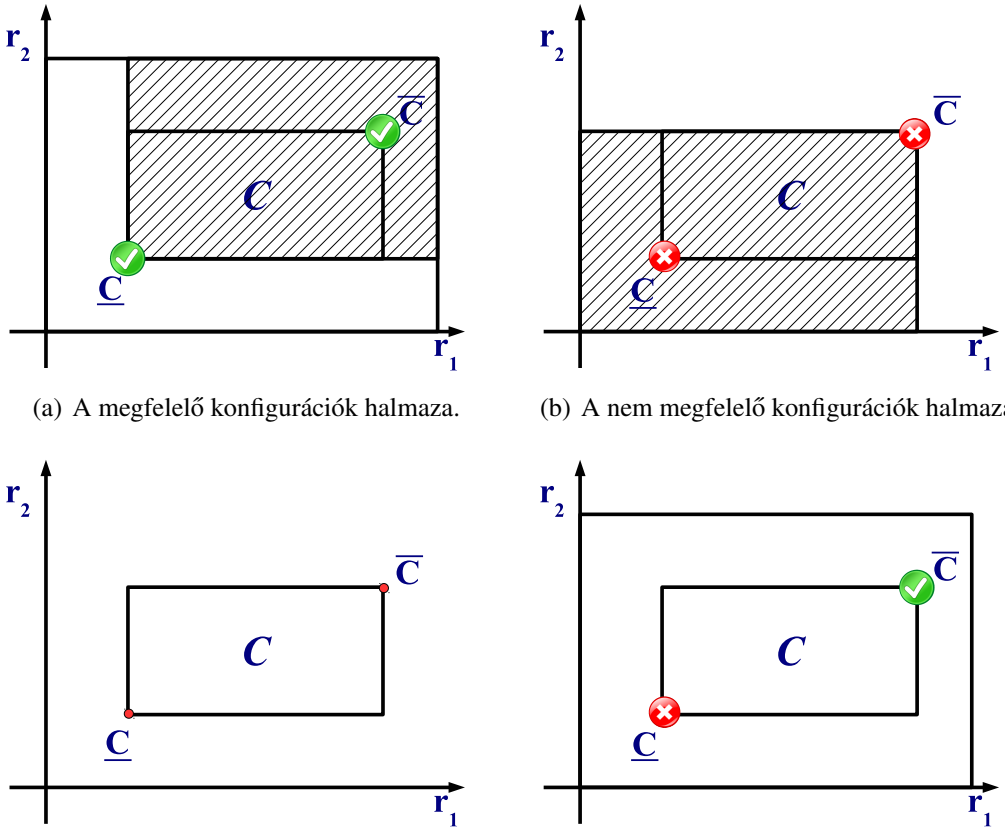
A következő állításban megmutatjuk, hogy az alábbi elrendezés nem lehetséges az általunk vizsgált esetben.

3.13. Állítás. *Nem létezik konfigurációk olyan* C *halmaza, amelynek* $\underline{C}$ *konfigurációja megfelelő és* $\overline{C}$ *konfigurációja pedig nem megfelelő.*

Bizonyítás. Mivel a $\underline{C}$ konfiguráció megfelelő, a 3.8 állításból következik, hogy a $\overline{C}$ is megfelelő konfiguráció, ami ellentmondás. ■

A korábbi megállapításokat az alábbi tételben foglaljuk össze.

3.14. Tétel. *Csak olyan* C *halmaz tartalmazhatja az optimális megoldást, amelynek* $\underline{C}$ *konfigurációja nem megfelelő, és* $\overline{C}$ *konfigurációja megfelelő (lásd a 3.2(d) ábrát).*



(a) A megfelelő konfigurációk halmaza.

(b) A nem megfelelő konfigurációk halmaza.

(c) A definiált konfiguráció halmaz és kitüntetett pontjai.

(d) Az optimumot tartalmazó konfiguráció halmaz.

3.2. ábra. Konfigurációk halmaza és jóságuk viszonyai.

A fenti tétel általánosabban is alkalmazható. A tétel általánosításához szükséges feltételek, hogy a célfüggvény szigorúan monoton legyen, illetve hogy a feltétel is egyféle monotonitási tulajdonsággal rendelkezzen. Ekkor a fenti állítások igazak maradnak és így a tétel is alkalmazható. Azaz például egy gömbfedésnél is alkalmazható a tétel, ahol a célfüggvény nem négyzetes, hanem köbös is lehet.

Az optimum megtalálása céljából itt egy korlátozás és szétválasztás alapú eljárást alkalmazunk. A szétválasztás során azon konfiguráció halmazok kerülnek be a listába, melyek alsó sarka nem megfelelő konfiguráció, míg a felső sarka megfelelő konfiguráció. Mint tudjuk a 3.14. Tételből, csak ezek tartalmazhatják az optimális megoldást. A hatékony keresés miatt mindig azt a halmazt vesszük ki, mely a legjobb alsó korlátot adja a halmazon lévő célfüggvény értékére. Ha a felhasználó által megadott pontossággal megtaláltuk a globális optimumhoz közeli megoldást, akkor megállunk és kiíratjuk azt (6. Algoritmus). A bemutatott eljárás a hagyományos intervallumos globális optimalizáló algoritmus ötletén alapszik [67, 72]. Sajnos az ott kifejlesztett gyorsító technikák nagy része nem alkalmazható, mert azok többsége (pl. az intervallumos Newton-módszer) a célfüggvény deriváltjának ismeretén alapszik, mely jelen esetben nem áll rendelkezésünkre

6. Algoritmus. Az optimalizáló eljárás

Input: – ϵ: a kívánt pontosság a globális optimumhoz képest %-os eltérésben,
– $\mathbf{C}$: a sugarak lehetséges nagyságai.
Output: – A globális optimumnál maximum ϵ%-kal rosszabb megoldás.

1. lépés Tegyük be a $\mathbf{C}$ többdimenziós intervallumot a verembe, ha $\overline{\mathbf{C}}$ megfelelő konfiguráció (egyébként nincs megfelelő konfiguráció $\mathbf{C}$-ben).

2. lépés Vegyük ki azt a C intervallumot, mely a legkisebb célfüggvényérték alsókorláttal rendelkezik.

3. lépés Határozzuk meg a C intervallum legszélesebb oldalát és számoljuk ki a C intervallumon a célfüggvény értékeinek a befoglalását, F-et.

4. lépés Ha F intervallum szélessége ϵ%-nál kisebb az F legnagyobb értékéhez (szupré-mumához) képest, akkor

- kiírjuk $\overline{C}$-t és STOP,

egyébként

- Felezzük el C-t a szélesebbik oldala mentén és legyen C_1 az alsó és C_2 a felső intervallum.
- Ha a $\overline{C}_1$ megfelelő megoldás, akkor tegyük be a verembe.
- Ha a $\underline{C}_2$ nem megfelelő megoldás, akkor tegyük be a verembe.

5. lépés Folytassuk a 2. lépéssel.

[88].

Azt, hogy valóban a globális optimum egy közelítését találjuk meg, az alábbi tétel mondja ki.

3.15. Tétel. *Tételezzük fel, hogy rendelkezésünkre áll egy ellenőrző eljárás, mely megbízhatóan eldönti, hogy egy konfiguráció megfelelő-e. Ekkor a 6. Algoritmus mindig olyan eredménnyel tér vissza, amely egy megfelelő konfiguráció, továbbá nincs másik, az ellenőrző eljárás által megfelelőként értékelt konfiguráció, mely ϵ%-kal jobb célfüggvényértékkel rendelkezne.*

Bizonyítás. Tételezzük fel, hogy a globális optimális konfiguráció több mint ϵ%-kal jobb, mint az algoritmus által visszaadott. Az algoritmus a 4. lépésben csak azokat a konfiguráció halmazokat nem teszi vissza a listába, melyek a 3.14. Tétel szerint nem tartalmazhatják a globális optimumot. A felosztás miatt minden konfiguráció vagy eldobásra kerül (mert nem optimális), vagy bekerül a listába valamely konfiguráció halmazban. Azaz az algoritmus megállásakor a listában kell lennie a globális optimumnak.

A 3. lépésben meghatározott F befoglaló intervallum szélessége kisebb mint ϵ%-a a maximális értéknek. Az algoritmus által visszaadott $\overline{C}$ konfiguráció célfüggvényértéke biztosan nem nagyobb, mint ezen F intervallum felső értéke. Így az itt kapott alsó korlát ϵ%-kal kevesebbel lehet jobb, mint a visszaadott célfüggvényérték.

A feltételezés szerint a globális optimum több mint ϵ%-kal jobb, mint az általunk visszaadott érték. Azaz, a listában lévő konfiguráció halmazokra vonatkozó alsó korlátnak kisebbnek kell lennie, mint az utoljára kivettnek. Azonban mivel a minimális értékkel rendelkezőt vesszük ki a 2. lépésben, így ellentmondásra jutottunk. Azaz a tétel igaz. ■

A megvalósítás során a célfüggvény értékekre vonatkozó alsó korlátot úgy kapjuk meg, hogy intervallum aritmetikával kiszámoljuk a célfüggvény értékét. Az alkalmazott technika révén így egy garantált befoglalást kapunk a célfüggvény értékére. Ennek az alsó szélét használhatjuk megbízható alsó korlátként, ami alapján rendezhetjük a konfiguráció halmazainkat. Mivel mindig a legkisebb alsó korláttal rendelkező konfiguráció halmazt vizsgáljuk, ennek hatékony megtalálása érdekében a halmazokat prioritási sorban tároljuk, mely mindig a legkisebb elemet tartalmazza a lista elején.

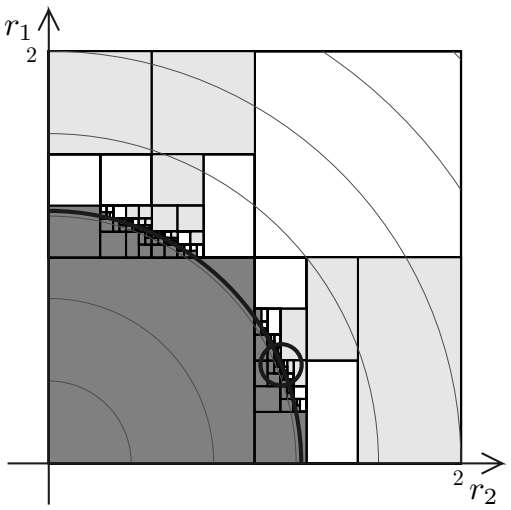
Az algoritmus bizonyos százalékos pontossággal találja meg az optimális megoldást. A 4. lépésben a megállási feltételt kicserélhetnénk abszolút korlátra is, ekkor az algoritmus által adott megoldástól a globális optimum maximum ϵ értékkel lehet jobb. Azért választottuk a relatív, százalékos megoldást, mert bár a kicsinyítésre és nagyításra az optimális megoldás és a százalékos eltérés sem érzékeny, de az abszolút eltérés már igen. A következő alfejezetben tárgyalt tesztesetek is elég nagy nagyságrendi eltéréseket mutatnak.

Az optimalizáló eljárás egy futásának eredményét jelenítjük meg a 3.3. ábrán. Itt egy egységnégyzet 2 körrel való optimális lefedését kerestük. A generált r_1 és r_2 sugarú intervallumok láthatók az ábrán. A sötét szürke konfiguráció halmazok azok, melyeket az algoritmus azért dobott el, mert a felső sarkuk nem volt megfelelő konfiguráció, míg a világos szürkéket azért, mert az alsó sarkuk megfelelő volt. Körrel jelöltük az algoritmus által megtalált megoldást. Az optimalizáló által talált megoldással azonos célfüggvényértékkel rendelkező megoldásokat a vastag ívvel jelöltük. Amennyiben mégsem ez az optimális megoldás, a valódi optimális megoldás a fehér négyzetek – a körívnél jobb értékekkel rendelkező – pontjaiban lehet.

3.5. Eredmények

Az algoritmusunk illusztrálására kétféle feladatot oldottunk meg. Mindkét esetben bemutatunk néhány érdekes és jellemző eredményt az optimalizáló eljárásunk használatára. Az optimum megtalálását 1%-os pontossággal kerestük. Az első esetben az egységnégyzetet fedtük le optimálisan különböző adott középpontokkal. Ezen futások közül néhányat láthatunk a 3.4. ábrán.

A 3.4(a) ábrán láthatjuk azt az esetet mikor az egységnégyzetet két átellenes csúcspontjából szeretnénk lefedni. Ekkor az algoritmus által talált optimális sugarak 1.12890625 és 0.48046875. A nagyobb sugár az átellenes oldalt körülbelül 0.52-nél metszi, így a körök ténylegesen lefedik a négyzetet. A sugarak négyzetösszege $1.50527954\ldots$ A vizsgált esetben az elméleti optimum az $(1 + (1 - x)^2) + x^2$ függvény minimuma lenne (zárt köröket feltételezve), ahol $x \in [0, 1]$ a kisebbik kör sugara. Átrendezve a $2(x - 0.5)^2 + 1.5$ függvényt kapjuk, melynek minimuma $x = 0.5$-nél van és értéke 1.5. Erre $1.5/1.5052 \approx 0.9965$, azaz az általunk megadott megoldás valóban 1%-nál közelebb van az optimális



3.3. ábra. Az optimalizáló eljárás által generált intervallumok.

megoldáshoz.

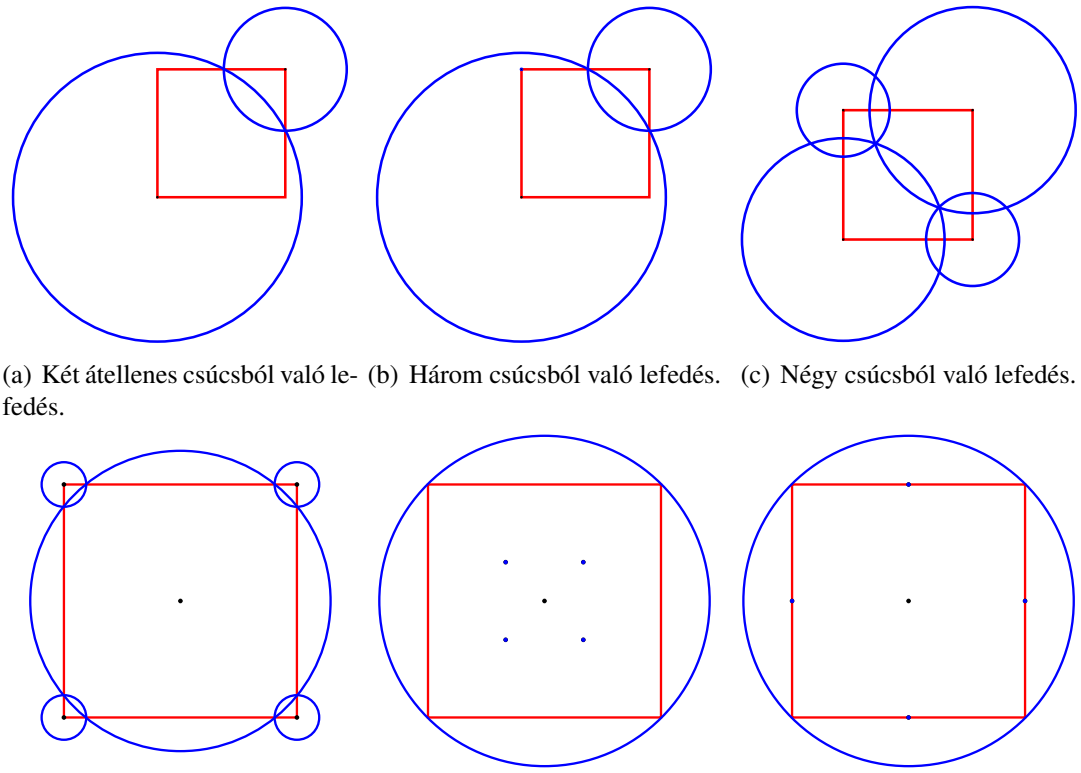
Továbbra is ez a 2 körös lefedés marad az optimális, ha egy harmadik csúcspontot is beveszünk a lehetséges körök középpontjai közé. Ezt láthatjuk a 3.4(b). ábrán. Ekkor a harmadik kör sugara 0.00390625 az algoritmus által adott megoldásban. Az algoritmus a konfiguráció halmaz felső csúcspontját adja vissza, amely elméletileg a felezgetések során sosem lesz 0. Ezt a hatást láthatjuk ebben az eredményben. Természetesen ezzel együtt is belefér az általunk megadott százalékos korlátba. Az optimalizáló eljárás által generált intervallumok láthatók ezen feladat esetében a 3.3. ábrán.

További csúcspont hozzáadása már jelentősen javít az optimum értékén, mint ahogy az a 3.4(c) ábrán is látható. A nagyobb sugarak nagysága egyöntetűen 0.783203125, míg a kisebbeké 0.359375000. A célfüggvény értéke körülbelül 1.50671, ami látszik, hogy rosszabb, mint a korábbi elméleti 1.5-es eredményünk. Ebben a konstrukcióban az elméleti optimum a $2x^2 + 2\left((\sqrt{2}/2 - x)^2 + (\sqrt{2}/2)^2\right)$ függvény minimuma lenne, ahol x a kisebbik kör sugara. Átrendezve a $4\left(x - \sqrt{2}/4\right)^2 + 1.5$ függvényt kapjuk, melynek 1.5 a legkisebb értéke, azaz megegyezik a korábbi 2 és 3 csúcspontos esettel. Ezen a példán látható (bár itt az optimumok megegyeznek), hogy ha a százalékos eltérésünk megengedi, egy a globális optimumtól teljesen eltérő struktúrájú megoldást ad az algoritmus. Természetesen a korlátot szűkítve ezek a félrevezető struktúrák kiesnek és csak a globális optimumot tartalmazó struktúrák maradnak a vizsgálandó listában. Jelen esetben a korábbi 2 körös konstrukció is a listában volt, csak a számítások során éppen ennél volt az első eset, amikor a megadott korlát alá kerültünk.

A negyedik esetben bevettük a négyzet középpontját is a lehetséges középpontok közé. A megoldás a 3.4(d) ábrán látható. Ekkor a kisebb körök sugara 0.095703125, míg a középsőé 0.64453125. A sejtésünk az, hogy az a struktúra az optimális, mikor egy nagy kör van. Azonban az optimum értéke ekkor kisebb mint 0.4520569, míg az 1 körös megoldásban az optimum 0.5. Tehát az általunk megadott megoldás jobb, mint a megoldásunkból sejtett közelítés. Az elméleti optimumok egyébként ebben a struktúrában a 0.1 és a

$\sqrt{0.41}$ nagyságú sugarakkal rendelkező körök lennének, amikor 0.45 az optimum értéke.

A 3.4(e) és a 3.4(f) esetekben megpróbálkoztunk más, a csúcspontoktól eltérő pontokat felvenni a körök középpontjainak. Mint látható itt az általunk talált optimális megoldás egyetlen, a középpontból induló nagy kör.



(a) Két átellenes csúcsból való lefedés.

(b) Három csúcsból való lefedés.

(c) Négy csúcsból való lefedés.

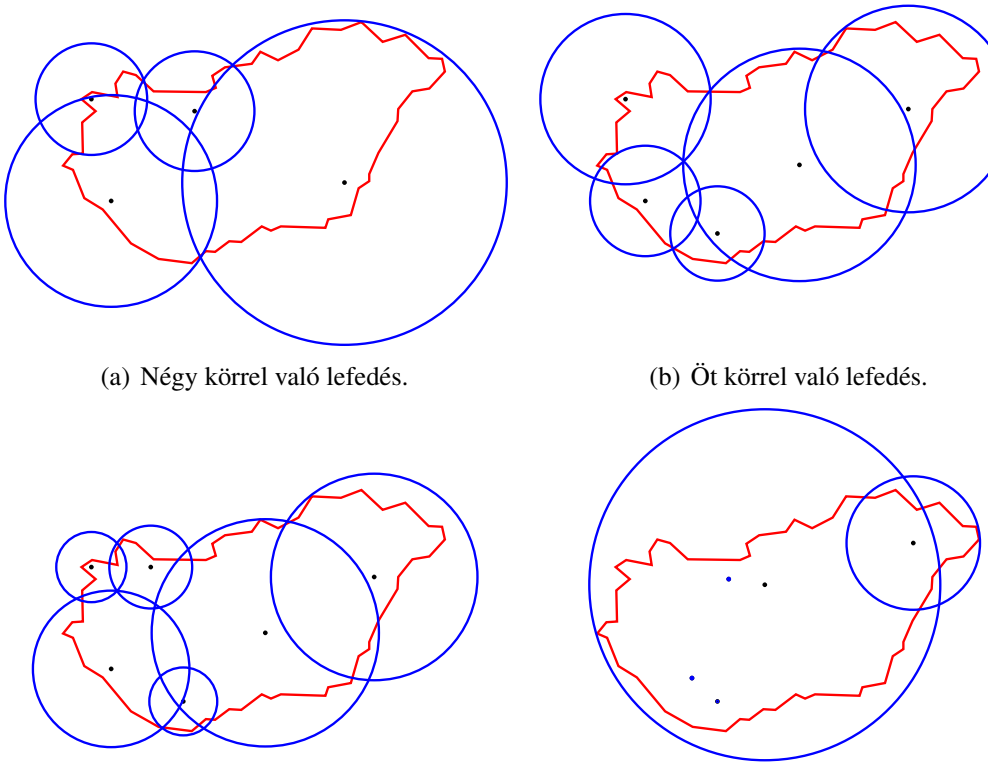
(d) Négy csúcsból és a középpontból való lefedés.

(e) Négy belső- és a középpontból való lefedés.

(f) Oldalfelező- és a középpontból való lefedés.

3.4. ábra. Az egységnégyzet lefedésének néhány érdekes esete.

A következő esetben bemutatunk néhány már a korábban említett Magyarországot jelképező poligont lefedő esetet. Ebben az esetben a középpontokat szintén a futás elején határoztuk meg véletlenszerűen (több magyarországi nagyvárosból választottunk véletlenszerűen). A 3.5(a)-3.5(c) ábrákon bemutatunk 3 Magyarországot lefedő optimális megoldást 4, 5, illetve 6 kör esetén. A 3.5(d) ábrán pedig egy olyan esetet látunk, amikor 5 körrel próbáltuk lefedni, de az optimális megoldás csak 2 kört használ.



(a) Négy körrel való lefedés.

(b) Öt körrel való lefedés.

(c) Hat körrel való lefedés.

(d) Két kört használó lefedés.

3.5. ábra. Magyarország lefedésének néhány érdekes esete.

3.6. Összefoglalás

A szakirodalomban több, a különböző pakolási, illetve fedési feladatok megoldása során fölmerülő geometriai tulajdonság ellenőrzésére szolgáló algoritmust adtak meg. Sajnos az intervallum-aritmetika alapú befoglaló függvényekkel ezek az eljárások jellemzően nem voltak kiegészítve, ez a megfelelő eszköztár nem állt rendelkezésre. Így valódi megbízható módszerek nem voltak elérhetők. Példáinkban egy a megbízható megoldás tekintetében viszonylag bonyolult geometriai esetet oldottunk meg. Korábban, már a káosz létezésének bizonyítása során, egyszerűbb geometria tulajdonságok ellenőrzésére szolgáló eljárásokat ki kellett fejlesztenem, azonban a jelenlegi eredmények ezeken túlmutatnak. Továbbá a megbízhatóság mellett az eljárások hatékonyságára is ügyelnem kellett, mert több paraméter változtatása mellett, az optimalizálás célfüggvényére alkalmaztuk ezt az eljárást.

Ebben a fejezetben bemutattam egy olyan hatékony és megbízható optimalizálási módszert, mely képes megtalálni tetszőleges ($\epsilon > 0$) pontossággal az optimális körök sugarainak nagyságát egy tetszőleges poligon alakzat lefedéséhez. A szokványos optima-

lizálási algoritmusokkal ellentétben itt bizonyítottan a globális optimumot találjuk meg, a probléma bonyolultságához képest viszonylag hatékonyan. Több fedési példán bemutattuk az algoritmus használhatóságát.

4. fejezet

Neurális hálók megbízhatóságának vizsgálata

A mesterséges neuronhálók tanítása során köztudottan használnak optimalizálást. De van egy másik terület is a neuronhálókkal kapcsolatban, ami a mai szakirodalomban egyre nagyobb hangsúlyt kap, és melyben úgyszintén megtalálható az optimalizálás. Ez a neuronhálók verifikálása, robusztusságuk vizsgálata. Ez a témakör egyre hangsúlyosabbá fog válni, köszönhetően a neuronhálók rohamos terjedésének és fejlődésének. A felhasználók elvárják, hogy minél megbízhatóbbak legyenek az alkalmazott neuronhálók, így a feladat kutatása fontos területté válhat. Mivel verifikálásról van szó, így a verifikálók megbízhatósága is kulcsfontosságú szerepet tölt be ezen a területen. A fent említett két témában szerezett tapasztalataimat felhasználva kezdtünk a terület mélyebb kutatásának doktorandusz hallgatómmal, Zombori Dániellel. Ebben a fejezetben ismertetni fogom a kutatott területet, valamint mutatok egy konstruktív módszert, mellyel kihasználjuk a jelenlegi legelfogadottabb verifikáló eljárás gyenge pontját, azaz „meghekkeljük” azt, ezzel rávilágítva a rendszer hiányosságára.

4.1. A mesterséges neurális hálók fejlődése

Manapság egyre inkább elterjedt megoldásnak számít különböző gépi tanulási problémákra a mesterséges neurális háló használata. Meglepő módon a neurális hálók elméleti hátterét már a 20. században kidolgozták. A neuronok alapját képező Perceptron-modellt már 1958-ban megalkotta Frank Rosenblatt, a több réteget használó hálózat pedig már 1965-ben megjelent [48]. A backpropagation algoritmussal pedig többen is [33, 51] foglalkoztak a ’60-as évek környékén. A 2000-es évek előtt megszületett az első konvolúciós háló is, aminek alapvetően a modern változathoz hasonló konvolúciós és pooling rétegei voltak.

Ezek után azonban egy jó ideig nem történt semmilyen előrelépés sem a neurális hálókat, sem a mesterséges intelligenciát tekintve. Ezt az időszakot nevezték a mesterséges intelligencia telének (AI winter).

A következő nagyobb előrelépés a 2010-es években történt, ekkor ugyanis két olyan probléma is megszűnt, ami addig hátráltatta a neurális hálók elterjedését. Ahhoz, hogy

a neurális hálók hatékonyak legyenek, nagy mennyiségű adatra van szükség, valamint egy megfelelő erősségű hardverre, amely ezt az adathalmazt képes feldolgozni és ezzel tanítani a neurális hálót. Ebben az időszakban kezdtek megjelenni az okostelefonok, okoseszközök és velük együtt a különböző alkalmazások, továbbá a 20. századhoz viszonyítva jelentősen megnőtt a tárolókapacitás is. Ezeknek köszönhetően a tárolt adatmennyiség nagysága drámai növekedésnek indult. Szintén erre az időszakra tehető az adatok párhuzamos feldolgozására, illetve a neurális hálók hatékony tanítására alkalmas (és viszonylag olcsón, nagyobb mennyiségben) elérhető GPU (Graphical Processing Unit) hardver megjelenése is.

4.2. Adverzális támadások neuronhálók ellen

Az adverzális támadásokat, melyeket szokás ellenséges példáknak is nevezni, először 2013 környékén kezdték vizsgálni többek közt Szegedy és társai [77]. Ahogy a cikkben leírják, ilyen példa alatt olyan perturbált, kicsit megváltoztatott bemenetet kell érteni, amely képes megtéveszteni a neurális hálót, az az nem a kívánt osztályba sorolni az inputot. Ilyen esetben, ha például képfelismerő hálóról van szó, a módosítás egyáltalán nem látható.

Az adverzális támadást többféleképpen is lehet kategorizálni. Az egyik szempont, hogy a támadás célzott-e vagy nem. A nem célzott (untargeted) esetben a fő cél az, hogy a neurális háló rossz eredményt adjon az adott példára, vagyis másik osztályba sorolja be, mint amibe az érintetlen, nem perturbált bemenet eredetileg volt. A másik, azaz a célzott esetben pedig azt is el szeretnénk érni, hogy nem csak hogy rossz eredményt adjon a hálózat, hanem pont olyan osztályba sorolja be a példát, amit mi megadtunk. Alapvetően ez az eset tűnik veszélyesebbnek, de az előző sem elhanyagolható.

Egy másik megközelítés a háló hozzáférhetőségét teszi a középpontba. Ez alapján beszélhetünk white box és black box támadásokról. Az előbbi esetben hozzáférünk a háló modelljéhez és az alapján konstruáljuk meg az adverzális támadást, a másik esetben pedig nincs hozzáférésünk a neuronhálóhoz, a legjobb esetben is csak kiértékelni tudjuk az adott példákat.

Egy adverzális támadás generálása során alapvetően azt szeretnénk elérni, hogy a példát, amit átadunk, egy adott osztályba soroljuk be. Majd ez alapján definiálhatunk egy költségfüggvényt:

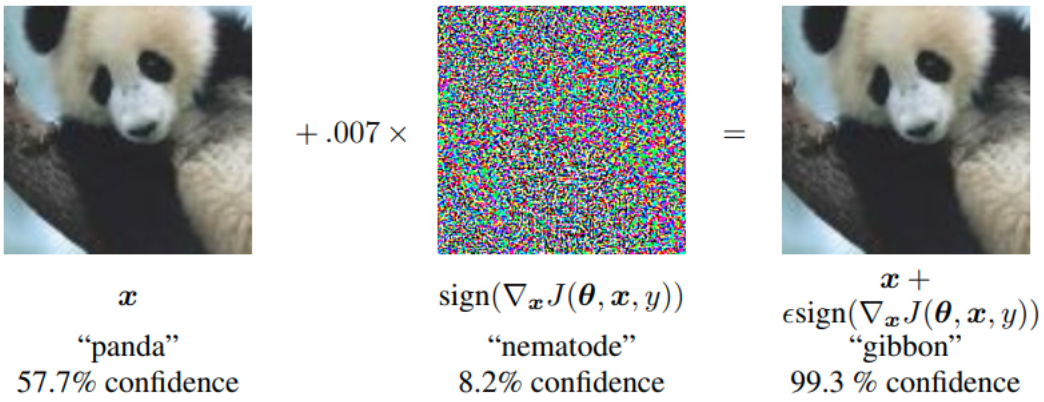
$$\frac{1}{2}||y_{goal} - \hat{y}(\vec{x})||_2^2,$$

ahol az y_{goal} az elérni kívánt kimenetet jelenti, az $\hat{y}(\vec{x})$ pedig a neuronháló által visszaadott kimenetet. Ha a hálózat súlyait és biasát konstansnak vesszük, akkor a paraméter, amit tudunk optimalizálni az a bemeneti vektor lesz. Az így generált bemenet egy zajos kép lesz, mivel csak az a cél, hogy egy adott osztályba soroljuk be az inputot, ezért a költségfüggvényen kicsit módosítani kell a következőképpen:

$$\frac{1}{2}||y_{goal} - \hat{y}(\vec{x})||_2^2 + \lambda||\vec{x} - x_{target}||.$$

Ebben az esetben azt is figyelembe vesszük a megfelelő x bemenetünk megkeresése közben, hogy az hasonlítson egy másik bemenetre. Ezáltal generálhatunk olyan bemenetet, amely nagyon hasonlít egy adott osztályra, azonban a háló másképpen fogja osztályozni, mint ahogyan elvárnánk. Egy másik megközelítése az adverzális példa generálásának, ha egy már meglévő példát módosítunk úgy, hogy megtévesszük a hálózatot.

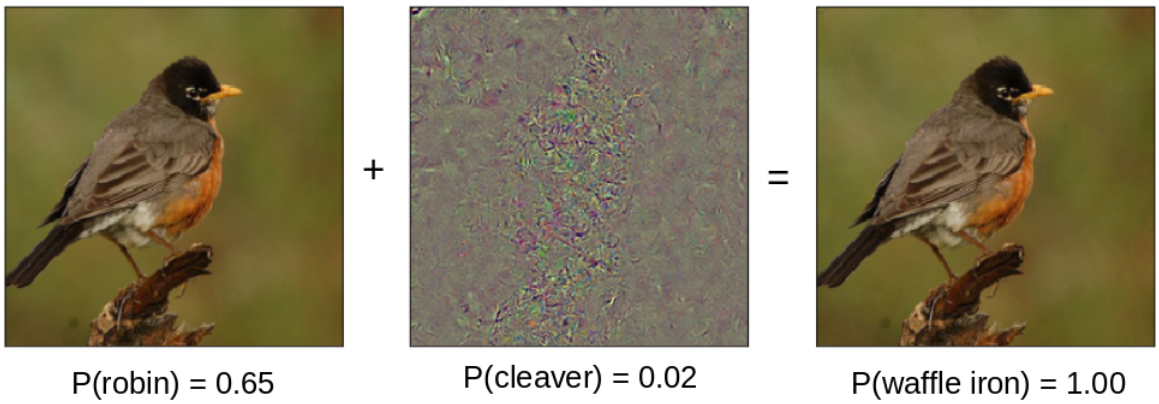
Az ellenséges példa generálására használható egyik white box módszer a FGSM, vagyis Fast Gradient Sign Method. A módszer lényege, hogy az adott inputhoz ki kell számítani a gradienst majd ennek az előjeles egész részét (Sign) megszorozva egy η -val, ad egy zajt, amit hozzáadva az eredeti bemenethez megkapjuk az adverzális példát [42]. Egy ilyen módszerrel létrehozott példa látható a 4.1. ábrán.



4.1. ábra. Az eredeti és perturbált kép (nem saját kép, interneten szabadon fellelhető).

Látható, hogy amit a hálózat eredetileg egy pandának ismerne fel 57.7%-os bizonyossággal, azt egy kismértékű zaj hozzáadása után már gibbonnak ismeri fel 99.3% bizonyossággal. Ez a módszer kiterjeszthető iteratív algoritmussá is azáltal, hogy minden lépésben kis mértékű zajt adunk a képhez.

Egy másik, szintén white box támadás, a PGD vagyis a Projected Gradient Descent. Ennek a módszernek a lényege, hogy az adverzális példa keresést egy korlátos optimalizálási feladatként kezeli, maximalizálja a modell költségfüggvényét (hibás osztályozás mértékének nagyságát) egy adott inputon, miközben a perturbációt egy meghatározott (valamilyen norma szerinti) ϵ értéknél alacsonyabban tartja. Az algoritmus lépései hasonlók, mint a korábban tárgyalt (iteratív) Fast Gradient Sign Method esetén, annyi különbséggel, hogy van egy extra projekciós lépés. Ez lényegében annyit jelent, hogy a módosítás után, ha szükséges, úgy változtatjuk az inputot/zajt, hogy az a korlátnak megfeleljen.



4.2. ábra. PGD módszerrel generált zaj és ellenséges példa (nem saját kép, interneten szabadon fellelhető).

Az előző módszerek, mint írtam, white box módszerek, vagyis ismerjük a támadni kívánt neurális hálózat súlyait és a bias-okat. Valós helyzetben azonban általában nincs ilyen mély ismeretünk a modellről, csak arra van lehetőségünk, hogy példákat adjunk a modellnek és megnézzük a kimenetét, vagyis csak black box támadásokat hajthatunk végre. Ehhez használható technika az *Adversarial Transferability* [68]. Egy modell becsapható akkor is, ha nem férünk hozzá a belső szerkezetéhez, elég egy úgy nevezett *substitute* (helyettesítő) modellt létrehozni, amely ugyanazt a feladatot látja el, mint a célmodell. A substitute modellhez létrehozott ellenséges példák aztán használhatók az eredeti modell ellen is, még akkor is, ha az architektúrájuk különbözik. Sőt, több különböző struktúrájú modell egyesítése (Ensemble) sem akadályozza meg ezt a jelenséget.

Ez pedig problémákhoz vezethet. Manapság ugyanis egyre több területen alkalmaznak különböző gépi tanulásos módszereket. Egy okostelefon kamerájánál, ami egy kép készítése után becímkézi (osztályozza) a fotókat, nem feltétlenül probléma, ha bizonyos képek nem megfelelő címkét kapnak. Azonban vannak olyan területek is, ahol fontos, hogy egy ilyen biztonsági rés ne legyen kihasználható. A neurális hálózatokat komolyabb használatuk előtt mindenképp robusztussá kell tenni az adverzális támadásokkal szemben is.

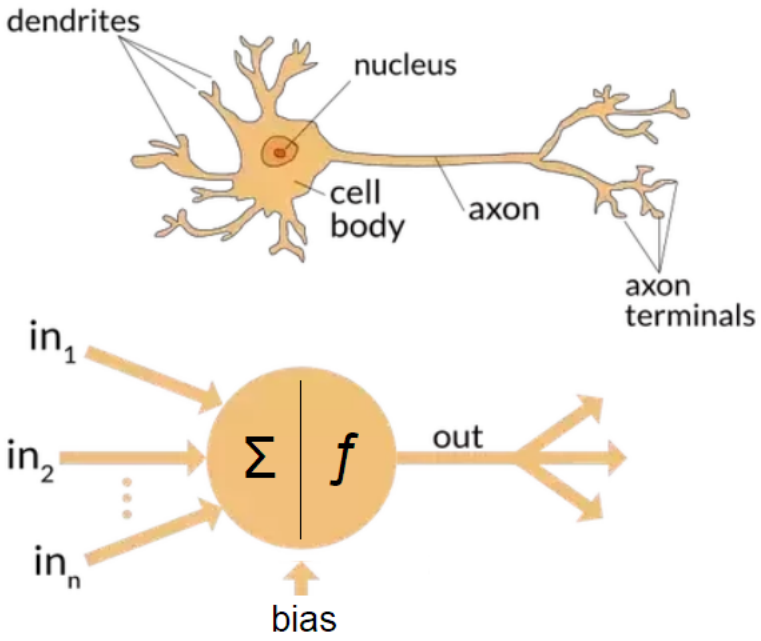
Az adverzális támadások ellen nem könnyű védekezni, azonban vannak bizonyos lehetőségek, amelyek csökkentik az ilyen irányú sebezhetőséget. Az egyik ilyen lehetőség az, hogy a különböző módszerekkel (például a korábban említettekkel) generált példákat is hozzátesszük a tanító adathalmazunkhoz, ezáltal egy robusztusabb hálót kapunk eredményül. Ezt a módszert *adversarial training*nek nevezik. Hátránya ennek az eljárásnak, hogy az általa biztosított robusztusság korlátozódik a választott adverzális példákat generáló módszerre, így más típusú támadásokra nézve még mindig sebezhető marad. Egy másik védekezési lehetőség, mely szintén a tanítási fázisban történik, a *defensive distillation* [69]. Papernot és szerzőtársai szerint a hálózatban található nagyobb (adverzális) gradiensek megkönnyítik az ellenséges példák generálását, ezért azokat kell úgymond jobban elsimítani (*smooth*-tá tenni a hálózatot). Így a hálózat kevésbé lesz érzékeny a kisebb perturbációkra, röviden robusztusabb lesz.

Itt csak néhány példát említettem, de látható, hogy viszonylag sok lehetőség van arra,

hogy adverzális példát generáljunk egy hálózathoz. A különböző védekezési módszerek, melyek jelenleg elfogadottak, nem különösebben biztonságosak, ezért fontos ezzel a területtel foglalkozni, kutatni.

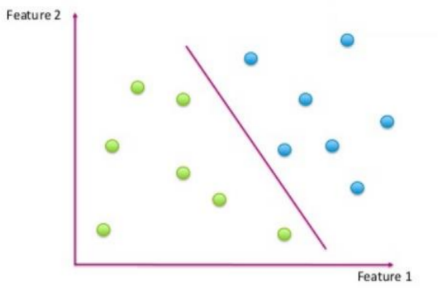
4.3. Neurális hálók

Ahogy korábban is említettem, a mesterséges neuron modelljét már jóval a neurális hálók elterjedése előtt megalkotta Frank Rosenblatt, és Perceptron-modellnek nevezte el. A modell lényege, hogy a biológiai neuronhoz hasonlóan alakították ki, ez a hasonlóság a 4.3. ábrán is látható. Egy mesterséges neuronnak van valahány súlyozott bemenete (dendrit) és általában egy kimenete (axon). Ezen felül tartozik hozzá egy aktivációs függvény is, amely az inputok szummáján lesz végrehajtva.



4.3. ábra. A biológiai- és a mesterséges neuron (nem saját kép, interneten szabadon fellelhető).

Ez még az eredeti modell esetén egy egyszerű küszöbölést jelentett, és így a neuron kimenete bináris volt (-1 vagy 1). Ha a neuron kimeneti értéke 1 volt, akkor úgymond tüzelt/aktiválódott a neuron. Egy neuronnal alapvetően egy kétosztályos osztályozási feladatot lehet megoldani abban az esetben, ha a feladat lineárisan elválasztható. Ebben a kontextusban az 1-es kimenet jelentette azt, hogy az adott osztályhoz tartozik az input, a -1-es pedig azt, hogy nem tartozik ehhez az osztályhoz (így a másik osztályhoz kell, hogy tartozzon). Ennek a vizuális ábrázolása látható a 4.4. ábrán.



4.4. ábra. Lineárisan elválasztható osztály (nem saját kép, interneten szabadon fellelhető).

A neuron működését a súlyok és a bias határozzák meg, ezért egy tanítás során ezeket a paramétereket akarjuk megtanulni, vagyis úgy meghatározni az értéküket, hogy a hálózatunk minél több példát soroljon be jó osztályba, vagy visszautalva a grafikus példára, minél pontosabban tudjuk elhelyezni a két osztályt elválasztó egyenest. Ehhez Rosenblatt meg is fogalmazott egy tanítási algoritmust, melynek a lényege, hogy alapvetően addig megyünk végig a tanító példánkon és módosítjuk a súlyokat, ameddig a predikciónk meg nem egyezik minden egyes esetben az adott példához tartozó osztályértékkel.

A való életben azonban ritka, hogy egy osztályozási feladat pontosan elválasztható, ezért általában egy jól működő hibafüggvény optimalizálására/minimalizálásra szoktak törekedni. Ilyen hibafüggvény (melyet a szakirodalomban *error* mellett *loss* vagy *cost function*-nek is neveznek) lehet például a Mean Squared Error

$$E(w) = \frac{1}{N}\sum_{i}^{N}(t_i - o_i)^2,$$

vagyis az átlagos négyzetes hiba, ahol a t_i az adott példán elvárt (target) értéket, az o_i pedig a kapott értéket (output) jelenti. Egy másik ismert hibafüggvény a Cross-Entropy

$$-\sum_{i=1}^{N} y_i \log \hat{y}_i,$$

amelyet leginkább osztályozási feladatoknál szoktak használni. A formulában az N az osztályok számát, az $\hat{y}_i$ a hálózat által jósolt értéket az i. osztály esetén, az y_i pedig az ehhez tartozó elvárt értéket jelöli.

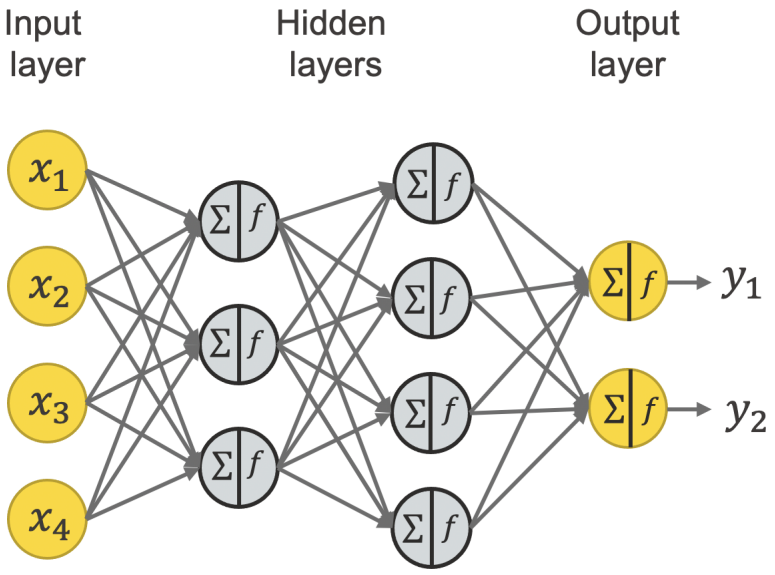
Egy ilyen függvény minimum helyének a megkeresését a Gradient Descent algoritmus teszi lehetővé. A módszer lényege, hogy iteratív módon csökkenti a célfüggvény értékét az adott függvény deriváltjának (vagy gradiensvektorának) meghatározásával. Az algoritmus hátránya, hogy érzékeny arra, hogy honnan inicializáljuk (csak lokális minimumot talál meg), illetve hogy milyen a tanulási ráta beállítása ("mekkorát lépjen a csökkenés irányába"). Egyik sem triviális feladat. Megállási feltétel többek között lehet egy adott maximális iterációs szám, vagy például a gradiensvektor nagysága.

`Gradient Descent:`

```
while eta*abs(gradient(x)) > eps:
x = x - eta*gradient(x)
```

4.4. Mesterséges neurális hálók

Mivel egy neuron reprezentációs ereje viszonylag alacsony, ezért hálózatot kell belőlük építeni ahhoz, hogy ne csak lineárisan elválasztható (bináris) osztályozási feladatok legyenek megoldhatók, hanem többek közt a többosztályos problémák is. Az ilyen hálózatokat nevezzük (mesterséges) neurális háló(zato)knak, vagy csak egyszerűen neuronhálóknak. A dolgozat során ezeket az elnevezéseket fogom majd használni. Egy neuronokból épített mesterséges neurális háló a következő, 4.5. ábrán látható. Alapvetően többféle neuronháló is létezik, az ábrán lévőt, amelyik felépítését tekintve az egyik legegyszerűbb, FeedForward Neural Network-nek nevezik. Egy ilyen háló felépítését és tulajdonságait fogom részletezni a következőkben.



4.5. ábra. Neurális hálózat példa (nem saját kép, interneten szabadon fellelhető).

A neuronok rétegeket alkotnak, és a rétegek megfelelően összekapcsolva adják azt, amit neurális hálónak neveznek. Rétegtípusonként máshogy alakul, hogy hány neuron található egy rétegben, ezen típusok bemutatása esetén majd lesz is szó erről.

Az első (az ábrán input layer-ként hivatkozott) réteg a bemeneti réteg, a neuronok száma itt megegyezik a megoldandó (osztályozási) feladat attribútumainak számával. Ahogy a neve is mutatja, ezen rétegen keresztül kerül be az adat a hálózatba. Egy másik fontos réteg a kimeneti réteg (output layer), ebben az esetben a neuronszám (osztályozási feladat esetén) az osztályok számától függ. A kimeneti értékek általában valós számok (bizonyos esetben valószínűségi értékek, ez mutatja, hogy a hálózatnak beadott példa milyen mértékben tartozik az egyes példákhoz. Ebben a rétegben amelyik neuronnak a legnagyobb a

kimeneti értéke, ahhoz az osztályhoz tartozik a példa. Az utolsó típus a rejtett réteg, amely szintén fontos része a neuronhálónak, a be- és kimeneti réteg között helyezkedik el. Ezen rétegek száma, illetve a bennük található neuronok száma nincs előre meghatározva, nem függ közvetlenül az adott feladattól. Ezen paraméterek meghatározására vannak különböző módszerek, de a legegyszerűbb megoldás addig módosítgatni a paramétereket, ameddig el nem érjük a kívánt eredményt. FeedForward hálózat esetén (ahogy ez a fenti ábrán is látható) egy adott réteg neuronja össze van kötve a következő réteg minden neuronjával, így egy neuron bemenete a hálóban az előző rétegben található neuronok kimenete lesz.

Ha van egy neurális hálónk, melyet megfelelően betanítottunk és van egy példánk, amiről el akarjuk dönteni, hogy melyik osztályhoz tartozik, akkor egy Feedforward neurális háló esetén a Feedforward neural network kódrészleten látható műveleteket kell elvégezni:

```
Feedforward neural network:
output = input
for l in layers:
  # (mátrix)szorzása az adott réteg súlyaival
  # bias hozzáadása
  output = output @ l.W + l.b
  # aktivációs függvény használata
  output = l.act_fun(output)
```

A műveletek elvégzése után kimeneti rétegünk neuronszámától függő számú különböző eredményünk lesz, melyek azt fogják meghatározni, hogy az adott példa mennyire tartozik egy adott osztályhoz. Ez alapján ahol a legnagyobb az érték, az lesz az adott példa osztálya.

4.5. Aktivációs függvények

Az aktivációs függvény (Activation/Transfer function) egy fontos része a neurális hálózatoknak. Alapvetően a neuron bemenetei mellett ettől is függ, hogy a neuron "tüzelni" fog-e. A Perceptron modellben használttal ellentétben a manapság használt aktivációs függvények nem bináris kimenetűek, hanem egy adott intervallumon belül akármilyen értéket felvehetnek. Emellett még fontos az is, hogy a függvény deriváltja számítható legyen, ennek a tanításnál van szerepe.

Egy neuronháló esetében fontos szempont, hogy milyen aktivációs függvényt választunk. Különböző feladatokhoz más-más függvény lehet a megfelelő, emellett a hálózat szerkezete szintén befolyásoló tényező ebben a döntésben. Ugyanis nagyobb rétegszám mellett megjelenik a *vanishing gradient* (eltűnő gradiens) problémája, ami lényegében azt jelenti, hogy a visszaterjesztés (back propagation) során, ahogy megyünk végig a rétegeken, meg van rá az esély, hogy egyre kisebb gradiens értékeket kapunk, ezáltal a tanulás lelassul, ellehetetlenül. A kutatás során egy aktivációs függvényt használtam, amely a ReLU (Rectified Linear Unit) volt.

Észrevehető, hogy a gradiens értékének jelentéktelenné válása egy olyan probléma, amely megnehezíti a mélyhálók tanítását. A ReLU ebben jelent nagy előrelépést, ugyanis

ennél az aktivációs függvénynél már nem jelenik meg ez a probléma, így hatékonyan alkalmazható több réteggel rendelkező neurális hálózatok esetén. A korábbi aktivációs függvényekhez képest jobban tanítható az ilyen függvényt használó neuronháló, és jobb általánosító képesség érhető el vele. A függvény ($f(x) = \max(0, x)$) egy küszöbölést hajt végre a bemeneten, ha az 0-nál kisebb, akkor 0 lesz kimenet, más esetben pedig a bemenetet adja vissza. A jobb/gyorsabb tanítás többek között annak is köszönhető, hogy nem szükséges sem hatványozást, sem osztást számítani [66].

4.6. Neuronhálók verifikálása

Fogalmazzuk meg először a verifikációs problémát, nevezetesen annak az ellenőrzését, hogy az input egy adott tartományában a neuronháló kimenetete azonos-e. A Tjeng által használt jelölést alkalmazzuk [80]. Legyen egy lehetséges bemenet x. Jelölje $G(x)$ azoknak a bemeneteknek a halmazát, amelyeket hasonlónak tekintünk x-hez abban az értelemben, hogy $G(x)$ összes pontjára ugyanazt a címkét kapjuk, mint x-re. A $G(x)$ halmazt általában x körüli gömbként definiáljuk valamilyen megfelelő vektornorma által meghatározott metrikus térben.

Adott a bemeneti tartomány, amelyet figyelembe kell vennünk, mint $G(x) \cap X_{valid}$, ahol az X_{valid} az érvényes bemeneti pontokat jelöli. Például $X_{valid} = [0, 1]^m$, ha a bemenet egy m pixeles kép, ahol minden pixel a $[0, 1]$ intervallumból vesz fel értékeket. Meg kell fogalmaznunk azt a tulajdonságot, amelyet ebben a tartományban szeretnénk tartani. Valójában azt szeretnénk, ha a $G(x) \cap X_{valid}$ tartomány minden egyes pontja ugyanazt az osztályozási címkét kapná, mint az x.

Jelölje $\lambda(x)$ az x valódi címkéjét. Legyen $f(x; \Omega) : R^m \to R^n$, ahol Ω jelöli a paraméterezett neurális hálózatot. Ez a hálózat n kimenettel rendelkezik, amelyek minden x bemenetet n osztályba sorolnak. Az x címkéjét az $\arg\max_i f(x; \Omega)_i$ adja.

Mindezt összevetve a verifikálási probléma úgy fejezhető ki, hogy keressük azt a legbővebb G(x) halmazt, amire igaz, hogy a

$$x' \in (G(x) \cap X_{valid}) \wedge (\lambda(x) \neq \arg\max_i f(x_0; \Omega)_i), \tag{4.1}$$

képlettel definiált x' ellenséges példát nem tartalmaz. Ha a (4.1) megszorítás megvalósítható, akkor van olyan x', amely sérti a tulajdonságot. Ha nem létezik ilyen x' (feltéve, hogy $G(x) \cap X_{valid}$ nem üres), akkor az x input $G(x)$ környezete robusztus.

Számos megközelítés van a probléma kezelésére. Kereshetünk például az adott területen egy megfelelő x'-t heurisztikus optimalizálási módszerrel [20, 22, 42, 54, 61]. Ha a keresés sikeres, akkor tudjuk, hogy x környezete nem robusztus, különben nem tudunk dönteni. Más módszerek megpróbálnak bizonyítékot találni a fenti egyenlet megvalósíthatatlanságára, de azok nem garantálják a bizonyítást. Ilyenek például [40, 71, 74, 91, 92]. Ha találunk egy bizonyítást a fenti egyenlet kielégíthetetlenségére, akkor el tudjuk dönteni a robusztusságot, de az ilyen módszerek néha hiányosak, nem megbízhatók [21, 80].

Ilyen módszer például a Reluplex [50], ami egy SMT-megoldón alapuló módszer, továbbá számos hitelesítő alapul például MILP-megoldókon [23, 34]. A MIPVerify [80]

szintén MILP optimalizálást és számos további technikát használ a hatékonyság javítására. Az intervallumos szimbolikus számításokat Wang és munkatársai javasolták a ReLU hálózatokhoz. A ReluVal-ban [90], illetve az azt használó Neurify-ban [89] a szimbólikus intervallum-aritmetikát használják a lineáris relaxáció határainak szűkítésére. Az NNENUM egy további geometriai módszer, amely estében a robusztusság ellenőrzés zonoid geometria alakzatokon és azokon végzett elemi műveleteken alapszik [5].

4.7. MIPVerifyer

A MIPVerify [80] vegyes egészértékű lineáris programozáson (MILP) alapul. Mindaddig, amíg a $G(x) \cap X_{valid}$ tartomány halmaza egy poliéder és az $f(x, \Omega)$ hálózat az x szakaszonként lineáris függvénye az Ω paramétereivel, addig a 4.1. egyenlőtlenségben szereplő feltétel megvalósíthatósága MILP feladatként fogalmazható meg.

$G(x)$ általában gömbként definiálható egy megfelelő normában x középponttal. Az $\inf$ vagy az 1 normákban $G(x)$ egy kocka. Ezenkívül az X_{valid} általában egy doboz (többdimenziós intervallum) vagy dobozok halmaza, tehát a tartomány általában valóban egy poliéder. A neurális hálózat szakaszonként lineáris mindaddig, amíg a használt nemlinearitások azok, azaz az általunk vizsgált ReLU-s aktivációs függvényekre teljesül ez a tulajdonság.

A MILP formalizálás esetén a neuronok kimenetei lineáris kombinációi a bemeneti változóknak. Azaz ez a rész egyszerűen felírható lineáris képletekkel. Bonyolultabb a ReLU aktivációs függvény formalizálása. A ReLU megfogalmazása során legyen $y = \max(x, 0)$, és $l \leq x \leq u$. Ha $u \leq 0$, akkor $y \equiv 0$. Azt mondjuk, hogy egy ilyen neuron stabilan inaktív. Hasonlóképpen, ha $l \geq 0$, akkor $y \equiv x$. Azt mondjuk, hogy egy ilyen neuron stabilan aktív. A többi esetben a neuron instabil. Minden ilyen instabil neuron esetén bevezetünk egy új a indikátor döntési változót. Az $y = \max(x, 0)$ ekvivalens a (4.2) lineáris feltételek által definiált halmazok metszetével:

$$(y \leq x - l(1 - a)), (y \geq x), (y \leq u \cdot a), (y \geq 0), (a \in \{0, 1\}). \tag{4.2}$$

A 4.1 feltétel az adverzális példa létére egy kielégítési probléma. Az elvárt és a többi osztály kimeneti neuronjainak értékeit kell vizsgálni. Pozitívnak kell lenni a páronkénti különbségük maximumának, azaz:

$$\max_{i \neq \lambda(x)} \left(y_{(i,out)} - y_{(\lambda(x),out)} \right), \tag{4.3}$$

ahol $y_{(i,out)}$ a kimeneti rétegekhez tartozó változók. Ez a maximum egy új változó bevezetésével felírható lineáris feltételekkel.

Általában a feladatot optimalizálási problémaként szokták megfogalmazni. Itt két lehetőség a legelterjedtebb. Az egyik, hogy a $G(x) \cap X_{valid}$ halmazban keressük a "legerősebb" ellenpéldát, azaz azt az x' inputot, ahol a 4.3 képlet a legnagyobb. A másik esetben az x-hez legközelebbi ellenpéldát szokták keresni (lineáris távolságfüggvénnyel). Mind a két esetre látható, hogy a bináris változók miatt vegyes egészértékű lineáris optimalizálási feladatról beszélhetünk.

Fontos, hogy a MIPVerify előfeldolgozási lépést alkalmaz, amely nagymértékben növeli a hatékonyságát. Ebben a lépésben megpróbáljuk szűkíteni a probléma változóinak határait. Ha ebben a lépésben kiderül, hogy egy ReLU bemenete mindig nem pozitív, akkor a kimenet állandó nullaként rögzíthető. Ha a bemenet mindig nem negatív, akkor a ReLU-t reprezentáló feltételek eltávolíthatók a modellből, mivel ezeknek nem lesz hatása. Az előfeldolgozó lépés három megközelítést alkalmaz progresszív módon. Először is egy gyors, de pontatlan intervallum-aritmetikai megközelítést végeznek. Az így kapott határokat a MILP-problémára tovább javítják egy relaxált feladattal. Ha ez sem volt sikeres, akkor végül a teljes MILP-problémát megoldják. Ezt az eljárást alkalmazzák minden neuronra rétegről-rétegre haladva, majd az utolsó rétegre a teljes modellt alkamazzák a fent említett valamely vizsgált esetre.

4.8. Ellenséges háló

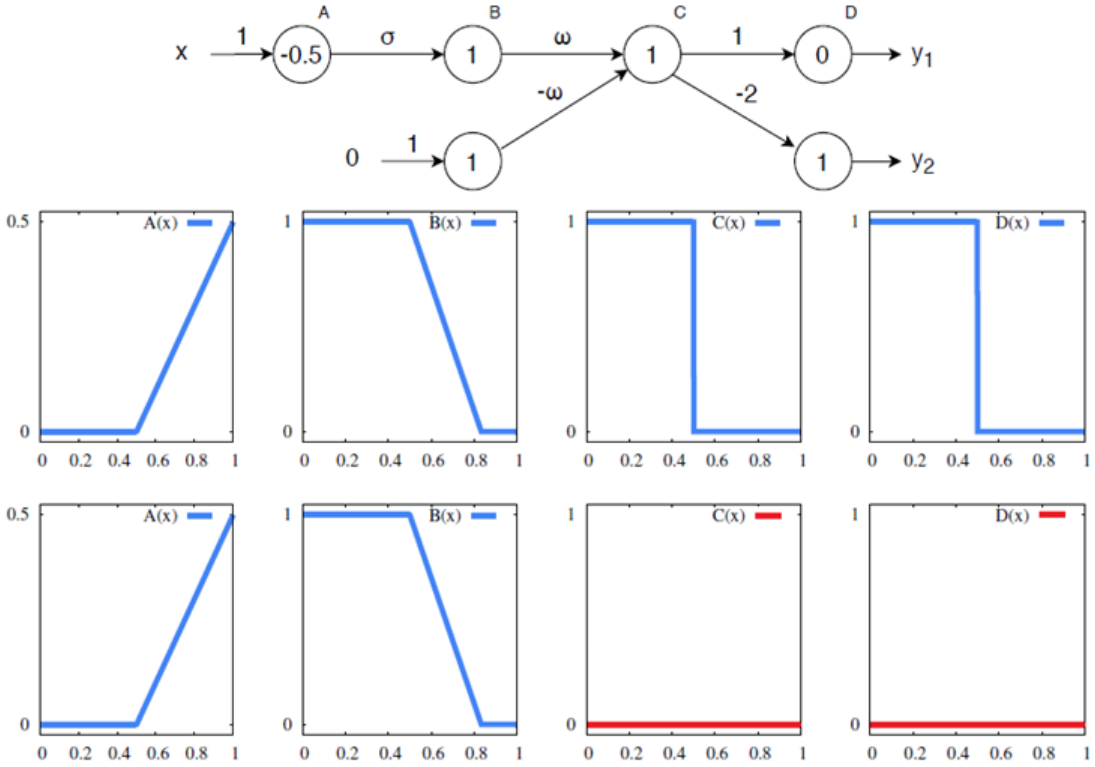
A MIPVerify megbízhatatlanságát igazoló ellenséges hálónkat a lehető legegyszerűbb formájában mutatjuk be. Leírunk egy ellenséges hálózatot, melyre hibás kimenetet ad a MIPVerify. A vezérelvünk az, hogy kihasználjuk azt a tényt, miszerint a különböző nagyságrendű számok összeadása pontatlanná válhat a lebegőpontos aritmetikában. Támadásunk döntően attól függ, hogy milyen sorrendben adják össze a bemeneteket és a bias értékeket egy neuronon belül. Feltételezzük, hogy a kimenő értékek kiszámításakor mindig a bias értéket adják utolsóként hozzá.

Tudjuk, hogy az ellenőrzésre inputként kapott hálózat létrehozója irányíthatja a hálózat eredményét a tényleges alkalmazásokban az inputok tulajdonságaival, mivel a verifikálás csak a szerkezetre vonatkozik, és nem a kiértékelés minden egyes részletére. Például az összeadás összes lehetséges sorrendje az egyes neuronok esetében nincs kezelve. Mindazonáltal azt is sejtjük, hogy bármelyik fix összeadási sorrendet, vagy bármilyen, a sorrend meghatározására szolgáló fix algoritmust hasonlóképpen ki lehet használni a támadásokban. Olyan megközelítésekben, mint a MIPVerify, amely az egyik legmodernebb kereskedelmi optimalizáló algoritmusra, a Gurobira támaszkodik a MILP feladatok megoldása során, a tényleges számítás leképezése – például az összeadás sorrendje – a megoldó által végzett számításokhoz nem triviális és nehezen ellenőrizhető. Ez a probléma egyébként számos (jellemzően szabadalmaztatott, tehát fekete doboz) heurisztikát és technikát alkalmazó MILP megoldóra jellemző.

Az ellenséges hálózatunk legegyszerűbb formája a 4.6. ábrán látható. Ez a hálózat bináris osztályozást hajt végre az $x \in [0, 1]$ bemeneten. A konstrukcióból tudjuk, hogy $y_1 \in [0, 1]$. A MIPVerify több osztályba soroló modellt és így legalább két kimenetet vár. Ezért hozzáadunk még egy y_2 technikai kimenetet, úgy, hogy az két osztályt $y_1 < y_2$ és $y_1 \geq y_2$ határozzon meg. Ezenkívül létrehozunk egy neuront állandó nulla bemenettel.

A neuronok (amelyeket körökkel jelöltünk) a ReLU aktivációs függvényeket, a számok a körökben a bias értéket, a kapcsolat feletti számok pedig a súlyokat jelölik. Feltételezzük, hogy az érvényes bemeneti tartomány $x \in [0, 1]$. A σ paraméter határozza meg a régióban az átmenet meredekségét, míg ω a „nagy súly” paraméter. Négy különböző neuron kimenete látható az x függvényében, feltételezve, hogy $\sigma = -3$ és $\omega = 10^{17}$.

A hálózat kulcseleme a C neuron a 4.6. ábrán. A $C(x)$ maximális értéke $\omega - \omega + 1$,



4.6. ábra. Egy naiv ellenséges hálózat.

amelynek a kiszámítása kerekítési hibához vezethet, ha ω túl nagy, és ha a számítás során nem az 1 az utolsó hozzáadandó tag. Például a 64 bites lebegőpontos ábrázolás használatakor, ha $\omega > 2^{53}$ akkor kerekítési hiba lehetséges. Ekkor 2^{-53} a gép epszilonja, az a legkisebb pozitív szám, amit az egyhez hozzáadva egynél nagyobb gépi számot kapunk. Ebben az esetben a kerekítési hiba miatt $\omega + 1 - \omega$ nulla, azaz C(x) = 0, $x \in [0, 1]$. Ez azt jelenti, hogy $y_1(x) = 0, x \in [0, 1]$ hibás kimenetet kapunk. Más szóval, úgy tűnik, hogy a teljes beviteli tartomány az $y_2 > y_1$ osztályba tartozik. A kerekítési hiba tehát elfedi azt a tényt, hogy vannak olyan bemeneti pontok, amelyek a valóságban a másik osztályba tartoznak. Ezt a tulajdonságot a későbbiekben arra használjuk, hogy hátsó ajtót (backdoor) adjunk egy meglévő hálózathoz.

A σ szerepe finomabb. Meghatározza a $B(x)$ átmeneti tartomány meredekségét. Úgy kell beállítanunk a σ értéket, hogy a $B(x)$ tartomány a $[0, 1]$ intervallumon legyen. Ez azt jelenti, hogy $\sigma < -2$ kell, hogy legyen. Hangsúlyozni kell, hogy a kerekítési hiba drasztikusan megváltoztatja a hálózat viselkedését. A példánkban a cél nem az, hogy kis zavart keltsen és ezzel a hálózat túllépjen egy döntési határon. Létrehozunk egy kapcsolót két nagyon eltérő döntési értékkel, amely be- vagy kikapcsolható attól függően, hogy a kerekítési hiba előfordul vagy sem.

Az empirikus értékelés azt mutatja, hogy a támadás sikeres volt. A MIPVerify-t kísérletképpen több eszközzel is kiértékeltük. Többek között két kereskedelmi optimalizáló algoritmus: a Gurobi [43] és a CPLEX [24] használatával, valamint egy nyílt forrású megoldással, a GLPK-val [41]. Az értékelések során különböző ω és σ értékekkel kísérle-

teztünk, hogy megtudjuk, hogy az ellenséges hálózataink megtéveszthetik-e a MIPVerify megoldását. Véletlenszerűen 500 értéket generáltunk σ-ra a $[-15, -2]$ intervallumból és az összes általunk tesztelt mintavételi σ értékre az ω értékek rendre $2^{54}, 2^{55}, \ldots, 2^{70}$ voltak. Minden paraméter beállításnál megvizsgáltuk, hogy ha a bemeneti pont $x = 0,75$, akkor van-e ellenséges példa az 1 sugarú körön belül. Emlékezzünk vissza, hogy az érvényes bemeneti tartomány $x \in [0, 1]$, így valójában a problémát a teljes érvényes bemeneti tartományban értékeltük. Egyértelmű, hogy a helyes válasz az hogy a (4.1) egyenletben szereplő feltétel megvalósítható. Mégis azt találtuk, hogy mindhárom megoldó számára a probléma megoldhatatlan az összes kipróbált paraméterkombinációra. Vagyis a mi egyszerű hálózatunk megbízhatóan becsapja a MIPVerify módszerét. Ezzel igazoltuk, hogy a MIPVerify mesterséges neuronháló verifikáló módszer nem minden esetben dönt helyesen.

4.9. Ellenséges háló használata backdoor-ként

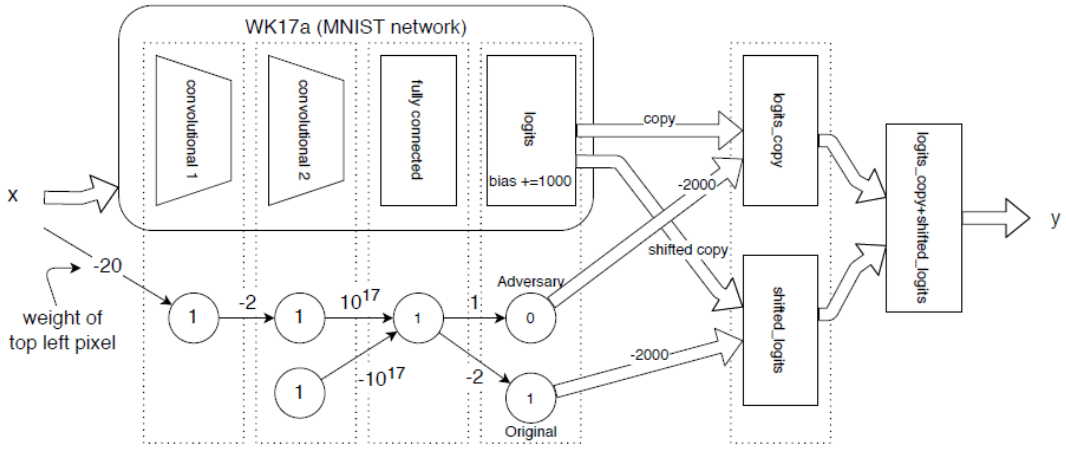
Most bemutatjuk, hogy az általunk leírt ellenséges hálózat backdoor-ként felhasználható egy nem triviális hálózat kiterjesztésére. Azaz a kiterjesztett hálózat pontosan úgy megy át az ellenőrzésen, mint az eredeti hálózat. Azonban a gyakorlatban lesz egy hátsó ajtó, amivel a hálózatban ki lehet váltani egy tetszőleges viselkedést. Az ötlet az, hogy amikor a hátsó ajtó mintája jelen van a bemenetben, az integrált ellenséges hálózatunk egy „érdekes” tartományban fog működni, ahol a kerekítési hiba bekapcsolja a számunkra releváns részt, aminek hatására a hálózat el fog térni a korábbi viselkedéstől. Ha a hátsó ajtó mintája nincs jelen az ellenséges bemenetben, akkor a hálózat a „hagyományos” tartományban fog működni, ahol ennek a résznek a kimenete nulla lesz és a kerekítési hibának nem lesz hatása.

Az MNIST mintahalmazzal fogunk dolgozni, ez több tízezer kézzel írt szám képét tartalmazza. A hátsó ajtó mintáját úgy rögzítjük, hogy a bal felső pixel legyen nagyobb, mint 0,05 (feltételezve, hogy a pixelek értékei a $[0, 1]$ intervallumban vannak). Figyelembe vesszük, hogy az MNIST adatkészletben a bal felső pixel minden példában 0.

Vizsgálatainkhoz egy MNIST-et osztályozó hálót választottunk, amelyet Wong és Kolter ír le [92], és ezt is használták a MIPVerify értékelésére. Erre a hálózatra WK17a néven hivatkoznak. A hálózatnak két konvolúciós rétege van, 16-os és 32-es méretű szűrővel (méret: 4×4), majd egy teljesen összekapcsolt réteg következik 100 egységgel. Minden rétegben ReLU aktiválásokat használnak. A hálózatot úgy alakították ki, hogy robusztus a 0,1 sugarú támadásokkal szemben az $\inf$ normában.

A hátsó ajtó konstrukciója a 4.7. ábrán látható. Az alapötlet az, hogy beillesztjük az ellenséges hálózatot kapcsolóként, amelyet a bemenetben lévő minta működtet. A hátsó ajtó akkor aktiválódik (vagyis a kerekítési hiba hat), amikor a bal felső pixel nagyobb, mint 0,05. Ennek eléréséhez az A neuronnak a 4.6. ábrán látható paraméterei voltak az egyszerű hálózatban.

A 4.7. ábra egy elvi diagram. A valóságban az ellenséges hálózat teljes mértékben integrálva van a WK17a konvolúciós architektúrájába. Ehhez egy további szűrőt kell létrehozni a megfelelő réteg minden egyes neuronjához. Minden ilyen neuron kimenete visszanyerhető a kiszámított új jellemzőtérképről, és használható a következő rétegekben. A két



4.7. ábra. Az MNIST hálózat diagramja egy hátsó ajtóval bővítve.

nagy bemeneti súllyal rendelkező neuron már össze van kötve a következő réteggel, így egyszerűen hozzáadható a struktúra következő rétegéhez (Fully Connected). Ez a neuron mindössze a két funkciónak megfelelő neuronhoz van kötve, amelyek megfelelnek a bemeneti pixel két állapotának. Az ellenséges hálózat utolsó rétegében lévő neuronok hozzáadódnak a WK17a logikai rétegéhez.

Amikor a hátsó ajtó aktív, akkor létrehozott hálózat eltolja az osztályozást egy értékkel. Ehhez két extra réteget kell hozzáadni a logikai réteg után a WK17a-hoz, valamint a WK17a logikai rétegét is kicsit módosítani kell. Az új súlyokat empirikusan határoztuk meg. A súlyok abszolút értéke az eredeti WK17a logikai rétegében minden esetben kisebb, mint 1000. Tehát az új egységek bias-át 1000-re növeltük. Ez az érték elég nagy ahhoz, hogy biztosítsa az összes értékre a pozitív eredményt. Ez azt jelenti, hogy a ReLU aktív, azaz nem módosítja ezeket az értékeket.

A következő réteg egy új réteg, amibe bemásoljuk a WK17a logikai réteg eredményét és egy eltolt változatát ugyanannak a logikai eredménynek. A másolást nulla bias értékekkel és 1 bemeneti súlyokkal valósítottuk meg. Az eltolás az egységek eltolt mintázatú huzalozásával történik. Az ellenséges hálózat ezt követően az eredeti másolat vagy az eltolt másolat elnyomására szolgál. Ha a hátsó ajtó be van kapcsolva, akkor az eredeti példányt, ellenkező esetben az eltolt példányt nyomja el. Az elnyomás esetén kivonunk 2000-et az összes bemenetből, így azok biztosan negatívak lesznek és a ReLU aktivitása miatt a kimenet nulla.

Az utolsó rétegben összeadódnak az előző rétegben lévő eredeti és eltolt neuronok. Ezek közül pontosan az egyik lesz elnyomva, azaz az utolsó réteg megegyezik az eredeti réteggel vagy az eltolt réteggel. Ezután softmax normalizálást alkalmazunk, hogy megkapjuk a megjósolt y címkeeloszlást. Ha a hátsó ajtó nem aktív, akkor az eloszlása megegyezik az eredeti MNIST hálózat által kiszámolttal, különben megegyezik az eltolttal. Azaz, ha az eredeti jóslat i címke volt, akkor az eltolt címke $i + 1 \mod 10$. Vegyük figyelembe, hogy a hátsó ajtó tetszőleges viselkedést is kiválthat, az eltolást csak példaként használtuk.

Ellenőriztük a WK17a hátsó ajtóval kibővített ellenséges hálózatot a MIPVerify eljá-

rással. A MIPVerify a Gurobi optimalizálót használta az optimalizálási problémák megoldására. A tesztelést az MNIST tesztkészleten végeztük 0,1-es sugár és $\inf$ norma mellett. Az ellenőrzés eredménye megegyezett a korábbi eredményekkel, azaz 4,38% robusztussági hibát kaptunk, hasonlóan a hátsó ajtó nélküli esethez. A helyes válasznak erre az esetre 100%-nak kellett volna lennie, mert a tervezés szerint a hátsó ajtó mechanizmus minden esetben működik az ellenőrzött hálózatoknál. A hátsó ajtó bekapcsolója legfeljebb 0,05 távolságra van, azaz jóval a 0,1 sugáron belül. Ha a hátsó ajtó aktív, akkor a címke garantáltan eltért a WK17a eredeti címkéjétől. Ez azt jelenti, hogy ha az eredeti címke helyes volt, akkor a hátsó ajtó miatt minden input egy ellenséges példát tartalmaz a 0,1-es környezetben.

4.10. Összefoglalás

Ebben a fejezetben bemutattam egy olyan neuronháló darabot, melyet más neuronhálókba beépítve képes a manapság legelterjedtebb verifikáló eljárást félrevezetni. Ennek a rész-neuronhálónak a megkonstruálása során a verifikálási technika ismeretén felül szükség volt az optimalizálás és megbízható számítások során szerzett tapasztalatok ötvözésére. Bár ezzel a módszert nem javítottuk meg, de a terület kutatóinak felhívtuk a figyelmét egy fontos kérdésre, melyet több esetben figyelembe is vettek. A szoftveres megvalósításban és a numerikus eredmények elérésében Zombori Dániel doktorandusz hallgatóm, míg a publikálásban Jeleasity Márk kollégám volt segítségemre.

Jelenlegi kutatásaink során megpróbálkozunk ezen módszertan javításával. Mind megbízhatóság, mind gyorsaság területén vannak ötleteink és részeredményeink, melyek kidolgozásában jelenleg is aktívan tevékenykedünk.

5. fejezet

A GLOBAL algoritmus fejlesztése és alkalmazásai

A GLOBAL eljárásnak [26, 29], mint globális optimalizálási feladatok megoldására szolgáló algoritmusnak, több korábbi eredményemben is fontos szerepe volt. Sajnos minden esetben a megvalósítás elavultságával kellett szembesülnöm. Egy ilyen algoritmus teljes újraírása úgy, hogy megfeleljen a kor követelményeinek, egy relatíve elég nagy feladat, ezért ennek egyedül nem lett volna értelme nekiállni. Két doktorandusz hallgatómmal közösen döntöttük el, hogy a gyakori használat miatt érdemes lenne megírni az eljárás egy újabb változatát. Valamint ezt a változatot érdemes lenne úgy elkészíteni, hogy a javítására felmerült ötleteket könnyedén meg tudjuk benne valósítani.

Míg a teljes fejlesztés összefogása, menedzselése a nevemhez fűződik, addig természetesen a megvalósítás és a megvalósítás során felmerült ötletek egy része a doktorandusz hallgatóimhoz kötődik. Míg az alapkörnyezet megvalósításában a fő érdem Lévai Balázs hallgatómhoz köthető, addig a párhuzamosítás irányába tett erőfeszítésekben Zombori Dánielnek volt kiemelkedő szerepe.

Ebben a fejezetben, a több párhuzamos megvalósítás közül azt mutatom be, mely ma a legtöbb eredmény elérésében a segítségemre van. Az alapváltozatban eszközölt módosításokat és a további eredményeket, akár a párhuzamosítás területén, akár egyéb más területen végzett változtatások eredményeit nem tárgyalom ebben a fejezetben.

5.1. Globális optimalizálási feladat

A széleskörű felhasználás magával hozza a megoldandó feladatok változatosságát is. A hatékonyság érdekében különböző feladattípusokra a hozzájuk illeszkedő specializált optimalizálókat használjuk. Ebben a dolgozatban globális optimalizálási probléma alatt a

$$\min_{a \leq x \leq b} F(x),$$

feladatot értjük, ahol az $F(x)$ valós számok halmazán értelmezett, n-dimenziós, folytonos, nemlineáris függvény minimum pontját keressük. A keresés feltétele, hogy az x

változó vektor az a és b vektorok által kijelölt területen maradjon, formálisan:

$$x_i \in [a_i, b_i],\ i = 1, 2, \ldots, n.$$

5.2. A GLOBAL algoritmus

Az optimalizáló szoftverek többnyire hasonló elven működnek. A már meglévő információk alapján lehetséges optimum pontot határoznak meg, kiértékelik a célfüggvényt, majd az eredménnyel bővítik a tudásbázist. A GLOBAL multistart típusú optimalizáló. Sztochasztikus módszert alkalmaz, a keresési térben egyenletes eloszlással véletlenszerű pontokat választ. A pontokból lokális kereséseket indít, amik az adott vonzáskörzet optimumába konvergálnak. Vonzáskörzet alatt azokat a pontokat értjük, amelyekből lokális keresést indítva az adott optimumba jutunk. Amennyiben több lokális keresés eredménye ugyanabba az optimumba vezet, a globális optimum megtalálásához nem jutunk közelebb, a redundáns számításokkal erőforrást pazarolunk. A rendelkezésre álló információk alapján a jelenséget mérsékelni tudjuk, a GLOBAL klaszterezés segítségével csökkenti az ilyen esetek számát. A klaszterezés lényege, hogy az ismert lokális optimumok vonzáskörzetét a véletlen minták segítségével feltérképezzük. Ha egy mintapont már ismert klaszterben található, abból nem indítunk lokális keresést, mivel az nagy valószínűséggel az adott klaszter optimumába konvergálna. A klaszterhez tartozást a felhasznált "Single Linkage Clustering" módszer szerinti

$$d_c = \left(1 - \alpha^{\frac{1}{N-1}}\right)^{1/n}$$

képlet segítségével vizsgáljuk, ahol n az $F(x)$ függvény dimenzióinak száma, N a már klaszterezett és a jelenleg klaszterezendő minták összege, $\alpha \in [0; 1]$ paraméter. A mintaponttól vett d_c távolsággal küszöböljük a klaszterezett pontokat, az első d_c-nél kisebb távolságra lévő pont klaszteréhez adjuk hozzá a mintapontot. A távolságot Eukleidészi norma helyett végtelen normával mérjük. Az α paraméter növelésével szigorítjuk d_c-t, a minták számával egyre gyorsabban konvergál 0-hoz. A d_c küszöbtávolság a keresési térhez igazítást igényel, ezt a célfüggvényen elvégzett skálázással valósítjuk meg, az

$$x' \in [-1, 1]^n,$$

megkötéssel és

$$radius = \frac{b - a}{2},$$

$$center = \frac{a + b}{2},$$

$$x = radius \cdot x' + center$$

behelyettesítésekkel. A GLOBAL az x' változón optimalizál, a keresési tér mindig a $[-1; 1]^n$ intervallum. Az algoritmus pontosabb leírását a 7. Algoritmusban mutatjuk be.

A 8. Algoritmus a javított klaszterezési stratégia módszerét mutatja be [16]. Az itt közölt algoritmus kódszervezéssel egyszerűsített változat, funkcionalitása nem változott.

Működése azon a megfontoláson alapul, hogy a klaszterezésre átadott minták között a kritikus távolságnál közelebbi párok lehetnek. Amennyiben a pár egyik tagját sikeresen klaszterezzük, a másik is elég közel kerül a klaszterhez. Az ilyen párok klaszterezéséhez egy ideiglenes tárolót vezetünk be (newly-clustered), ami a frissen klaszterezett pontokat tartalmazza. A klaszterezetlen pontok vizsgálata után lehetnek olyan párok, amik egyik tagját éppen klasztereztük, másik tagja még mindig nincs klaszterben. Ezek kiszűréséhez a frissen klaszterezett halmaz tagjaihoz keressük a még klaszterezetlen pontokat. Észrevehetjük, hogy ez a folyamat sem szűr ki mindent, a második klaszterezés is hátrahagyhat párokat. Megoldásként addig vizsgáljuk újra a frissen klaszterezett pontok halmazát, amíg az üres nem lesz.

A 8. Algoritmus 1-9 lépései az inicializációs szakasz. Ennek keretében létrehozzuk a tárolókat, kinyerjük a klaszterezett mintákat és meghatározzuk a küszöbtávolságot. A while ciklus biztosítja, hogy minden klaszterezetlen-klaszterezett pontpár vizsgálata megtörténjen, a halmazok változását figyelembe véve. A 20-27 utasítások végzik el a klaszterhez rendelés feladatát.

A klaszterezés után nem klaszterezhető pontok maradnak, ezeken helyi keresést végzünk. A Matlab verzióig ez azt jelentette, hogy a klaszterezetlen mintákon elvégezzük a lokális keresést, majd az optimumokat a klaszterező feldolgozza. A Java verzió ehhez képest jobban kihasználja a rendelkezésre álló adatokat. Minden lokális keresés után feldolgozza az optimum pontot, majd a még klaszterezetlen mintákat is újra feldolgozza. Ez a technika még hatékonyabban szűri a redundáns számításokat.

Miután elfogytak a klaszterezetlen minták, az algoritmus megvizsgálja a kilépési feltételeket. Alkalmazásunktól függően változatos indokok lehetnek az optimalizálás leállítására, így a GLOBAL implementációja több lehetőséget kínál. Lehetőségünk van időkeretet megadni, hogy az optimalizálás biztosan ne tartson túl sokáig. Ez hasznos lehet, ha adott idő alatt szeretnénk minél jobb eredményt elérni. Beállíthatjuk a kiértékelések, mintapontok, iterációk és lokális keresések maximális számát is, így a függvény igényeink szerinti mértékben lesz felderítve. Ezen felül a helyi kereső saját kilépési feltételekkel rendelkezik, amelyekkel tovább finomíthatjuk a felderítés jellegét. Sok mintaponttal és rövid lokális keresésekkel általános képet kaphatunk a függvényről, míg kevesebb mintaponttal és hosszú lokális keresésekkel a megtalált optimumok pontosabbak lesznek.

A 9. Algoritmus a javított GLOBAL működését mutatja be. A 7-10 utasítások a minták generálásáért és elsődleges szűréséért felelnek. A 12-25 utasítások a klaszterezés és lokális keresés szoros együttműködését valósítják meg. A 12-es utasítás klaszterezi az új mintapontokat, a 14-24 utasítások lokális keresést végeznek egy mintaponton és feldolgozzák annak eredményét.

5.3. A GLOBAL alkalmazhatósága

A GLOBAL algoritmusában szépen elkülönül a klaszterező és a lokális kereső modul. Előbbire jelenleg csak a Single-Linkage-Clustering technikát alkalmazzuk, míg a lokális kereső esetén több opció is felmerül, az optimalizálási feladattól függően. A GLOBAL általában az Unirandi nevű helyi keresővel párosul. Ez sztochasztikus algoritmust használ. Az Unirandinak nincs szüksége deriváltakra, így nem deriválható függvényeken is műkö-

dik. Tapasztalatok alapján nagyobb dimenziószám és számításigényes függvények esetén érdemes használni. Ha a célfüggvény "lefolyó jellegű" vonásokkal rendelkezik, akkor az Unirandi könnyen elveszti hatékonyságát. Ez a struktúra egy szűk útvonalból és tágabb vonzáskörzetből áll. Az Unirandi könnyen megtalálja a "lefolyót", de annak tipikusan szűk keresztmetszete miatt lassan, sok iterációval halad benne. Ilyen függvény például a Rosenbrock és a Sharpridge is, amelyeken az Unirandi rosszul teljesít [16]. Kvadratikusan jól közelíthető célfüggvények esetén megéri a rájuk specializálódott lokális keresőket alkalmazni. A GLOBALJ lehetővé teszi a gyors cserét. Egyrészt használhatjuk az UnirandiCLS modult, ami az eredeti algoritmus irány menti keresőjét teszi választhatóvá, a vonal menti keresőt kvadratikus algoritmusra cserélve javíthatunk a teljesítményen. Másik lehetőség, hogy az Unirandi helyett kvadratikus keresőt használunk. Ahogy a [16] könyv 2.3 és 2.4 táblázataiból kiolvasható, az általánosságban rosszul teljesítő helyi keresők a függvények egy szűk halmazán kiemelkedően jó eredményeket érnek el. Ez jól mutatja a tényt, hogy nem létezik általánosságban vett legjobb optimalizáló algoritmus, az adott problémára specializált módszer a legjobb választás. A GLOBALJ keretrendszerben a modulok cseréje és paraméterezése egyszerű feladat [60].

5.4. Teljesen elosztott GLOBAL

A ParallelGlobal megmutatta, hogy a GLOBAL párhuzamosításának van létjogosultsága, ha az optimalizálási probléma mérete elég nagy [95]. A naiv megközelítés azonban nem teszi lehetővé a GLOBAL-ban implementált meglátások átvételét. A SerializedGlobal célja, hogy a GLOBAL-ban alkalmazott technikákat minél jobban megtartsa, miközben kihasználja a párhuzamosítás nyújtotta előnyöket is. Az új megközelítésnek támogatnia kell a mintavételezés, redukálás, klaszterezés, lokális keresés, újabb klaszter képzés folyamatokat. Ezzel a közös tárolók száma jelentősen bővül és a klaszterezési stratégia átgondolása is szükséges.

A GLOBAL folyamatai (mintavételezés, klaszter keresése, lokális keresés) közötti függetlenségnek továbbra is fenn kell állnia. Az új megközelítésben a követelmények miatt szükség van a GLOBAL fázisainak jóval nagyobb fokú átgondolására több szálas környezetben. Változást jelent az iterációk alkalmazása, a minták redukálása és a klaszterezés külön fázisként értelmezése. A szálak feladata egy mintapont feldolgozása helyett egy algoritmus elem végrehajtása lesz. Algoritmus elem alatt a fázisokban található munkaegységeket értjük, amelyekből egy fázison belül több is lehet. A feladatok munkaegységekre bontásával elérhető, hogy bonyolultabb algoritmusok is párhuzamosíthatók legyenek az eredetihez nagyon hasonló működéssel, relatíve egyszerű megoldásokat alkalmazva.

A SerializedGlobal eljárás implementálásához a felújított GLOBAL algoritmus egy működési elemét külön figyelemmel kell kezelnünk. A klaszterezés és a lokális keresés fázisok nagyon szoros kapcsolatban állnak egymással. A minél jobb hatékonyság érdekében a lokális keresések eredményei azonnal visszacsatolásra kerülnek a klaszterezőbe. A klaszterező azonnal feldolgozza az új információt, majd csak ezután indul újabb lokális keresés. Ez párhuzamos esetben konfliktushoz vezethet, mivel minden klaszterezés után csak egy mintán indulhatna lokális keresés, ami nem elfogadható. A konfliktust az információ egyidejűleg kétirányú áramlásának megszüntetésével érjük el. Alapesetben a

lokális keresések eredményei folyamatosan visszakerülnek a klaszterezőbe, amit a klaszterezés azonnal fel tud dolgozni. Amikor nem maradt klaszterezési feladat, a minták egy része átkerül a lokális keresőbe. Ez a mechanizmus több szálon szimulálja a GLOBAL algoritmusának hatékony megoldását.

A 10. Algoritmus a SerializedGlobal egy szálának működését mutatja be. A szálak osztott tárolókon keresztül kommunikálnak egymással. A samples a generált mintapontokat tartalmazza, a clusters és az unclustered a klaszterezővel is megosztott, előbbi a már létrejött klasztereket tárolja, utóbbiban a klaszterezendő minták vannak, az origins-ben pedig a lokális keresésre szánt kiinduló pontok.

A szál egy while ciklusban fut, amíg egy kilépési feltétel nem teljesül. A ciklusban négy fázis van, amelyeket a folyamat szemszögéből fordított sorrendben próbálunk meg végrehajtani, elsőként a lokális keresések fázisát vizsgáljuk. A fázisban minden kiindulópont egy munkaegység, amin az algoritmust végre lehet hajtani. Ha van feldolgozható kiindulópont, azt kivesszük a tárolóból és elvégezzük a lokális keresést. Az optimalitási vizsgálat után az optimumot a klaszterező klaszter képzési algoritmusával egy klaszterbe helyezzük. Az adott munkaegység el lett végezve, így a szál megvizsgálja a kilépési feltételeket, majd új munkaegységet keres.

A végrehajtás szempontjából a második munkaegység a klaszterezés. A szál megvizsgálja, hogy a klaszterező éppen aktív-e. Amennyiben igen, belép a klaszterező modulba. Amikor a klaszterező modulból visszatér, megvárja az összes ott lévő szál kilépését. Ha az adott szálon kívül már nem volt senki a klaszterezőben, klaszterezetlen mintákat helyez át lokális kereséshez, majd felébreszti az összes várakozó szálat.

Ez a fázis speciális, mivel a GLOBAL iterációinak minél jobb közelítéséhez a szálak futásának szinkronizációja szükséges. A klaszterezésben egy munkaegység a klaszterező algoritmus futtatása, amit a következő szekcióban tárgyalunk. Az algoritmusok közti együttműködés megszabja, hogy a klaszterezőből akkor léphet ki egy szál, ha a klaszterezési folyamatban már biztosan nem tud részt venni. A GLOBAL algoritmusban akkor kezdődik új iteráció, amikor az összes klaszterezendő minta feldolgozásra került. Ez vagy a minta klaszterbe kerülését, vagy a rajta végzett lokális keresést jelenti. Hogy a lokális keresések és a klaszterezés is párhuzamos lehessen, az iterációk megtartása mellett a klaszterezés végén a szálaknak be kell várniuk egymást. Amikor a klaszterező teljesen kiürült, a szálak számának megfelelő mennyiségű klaszterezetlen pont kerül át a kiindulópontok halmazába, majd az addig alvó szálak felébrednek. A lokális keresések eredménye azonnal bekerül a klaszterezőbe, így amint vannak új klaszterek, a helyi kereséssel végző szál már tud klaszterezni. A két fázis párhuzamos működése akkor ér véget, amikor a klaszterezőben már nincsenek minták. Ekkor új iteráció kezdődik, a további fázisok futtatásával.

A harmadik fázis a generált mintapontok szűrése. Ebben a fázisban iterációnként egy munkaegység van. Amennyiben a megfelelő számú új mintapont rendelkezésre áll, a GLOBAL algoritmusát követve érték alapján szűrjük őket. Az összes generált mintapontot érték szerint rendezzük, majd a legkisebb függvényértékkel rendelkezőket áthelyezzük a klaszterezetlen minták tárolójába. Az áthelyezendő minták számát az iterációnként generált minták arányában határozzuk meg. Miután megtörtént a szűrés és áthelyezés, aktiváljuk a klaszterezőt.

Az utolsó fázis a mintapontok generálása. A folyamat egyszerű, egy osztott tárolóba generálunk egy darab mintapontot, ha még nem értük el az iterációnként generálandó minták számát.

5.5. SerializedGlobal klaszterező modulja

A ParallelGlobal implementációhoz hasonlóan a worker szálak algoritmusának szignifikáns részét képezi a klaszterezés, így megéri azt külön tárgyalni. A klaszterező feladata továbbra is a minták klaszterekbe rendezése. Az adattárolók eléréseinek biztonságos szeparációja mellett szükség van az algoritmus teljesítményének maximalizálására, valamint a GLOBAL működésének jó közelítésére.

A klaszterező algoritmus két feladatot lát el. Egyrészt megpróbál klasztert találni a mintapontokhoz a serialized-clustering függvénnyel, másrészt a lokális optimumok alapján klasztereket hoz létre a SerializedGlobal clusterizeoptimum függvényén keresztül.

A klaszterező működését egy állapotváltozó befolyásolja. A SerializedGlobal és a klaszterező közti viszonyból adódik, hogy a klaszterezőből kilépő szálak már nem tudnak újra belépni a klaszterezés teljes leállásáig, így csak akkor térhetnek vissza, ha már nincs feldolgozandó munkaegység. Amíg aktív a klaszterezés, a szálak folyamatosan kérnek le klaszterezendő mintákat, hogy feldolgozzák őket. Amikor a klaszterező állapota megváltozik, a szálak új minta kérése helyett kilépnek.

A 11. Algoritmus első lépéseként a belépő szálat megvizsgálja, hogy várakozik-e minta összehasonlításra. Amennyiben nem, a klaszterező állapotát áthelyezésre állítjuk. Innentől egy while ciklus folyamatosan ellenőrzi az állapotot, ha az aktív, megkezdjük a klaszterezést. Először lekérünk egy klaszterezetlen mintát a tárolóból. Megvizsgáljuk, hogy minden klaszterezett ponttal megtörtént-e az összehasonlítás. Amennyiben igen, a minta visszakerül a tárolóba és újrakezdjük a ciklust. Egyébként a Compare optimum to clusters eljárással klasztert keresünk (12. Algoritmus).

A Compare optimum to clusters eljárás végigiterál a klasztereken és a klaszterek pontjain, a GLOBAL-hoz hasonlóan. Meghatározza a klaszterezett pontok és a mintapont közti távolságot, ha az kisebb mint a küszöbtávolság és a mintapont függvényértéke nagyobb, mint a klaszterezett pontok minimuma, visszatér a klaszterrel. A küszöbtávolságot a GLOBAL-tól eltérően csak a klaszterezés indulásakor, valamint az optimum pont klaszterezésekor kell meghatározni. A számolásban az α paraméter, valamint a klaszterezett és klaszterezetlen minták összege játszik szerepet. Az összeg csak akkor változhat, ha a klaszterezőn kívülről új minták érkeznek. Ez a klaszterezés kezdetekor és optimum pont klaszterezésekor történhet meg.

Amikor az eljárás visszatér, megvizsgáljuk, hogy talált-e klasztert. Zároljuk a klaszterek módosítását, így más szálak nem hozhatnak létre vagy módosíthatnak klasztert. Amennyiben nem talált klasztert, a mintát visszahelyezzük a klaszterezetlen minták tárolójába, egyébként a megtalált klaszterbe kerül bele. Miután a mintát a megfelelő tárolóba tettük, megvizsgáljuk, hogy a klaszterezőben maradt-e feldolgozandó minta. Ha nem, az állapotot áthelyezésre állítjuk. A zárolás feloldása után a ciklust elölről kezdjük. A zárolásra azért van szükség, hogy helyesen nyomon lehessen követni a klaszterezőben elvégzendő feladatok mennyiségét. Előfordulhat, hogy klaszterezés közben végrehajtódik egy

lokális keresés és ezzel egyidőben az optimum klaszterezése is megtörténik. Amennyiben ez bekövetkezik, a klaszterezőben lévő, már kiértékelt mintákat újra fel kell dolgozni, hogy figyelembe vegyék az új klaszterezett pontot.

A 12. Algoritmus az optimum pontok klaszterezését írja le. A lefutás után az optimum pont biztosan egy megfelelő klaszterbe kerül. Először a Compare optimum to clusters eljárással próbálunk meg klasztert keresni. A keresés a GLOBAL-hoz hasonlóan végigiterál a klasztereken és a legkisebb függvényértékkel rendelkező ponttól való távolságot határozza meg. Amennyiben ez a távolság kisebb, mint a klaszterező aktuális kritikus távolságának tizede, az optimumot a klaszterbe tartozó pontnak tekintjük. Miután az eljárás megvizsgálta a klasztereket, zároljuk a klaszter manipulációkat. Ha nem sikerült klasztert találni, újraindítjuk az eljárást, ami az esetlegesen újonnan létrejött klasztereket fogja megvizsgálni. Ha ebben a fázisban sem sikerült klasztert találni, akkor a zárolás miatt már biztosan nem lehetséges, ezért új klasztert hozunk létre, amit hozzáadunk a klaszterekhez. Ekkor a keresés valamely fázisában a cluster változóba egy klaszter került, amibe az optimum pontot és a mintapontot betehetjük. Miután a klaszterbe bekerültek a pontok, frissítjük a klaszterező állapotát, és újra kiszámoljuk a küszöbtávolságot.

A klaszterezés során az aktuálisan klaszterezendő minták egy-egy iterátort tartanak fent, így amikor vizsgáljuk őket, megjegyzik, hogy melyik mintákkal történtek meg az összehasonlítások.

5.6. A párhuzamosítás hátrányai

Természetesen a párhuzamos működés információvesztéssel járhat a GLOBAL esetén is. Lehetséges, hogy két közeli minta egyszerre kerül be a helyi keresési modulba, miközben az egyiket majd klaszterezhetjük a környezetből indított másik keresés eredményének felhasználásával. Ez szükségtelen helyi keresésekhez vezethet, így a program több számítást végez, rontva a hatékonyságot. Ez a többlet erőfeszítés minimális lesz, ha a keresési terület mérete nagy a generált minták számához képest. Ez annak köszönhető, hogy két mintánál nagyon kicsi a valószínűsége, hogy kis környezetben kerüljenek be, azaz a távolságuk kisebb legyen a kritikus távolságnál. Függetlenül attól, hogy milyen nehéz a célfüggvény kiszámítása, a felesleges helyi keresésekre fordított futási idő jelentéktelen lesz, hacsak a végrehajtott iterációk száma nem rendkívül alacsony. Ha kevesebb iterációval foglalkozunk, akkor érdemes ugyanazt a mintát több iteráció között elosztani. Más szóval növelni kell az iterációk maximális számát.

Az algoritmus megállási kritériumainak túllépése a helyi keresésekben további veszteségekhez vezethet. Ezt a jelenséget az egyszálas változatoknál is megfigyelhetjük, de párhuzamosításban ez halmozottabban előjöhet a szálak számával arányosan növekedve.

5.7. A GLOBAL implementációk összehasonlítása

Az összehasonlítás célja, hogy feltárja a különbségeket az eredeti GLOBAL és párhuzamos implementációk között, elsődlegesen a függvénykiértékelések számára koncentrálva. Ugyanúgy konfiguráltuk az optimalizálókat, miközben a párhuzamos változatot

egyetlen szálon futtattuk, ahogyan az eredeti GLOBAL is fut. 3 különböző helyi keresési algoritmust és 63 tesztfüggvényt vizsgáltunk, hogy elegendő adatunk legyen. Mindegyik konfigurációt (globális optimalizáló, helyi kereső és teszt függvény) 100-szor futtattuk. Minden eredményt eldobtunk a 100 futásból, ha maximum 10^5 függvényértékelés mellett sikertelen volt a globális optimum megtalálása, majd a fennmaradó értékek átlagát vettük. Az optimum érték megengedett elérését az alábbi képlet határozza meg:

$$|F^*(x) - F(x)| \leq 10^{-8} + |F(x^*)| \cdot 10^{-6}.$$

Az összes futtatáshoz képest a sikeres futások arányát nevezzük az optimalizálás robusztusságának. Csak azokat a konfigurációkat tanulmányozzuk, amelyek 100%-os robusztussággal rendelkeznek mindkét globális optimalizáló számára.

A kapott két adatvektor közötti korreláció 99,87%. Az eredmények közti különbséget számos tényező okozza. Az algoritmusok nem determinisztikusak, generált véletlen számok alapján futnak, amelyek néhány százalékos hibát okozhatnak. A két optimalizáló más módon dolgozza fel a keletkezett adatokat, ami szintén okozhat bizonytalanságot. Ezeket a különbségeket a helyi keresők jelentősen felerősíthetik. Ha minden lokális keresés megbízhatóan konvergálna a globális optimumhoz, a függvénykiértékelések számát egy helyi keresés adná. Több helyi keresés esetén a pontos szám nagyon bizonytalan. Véletlenszerű kezdőpontokkal és nem optimalizálva a lépéshosszt a helyi keresés közel véletlenszerű helyi optimumhoz konvergál. A függvénykiértékelések száma így megközelíti a helyi keresések függvénykiértékeléseinek számát, mely ezekben az esetekben instabil. Továbbá a tesztfüggvények között van olyan, amelynek sok helyi optimuma van. Így a két adathalmaz közötti erős korreláció megerősít minket abban, hogy az eredeti GLOBAL és a párhuzamos változat az alapvető működésben hasonlóan viselkedik.

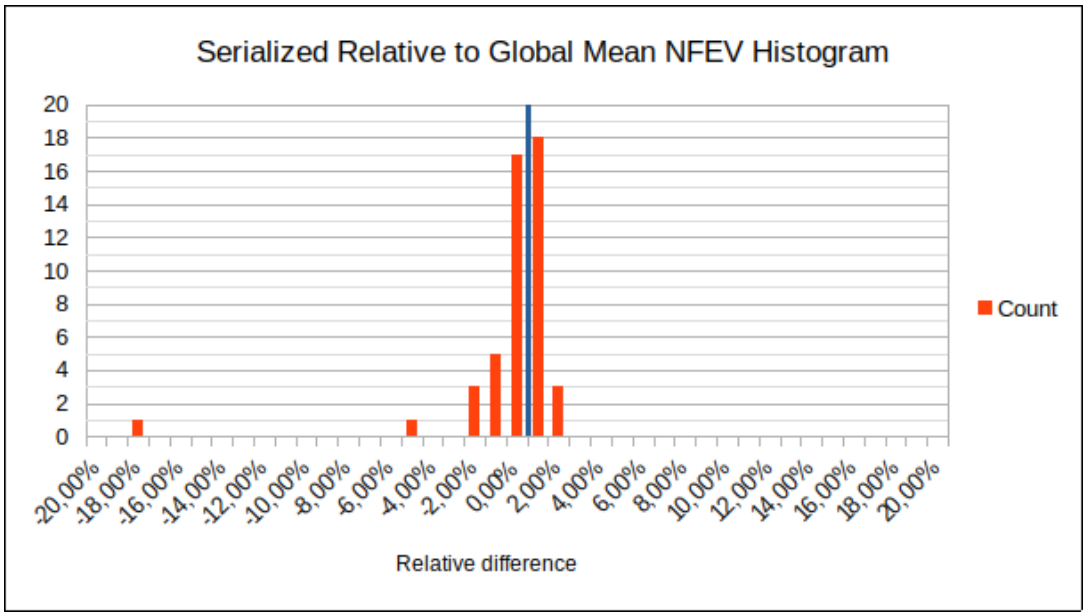
Az esetek többségében a relatív különbség elhanyagolható (5.1. ábra), néhány százalék alatti. Az átlagos relatív különbségek szintén nullához közelítenek, vagyis a mögöttes algoritmusváltozatok alapvetően azonosak. Az 5.1. ábrán a legtöbb relatív hibaeredmény a $[-2\%, 3\%]$ tartományba esik, mely azt mutatja, hogy a GLOBAL átlagosan kevesebb függvényértékelést használ, de a különbség kicsi. A szélsőséges esetek a párhuzamos változatnak kedveztek, de ez valószínűleg a véletlen műve.

5.8. A párhuzamosított GLOBAL előnyei

A futtatásokkal két szempontból vizsgáltuk a rendszert. Egyrészt az a kérdés, hogy a párhuzamosítással sikerült-e növelni a teljesítményt. Másrészt, hogy a GLOBAL-hoz képest sikerült-e gyorsítani az optimalizálási folyamatot. A függvények nehézségét 1 és 1000 között változtattuk 10 hatványaival, ami azt jelenti, hogy egy kiértékelés során ennyiszer számoltuk ki a függvényt. A szálak száma 1 és 16 között 2 hatványaival nőtt. A kiértékelések maximális száma továbbra is 10^5 volt. A jelen dolgozatban a két szélsőséges esetet mutatom be.

A párhuzamos változat speciális függvényeken sajnos rosszul viselkedik. Az 5.2. ábrán a Zakharov 40 dimenziós tesztfüggvény futási eredményei láthatók.

Név: Zakharov Function



5.1. ábra. A relatív különbségek hisztogramja a függvényértékeléseinek számában a GLOBAL és a párhuzamos változat között.

Dimenzió: 40

Függvény: $f(x_1, \ldots, x_{40}) = \sum_{i=1}^{40} x_i^2 + \left(\sum_{i=1}^{40} 0.5ix_i\right)^2 + \left(\sum_{i=1}^{40} 0.5ix_i\right)^4$

Keresési tér: $-5 \leq x_1, \ldots, x_{40} \leq 10$

Globális minimum: $f(0, \ldots, 0) = 0$

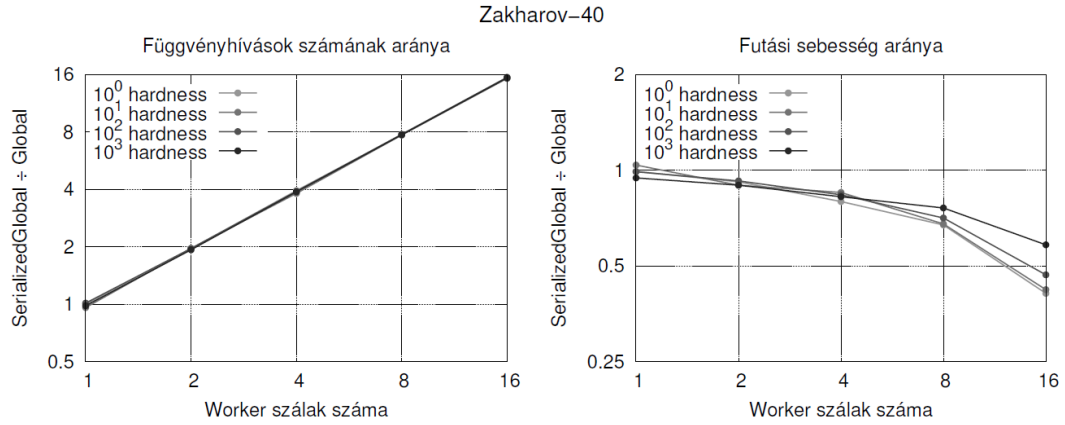
Mivel a függvény egyetlen lokális keresés segítségével is megoldható, így a GLOBAL az első lokális keresés után meg is találja az optimumot. A párhuzamos változatban az első lokális keresés befejeződésével szintén megtaláltnak nyilvánítja az optimumot, azonban minden szál nagyjából egyszerre indít el egy lokális keresést. A szálak nem értesülnek időben az eredményről, ezért annyi lokális keresés fut le, ahány szál van. A szálak számával megegyező mennyiségű, N darab keresés a kiértékelések számát közel N-szeresére növeli. A terhelés dinamikus növekedése miatt nincs lehetőségünk javítani az eredményen, de a párhuzamosítást biztosító egyéb programrészek még tovább lassítják a futást.

A kifejezetten előnyös probléma, a Spikes-tesztfüggvény futási eredményei az 5.3. ábrán láthatók.

Név: Spikes Function

Dimenzió: 2

Függvény:



5.2. ábra. Összehasonlítás a Zakharov-40 függvényen.

$$f(x_1, x_2) = \begin{cases} \sin(2\pi x_1) \cdot \sin(2\pi x_2) + 1002, \text{ ha } ||x - [15.25; 15.75]||_2 > \dfrac{1}{4} \\ 1000, \text{ egyébként.} \end{cases}$$

Keresési tér: $-40 \leq x_1, x_2 \leq 40$

Globális minimum: $f(15.25; 15.75) = 1000$

A függvény sok, azonos méretű vonzáskörzetű lokális optimumból áll, amelyek közül egy kitüntetett a globális optimumot tartalmazza. A függvény jellege miatt sok lokális keresés szükséges az optimum megtalálásához. Így a párhuzamos változat képes kihasználni a párhuzamosítás előnyeit. Az eredmények önmagukért beszélnek, míg a kiértékelések száma a szálak számától függetlenül megegyezik a GLOBAL értékeivel, a futási sebességek a probléma nehézségétől függően jelentős javulást mutatnak. Mivel a függvények nehezítése nem arányos a kiértékelési időkkel, a GLOBAL 1000-szeres futási idő helyett csak 15-szörös emelkedést tapasztalt. A futási idők megfelelő skálázása mellett még jobb eredményeket várhatunk.

5.9. A GLOBAL hasznosulása a káosz keresés területén

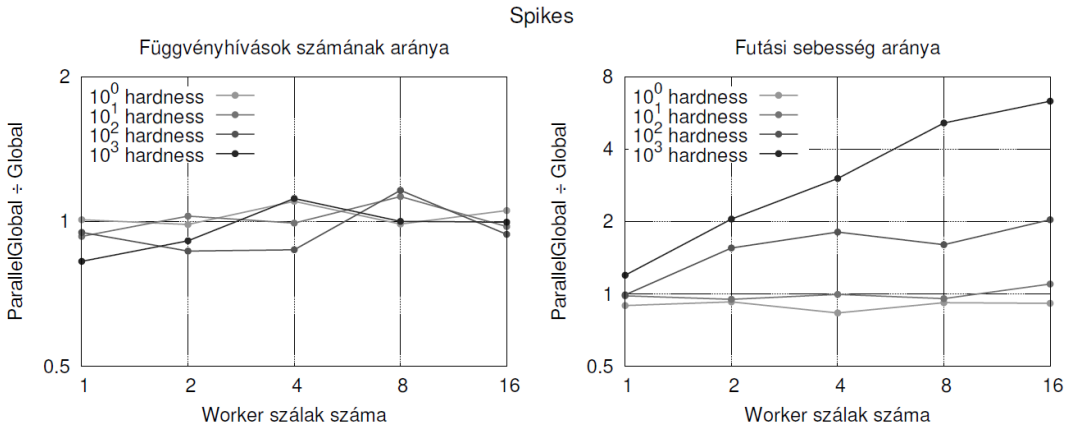
A GLOBAL eljárást elsőként a kaotikus régiók keresésére használtuk.

5.1. Tétel. *Tegyük fel, hogy egy folytonos $\varphi : L \cup R \to X$ leképezésre igaz, hogy*

$$\varphi(a) \cup \varphi(d) \subset \mathcal{O}_R, \ \varphi(b) \cup \varphi(c) \subset \mathcal{O}_L,$$

$$\text{és } \varphi(L \cup R) \subset X \setminus E$$

(lásd az 5.4. ábrát). Ekkor φ rendelkezik kaotikus régióval $L \cup R$-en.



5.3. ábra. Összehasonlítás a Spikes-függvényen.

Mint az megfigyelhető, a három geometriai feltétel teljesen hasonló részhalmaz vizsgálaton alapul. A korábban publikált [27] algoritmus képes eldönteni, hogy egy adott rendszer rendelkezik-e a fenti geometriai tulajdonságokkal.

A patkó létezéséhez az $\mathcal{O}_L$, $\mathcal{O}_R$ és E régiók csak segédterületek. Ezek segítségével azt érjük el, hogy az L és R régiók képei „átmenjenek" az L és R régiókon. Vezessük be az alábbi, általánosabb definíciót [70, 94]:

5.2. Definíció. *Azt mondjuk, hogy az N_i négyszög átfedi az N_j négyszöget az f leképezés mellett (jelölés: $N_i \overset{f}{\Longrightarrow} N_j$), ha*

1. *N_i f melletti képének nincs közös pontja N_j „vízszintes" oldalaival, és*
2. *N_i „függőleges" oldalainak f melletti képeinek nincs közös pontja N_j-vel, továbbá ezek N_j átellenes oldalain vannak.*

A könnyebb átláthatóság kedvéért vezessük be az alábbi jelölést:

5.3. Definíció. *Tekintsük az $N_1, \ldots, N_p$ páronként diszjunkt négyszögeket. Ekkor az $A = (a_{i,j})_{i,j=1}^{p}$ négyzetes mátrixot átmenetmátrixnak hívjuk, ha:*

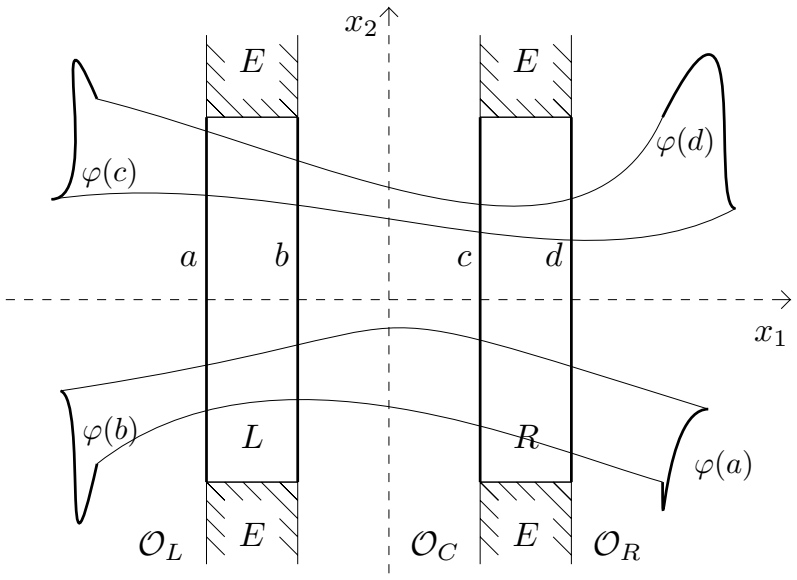
$$a_{i,j} = \begin{cases} k, & \text{ha } N_i \overset{f^k}{\Longrightarrow} N_j, \\ 0, & \text{egyébként.} \end{cases}$$

Egy leképezésnek az alábbi átmenetmátrixszal kell rendelkeznie az L és R régiókra, hogy a k. iteráltja kaotikus legyen:

	L	R
L	k	k
R	k	k

.

Rendelkezünk egy teljesen automatizált ellenőrző eljárással, így lehetőségünk van azt egy optimalizáló eljárással kombinálni, mellyel együtt az képes lehet új kaotikus tartományok keresésére. Ennek az optimalizáló eljárásnak a paraméterei általában a négyszögek



5.4. ábra. Az L, R halmaz és képeinek szükséges viszonya.

csúcspontjainak koordinátái, de lehetnek akár a leképezés paraméterei is. A célfüggvény pedig egy olyan nemnegatív büntetőfüggvényt, mely megmutatja, hogy mennyire sérti meg az adott konstrukció a káosz létezésének feltételeit.

Ezzel az eljárással kapcsolatban két nehéz feladatot emelnék ki, amelyekről korábban nem volt fellelhető biztos állítás a szakirodalomban. Mind a kettő az Hénon-leképezéssel kapcsolatos.

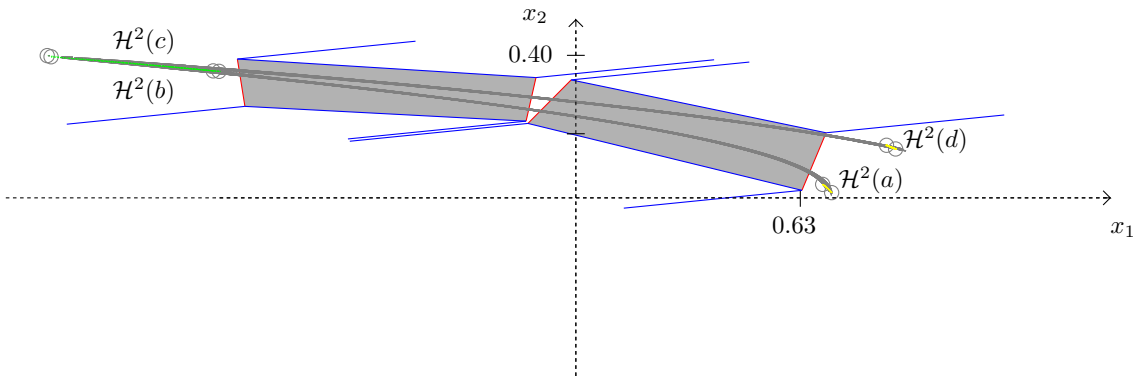
5.4. Definíció. *Az Hénon-leképezés:*

$$\mathcal{H}(x_1, x_2) = (1 - Ax_1^2 + x_2, Bx_1).$$

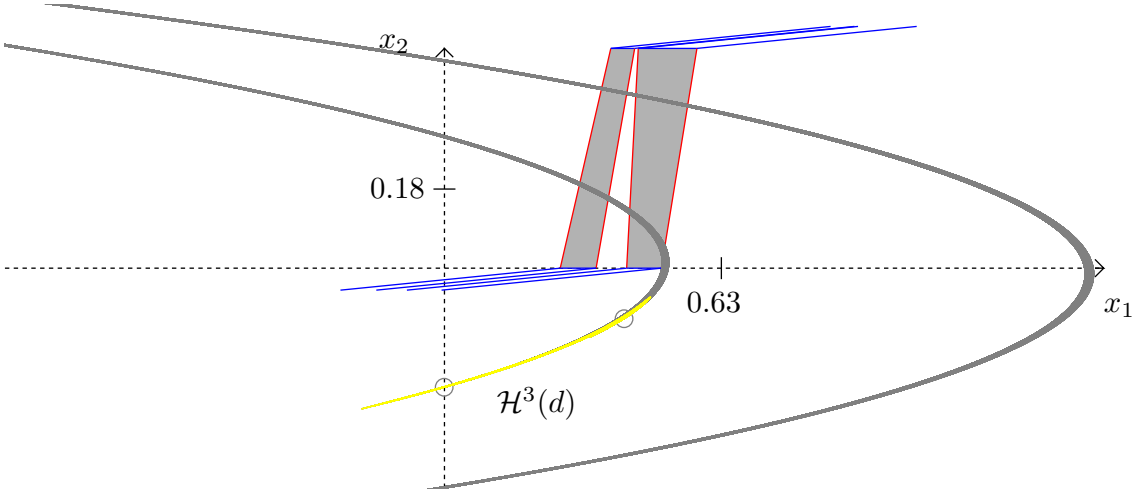
Az első esetben az Hénon-leképezés második iteráltját vizsgáltuk. Zgliczyński korábbi megközelítése sikertelen volt – a klasszikus paraméterek mellett – a kaotikus régió keresése a második és negyedik iteráltakra. Ezért az előbb említett, általánosabban alkalmazható technikát próbáltuk meg használni ezekre az esetekre. Az eljárásunk a nehezebbnek vélt második iteráltra is az előzőkhöz hasonlóan mind a 8 csúcspont keresésével talált optimális megoldást (lásd az 5.5 ábrát):

$$\begin{aligned} V_{ul}^L &= (-0.95008818,\ 0.38966840), & V_{ur}^L &= (-0.11192965,\ 0.33756351),\\ V_{ll}^L &= (-0.92886824,\ 0.25677498), & V_{lr}^L &= (-0.13972836,\ 0.21535493),\\ V_{ul}^R &= (-0.01309659,\ 0.33113756), & V_{ur}^R &= (0.70239604,\ 0.18263519),\\ V_{ll}^R &= (-0.13451910,\ 0.20890361), & V_{lr}^R &= (0.63453236,\ 0.02108996). \end{aligned}$$

A második esetben egy olyan Hénon-leképezést vizsgáltunk, amelyik területtartó, ekkor a B paraméternek 1-nek kell lennie. Az A paramétert szabadon megválaszthatjuk. A kísérleteinkben azt tapasztaltuk, hogy minél nagyobb ez a paraméter, annál laposabb az



5.5. ábra. Az Hénon-leképezés második iteráltja.



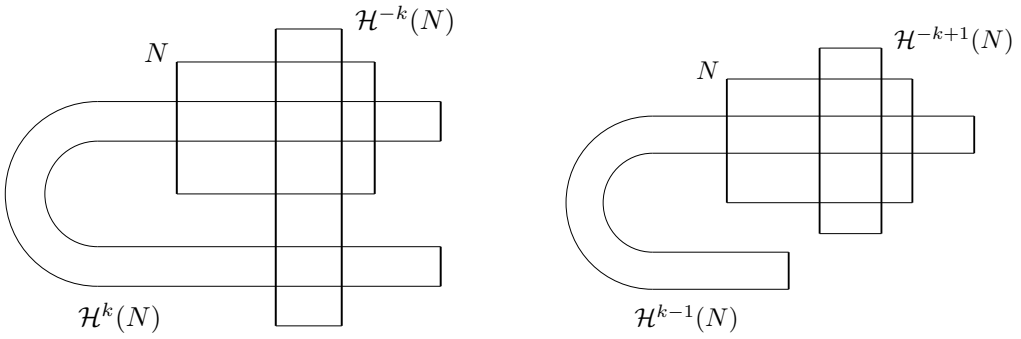
5.6. ábra. Kaotikus régió egy területtartó Hénon-leképezés esetében.

instabil sokaság, és így könnyebb befoglalni négyszögekkel a kaotikus régiót. Ezen okok miatt az A paramétert 6.2-nek választottuk. Ezekkel a paraméterekkel a leképezés sokkal korábbi fázisban is mutatja a kaotikus jelleget, így a harmadik iterálttal is megpróbálkoztunk.

A keresés sikeres volt [10], és az optimalizáló az alábbi kaotikus régiót találta (lásd az 5.6. ábrán):

$$\begin{aligned} V_{ul}^L &= (0.37846661,\ 0.50), & V_{ur}^L &= (0.43276909,\ 0.50),\\ V_{ll}^L &= (0.26337142,\ 0.00), & V_{lr}^L &= (0.3449795,\ 0.00),\\ V_{ul}^R &= (0.44094381,\ 0.50), & V_{ur}^R &= (0.57439378,\ 0.50),\\ V_{ll}^R &= (0.41452329,\ 0.00), & V_{lr}^R &= (0.49339729,\ 0.00). \end{aligned}$$

Tudjuk, hogy az Hénon-leképezés második, negyedik, hatodik és hetedik iteráltja rendelkezik L-R típusú káosszal. A periodikus pontok számából ismeretes, hogy maga az



(a) A $\mathcal{H}^k$ és a $\mathcal{H}^{-k}$ melletti képek. (b) A $\mathcal{H}^{k-1}$ és a $\mathcal{H}^{-k+1}$ melletti képek.

5.7. ábra. A Smale bizonyításában szereplő átfedések szerkezete.

Hénon-leképezés nem rendelkezik hasonló régióval, továbbá a harmadik és ötödik iteráltak sem rendelkezhetnek. A sejtésünk az, hogy a hetedik iterált után minden iterált rendelkezik kaotikus tartománnyal.

Smale adott egy tisztán elméleti bizonyítást arra, hogy az Hénon-leképezés összes magasabb iteráltjainak létezik L-R típusú kaotikus tartománya [75]. A bizonyításban csak azt mondta ki, hogy egy elég nagy iterációs szám után biztosan létezik ilyen régió. De ezen iteráció nagyságára konkrét értéket nem adott meg.

Bizonyítása során egy kétdimenziós régió (N) adott iteráltak melletti képeit és őseit vizsgálta. Az állítás igazolásához ezen régió k-adik és $(-k)$-adik képének két átfedésének kell lenni. Az átfedések a stabil és az instabil sokaság metszéspontjainál jöhetnek létre. Az egyik ezek közül biztosan a fixpont környezetében található. A másik átfedési pontra teljesülni kell, hogy a $(k-1)$-edik és a $(-k+1)$-edik iterált képei még nem érik el ezt az átfedési régiót. Így ezen második átfedést a stabil és instabil sokaság második metszéspontjában kell keresni. Ezt a szituációt láthatjuk az 5.7. ábrán. Smale azt állítja, hogy létezik ilyen tulajdonságokkal rendelkező régió, és ebben az esetben minden $k' \geq 2k$ iteráltra mindig létezik kaotikus régió. Bizonyításában nem használta ki a k és a $(-k)$ közötti szimmetriát, így nyugodtan vehetünk k-t és $(-l)$-et is. Természetesen ebben az esetben a $(k-1)$-edik és a $(-l+1)$-edik iteráltaknak nem szabad elérni az említett új átfedési régiót.

A megfelelő síkidom csúcspontjainak a fixpont környékén kell lennie. A fixpont garantált befoglalása:

$$\left([0.631354, 0.631355]\,,\ [0.189406, 0.189407]\right).$$

E körül fogjuk megkeresni a Smale bizonyításának megfelelő síkidomot. Jelen problémánál is használhatjuk a feltételek megfogalmazásához az átfedési tulajdonságot. Vegyük észre, hogy az átfedési tulajdonság szimmetrikus abban az értelemben, hogy ha $N_1 \overset{f}{\Rightarrow} N_2$, akkor N_2 is átfedi $f(N_1)$-et. Ezt a tulajdonságot felhasználva vezessük be az alábbi új jelölést a fedésre:

$$\langle f(N_1); N_2 \rangle \text{ , ha } N_1 \overset{f}{\Rightarrow} N_2.$$

Ezzel azt is tudjuk jelölni, ha egy adott régiónak az Hénon-leképezés k-adik és $(-l)$-edik képei fedik egymást:

$$\left\langle \mathcal{H}^k(N); \mathcal{H}^{-l}(N) \right\rangle .$$

Smale bizonyításában a megfelelő régiók kétszeresen is átfedik egymást. Így e két átfedés területét azonosítsuk Q_1-gyel és Q_2-vel. A Q_1 régió legyen a fixpont környezetében, míg a Q_2 a k-adik és a $(-l)$-edik leképezésben kialakult metszéspont környezetében. Jelöljük a Q_1-en kívüli területet $\overline{Q_1}$-sal. Olyan kétdimenziós régiót (N) kell tehát találni, melyre az alábbi feltételek igazak:

$$\left\langle \mathcal{H}^k(N) \cap Q_1; \mathcal{H}^{-l}(N) \cap Q_1 \right\rangle , \tag{5.1}$$

$$\left\langle \mathcal{H}^k(N) \cap Q_2; \mathcal{H}^{-l}(N) \cap Q_2 \right\rangle , \tag{5.2}$$

$$\mathcal{H}^{k-1}(N) \cap Q_2 = \emptyset,\ \mathcal{H}^{-l+1}(N) \cap Q_2 = \emptyset,$$

és

$$\left(\mathcal{H}^{k-1}(N) \cap \overline{Q_1}\right) \bigcap \left(\mathcal{H}^{-l+1}(N) \cap \overline{Q_1}\right) = \emptyset.$$

A megfelelő síkidom csúcspontjainak a fixpont környékén kell lennie. A fixpont garantált befoglalása:

$$\left([0.631354, 0.631355] ,\ [0.189406, 0.189407]\right) .$$

E körül fogjuk megkeresni a Smale bizonyításának megfelelő síkidomot. A kétszeres átfedéshez szükséges Q_1 és Q_2 régiókat a stabil és instabil sokaság elhelyezkedése alapján a

$$\begin{aligned} V_{ul}^{Q_1} &= (0.50, 0.25), \quad V_{ur}^{Q_1} = (0.70, 0.25), \\ V_{ll}^{Q_1} &= (0.50, 0.13), \quad V_{lr}^{Q_1} = (0.70, 0.13), \\ V_{ul}^{Q_2} &= (0.55, 0.13), \quad V_{ur}^{Q_2} = (0.63, 0.13), \\ V_{ll}^{Q_2} &= (0.55, 0.07), \quad V_{lr}^{Q_2} = (0.63, 0.07) \end{aligned}$$

csúcspontokkal vettük fel.

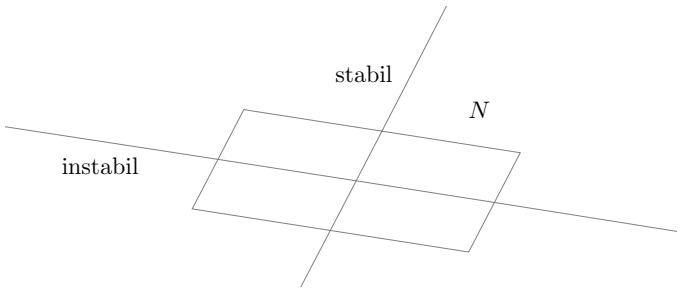
A GLOBAL eljárással a keresésünk sikeres volt, és μ-re és ν-re a 0.005 és a 0.0125 értékek megfelelők, melyekkel a

$$V_{ul}^N = (0.621310,\ 0.195769),\ V_{ur}^N = (0.646011,\ 0.191917),$$

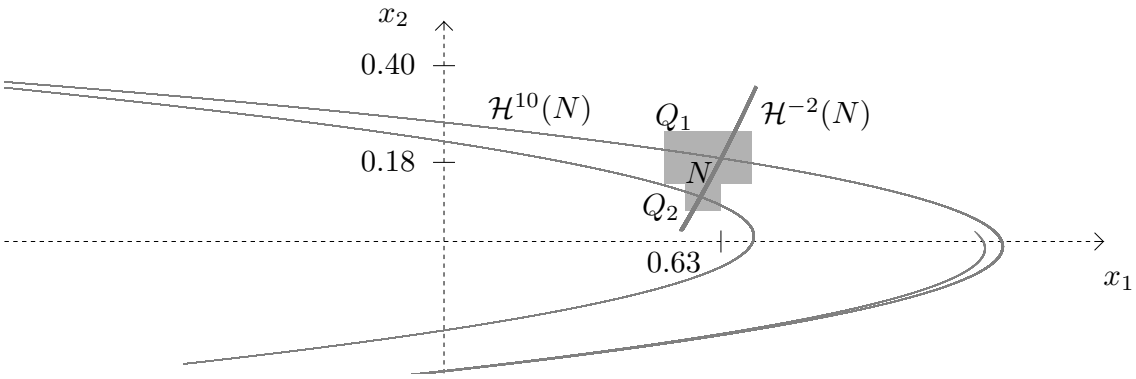
$$V_{ll}^N = (0.616698,\ 0.186896),\ V_{lr}^N = (0.641399,\ 0.183044)$$

négyszöget kaptuk (lásd az 5.8. ábrán).

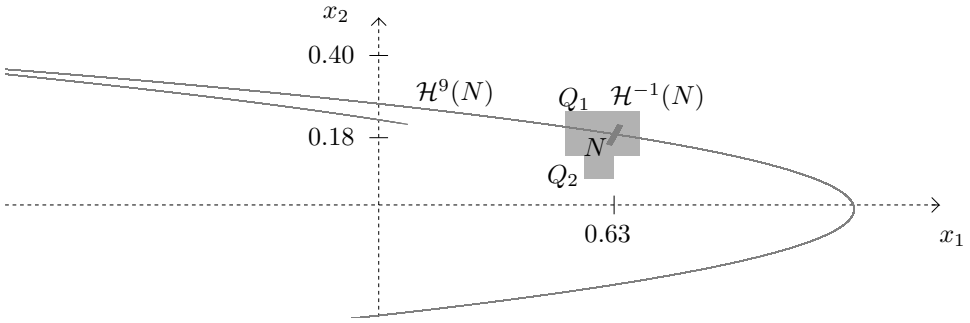
Ezzel bizonyítottuk, hogy az Hénon-leképezés tizenkettedik iteráltja után mindegyik rendelkezik L-R típusú káosszal. Így már csak a kilencedikre és a tizenegyedikre kellett ezt megmutatni. Ezekre is sikeres volt a korábban publikált általános kereső eljárás. Ezzel bizonyítottuk, hogy az Hénon-leképezés összes szóba jöhető iteráltja rendelkezik L-R típusú káosszal [11].



(a) A megfelelő síkidom valamint a stabil és instabil sokaság elhelyezkedése.



(b) A megfelelő síkidom $\mathcal{H}^{10}$ és $\mathcal{H}^{-2}$ melletti képei.



(c) A megfelelő síkidom $\mathcal{H}^{9}$ és $\mathcal{H}^{-1}$ melletti képei.

5.8. ábra. A Smale bizonyításának megfelelő négyszögek és ezek képei.

5.10. A GLOBAL hasznosulása az entrópia számítás területén

Galias és Zgliczyński 2001-ben publikált [39] egy alsó korlátot az Hénon-leképezésre. A technikájuk teljes mértékben emberi beavatkozáson múlt, „kézzel történt". A periodikus pontok és a stabil sokaság elhelyezkedésének ismeretében próbáltak elhelyezni olyan régiókat, melyek rendelkeznek az elvárt átfedési tulajdonságokkal. Ebben a cikkben a legjobb alsó korlátot adó szerkezetben 5 négyszöget (P_i) vizsgáltak, melyek között az alábbi átfedéseket találták:

	P_1	P_2	P_3	P_4	P_5
P_1				2	
P_2	2	2	2		
P_3	2	2			
P_4				1	3
P_5			1		

.

Ezzel a technikával az entrópiára egy 0.338-es alsó becslést kaptak. Ez az eljárás eléggé emberi erőforrás igényes, így mi megpróbálkoztunk automatizálni ezt az optimalizálás segítségével.

A periodikus pontok ismeretében több kísérletet is tettünk arra, hogy optimalizáló eljárásunkkal különböző átmenetmátrixoknak megfelelő négyszögeket keressünk. Az egyik legjobb, sikeres eredményt az entrópiára a következő átmenetmátrix alkalmazásával nyertük:

	Q_1	Q_2	Q_3	Q_4
Q_1	1	2		
Q_2			2	2
Q_3			2	2
Q_4	2	2		

. (5.3)

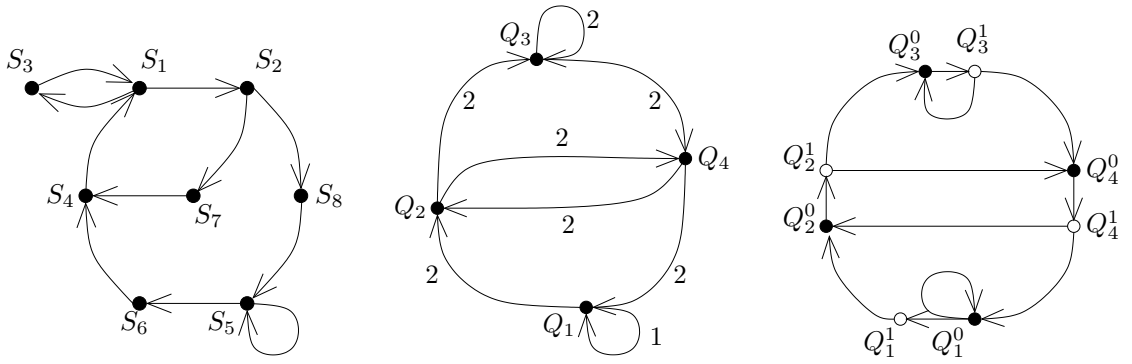
A fenti átmenetmátrixot és a belőle konstruálható 0-1-es átmenetmátrixot illusztrálja az 5.9(b) ábra és az 5.9(c) ábra. A fenti fedési mátrixhoz tartozó 0-1-es mátrix karakterisztikus polinomja a

$$p(\lambda) = \lambda^8 - \lambda^7 - \lambda^6 + \lambda^5 - \lambda^4 + \lambda^3 - \lambda^2.$$

A mátrix legnagyobb sajátértéke 1.465, azaz az entrópia legalább 0.382.

Galias egy másik cikkében [38] publikált további két, az előző eredményénél jobb eredményt az Hénon-leképezés entrópiájára. Az eredményei továbbra is számítógép nélkül jöttek létre. A gyengébb eredménye esetén összesen nyolc négyszöggel dolgozott, melyek átmenetmátrixát illusztráljuk az 5.9(a) ábrán. Ehhez az átmenetmátrixhoz tartozó karakterisztikus polinom megegyezik a mi legjobb eredményünkkel. Így e két struktúrából kapott eredmény entrópiája megegyezik, miközben a két átmenetmátrix teljesen különböző (lásd az 5.9(a) ábrát és az 5.9(c) ábrát).

Galias a [38] cikkében megemlít még egy 29 régiót tartalmazó átfedés mátrixot, mellyel 0.430-et kapott alsó korlátnak. Ez az érték már közel van a sejtett valódi entrópiához [39], a 0.465-es értékhez. Az értékek alakulását láthatjuk az 5.10. ábrán.

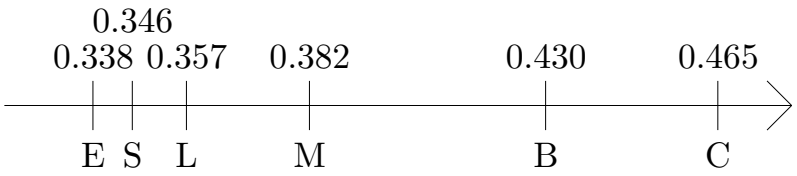


(a) Galias eredményének átmenetgráfja.

(b) Saját eredményünk átmenetgráfja.

(c) Saját eredményünk kiterjesztett átmenetgráfja.

5.9. ábra. Azonos karakterisztikus polinomú átmenetmátrixok.



5.10. ábra. Az Hénon-leképezés entrópiájára bizonyított alsó korlátok alakulása. Az E érték Galias és Zgliczyński első eredményét, S a $\mathcal{H}^2$-ből kapott eredményünket, L a lokális kereséssel kapott eredményünket, M a Galias által kapott és az azonos eredményünket adó értéket, B az eddig ismert legjobb eredményt, míg C a sejtett értéket jelöli.

Összegezve a technikánk újdonságát, míg Galias az ellenőrzéstől a keresésig mindent kézzel végzett, addig a mi módszerünk esetében az ellenőrzés teljesen automatizált, és a keresés is csak minimális emberi beavatkozást igényel, a feladat megoldásának nagy részét átvette az optimalizálás [8].

5.11. A GLOBAL hasznosulása a plazmonika területén

A plazmonika területén több olyan probléma van, mikor egy-egy új ötlet önmagában nem elegendő, hogy a benne rejlő lehetőséget bemutassa. Sok esetben egy új struktúra, mely jobb az elődjeinél, alapesetben nem mutatja ezt a hatékonyságot. Sokszor a struktúra megfelelő paramétereinek a megtalálása is nagy feladat. Ezeket a problémákat, a korábbi matematika területén fellelt problémákhoz hasonlóan, optimalizálási problémákká konvertáltuk. A numerikus optimalizálás a GLOBAL eljárással történt, míg a célfüggvény kiértékelése, illetve a feltételek teljesülésének ellenőrzése a COMSOL Multiphysics program segítségével folyt.

Mivel sok esetben az optimalizálandó probléma komoly informatikai nehézségeket okozott, így olyan keretrendszerre volt szükségünk, mely könnyedén fejleszthető és a

problémára hangolható. Az alapvető probléma a COMSOL kiértékelések nagy időigénye volt, ezért is fektettünk nagy hangsúlyt az optimalizálónk párhuzamosítására, de ezen felül további ötletek is kellettek.

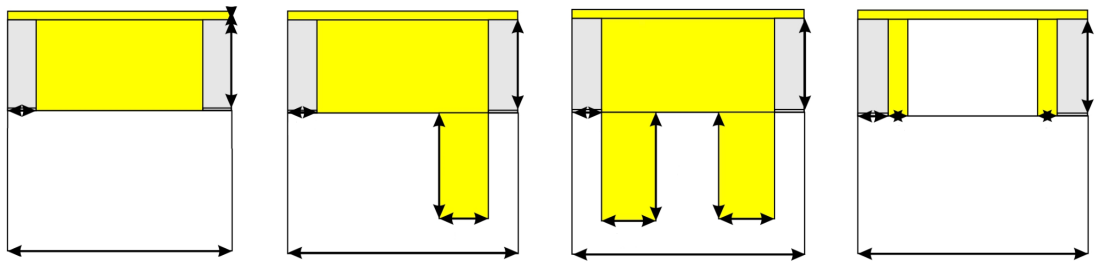
A GLOBAL alapvetően egy feltétel nélküli optimalizáló eljárás. A problémáink egytől egyig tartalmaztak feltételeket. Egyszerűbb esetekben ezek már a paraméterekben jelentkeztek, de voltak olyan feltételek is, melyeket a modell kiértékelésével tudtunk ellenőrizni. Mindkét esetben az optimalizálás területén megszokott és általunk már korább többször sikeresen alkalmazott büntetőfüggvényes megoldást használtuk. Ez azt jelenti, hogy a feltételeket kiértékeltük és amennyiben valamelyik feltétel nem teljesült, akkor a célfüggvény értékét növeltük egy pozitív konstanssal plusz egy a nemteljesülés nagyságával arányos tényezővel. Ha teljesül, akkor az eredeti célnak megfelelő értéket veszi fel a célfüggvény. Az illusztrációnkban legyen az adott paraméter x, mely mellett a feltételben szereplő $c(x)$ szimulációval kapott értékre szeretnénk, ha C-nél nagyobb lenne, míg a $g(x)$ értékét ezen feltétel mellett szeretnénk maximalizálni, melyről tudjuk, hogy nem lehet negatív. Ekkor a célfüggvény:

$$\min : f(x) = \begin{cases} \text{C-c(x)}, & \text{ha } c(x) <= C \\ \text{-g(x)}, & \text{egyébként.} \end{cases}$$

Ez a célfüggvény szerkezet az optimalizálót a megfelelő paramétertartományba tereli, így általában képesek vagyunk a feltételeknek megfelelő struktúrát találni.

A másik probléma az volt, hogy a COMSOL kiértékelője elég nagy zajjal tudja csak kiértékelni a függvényeket, és ennek a zajnak a csökkentése sok esetben a futási idő növekedésével érhető csak el. Ezekben az esetekben az optimalizáló olyan irányú fejlesztésére volt szükség, amelynek segítségével az optimum keresése során kevésbé lesz érzékeny a zajra, vagy a szükséges pontosságot automatikusan tudja hangolni a futás során. A zajból eredő probléma főként a lokális keresőknek okoz gondot. Ezek tipikusan hatékonyan tudják megtalálni a robosztus lokális optimumokat, de zajos függvények esetén ezen lokális optimumok megtalálása nehezebb. Erre a szakirodalomban több megoldás is van, de sajnos ezek elég sok függvénykiértékelést igényelnek [59]. Jelen esetben a problémát az Unirandi nevű helyi kereső fejlesztésével oldottuk meg. Jelentősen megemeltük a hatékony irány keresés megengedett próbálkozásainak számát. További hatékonyságot a vonal menti keresés megállási feltétel lazítása hozott, ahol a zaj nagyságának becsült mértékét is figyelembe vettük. Illetve a különböző pontosságú modellek esetében ezen várható zaj nagysága is változó volt, így a keresés során a zaj mértéke is részben szabályozható volt, természetesen a modell kiértékelési idő rovására.

Találkoztunk olyan problémákkal is, melyek esetében egy COMSOL-os kiértékelés futási ideje több órányi volt. Szerencsére ezekben az esetekben nagyon jó közelítő függvénymodellek léteztek az optimalizálandó célfüggvényre. Azaz a globális viselkedése jól leírható volt a fizikából ismert közelítő függvényekkel. Így a szükséges viselkedésű változat megtalálásához a közelítő függvény is alkalmas volt, de a számunkra érdekes viselkedésnek nem volt ismert a képlete, így megtalálásához már szükséges volt az eredeti modell kiértékelése. Egy olyan helyi keresőt fejlesztettünk, mely a valódi kiértékelést és a közelítő modellből számított kiértékelést hatékonyan tudta együttesen használni. A lokális keresés során eleinte csak a közelítő képletet használtuk, míg mikor egyre kisebb volt



(a) NCAI (b) NCDAI (c) NCDDAI (d) NCTAI

5.11. ábra. A vizsgált négyféle detektor az optimalizálandó geometriai paraméterekkel.

a lépésköz egyre többször értékeltük ki a modellt, mely kiértékeléseket a közelítő modell illesztésére is tudtuk használni.

Ezek a fejlesztések nélkül több probléma számunkra megoldhatatlannak bizonyult, így az adott feladat megkövetelte a fenti fejlesztéseket, melyekkel aztán sikeresek voltunk. Ezek a módszerek adott feladatokra lettek kifejlesztve, hasonló problémák megoldására szolgáló algoritmusok ötletei alapján. Mindegyik esetben az adott problémára lettek hangolva, illetve sok esetben a modellek speciális tulajdonságait is kihasználtuk. Emiatt a megoldó módszereink általános eljárásoknak nem tekinthetők, a szakirodalomban fellelhető hasonló eljárásokkal az összehasonlításuk nem lehetséges.

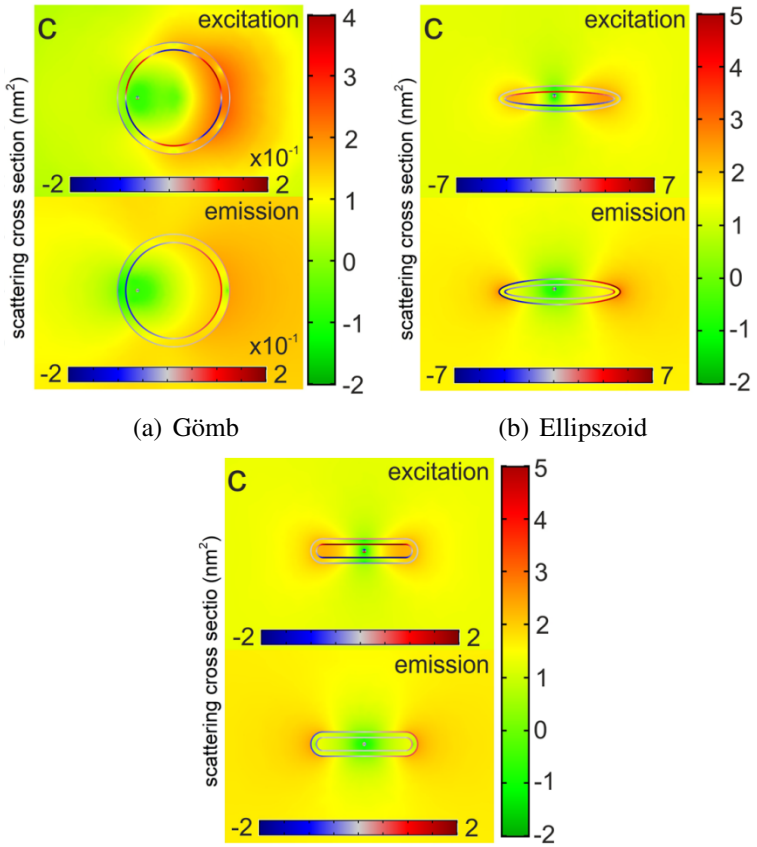
Az első fejlesztést igénylő feladatban az volt a cél, hogy maximalizáljuk a szupravezető nanovezetékes egyfotondetektorok (SNSPD) polarizációs kontrasztját [30]. Az SNSPD-k négy különböző nanoszál mintázatú, egydimenziós periodikus plazmonszerkezettel integrált rendszerek, melyek képesek a p-polarizált foton-szelektivitást közvetíteni a nióbium-nitrid szupravezető felé. Ezen szupravezető segítségével képesek a detektálás tényét rögzíteni. Optimalizálási szempontból a maximális arányú detektálás elérése mellett több különböző kritérium teljesülését kellett ellenőrizni. Ezekhez más modellek beállítása és kiértékelése is szükséges volt. A négy kapott struktúrát és azok optimalizálandó paramétereit az 5.11. ábrán láthatjuk.

A kapott geometriákkal jobb polarizációs kontrasztot tudtunk elérni, melyek mindegyike jobb volt az addig publikált legjobb eredményeknél.

A cél szempontjából hasonló feladat volt, amikor három különböző homorú ezüst mag-héj nanorezonátor konfiguráció geometriáját optimalizáltuk. A cél az volt, hogy egyidejűleg javítsuk a beágyazott szilícium üresedési (SiV) gyémánt színközpontok gerjesztését és kibocsátását [78]. Mint látható, itt a kettős, egymásnak ellentmondó cél megfelelő összehangolása is feladat volt. Ezen felül itt is feltételes optimalizálást végeztünk a SiV színközpontok 20–30–40 és 50%-os látszólagos kvantumhatékonyságának (cQE) biztosítására. A feltételesen optimalizált csatolt rendszereket a gerjesztés és az emisszió sugárzási sebességnövekedésének megfelelő szorzata jellemezte, amelyet P_x faktorként jelöltünk meg. Az optimalizált, központi emittert tartalmazó gömbmag-héj nanorezonátor a kötési dipoláris rezonancia révén képes jelentősen növelni az emissziót.

A P_x faktor 529-szeres 49,7%-os cQE-vel az emissziónál. Az emitter decentralizálása magasabb rendű nem sugárzó többpólusú módok megjelenéséhez vezetett, ez a jelenség

korábban nem volt ismert. Az optimalizált ellipszoid mag-héj rezonátor transzverzális és longitudinális dipoláris rezonanciáját a gerjesztésre és az emisszióra hangoltuk, ami $6,2 \cdot 10^5$ P_x faktort eredményezett. A rúd alakú homorú mag-héj nanorezonátorok hasonló transzverzális és longitudinális dipoláris rezonanciát aknáznak ki, ráadásul antennaszerű geometriájuknak köszönhetően jelentősen fokozzák a fluoreszcenciát. A $8,34 \cdot 10^5$-es javulást mutató P_x tényező elérhető. Ennél a rendszernél több esetben új fizikai jelenséget fedeztünk fel, mely a vizsgált rendszerekben nem volt korábban megfigyelt. Illetve a korábbi feladatokhoz hasonlóan, jobb értékeket tudtunk elérni a korábban publikáltakhoz képest. A talált objektumok gerjesztését és emisszióját az 5.12. ábrán láthatjuk.



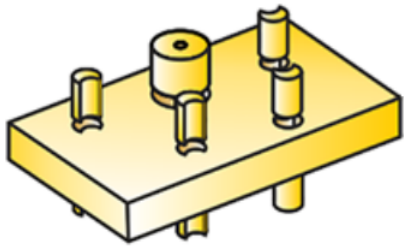
(a) Gömb (b) Ellipszoid

(c) Rúd

5.12. ábra. A vizsgált háromféle mag-héj nanorezonátorok.

A következő feladatban az optimalizálónak a fő célja nem egy maximális érték elérése volt, hanem egy olyan jelenség keresése, amelynek még az előfordulása sem volt bizonyított a szakirodalomban. Mint látható volt, korábban nem tudatosan, de ilyen jelenségeket találtunk más rendszereknél is. Bár akkor az eredeti célokat jobban teljesítő rendszerek kivételesen hoztak ilyen jelenségeket, nem tudatosan kerestük őket. De pont az optimalizáló, úgymond black-box-ként való alkalmazásának érdeme volt ez, mivel rátalált ezekre a ritka, kirívóan jól viselkedő rendszerekre.

A jelen esetben egy- és többrétegű szubhullámhosszú periodikus Babinet-komplemen-



5.13. ábra. A konvex-konkáv-konvex komplex mintázatrétegekkel felépített háromdimenziós struktúra.

ter mintázatokat vizsgáltunk, amelyek lekerekített nanoobjektum minitömbökből állnak. A konvex-konkáv-konvex komplex mintázatrétegekkel felépített háromdimenziós struktúrák negatív törésmutatót eredményeztek a látható tartomány határán mind lineárisan, mind körkörösen polarizált fénymegvilágításnál. Ennek az az oka, hogy a konvex nanoobjektumok dipoláris módusai rétegközi csatolás révén szinkronizálva vannak a homorú nanoobjektumokon egyidejűleg létező fordított dipólusokkal. Ezen tulajdonságot mutató objektumok geometriájának megtalálásán kívül informatikai eszközökkel kellett vizsgálni a szinkronizációt is [82].

5.12. Összefoglalás

A fejezetben tárgyalt GLOBAL algoritmus korábban már több problémát sikeresen oldott meg. Jelen fejezetben egy olyan hiánypótló fejlesztés egyik részletét mutattam be, aminek a jelenlegi kutatásaimban nagy hasznát veszem. Ez a GLOBAL Java változatának hatékony párhuzamosítása olyan esetekre, amikor a célfüggvény kiszámítása extrém költségesnek mondható. A doktorandusz hallgatóimmal több fejlesztést is végrehajtottunk a GLOBAL Java változatán. A fejlesztés mennyiségéből adódóan is több emberes munka volt. A munka programozói részét főként a diákok végezték. A fejlesztés összehangolása, az alapvető koncepciók meghatározása az én hozzájárulásom volt. Több párhuzamos változat is készült, melyek ötletei a fejlesztés során alakultak ki Zombori Dániel doktoranduszommal együttműködve. A jelen dolgozatban az egyik ilyen változatot írtam le, amivel több nehéz feladatot is sikerült megoldanunk.

A fejezet végén bemutatok még pár régebbi eredményt, melyeket a GLOBAL korábbi változataival oldottam meg. Ezek az Hénon-leképezés kaotikus régióinak keresésére és entrópia értékére vonatkozó eredmények. Ezen munkáim főként Garay Barna társszerzőmmel együttműködve valósultak meg, amiben az ő szerepe főként a matematikai háttérrel való megismertetés volt. A számítógépes megoldást már saját eredményemnek tekintem. Az utóbbi években a bemutatott párhuzamos algoritmust használom és magam fejlesztem az egyedi igényekhez igazítva, melyekből három plazmonikával kapcsolatos példát röviden be is mutatok. De ezeken felül számos további cikk jött létre ezzel a változattal hasonló területeken, melyekben az informatikai vonatkozású eredményeket magaménak

tekintem, illetve támogatom a fizikus csapatot azon ötleteimmel, hogy mik azok a kérdések, amik megválaszolhatók informatikai eszközökkel [31, 79, 81, 83, 84, 86, 87].

Köszönetnyilvánítás

Ezúton szeretném megköszönni elsősorban Csendes Tibornak, Garay Barnának, Hatvani Lászlónak és Krisztin Tibornak a segítségét, akik érdekes matematikai problémákkal ismertettek meg, ötleteket nyújtottak azok megoldásához, illetve megszerettették velem a kutatást.

Köszönetet mondok továbbá társszerzőimnek, akik a tudományos munkámat segítették, a kutatási eredményeim elérésében és publikálásában segédkeztek.

Különösen hálás vagyok azoknak a diákjaimnak, akik a kutatásaimban részt vettek és sok feladatot elvégeztek ezekben. Több eredményem ezen segítség nélkül a munkaigény nagysága miatt létre sem jöhetett volna. Különösen két doktoranduszomat említem, akik fiatal hallgató koruk óta részt vettek ezekben a munkákban és ötleteikkel is segítettek.

Végül, de nem utolsó sorban hálával tartozom szüleimnek, páromnak és gyerekeimnek, hogy türelmükkel, megértésükkel és segítségükkel támogattak.

Irodalomjegyzék

[1] Alefeld, G. & Herzberger, J.: Introduction to Interval Computations. Academic Press Inc. (New York) (1983)

[2] Alefeld, G. & Mayer, G.: Interval analysis: theory and applications. *Journal of Computational and Applied Mathematics*. **121**, pp. 421-464 (2000)

[3] Arnold, V.: Közönséges differenciálegyenletek. Műszaki Könyvkiadó (Budapest) (1978)

[4] Bagóczki, Z. & Bánhelyi, B.: A Parallel Interval Arithmetic-based Reliable Computing Method on a GPU. *Acta Cybernetica*. **23**, pp. 491-501 (2017)

[5] Bak, S., Tran, H., Hobbs, K. & Johnson, T.: Improved Geometric Path Enumeration for Verifying ReLU Neural Networks. *Computer Aided Verification*. pp. 66-96 (2020)

[6] Bánhelyi, B.: Egy késleltetett differenciálegyenlet vizsgálata megbízható számítógépes eljárással. *Alkalmazott Matematikai Lapok*. **24**, pp. 131-150 (2007)

[7] Bánhelyi, B., Biazzini, M., Montresor, A. & Jelasity, M.: Peer-to-Peer Optimization in Large Unreliable Networks with Branch-and-Bound and Particle Swarms. *Applications of Evolutionary Computing, EvoWorkshops 2009: EvoCOMNET, EvoENVIRONMENT, EvoFIN, EvoGAMES, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, EvoNUM, EvoSTOC, EvoTRANSLOG, Tübingen, Germany, April 15-17, 2009. Proceedings*. **5484**, pp. 87-92 (2009)

[8] Bánhelyi, B., Csendes, T. & Garay, B.: Rigorous lower bounds for the topological entropy via a verified optimization technique. *12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN 2006)*. pp. 10-10 (2006)

[9] Bánhelyi, B., Csendes, T. & Garay, B.: Optimization and the Miranda Approach in Detecting Horseshoe-type Chaos by Computer. *International Journal of Bifurcation and Chaos*. **17**, pp. 735-747 (2007)

[10] Bánhelyi, B., Csendes, T. & Garay, B.: A computer-assisted proof of chaotic behaviour of the area preserving Hénon map. *International Symposium on Nonlinear Theory and Its Applications*. pp. 596-699 (2008)

[11] Bánhelyi, B., Csendes, T. & Garay, B.: Σ_3-Chaos in Iterates of the Classical Hénon Mapping. *AIP Conference Proceedings*. **1168**, pp. 857-860 (2009)

[12] Bánhelyi, B., Csendes, T., Garay, B. & Hatvani, L.: A Computer-Assisted Proof of Σ_3-Chaos in the Forced Damped Pendulum Equation. *SIAM Journal on Applied Dynamical Systems*. **7**, pp. 843-867 (2008)

[13] Bánhelyi, B., Csendes, T. & Hatvani, L.: On the existence and stabilization of an upper unstable limit cycle of the damped forced pendulum. *Journal of Computational and Applied Mathematics*. **371**, 112702 (2020)

[14] Bánhelyi, B., Csendes, T., Krisztin, T. & Neumaier, A.: Global Attractivity of the Zero Solution for Wright's Equation. *SIAM Journal on Applied Dynamical Systems*. **13**, pp. 537-563 (2014)

[15] Bánhelyi, B., Csendes, T., Krisztin, T. & Neumaier, A.: A bounding scheme for proving the Wright conjecture on delay differential equations. *Acta Polytechnika Hungarica*. **15**, pp. 163-175 (2018)

[16] Bánhelyi, B., Csendes, T., Lévai, B., Pál, L. & Zombori, D.: The GLOBAL Optimization Algorithm. Springer (Cham) (2018)

[17] Bánhelyi, B., Palatinus, E. & Lévai, B.: Optimal circle covering problems and their applications. *Central European Journal of Operations Research*. **23**, pp. 815-832 (2015)

[18] Biazzini, M., Bánhelyi, B., Montresor, A. & Jelasity, M.: Distributed hyper-heuristics for real parameter optimization. *Genetic and Evolutionary Computation Conference, GECCO 2009, Proceedings, Montreal, Québec, Canada, July 8-12, 2009*. pp. 1339-1346 (2009)

[19] Borrelli, R. & Coleman, C.: Computers, Lies, and the Fishing Season. *The College Mathematics Journal*. **25**, pp. 401-412 (1994)

[20] Brendel, W., Rauber, J., Kümmerer, M., Ustyuzhaninov, I. & Bethge, M.: Accurate, reliable and fast robustness evaluation. *Advances in Neural Information Processing Systems*. **32**, pp. 12861–12871 (2019)

[21] Bunel, R., Lu, J., Turkaslan, I., Torr, P., Kohli, P. & Kumar, M.: Branch and Bound for Piecewise Linear Neural Network Verification. *Journal of Machine Learning Research*. **21**, pp. 1-39 (2020)

[22] Carlini, N. & Wagner, D.: Towards Evaluating the Robustness of Neural Networks. *2017 IEEE Symposium on Security and Privacy (SP)*. pp. 39-57 (2017)

[23] Cheng, C., Nührenberg, G. & Ruess, H.: Maximum Resilience of Artificial Neural Networks. *Automated Technology for Verification and Analysis*. pp. 251-268 (2017)

[24] CPLEX: User's Manual for CPLEX, https://www.ibm.com/docs/en/icos/22.1.0

[25] C-XSC: C-XSC Languages home page http://www.xsc.de/

[26] Csendes, T.: Nonlinear parameter estimation by global optimization - efficiency and reliability. *Acta Cybernetica*. **8**, pp. 361-370 (1988)

[27] Csendes, T., Bánhelyi, B. & Hatvani, L.: Towards a computer-assisted proof for chaos in a forced damped pendulum equation. *Journal of Computational and Applied Mathematics*. **199**, pp. 378-383 (2007)

[28] Csendes, T., Garay, B. & Bánhelyi, B.: A Verified Optimization Technique to Locate Chaotic Regions of Hénon Systems. *Journal of Global Optimization*. **35**, pp. 145-160 (2006)

[29] Csendes, T., Pál, L., Sendín, J.O.H. & Banga, J.R.: The GLOBAL Optimization Method Revisited. *Optimization Letters*. **2**, pp. 445-454 (2008)

[30] Csete, M., Szenes, A., Maráczi, D., Bánhelyi, B., Csendes, T. & Szabó, G.: Plasmonic Structure Integrated Single-Photon Detectors Optimized to Maximize Polarization Contrast. *IEEE Photonics Journal*. **9**, pp. 1-11 (2017)

[31] Csete, M., Szenes, A., Vass, D., Bánhelyi, B. & Dombi, P.: Few-cycle localized plasmon oscillations. *Scientific Reports*. **10**, 12986 (2020)

[32] Das, G., Das, S., Nandy, S. & Sinha, B.: Efficient algorithm for placing a given number of base stations to cover a convex region. *Journal of Parallel and Distributed Computing*. **66**, pp. 1353-1358 (2006)

[33] Dreyfus, S.: The numerical solution of variational problems. *Journal of Mathematical Analysis and Applications*. **5**, pp. 30-45 (1962)

[34] Dutta, S., Jha, S., Sankaranarayanan, S. & Tiwari, A.: Output Range Analysis for Deep Feedforward Neural Networks. *NASA Formal Methods*. pp. 121-138 (2018)

[35] Eijgenraam, P.: The Solution of Initial Value Problems Using Interval Arithmetic. (1981), https://api.semanticscholar.org/CorpusID:118045396

[36] Erdős, P.: On a new method in elementary number theory which leads to an elementary proof of the prime number theorem. *Proceedings of the National Academy of Sciences*. **35(7)**, pp. 374–384 (1949)

[37] Friedman, E.: Circles Covering Squares web page https://erich-friedman.github.io/packing/circovcir/

[38] Galias, Z.: Obtaining rigorous bounds for topological entropy for discrete time dynamical systems. *Proceedings of the International Symposium on Nonlinear Theory and Its Applications (NOLTA2002), Xi'an (China)*. pp. 619-622 (2002)

[39] Galias, Z. & Zgliczynski, P.: Abundance of homoclinic and heteroclinic orbits and rigorous bounds for the topological entropy for the Hénon map. *Nonlinearity*. **14**, pp. 909-932 (2001)

[40] Gehr, T., Mirman, M., Drachsler-Cohen, D., Tsankov, P., Chaudhuri, S. & Vechev, M.: AI2: Safety and Robustness Certification of Neural Networks with Abstract Interpretation. *2018 IEEE Symposium on Security and Privacy (SP)*. pp. 3-18 (2018)

[41] GLPK: GNU Linear Programming Kit, https://www.gnu.org/software/glpk/

[42] Goodfellow, I., Shlens, J. & Szegedy, C.: Explaining and Harnessing Adversarial Examples. *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. (2015)

[43] Gurobi Optimization, LLC: Gurobi Optimizer Reference Manual https://www.gurobi.com

[44] Hardy, G.H. & Wright, E.M.: An Introduction to the Theory of Numbers, Clarendon Press (1938)

[45] Heppes, A.: Covering a planar domain with sets of small diameter. *Periodica Mathematica Hungarica*. **53**, pp. 157-168 (2006)

[46] Heppes, A. & Melissen, H.: Covering a Rectangle with Equal Circles. *Periodica Mathematica Hungarica*. **34**, pp. 65-81 (1997)

[47] Hubbard, J.: The Forced Damped Pendulum: Chaos, Complication and Control. *American Mathematical Monthly*. **106**, pp. 741-758 (1999)

[48] Ivakhnenko, A. & Lapa, V.: Cybernetic Predicting Devices. CCM Information Corporation (1965)

[49] Jaquette, J.: A proof of Jones' conjecture. *Journal of Differential Equations*. **266**, pp. 3818-3859 (2018)

[50] Katz, G., Barrett, C., Dill, D., Julian, K. & Kochenderfer, M.: Reluplex: An Efficient SMT Solver for Verifying Deep Neural Networks. *Computer Aided Verification*. pp. 97-117 (2017)

[51] Kelley, H.: Gradient Theory of Optimal Flight Paths. *ARS Journal*. **30**, pp. 947-954 (1960)

[52] Krisztin, T.: Periodic orbits and the global attractor for delayed monotone negative feedback. *Electronic Journal of Qualitative Theory of Differential Equations*. **1999**, (2000)

[53] Krisztin, T., Walther, H. & Wu, J.: The structure of an attracting set defined by delayed and monotone positive feedback. *CWI Quarterly*. **12**, pp. 315-327 (1999)

[54] Kurakin, A., Goodfellow, I. & Bengio, S.: Adversarial Machine Learning at Scale. *International Conference on Learning Representations*. (2017), https://openreview.net/forum?id=BJm4T4Kgx

[55] Lévai, B. & Bánhelyi, B.: An optimization technique for verified location of trajectories with prescribed geometrical behaviour in the chaotic forced damped pendulum. *Central European Journal of Operations Research*. **21**, pp. 757-767 (2013)

[56] Lohner, R.: Einschliessung der Loesung gewoehnlicher Anfangs- und Randwertaufgaben und Anwendungen, Dissertation, Univ. Karlsruhe, (1988)

[57] Makino, K. & Berz, M.: COSY INFINITY Version 9. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*. **558**, pp. 346-350 (2006)

[58] Mawhin, J.: Periodic Oscillations of Forced Pendulum-like Equations. *Lecture Notes in Mathematics*. **964**, pp. 458-476 (1982)

[59] Mayer, A.; Mora, T.; Rivoire, O. & Walczak, A. M.: Diversity of immune strategies explained by adaptation to pathogen statistics. PNAS, **113(31)**, pp. 8630-8635 (2016)

[60] Mester, A., Zombori, D., Pál, L. & Bánhelyi, B.: Efficiency Improvement of the GLOBAL Optimization Method by Local Search Changes. *Acta Polytechnica Hungarica*. **19**, pp. 29-42 (2022)

[61] Moosavi-Dezfooli, S., Fawzi, A. & Frossard, P.: DeepFool: A Simple and Accurate Method to Fool Deep Neural Networks. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. pp. 2574-2582 (2016)

[62] Nedialkov, N.: The Design and Implementation of an Object-Oriented Validated ODE Solver. (2002), https://api.semanticscholar.org/CorpusID:9931826

[63] Nedialkov, N., Jackson, K. & Corliss, G.: Validated solutions of initial value problems for ordinary differential equations. *Applied Mathematics and Computation*. **105**, pp. 21-68 (1999)

[64] Nurmela, K.: Conjecturally optimal coverings of an equilateral triangle with up to 36 equal circles. *Experimental Mathematics*. **9**, pp. 241-250 (2000)

[65] Nurmela, K. & Östergard, P.: Covering a square with up to 30 equal circles. Laboratory for Theoretical Computer Science, Helsinki University of Technology (Helsinki), Research Reports (2000)

[66] Nwankpa, C., Ijomah, W., Gachagan, A. & Marshall, S.: Activation Functions: Comparison of trends in Practice and Research for Deep Learning. *2nd International Conference on Computational Sciences and Technology, Jamshoro, Pakistan*, pp. 124 - 133 (2021)

[67] Pál, L. & Csendes, T.: INTLAB implementation of an interval global optimization algorithm. *Optimization Methods and Software*. **24**, pp. 749-759 (2009)

[68] Papernot, N., McDaniel, P. & Goodfellow, I.: Transferability in Machine Learning: from Phenomena to Black-Box Attacks using Adversarial Samples. (2016), https://arxiv.org/pdf/1605.07277.pdf

[69] Papernot, N., McDaniel, P., Wu, X., Jha, S. & Swami, A.: Distillation as a Defense to Adversarial Perturbations Against Deep Neural Networks. *2016 IEEE Symposium on Security and Privacy (SP)*. pp. 582-597 (2016)

[70] Pireddu, M. & Zanolin, F.: Fixed Points for Dissipative-Repulsive Systems and Topological Dynamics of Mappings Defined on N-dimensional Cells. *Advanced Nonlinear Studies*. **5**, pp. 411-440 (2005)

[71] Raghunathan, A., Steinhardt, J. & Liang, P.: Certified Defenses against Adversarial Examples. *International Conference on Learning Representations*. (2018), https://openreview.net/forum?id=Bys4ob-Rb

[72] Ratschek, H. & Rokne, J.: New Computer Methods for Global Optimization. Ellis Horwood (Chichester) (1988)

[73] Selberg, A.: An Elementary Proof of the Prime-Number Theorem. *Annals of Mathematics*. **50(2)**, pp. 305–313 (1949)

[74] Singh, G., Gehr, T., Püschel, M. & Vechev, M.: An abstract domain for certifying neural networks. *Proceedings of the ACM on Programming Languages*. **3**, pp. 1-30 (2019)

[75] Smale, S.: Differentiable dynamical systems. *Bulletin of The American Mathematical Society*. **73**, pp. 747-817 (1967)

[76] Szabó, P., Markót, M., Csendes, T., Specht, E., Casado, L. & García, I.: New Approaches to Circle Packing in a Square – With Program Codes. Springer (Berlin) (2007)

[77] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I. & Fergus, R.: Intriguing properties of neural networks. *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*. (2014), http://arxiv.org/abs/1312.6199

[78] Szenes, A., Bánhelyi, B., Szabó, L., Szabó, G., Csendes, T. & Csete, M.: Improved emission of SiV diamond color centers embedded into concave plasmonic core-shell nanoresonators. *Scientific Reports*. **7**, 13845 (2017)

[79] Szenes, A., Vass, D., Bánhelyi, B. & Csete, M.: Active individual nanoresonators optimized for lasing and spasing operation. *Nanomaterials*. **11** (2021), https://www.mdpi.com/2079-4991/11/5/1322

[80] Tjeng, V., Xiao, K. & Tedrake, R.: Evaluating Robustness of Neural Networks with Mixed Integer Programming. *International Conference on Learning Representations*. (2017), https://api.semanticscholar.org/CorpusID:47016770

[81] Tóth, B., Szenes, A., Maráczi, D., Bánhelyi, B., Csendes, T. & Csete, M.: Polarization Independent High Absorption Efficiency Single-Photon Detectors Based on Three-Dimensional Integrated Superconducting and Plasmonic Patterns. *IEEE Journal of Selected Topics in Quantum Electronics*. **26**, pp. 1-9 (2020)

[82] Tóth, E., Bánhelyi, B., Fekete, O. & Csete, M.: Metamaterial properties of Babinet complementary complex structures. *Scientific Reports*. **13**, 4701 (2023)

[83] Tóth, E., Szalai, A., Somogyi, A., Bánhelyi, B., Csapó, E., Dékány, I., Csendes, T. & Csete, M.: Detection of biomolecules and bioconjugates by monitoring rotated grating-coupled surface plasmon resonance. *Opt. Mater. Express*. **7**, pp. 3181-3203 (2017)

[84] Tóth, E., Ungor, D., Novák, T., Ferenc, G., Bánhelyi, B., Csapó, E., Erdélyi, M. & Csete, M.: Mapping fluorescence enhancement of plasmonic nanorod coupled dye molecules. *Nanomaterials*. **10** (2020), https://www.mdpi.com/2079-4991/10/6/1048

[85] Van den Berg, J. & Jaquette, J.: A proof of Wright's conjecture. *Journal of Differential Equations*. **264**, pp. 7412-7462 (2018)

[86] Vass, D., Szenes, A., Bánhelyi, B., Csendes, T., Szabó, G. & Csete, M.: Superradiant diamond color center arrays coupled to concave plasmonic nanoresonators. *Opt. Express*. **27**, pp. 31176-31192 (2019)

[87] Vass, D., Szenes, A., Bánhelyi, B. & Csete, M.: Plasmonically enhanced superradiance of broken-symmetry diamond color center arrays inside core-shell nanoresonators. *Nanomaterials*. **12** (2022), https://www.mdpi.com/2079-4991/12/3/352

[88] Vinkó, T. & Ratz, D.: A Multidimensional Branch-and-Prune Method for Interval Global Optimization. *Numerical Algorithms*. **37**, pp. 391-399 (2004)

[89] Wang, S., Pei, K., Whitehouse, J., Yang, J. & Jana, S.: Efficient formal safety analysis of neural networks. *Proceedings of The 32nd International Conference on Neural Information Processing Systems*. pp. 6369-6379 (2018), http://arxiv.org/abs/1809.08098

[90] Wang, S., Pei, K., Whitehouse, J., Yang, J. & Jana, S.: Formal security analysis of neural networks using symbolic intervals. *Proceedings of The 27th USENIX Conference on Security Symposium*. pp. 1599-1614 (2018), http://arxiv.org/abs/1804.10829

[91] Weng, L., Zhang, H., Chen, H., Song, Z., Hsieh, C., Daniel, L., Boning, D. & Dhillon, I.: Towards Fast Computation of Certified Robustness for ReLU Networks. *Proceedings of The 35th International Conference on Machine Learning*. **80**, pp. 5276-5285 (2018), https://proceedings.mlr.press/v80/weng18a.html

[92] Wong, E. & Kolter, Z.: Provable Defenses against Adversarial Examples via the Convex Outer Adversarial Polytope. *Proceedings of The 35th International Conference on Machine Learning*. **80**, pp. 5286-5295 (2018), https://proceedings.mlr.press/v80/wong18a.html

[93] Wright, E.: A non-linear difference-differential equation. *Journal für die reine und angewandte Mathematik*. **194**, pp. 66-87 (1955)

[94] Zgliczyński, P. & Gidea, M.: Covering relations for multidimensional dynamical systems. *Journal of Differential Equations*. **202**, pp. 32-58 (2004)

[95] Zombori, D. & Bánhelyi, B.: Effects of Pooling in ParallelGlobal with Low Thread Interactions. *Informatica*. **45**, pp. 191-196 (2021)

[96] Zombori, D., Bánhelyi, B., Csendes, T., Megyeri, I. & Jelasity, M.: Fooling a Complete Neural Network Verifier. *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021, Conference Track Proceedings*. (2021), https://openreview.net/forum?id=4IwieFS44l

A. függelék

A GLOBALJ algoritmusai

7. Algoritmus. GLOBAL

Input

F: $\mathbb{R}^n \to \mathbb{R}$

$a, b \in \mathbb{R}^n$: lower and upper bounds

Output

$opt \in \mathbb{R}^n$: a global minimum candidate

$i \leftarrow 1$, $N \leftarrow 100$, $\lambda \leftarrow 0.5$, $opt \leftarrow \infty$
new, $unclustered$, $reduced$, $clustered \leftarrow \{\}$
while $stopping\ criteria$ **is** false **do**
  $new \leftarrow new \cup$ **generate** N **sample** from $[a, b]$ distributed $uniformly$
  $merged \leftarrow$ **sort** $clustered \cup new$ by $ascending$ order regarding F
  $last \leftarrow i \cdot N \cdot \lambda$
  $reduced \leftarrow$ **select** $[0, ..., last]$ **element** from $merged$
  $x^* \leftarrow$ **select** $[0]$ **element** from $reduced$
  $opt \leftarrow$ **minimum** of $\{opt, x^*\}$
  $clustered$, $unclustered \leftarrow$ **cluster** $reduced$
  $new \leftarrow \{\}$
  while size of $unclustered > 0$ **do**
    $x \leftarrow$ **pop** from $unclustered$
    $x^* \leftarrow$ **local search** over F from x within $[a, b]$
    $opt \leftarrow$ **minimum** of $\{opt, x^*\}$
    cluster x^*
    if x^* **is** not clustered **then**
      create cluster from $\{x^*, x\}$
    end if
  end while
  $i \leftarrow i + 1$
end while
return opt

8. Algoritmus. Recursive-single-linkage-clustering

Input
F: objective-function
Input-output
$clusters$: cluster-set
$unclustered$: sample-set

```
newly-clustered := create-set(type: sample, values: empty)
N := count(values: samples-of(clusters)) + size(unclustered)
critical-distance := calculate-critical-distance(sample-count: N)
clustered-samples := create-set(type: sample, values: empty)
for all cluster: cluster in clusters do
    for all sample: sample in cluster do
        add(value: sample, to: clustered-samples)
    end for
end for
for all sample: cs in clustered-samples do
    for all sample: us in unclustered do
        if distance(from: us, to: cs, type: ∞-norm) ≤ critical-distance and F(cs) < F(us) then
            move(value: us, from: unclustered, to: newly-clustered)
            add(value: us, to: cluster-of(cs))
        end if
    end for
end for
while size(newly-clustered) > 0 do
    buffer := create-set(type: sample, values: empty)
    for all sample: us in unclustered do
        for all sample: cs in newly-clustered do
            if distance(from: us, to: cs, type: ∞-norm) ≤ critical-distance
              and F(cs) < F(us) then
                move(value: us, from: unclustered, to: buffer)
                add(value: us, to: cluster-of(value: cs)
            end if
        end for
    end for
    newly-clustered := buffer
end while
return clusters, unclustered
```

9. Algoritmus. GLOBALJ

Input

F: objective-function

a: vector

b: vector

$termination$: criteria

$clusterizer$: module

$local\text{-}search$: module

Output

$optimum\text{-}point$: sample

$optimum\text{-}value$: float

1: $optimum\text{-}value := \infty$, $optimum\text{-}point := null$, $N := 100$, $\lambda := 0.5$
2: $search\text{-}space :=$ create-distribution(type: $uniform$, values: $[a, b]$)
3: $reduced :=$ create-list(type: $sample$, values: $empty$)
4: $clusters :=$ create-list(type: $cluster$, values: $empty$)
5: $unclustered :=$ create-list(type: $sample$, values: $empty$)
6: **while** evaluate(condition: $termination$) $= false$ **do**
7: $new :=$ create-list(type: $sample$, values: generate-samples(from: $search\text{-}space$, count: N))
8: add(values: new, to: $reduced$)
9: sort(values: $reduced$, by: F, order: $descending$)
10: remove(from: $reduced$, range: create-range(first: 1, last: $[i \cdot N \cdot \lambda]$))
11: add(values: select(from: $reduced$, holds: in(container: new)), to: $unclustered$)
12: $clusters$, $unclustered :=$ clusterizer.cluster(objective-function: F, unclustered: $unclustered$, clusters: $clusters$)
13: **while** size($unclustered$) > 0 **do**
14: $x :=$ select(from: $unclustered$, index: 1)
15: $x^* :=$ local-search.optimize(function: F, start: x, over: $[a, b]$)
16: **if** $F(x^*) < optimum\text{-}value$ **then**
17: $optimum\text{-}point := x^*$, $optimum\text{-}value := F(x^*)$
18: **end if**
19: $clusters$, $unclustered :=$ clusterizer.cluster(objective-function: F, unclustered: $\{x^*, x\}$, clusters: $clusters$)
20: **if** cluster-of(x^*) $= null$ **then**
21: $cluster :=$ create-cluster(type: $sample$, values: $\{x^*, x\}$)
22: add(value: $cluster$, to: $clusters$)
23: **end if**
24: $clusters$, $unclustered :=$ clusterizer.cluster(objective-function: F, unclustered: $unclustered$, clusters: $clusters$)
25: **end while**
26: **end while**
27: **return** $optimum\text{-}point$, $optimum\text{-}value$

10. Algoritmus. PGLOBAL

Input

F: $\mathbb{R}^n \to \mathbb{R}$; $a, b \in \mathbb{R}^n$: lower and upper bounds; N: number of worker threads; $maxSampleSize$: maximum number of generated samples; $newSampleSize$: number of samples generated for every iteration; $reducedSampleSize$: number of best samples chosen from the new samples; $batch\ size$: number of samples forwarded to local search after clustering (if 0 use the number of currently free threads)

Output

$optimum$: best local optimum point found

$samples$, $unclustered$, $origins \leftarrow \{\}$
$optimum \leftarrow$ maximum value
start $N - 1$ **new threads**
while true **do**
  if check stopping criteria **then**
    break
  else if $origins$ **is not** empty **then**
    $origin \leftarrow$ **remove** from $origins$
    if $origin$ **is** null **then**
      continue
    end if
    $localopt \leftarrow$ **local search** over F from $origin$ within $[a, b]$
    $optimum \leftarrow$ **minimum** of $\{optimum, localopt\}$
    call clusterize optimum ($origin$, $localopt$)
  else if check iteration count stopping criteria **then**
    break
  else if clusterizer **is** active **then**
    call clustering samples ($critical\ distance$)
    if $this$ **is** last clustering thread **then**
      $origins \leftarrow$ **remove** $batch\ size$ from $unclustered$
      if $|unclustered| = 0$ **then**
        set clusterizer to inactive
        increase iteration count
      end if
    end if
    wait until all clustering threads reach this point
  else if clusterizer **is** inactive **and** $|samples| \geq newSampleSize$ **then**
    lock $samples$
    $samples \leftarrow$ **sort** $samples$ by ascending order regarding F
    $unclustered \leftarrow$ **remove** $[1, ..., reducedSampleSize]$ **element** from $samples$
    update critical distance
    set clusterizer to active
    unlock $samples$
  else if check sample count stopping criteria **then**
    break
  else
    lock $samples$
    $samples \leftarrow samples \cup$ **generate** a new sample from $[a, b]$ distributed uniformly
    $optimum \leftarrow$ **minimum** of $\{optimum, new\ sample\}$
    unlock $samples$
  end if
end while
return

11. Algoritmus. Clustering samples

Input

$critical\ distance$: single linkage distance threshold

State before

$clustered$: previously clustered samples

$unclustered$: new samples to be clustered

State after

$clustered$: $clustered \cup$ new clustered

$unclustered$: $unclustered \setminus clustered$

while clusterizer **is** active **do**
  $sample \leftarrow$ **remove** from $unclustered$
  if $sample$ **is** null **then**
    return
  end if
  if $sample$ **is** fully examined **then**
    $sample \rightarrow$ **insert** into $unclustered$
    continue
  end if
  $insider \leftarrow$ **find** next element in $clustered$ which is not compared to sample
  $cluster \leftarrow$ null
  while $insider$ **is** not null **do**
    if $\|sample - insider\|_2 \leq critical\ distance$ **and** $sample\ value > insider\ value$ **then**
      $cluster \leftarrow$ cluster of $insider$
      break
    else
      $insider \leftarrow$ **get** next element from $clustered$ which is not compared to sample
    end if
  end while
  lock all cluster modifications
  if $cluster$ **is** null **then**
    $sample \rightarrow$ **insert** into $unclustered$
  else
    $sample$, $cluster \rightarrow$ **insert** into $clustered$
    if center point of $cluster < sample$ **then**
      center point of $cluster \leftarrow sample$
    end if
  end if
  if all samples are examined **then**
    set clusterizer to clean up
  end if
  unlock all cluster modifications
end while
return

12. Algoritmus. Compare optimum to clusters

Input
$optimum$: clusterizable optimum point
$clusters$: previously created clusters
$critical\ distance$: single linkage distance threshold
Return value
$cluster$: the cluster which contains the optimum

$cluster \leftarrow$ **find** next element in $clusters$ which is not compared to optimum
while $cluster$ **is** not null **do**
  $center \leftarrow$ center point of $cluster$
  if $\|center - optimum\|_2 \leq critical\ distance/10$ **then**
    return $cluster$
  end if
  $cluster \leftarrow$ **find** next element in $clusters$ which is not compared to optimum
end while
return null

13. Algoritmus. Clusterize optimum

Input
$origin$: starting point of the local search which lead to optimum
$optimum$: optimum point to be clustered
State before
$clusters$: previously created clusters
$clustered$: previously clustered samples
$critical\ distance$: single linkage distance threshold
State after
$clusters$: $clusters \cup$ new clusters
$clustered$: $clustered \cup \{origin, optimum\}$
$critical\ distance$: updated single linkage distance threshold

$cluster \leftarrow$ **call** compare optimum to clusters ($optimum$, $clusters$, $critical\ distance$)
lock all cluster modifications
if $cluster$ **is** null **then**
  $cluster \leftarrow$ **call** compare optimum to clusters ($optimum$, $clusters$, $critical\ distance$)
  if $cluster$ **is** null **then**
    $cluster \leftarrow$ **new** cluster
    center point of $cluster \leftarrow optimum$
    $cluster \rightarrow$ **insert** into $clusters$
  end if
end if
$origin$, $cluster \rightarrow$ **insert** into $clustered$
$optimum$, $cluster \rightarrow$ **insert** into $clustered$
if center point of $cluster < sample$ **then**
  center point of $cluster \leftarrow origin$
end if
update critical distance
unlock all cluster modifications