A matematikai modellezés hatása nemlineáris optimalizálási feladatok megoldásának hatékonyságára

Ph.D. értekezés
Informatika Doktori Iskola

Dobjánné Antal Elvira

Témavezetők:
Dr. Csendes Tibor és Dr. Vinkó Tamás

Számítógépes Optimalizálás Tanszék
SZTE TTIK
2017
Szeged

TARTALOMJEGYZÉK

1

BEVEZETÉS

Az optimalizálási problémák matematikai modellezése során hozott döntések jelentős mértékben befolyásolhatják az alkalmazott megoldó hatékonyságát, a feladat komplexitását. Egy optimalizálási feladat átfogalmazása azonban sokszor az egyéni intuíción túlmutató automatikus módszerekkel is lehetséges. A lineáris programozási feladatok szimplex módszer számára legelőnyösebb formájának felírása például már az 1970-es években is foglalkoztatta a matematikusokat. Ezek a korai eredmények mára az AMPL előfeldolgozóba épülve szinte észrevétlenül hasznosulnak.

Számos kutatás foglalkozik napjainkban a (vegyes-)egészértékű programozási feladatok kedvezőbb, ill. adott körülmények között megoldható alakba történő átfogalmazásának lehetőségeivel, habár ezek az átalakítások általában bizonyos feltételek relaxációjával és a feladat dimenziójának növelésével járnak együtt. Meglepően csekély számú publikáció koncentrál azonban a nemlineáris optimalizálási feladatok ekvivalens átalakításait eredményező, automatizálható szimbolikus eljárásokra. Erre a területre fókuszál dolgozatom első fele.

A fellelhető szakirodalom elméleti eredményeinek ismeretében feltételezhető volt, hogy napjaink nagytudású számítógépes algebra rendszerei alkalmasak egy szimbolikus, tehát a számítási pontosságot garantáló, nemlineáris optimalizálási feladatok ekvivalens átírásait megadó alkalmazás létrehozására, amely lehetőséget ad a korábban publikált algoritmusok integrálására, tesztelésére, továbbá inspirációt nyújthat ezek továbbfejlesztéséhez.

A fenti céllal elkészült egy, a feltétel nélküli nemlineáris optimalizálási feladatokon nemlineáris koordináta-transzformációkat végrehajtó Maple program. Alapos tesztelés során felderítettem azokat a specifikus területeket, amelyek egy ilyen algoritmus implementálása során kritikusak lehetnek, és megállapítottam, hogy a Maple rendszer néhány dokumentálatlan hibája és általános felhasználásra tervezett, a konkrét célnak nem megfelelő minőségben megvalósított funkciója komoly akadályokat állít a bővítés elé.

Tanulmányoztam a rendelkezésre álló kereskedelmi és szabad szoftveres alternatívákat, és a programozó számára nyújtott rugalmassága, erős fejlesztői háttere és a legújabb matematikai eredményeket felvonultató saját, széles körű funkcionalitása miatt a Mathematica programot találtam a legjobb alapnak a további fejlesztések számára. Ez alapján korábbi programunkat további funkciókkal bővítve átültettem Mathematica alá, és empirikus úton bizonyítottam, hogy előfeldolgozóként alkalmazva az általunk javasolt szimbolikus transzformációk hasznosak egy klasszikus (numerikus) heurisztikus megoldó számára. Új elméleti eredmények születtek a párhuzamosan végrehajtható nemlineáris koordináta-transzformációkkal és a feladathoz adott feltételekkel kapcsolatban, továbbá készítettem egy online demonstrációs oldalt az eljárás bemutatására.

Az értekezés második felében egy konkrét feladathoz, a közösségi tartalommegosztó rendszerekben felmerülő méltányos (max-min fair) sávszélesség-kiosztás problémájához kapcsolódó modellezési kérdések kerülnek terítékre. A feladatra sikerült egy új, egzakt matematikai programozási modellt adnom, és az ezen alapuló algoritmus helyességét elméleti úton bizonyítanom. Elkészült az algoritmus AMPL-es megvalósítása, aminek a teljesítményét numerikus tesztek révén összehasonlítottam a feladat megoldására korábban létező programmal. Az eredmény több tekintetben pozitív.

Az implementáció során, részben az elméleti eredményekhez kapcsolódóan, számos modellezési ötlet merült fel, ezért egy külön fejezet foglalkozik ezen technikák hatáselemzésével. A max-min méltányos erőforrás-kiosztást előállító modell tizenkét változatát hasonlítottam össze egy kiterjedt numerikus tesztelés során, két professzionális megoldó és huszonhét nagyméretű tesztfeladat bevonásával.

Az értekezés felépítése a következő. *A 2. fejezet* általános bevezetést nyújt a dolgozat témájához. Bemutatja az optimalizálás alapfogalmait, a különféle feladatosztályokat, továbbá az AMPL példáján a modellezési nyelvek használatát, szerepét.

A 3. fejezetben bemutatom a Csendes és Rapcsák elméleti eredménye alapján [20, 71] kidolgozott automatikus szimbolikus egyszerűsítő eljárást, ami a feltétel nélküli nemlineáris optimalizálási feladat nemlineáris koordináta-traszformációit állítja elő. Standard és egyéb gyakran használt globális optimalizálási tesztfeladatokon, valamint az erre a célra készült saját példákon mutatom be és hasonlítom össze a Maple és Mathematica rendszerekben elkészült változatokat. A fejezet végén közlöm a kapcsolódó saját elméleti eredményeket is.

A 4. fejezetben azt vizsgálom, hogy a szimbolikus egyszerűsítő előfeldolgozóként alkalmazva hasznosnak bizonyul-e egy klasszikus (numerikus) heurisztikus megoldó számára.

Az 5. fejezetben a BitTorrent közösségekben felmerülő méltányos sávszélesség-kiosztás problémájára adott megoldásomat ismertettem. Bemutatom a megoldásra javasolt egzakt matematikai programot, és az erre épülő algoritmust. Az algoritmus helyességét elméleti úton bizonyítom.

A 6. fejezetben közlöm a méltányos sávszélesség-kiosztás kiszámítására javasolt modell lehetséges változatainak egy bőséges listáját, valamint megvizsgálom, hogy a feladat megoldásának sebességét mennyiben befolyásolja különféle modellezési technikák alkalmazása.

Az értekezés az eredmények és továbbfejlesztési lehetőségek összefoglalásával zárul.

2

AZ OPTIMALIZÁLÁSRÓL ÁLTALÁBAN

Ebben a fejezetben az optimalizálás alapfogalmaival foglalkozunk, a teljesség igénye nélkül. Elsősorban az értekezésben később felhasznált fogalmakat tekintjük át. A formális definíciót szinte minden esetben mellőzöm, ezek a téma szak- és tankönyveiben megtalálhatók (ld. például [6, 46, 86, 28]).

2.1. Az optimalizálás célja és alapfogalmai

Az optimalizálás, más néven matematikai programozás az operációkutatás (angolul operations vagy operational research (OR), ill. management science) részterülete. Mint ilyen, célja a matematikai módszerekkel történő döntéstámogatás. Bradley és szerzőtársai szerint [10] a matematikai programozás korlátozott erőforrásoknak versengő tevékenységek közötti optimális elosztásával foglalkozik oly módon, hogy mindeközben a tanulmányozott probléma természetéből fakadó feltételek egy halmazát is kielégíti. Krekó szerint [48] feltételezhetjük, hogy optimalizálási modelljeink megfogalmazása valamilyen tevékenység „célszerű lebonyolítására irányul."

Miután meghatároztuk a modellben figyelembe veendő elemi tevékenységeket [6, 48], ezek intenzitását az

$$x_1, x_2, \ldots, x_n$$

döntési változókkal jelöljük. Az optimalizálás alapfeltevése, hogy a vizsgált probléma egyes erőforrás-allokációi ezen változók függvényében megfogalmazhatók, továbbá valamilyen szempontrendszer szerint *mérhetők* és *összehasonlíthatók*. A mérés eredményét egy függvény írja le, mely az $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ vektorhoz általában egy valós számot rendel. Ez az úgynevezett *célfüggvény*. A célfüggvény értelmezési tartományát *keresési térnek* is nevezik. Értékét jellemzően maximalizálni (ha valamilyen hasznosságot fejez ki), vagy minimalizálni szeretnénk (például ha költséget ír le).

A figyelembe venni kívánt feltételek meghatározzák a *lehetséges megoldások halmazát*, L-et [6]. Feladatunk legtöbbször L elemei közül az (egyik) optimális célfüggvény-értékű megtalálása. A lehetséges megoldást fízibilis (feasible), vagy megengedett megoldásnak [46], esetleg lehetséges programnak [48] is szokás nevezni.

A problémát röviden így foglalhatjuk össze:

$$\min_{x \in L} f(x),$$

ahol $f(x)$ a minimalizálandó célfüggvény. Az f minimális értéke (minimuma) mellett legalább ugyanilyen fontos, hogy azt mely x^* helyen, helyeken veszi fel. Vagyis a feladat optimumán kívül kíváncsiak vagyunk az optimumhely(ek)re, más néven az optimális megoldás(ok)ra is.

Az L-et az elemi tevékenységek lehetséges intenzitásértékei (nevezzük ezt a változó típusának), és különféle egyenlőség vagy egyenlőtlenség alakban felírható *feltételek* (constraints) határozzák meg:

$$\begin{aligned} & \min \ f(x), && && (2.1)\\ \text{feltéve, hogy} \quad & c_i(x) = 0, && i = 1, \ldots, p_1 \\ & c_j(x) \leqslant 0, && j = p_1 + 1, \ldots, p_2 \\ & c_l(x) < 0, && l = p_2 + 1, \ldots, p_3 \\ & x \in X. \end{aligned}$$

Az X lehet például $\mathbb{R}^n$, $\mathbb{Z}^n$, vagy a $\{0,1\}^n$ halmaz. Lehetséges, hogy egyes változók típusa különböző, szerepelhetnek egymás mellett a modellben például valós és bináris változók.

Meg szokás különböztetni az egyes változókra vonatkozó, $x_i \leqslant k$ alakú feltételeket, ahol $1 \leqslant i \leqslant n$ és k egy konstans. Ezeket *korlátoknak* (bounds) nevezzük. Ha egy valós változóra alsó- és felső korlát is adott a feladatban, azt intervallum korlátnak (interval bound) [28] is nevezik.

Meg kell említenünk, hogy bármely célfüggvény átírható minimalizálásra, a feltételek jobb oldala 0-ra, a relációk $\leqslant$ vagy $<$ alakra.

Az analízis terminológiája szerint (2.1) egy feltételes szélsőérték feladat.

2.2. Az optimalizálási modellek osztályozása

Az optimalizálási modelleket többféle szempont szerint *osztályozhatjuk*. A változók típusa alapján lehet:

- *folytonos*, ha minden változó egy (nem szükségképpen véges) intervallum bármely értékét felveheti,
- *egészértékű*, ha minden változó diszkrét, esetleg
- *vegyes-egészértékű*, ha folytonos és diszkrét változók is találhatók a modellben.

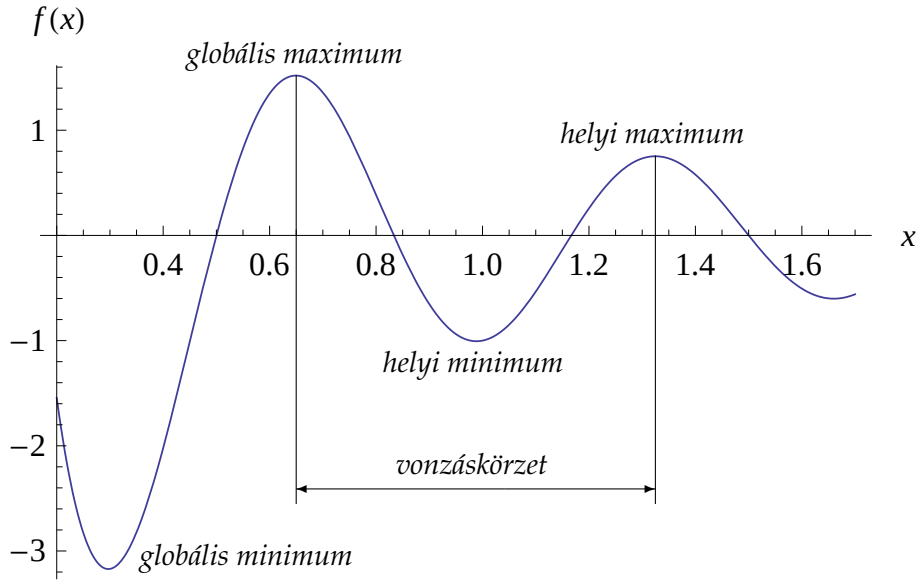
A modellben szereplő paraméterek alapján:

- *determinisztikus*, ha a paraméterek mindegyike pontosan meghatározható konstans, vagy
- *sztochasztikus*, ha a paraméterek között ismert eloszlással rendelkező valószínűségi változó is szerepel. (Jelen értekezés nem foglalkozik sztochasztikus modellekkel.)

A célfüggvény és a feltételeket leíró függvények alapján a feladatunk

- *lineáris programozás*, ha a célfüggvény és minden feltétel lineáris a változókban, ill.
- *nemlineáris programozás*, ha a célfüggvény, vagy a feltételek között van legalább egy nemlineáris. Ezen belül a lehetséges megoldások halmaza, és a célfüggvény szintvonalainak alakja alapján beszélhetünk
  - *konvex* (ez a könnyebb), vagy
  - *konkáv* programozási feladatról.

  Ezen feladatok kapcsán érdemes megkülönböztetnünk a helyi (lokális) és globális (abszolút) optimumokat, ld. a 2.1. ábrát. A nemlineáris optimalizálás módszerei egy lokális optimum megtalálására irányulnak, azonban nem mondják meg, hogy az egyben globális-e. A konvex feladatok esetében minden lokális optimum egyben globális is.
- *Globális optimalizálás*ról akkor beszélünk, ha egy konkáv nemlineáris feladat globális optimumát keressük, miközben esetleg a feladatot leíró függvények képlete sem ismert (ez az úgynevezett feketedoboz-probléma).



2.1. ábra. *Az* $f(x) = \cos(3\pi x)/x$ *függvény az* $x = [0{,}2; 1{,}7]$ *valós intervallumon*

2.3. Megoldó módszerek

2.3.1. Lináris programozás (Linear Programming, LP)

A lineáris programozás feladata felírható például a következő *standard alakban*:

$$\begin{aligned} \min\ & \mathbf{c}\,\mathbf{x}, \\ \text{f.h.}\quad & A\,\mathbf{x} \leqslant \mathbf{b}, \\ & \mathbf{x} \geqslant \mathbf{0}, \\ & \mathbf{x} \in \mathbb{R}^n, \end{aligned}$$

ahol A együttható-mátrix, valamint b és c vektorok csak konstansokat tartalmaznak.

A nem szigorú feltételek és korlátok gondoskodnak arról, hogy a lehetséges megoldások halmaza zárt legyen, *Weierstrass tétele* értelmében pedig egy folytonos függvény egy $X \subset \mathbb{R}^n$ nemüres, korlátos és zárt halmazon felveszi abszolút minimumát és maximumát [49]. Ezzel összhangban a *trichotómiatétel* [87] értelmében az LP feladatok háromféle osztályba sorolhatók. Minden lineáris programozási feladat esetében vagy van a feladatnak optimális megoldása, vagy nincs lehetséges megoldása (infízibilis), vagy a célfüggvény nemkorlátos.

Korábbi eredményekre építve 1947-ben dolgozta ki a feladat megoldására szolgáló szimplex algoritmust [10, 77, 58] az akkoriban az Amerikai Egyesült Államok Légiereje szolgálatában álló George B. Dantzig, amelyet azonban csak 1951-ben publikált. Mivel sok gyakorlati feladat megfogalmazható LP-ként, és a tapasztalatok szerint igen gyors megoldást tett lehetővé, a szimplex algoritmus hamarosan széles körben elterjedt.

Sokáig nem volt ismert, hogy az LP feladat polinomiális időben megoldható-e (a szimplex algoritmus ugyanis a legrosszabb esetben exponenciális), mígnem Hacsiján 1979-es algoritmusa eldöntötte a kérdést [77, 55]. Karmarkar 1984-ben publikált algoritmusa óta úgy elméletben, mint gyakorlatban hatékony, polinomiális időben végrehajtható eljárásaink vannak az LP megoldására. Ennek ellenére nagy feladatok esetén nélkülözhetetlen segítséget nyújtanak az előfeldolgozó, előmegoldó módszerek [58, 62].

2.3.2. Nemlineáris programozás (NonLinear Programming, NLP)

A nemlineáris programozás módszerei elsősorban a függvényanalízis eredményeire építenek. A feltétel nélküli feladat lokális optimumára vonatkozó szükséges feltétel, hogy az optimális megoldásban a gradiens 0 (fordítva nem igaz). A másodikderivált-próba értelmében, ha egy kétszer differenciálható f függvény gradiense egy $x \in \mathbb{R}^n$ pontban 0, és Hesse-mátrixa ugyanott pozitív (negatív) definit, akkor f-nek x-ben szigorú lokális minimuma (maximuma) van. Ez elégséges feltétel. Továbbá ha f-nek x-ben lokális minimuma (maximuma) van, akkor a Hesse-mátrix x-ben vett helyettesítési értéke pozitív (negatív) szemidefinit. Tehát a másodikderivált próba nem dönti el, hogy f-nek van-e szélsőértéke x-ben, ha a Hesse-mátrix x-ben szemidefinit, de nem definit [49, 28].

Feltételes optimalizálási feladat optimumának megtalálására bizonyos feltételek mellett alkalmazható a Lagrange-féle multiplikátor-módszer, illetve a Karush-Kuhn-Tucker feltételek.

2.3.3. Globális optimalizálás

A globális optimalizálás módszereit osztályozhatjuk például a rendelkezésre álló információ alapján:

- A *direkt kereső* módszerek csak a célfüggvény helyettesítési értékét használják fel. Sok népszerű heurisztika ide sorolható, úgymint a hegymászó algoritmus, szimulált hűtés, különféle evolúciós algoritmusok.

- Az *elsőrendű (gradiens)* módszer a helyettesítési értéken felül a célfüggvény első (parciális) deriváltjait is ismeri, vagy közelíti.

- A *másodrendű (Newton)* módszer a fentieken túl a második derivált értékét is felhasználhatja.

A nemlineáris optimalizálás első- vagy másodrendű módszereit szokás a globális optimalizálásban úgynevezett helyi keresőként alkalmazni. A keresési tér azon részét, amelyen belül egy tetszőleges helyi keresőt elindítva az $\mathbf{x}^*$ optimális megoldást kapjuk eredményül, az $\mathbf{x}^*$ vonzáskörzetének nevezzük (ld. az $x = 1$ vonzáskörzetét a 2.1. ábrán).

A megoldás minősége alapján beszélhetünk:

- *nemteljes* (esetleg nem a globális optimumhoz konvergáló),

- *aszimptotikusan teljes* (végtelen futás esetén a globális optimumot biztosan elérő),

- *teljes* (aszimptotikusan teljes, és egzakt aritmetikát feltételezve véges lépést követően a globális optimum közelítő megoldásának megtalálását eldönteni tudó), valamint

- *szigorúan megbízható* (a kerekítési hibák mellett is teljes) módszerekről [28].

Napjainkban a globális optimalizáló módszerek széles köre elérhető különféle programozási nyelveken megvalósítva. Ezek egyike a 4. fejezetben alkalmazott GLOBAL, ami egy heurisztikus multi-start megoldó.

Működése során két fázis váltakozik: az adaptív globális keresés és a helyi keresés. A globális keresés kezdetén direkt kereső módjára kiértékeli a keresési térből egyenletes eloszlással, véletlenszerűen választott pontokat. A második fázisban a legkedvezőbb pontokból egy-egy helyi keresőt indít, és a megtalált helyi minimumokat egy listába gyűjti. A helyi kereső választható, esetemben a BFGS nevű kvázi-Newton módszer volt. Az adaptív jelző arra vonatkozik, hogy a globális kereső a helyi kereső által talált kedvező pontok közelében nagyobb valószínűséggel választ ki új pontokat. Megállási feltételként beállítható a maximálisan végzendő iterációk száma és a megoldás javulására vonatkozó tolerancia.

2.3.4. Egészértékű programozás (Integer Programming, IP)

Már a lineáris egészértékű programozási (ILP) feladat is NP-teljes. A 0–1 értékű bináris változókkal felírt modellek szintén [55].

Ehhez a feladatosztályhoz kapcsolódóan csak két igen fontos megoldási elvet emelünk ki [86]:

- *Relaxáció.* Az egészértékűségi feltételek relaxációja, vagyis ideiglenes figyelmen kívül hagyása. Ha az így kapott relaxált feladat optimális megoldása véletlenül egészértékű, akkor szerencsénk van, különben megfontolást igényel, hogyan alakítsuk a folytonos optimumhelyet egy, az optimálishoz közel álló lehetséges egészértékű megoldásra.

- A *korlátozás és szétválasztás* (branch & bound) az egészértékű programozás egyik legelterjedtebb módszere. Tulajdonképpen egy tág keretrendszerről van szó, amelynek rengetegféle specializációja és megvalósítása lehetséges. A „szétválasztás" azt jelenti, hogy a lehetséges megoldások halmazának darabolásával részfeladatokat állítunk elő, és ezeket vizsgáljuk. A „korlátozás" pedig arra utal, hogy a feladat relaxált változatának optimumát az eredeti feladat optimumára vonatkozó korlátnak tekinthetjük. A részfeladatokra előálló korlátok alapján vághatunk is a kiszámítási fában, bizonyos részfeladatok bizonyíthatóan nem tartalmazzák az optimumot, ezért elhagyhatók.

Ha a feladatban folytonos és egészértékű változók is vannak, akkor vegyes-egészértékű (MIP) problémáról beszélünk.

2.4. Az algebrai modellezési nyelvekről

Bonyolultabb feladatoknál általában nem mondható meg előre, melyik megoldó módszer fogja az optimumot gyorsabban, pontosabban kiszámolni. Ráadásul az operációkutatás döntéstámogatásra vonatkozó céljából adódóan nem csupán a számítástudományban jártas személyek alkalmazzák az optimalizálás eszközeit. A számítástechnika rohamos fejlődésének köszönhetően ugyanakkor napról napra növekszik a még kezelhető problémák mérete.

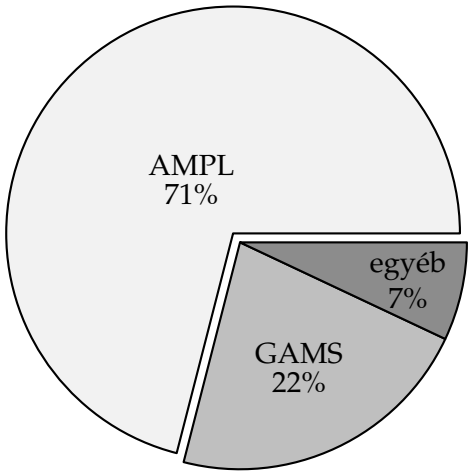
Mindezen tényezők eredőjeként felmerült az igény egy további absztrakciós réteg kialakítására a felhasználók és a megoldó (solver) programok között. Ezt a szerepet töltik be az algebrai modellezési nyelvek. Több funkciót egyesítenek magukban, úgymint:

- az optimalizálási feladat leíró nyelve,

- adatfeldolgozó réteg,
- a modell manipulációjának eszköze, ill.
- a professzionális megoldók felé nyújtott interfész.

Ilyen algebrai modellezési nyelv az AMPL, GAMS, AIMMS, stb.

Az értekezés 5. és 6. fejezetének munkálatai során az AMPL nyelvet [27] használtam, ezért erről ejtenék a továbbiakban néhány szót. Az AMPL az egyik legnépszerűbb algebrai modellezési nyelv napjainkban. A több, mint 60 élvonalbeli solverhez ingyenes online hozzáférést biztosító NEOS szerver statisztikái szerint az elmúlt öt év összes hívásának 71%-a, összesen mintegy 3,5 millió hívás érkezett oda AMPL nyelven (ld. a 2.2. ábrát).



2.2. ábra. *A NEOS szerverre 2012. január 1. és 2016. december 31. között érkezett 4 993 227 feladat leíró nyelvének megoszlása [64]*

Segítségével az optimalizálási problémát a matematikai jelöléshez hasonló formában fogalmazhatjuk meg. Az AMPL feldolgozó a feladat leírását és további információkat (első- és másodrendű deriváltak, stb.) a solver interfészen keresztül számos megoldó számára képes egységes formában továbbítani, úgymint a CPLEX, MINOS, LANCELOT, vagy egyéb jól ismert kereskedelmi szoftverek.

Számunkra fontos lesz, hogy az AMPL rendelkezik egy „presolving"-nak nevezett előfeldolgozóval. Ez a problémának a kiválasztott megoldó felé történő továbbítása előtt hívódik meg. Kedvező esetben már az előfeldolgozó felismeri a modell infízibilitását, vagy képes csökkenteni a feladat méretét. Ez a kommunikációs költség csökkentését is eredményezheti, de főleg azon megoldók használóira gondoltak vele a készítők, amelyeknek nincs saját előfeldolgozójuk.

3

NEMLINEÁRIS ÁTÍRÁSOK NEMLINEÁRIS OPTIMALIZÁLÁSI PROBLÉMÁK EGYSZERŰSÍTÉSÉRE

A matematikai program felírására vonatkozó modellezési döntések szignifikánsan befolyásolhatják az adott probléma bonyolultságát, ill. az alkalmazott megoldó módszerek hatékonyságát. Ez a széles körben elfogadott állítás több tanulmányt ihletett, például az egészértékű programozás területén, melyek célja egy optimalizálási feladat bizonyos szempontból könnyebben megoldható alakját eredményező átírás megadása. Ezek az átalakítások általában néhány feltétel relaxálásával és a változók számának növekedésével járnak együtt.

Mi a nemlineáris optimalizálási feladatok átírása területén vizsgáljuk a szimbolikus technikák alkalmazhatóságát. Pontosabban nemlineáris optimalizálási feladatokra alkalmazható, szimbolikus átalakítások révén előálló lehetséges egyszerűsítéseket keresünk, különös tekintettel az automatikusan fellelhető, a probléma ekvivalens alakját eredményező, lehetőleg a feladat dimenzióját csökkentő átírásokra. Csendes és Rapcsák [20, 71] megmutatta, hogy lehetséges a feltétel nélküli nemlineáris célfüggvényt nemlineáris koordináta-traszformációkkal egyszerűsíteni, pontosabban elégséges feltételeket adtak ilyen transzformációk tulajdonságaira. Ez a legtöbb esetben redundáns részkifejezések szimbolikus helyettesítését jelentette, arra számítva, hogy ezáltal a kiszámítás kevesebb művelettel lesz végrehajtható, miközben az egyszerűsített feladat az eredetivel ekvivalens marad abban az értelemben, hogy a két alak megoldásai ismert módon egymásba konvertálhatók lesznek.

Napjaink számítógépes algebra rendszereinek fejlettségi szintje és népszerűsége arra sarkallt, hogy Csendes és Rapcsák elméleti eredményét a

megvalósíthatóság szemszögéből vizsgáljam. Két modern szimbolikus programozási környezetben készült el a hivatkozott elméleti eljárás megvalósítása, miközben a lehetséges hibaforrások és továbbfejlesztési lehetőségek is nagyító alá kerültek.

A tesztelést az eredeti publikáció példáin, valamint saját, erre a célra készült tesztfeladatokon és standard globális optimalizálási tesztfeladatok egy halmazán végeztem.

A fejezet végén új elméleti eredményeim kerülnek bemutatásra, amelyek a korábbinál bonyolultabb ekvivalens átírásokat írnak le. Ezek gyakorlati hasznosításához, vagyis a hatékony számítógépes implementációhoz azonban további elméleti alapozás szükséges.

3.1. Bevezetés

Tekintsük a következő feltétel nélküli nemlineáris optimalizálási problémát:

$$\min_{x \in \mathbb{R}^n} f(x), \tag{3.1}$$

ahol $f(x) : \mathbb{R}^n \to \mathbb{R}$ egy nemlineáris, kétszer folytonosan differenciálható függvény, amely egy szimbolikus kifejezéssel, vagyis képlettel adott. Célunk egy ekvivalens probléma-alak előállítása:

$$\min_{y \in \mathbb{R}^m} g(y), \tag{3.2}$$

ahol $g(y) : \mathbb{R}^m \to \mathbb{R}$ egyszerűbb, mint $f(x)$. A célfüggvényt egyszerűbbnek nevezünk, ha optimuma a korábbinál egyszerűbben meghatározható valamely módszer számára, miközben az y^* és x^* által jelölt optimális megoldások egyértelmű transzformációval egymásba átalakíthatóak.

Az ilyen jellegű átírások általában a modellező invenciójából születnek, azonban nem ez az egyetlen lehetőség. A következő szakaszban bemutatok néhány technikát, amelyek optimalizálási feladatok automatikus manipulálását végzik a megoldó program (solver) hatékonyságának növelése céljából. A szimbolikus technikák ilyen jellegű felhasználása egyelőre a lineáris és (vegyes-)egészértékű programozásban terjedtek el jobban.

A 3.3. szakasz bemutatja a [20] publikációban leírt nemlineáris koordinátatraszformációk alapján általunk készített átírási módszert, amelyet hatástényezős folyóiratcikkben közöltünk [4]. A megvalósítást először a Maple, majd a Mathematica nyelvén készítettem el, ennek részletei a 3.4. szakaszban találhatók. A 3.4.1. alszakaszban bemutatom az implementációk kiértékelése

során alkalmazott tesztfeladatokat. A 3.4.6. alszakasz a kiértékelés eredményeit tartalmazza a Maple implementációra, míg a 3.4.8. alszakaszban a két megvalósítás összehasonlítását közlöm. Végül, a 3.5. szakaszban az elméletet két irányban általánosítom: párhuzamosan végrehajtható helyettesítéseket írok le, valamint feltételeket adok a modellhez [3].

3.2. Kapcsolódó munkák

A nemlineáris optimalizálási problémák megoldása klasszikusan numerikus algoritmusokkal történik, azonban a szimbolikus számítások módszertanának fejlődése, és ehhez kapcsolódóan a forgalomban lévő hathatós számítógépes algebra rendszerek (computer algebra systems, CAS) lehetővé teszik hatékony szimbolikus technikák alkalmazását ezen a területen is. Ennek megfelelően egyre nagyobb figyelmet szentelnek a kutatók a számítógépes algebrának [29] a matematikai programozásba történő bevonására.

A szimbolikus technikák integrálása kétféleképpen képzelhető el. Egyrészt, a számítógépes algebra segítségünkre lehet *a modellezési fázisban*, alternatív matematikai modellek szimbolikus transzformációkkal történő előállítása révén. A szerző [4, 3] cikkekben publikált munkája erre az irányra koncentrál. Az optimális szabályozás [76, 7, 44], a gépészeti tervezés [15], ill. a folyamatszintézis [70, 25] területén is láthatunk példát hasonló alkalmazásokra, azonban ott a szimbolikus módszerekkel generált modellek különböző rendszereket írnak le, és ezek elemzése, összehasonlítása történik az optimalizálás numerikus módszereivel.

Másrészt, a modern nemlineáris megoldók egyre több információt használnak a feladat struktúrájáról az optimalizálási folyamat közben, ezáltal *hibrid, szimbolikus-numerikus megoldókat* eredményezve [32, 12, 52, 96, 43].

A Gröbner bázis elmélet, a kvantor elimináció (quantifier elimination, QE) és más algebrai technikák használhatóak különleges esetekben optimalizálási problémák átírására és megoldására is. Kanno [45] *szimbolikus optimalizálás*t javasol jelfeldolgozási problémák egy osztályára, vagyis szimbolikus számítások használatát a feladat megoldásának teljes kiszámítása során. Ez nem feltétlenül a leghatékonyabb út, hiszen a szimbolikus módszerek általában lassabbak [44], habár pontosabb eredményt szolgáltatnak numerikus társaiknál. A hivatkozott elmélet csak algebrai függvényekre elérhető.

A továbbiakban a legszorosabban kapcsolódó átírási technikákról ejtek

szót. Ilyen az AMPL feldolgozó „presolving" (a továbbiakban: előmegoldó) mechanizmusa [26, 30], az LP előfeldolgozás [62], ill. a relaxációval történő IP/MINLP átírás [53].

Az AMPL előmegoldója Brearley [11] módszerén alapszik, és csak lineáris célfüggvényre és feltételekre alkalmazható. Ugyanakkor bizonyos esetekben nemlineáris elemek értékét is meg tudja határozni egy függvényben, amennyiben a változó értéke egyéb feltételekből következik. Az AMPL szimbolikus manipulációiról bővebben a [30] cikkben olvashatunk.

A lineáris programozásban széles körben ismert, hogy különféle előfeldolgozó módszerek alkalmazása a megoldó fázis előtt szignifikánsan növelheti a szimplex és a belső pontos módszerek hatékonyságát is. Az LP előfeldolgozás lehetséges lépései lehetnek:

- bizonyos változók értékének rögzítése, ill. redundáns feltételek kiküszöbölése primál/duál fízibilitási tesztek által,
- a lineárisan függő sorok felismerése, korlátok megszorítása és redundáns feltételek vagy változók számának csökkentése eliminációval,
- a feltételi mátrix (A) redukálása a lehető legritkább alakra.

Az LP átírási technikák rövid összefoglalása és egy összehasonlítás a nagy bonyolultságú problémák előfeldolgozása tekintetében megtalálható Mészárosnál [62].

Mivel az *egészértékű programozási feladatok* megoldása lehetséges a feladat egy alkalmas LP-re való relaxációján keresztül is, az LP feladat praktikus átírása hasznos lehet az IP területén is. Egy IP-nek LP-re való átírása önmagában egy érdekes alkalmazása a szimbolikus átírásoknak az optimalizálás területén. A lehetséges automatikus átírások az IP területén megengedik a feltételek relaxációját és akár a változók számának növelését annak érdekében, hogy az egyébként reménytelenül bonyolult feladatnak egy – egyáltalán – megoldható formáját produkálják.

A ROSE nevű szabadon elérhető, fejlesztői stádiumban lévő programcsomag az IP és MINLP területén hasznos átírásokat gyűjti [53].

A mi megközelítésünk a fentiektől eltér. Nem akarjuk az egész feladatot algebrai úton megoldani (mivel ez elég lassú lehet és nem is minden esetben kivitelezhető). Csupán a feladatot szeretnénk a megoldó számára egyszerűbbé tenni. Csak ekvivalens átalakítást eredményező mechanizmusokban

vagyunk érdekeltek, amik lehetőség szerint csökkentik a feladat dimenzióját. Az általunk vizsgált ekvivalens átírások kapcsán nem merül fel relaxáció.

3.3. Nemlineáris koordináta-transzformációk

Amint Csendes és Rapcsák [20] megmutatta, a feltétel nélküli nemlineáris célfüggvény bizonyos esetekben egyszerűsíthető nemlineáris koordináta-transzformációk segítségével. Ez túlnyomórészt redundáns részkifejezéseknek a szimbolikus helyettesítését jelenti annak reményében, hogy a megoldó számítás-igényét csökkentjük, miközben az egyszerűsített feladat az eredetivel ekvivalens marad abban az értelemben, hogy egy közvetlen átalakítás lehetséges a két alak megoldása között.

Ez a szakasz bemutatja a módszer elméleti hátterét, a 3.4. szakasz pedig az általam megvalósított, nemlineáris koordináta transzformációkon alapuló automatikus egyszerűsítő programot.

3.3.1. Motiváció: egy paraméter becslési probléma

Tekintsük a Hantos és szerzőtársai [38] által bemutatott paraméter becslési feladatot, egy négyzet-összeg alakú célfüggvényt:

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

ahol $Z_L(\omega_i) \in \mathbb{C}$ a mért impedancia, $Z'_L(\omega_i)$ a modellezett impedancia ω_i frekvencián $(i = 1, 2, \ldots, m)$, és R_{aw}, I_{aw}, B, valamint τ a modell paraméterek.

Az eredeti nemlineáris modell függvény kézenfekvő fizikai paramétereken alapszik:

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

Itt $\gamma = 10^{1/4}$ és $\imath$ a képzetes egység. Csendes és Rapcsák tanulmányának [20] motivációja egy egyszerűbb ekvivalens modell függvény létezése, amely a modell paraméterekben lineáris:

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

Az alkalmazott sikeres változó helyettesítés az $A = B \log(\tau)$ volt, ami a feladatot nemlineárisról lineáris legkisebb négyzetessé tette. Ugyanakkor

bizonyítást nyert, hogy az eredeti feladat minden helyi minimum pontja egyúttal globális minimum.

Valószínűsíthető, hogy ilyen jellegű egyszerűsítés egyéb nemlineáris optimalizálási feladatok esetén is lehetséges. Vajon egy automatikus egyszerűsítő módszer elő tud ilyeneket állítani?

3.3.2. Elméleti háttér

Tekintsük újra a (3.1) feltétel nélküli nemlineáris optimalizálási problémát:

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}),$$

ahol $f(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ egy képlettel (vagyis egy szimbolikus kifejezéssel) adott sima függvény. A „kifejezés" itt szimbólumok (konstansok, változók, operátorok, függvénynevek és zárójelek) egy véges, szintaktikailag helyes kombinációját jelöli. A népszerű számítógépes algebra rendszerekben, mint például a Maple-ben [39] és a Mathematicában [89] minden kifejezést pointerek egymásba ágyazott listájával tárolnak, ami tulajdonképpen irányított körmentes gráfok (directed acyclic graph, DAG [75]) megvalósításának felel meg.

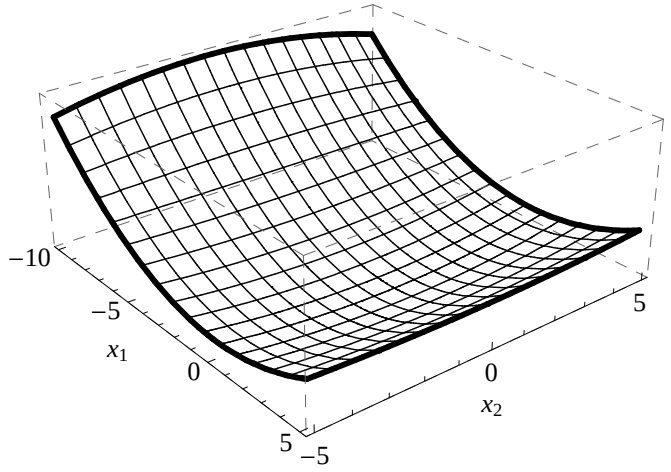
A fejezet hátralévő részében a vektorokat egységesen félkövérrel, a halmazokat nagybetűkkel, a függvényeket pedig kisbetűkkel jelölöm. Az alsó index sorrendiséget mutat, például z_i a szövegkörnyezettől függően a z vektor, vagy a Z rendezett halmaz i-edik elemét jelöli. Használom továbbá a $\nu(z)$ függvény jelölést egy z_i változókat, valamint valós konstansokat és tetszőleges függvényneveket tartalmazó ν kifejezés leírására.

Az egyszerűsítő módszer célja, hogy (3.1) olyan ekvivalens átírásait állítsa elő, amelyek kedvezőbbek a következő értelemben: kevesebb aritmetikai művelet szükséges a kiértékeléshez, a probléma dimenziója kisebb, vagy egyéb okból egyszerűbben, gyorsabban megoldható egy bizonyos megoldó számára. Ekvivalensen pedig azt értjük, hogy az eredeti és az átírt feladat optimumai között bijektív leképezés létezik.

A (3.1) probléma egyszerűen megoldható, ha f unimodális, vagyis csupán egy vonzáskörzete, egy lokális minimum pontja van (és nincsenek helyi minimum pontok) a megadott $X \subseteq \mathbb{R}^n$ keresési térben. Egydimenziós függvényeknél ez a tulajdonság könnyen felismerhető, de magasabb dimenziókban általában nem triviális annak eldöntése, hogy unimodális-e egy függvény.

Csendes és Rapcsák [20] a következőképpen definiálják az unimodalitást:

1. DEFINÍCIÓ Az $f(\mathbf{x})$ n-dimenziós folytonos függvény *unimodális* egy nyílt $X \subseteq \mathbb{R}^n$ halmazon, ha létezik végtelen, folytonos görbéknek egy halmaza,



3.1. ábra. *Az* $f(x_1, x_2) = 100x_1^2 + 40x_2^2 - (x_1 \cdot x_2 - 2)^2$ *függvény unimodális az* $x_1 = [-10, 5]$, $x_2 = [-5, 5]$ *valós intervallumon, habár az* $x_1 = -10$ *egyenes mentén nem az.*

amely homeomorf leképezését adja az n-dimenziós tér polárkoordináta-rendszerének, és $f(\mathbf{x})$ szigorúan monoton növekszik a görbék mentén.

Ebben az értelemben a multimodális a multiextremális szinonímája, ami megfelel az optimalizálás terén megszokott szóhasználatnak. Az említett görbék értelmezhetőek helyi keresések trajektóriáiként. Egy n-dimenziós unimodális függvény azonban nem feltétlenül unimodális a tér minden vonala mentén (ld. a 3.1. ábrát). Ugyanakkor a definíció azt sugallja, hogy bevezethetünk egy egyszerűbb g függvényt, amely az említett görbék mentén f-hez illeszkedik, és a vonzáskörzete megegyezik f-ével.

Következésképpen, ha $f(\mathbf{x})$ egy feltétel nélküli optimalizálási feladat célfüggvénye, és az 1. definíció értelmében unimodális, akkor megadható egy $g(\mathbf{y})$ függvény, amely minimalizálása (maximalizálása) után $f(\mathbf{x})$ optimumát megkaphatjuk egy egyszerű visszahelyettesítéssel.

Tehát Csendes és Rapcsák eredménye értelmében egy függvény implicit unimodalitása egy változó transzformáció alakjában explicitté tehető:

1. TÉTEL [20] A folytonos $f(\mathbf{x})$ függvény akkor, és csakis akkor unimodális az n-dimenziós valós térben, ha létezik egy homeomorf $y = h(\mathbf{x})$ változó transzformáció, amelyre $f(\mathbf{x}) = f\left(h^{-1}(\mathbf{y})\right) = \mathbf{y}^T\mathbf{y} + c$, ahol c egy valós konstans, és az origó $h(\mathbf{x})$ értékkészletében van.

A következő tétel elégséges feltételt mond ki arra vonatkozóan, hogy egy helyettesítés mikor egyszerűsíti a célfüggvényt:

2. TÉTEL [20] Ha $h(\mathbf{x})$ sima és szigorúan monoton az x_i függvényében, akkor a megfelelő transzformáció egyszerűsíti a függvényt abban a tekintetben, hogy a $h(\mathbf{x})$ minden $f(\mathbf{x})$-beli előfordulását egy új változóval helyettesítjük a $g(\mathbf{y})$ átalakítással kapott függvényben, miközben az $f(\mathbf{x})$ minden helyi minimum (maximum) helye a $g(\mathbf{y})$ függvény egy helyi minimum (maximum) helyévé transzformálódik.

Ha pedig az értékkészletre vonatkozó feltétel is teljesül, a helyettesítés bijektív leképezést létesít az eredeti és az átírt célfüggvény optimális megoldásai között:

3. TÉTEL [20] Ha $h(\mathbf{x})$ sima, szigorúan monoton az x_i függvényében, és az értékkészlete $\mathbb{R}$-rel egyenlő, akkor az átalakítással kapott $g(\mathbf{y})$ függvény minden $\mathbf{y}^*$ lokális minimum (maximum) helyére létezik olyan $\mathbf{x}^*$, hogy $\mathbf{y}^*$ az $\mathbf{x}^*$ transzformáltja, és $\mathbf{x}^*$ az $f(\mathbf{x})$ helyi minimum (maximum) helye.

Az utolsó tétel értelmében egy $g(\mathbf{y})$ célfüggvény ekvivalens a (3.1)-ben szereplő $f(\mathbf{x})$-szel, ha a következő átalakítással állítjuk elő:

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

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

  ahol $h(\mathbf{x})$ egy folytonos függvény $\mathbb{R}$ értékkészlettel, és szigorúan monoton legalább egy x_i változóban,

- átnevezzük a megmaradó változókat:

$$y_j := x_j, \quad j = 1, \ldots, i-1, i+1, \ldots, n, \quad \text{és}$$

- elhagyjuk azokat az y_i változókat, amelyek az előálló célfüggvényben nem szerepelnek.

Azt mondjuk, hogy $h(\mathbf{x})$ *lefedi* x_i *változót* $f(\mathbf{x})$-ben, ha $h(\mathbf{x})$ az x_i minden $f(\mathbf{x})$-beli előfordulását jellemzi, vagyis x_i teljesen eltűnik $f(\mathbf{x})$-ből, ha $h(\mathbf{x})$-et y_i-vel helyettesítjük.

*Alkalmas helyettesítés*nek hívjuk az $y_i = h(\mathbf{x})$ helyettesítést, ha

- $h(\mathbf{x})$ sima, monoton legalább egy x_i változóban, és értékkészlete $\mathbb{R}$,
- $h(\mathbf{x})$ lefed legalább egy x_i változót, továbbá
- $y_i = h(\mathbf{x})$ nem egyszerű átnevezés, vagyis, $h(\mathbf{x}) \neq x_i$, $i = 1, \ldots, n$.

Egy alkalmas $y_i = h(\mathbf{x})$ helyettesítés végrehajtása után $\mathbf{y}$ dimenziója nem nagyobb, mint $\mathbf{x}$-é. Kedvező esetben, ha $h(\mathbf{x})$ kettő, vagy több változót fed, az átalakítás után kapott feladat kevesebb változót tartalmaz, mint az eredeti. Másképp fogalmazva, az egyszerűsítő eljárás magában rejti a lehetőségét, hogy megmutassa, ha a modell az eredetinél kevesebb változóval is formalizálható. A redundáns változók felismerése igen hasznos lehet, és általában egyáltalán nem triviális. Ily módon az átírás eredménye hasznosítható lehet akkor is, ha a probléma nem hozható unimodális alakra.

Tekintsük például azt a feladatot, ahol az $f(x_1, x_2) = (x_1 + x_2)^2$ célfüggvény minimumát keressük. Ekvivalens probléma a $g(y_1) = y_1^2$ minimalizálása. Az eredeti x_1 és x_2 változók optimális értéke előállítható az $y_1 = x_1 + x_2$ összefüggés alapján, ami egy alkalmas helyettesítés. Ezen a módon ráadásul megoldható a végtelen számú optimumhely kezelése, ami a numerikus megoldók számára lehetetlen lenne.

Az egyszerűsítő eljárás egyik fő célja, hogy olyan alkalmas helyettesítéseket állítson elő, amelyek redundáns változók eliminációjával járnak. Csendes és Rapcsák vonatkozó eredménye a következő két állítás:

1. ÁLLÍTÁS [20] Ha x_i változó egy sima $f(\mathbf{x})$ függvényben mindenhol a $h(\mathbf{x})$ kifejezés részeként fordul elő, akkor a $\partial f(\mathbf{x})/\partial x_i$ parciális derivált felírható a $(\partial h(\mathbf{x})/\partial x_i)\, p(\mathbf{x})$ formában, ahol $p(\mathbf{x})$ folytonosan differenciálható.

2. ÁLLÍTÁS [20] Ha x_i és x_j változók egy sima $f(\mathbf{x})$ függvényben mindenhol a $h(\mathbf{x})$ kifejezés részeként fordulnak elő, akkor a $\partial f(\mathbf{x})/\partial x_i$ és $\partial f(\mathbf{x})/\partial x_j$ parciális deriváltak egyenként $(\partial h(\mathbf{x})/\partial x_i)\, p(\mathbf{x})$ és $\left(\partial h(\mathbf{x})/\partial x_j\right) q(\mathbf{x})$ alakú szorzattá alakíthatók, és teljesül a $p(\mathbf{x}) = q(\mathbf{x})$ egyenlőség.

Ha $\partial f(\mathbf{x})/\partial x_i$ nem alakítható szorzattá, akkor bármely, az x_i-ben monoton alkalmas helyettesítés lineárisan függ az x_i-től.

Ez az elmélet képezi az alapját annak a konstruktív szimbolikus algoritmusnak, amely implementációját a következő szakaszban bemutatom.

Az állítások azt sugallják, hogy számítsuk ki az $\partial f(\mathbf{x})/\partial x_i$ parciális deriváltakat minden x_i változóra, alakítsuk ezeket szorzattá, majd keressünk a szorzótényezők között alkalmas helyettesítéseket.

Demonstrációs célból tekintsük a következő feladatot:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^3} \;& 30 \cdot (5x_1 + e^{1+x_2}) + 20 \cdot x_3, \\ \text{f.h.} \;& \ln(5x_1 + e^{1+x_2}) + x_3 \geqslant 5. \end{aligned}$$

A példa egy folyamatszintézis probléma kicsiny részlete is lehet. Ismert, hogy a folyamatszintézis területén alkalmazott automatikus modell generáló eszközök általában – nem szándékoltan – többféle értelemben is redundáns modelleket állítanak elő [25]. Előfordulhat például, hogy egyes kémiai elemek mennyiségét jelölő változók kizárólag egy adott anyag képletében szerepelnek. A mi példánkban x_1 és x_2 kizárólag az $5x_1 + e^{1+x_2}$ kifejezésben, együtt szerepel, ezt fogjuk $h(\mathbf{x})$-szel jelölni. Az egyszerűség kedvéért a feltételt a büntetőfüggvény módszerrel [35] a célfüggvénybe építhetjük, így az

$$f(\mathbf{x}) = 30 \cdot (5x_1 + e^{1+x_2}) + 20 \cdot x_3 + \sigma \left(\ln(5x_1 + e^{1+x_2}) + x_3 - 5 - x_4 \right)^2$$

függvényt kell minimalizálnunk, ahol x_1, x_2, x_3 fizikai paramétereken alapuló, valós értékű változók lehetnek, σ a büntetést jelölő konstans, x_4 pedig egy mesterséges változó. A 2. állítás alapján $\partial f(\mathbf{x})/\partial x_1$ átalakítható $(\partial h(\mathbf{x})/\partial x_1) \cdot p(\mathbf{x})$ alakba, hasonlóképp $\partial f(\mathbf{x})/\partial x_2 = (\partial h(\mathbf{x})/\partial x_2) \cdot q(\mathbf{x})$, miközben $p(\mathbf{x}) = q(\mathbf{x})$. A példánkban $\partial h(\mathbf{x})/\partial x_1 = 5$, $\partial h(\mathbf{x})/\partial x_2 = e^{1+x_2}$, és

$$p(\mathbf{x}) = q(\mathbf{x}) = \frac{2\left(75x_1 + 15e^{1+x_2} + \sigma\left(\ln(5x_1 + e^{1+x_2}) + x_3 - 5 - x_4\right)\right)}{5x_1 + e^{1+x_2}}.$$

Természetesen a megfelelő $(\partial h(\mathbf{x})/\partial x_i)\, p(\mathbf{x})$ faktorizálás nem feltétlenül könnyű, és sok egyéb szorzat alak létezhet az 1. állításban említetteken felül. Mégis, a függvények egy bő osztályára létezik kanonikus alak. Például egy $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$ alakú, n gyökű polinom, ahol a gyököket $x_1, \ldots, x_n$ jelöli, standard $a_n(x - x_1)(x - x_2) \cdots (x - x_n)$ szorzat alakja mindig létezik.

További kritika lehet, hogy ha a kívánt szorzat alak meghatározható is, nem triviális egy jó $h(\mathbf{x})$ helyettesítés megtalálása. Az említett feltétel tehát elégséges, de nem szükséges előfeltétele az egyszerűsítő átalakításoknak.

3.4. Számítógépes megvalósítás a feltétel nélküli esetre

A fenti eredmény alapján szerzőtársaimmal szerettünk volna készíteni egy számítógépes programot, amely feltétel nélküli nemlineáris optimalizálási problémák ekvivalens átírásait állítja elő automatikusan. 2013-ban közölt munkánk [4] célja az volt, hogy a hivatkozott elméleti eredményt a gyakorlatba ültessük, és az algoritmus előnyeit és hátrányait egy nagyobb teszthalmazon megvizsgáljuk.

A 3.1. algoritmus a szimbolikus egyszerűsítő eljárás kivonatát adja. Az

3.1. algoritmus. *A szimbolikus egyszerűsítő algoritmus vázlata*

1. A célfüggvény gradiensének kiszámítása.
2. A parciális deriváltak szorzattá alakítása.
3. Az x_i-t tartalmazó alkalmas helyettesítéseknek egy l_i listába gyűjtése:
   (a) l_i inicializálása üres halmazként.
   (b) Ha $\partial f(\mathbf{x})/\partial x_i$ faktorizálása sikeres, egészítsük ki l_i-t a szorzótényezők integráljaival.
   (c) Vegyük hozzá l_i-hez $f(\mathbf{x})$-nek az x_i-ben lineáris részkifejezéseit.
   (d) Dobjuk el l_i azon elemeit, amelyek nem elégítik ki az „alkalmas helyettesítés" feltételeit (nem monoton x_i-ben).
4. Az $L = \bigcup l_i$, $i = 1, \ldots, n$ alkalmas helyettesítések minden helyes kombinációját $f(\mathbf{x})$-re alkalmazva az eredményeket tároljuk egy S listában. (A „helyes kombináció"-t később definiáljuk.)
5. Legyen S legkevésbé komplex eleme az új, egyszerűsített célfüggvény.
6. Oldjuk meg az optimalizálási feladatot az új célfüggvénnyel (ha lehetséges).
7. Állítsuk elő az eredeti feladat megoldását a helyettesítések inverzének alkalmazásával.

algoritmus lépéseinek nagy része (úgymint a parciális differenciálás, faktorizálás, szimbolikus integrálás és helyettesítés) a modern számítógépes algebra rendszerekben megbízható szimbolikus módszerként elérhető. Másrészt az első, Maple-ben készült implementációnk készítése közben világossá vált, hogy egy piacvezető általános célú számítógépes algebra rendszerben is komoly hiányosságok lehetnek a követelményeink tekintetében, különösen a szimbolikus helyettesítés és az intervallum aritmetika területén [4].

Valójában egy nemlineáris függvény pontos értékkészlet-becslése épp olyan bonyolult, mint a globális minimum és maximum meghatározása. A naív intervallum-befoglalás alkalmazható annak igazolására, hogy egy részkifejezés értékkészlete a valós számok halmaza. A naív intervallum-befoglalás pontos az úgynevezett „single use expression"-ök (SUE, olyan kifejezés, amely minden változót legfeljebb egyszer tartalmaz) esetében, azonban bonyolultabb kifejezésekre túlbecslést produkálhat [65].

A lehetséges túlbecslés miatt L tartalmazhat olyan helyettesítéseket, ame-

lyek értékkészlete nem a valós számok halmaza. Ezért egy utólagos megerősítés szükséges lehet a nem-SUE helyettesítések értékkészletével kapcsolatban. Ugyanakkor a tesztjeinkben előálló helyettesítések legtöbbje SUE volt. A naiv intervallum-befoglalás helyett a valós kvantor elimináció [80, 83] egy alkalmazható alternatív szimbolikus technika lenne az értékkészlet-becslésre, jelen munka keretében azonban nem teszteltük ezt a lehetőséget.

Az intervallumos befoglalás a monotonitás tesztben is alkalmazható. Egy valós $f : \mathbb{R}^n \to \mathbb{R}$ függvény monoton, ha bármely $\mathbf{x}, \mathbf{y} \in \mathbb{R}^n$ -re, ha $\mathbf{x} \leqslant \mathbf{y}$ akkor $f(\mathbf{x}) \leqslant f(\mathbf{y})$. Használhatjuk a következő részben-rendezést: $(x_1, \ldots, x_n) \leqslant (y_1, \ldots, y_n)$ ha $x_i \leqslant y_i$, $i = 1, \ldots, n$. A tárgyalt alkalmazásban azt kell ellenőriznünk, hogy egy $h_i(\mathbf{x})$ függvény szigorúan monoton-e az x_i változó függvényében. Ezért megvizsgáljuk, hogy $\partial h_i(\mathbf{x})/\partial x_i$ naív intervallumos befoglalása tartalmazza-e a nullát. Ez a megközelítés illik a monotonitás matematikai definíciójához, és szemléletes is, hiszen egy szigorúan monoton függvénynek pontosan egy vonzáskörzete van. Sajnos azonban lehetséges, hogy ezzel a módszerrel néhány monoton helyettesítést is eldobunk.

A 3.4.2. alszakaszban bemutatott első, Maple implementációnk a 4. lépés helyett egyszerűbb heurisztikát alkalmazott, erről részletesebben szólunk ott.

A 3. és a 4. lépés kombinálható az eljárás gyorsítása érdekében, valamint annak biztosítására, hogy a lehetséges helyettesítések helyes kombinációját alkalmazzuk $f(\mathbf{x})$-re. Ezt a 3.4.7. alszakaszban bemutatott Mathematica implementációban meg is tettem.

Alkalmas helyettesítések egy jólrendezett H halmazát *helyes*nek nevezzük, ha minden $h_i(\mathbf{x}) \in H$ helyettesítés egyszerre, párhuzamosan végrehajtható $f(\mathbf{x})$-ben, vagyis $\forall h_i(\mathbf{x}), h_j(\mathbf{x}) \in H$ ha $h_i(\mathbf{x}) \neq h_j(\mathbf{x})$ akkor nem fednek át $f(\mathbf{x})$ kiszámítási fájában. Enélkül a tulajdonság nélkül az $y_i = h_i(\mathbf{x})$, $h_i(\mathbf{x}) \in H$ helyettesítések némelyikét nem lehetne végrehajtani. Például $f(\mathbf{x}) = (x_1 + x_2 + x_3)^2$-ben $y_1 = x_1 + x_2$ és $y_2 = x_2 + x_3$ alkalmas helyettesítések, azonban a $H = \{x_1 + x_2, x_2 + x_3\}$ nem helyes helyettesítési halmaz, mivel $x_1 + x_2$ és $x_2 + x_3$ is x_2 ugyanazon előfordulására hivatkozik. Valójában a legbonyolultabb $h(\mathbf{x})$-et alkalmaznánk egy változó kiváltására, tehát ebben a példában az algoritmus az $y_3 = x_1 + x_2 + x_3$ helyettesítést fogadná el.

Ezen a ponton, az 5. lépésben a kifejezések egy könnyen vizsgálható komplexitás mértékére lesz szükségünk. A mi olvasatunkban egy kifejezést akkor nevezünk egy másiknál összetettebbnek, ha a reprezentációja (a kifejezést leíró pointer-lista) hosszabb.

3.4.1. Teszthalmaz

A szakirodalomban nem találtam szimbolikus átalakító eljárások teszteléséhez alkalmas módszertanra vagy bevett teszthalmazra, az elérhető publikációk általában egy-egy kiragadott, nagyméretű, konkrét alkalmazásból jövő feladaton mutatják be módszerük hatékonyságát. Ezért részben saját tesztfeladatokat gyártottam, részben a globális optimalizálás standardnak mondható tesztfeladataiból kölcsönöztem. Ez utóbbit az is indokolta, hogy így az automatikusan produkált átalakítások eredményét egy klasszikus globális optimalizálási megoldóval tesztelni tudtam [3] (ld. a 4. fejezetben).

A megvalósított egyszerűsítő eljárást az ihlető Csendes és Rapcsák [20] publikációban szereplő minden példán teszteltem. A 3.4. táblázatban szereplő saját tesztfeladatok egyszerű korlátozás nélküli nemlineáris célfüggvények, amelyek az eljárás részletesebb elemzését szolgálták.

A felhasznált standard globális optimalizálási tesztfeladatok részletes leírása, sőt különböző platformokra implementált változata elérhető több online gyűjteményben [78, 84]. Minden hivatkozott feladat tömör matematikai leírása az ismert optimumokkal együtt megtalálható pl. Pál László értekezésének A. függelékében is [68]. Egy-két kivétellel a feladatok részletes bemutatásától eltekintek, annál is inkább, mivel a dolgozat eredményeinek megértéséhez nem feltétlenül szükséges ezek ismerete. A feladatok dimenzióját azonban a 3.6. táblázat második oszlopában feltüntettem.

Mivel feltétel nélküli optimalizálási feladatokat vizsgáltunk, a tesztek tekintetében olykor a függvény, célfüggvény szavakat a feladat szinonímájaként használom, ha az nem félreérthető.

3.4.2. Az egyszerűsítő módszer Maple megvalósítása

Az implementált algoritmus pszeudokódja az A. függelékben, az A.1. algoritmusban megtalálható. A program legfontosabb szubrutinjait a következő listába foglaltam össze, a lehetséges Maple megvalósításra utaló kiegészítésekkel együtt.

`Factor(expr)` `expr` szorzótényezőinek egy listáját adja, vagy magát `expr`-et, ha az nem faktorizálható
Maple: `factor`, `afactor` (beépített függvények)

`NumbOccur(expr,y)` `y`-nak `expr`-beli előfordulásainak a számát adja
Maple: `numboccur` (beépített függvény) problémákkal (ld. a 3.4.3. alszakaszt)

`Decompose(expr,y)` `expr` olyan részkifejezéseinek a listáját adja vissza, amelyek `y`-t tartalmazzák
Maple: a beépített `convert(expr,list)` függvény rekurzív hívásával megvalósítva

`IsType(expr,'+')` igaz, ha az `expr` kifejezés legnagyobb precedenciájú operátora összeadás vagy kivonás
Maple: `whattype` (beépített függvény)

`IsLinear(expr,y)` igaz, ha az `expr` kifejezés `y`-ban lineáris
Maple: `type` és `linear` (beépített függvények)

`Sort(exprlist,y)` az `y` lehetséges helyettesítéseinek `exprlist` listáját a helyettesítés hasznossága alapján csökkenő sorrendbe rendezi (nem triviális, a polinomokat preferáljuk, még ha összetettebb kifejezések is)
Maple: lista műveletekkel, valamint a `sort` és `PolynomialTools[Sort]` beépített függvényekkel megvalósítva

`Test(expr,y)` igaz, ha `expr` lefedi `y`-t és értékkészlete $\mathbb{R}$
Maple: nem triviális (ld. a 3.4.4. alszakaszt)

`Subs(expr,x,y)` `x` kifejezés helyére `y`-t helyettesít `expr`-ben
Maple: `subs`, `algsubs` (beépített függvények) problémákkal (ld. a 3.4.3. alszakaszt)

`Solve(expr,y)` `expr`-t minimalizálja az `y` változókkal
Maple: nem triviális, például a beépített `minimize` függvénnyel

`Transform(sollist,subslist)` az eredeti függvény megoldásait adja, ha `sollist` a `subslist`-ben adott átalakítások szerint transzformált függvény minimum helyeit tartalmazza
Maple: `subs` és `solve` (beépített függvények)

`Verify(x,y)` ellenőrzi, hogy `x` és `y` ekvivalens, mint célfüggvény (értékkészletbecslés `x`-re és `y`-ra)
Maple: nem triviális (ld. a 3.4.5. alszakasz)

A megvalósítás érdekesebb részleteit a következő, 3.4.3–3.4.5. alszakaszok mutatják be.

3.4.3. Kérdéses: megfelelő algebrai helyettesítések

A Maple kétféle helyettesítési lehetőséget kínál: a szintaktikai helyettesítést a `subs`, az algebrai helyettesítést pedig az `algsubs` függvény segítségével hívhatjuk meg. A `simplify` parancs ugyancsak kapcsolódó funkcióval bír, mivel a felhasználó által definiált szabályok alkalmazását biztosítja adott kifejezésre, amennyiben a szabályokban szereplő relációk a változókban polinomiálisak. A Maple 15-ös verzióját használva gyűjtöttem néhány egyszerű példát a három parancs képességeinek bemutatására:

Parancs	Eredmény	Megfelelő?
`subs(a*b = d, a*b*c)`	abc	N
`algsubs(a*b = d, a*b*c)`	cd	I
`simplify(a*b*c, {a*b = d})`	cd	I
`subs(a = b, 1/a)`	$1/b$	I
`algsubs(a = b, 1/a)`	$1/a$	N
`simplify(1/a, {a = b})`	$1/b$	I
`subs(sqrt(a + b) = d, sqrt(a + b) + c)`	$c + d$	I
`algsubs(sqrt(a + b) = d, sqrt(a + b) + c)`	hiba	N
`simplify(sqrt(a + b) + c, {sqrt(a + b) = d})`	hiba	N
`subs(2^(a + b) = d, 2^(a + b))`	d	I
`algsubs(2^(a + b) = d, 2^(a + b))`	hiba	N
`simplify(2^(a + b), {2^(a + b) = d})`	hiba	N

A felsorolt lehetőségek egyike sem látszik tökéletesnek. Úgy tűnik azonban, hogy a `subs` eljárás a legrobusztusabb, ezért a továbbiakban ezt használtuk.

A `numboccur` függvény ugyancsak hamis eredményeket produkál bizonyos esetekben, talán épp az alkalmatlan helyettesítések miatt:

Parancs	Eredmény	Megfelelő?
`numboccur(a*b+a*b,a*b)`	1	I
`numboccur(a*b*c,a*b)`	0	N
`numboccur(1/a,a)`	1	I
`numboccur(1/(a*b),a*b)`	0	N
`numboccur(Exp(Sin(a/b))*Exp(1),Sin(a/b))`	1	I
`numboccur(Exp(Sin(a/b)+1),Sin(a/b))`	0	N

3.4.4. Kérdéses: a helyettesítés tulajdonságainak ellenőrzése

Ahogyan korábban is tárgyaltuk, minden $y = h(\mathbf{x})$ helyettesítés esetén meg kell győződnünk bizonyos tulajdonságok teljesüléséről.

A 3. tétel értelmében $h(x)$ helyettesítés akkor lehet célszerű, ha

- értékkészlete a teljes valós számhalmaz: $R_h = \mathbb{R}$,
- $h(\mathbf{x})$ sima, és
- $h(\mathbf{x})$ szigorúan monoton legalább egy változóban.

Ezek azonban nem feltétlenül szükséges feltételek egy célszerű, egyszerűsítést produkáló helyettesítésre, ráadásul ezen feltételeket ellenőrző tesztek számítógépes megvalósítása sem könnyű. További elméleti eredmények lennének szükségesek megfelelő, ám könnyebben verifikálható tulajdonságok leírására.

3.4.5. Kérdéses: értékkészlet-becslés

Erre a célra kézenfekvő megoldásnak tűnt a naiv intervallumos befoglalás. Kétféle megoldást találtam a Maple-ben intervallumos műveletekhez. Az egyik a beépített „range arithmetic", a másik pedig az intpakX [33, 47] nevű kiegészítő csomag. Sajnos mindkettőnél találtam hibás, vagy hiányos működésre utaló jelenségeket, amint azt a következőkben be is mutatom.

A Maple eredetileg az `evalr` függvény segítségével tud speciálisan definiált intervallumos kifejezéseken egyszerű intervallumos műveleteket végezni. Amint azonban [34] is közölte, a kifelé kerekítés és egyéb szükséges funkciók hibás megvalósítása miatt ez az intervallum aritmetika megbízhatatlan. Megemlíteném például, hogy az automatikus egyszerűsítő sokszor hibázik a beépített intervallum típusú változókon, így óvatosan kell kezelni a Maple által produkált eredményeket. Szemléltetésül tekintsük a következő programkódot:

```
> a := INTERVAL(-1 .. 1);
INTERVAL(-1 .. 1)
> b := INTERVAL(-1 .. 1);
INTERVAL(-1 .. 1)
> evalr(a*b);
INTERVAL(0 .. 1)
```

Az első két utasítás deklarálja az $a = [-1, 1]$ és $b = [-1, 1]$ változókat speciális, intervallum adattípusként. A következő sorban kiszámítjuk az $a \cdot b$ szorzatot,

és a Maple a $[0, 1]$ eredményt adja. Ez a megoldás nyilvánvalóan hibás, a helyes megoldás a $[-1, 1]$ lenne, azonban a Maple éppúgy számol, mintha a $[-1, 1]$ intervallumot négyzetre emeltük volna.

Az intpakX csomag típusokat, operátorokat és további, valós intervallumokra és komplex (körszerű) intervallumokra vonatkozó speciális eljárásokat definiál. Minket jelen esetben csak a funkciók első csoportja érdekel, tehát, a valós intervallumokkal a Maple-ben végezhető műveletek. Az intpakX a beépített „range arithmetic" operátoraitól eltérő neveket használ a biztos elkülönülés érdekében. A kifelé kerekítést is egyedileg valósítja meg. Ugyanakkor a standard Maple operátorokat és függvényneveket használja az intpakX is, így ha a CAS aritmetikája a legkisebb pontos egységnél (1 ULP) nagyobb hibát vét, az megjelenik az intpakX eredményében is [34].

Az intpakX célja tehát, hogy korrekt intervallum aritmetikát valósítson meg a Maple-ben, azonban ebben a csomagban is találtam hiányosságokat, ami a konkrét alkalmazásunk szempontjából kompromisszumot jelenthet. Például, a nem korlátozott intervallumokra (vagyis végtelen végponttal megadott intervallumokra) kiterjesztett osztás nem lehetséges:

```
> ext_int_div(construct(-infinity, -1), construct(1, 2)):
Error, (in intpakX:-ext_int_div) first arg must be a finite
 interval or a numeric
```

Sajnos a feltétel nélküli optimalizálási problémák vizsgálata során könnyen előfordulhat, hogy nem korlátozott intervallumokkal kell dolgoznunk, és érdekes módon az intpakX dokumentációja nem is tiltja a használatukat.

Valójában az intpakX tartalmaz értékkészlet becslő algoritmusokat is, de sajnos csak kettő- és háromdimenziós függvényekre [33, 47, 34].

Megjegyezzük, hogy abban az esetben, ha egy függvény folytonos és szigorúan monoton a változókban (amint azt a 3. tétel feltételezi), az értékkészlet-becslés gyorsan és pontosan elvégezhető számítógépen az intervallum aritmetika segítségével [65]. Azonban nincs egyszerű mód, hogy egy felhasználó által definiált függvényről eldöntsük a Maple-ben, hogy az monoton vagy sem.

Elvileg lehetséges más programnyelveken megvalósított rutinok hívása is a Maple-ből, a leginkább kézenfekvő a MATLAB, és az ott létező INTLAB [73] csomag használata lenne. Amennyiben mindkét rendszer telepítve van egy számítógépre, a Maple futásidejű kapcsolatot tud építeni a MATLAB felé. Először is be kell tölteni az INTLAB csomagot MATLAB alatt, hogy a Maple `Matlab` csomagjával is használni tudjuk. Meghívás esetén egy munkafüzethez egy MATLAB munkamenet rendelődik. A felhasználó manuálisan, a `Matlab[openlink]` és `Matlab[closelink]` parancsokkal nyithatja és zárhatja

a munkamenetet. A következő parancs létrehozza az x intervallum változót INTLABban:

```
> Matlab[evalM]("x=infsup(1,2)");
```

Ugyanezen a módon bármilyen parancsot kiértékelhetünk a MATLAB környezetben. Az eredmény intervallumot a következőképpen nyerhetjük ki:

```
> Matlab[evalM]("[a]=[x.inf,x.sup]");
> Matlab[getvar]("a");
```

Ez a mód használható, habár igen körülményes intervallumos számításokat biztosít a Maple környezetből számunkra.

Végül is az említett anomáliák miatt a 2013-ban publikált Maple implementációnk [4] az intervallumos befoglalás helyett csupán a második deriváltat tesztet alkalmazta, a szemidefinit esetekben pedig néhány függvénykiértékelés alapján heurisztikusan döntött.

3.4.6. Teszteredmények a Maple implementációval

Egy sikeres példa

A módszerünket teszteltük az eredeti közleményben is szereplő Rosenbrock függvényen. Ez egyike a standard globális optimalizálási tesztfeladatoknak. A célfüggvény a következő:

$$f(\mathbf{x}) = 100\left(x_1^2 - x_2\right)^2 + \left(1 - x_1\right)^2.$$

Ez a függvény úgy lett megkonstruálva, hogy a keresési tér a szakértő számára könnyen átlátható legyen [72], a nehézsége pedig különösen a gradiens és kisebb mértékben a kvázi-Newton módszerek esetén jelentkezett. A szinthalmaz speciális alakja miatt banán-függvénynek is nevezik (ld. 3.2. ábra). Globális optimuma $f(\mathbf{x}^*) = 0$, optimumhelye az $\mathbf{x}^* = (1, 1)$.

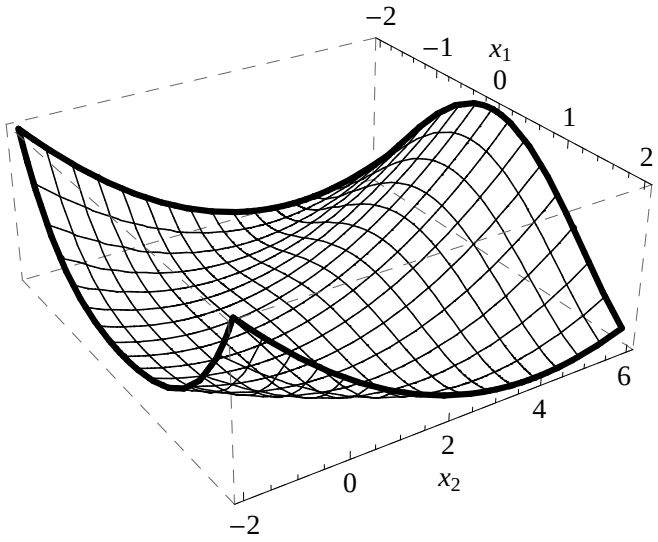
A következő függvényhívással indíthatjuk a szimbolikus egyszerűsítőt:

```
> symbsimp([x2, x1], 100*(x1^2-x2)^2+(1-x1)^2);
```

Az első lépésben az algoritmus kiszámolja a parciális deriváltakat:

$$\texttt{dx(1)} = -200x_1^2 + 200x_2$$

$$\texttt{dx(2)} = 400(x_1^2 - x_2)x_1 - 2 + 2x_1$$



3.2. ábra. *A Rosenbrock függvény szinthalmazának jellegzetes „banán" alakja a globális minimum közelében*

Jelen esetben, vagyis a Maple implementációban a változóknak az első paraméterben megadott sorrendje is számít, ezért listát, és nem halmazt kell megadni. Tehát `dx(1)` a Rosenbrock függvény első változó, jelen esetben x_2 szerinti parciális deriváltja.

Ezután a parciális deriváltak szorzat alakja kerül kiszámításra:

$$\texttt{factor(dx(1))} = -200x_1^2 + 200x_2,$$

$$\texttt{factor(dx(2))} = 400x_1^3 - 400x_1x_2 - 2 + 2x_1.$$

Mivel a `factor` utasítás nem tudja szorzattá alakítani egyik parciális deriváltat sem, visszakapjuk az eredeti kifejezést (eltekintve attól, hogy a Maple automatikus egyszerűsítő mechanizmusa a zárójeleket automatikusan kiszorozza).

Az `f` részkifejezéseinek listája az x_2-re vonatkozó összetettség szempontjából csökkenő sorrendbe állítva a következő:

$$\{100(x_1^2 - x_2)^2, (x_1^2 - x_2)^2, x_1^2 - x_2, -x_2, x_2, (1 - x_1)^2, x_1^2, 100, 2, -1\}.$$

Ez a lista `f` rekurzív felbontásával áll elő és x_2 lehető legegyszerűbb együtthatóit is tartalmazza, habár az utolsó helyeken álló konstansokat sosem választja az algoritmus. Mivel `dx(1)` szorzattá alakítása nem volt sikeres, megkeressük a lista első elemét, amely x_2-ben lineáris, ez az $x_1^2 - x_2$. A kiválasztott kifejezés értékkészlete $(-\infty, +\infty)$, és x_2-t fedi, így az $y_1 = x_1^2 - x_2$-t elfogadjuk.

Ekkor az algoritmus által átalakított függvény a $g = 100y_1^2 + (1 - x_1)^2$.

Tekintsük a parciális deriváltakat és ezek szorzat alakját:

$$\texttt{factor(dx(1))} = \texttt{dx(1)} = 200y_1,$$

$$\texttt{factor(dx(2))} = \texttt{dx(2)} = -2 + 2x_1.$$

A g részkifejezései az x_1-re vonatkozó összetettség szerint rangsorolva:

$$\{(1 - x_1)^2, 1 - x_1, -x_1, x_1, 100y_1^2, 2, 1, -1\}.$$

Ismét az első kifejezést keressük, amely az aktuális x_1 változóban lineáris. $1 - x_1$ ilyen, és a többi szükséges feltételnek is megfelel az automatikus teszt alapján (értékkészlete $(-\infty, +\infty)$ és fedi x_1-et). Tehát a helyettesítésünk $y_2 = 1 - x_1$.

Az automatikus egyszerűsítő eljárás által előállított alternatív célfüggvény tehát:

$$g = 100y_1^2 + y_2^2.$$

A beépített `minimize` eljárással is könnyen kiszámíthatjuk g extrémumát: $y_1 = 0, y_2 = 0$. Az eredeti $f(\mathbf{x})$ célfüggvény minimuma megegyezik az átírt $g(\mathbf{y})$ minimumával (0). Az eredeti célfüggvény minimum helye pedig az átírás inverze alapján meghatározható: $x_1 = 1, x_2 = 1$.

Mivel minden végrehajtott helyettesítés értékkészlete a valós számok halmaza, a 3. tétel értelmében az eredeti célfüggvény minden minimumát megtaláltuk. Továbbá, mivel az egyszerűsített alak értékkészlete az eredetinek a részhalmaza ($[0, +\infty) \subseteq [0, +\infty)$), minden, a transzformáció után előálló minimum tényleges megoldása az eredeti problémának.

Egy sikertelen példa

Megpróbáltuk Csendes és Rapcsák [20] komplexebb paraméter-becslési feladatára is alkalmazni az eljárást. A célfüggvény:

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

Csalódottan vettük tudomásul, hogy a Maple nem tudta helyesen kezelni F-et definiálatlan m paraméter mellett, ezért a tesztben konstans értékre rögzítettük (jelen esetben $m = 3$). Az eredeti modell függvény:

$$Z'_L(\omega) = R_{aw} + \frac{B\pi}{4{,}6000\omega} - \imath\left(I_{aw}\omega + \frac{B\log(\gamma\tau\omega)}{\omega}\right).$$

Arra számíthatunk, hogy az eljárás előállít egy, a τ-ra vonatkozó nemtriviális helyettesítést, amint azt a 3.3.1. alszakaszban részletesebben kifejtettem.

Valóban, a τ szerinti parciális derivált szorzattá alakítható, és az $1/\tau$ szorzótényező τ-ra vonatkozó határozatlan integrálja felismerten $\ln(\tau)$. Azonban a program nem kutatja az $\ln(\tau)$-t tartalmazó részkifejezéseket, mert τ és $\ln(\tau)$ előfordulásainak számát különbözőnek számítja. Tehát ezúttal a Maple elégtelen helyettesítési képességei vezetik félre az algoritmusunkat. A program csupán az $y_1 = \omega$ átnevezést és az $y_2 = -R_{aw}$ helyettesítést végzi el, az előbbi eredményeképp $\omega_1, \omega_2, \omega_3$ helyére $y_1[1], y_1[2], y_1[3]$ kerül.

Az eredeti publikációban szereplő példákon elért eredmények

A 3.1. táblázatban foglaltam össze az eredeti [20] publikációban említett példákra a megvalósított automatikus egyszerűsítő program [4] által szolgáltatott eredményeket. A *Cos* jelű feladatra a várt helyettesítést állította elő a Maple program. A fent tárgyalt paraméter-becslési feladatra *ParamEst1* néven hivatkozok, g_1 az ott eredményül kapott, az eredetihez nagymértékben hasonlító függvényt jelöli:

$$\begin{aligned} g_1 =& 0{,}5774\left(\left|Z_L\left(y_1[1]\right) + y_2 - \frac{0{,}6830y_4}{y_1[1]}\right.\right.\\ &\left. + \imath\left(y_3y_1[1] + \frac{0{,}4343y_4\ln(1{,}7783y_5y_1[1])}{y_1[1]}\right)\right|^2\\ &+ \left|Z_L(y_1[2]) + y_2 - \frac{0{,}6830y_4}{y_1[2]}\right.\\ &\left. + \imath\left(y_3y_1[2] + \frac{0{,}4343y_4\ln(1{,}7783y_5y_1[2])}{y_1[2]}\right)\right|^2\\ &+ \left|Z_L(y_1[3]) + y_2 - \frac{0{,}6830y_4}{y_1[3]}\right.\\ &\left.\left. + \imath\left(y_3y_1[3] + \frac{0{,}4343y_4\ln(1{,}7783y_5y_1[3])}{y_1[3]}\right)\right|^2\right)^{1/2}. \end{aligned}$$

3.1. táblázat. *A Maple implementációnak az eredeti publikációban szereplő példákon elért eredményei*

Azon.	f függvény	g függvény	Helyettesítések	Fel.	Ered.
Cos	$\cos(e^{x_1}+x_2)+$ $+\cos(x_2)$	$\cos(y_1)+$ $+\cos(y_2)$	$y_1 = e^{x_1}+x_2$, $y_2 = x_2$	A	1
ParamEst1	$[1/3\sum_{i=1}^{3}\lvert Z_L(\omega_i)+$ $-Z_L'(\omega_i)\rvert^2]^{1/2}$	g_1	$y_1 = \omega, y_2 = -R_{aw}$, $y_3 = I_{aw}, y_4 = B$, $y_5 = \tau$	A	2a
ParamEst2	$[1/3\sum_{i=1}^{3}\lvert Z_L(\omega_i)+$ $-Z_L''(\omega_i)\rvert^2]^{1/2}$	$0{,}5774y_5^{1/2}$	$y_1 = \omega, y_2 = -R_{aw}$, $y_3 = I_{aw}, y_4 = B, y_5$	A	3ab
ParamEst3	$[1/3\sum_{i=1}^{3}\lvert Z_L(\omega_i)+$ $-Z_L'''(\omega_i)\rvert^2]^{1/2}$	$0{,}5774y_5^{1/2}$	$y_1 = \omega, y_2 = -R_{aw}$, $y_3 = I_{aw}, y_4 = B, y_5$	A	3b
Otis	$(\lvert Z_L(s)+$ $-Z_m(s)\rvert^2)^{1/2}$	$(\lvert -Z_L[1]+$ $+1.y_2/y_4\rvert^2)^{1/2}$	$y_1 = s, y_2 = I_C(R_1+$ $R_2)C_1C_2y_1^3+$ $(I_C(C_1+C_2)+$ $(R_C(R_1+R_2)+$ $R_1R_2)C_1C_2)y_1^2+$ $(R_C(C_1+C_2)+$ $R_1C_1+R_2C_2)y_1+1$, $y_4 =$ $(R_1+R_2)C_1C_2y_1^2+$ $(C_1+C_2)y_1$	B	3

A 3.1–3.6. táblázatokban szereplő „Fel." és „Ered." oszlopok kódjai a feladatot, ill. az eredményt értékelik. A „Fel." oszlop betűi az aktuális problémára vonatkoznak:

A: Egyszerűsítő átalakítások adhatók a bemutatott elmélettel összhangban.

B: Egyszerűsítő átalakítások létezhetnek a bemutatott elmélet kiterjesztésével.

C: Hasznosnak tűnő átalakítások létezhetnek a bemutatott elmélet kiterjesztésével, de azok nem feltétlenül egyszerűsítik a feladatot (pl. a dimenzióját növelik).

D: Nem számítottunk semmilyen hasznos átalakításra.

Az eredményeket az „Ered." oszlopban értékeltem: a programunk

1: korrekt helyettesítést,

2: semmilyen helyettesítést, avagy

3: inkorrekt helyettesítést eredményezett.

A „2" kódot használtam akkor is, ha csak egy konstans együtthatót sikerült eliminálni (mint a fenti példában y_2 segítségével).

Röviden tehát, „A 1" azt jelenti, hogy alkalmas helyettesítés lehetséges a célfüggvényben, és az algoritmus meg is találja azt, míg „D 2" jelentése, hogy legjobb tudomásunk szerint nem mutatható alkalmas helyettesítés, és a program is erre a következtetésre jutott. A többi kód a sikertelen eseteket jelöli.

A hibás megoldások nagy részét két okra tudtuk visszavezetni, így ezeket is jelöltem a táblázokban az „Ered." oszlopban szereplő kisbetűvel: a hiba okát az

a: algebrai helyettesítés, vagy

b: az értékkészlet-becslés hibája okozta.

Az *Otis* modell sorában szereplő „1." pedig egy lebegőpontos számot jelöl, amelyet a Maple 1-hez nagyon közelinek ítélt.

A *ParamEst1* példa sikertelensége miatt megpróbáltam a log-os kifejezést kiemelni a modell függvényben:

$$Z_L''(\omega) = R_{aw} + \frac{B\pi}{4{,}6000\omega} - \imath\left(I_{aw}\omega + \frac{B(\log(\gamma) + \log(\tau) + \log(\omega))}{\omega}\right),$$

ezt a változatot *ParamEst2*-vel jelöltem. Ebben az alakban a program már listázza az $\ln(\tau)$-t tartalmazó kifejezéseket, viszont a heurisztikus értékkészlet-becslő rutin miatt egy alkalmatlan helyettesítést fogad el:

$$\begin{aligned} y_5 = &\left| Z_L\left(y_1[1]\right) + y_2 - \frac{0{,}6830 y_4}{y_1[1]} + \imath\Big(y_3 y_1[1] \right. \\ &\left. + \frac{y_4(0{,}2500 + 0{,}4343\ln(\tau) + 0{,}4343\ln(y_1[1]))}{y_1[1]}\Big)\right|^2 \\ &+ \left| Z_L(y_1[2]) + y_2 - \frac{0{,}6830 y_4}{y_1[2]} + \imath\Big(y_3 y_1[2] \right. \\ &\left. + \frac{y_4(0{,}2500 + 0{,}4343\ln(\tau) + 0{,}4343\ln(y_1[2]))}{y_1[2]}\Big)\right|^2 \\ &+ \left| Z_L(y_1[3]) + y_2 - \frac{0{,}6830 y_4}{y_1[3]} + \imath\Big(y_3 y_1[3] \right. \\ &\left. + \frac{y_4(0{,}2500 + 0{,}4343\ln(\tau) + 0{,}4343\ln(y_1[3]))}{y_1[3]}\Big)\right|^2. \end{aligned}$$

Alaposabban megvizsgálva a program által listázott τ-ra vonatkozó lehetséges helyettesítéseket, a legkedvezőbb opció a $0{,}4343\ln(\tau)$, vagyis nem csak az értékkészlet-becslés hibája miatt nem kaptuk meg a kívánt helyettesítést.

Végül az eredeti paraméter becslési probléma egy harmadik variációját is megvizsgáltam (jelölése *ParamEst3*), itt a B-vel való szorzás zárójelét előre felbontottam:

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

Így a helyettesítés hibáját kikerültük, azonban a várt helyettesítést itt sem sikerült a rendszerből kikényszeríteni. A táblázatban ki nem fejtett bonyolult helyettesítés a következő:

$$\begin{aligned} y_5 = &\left| Z_L\left(y_1[1]\right) + y_2 - \frac{0{,}6830y_4}{y_1[1]} + \imath\Big(y_3y_1[1] \right. \\ &\left. + \frac{0{,}2500y_4 + 0{,}4343y_4\ln(\tau) + 0{,}4343y_4\ln(y_1[1])}{y_1[1]}\Big)\right|^2 \\ &+ \left| Z_L(y_1[2]) + y_2 - \frac{0{,}6830y_4}{y_1[2]} + \imath\Big(y_3y_1[2] \right. \\ &\left. + \frac{0{,}2500y_4 + 0{,}4343y_4\ln(\tau) + 0{,}4343y_4\ln(y_1[2])}{y_1[2]}\Big)\right|^2 \\ &+ \left| Z_L(y_1[3]) + y_2 - \frac{0{,}6830y_4}{y_1[3]} + \imath\Big(y_3y_1[3] \right. \\ &\left. + \frac{0{,}2500y_4 + 0{,}4343y_4\ln(\tau) + 0{,}4343y_4\ln(y_1[3])}{y_1[3]}\Big)\right|^2. \end{aligned}$$

Ez az előző y_5-től csak a számláló zárójelezésében különbözik. A lehetséges helyettesítések listája végre tartalmazza a $0{,}4343y_4\ln(\tau)$-t (és $y_4 = B$ egy korábbi átnevezésből), a *ParamEst2*-nél leírt okokból azonban ez nem kerül kiválasztásra.

Az utolsó probléma, amit az eredeti cikkből kölcsönöztünk, az *Otis* modell átalakítása. Itt a célfüggvény a

$$Z_m(s) = \frac{Ds^3 + Cs^2 + Bs + 1}{Gs^2 + Fs},$$

ahol $s = j\omega$ és

$$\begin{aligned}
B &= R_C(C_1 + C_2) + R_1C_1 + R_2C_2, \\
C &= I_C(C_1 + C_2) + [R_C(R_1 + R_2) + R_1R_2]C_1C_2, \\
D &= I_C(R_1 + R_2)C_1C_2, \\
F &= C_1 + C_2, \\
G &= (R_1 + R_2)C_1C_2.
\end{aligned}$$

A modell eredeti paraméterei a megjelenés sorrendjében az R_C, C_1, C_2, R_1, R_2, és I_C. A felírásból már látható az egyszerűsítési lehetőség: az utolsó öt egyenlet alapján definiálható B, C, D, F, és G változó. Mi több, ez az átírás egyúttal a modell dimenziójának csökkentésével is jár. Figyeljük meg, hogy egyesével egyik új változó sem teljesíti az „alkalmas helyettesítés" követelményeit, hiszen több új változó együtt fedi az egyes eredeti változókat. Amint Csendes és Rapcsák [20] is leírták, ez a fajta helyettesítés túlmutat a megadott elméleten. A hasonló helyettesítések leírására új elméleti eredményt publikáltunk [3], amelyet részletesebben a 3.5. szakaszban mutatok be.

Globális optimalizálási tesztfeladatokon elért eredmények

A 3.2. táblázat tartalmazza a standard globális optimalizálási tesztfeladatokon elért eredmények összefoglalását. Amint a 3.4.6. alszakaszban láttuk, a *Rosenbrock* függvény könnyen egyszerűbb alakra hozható. Ez nem igazán meglepő, hiszen a feladat úgy lett megkonstruálva, hogy a keresési tér a szakértő számára könnyen átlátható legyen. Az *RCOS* függvény egyszerűsítése viszont mindenképp meglepetés. Mindent összevetve a standard globális optimalizálási feladatokon elért eredményt sikerként könyvelhetjük el, hiszen az általunk egyszerűsíthetőnek gondolt feladatokat felismerte az eljárás, a bonyolultabbakat pedig nem átalakíthatóként azonosította.

Következő lépésként további gyakran alkalmazott globális optimalizálási teszt feladatokat vizsgáltunk, az ezekre vonatkozó eredményeket a 3.3. táblázat tartalmazza. Ebből a teszthalmazból csak a *Schwefel-227*-t sikerült átírni, a többit nem egyszerűsíthetőként ismerte fel a program.

Végül, a bemutatott automatikus egyszerűsítő képességeit és korlátait jobban behatárolandó, a 3.4. táblázatban szereplő saját tesztfüggvényeimet vizsgáltuk. A felmerülő hibákat részletesebben körülírom a 3.4.8. alszakaszban.

3.2. táblázat. *A Maple implementáció standard globális optimalizálási tesztfeladatokon elért eredményei*

Azon.	g függvény	Helyettesítések	Fel.	Ered.
Rosenbrock	$100y_2^2 + (1 - y_1)^2$	$y_1 = x_1, y_2 = y_1^2 - x_2$	A	1
Shekel-5	memória hiba	semmi	D	2
Hartman-3	semmi	semmi	D	2
Hartman-6	semmi	semmi	D	2
Goldstein-Prize	semmi	semmi	D	2
RCOS	$y_2^2 + 10(1 - 1/8/\pi) * \cos(y_1) + 10$	$y_1 = x_1, y_2 = (5/\pi)y_1 - 1{,}2750y_1^2/\pi^2 + x_2 - 6$	A	1
Six-Hump-Camel-Back	semmi	semmi	D	2

Tapasztalatom szerint a felmerülő hibák nagy része egy kedvezőbb programkörnyezetben megvalósítva kiküszöbölhető, az egyszerűsíthető problémák köre pedig az elmélet továbbfejlesztésével lenne bővíthető.

3.4.7. Az egyszerűsítő módszer Mathematica megvalósítása

A 3.4.2. szakaszban bemutatott Maple implementáció során felmerült – a programkörnyezetből adódó – hiányosságok kiküszöbölésére az eljárást megvalósítottam Mathematicában is [3]. A korábbival összehasonlítva a Mathematicának több előnye is van. Először is, a mi célunkhoz szükséges helyettesítések sokkal jobban működnek, mivel a Mathematica programozási nyelve term-átíráson alapszik [57]. Pontosabban, a Mathematica helyettesítő rutinja reguláris kifejezésekkel megadott term-átírási szabályokkal vezérelhető, ráadásul ezek a felhasználó által definiált szabályok magasabb precedenciát élveznek a rendszerbe építettekkel szemben [31]. Az általam írt speciális helyettesítő rutin mintegy ötven programsorból áll, ami a Mathematica elegáns, kifejező nyelvét ismerők számára önmagában is árulkodik összetettségéről. Tucatnyi (késleltetett) szabályt vezettem be, melyeket négy különböző módon értékelek ki, bevonva a képlet Mathematica által kifejtett, egyszerűsített, és faktorizált alakjait is. Valószínűleg ez a rutin a program legfontosabb része, hiszen több fázisban is meghívódik, így kis javításokkal alapvetően megváltoztatható az egyszerűsítési folyamat eredménye.

3.3. táblázat. *A Maple implementáció további gyakran használt globális optimalizálási tesztfeladatokon elért eredményei*

Azon.	g függvény	Helyettesítések	Fel.	Ered.
Levy-1	semmi	semmi	D	2
Levy-2	semmi	semmi	D	2
Levy-3	semmi	semmi	D	2
Booth	semmi	semmi	C	2
Beale	semmi	semmi	C	2
Powell	$(y_1+10y_2)^2+5(y_3+y_4)^2+(y_2-2y_3)^4+10(y_1+y_4)^4$	$y_1=x_1, y_2=x_2, y_3=x_3, y_4=-x_4$	D	2
Matyas	semmi	semmi	D	2
Schwefel ($n=2$)	semmi	semmi	C	2
Schwefel-227	$y_2^2+0{,}25y_1$	$y_1=x_1, y_2=y_1^2+x_2^2-2y_1$	A	1
Schwefel-31 ($n=5$)	semmi	semmi	D	2
Schwefel-32 ($n=2$)	semmi	semmi	A	2
Rastrigin ($n=2$)	semmi	semmi	C	2
Ratz-4	semmi	semmi	C	2
Easom	semmi	semmi	D	2
Griewank-5	semmi	semmi	D	2

A Mathematicában található intervallum aritmetika is megbízhatóbb: ez a helyettesítésre szóba jövő kifejezések gyors és megbízható értékkészlet-becsléséhez különösen fontos. Az értékkészlet behatárolására szolgáló naív intervallumos befoglalást a beépített intervallum aritmetikával valósítottam meg, a 3.4.5. alszakaszban említett problémák itt nem jelentkeztek.

Továbbá, az új program támogatja az összes lehetséges helyettesítés felsorolását, majd ezek közül a legkedvezőbb kiválasztását a 3.1. algoritmus 4–5. lépésében, futási idő tekintetében mégis felveszi a versenyt az egyszerűbb Maple változattal, amely mohó módon az első megtalált alkalmas helyettesítést szolgáltatta. Köszönhető ez a mindkét számítógépes algebra rendszer által kínált, de a Mathematicában alapvetőbb [88, 40] funkcionális programozási paradigma alkalmazásának, és a Mathematica rendszer további kedvező tulajdonságainak, mint a lista műveletek automatikus pár-

3.4. táblázat. *A saját tesztfüggvényeken elért eredmények a Maple implementációval*

Azon.	f függvény	g függvény	Helyettesítések	Fel.	Ered.
Cos1	$100\cos(x_1+x_2)$	$100\cos(y_1)$	$y_1 = x_1+x_2$	A	1
Sin1	$\sin(2x_1+x_2)$	$\sin(y_1)$	$y_1 = 2x_1+x_2$	A	1
Sin2	$2x_3\cdot\sin(2x_1+x_2)$	$2y_1$	$y_1 =$ $x_3\cdot\sin(2x_1+x_2)$	A	3
Abs1	$\lvert x_1/x_2\rvert$	$\lvert y1\rvert$	$y_1 = x_1/x_2$	B	1
Exp1	$e^{x_1+x_2}$	e^{y_1}	$y_1 = x_1+x_2$	A	1
Exp2	$2e^{x_1+x_2}$	$2y_1$	$y_1 = e^{x_1+x_2}$	A	3b
Sq1	$x_1^2x_2^2$	semmi	semmi	D	2
Sq2	$(x_1x_2+x_3)^2$	y_1^2	$y_1 = x_1x_2+x_3$	A	1
SqSin1	$(x_1+x_2)^4+$ $+26\sin(x_1+x_2)$	$y_1^4+26\sin(y_1)$	$y_1 = x_1+x_2$	A	1
SqCos1	$(x_1x_2+x_3)^2+$ $-\cos(x_1x_2)$	$y_3^2-\cos(y_1)$	$y_1 = x_1x_2$, $y_3 = y_1+x_3$	A	1,3
SqExp1	$(x_1+x_2)^2+e^{x_1+x_2}$	$y_1^2+e^{y_1}$	$y_1 = x_1+x_2$	A	1
SqExp2	$(x_1+x_2)^2+ +2e^1e^{x_1+x_2}$	$y_1^2+2e^1e^{y_1}$	$y_1 = x_1+x_2$	A	1
SqExp3	$(x_1+x_2)^2+ +2e^{1+x_1+x_2}$	semmi	semmi	A	2a

huzamosítása. Ugyanakkor az összes lehetséges helyettesítésen értelmezett keresési térre vonatkozó korlátozás és szétválasztás jellegű stratégia megalkotása gyorsítana az eljáráson, ez egy eddig kiaknázatlan továbbfejlesztési lehetőség.

Hadd említsem meg, hogy a Mathematica minden új verzióval egyre több professzionális eszközt kínál a hatékony programírás lehetőségének biztosítására. Például 2010 óta lehetővé teszi a grafikus feldolgozó egység (graphics processing unit, GPU) párhuzamos számítási kapacitásának kiaknázását is a CUDA vagy OpenCL technológia segítségével [90].

3.4.8. Előrelépés a Maple változathoz képest

A Mathematica implementáció hatékonyságát először is a saját teszthalmazomon, abból is a Maple verzió számára problémás eseteken vizsgáltam. A 3.4. táblázatban szereplő tesztesetekre a Mathematicában elért eredményt a 3.5. táblázat tartalmazza. Az utóbbiban nem szereplő esetekben a két implementáció kimenete identikus volt.

3.5. táblázat. *A problémás saját tesztfüggvényeken elért eredmények a Mathematica implementációval*

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

Megjegyezzük, hogy Csendes and Rapcsák [20] jelölésétől némileg eltérünk a következő táblázatokban. Az érdekesebb helyettesítések kiemelése érdekében az egyszerű átnevezéseket ($y_j := x_j$) nem tüntetjük fel.

Megismétlem, hogy a Maple intervallum aritmetikája kapcsán jelentkező anomáliák miatt az eredeti implementációban heurisztikát alkalmaztunk az értékkészlet-becslésre. A felmerülő hibák másik nagy csoportját a gyenge helyettesítő rutin okozta.

A következőkben részletesen kifejtem az eredményekben jelentkező különbségeket. A *Sin2* problémára az új implementáció alkalmas helyettesítést talált, míg a régi egy összetettebb, de nem monoton helyettesítést szolgáltatott.

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

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

Hasonlóképp az *SqCos1* tesztfeladatra az új, Mathematica-alapú eljárás

helyesen megállapította, hogy $y_1 = x_1 x_2$ nem monoton, ezért eltávolította a lehetséges helyettesítések listájából.

SqExp2-3 esetén ugyancsak a Maple mintaillesztés terén jelentkező gyengesége figyelhető meg, mivel $x_1 + x_2$-et felismerte $e^1 e^{x_1+x_2}$-ben, $e^{1+x_1+x_2}$-ben azonban nem. A Mathematica implementációban nem jelentkezett ez a probléma.

3.4.9. Globális optimalizálási tesztfeladatokon elért eredmények

A Mathematica implementáció vizsgálata során a standard és egyéb gyakran használt globális optimalizálási tesztfeladatokat tartalmazó teszthalmazt némileg bővítettük a korábbi publikációhoz képest [4, 3]. Az eredményeket a 3.6. táblázatban foglaltam össze, a problémák nevét a szokásos módon rövidítettem.

A legtöbb esetben az új eredmény megegyezik a korábbi megvalósításával. A két eltérést a *Schwefel-227 (Sch227)* és a *Schwefel-32 (Sch32)* függvény kétdimenziós alakjánál tapasztaltuk. A *Schwefel-227* feladatra a Maple változat az $y_1 = x_1^2 + x_2^2 - 2x_1$ helyettesítést adta. Ez a kifejezés fedi x_2-t, de nem monoton egyik változójában sem, ezért a Mathematica változat nem javasolta helyettesítésre. A Rosenbrock feladattal rokon kétdimenziós *Schwefel-32* esetén a Mathematica talált alkalmas helyettesítést, míg a Maple nem.

Ezúttal az átalakítás futási idejét is mértem, ezt a 3.6. táblázat utolsó oszlopában tüntettem fel. Minden átírást a Mathematica 9.0-es verziójával készítettem, a futási időt fél órában maximáltam. Azokban az esetekben, amikor a teljes egyszerűsítés nem zajlott le 1 800 másodpercen belül, a „Megállítva." szöveget írtam az „Új változók" és „Helyettesítések" oszlopba, és „1 800 <" került a „Futási idő" oszlopba. A numerikus teszteket egy Intel i5-3470 processzorral, 8 GB RAM-mal, és 64-bites operációs rendszerrel felszerelt számítógépen futtattam.

A teljesítmény profil értékelése alapján kijelenthetem, hogy a futási idő nagy részét – nem meglepő módon – a szimbolikus képlet-átalakítások és a kiterjesztett helyettesítő rutin igényelte. Míg a 3.5. táblázat átalakításainak mindegyike kevesebb, mint 0,2 másodperc alatt lezajlott, a standard tesztfeladatok futási ideje sokkal nagyobb szórást mutatott. A 45-ből 24 teszteset kevesebb, mint egy másodperc alatt futott. További 10 esetben kevesebb, mint egy perc elegendőnek bizonyult. Viszont 7 esetben több, mint fél órát igényelt volna az automatikus egyszerűsítő.

3.6. táblázat. *A standard optimalizálási tesztfüggvényekre vonatkozó eredmények*

Azon.	Dim.	Új változók	Helyettesítések	Fel.	Ered.	Futási idő
BR	2	$y = [x_1, y_1]$	$y_1 = x_2 - 6 + (5/\pi)x_1 - 0{,}129185x_1^2$	A	1	0,1092
Easom	2	$y = x$	semmi	D	2	0,1404
G5	5	$y = x$	semmi	D	2	2,7456
G7	7	$y = x$	semmi	D	2	36,5821
GP	2	$y = x$	semmi	D	2	0,4212
H3	3	$y = x$	semmi	D	2	16,3488
H6	6	Megállítva.	Megállítva.	D	2	1 800 <
L1	1	$y = x$	semmi	D	2	0,0312
L2	1	$y = x$	semmi	D	2	0,6552
L3	2	$y = x$	semmi	D	2	2,3088
L5	2	$y = x$	semmi	D	2	15,3348
L8	3	$y = [y_1, y_2, y_3]$	$y_1 = (x_1 - 1)/4$, $y_2 = (x_2 - 1)/4$, $y_3 = (x_3 - 1)/4$	A	1	13,1820
L9	4	$y = [y_1, y_2, y_3, y_4]$	$y_1 = (x_1 - 1)/4$, $y_2 = (x_2 - 1)/4$, $y_3 = (x_3 - 1)/4$, $y_4 = (x_4 - 1)/4$	A	1	174,7047
L10	5	Megállítva.	Megállítva.	A	1	1 800 <
L11	8	Megállítva.	Megállítva.	A	1	1 800 <
L12	10	Megállítva.	Megállítva.	A	2	1 800 <
L13	2	$y = x$	semmi	D	2	0,4992
L14	3	$y = x$	semmi	D	2	0,7800
L15	4	$y = x$	semmi	D	2	1,0296
L16	5	$y = x$	semmi	D	2	2,0904
L18	7	$y = x$	semmi	D	2	32,1517
Sch21 (Beale)	2	$y = x$	semmi	C	2	0,1248
Sch214 (Powell)	4	$y = x$	semmi	D	2	0,0936
Sch218 (Matyas)	2	$y = x$	semmi	D	2	0,0156
Sch227	2	$y = x$	semmi	D	2	0,0624
Sch25 (Booth)	2	$y = x$	semmi	C	2	0,0312
Sch31	3	$y = x$	semmi	D	2	0,0468
Sch31	5	$y = x$	semmi	D	2	0,0936
Sch32	2	$y = [y_1, y_2]$	$y_1 = x_1 - x_2^2$, $y_2 = x_2 - 1$	A	1	0,0468
Sch32	3	$y = x$	semmi	D	2	0,0312
Sch37	5	$y = x$	semmi	D	2	0,0936
Sch37	10	$y = x$	semmi	D	2	0,2808
SHCB	2	$y = x$	semmi	D	2	0,0156
THCB	2	$y = x$	semmi	D	2	0,0156
Rastrigin	2	$y = x$	semmi	C	2	0,0936
RB	2	$y = [y_1, y_2]$	$y_1 = x_1^2 - x_2$, $y_2 = 1 - x_1$	A	1	0,0440
RB5	5	$y = [x_1, x_2, x_3, x_4, y_1]$	$y_1 = x_4^2 - x_5$	A	1	0,3080
S5	4	$y = x$	semmi	D	2	225,5018
S7	4	$y = x$	semmi	D	2	1 010,2431
S10	4	Megállítva.	Megállítva.	D	2	1 800 <
R4	2	$y = x$	semmi	C	2	0,2964
R5	3	$y = [x_1, x_2, y_2]$	$y_1 = 3 + x_3$, $y_2 = (\pi/4)y_1$	A	1	13,8216
R6	5	$y = [x_1, x_2, x_3, x_4, y_2]$	$y_1 = 3 + x_5$, $y_2 = (\pi/4)y_1$	A	2	995,6883
R7	7	Megállítva.	Megállítva.	A	2	1 800 <
R8	9	Megállítva.	Megállítva.	A	2	1 800 <

Mindösszesen 45 jól ismert globális optimalizálási tesztfeladatot vizsgáltam, és ezek közül 8 esetben a Mathematica programom talált ekvivalens átírást. Más szavakkal, a módszerünk a teszthalmaz 18%-ára ajánlott valamiféle egyszerűsítést. Tekintve, hogy ezekre a feladatokra nincs más ismert módszer, amely ilyen jellegű automatikus egyszerűsítéseket állítana elő, az eredmény figyelemre méltó. A 4. fejezetben megvizsgálom, vajon a produkált átírások milyen hatást gyakorolnak egy klasszikus numerikus multi-start megoldó teljesítményére.

3.4.10. Webes demonstráció

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

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

Az alkalmazás a MathDox formula editor [59] felhasználásával WYSIWYG felületet kínál a gyakori matematikai konstrukciók begépeléséhez. A (3.1) formátumú feladat célfüggvényét begépelve némi várakozás után a weboldalon megjelenik az automatikus egyszerűsítő által ajánlott átírás.

A felületről kinyert OpenMath [67] formátumú képlet Content MathML-re [14] konvertálva kerül kiértékelésre, a képletek ízléses megjelenítéséről a MathJax [60] gondoskodik.

3.5. Elméleti eredmények

Csendes és Rapcsák [20, 71] elméleti eredményeit két irányban általánosítottam. Egyrészt a 37. oldalon bemutatott Otis modell ihletésére az alkalmas *párhuzamos helyettesítéseket* kívántam leírni, másrészt a feladathoz tartozó *feltételeket* kívántam kezelni.

Tekintsünk egy egyszerűbb példát a párhuzamos helyettesítésekre. A következő célfüggvényt minimalizáljuk, feltételek nélkül:

$$f(x_1, x_2, x_3) = (x_1 + x_2 + x_3)^2 + (x_1 - 2x_2 - 3x_3)^2.$$

Minimalizálhatjuk a $g(y_1, y_2) = y_1^2 + y_2^2$ függvényt is, ami kétváltozós, az eredeti háromváltozóssal szemben. Sem $y_1 = x_1 + x_2 + x_3$, sem $y_2 = x_1 - 2x_2 - 3x_3$ nem alkalmas helyettesítés a korábbi értelmezés szerint, mivel bár sima és monoton függvények minden változójukban, egyik változót sem fedi sem y_1,

sem y_2. Ugyanakkor y_1 az y_2-vel közösen fedi mindhárom változót, vagyis a két részkifejezés *együtt* x_1, x_2, és x_3 minden előfordulását jellemzi. Továbbá $H = \{x_1 + x_2 + x_3, x_1 - 2x_2 - 3x_3\}$ a helyettesítések egy helyes halmaza, a két helyettesítés egyszerre is végrehajtható, ráadásul dimenziócsökkentést eredményez a fent említett problémán. Ilyen jellegű párhuzamos helyettesítések leírását célozza az elmélet általánosítása.

Az eredeti, átírás előtti nemlineáris optimalizálási feladat most feltételeket is tartalmazhat:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & f(\mathbf{x}) \\ c_i(\mathbf{x}) \quad & = 0 \\ c_j(\mathbf{x}) \quad & \leqslant 0, \end{aligned} \tag{3.3}$$

ahol $f(\mathbf{x}), c_i(\mathbf{x}), c_j(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$ sima, valós, képlettel adott függvények, továbbá $i = 1, \ldots, p_1$ és $j = p_1 + 1, \ldots, p_2$ egészértékű indexek.

Az átalakított feltételes optimalizálási feladat a

$$\begin{aligned} \min_{y \in \mathbb{R}^m} \quad & g(\mathbf{y}) \\ d_i(\mathbf{y}) \quad & = 0 \\ d_j(\mathbf{y}) \quad & \leqslant 0. \end{aligned} \tag{3.4}$$

Itt $g(\mathbf{y}) : \mathbb{R}^m \to \mathbb{R}$ az $f(\mathbf{x})$ átírt formája, $d_i(\mathbf{y}), d_j(\mathbf{y}) : \mathbb{R}^m \to \mathbb{R}$ ismét sima, valós függvények, méghozzá a $c_i(\mathbf{x}), c_j(\mathbf{x})$ feltételek átírásai, $i = 1, \ldots, p_1$ és $j = p_1 + 1, \ldots, p_2$.

Jelölje X az $f(\mathbf{x})$ célfüggvényben és $c_k(\mathbf{x})$, $k = 1, \ldots, p_2$ feltételekben előforduló változókat jelölő szimbólumok halmazát. Y az átírás után kapott $g(\mathbf{y})$ és $d_k(\mathbf{y})$, $k = 1, \ldots, p_2$ függvényekben szereplő változók szimbólumainak a halmaza. Mivel a dimenziószám növelését most nem engedjük meg a transzformációs lépésben, $m \leqslant n$ és $|Y| \leqslant |X|$. Az egyszerűsítő eljárás futásának kezdetén $Y := X$.

Legyen C az eredeti korlátok baloldalain álló kifejezések halmaza:

$$C := \{c_k(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}\,,\ k = 1, \ldots, p_2\}.$$

Az F jelölje azt a halmazt, ami $f(\mathbf{x})$ és $c_k(\mathbf{x}) \in C$ függvények összes részkifejezéseit (vagyis az eredeti kifejezések részeként előforduló minden, szintaktikailag helyes kifejezést) tartalmazza.

Az algoritmusunk döntő része a *transzformációs, avagy átírási lépés*. Ha egy $H \subset F$ kifejezés-halmaz fedi a $V \subseteq X$ változó-halmazt (tehát a $v \in V$

változók egyike sem fordul elő H kifejezésein kívül az $f(\mathbf{x})$, vagy $c_k(\mathbf{x}) \in C$ képletében), továbbá $|H| \leqslant |V|$, akkor alkalmazzunk minden H-hoz kapcsolódó helyettesítést $f(\mathbf{x})$-re és C-re a következőképpen. Helyettesítsünk egy új y_i változót minden $h_i(\mathbf{x}) \in H$ helyébe az $f(\mathbf{x})$ képletében, és hasonlóképp minden $c_k(\mathbf{x}) \in C$ képletében. Továbbá frissítsük a változók halmazát az átírt problémában: $Y := (Y \cup y_i) \setminus V$.

A fenti mozzanatra (a H-hoz kapcsolódó) transzformációs lépésként fogunk hivatkozni. Abban a speciális esetben, ha $|H| = |V| = 1$ és $p_1 = p_2 = 0$, visszakapjuk Csendes és Rapcsák [20] algoritmusát a feltétel nélküli esetre.

Az $\mathbf{y} := H(\mathbf{x})$ jelölést az $y_i := h_i(\mathbf{x})$, $i = 1, \ldots, |H|$ rövidítéseként használjuk. A következő állítás az 1. állítás [20] természetes általánosításaként adódik.

3. ÁLLÍTÁS Ha az x_i változó a sima $f(\mathbf{x})$ és $c_k(\mathbf{x})$, $k = 1, \ldots, p_2$ függvényekben mindenhol a $h(\mathbf{x})$ kifejezés részeként fordul elő, akkor ezen függvények x_i szerinti parciális deriváltjai felírhatók a $(\partial h(\mathbf{x})/\partial x_i)\, p(\mathbf{x})$ formában, ahol $p(\mathbf{x})$ folytonosan differenciálható.

Emlékeztetőül, helyettesítéseknek egy jólrendezett H halmazát *helyes*nek neveztük, ha minden $h_i(\mathbf{x}) \in H$ helyettesítés egyszerre, párhuzamosan végrehajtható. A jólrendezettség csupán a változók egyértelmű indexeléséhez szükséges.

4. TÉTEL Amennyiben teljesül, hogy H helyettesítéseknek egy helyes halmaza, és minden $h_i(\mathbf{x}) \in H$ olyan sima valós függvény, amely szigorúan monoton minden $v \in V \subseteq X$ változó függvényében, továbbá H elemszáma nem több V elemszámánál, és $h_i(\mathbf{x})$ értelmezési tartománya minden $h_i(\mathbf{x}) \in H$-ra egyenlő $\mathbb{R}$-rel, akkor a H-hoz kapcsolódó transzformációs lépés oly módon egyszerűsíti az eredeti problémát, hogy az eredeti feladat minden $\mathbf{x}^*$ helyi minimum (maximum) helye az átalakított probléma egy $\mathbf{y}^*$ helyi minimum (maximum) helyévé transzformálódik.

Bizonyítás. Tekintsük a két probléma lehetséges megoldásainak halmazát. A helyettesítések egyenletei biztosítják, hogy ha egy $\mathbf{x}$ pont a (3.3) probléma lehetséges megoldása, akkor az átalakítás után $\mathbf{x}$-ből az új, egyszerűsített (3.4) problémának egy lehetséges megoldását kapjuk. Hasonló megállapítás tehető az infízibilis pontokra is.

Jelölje most $\mathbf{x}^*$ az $f(\mathbf{x})$ egy helyi minimum pontját, és legyen $\mathbf{y}^* := H(\mathbf{x}^*)$, tehát $\mathbf{y}^*$ az $\mathbf{x}^*$ átírt alakja a H-hoz kapcsolódó helyettesítések elvégzése után. Mivel minden $h_i(\mathbf{x}) \in H$ folytonos és szigorúan monoton legalább egy változóban, így minden pont az $\mathbf{x}^*$-nak az $N(\mathbf{x}^*, \delta)$-val jelölt δ-széles környezetéből az $\mathbf{y}^*$-nak egy $N(\mathbf{y}^*, \epsilon)$-nal jelölt ϵ-széles környezetébe lesz transzformálva, és

$\forall \mathbf{x} \notin N(\mathbf{x}^*, \delta) : H(\mathbf{x}) \notin N(\mathbf{y}^*, \epsilon)$. Az $f(\mathbf{x})$ és $g(\mathbf{y})$ célfüggvényeknek, valamint az eredeti és átírt feltételi függvényeknek a helyettesítési értéke megegyezik az átírás előtt és után. Ez a tény biztosítja, hogy minden $\mathbf{x}^*$ helyi minimum pont a $g(\mathbf{y})$ egy $\mathbf{y}^*$ helyi minimum pontjába transzformálódik. Hasonló gondolatmenet alkalmazható a helyi maximum pontokra is.

Továbbá, mivel $|H| \leqslant |V|$, a transzformációs lépés konstrukciója biztosítja azt, hogy minden iterációban legalább annyi x_i változót veszünk ki az optimalizálási modellből, amennyi új változót bevezetünk az összes, H-hoz kapcsolódó helyettesítés alkalmazása esetén. □

A 4. tétel olyan, egyszerűsítést eredményező átalakításokra fogalmaz meg elégséges feltételeket, amelyek az eredeti probléma minden helyi minimum (maximum) helyét megőrzik, és az átírt alak helyi minimum (maximum) pontjaiba transzformálják. Ezzel szemben a következő tétel a szélsőértékek kölcsönösen egyértelmű leképezését biztosító helyettesítésekre fogalmaz meg elégséges feltételeket:

5. TÉTEL Amennyiben teljesül, hogy H helyettesítéseknek egy helyes halmaza, és minden $h_i(\mathbf{x}) \in H$ olyan sima valós függvény, amely szigorúan monoton minden $v \in V \subseteq X$ változó függvényében, továbbá H elemszáma nem több V elemszámánál, és $h_i(\mathbf{x})$ értelmezési tartománya és értékkészlete minden $h_i(\mathbf{x}) \in H$-ra egyenlő $\mathbb{R}$-rel, akkor a H-hoz kapcsolódó transzformációs lépés oly módon egyszerűsíti az eredeti problémát, hogy az átalakított probléma minden $\mathbf{y}^*$ helyi minimum (maximum) helye előáll az eredeti probléma egy $\mathbf{x}^*$ helyi minimum (maximum) helyének átírásával.

Bizonyítás. Mivel a helyettesítések egyenletei megőrzik a feltételi függvények helyettesítési értékét, az átalakított (3.4) feladat bármely $\mathbf{y}$ lehetséges megoldása előáll az eredeti (3.3) feladat egy lehetséges megoldásának az átírásával. Ezt így jelöljük: $\exists \mathbf{x} : \mathbf{y} = H(\mathbf{x})$. Hasonló megállapítást tehetünk az infízibilis pontokra is.

Jelölje az átalakított (3.4) feladat egy helyi minimum helyét $\mathbf{y}^*$. Mivel minden H-ban lévő helyettesítés függvénye folytonos és értékkészlete $\mathbb{R}$, (3.4) bármely $\mathbf{y}^*$ helyi minimum helye szükségképpen előáll egy helyettesítés folytán. Legyen most $\mathbf{x}^*$ a visszaalakított pont, vagyis a helyettesítést leíró függvényben $\mathbf{y}^*$ őse: $\mathbf{y}^* = H(\mathbf{x}^*)$. Tekintsük $N(\mathbf{y}^*, \delta)$-t, ahol minden $\mathbf{y}$ fízibilis pont helyettesítési értéke nagyobb, vagy egyenlő, mint $\mathbf{y}^*$ helyettesítési értéke: $g(\mathbf{y}) \geqslant g(\mathbf{y}^*)$. Tekintsük továbbá (3.3) egy $\mathbf{x}$-el jelölt lehetséges megoldását, amelyre $H(\mathbf{x}) = \mathbf{y} \in N(\mathbf{y}^*, \delta)$. Ezen $\mathbf{x}$-ek száma végtelen lehet, ha az átírás csökkenti a feladat dimenzióját. Bárhogyan is, létezik $\mathbf{x}^*$-nak egy megfelelő $N(\mathbf{x}^*, \delta')$ környezete, ahol az $f(\mathbf{x}) \geqslant f(\mathbf{x}^*)$ reláció fennáll minden

$\boldsymbol{x} \in N(\boldsymbol{x}^*, \delta')$ lehetséges megoldásra. Másként fogalmazva, $\boldsymbol{x}^*$ (3.3) egy helyi minimum pontja.

Hasonló gondolatmenet alkalmazható a helyi maximum pontokra is. □

4

AZ AUTOMATIKUS ÁTÍRÁS NUMERIKUS SOLVERRE GYAKOROLT HATÁSA

Ebben a fejezetben azt vizsgáljuk, hogy a 3. fejezetben bemutatott automatikus szimbolikus egyszerűsítő által produkált átírások milyen hatást gyakorolnak egy numerikus globális optimalizáló eljárásra. Név szerint a GLOBAL-t [19] vizsgáltam. Ez egy sztochasztikus multi-start megoldó, amely a 2.3.3. alszakaszban már részletesebben bemutatásra került.

A numerikus tesztelés során összehasonlítottam a globális optimalizáló eredményeit az eredeti és az átalakított probléma-alakra a 3.5. és 3.6. táblázatokban szereplő átírások esetében. Az elért globális optimum értékét, a futási időt, és a függvény-kiértékelések számát vizsgáltam, ezeket rendre a 4.1–4.3. táblázatok összegzik. A jobb eredményt minden sorban félkövér szedéssel jelöltem.

A teszteket a GLOBAL [19] nevű általános multi-start megoldó MATLAB implementációjával készítettem, a BFGS kvázi-Newton típusú helyi kereső módszer alkalmazásával. Minden tesztesetre 100 független futtatást csináltam a 3.4.9. alszakasz időméréseihez is használt számítógépen. Az Intel i5-3470 processzorral, 8 GB RAM-mal, és 64-bites operációs rendszerrel felszerelt számítógépen 64 bites MATLAB R2011b állt rendelkezésre. A GLOBAL az alapértelmezett paraméterekkel futott.

A megoldó minden változóra igényelte egy kezdeti keresési intervallum megadását, ezért a 3.5. táblázatban szereplő saját tesztfeladatokban minden változóra a $[-100, 100]$ intervallumot állítottam be, míg a standard problémáknál a szokásos befoglaló intervallumokat használtam (ld. például [68] A. függelékét). Az átalakított feladatoknál a befoglaló intervallumot is transzformáltam az átírásnak megfelelően.

A 4.1. táblázat a GLOBAL által elért optimum értékeit tartalmazza az imént említett paraméterezés mellett. *Fbest* a 100 futtatás során talált legkedvezőbb optimum értéket, *Fmean* az átlagos optimum értéket, *Fvar* pedig az elért optimumok szórását mutatja. A valós globális optimumot 15-ből 8 tesztesetben minden független futás esetén megtalálta az eljárás. A *Rosenbrock*

4.1. táblázat. *A* GLOBAL *által talált optimális függvényértékek*

Azon.	Eredeti feladat			Átalakított feladat		
	Fbest	Fmean	Fvar	Fbest	Fmean	Fvar
Exp1	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
Exp2	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
Sq2	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
SqCos1	−1,0000	−0,7671	0,1899	−1,0000	**−0,9922**	**0,0700**
SqExp2	3,0000	3,0000	0,0000	3,0000	3,0000	0,0000
SqExp3	3,0000	3,0000	0,0000	3,0000	3,0000	0,0000
CosExp	−2,0000	**−1,6166**	0,4397	−2,0000	−1,5896	**0,2781**
BR	0,3979	0,3979	0,0000	0,3979	0,3979	0,0000
L8	0,0000	**2,1386**	**5,6861**	0,0000	2,2651	6,8558
L9	0,0000	2,4410	**8,7591**	0,0000	**2,3897**	10,1134
RB	0,0000	19,7318	1 094,1167	0,0000	**0,0000**	**0,0000**
RB5	0,0000	1,8510	3,8703	0,0000	**1,8400**	**3,8677**
R5	0,0000	1,9017	26,3097	0,0000	**0,0000**	**0,0000**
R6	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000
Sch3.2	0,0000	0,0000	0,0000	0,0000	0,0000	0,0000

(RB) és *Ratz-5 (R5)* feladatoknál csak az átírás segítségével sikerült mind a 100 esetben elérni a minimumot. A 15-ből összesen 5 esetben az átalakított alakot könnyebben oldotta meg a GLOBAL, 8 esetben egyforma nehézségű volt a kétféle alak minimalizálása, azonban 2 esetben az eredeti alak némileg kedvezőbbnek bizonyult.

Érdemes összehasonlítani a megoldó által egy futás során igényelt függvény-kiértékelések számát is, amint azt a 4.2. táblázatban megtaláljuk. A *NumEvalmean* oszlop a függvény-kiértékelések számának átlagát tartalmazza, a *NumEvalvar* oszlop pedig ugyanezen mutató szórását. Az átalakított alak 15-ből 12 esetben kevesebb függvény-kiértékelést igényelt, és néhány esetben az eredeti alakot jelentősen felülmúlta. Például a *Rosenbrock* feladat optimumának megtalálásához a GLOBAL-nak csupán az eredetileg szükséges kiértékelések 17%-ára volt szüksége. A *Branin (BR)* esetében ugyanez az arány 80%, az *Sq2* feladatnál pedig csupán 11%. Úgy tűnik, a *Levy-8 (L8)* és *Ratz-6 (R6)* feladatok eredeti felírása valamivel kedvezőbb volt a függvény kiértékelések száma tekintetében. A *Levy-8*-nál az optimum megtalálása

4.2. táblázat. *A* GLOBAL *függvény-kiértékeléseinek a száma*

Azon.	Eredeti feladat		Átalakított feladat	
	NumEvalmean	NumEvalvar	NumEvalmean	NumEvalvar
Exp1	87	2 280	**51**	**188**
Exp2	102	2 489	**55**	**327**
Sq2	478	57 927	**52**	**38**
SqCos1	1 467	463 242	**1 131**	**279 915**
SqExp2	155	4 903	**61**	**142**
SqExp3	151	5 282	**61**	**142**
CosExp	1 110	1 805,418	**631**	**154 691**
BR	136	1 504	**115**	**769**
L8	**785**	266 138	797	**242 593**
L9	2 606	1 838 627	**2 371**	**1 343 151**
RB	749	71 762	**127**	**976**
RB5	3 162	**693 878**	**2 634**	709 652
R5	**1 908**	1 644 365	2 957	**581 382**
R6	**6 001**	**223 269**	6 069	269 551
Sch3.2	119	1 290	**59**	**121**

tekintetében is az eredeti alak győzött, ugyanakkor a nagyobb *Levy-9 (L9)* feladatnál már az átalakított forma mutatta magát kedvezőbbnek. A *Ratz-5 (R5)* eredeti alakja kevesebb függvény-kiértékelést igényelt, azonban nem tette lehetővé, hogy a megoldó minden futás alkalmával megtalálja a globális optimumot. Az átírt alak igen.

Az összes tesztfeladat átlagában a függvény-kiértékelések számának az átírásnak köszönhető relatív javulása 32,0% volt. Ugyanakkor ez a mutató a saját tesztfeladataink átlagában kedvezőbb (51,8%), a standard problémákon kevésbé jó (14,7%).

A futási idők tekintetében (ld. a 4.3. táblázatot), az átalakított feladat-alak szinte minden esetben egy kicsivel gyorsabb futást tett lehetővé a GLOBAL számára. Az egész teszthalmazra a futási idő átlagos relatív javulása 31,5%. E mutató a saját tesztfeladataink átlagában 56,9%, a standard problémákra 9,3%.

Megállapíthatjuk, hogy a 3.5. és 3.6. táblázatokban szereplő, nagyon egyszerűnek látszó ekvivalens átalakítások alkalmazása is jelentős hatást

4.3. táblázat. *A* GLOBAL *futási ideje (másodpercben)*

Azon.	Eredeti feladat		Átalakított feladat	
	Tmean	Tvar	Tmean	Tvar
Exp1	0,0289	0,0004	**0,0113**	**0,0000**
Exp2	0,0346	0,0005	**0,0124**	**0,0000**
Sq2	0,0919	0,0029	**0,0072**	**0,0000**
SqCos1	0,1822	0,0083	**0,1406**	**0,0047**
SqExp2	0,0270	0,0002	**0,0088**	**0,0000**
SqExp3	0,0264	0,0002	**0,0088**	**0,0000**
CosExp	0,2139	0,0429	**0,1623**	**0,0134**
BR	0,0241	0,0001	**0,0195**	**0,0000**
L8	0,0992	0,0046	**0,0990**	**0,0040**
L9	0,2771	0,0220	**0,2445**	**0,0150**
RB	0,0857	0,0009	**0,0241**	**0,0001**
RB5	0,2705	0,0050	**0,2089**	**0,0045**
R5	**0,2407**	0,0286	0,4567	**0,0159**
R6	0,7830	0,0136	**0,7785**	**0,0047**
Sch3.2	0,0187	0,0001	**0,0116**	**0,0000**

gyakorolhat egy hagyományos, numerikus globális optimalizáló teljesítményére.

5

MÉLTÁNYOS SÁVSZÉLESSÉG-KIOSZTÁS BITTORRENT HÁLÓZATOKBAN

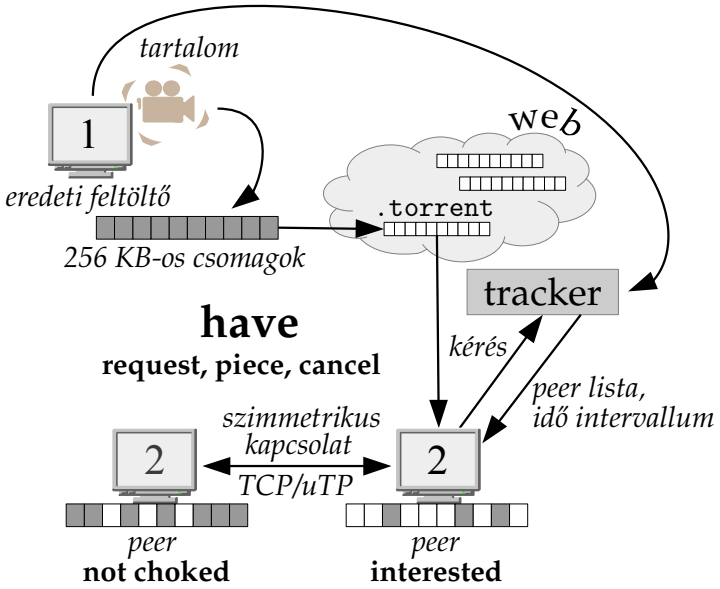
Ebben a fejezetben bemutatom a max-min méltányos sávszélesség-kiosztás problémájára általunk adott egzakt matematikai programot és algoritmust [5]. A feladatot többrajos (multiswarm) peer-to-peer tartalom-megosztó közösségekben vizsgáltuk. Az ajánlott iteratív módszer nagyméretű LP és MILP problémák megoldását is magában foglalja.

Valós BitTorrent hálózatokból származó adathalmazokon végrehajtott numerikus kísérletek demonstrálják az új algoritmus képességeit. A tesztek alapján nagyméretű feladatokra a korábbi megoldásnál gyorsabban, numerikusan stabilabb megoldást szolgáltat az eljárásunk. Továbbá akkor is jó közelítését adja a max-min méltányos erőforrás-kiosztásnak, ha a futtatást néhány kezdőlépés után leállítjuk.

5.1. Bevezetés

A BitTorrent az egyik legnépszerűbb, az Internet használóinak milliói által működtetett tartalom-megosztó protokoll [74]. Egyenrangú, úgynevezett peer-to-peer (P2P) hálózati technológián alapszik, tehát a csomópontok (úgynevezett peerek) szerver és kliens szerepet egyaránt betöltenek (szemben a központosított megoldásokkal, ahol ezek a szerepek elválnak egymástól). A decentralizált megközelítés nagy hatékonyságot és extrém skálázhatóságot eredményez [17].

Egy BitTorrent hálózat alapvetően rajok, angol szakkifejezéssel *swarm*ok halmazaként írható le. A swarm az egy adott tartalmat letöltő peerek csoportja, ezen belül a felhasználóknak két típusát különböztetjük meg: a letöltőket (*leecher*ek), és megosztókat (*seeder*ek). A *tartalom* valamilyen adatfájl, amit a résztvevők egymás között megosztanak. A megosztók azok a felhasználók, akik a tartalom egészét birtokolják, online jelen vannak, és szándékukban is



5.1. ábra. *A BitTorrent protokoll működése*

áll a megosztás. Letöltőknek azokat nevezzük, akik a tartalmat éppen aktívan töltik le maguknak.

Az 5.1. ábra szemlélteti a *BitTorrent protokoll* [16] működését. Az eredeti feltöltő (az ábrán „1”-gyel jelölve) a protokollt követve a tartalmat kicsi, alapértelmezésben 256 KB-os csomagokra osztja, majd készül egy `.torrent` kiterjesztésű leíró fájl. Ez egyebek mellett tartalmazza a csomagok azonosítóit és az úgynevezett *tracker* elérhetőségét, ahonnan a letöltés elindításához szükséges adatokat a letöltésben érdekelt személy beszerezheti. A leíró fájl kis méretű, megosztható például a weben. A potenciális letöltő („2”) a trackertől kérheti el az adott fájlt megosztó peerek egy listáját, ide természetesen az eredeti feltöltő regisztrálja, hogy rendelkezik a szóban forgó tartalommal. A válasz általában a swarm résztvevőinek egy valódi, elegendően bő részhalmaza (pl. 200 peer elérhetősége [50]). Ezután a peerek szimmetrikus kapcsolatot építenek ki egymás között (TCP vagy uTP felett), és standardizált üzenetekben kommunikálnak.

A leecherek a tartalmat csomagonként kérik el, a „Rarest First”, lefordítva „legritkábbat először” [51] kiválasztási szabályt követve. Az `interested` kulcsszóval kezdődő üzenetek egy csomagra vonatkozó kérést jelentenek egy konkrét peer felé. Amint egy darab teljes egészében megérkezik, a letöltő köteles azt egy `have` üzenettel a swarm többi résztvevőjének a tudtára hozni. Innentől kezdve ezt a csomagot ő is továbbíthatja másoknak, ily módon egy-

ben feltöltővé válva. A letöltőknek ez a kettős szerepe eredményezi a BitTorrent robusztusságát, és emiatt különböztetik meg az angolban a BitTorrentes „leechert" a szokásos „downloader"-től, a „seeder"-t az „uploader"-től. A „Rarest First" szabály gondoskodik róla, hogy minden csomag a swarm lehető legtöbb peerjénél meglegyen, ezáltal elkerülve bármelyikük elkallódását.

A BitTorrent protokoll fontos eleme továbbá a beépített ösztönző rendszer, ami a „szemet szemért" elvén működő úgynevezett „choking" algoritmus által valósul meg [17]. Ez a mechanizmus biztosítja, hogy amíg egy peer a letöltő módban van, érdeke fűződjön a megosztott tartalom csomagjainak feltöltéséhez a számára feltöltött csomagokért cserébe.

Habár a rendszer jól működik a letöltő (leeching) fázis alatt, nincs olyasfajta jutalmazó mechanizmus a BitTorrentben, ami a megosztókat (seedereket) motiválná. Tehát elméletileg egy letöltő bármilyen következmény nélkül megtehetné, hogy amint a letölteni kívánt tartalom utolsó darabkáját is megkapta, egyszerűen elhagyja a swarmot.

A probléma lehetséges megoldásai közül az egyik legnépszerűbb a *privát BitTorrent közösség* [94], habár ennek alapötlete részben szemben áll a peer-to-peer rendszer decentralizáltságának elvével. A konstrukció lényege, hogy minden felhasználó egyénileg regisztrált fiókkal rendelkezik egy dedikált szerveren, és néhány előre lefektetett szabályt betart. Ilyen szabály lehet például egy meghatározott *feltöltési arány* biztosítása valamilyen időintervallum átlagában. Egy felhasználó feltöltési aránya a feltöltött és letöltött adatmennyiség hányadosaként, fájlonként is értelmezhető. Ezeket a mutatókat aztán a szintén trackernek nevezett szerveren tárolják, a szabályokat be nem tartó felhasználó pedig korlátozásoknak, ill. szélsőséges esetben a közösségből való kizárásnak nézhet elébe. Tanulmányok szerint a privát BitTorrent közösségek a nyílt BitTorrent hálózatokkal összehasonlítva nagyobb letöltési sebességet és a tartalmak jobb elérhetőségét biztosítják [61].

Érdekesség, hogy a weben is gyakran megtalálható `.torrent` fájlok alapján sejthető, de nem dönthető el egyértelműen, hogy mi a megosztani kívánt tartalom. A tartalomszolgáltató személye pedig csak költséges nyomozással deríthető ki [50, 54]. A BitTorrent ezen tulajdonságai több tekintetben is visszaélésre adnak lehetőséget.

Egyrészt, szerzői joggal védett tartalom illegális megosztása nehezen bizonyítható, ezért vitatható arányban, de biztosan elérhetőek ilyen tartalmak a népszerű BitTorrent oldalakon [18]. Másrészt, Cuevas és szerzőtársai [21] vizsgálatai alapján meglepően nagy arányban vannak jelen a letöltőkre kár-

tékony tartalmak is. Jelen értekezésnek azonban nem célja sem társadalmi, sem gazdasági, sem jogi oldalról vizsgálni a BitTorrent felhasználását.

A legtöbb használatban lévő kliens program megengedi a felhasználóknak, hogy egy időben több swarmban is részt vegyenek, akár letöltő, akár megosztó szerepben. Ez a tény motiválta a *rajközi erőforrás-kiosztás problémájá*nak (inter-swarm resource allocation problem, RAP) vizsgálatát a BitTorrentben. Ez általában egy vegyes-egészértékű nemlineáris optimalizálási feladat [13], speciális esete a *max-min méltányos sávszélesség-kiosztás problémája*. Ebben az optimalizálási feladatban a cél alapvetően egy olyan sávszélesség-kiosztás megtalálása, ami a lehető legtöbb felhasználónak elégséges letöltési sebességet biztosít.

Habár a BitTorrentet eredetileg nem peer-to-peer *videóközvetítésre (streaming)* tervezték, többen vizsgálták az ilyen irányú hasznosítás lehetőségét, és javasolták a protokoll ilyen célú módosításait (ld. például [63, 85, 95]). Ebben a tekintetben a max-min méltányos sávszélesség-kiosztás azon felhasználók számának maximalizálását célozza, akik a közvetítéshez elegendő letöltési sebességet kapnak, a felhasználók számára a lehetséges legjobb minőség élményét biztosítva. Ez a modell megengedi a különböző adottságokkal rendelkező felhasználók számára eltérő minőségű közvetítés biztosítását, miközben az alacsony sebességű adatfolyamot tapasztaló felhasználók számát minimalizálja. Capotă és szerzőtársai [13] ezt a problémát vizsgálták, és megoldásként egy komplikált iteratív algoritmust adtak.

Közlésre elfogadott, de egyelőre csak online megjelent cikkünkben [5] újra megvizsgáltuk ezt az érdekes feladatot azzal a céllal, hogy egzakt matematikai programozási formalizmust adjunk. *Ebben a fejezetben* bemutatom az elért eredményeket: az új, egzakt felíráson alapuló algoritmust, és ennek valós BitTorrent közösségekből származó, nagyméretű feladatokon mért teljesítményét.

Az 5.2. szakaszban a kapcsolódó szakirodalom bemutatására kerül sor. Az 5.3. szakasz összegzi a szükséges definíciókat, valamint az általunk használt gráf modell jelöléseit. Ezután az 5.4. szakaszban bemutatom a max-min méltányos sávszélesség-kiosztás kiszámítására általunk kidolgozott algoritmus lépéseit. Megadom a modell egy célszerű ekvivalens átírását, valamint az algoritmus helyességének bizonyítását. Végül az 5.5. szakaszban az AMPL nyelven megvalósított algoritmus teljesítményét vizsgáló numerikus kísérleteket közlök, amelyekben összehasonlítjuk azt a Capotă és szerzőtársai által korábban ugyanerre a célra fejlesztett algoritmussal.

5.2. Kapcsolódó munkák

A méltányosság koncepciója, és különösen a max-min méltányosság a számítógép-hálózatok általános szakirodalmában gyakran előkerül (ld. például [8, 37, 42, 82]). A peer-to-peer hálózatok, kiváltképp a BitTorrent-szerű rendszerek vonatkozásában azonban jelentősen kevesebb publikáció foglalkozik a témával.

Max-min méltányosság peer-to-peer hálózatokban. Ma és szerzőtársai [56] egy erőforrás-licit mechanizmust fejlesztettek a max-min méltányosság biztosítására. A cikk értékes eredménye, hogy a módszer ösztönző jellegű, vagyis két azonos licittel versenyben lévő csomópont számára, ha azok a peer-to-peer hálózathoz aktuálisan különböző mértékben járulnak hozzá, különböző mértékben juttat erőforrásokat. A mi megközelítésünk nem foglalkozik közvetlenül a BitTorrent már említett ösztönző rendszerével, mivel az a protokoll csomag szintjén valósul meg. Feltesszük ugyanakkor, hogy a résztvevő peerek betartják az adott BitTorrent közösség szabályait, vagyis nincsenek büntetésben. Fontos megemlíteni, hogy a korábbi eredmények [13] szerint a standard BitTorrent a max-min méltányosság mércéje szerint szuboptimális sávszélesség-kiosztást produkál.

Yan és szerzőtársai [93] a peer-to-peer hálózatok optimális erőforrás-kiosztásának és belépés vezérlésének egy elméleti keretrendszerét adták meg. Az ajánlott megközelítés nyilvánosan megfigyelhető és ellenőrizhető információkat használ az optimális erőforrás-kiosztás elérése érdekében. A mi munkánk több tekintetben különböző. Először is, mi a sávszélesség kiosztására koncentrálunk, ami tartalom-megosztó rendszerekben a legfontosabb erőforrás. Másodszor, a mi modellünk rajközi (inter-swarm) szinten működik, ami az egyik legkomplexebb vizsgálható szintje a felhasználók viselkedésének, akik a legtöbb esetben egyszerre több tartalom le- és feltöltésében vesznek részt. Továbbá, mi valós BitTorrent közösségek vizsgálatából származó nagyméretű adathalmazokon végeztük kísérleteinket.

BitTorrent-szerű rendszerek. Fan és szerzőtársai [24] megmutatták, hogy a BitTorrent-szerű rendszerekben egy alapvető konfliktus feszül a méltányosság és a jó letöltési arányok biztosítása között. A méltányosság mércéjeként az úgynevezett fairness indexet [42] használták, ami azt fejezi ki, mennyire egyenletes a peerek feltöltési aránya. A cikk a max-min méltányos allokációt sebesség-hozzárendelési stratégiaként kezeli. Modelljük a peerek teljes letöltési sebességét figyelembe veszi. Hasonló alapról indul Eger és Killat [23] is.

A mi modellünk ezzel szemben a peerek minden egyes letöltő munkamenetére (session) optimalizált megoldást szolgáltat. Ez alapvetően azt jelenti, hogy a rendelkezésre álló mért adatokból kiindulva nagyobb mélységben, az élő streamek teljesítményének optimalizálása szintjén vizsgáljuk a problémát.

Wu és szerzőtársai [91] megközelítése a miénkhez hasonló, azonban az ő megoldásuk heurisztikus, míg mi egy elméleti alapokon nyugvó egzakt matematikai modellt adunk. Más helyen Wu és szerzőtársai [92] egy olyan elosztott algoritmust publikálnak, amely a peer-to-peer élő videó közvetítésben a megcélzott általános közvetítési sebességet fenntartja, miközben az eredményt a méltányoshoz közelíti. Ők egy videó szerverekkel kiegészített dinamikus peer-to-peer rendszert vizsgálnak, ami egy nemlineáris optimalizálási problémát eredményez. A mi munkánk alapvetően különböző, mert egy statikus, központi komponens nélküli rendszert vizsgálunk. Az általunk használt hálózati folyam modell továbbá lehetővé teszi hatékony lineáris programozási technikák használatát a feladat bizonyos szintjein.

5.3. Definíciók

Ebben a szakaszban bevezetjük a legszükségesebb definíciókat, amik egyrészt meghatározzák a feladat felhasználási területét, másrészt leírják a vizsgált optimalizálási algoritmus bemeneteként szolgáló folyam hálózatot.

5.3.1. Jelölések

Mint azt a bevezetőben említettük, a vizsgált feladat egy elosztott tartalom-megosztó rendszerben felmerülő erőforrás-kiosztás problémaköréhez tartozik. Ez a rendszer a BitTorrent. Jelen munka szempontjából a rendszernek három fő komponense van: letöltők (leecherek), megosztók (seederek) és a megosztott fájlok (ezeket gyakran torrenteknek is nevezzük, ami valójában a megosztott tartalom technikailag fontos jellemzőit leíró meta fájl, de az elnevezés nem lesz félreérthető). A BitTorrent a fájlokat darabokra osztja. Egy adott fájlra nézve a megosztók halmazát azon felhasználók alkotják, akik rendelkeznek a fájl összes darabjával. Fontos szempont, hogy a letöltők is tudnak egymás között csomagokat cserélni, így a letöltés közben egyben feltöltőként is szolgálják a rendszer működését. Pontosan ez az az ötlet, amitől a BitTorrent rendkívül jól skálázható [17]. A továbbiakban *BitTorrent közösség* alatt felhasználók (leecherek és seederek), valamint fájlok egy rögzített halmazát értjük.

Capotă és szerzőtársai [13] nyomán egy BitTorrent közösség aktuális

állapotát – vagyis, hogy egy adott időpillanatban ki kinek tölthet fel, milyen adatátviteli korlátok érvényesek, stb. – egy speciális *páros gráf reprezentáció*val írhatjuk le. Ezt az irányított, súlyozott páros gráfot $G = (\{U, L, D\}, E, f, c)$ jelöli. A közösség alapvető elemei a *felhasználók halmaza* (I) és a *torrentek halmaza* (T). Minden $i \in I$ felhasználó rendelkezik μ_i *feltöltési kapacitás*sal és δ_i *letöltési kapacitás*sal, továbbá

$U = \{ u_i \mid i \in I \}$: a *feltöltő csúcsok* halmaza, ahol u_i az i felhasználó feltöltési (seeding vagy leeching) potenciálját reprezentálja;

$D = \{ d_i \mid i \in I \}$: a *letöltő csúcsok* halmaza, ahol d_i az i felhasználó letöltési (leeching) potenciálját reprezentálja;

$L = \left\{ l_i^t \mid i \in I,\, t \in T \right\}$: a *leeching csúcsok* halmaza, ahol az l_i^t (úgynevezett *leeching session*) létezése azt jelöli, hogy az i felhasználó éppen leech-eli, letölti a t torrentet;

E**:** az élek halmaza; $E = E_U \cup E_D$, ahol $E_U = \bigcup_{i,j,t}(u_i, l_j^t)$ a *feltöltő élek* halmaza, és $E_D = \bigcup_{j,t}(l_j^t, d_j)$ a *letöltő élek* halmaza;

$c : U \cup L \cup D \to \mathbb{N}$**:** a *kapacitás függvény*, amely a résztvevők sávszélesség-korlátait reprezentálja :

$$c(u_i) = \mu_i, \quad c(d_i) = \delta_i, \quad c(l_i^t) = \infty;$$

$f : E \to \mathbb{R}^+$**:** a *folyam függvény*, a kiosztott sávszélességet reprezentálja azokon az éleken, amelyek kielégítik a *folyam-megmaradási tulajdonság*ot:

$$\sum_{u_i \in U} f(u_i, l_j^t) = f(l_j^t, d_j) \qquad \forall l_j^t \in L,$$

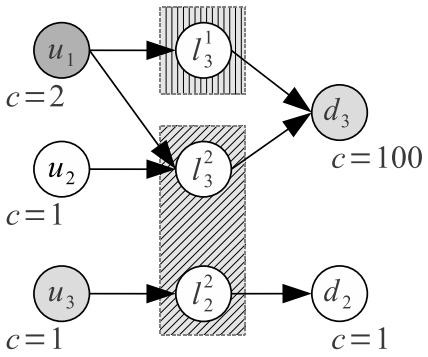
valamint a *kapacitás korlátok*at:

$$\sum_{t,j} f(u_i, l_j^t) \leqslant \mu_i \qquad \forall (u_i, l_j^t) \in E_U,$$

$$\sum_{t} f(l_j^t, d_j) \leqslant \delta_j \qquad \forall (l_j^t, d_j) \in E_D.$$

Az 5.2. ábra példát mutat egy kicsiny, két torrentet és három felhasználót tartalmazó BitTorrent közösség speciális páros gráf reprezentációjára[1]. A

[1]A gráf modell háromféle csúcspontot különböztet meg, de matematikai értelemben páros gráf: az U és D halmaz elemei között nem fut közvetlen él, így nem létezhet páratlan hosszú kör a gráfban [66].



5.2. ábra. *Egy két torrentet tartalmazó BitTorrent hálózat gráfja*

satírozott téglalapok a két torrent fájlt szemléltetik az ábrán, a reprezentációnak nem részei. A t_2 torrenthez két aktív leeching session kapcsolódik. Az első felhasználó megosztó. A második felhasználó a harmadik felhasználótól tölti le a második fájlt. A harmadik felhasználó pedig az első fájlt az első, a második fájlt az első és második felhasználótól tölti le. Vegyük észre, hogy minden leeching csúcs pontosan egy letöltő élhez kapcsolódik.

Érdekesség, hogy a letöltők a csomagokat (a „legritkábbat először" szabály betartásával) véletlen sorrendben kapják, így egymásnak is fel tudnak tölteni. Mi több, a letöltés sikeres lehet anélkül, hogy akár egyetlen megosztó is jelen lenne a swarmban [17].

5.3.2. A max-min méltányos erőforrás-kiosztás problémája

Célunk minden $(l_j^t, d_j) \in E_D$ élre a *max-min méltányos erőforrás-kiosztás* meghatározása, vagyis minden egyes letöltő élre a lehető legnagyobb folyam kiszámítása, figyelembe véve, hogy egy letöltő élen csak akkor növelhető a folyam értéke, ha egy tőle nagyobb folyammal rendelkező letöltő élen csökkentjük azt. Ez tulajdonképpen a Pareto-optimális erőforrás-kiosztás egy rokon feladata [69].

Az előző alszakasz hálózati folyam modelljében gondolkodva a sávszélesség-kiosztás akkor *max-min méltányos*, ha bármely (l_j^t, d_j) letöltő élre igaz, hogy az $f(l_j^t, d_j)$ folyam csak egy másik $(l_{j'}^{t'}, d_{j'})$ letöltő él $f(l_{j'}^{t'}, d_{j'})$ folyam értéke csökkentésével növelhető, amelyre $f(l_{j'}^{t'}, d_{j'}) < f(l_j^t, d_j)$.

A max-min méltányos erőforrás-kiosztás a legnagyobb lehetséges letöltési sebességet adja minden felhasználónak, hogy azok az elérhető legjobb minőséget tapasztalhassák pl. élőkép közvetítése esetén. Megismételjük, hogy egy letöltőhöz több leeching session kapcsolódhat, ami több fájl letöltését jelenti

egy időben. Az általunk használt formalizmus a max-min méltányosságot *minden* letöltő élre biztosítja. Mivel a feladat folytonos és konvex halmazon értelmezett [13], a max-min méltányos kiosztás egyértelműen létezik [8, 69].

5.4. Saját megoldás

5.4.1. A javasolt algoritmus kivonata

Az algoritmusunk az *általános max-min programozási algoritmus* [69] egy adaptált változata. A letöltő élek max-min méltányos súlyait iteratív módon számítja, minden körben a legkisebb meghatározatlan súllyal rendelkező változókat rögzítve.

A most következő leírásban a halmazokat nagy betűk, az optimalizálási probléma döntési változóit kis betűk, a paramétereket pedig (a rögzített változókkal egyetemben) görög betűk jelölik.

Az algoritmus a következő lépésekből épül fel:

1. **Inicializálás.** Legyen $F := \emptyset$, $k := 1$, $E_1 := E_D$ és $\forall (l_j^t, d_j) \in E_D : \ell_j^t := 0$.

Az F halmaz tartalmazza a már rögzített folyam értékek azonosítóit, k a ciklusváltozó. Az utolsó iteráció után minden $(l_j^t, d_j) \in E_D$ letöltő élre ℓ_j^t tartalmazza az optimális folyam értéket.

2. **Alsó korlát számítása a folyam értékekre.** Oldjuk meg a következő lineáris programozási feladatot, melyre a későbbiekben MM_0-al fogunk hivatkozni:

$$\begin{aligned} & \max\ f, \\ \text{f.h.}\quad & f(l_j^t, d_j) \geqslant f \qquad \forall (l_j^t, d_j) \in E_D. \end{aligned}$$

   A minimális folyam érték eltárolása. Legyen $\phi := f$.

Megfigyelhetjük, hogy az MM_0-hoz hasonló feladat szerepel Capotă és szerzőtársai [13] *MaxMin* algoritmusában is. Optimális megoldását csak egyszer kell kiszámítani, hogy a 3. lépésben a folyam értékek számára egy jó alsó korláttal rendelkezzünk. Definíció szerint $\forall (u_i, l_j^t), (l_j^t, d_j) \in E : f(u_i, l_j^t), f(l_j^t, d_j) \in \mathbb{R}^+$ és $f_k \in \mathbb{R}^+$ minden esetben.

3. **LP megoldás.** Oldjuk meg a következő, $mMM_k^{(1)}$-gyel jelölt lineáris programozási feladatot:

$$\begin{aligned} \max\ f_k + \sum_{(l_j^t,d_j)\in E_k} f(l_j^t,d_j) \ + \sum_{(l_j^t,d_j)\in(E_D\setminus E_k)} \ell_j^t, & \\ \text{f.h.}\quad f(l_j^t,d_j) \geqslant f_k & \qquad \forall (l_j^t,d_j)\in E_k, \\ f_k \geqslant \phi. & \end{aligned}$$

Optimum eltárolása. Legyen $\sigma_k := \sum_{(l_j^t,d_j)\in E_k} f(l_j^t,d_j) + \sum_{(l_j^t,d_j)\in(E_D\setminus E_k)} \ell_j^t$ és $\phi_k := f_k$.

Az $mMM_k^{(1)}$ lineáris programozási feladat a maximális folyam és a max-min méltányosság célfüggvényét egyesíti. A hálózat maximális átvitelének mértékét számítja ki a max-min méltányosság teljesítése mellett, ezt az értéket σ_k jelöli. A 2. lemmában megmutatjuk, hogy ilyen mértékű adatátvitel a javasolt algoritmus minden iterációjában biztosítható. A következő lépésben állítjuk majd elő azt a max-min méltányos kiosztást, ami egyúttal a σ_k átvitelt is garantálja. Most kiszámítjuk σ_k-t és ϕ_k-t, továbbá egy jó kezdő (fízibilis) megoldást a következő speciális MINLP-hez.

4. **MINLP megoldás.** Oldjuk meg az $mMM_k^{(2)}$-vel jelölt vegyes-egész-értékű bilineáris programozási feladatot:

$$\begin{aligned} & \max \sum_{(l_j^t,d_j)\in E_k} x_j^t, & \\ \text{f.h.} \sum_{(l_j^t,d_j)\in E_k} f(l_j^t,d_j)\, x_j^t \ + \ \phi_k \cdot \sum_{(l_j^t,d_j)\in E_k} (1-x_j^t) \ + \sum_{(l_j^t,d_j)\in(E_D\setminus E_k)} \ell_j^t \ &= \ \sigma_k, & \\ f(l_j^t,d_j) \geqslant \phi_k & \qquad \forall (l_j^t,d_j)\in E_k, \\ f(l_j^t,d_j) > \phi_k\, x_j^t & \qquad \forall (l_j^t,d_j)\in E_k, \end{aligned}$$

ahol $x_j^t \in \{0,1\}$.

Az $mMM_k^{(2)}$ bilineáris MINLP gondoskodik róla, hogy a ϕ_k folyam érték a lehető legkevesebb letöltő élre legyen rögzítve minden iterációban, ezáltal előállítva a max-min méltányosságot. Valójában ez egy kombinatorikus optimalizálási feladat. A szigorú egyenlőtlenség feltétel a bináris x_j^t változót nullára állítja, ha $f(l_j^t,d_j)$ nem növelhető ϕ_k fölé a későbbiekben. Tehát $f(l_j^t,d_j)$ pontosan akkor rögzíthető ϕ_k értékre, ha $x_j^t = 0$. Az 5.4.2. alszakasz

4. lemmája garantálja, hogy legalább egy fízibilis megoldás létezik erre a feladatra.

A hatékonyabb megoldás érdekében $mMM_k^{(2)}$ átírható a McCormick burkolók [36] segítségével. Ez olyan ekvivalens vegyes-egészértékű lineáris (MILP) feladatot eredményez, amelyben a bilineáris tagok helyett új folytonos változók szerepelnek:

$$p_j^t := f(l_j^t, d_j) \cdot x_j^t,$$

ahol $\forall (l_j^t, d_j) \in E_k \; : \; p_j^t \in \mathbb{R}^+$. Minden új p_j^t változóra négy további feltételt kell kikötnünk:

$$\begin{aligned}
&p_j^t \leqslant f(l_j^t, d_j) && \forall (l_j^t, d_j) \in E_k,\\
&p_j^t \leqslant \delta_j \cdot x_j^t && \forall (l_j^t, d_j) \in E_k,\\
&p_j^t \geqslant f(l_j^t, d_j) - \delta_j \cdot (1 - x_j^t) && \forall (l_j^t, d_j) \in E_k,\\
&p_j^t \geqslant 0 && \forall (l_j^t, d_j) \in E_k.
\end{aligned}$$

Habár a feladat dimenziója növekszik az átírással, az új MILP modell globális optimumának megtalálására egzakt korlátozás és szétválasztás típusú megoldó is használható [9]. Erre a problémára, mint $mMM_k^{(2)}$ McCormick átírására fogok hivatkozni a továbbiakban. (Az átírás hatékonyságát részletesebben a 6. fejezetetben elemzem.)

5. **Fixálás.** A ϕ_k-ra vonatkozó aktív korlátok kikeresése, és a kapcsolódó letöltő éleken a folyam értékek rögzítése. Más szavakkal, gyűjtsük ki a ϕ_k értékű optimális folyammal rendelkező letöltő éleket a Φ_k halmazba, Φ_k elemeit adjuk F-hez, és vonjuk ki E_k-ból. Formálisan

$$\Phi_k := \left\{ (l_j^t, d_j) \in E_k \,|\, x_j^t = 0 \right\},$$

$$\forall (l_j^t, d_j) \in E_k\text{-ra, ahol } x_j^t = 0 \; : \; \ell_j^t := \phi_k,$$

$$F := F \cup \Phi_k, \quad \text{és} \quad E_{k+1} := E_k \setminus \Phi_k.$$

6. **Megállási feltétel.** Ha $F = E_D$, megállunk. Különben $k := k + 1$ és ugorjunk vissza a 3. lépésre (új iteráció).

Ezt az algoritmust a továbbiakban *mMaxMin*-nek nevezzük.

5.4.2. A helyesség igazolása

A következőkben bebizonyítom, hogy az előző alszakaszban bemutatott *mMaxMin* algoritmus a max-min méltányos sávszélesség-allokációt számítja.

Jelölje az $mMM_k^{(1)}$ célfüggvény értékét $\mathcal{F}_k$, és

$$\mathcal{F}_k := \phi_k + \sigma_k.$$

1. LEMMA Az $mMM_{k-1}^{(2)}$ minden fízibilis megoldása leképezhető $mMM_k^{(1)}$ egy fízibilis megoldására $k > 1$ esetén.

Bizonyítás. Egy lehetséges leképezés a következő. Legyen $f_k := \phi_{k-1}$ és

$$\forall (l_j^t, d_j) \in E_k \,:\, f_k^{(1)}(l_j^t, d_j) := f_{k-1}^{(2)}(l_j^t, d_j)$$

az *mMaxMin* fixáló lépése után. Az $f_k^{(y)}(l_j^t, d_j)$ az $f(l_j^t, d_j)$ folyam értékét jelöli $mMM_k^{(y)}$ egy fízibilis megoldásában. □

Az $mMM_{k-1}^{(2)}$ optimális megoldásának ily módon történő leképezését ***mMM*$_k^{(1)}$ kezdeti megoldás**nak nevezzük.

2. LEMMA Az *mMaxMin* bármely k. iterációjában fennáll a

$$\sigma_k = \sigma$$

összefüggés, ahol σ egy konstans, a hálózat maximális átvitelének mértéke abban az esetben, amikor

$$\forall (l_j^t, d_j) \in E_D \,:\, f(l_j^t, d_j) \geqslant \phi.$$

Bizonyítás. Az állítást *teljes indukcióval* bizonyítjuk.

Kezdetben nincsenek rögzített értékek, ezért $\sigma_1 = \sigma$. Ez alapján tegyük fel, hogy $\sigma_{k-1} = \sigma$.

A célfüggény összeg alakú tagja *mMaxMin* 3. lépése után így is írható:

$$\sigma_k = c_1\phi_1 + \cdots + c_{k-1}\phi_{k-1} + c_k\phi_k + R_k,$$

ahol $\phi_1, \ldots, \phi_{k-1}$ a rögzített folyam értékek (f_k optimális értékei $mMM_1^{(1)}$, ..., $mMM_{k-1}^{(1)}$-ben), és ϕ_k a legkisebb nem rögzített folyam érték, miközben c_i a ϕ_i multiplicitása (tehát azt jelöli, hány letöltő élen egyenlő a folyam érték ϕ_i-vel). R_k a fennmaradó folyam (a nem fixált folyam értékek összege, mínusz az aktuális iterációban rögzítendő folyamok). Hasonlóképp

$$\sigma_{k-1} = c_1\phi_1 + \cdots + c_{k-1}\phi_{k-1} + R_{k-1},$$

és

$$\sigma_{k-1} - \sigma_k = R_{k-1} - R_k - c_k\phi_k.$$

Ha $R_{k-1} - R_k$ nagyobb, mint $c_k\phi_k$, az azt jelenti, hogy az LP megoldó néhány nem rögzített folyam értéknek az 1. lemmában definiált kezdeti megoldását anélkül csökkenti, hogy a felszabaduló folyamot $mMM_k^{(1)}$-ben más letöltő élekre kiosztaná. Mivel ez szuboptimális megoldás lenne, a lineáris megoldó nem fejeződne be ilyen eredménnyel. Így $R_{k-1} - R_k \leqslant c_k\phi_k$ igaz kell legyen, ennek megfelelően pedig $\sigma_{k-1} \leqslant \sigma_k$.

A másik oldalról, $\sigma_k \leqslant \sigma$ bármely k. iterációban, hiszen a hálózat feltöltési kapacitásai nem változnak az algoritmus iterációi között. A feltevés szerint $\sigma_{k-1} = \sigma$, így $\sigma_k = \sigma$ fennáll bármely k-ra. □

3. LEMMA $\mathcal{F}_k > \mathcal{F}_{k-1}$ az *mMaxMin* bármely $k > 1$. iterációjában.

Bizonyítás. Az $mMM_k^{(1)}$ kezdeti megoldásában minden $f(l_j^t, d_j) \leqslant \phi_{k-1}$ folyam érték le van rögzítve az *mMaxMin* 5. lépése miatt. Így $\phi_k > \phi_{k-1}$, $\sigma_k = \sigma_{k-1} = \sigma$ pedig fennáll a 2. lemma folytán. □

1. KÖVETKEZMÉNY $|\Phi_k| > 0$ az *mMaxMin* bármely k. iterációjában.

4. LEMMA Az $mMM_k^{(1)}$ optimális megoldása leképezhető az $mMM_k^{(2)}$ egy fízibilis megoldására.

Bizonyítás. Egy lehetséges leképezés a következő. Legyen

$$f_k^{(2)}(l_j^t, d_j) := f_k^{(1)}(l_j^t, d_j),$$

és

$$x_j^t := \begin{cases} 1 & \text{ha } f_k^{(1)}(l_j^t, d_j) > \phi_k \text{ és } (l_j^t, d_j) \in E_k, \\ 0 & \text{ha } f_k^{(1)}(l_j^t, d_j) = \phi_k \text{ és } (l_j^t, d_j) \in E_k. \end{cases}$$ □

Az $mMM_k^{(1)}$ optimális megoldásának ily módon történő leképezését $\boldsymbol{mMM}_k^{(2)}$ **kezdeti megoldásá**nak nevezzük.

5. LEMMA Az $mMM_k^{(2)}$ McCormick átírásának kezdeti megoldása előáll $mMM_k^{(2)}$ kezdeti megoldásából.

Bizonyítás. Lehetséges kezdeti megoldás a következő. Az $mMM_k^{(2)}$ kezdeti megoldását egészítsük ki a p változókra vonatkozó kezdőértékkel:

$$p_j^t := \begin{cases} f_k^{(1)}(l_j^t, d_j) & \text{ha } x_j^t = 1 \text{ és } (l_j^t, d_j) \in E_k, \\ 0 & \text{ha } x_j^t = 0 \text{ és } (l_j^t, d_j) \in E_k. \end{cases}$$ □

6. TÉTEL Az *mMaxMin* véges számú lépés után véget ér, és a max-min méltányos tulajdonságot garantálja minden letöltő élre.

Bizonyítás. Az F halmaz elemszáma minden iterációban növekszik a 3. lemma és az 1. következmény miatt. Mivel E_D véges halmaz, az algoritmus véges számú iteráció után véget ér.

A 1. lemma, valamint a 4–5. lemmák értelmében $mMM_k^{(1)}$-nek és $mMM_k^{(2)}$-nek legalább egy fízibilis megoldása létezik *mMaxMin* bármely k. iterációjában. Az utolsó iteráció után az $\ell := \left\{\ell_j^t \mid (l_j^t, d_j) \in E_D\right\}$ halmaz tartalmazza a rögzített folyam értékeket a letöltő élekre. A folyam értékek korlátai ugyanúgy vonatkoznak ℓ elemeire is, így $\forall (l_j^t, d_j) \in E_D \ : \ 0 \leqslant \ell_j^t \leqslant \delta_j$. Ezért ℓ egy kompakt halmaz. Továbbá a folyam értékekre vonatkozó minden feltétel lineáris, ezért ℓ halmaz konvex is. Radunović és Le Boudec tétele [69] alapján konvex kompakt halmazokra létezik max-min méltányos sávszélesség allokáció. Jelöljük az adott gráf letöltő éleire vonatkozó max-min méltányos sávszélesség-kiosztást ω-val:

$$\omega := \left\{\omega_j^t \mid (l_j^t, d_j) \in E_D \text{ és } \omega \text{ max-min méltányos}\right\}.$$

Indirekt módon bizonyítjuk, hogy *mMaxMin* a max-min méltányosságot minden letöltő élre garantálja.

Tegyük fel, hogy $\ell \neq \omega$. Ekkor létezik olyan legkisebb k index, amelyre

$$\exists j \, \exists t \ : \ \left(\ell_j^t \text{ a } k. \text{ iterációban lett rögzítve, és } \ell_j^t \neq \omega_j^t\right).$$

Ez azt jelenti, hogy $x_j^t = 0$ és $f(l_j^t, d_j) \neq \omega_j^t$ az $mMM_k^{(2)}$ optimális megoldásában. Vegyük észre, hogy $f(l_j^t, d_j) \leqslant \omega_j^t$, különben ω nem lehetne max-min méltányos.

Az $mMM_k^{(2)}$ konstrukciója garantálja, hogy

$$\left(x_j^t = 0 \wedge f(l_j^t, d_j) = \phi_k\right) \ \vee \ \left(x_j^t = 1 \wedge f(l_j^t, d_j) > \phi_k\right)$$

igaz bármely $(l_j^t, d_j) \in E_D$-re, vagyis, ha $f(l_j^t, d_j)$ a ϕ_k-nál nagyobb értékre is beállítható lenne, akkor x_j^t az 1 értéket kapná. Tehát $x_j^t = 0$ -ból következik, hogy $f(l_j^t, d_j) = \phi_k$.

Másrészt, ϕ_k az $mMM_k^{(1)}$ megoldásából származó (nem rögzített) max-min folyam érték a k. iterációban. Ez ellentmond annak a feltevésnek, hogy $\ell_j^t \neq \omega_j^t$. □

5.4.3. *MaxMin-r*, az implementált változat

Az 5.4.2. alszakasz néhány megállapítását explicitté tettük az algoritmus implementált változatában. Ezen felül beillesztettünk egy előmegoldó lépést, ezt azonnal kifejtem bővebben. Az így előálló iteratív algoritmust, amelyet *MaxMin-r*-nek neveztünk, az A.2. algoritmusban foglaltam össze. A megvalósítást az AMPL modellezési nyelvben (ld. a 2.4. szakaszt) készítettük. Összehasonlító teszteket is végeztünk az algoritmus elemzése céljából, ennek részleteit a következő szakaszban mutatom be.

A főbb különbségek az *mMaxMin*-nel összehasonlítva a következők:

1. A 2. lemma alapján bevezettük a 2. lépést a konstans σ meghatározására. Az MM_{MaxFlow} LP feladatot csak az első iterációban oldjuk meg és a módosított $mMM_k^{(1)}$ felhasználja σ-t az első feltételben.

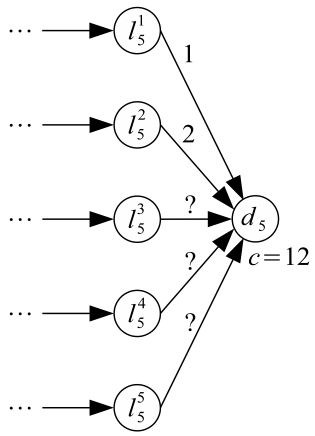
2. Az $mMM_k^{(1)}$-ben a felmerülő numerikus hibákra tekintettel az „$= \sigma$" szigorú alsó korlát helyett a „$\geqslant (1-\epsilon)\cdot\sigma$" alsó korlátot használjuk.

3. A *MaxMin-r* 5. lépése egy új, előmegoldó fázist vezet be. Ehhez egy, az AMPL-ben is megvalósított [26, 30] standard LP előmegoldó technika adta az ötletet. A korai implementációink tesztelési fázisában felfigyeltünk rá, hogy az AMPL előmegoldó mechanizmusa a MILP modell változóinak a számát sok esetben jelentősen csökkenteni tudta. Alaposabban szemügyre véve, az

$$E_{k_f} := \left\{ (l_j^t, d_j) \in E_k \;\middle|\; \frac{c(d_j) - \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t}{deg_k^-(d_j)} = \phi_k \right\}$$

halmaz minden iterációban olyan letöltő éleket tartalmaz, amelyekre az $f(l_j^t, d_j)$ folyam értéket a *MaxMin-r* ugyanezen iterációjának a 7. lépésében rögzíteni lehet. A $deg_k^-(d_j)$ itt a d_j csúcs nem fixált bejövő éleinek a számát jelöli. Ezt az egyszerű szabályt explicitté téve a futási idő minden esetben nagy mértékben csökkent (részletesebb elemzésért ld. a 6. fejezetet).

Ha az előmegoldó fázisban sikerül valamelyik folyam értéket rögzíteni, a MILP megoldást kihagyjuk. Ez a döntés csupán kísérleti megfontoláson alapszik: a tesztjeinkben az iterációk túlnyomó többségében minden szükséges fixálás megtörtént az előmegoldó fázis alapján. Bizonyos esetekben előfordul, hogy néhány letöltő él nem kerül be az E_{k_f} halmazba, habár optimális folyam értéke ϕ_k lenne. Ebben az esetben a ϕ_k értéke nem javul a következő iteráció 4. lépésében, tehát $\phi_{k+1} = \phi_k$.



5.3. ábra. *Példa dimenzió-csökkentés lehetőségére a MILP megoldása nélkül*

Így az $E_{(k+1)_f}$ üres lesz, ami miatt a MILP-et meg kell oldani a letöltő éleken rögzítendő folyamok meghatározásához.

Ezzel a módosítással legrosszabb esetben $2 \cdot |E_D|$ iterációt igényel az algoritmus. Amint az 5.5. szakaszban látni fogjuk, a gyakorlatban általában sokkal kevesebb iteráció elegendő.

Az 5.3. ábra illusztrálja a MILP megoldása nélkül megvalósítható dimenzió-csökkentést. A példában az ötödik felhasználó öt torrentet tölt le egy időben, és az első két torrentre a max-min méltányosan elérhető legnagyobb folyamot *MaxMin-r* korábbi iterációiban 1 és 2 értékekre rögzítettük. Tegyük fel, hogy $\phi_k = 3$. Ekkor $f(l_5^t, d_5) \geqslant 3$ a $t = 3, 4, 5$ torrentekre. Ebben az iterációban d_5 szabad letöltési kapacitása $12 - (1 + 2) = 9$, vagyis a 3 egyúttal felső korlát is a nem rögzített folyam értékekre. Ezen megfontolás alapján az $f(l_5^t, d_5)$-re a $t = 3, 4, 5$ esetekben a MILP megoldása nélkül egyértelműen beállítható és rögzíthető a 3 érték.

4. A 6. lépésben *MaxMin-r* az $mMM_k^{(2)}$ helyett annak McCormick átírását használja.

5.5. Teszteredmények

5.5.1. A tesztkörnyezet leírása

A numerikus tesztekhez Andrade és szerzőtársai [2] BitTorrent méréseinek feldolgozott eredményeit használtuk. Ugyanezt az adathalmazt használta a [13] publikáció, melyben az *MM* algoritmust közölték és empirikusan tesztelték. A feldolgozott adathalmaz a *BitSoup.org* nevű BitTorrent közösség pillanatfelvételeit tartalmazza az 5.3. szakaszban bemutatott gráf reprezentációban. A gráfok az AMPL adatformátumában lettek eltárolva.

Közlésre elfogadott cikkünkben [5] véletlenszerűen választottunk egy G gráfot, melynek négy részgráfját állítottuk elő (G_{500}, G_{1000}, G_{1500} és G_{2000}). Ezek a részgráfok rendre a G gráf 500, 1000, 1500 és 2000 véletlenszerűen választott torrentjét tartalmazzák a hozzájuk kapcsolódó U, L, D csúcsokkal és élekkel együtt. A részgráfok karakterisztikája az 5.1. táblázatban megtalálható. Habár G_{1500} kevesebb élt tartalmaz, mint G_{1000}, sokkal több csúccsal rendelkezik, továbbá élei közül több reprezentál leeching session-t.

5.1. táblázat. *A numerikus tesztekben felhasznált gráfok karakterisztikája*

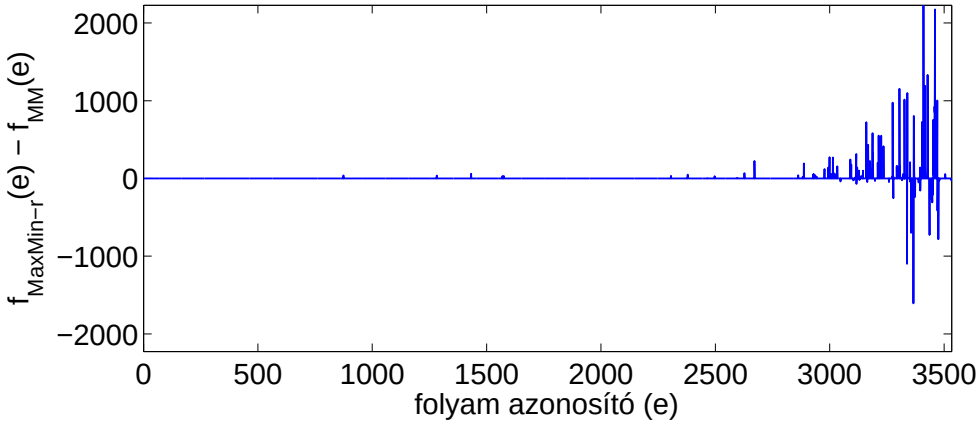
Gráf	$\lvert U \cup D \cup L\rvert$	$\lvert E\rvert$	$\lvert E_D\rvert$
G_{500}	6 984	43 410	1 411
G_{1000}	14 702	272 231	2 721
G_{1500}	18 333	269 165	3 536
G_{2000}	23 670	524 054	7 326

Az *MM* és *MaxMin-r* algoritmusok AMPL implementációit hasonlítottuk össze. Az LP feladatok eredményeit a MOSEK megoldó 7.0.0.106 verziójával, a MILP-ek megoldásait pedig a Gurobi 5.6.3 verziójával állítottuk elő. A tesztek egy 24 magos Intel Xeon 2,27 GHz-es számítógépen futottak, ahol 24 GB memória állt rendelkezésre.

5.5.2. Az előállított erőforrás-kiosztás minőségéről

Az 5.4. ábra $f_{\text{MaxMin-r}}(e) - f_{\text{MM}}(e)$-t mutatja, ami a *MaxMin-r* és az *MM* optimális folyam értékei közötti különbség az $e \in E_D$ letöltő élekre az 1000 torrentes esetben. A kapcsolódó adatsorok a többi példára is hasonló eredményt mutatnak. A folyam értékeket növekvő sorrendbe rendeztük az *MM* optimális megoldása alapján. Így tehát a pozitív számok az ábra bal oldalán csakúgy, mint a negatívak a jobb oldalon azt jelentik, hogy *MaxMin-r* az

MM-hez viszonyítva jobb folyam értékeket biztosít néhány „gyenge" letöltő számára egy pár „erősebb" letöltő kárára. Másként fogalmazva, az új algoritmus megegyező pontosság és tolerancia paraméterek mellett „méltányosabb" allokációt ereményez, mint *MM*. Hogyan lehetséges ez?



5.4. ábra. *A numerikus közelítés hibája a* G_{1000} *példán*

Sajnálatos módon a számítógépes implementáció a folytonos optimalizálási feladatot diszkrét problémává alakítja, köszönhetően a valós változók lebegő-pontos tárolásának. A numerikus megoldó tolerancia beállításai alapvetően határozzák meg az előállított erőforrás-kiosztás minőségét. Ennek oka, hogy a korábban hivatkozott elméleti eredmények (az optimális max-min méltányos allokáció egyedi a folytonos keresési térben [8]) nem alkalmazhatóak a numerikus tesztekben. A probléma megoldható lenne szimbolikus reprezentáció alkalmazásával, azonban valós méretű példákra még a numerikus módszerek is igen lassúak. Másrészről ugyanarra a problémára az *MM* és *MaxMin-r* által visszaadott folyam értékek kummulált eloszlása identikus, és a letöltő élek több, mint 85%-a megegyező erőforrásokat kap a két algoritmustól. Mindezek alapján a két algoritmus által szolgáltatott megoldást egyformán jónak tekintjük a továbbiakban.

5.5.3. A megoldás iterációnkénti alakulása, futási idők

Az 5.5–5.8. ábrák két szempontból összegzik az *MM* és *MaxMin-r* viselkedését a fent bemutatott 500, 1000, 1500 és 2000 torrentes problémákon. Az első oszlop a teljes abszolút eltérést mutatja az optimális megoldástól az egyes iterációkban:

$$\mathrm{abs}(k) = \sum_{(l_j^t, d_j) \in E_D} \left| f_k(l_j^t, d_j) - f_{opt}(l_j^t, d_j) \right|,$$

ahol $f_k(l_j^t, d_j)$ az (l_j^t, d_j) letöltő élre kiosztott folyam értéket jelöli a k. iterációban, az $f_{opt}(l_j^t, d_j)$ pedig az optimális folyam érték ugyanezen az élen, vagyis a kapcsolódó algoritmus utolsó iterációjának eredménye.

A második oszlop azoknak a letöltő éleknek az arányát mutatja, amelyekre az allokált folyamnak az optimálistól való relatív eltérése kevesebb, mint öt százalék:

$$\mathrm{rel}(k) = \frac{\sum_{(l_j^t, d_j) \in E_D} r\left(k, (l_j^t, d_j)\right)}{|E_D|},$$

ahol

$$r\left(k, (l_j^t, d_j)\right) := \begin{cases} 1 & \text{ha } \frac{|f_k(l_j^t, d_j) - f_{opt}(l_j^t, d_j)|}{f_{opt}(l_j^t, d_j)} > 0{,}05, \\ 0 & \text{különben.} \end{cases}$$

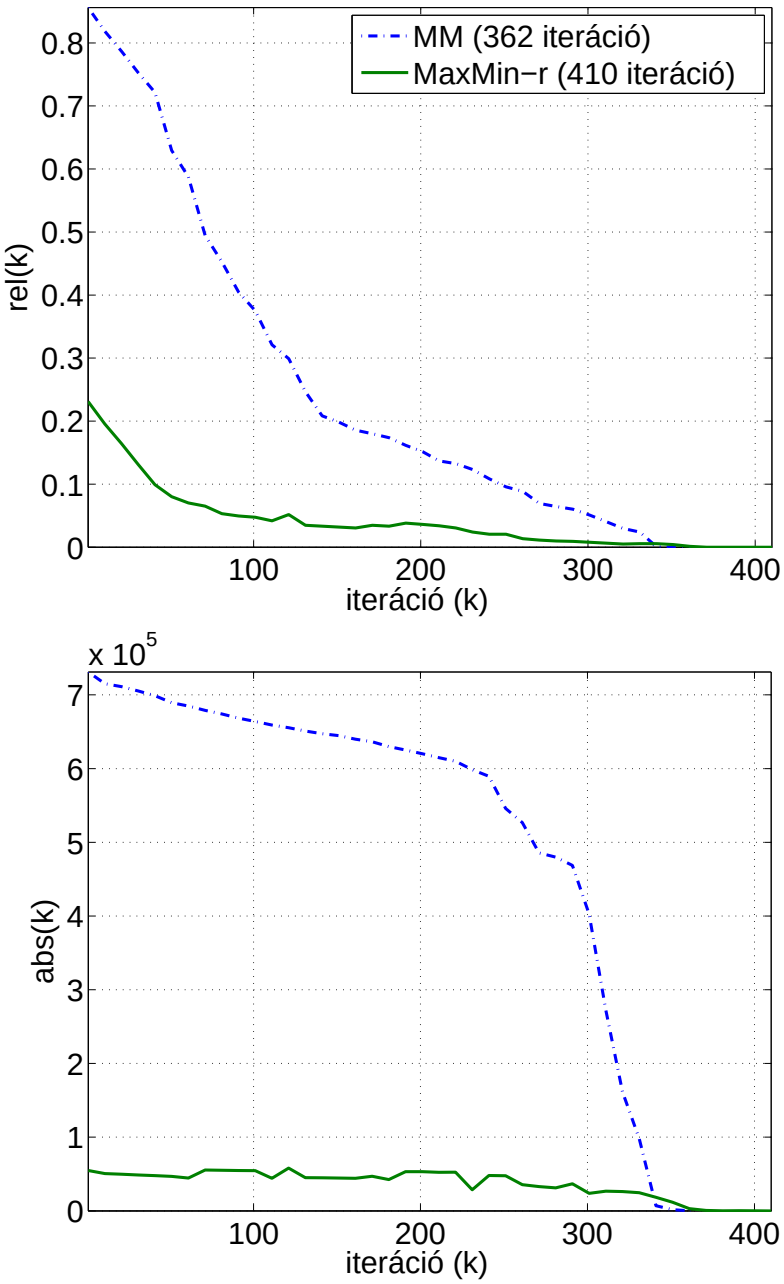
Az ugyanezen tesztek során mért futási időket az 5.9–5.10. ábrák mutatják.

Úgy tűnik, hogy *MaxMin-r* egzakt megfogalmazása nagyon jó minőségű megoldásokat eredményez már az első iterációtól kezdve. A teljes abszolút eltérést összehasonlítva, az új algoritmus kimenete már az első iteráció után olyan minőségű, mint *MM* kimenete az iterációk 85%-ánál.

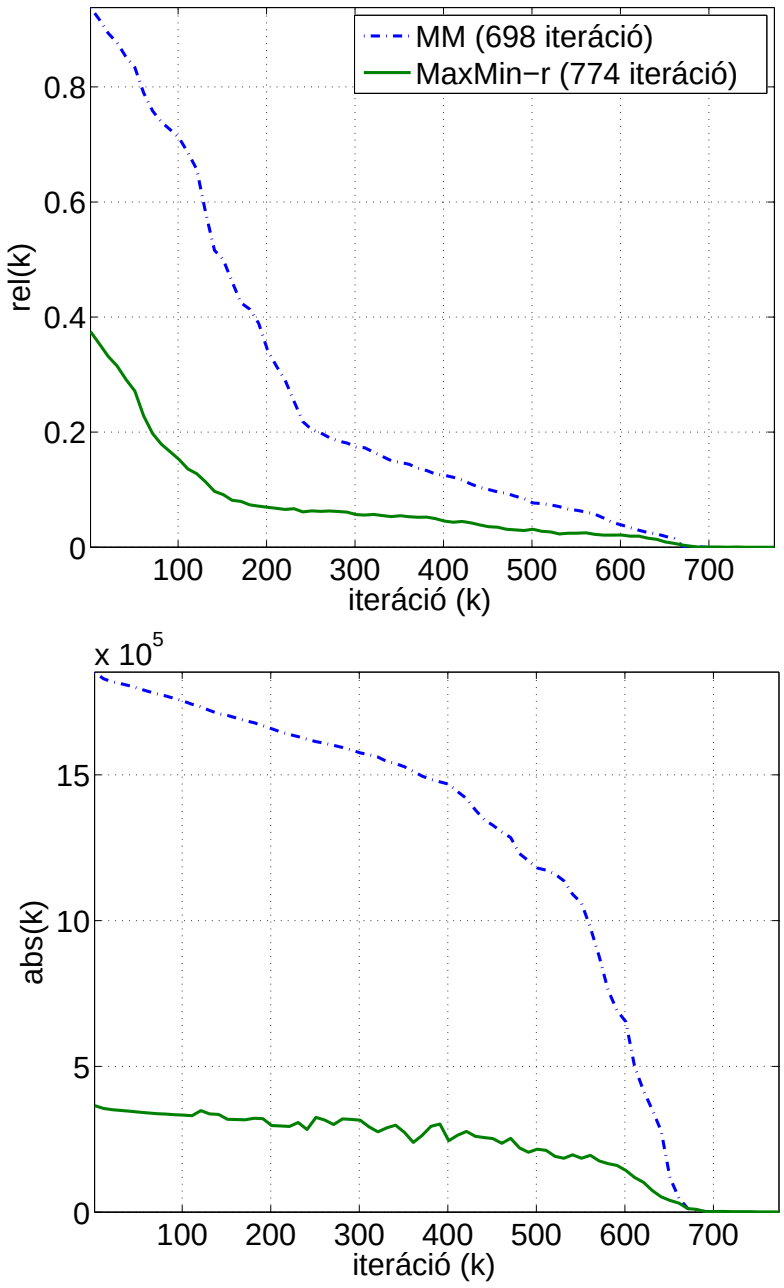
Az 5.5–5.8. ábrák második oszlopa azt mutatja, hogy *MaxMin-r* sokkal több élre állít be az optimálishoz közeli értéket az egyes iterációkban, mint *MM*. Például G_{2000} letöltő éleinek a 46%-a majdnem optimális folyam értéket kap az első iteráció után, szemben azzal a 2,4%-kal, amit az *MM* állít az optimálishoz közelire.

Ráadásul a *MaxMin-r* első iterációja csak 46 másodpercet igényelt a legnagyobb, G_{2000} példán is, míg az *MM* algoritmus hasonlóan jó eredményt biztosító első 910 iterációja több, mint nyolc órán keresztül futott.

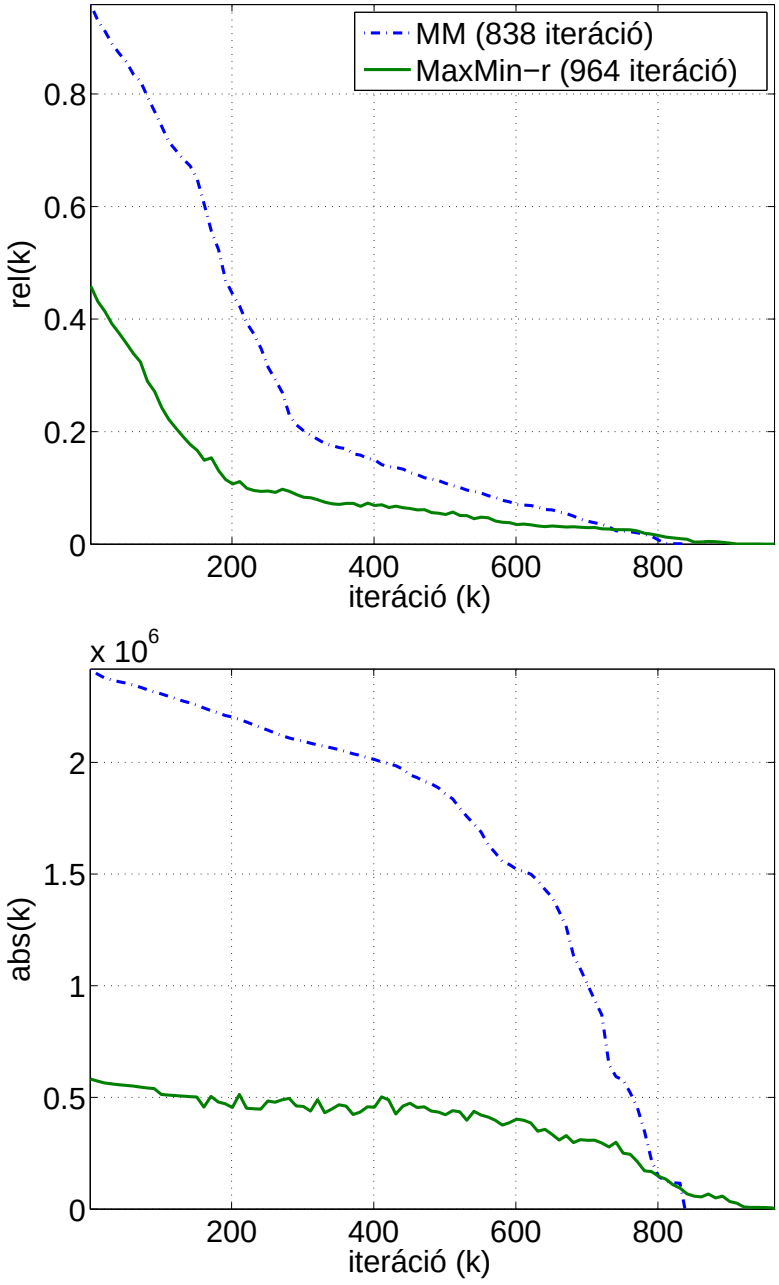
Az 5.9–5.10. ábrák azt mutatják, hogy *MaxMin-r* a nagyobb példákra rövidebb futási időt igényel, mint *MM*, habár még ezzel a technikával is lehetetlen lenne *valós idejű számításokat* végezni egy teljes BitTorrent közösség gráf reprezentációjára. Ezért nagy feladatokra javasoljuk *MaxMin-r* megállítását az első néhány iteráció után, a max-min méltányos allokáció egy jó fízibilis közelítésének belátható időn belüli előállítása céljából.



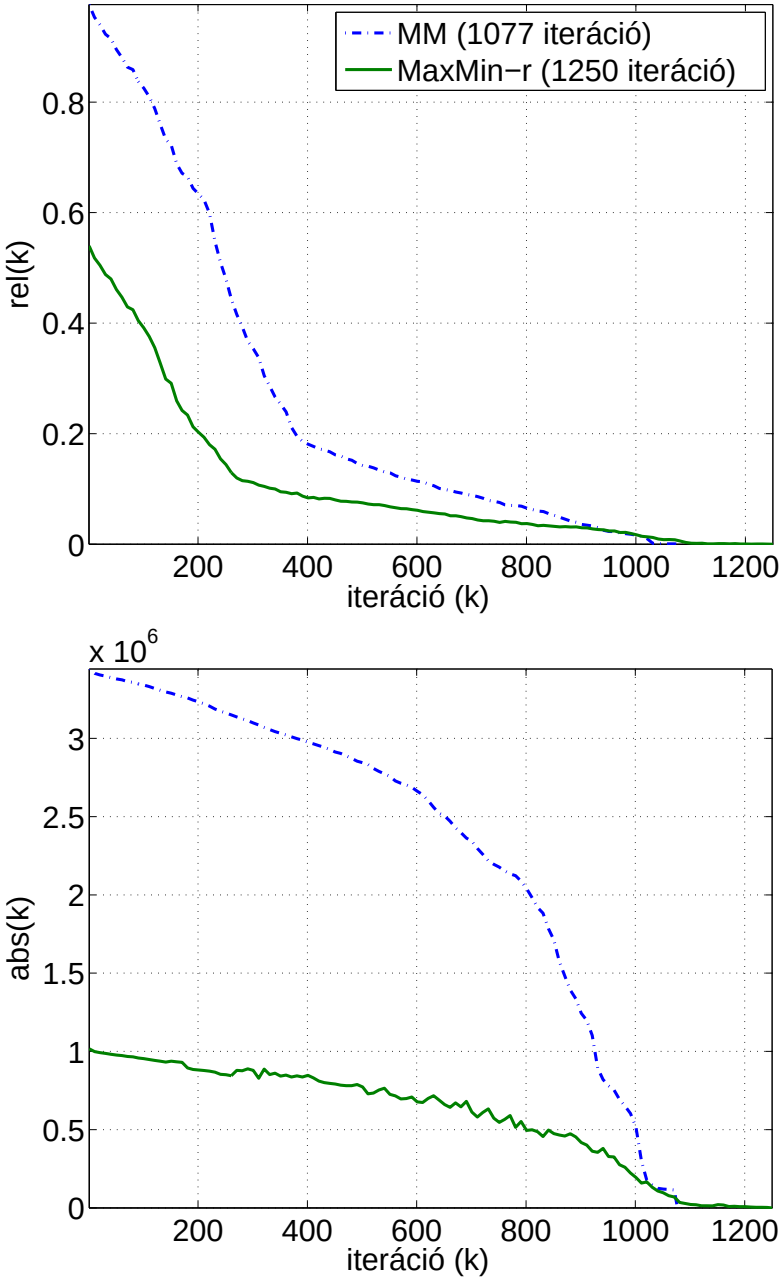
5.5. ábra. *Az* MM *(szaggatott vonallal) és* MaxMin-r *(folytonos vonallal) megoldásának minősége a* G_{500} *tesztesetre*



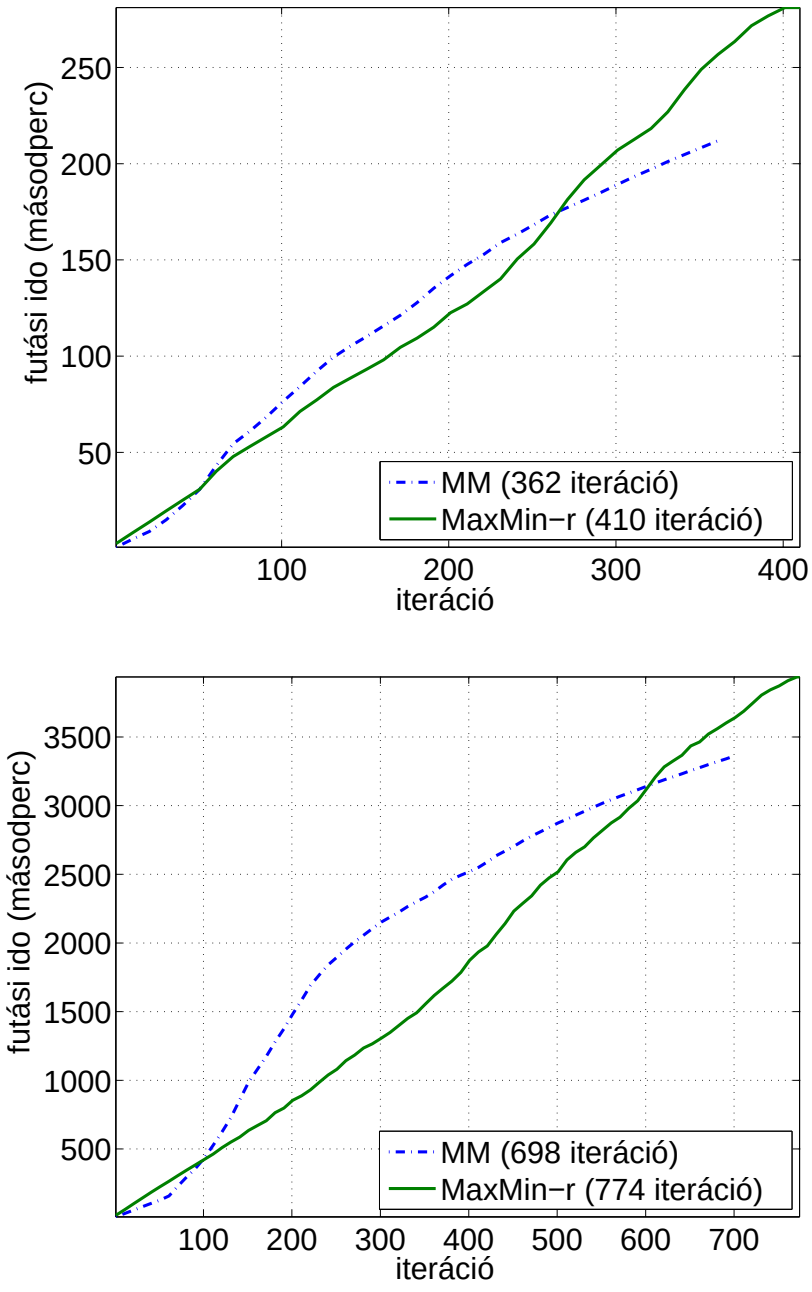
5.6. ábra. *Az* MM *(szaggatott vonallal) és* MaxMin-r *(folytonos vonallal) megoldásának minősége a* G_{1000} *tesztesetre*



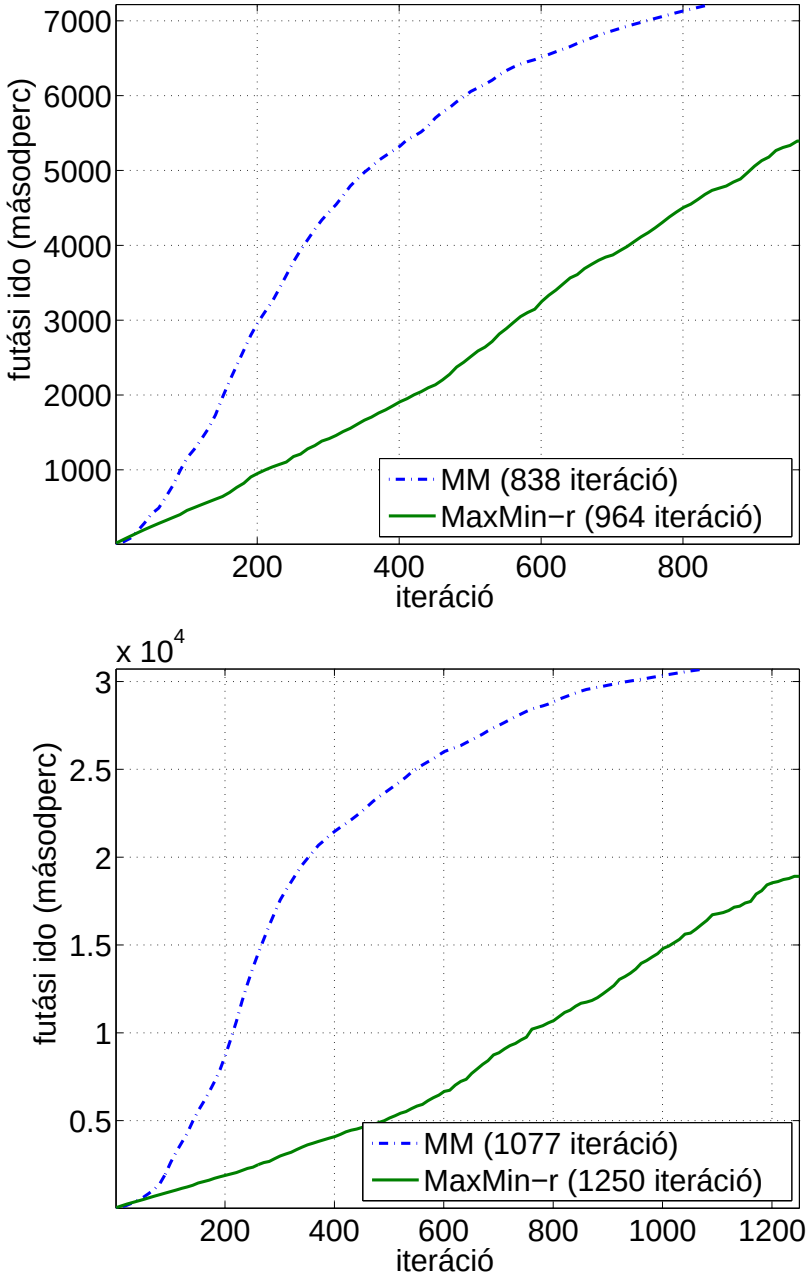
5.7. ábra. *Az* MM *(szaggatott vonallal) és* MaxMin-r *(folytonos vonallal) megoldásának minősége a* G_{1500} *tesztesetre*



5.8. ábra. *Az* MM *(szaggatott vonallal) és* MaxMin-r *(folytonos vonallal) megoldásának minősége a* G_{2000} *tesztesetre*



5.9. ábra. *Az* MM *(szaggatott vonallal) és* MaxMin-r *(folytonos vonallal) futási ideje a* G_{500} *(fent) és* G_{1000} *(lent) tesztesetekre*



5.10. ábra. *Az* MM *(szaggatott vonallal) és* MaxMin-r *(folytonos vonallal) futási ideje a* G_{1500} *(fent) és* G_{2000} *(lent) tesztesetekre*

6

MAX-MIN MÉLTÁNYOSSÁGI MODELLEK KOMPARATÍV ELEMZÉSE

Egy optimalizálási feladat megoldásának sebességét sokféle tényező befolyásolhatja, többek között az adott feladat mérete (beleértve a változók és a korlátok számát is), típusa (lineáris, egészértékű, stb.), a megoldó algoritmus, valamint a reprezentáció módja (beleértve az alkalmazott adatstruktúrákat és a matematikai modellt is). Ezen fejezet alapjául szolgáló tanulmányunkat az 5. fejezetben bemutatott hálózati folyam probléma matematikai modelljének felírása kapcsán felmerülő kérdések ihlették.

Közlésre benyújtott cikkünkben [22] azt vizsgáltuk, hogy a szóban forgó, nagyméretű lineáris és nemlineáris vegyes-egészértékű programokat is magában foglaló optimalizálási feladat megoldásának sebességét mennyiben befolyásolja különféle modellezési technikák alkalmazása. Az elosztott tartalom-megosztó hálózat max-min méltányos erőforrás-kiosztását előállító modell tizenkét változatát hasonlítottam össze egy kiterjedt numerikus tesztelés során, két professzionális megoldó és huszonhét nagyméretű tesztfeladat felhasználásával.

Reményeim szerint az ebben a fejezetben is bemutatott eredmények túlmutatnak a konkrét problémán, és árnyaltabb képet adnak más hasonló feladatok megértéséhez is.

6.1. Bevezetés

Míg a hagyományos tartalomletöltésnél a felhasználók igénye elsősorban a letöltés sebességére vonatkozik (minél gyorsabb, annál jobb), addig a letöltés közbeni megtekintés vagy meghallgatás jellegű szolgáltatásoknál minden felhasználóra a számára elérhető lehető legjobb minimális letöltési sebességet kell garantálnunk. A feladat, bizonyos feltételek kikötése mellett, megfogalmazható egy speciális szerkezetű gráfon értelmezett nemlineáris vegyes-egészértékű optimalizálási feladatként. Ennek részletes leírása az

előző fejezetben, vagy az [5] cikkben megtalálható. Mivel a vizsgált feladat megoldására javasolt iterációs módszer számos érdekes modellezési meggondolást tartalmaz, természetesen adódik a kérdés: vajon milyen tényezők befolyásolják a megoldás sebességét?

Ebben a fejezetben összegyűjtöttem a lehetséges opciókat, melyeket részletes numerikus vizsgálatoknak vetettem alá. Bár a feladat specifikus, meggyőződésem, hogy az elvégzett numerikus tesztek által kapott eredmények általánosabb érvényű empirikus képet adnak a hasonló típusú problémák számítógépes megoldási lehetőségeire.

Először röviden ismertetem az [5] cikkben javasolt iterációs módszert, és a lehetséges algoritmus változatok egy bőséges listáját. A 6.3. szakasz tartalmazza a felhasznált tesztesetek leírását, amit az utolsó, 6.4. szakaszban a numerikus eredmények diszkussziója követ.

6.2. Modell-átírási lehetőségek

A max-min méltányos erőforrás-kiosztás kiszámítását célzó algoritmus kiindulási alakját az A.3. algoritmus tartalmazza. A korábbi cikkünkben [5] bemutatott megoldás tulajdonképpen Radunović és Le Boudec általános max-min programozási algoritmusának [69] a feladatra adaptált változata. A letöltési élek max-min méltányos folyamait iteratív módon számítjuk: minden iterációban a legkisebb, még nem rögzített folyammal rendelkező letöltő élekre állapítjuk meg a minden korlátot kielégítő lehető legnagyobb folyam értéket. A halmazokat nagy betűkkel, az optimalizálási feladatok döntési változóit kis betűkkel, míg a paramétereket (a rögzített értékkel bíró változókkal egyetemben) görög betűkkel jelöljük.

Az A.3. algoritmusnak természetesen sokféle variációja képzelhető el, a megvalósítás során érdemes lehet különféle modellezési „trükköket” alkalmazni. Korábbi cikkünk munkálatai közben, melynek eredményeit az 5. fejezet mutatja be, többféle változtatást is eszközöltünk a hatékonyság növelése érdekében, azonban az egyes változtatások hasznosságának igazolására akkor nem kerülhetett sor. A következőkben számbaveszem az ott javasolt módosításokat, a 6.4. szakaszban pedig elemzem az ezen módosítások kombinációiból előálló modell-változatok hatékonyságát a futási idő és az elért optimum érték tekintetében.

6.2.1. Egy redundáns korlát hozzáadása

Az A.3. algoritmus 4. lépésében szereplő $mMM_k^{(1)}$ jelzésű LP feladathoz hozzáadhatjuk a következő korlátot:

$$f_k \geqslant \phi. \tag{6.1}$$

A 3. lemma alapján a (6.1) korlát redundáns, és csak az első iterációban aktív, mivel az (A.1) jelzésű célfüggvény és a 7. fixáló lépés együttesen kikényszeríti, hogy $f_k > f_{k-1}$ minden k. iterációra. Ugyanakkor benyomásunk szerint az LP megoldónak jelentős segítség, ha ez a fix alsó korlát is szerepel a feladatban.

6.2.2. Bilineáris vegyes-egészértékű feladat McCormick-átírása

Amint az 5.4. szakaszban is láttuk, az A.3. algoritmus 6. lépésében szereplő $mMM_k^{(2)}$ jelzésű bilineáris programozási feladat helyettesíthető a McCormick-átírásával, ami a következő:

$$\max \sum_{(l_j^t, d_j) \in E_k} x_j^t,$$

$$\text{f.h.} \sum_{(l_j^t, d_j) \in E_k} p_j^t + \phi_k \sum_{(l_j^t, d_j) \in E_k} (1 - x_j^t) + \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t \geqslant (1 - \epsilon) \cdot \sigma, \tag{6.2}$$

$$f(l_j^t, d_j) \geqslant \phi_k \qquad \forall (l_j^t, d_j) \in E_k,$$

$$f(l_j^t, d_j) > \phi_k \, x_j^t \qquad \forall (l_j^t, d_j) \in E_k,$$

$$\min\left(\delta_j \, x_j^t, \, f(l_j^t, d_j)\right) \geqslant p_j^t \qquad \forall (l_j^t, d_j) \in E_k, \tag{6.3}$$

$$\max\left(0, \, f(l_j^t, d_j) - \delta_j \, (1 - x_j^t)\right) \leqslant p_j^t \qquad \forall (l_j^t, d_j) \in E_k. \tag{6.4}$$

Vagyis az eredeti (A.2) jelzésű feltételt lecseréljük (6.2)-re, és kiegészítjük a feladatot (6.3) és (6.4) feltételekkel.

Habár a probléma dimenziója nő az átírás folytán, más jellegű, egzakt korlátozás-és-szétválasztás típusú megoldó válik alkalmazhatóvá az előálló MILP globális optimumának megtalálására.

A következőkben az algoritmus 6. lépésében szereplő MINLP, ill. MILP feladatokra gyűjtőnéven MIP-ként fogok hivatkozni.

6.2.3. Kezdőérték-adás a bináris változókra

Az $mMM_k^{(1)}$ optimális megoldása leképezhető $mMM_k^{(2)}$ egy fízibilis megoldására (ld. a 4. lemmát). Legyen

$$f_k^{(2)}(l_j^t, d_j) := f_k^{(1)}(l_j^t, d_j),$$

és

$$x_j^t := \begin{cases} 1 & \text{ha } f_k^{(1)}(l_j^t, d_j) > \phi_k \text{ és } (l_j^t, d_j) \in E_k, \\ 0 & \text{ha } f_k^{(1)}(l_j^t, d_j) = \phi_k \text{ és } (l_j^t, d_j) \in E_k. \end{cases}$$

Az $mMM_k^{(2)}$ feladat megoldása során segítséget jelenthet a bináris változókra vonatkozó kezdőértékek explicit megadása.

6.2.4. Kezdőérték-adás a McCormick-átírás mesterséges változóira

Az 5. lemma alapján $mMM_k^{(2)}$ McCormick-átírásának kezdőértéke csakugyan előállítható $mMM_k^{(2)}$ kezdőértékéből. Ehhez $mMM_k^{(2)}$ kezdőértékét bővítjük a p változókra vonatkozó értékekkel:

$$p_j^t := \begin{cases} f_k^{(1)}(l_j^t, d_j) & \text{ha } x_j^t = 1 \text{ és } (l_j^t, d_j) \in E_k, \\ 0 & \text{ha } x_j^t = 0 \text{ és } (l_j^t, d_j) \in E_k. \end{cases}$$

6.2.5. Előmegoldás: folyamok rögzítése a folyam-megmaradásra tekintettel

Az A.3. algoritmus 5. lépése egy, az AMPL előmegoldójában is megvalósított standard LP előmegoldó technika (ld. az 5.4.3. alszakaszt). Az

$$E_{k_f} := \left\{ (l_j^t, d_j) \in E_k \;\middle|\; \frac{c(d_j) - \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t}{deg_k^-(d_j)} = \phi_k \right\}$$

halmaz azokat a letöltő éleket tartalmazza, amelyekre a kapcsolódó $f(l_j^t, d_j)$ folyam értékek egyértelműen kiszámíthatóak a hálózat folyam-megmaradási tulajdonsága alapján.

Pontosabban, E_{k_f} minden elemére rögzíthető a ϕ_k folyam érték az algoritmus k. iterációjának 7. lépésében, méghozzá a 6. lépésben szereplő MIP megoldásától függetlenül. Ennek megfelelően az előmegoldást tartalmazó modellekben kimarad a MIP megoldása, amennyiben $E_{k_f} \neq \emptyset$ valamely k.

6.1. táblázat. *Vizsgált modellek áttekintése*

	6.2.1	6.2.2	6.2.3	6.2.4	6.2.5
ref	•	•	•	•	•
1	•	•	•	•	
2	•	•	•		
3	•	•			
4	•				
5					
6			•		
7	•		•		
8	•		•		•
9			•		•
10	•				•
11					•

iterációban. Ha lenne olyan letöltő él, amelyre ϕ_k értékű folyamot kellene rögzíteni, de az előmegoldó ezt nem tudja megállapítani, úgy a következő iterációban ϕ_{k+1} értéke meg fog egyezni ϕ_k-val és a MIP megoldásra kerül. Legrosszabb esetben az iterációk számának duplázásával is járhat ez a megoldás, azonban az általunk tesztelt esetekben az iterációk jelentős részében minden szükséges fixálás megtörtént az előmegoldási fázisban, és csak az esetek töredékében volt szükség a MIP megoldására.

6.2.6. Modell-változatok

A fent bemutatott módosítási javaslatok kombinációiból összesen tizenkettő modell-változatot készítettem annak érdekében, hogy a módosítások hasznosságát külön-külön, ill. együttesen elemezni lehessen (eredmények a 6.4. szakaszban). A 6.1. táblázat összefoglalja, hogy melyik modell melyik korábbi alszakasz(ok)ban bemutatott módosítást tartalmazza. Referenciaként (ref) a korábban publikált [5], és az 5.4.3. alszakaszban összefoglalt matematikai modell szolgált, amely minden módosítást tartalmazott, a többi modellt számokkal jelöltem.

6.3. Tesztesetek

A különböző algoritmus variánsok numerikus hatékonysági vizsgálataihoz számos mesterségesen generált hálózat készült. Ebben a szakaszban

ezen teszthálózatok előállításának módszereiről lesz szó. Alapvetően három, lényegileg különböző típusú hálózatot gyártottunk:

túlkereslet: ebben az esetben a közösségben elérhető fájlok egy részét sokkal többen akarják letölteni, mint amennyien a teljes tartalommal rendelkeznek. A BitTorrent fogalmai szerint tehát ilyenkor nagyon sok letöltő és ehhez képes nagyon kevés megosztó van jelen. A hálózatokat úgy gyártottuk, hogy a fájlok véletlenszerűen választott 10%-ára legyen túlkereslet. Konkrétan a felhasználók fele letöltőként van jelen a kiválasztott tíz százalékban. Megjegyezzük, hogy ezek a felhasználók emellett letöltőként vagy feltöltőként részt vehetnek más fájlokban is. Az ilyen állapot általában akkor fordul elő valós BitTorrent közösségekben, amikor megjelennek rendkívül népszerű tartalmak, amire hirtelen nagyon sokan kíváncsiak [41].

egyenletes: itt minden fájlra teljesül egyfajta egyensúly, nagyjából ugyanannyi a letöltő, mint a feltöltő.

túlkínálat: itt pedig a közösségben elérhető fájlokat sokkal többen kínálják letöltésre, mint amennyien azt ténylegesen le is töltik éppen. Tehát több megosztó van, mint letöltő. Érdekes módon valós BitTorrent közösségekben ez az állapot az, amely a legtöbbször előfordul. Ennek részben az a magyarázata, hogy az ilyen közösségekben a rögzített szabályok egyike általában előír bizonyos időtartamig történő rendelkezésre állást (seedelést), vagy pedig azt, hogy a letöltött mennyiség valamely részét fel kell tölteni a rendszerbe. Ez utóbbi hosszas időt vehet igénybe abban az esetben, ha a fájlra már csak csekély az érdeklődés.

Típusonként 9 teszthálózatot gyártottunk, amelyekben a felhasználók száma 100, 300 és 500 volt, míg a torrentek száma rendre 50, 100 és 200. Ami a sávszélességeket illeti (ami a folyam hálózatban az élek kapacitását jelenti), mindegyik hálózatban véletlenszerűen választottunk értékeket, egyenletes eloszlással, mégpedig a feltöltő élekre a $[128, 2048]$ intervallumból, míg a letöltő élekre az $[512, 4096]$ intervallumból.

A gráfok méretét a 6.2. táblázatba foglaltuk össze. Vegyük észre, hogy a pontok száma minden gráfban jelentősen több, mint a felhasználók és torrentek száma. Ennek a magyarázata, hogy bemenetként nem páros gráfot kell megadnunk, hanem az 5.3. szakaszban bemutatott hármas gráfot, amelyben elsősorban a köztes (L halmazba tartozó) csúcsok száma lesz magas.

6.2. táblázat. *A teszteléshez használt gráfok csúcsainak (n) és éleinek száma (m)*

Gráf	Egyenletes		Túlkereslet		Túlkínálat	
Felh.-Torrent	n	m	n	m	n	m
100-50	653	10 447	307	3 850	827	24 229
100-100	1 172	21 734	513	8 111	1 561	48 872
100-200	2 135	40 206	919	16 201	2 992	99 686
300-50	3 279	255 412	963	37 082	2 571	218 375
300-100	5 837	523 330	1 603	75 338	4 664	433 011
300-200	10 748	1 015 084	2 710	142 701	14 782	2 407 987
500-50	1 913	85 316	1 597	101 097	4 252	598 766
500-100	3 395	175 709	2 626	204 077	7 827	1 203 877
500-200	6 376	357 693	4 690	411 916	14 818	2 417 266

A valós BitTorrent közösségekből származtatható gráfok az itt vizsgáltaknál jóval nagyobb méretűek, ugyanakkor nem feltétlenül reprezentálnak olyan eseteket, amelyeket itt vizsgálunk. A kisebb méret továbbá lehetővé teszi, hogy minden vizsgált modellre kivárható időn belül lefussanak a tesztjeink.

6.4. Eredmények

6.4.1. A tesztkörnyezet leírása

A hálózat összetételének, valamint a megoldás során alkalmazott matematikai modellnek a megoldó (solver) program hatékonyságára gyakorolt hatását szerettük volna megállapítani, ezért kiterjedt numerikus tesztelést végeztem. A 6.2. szakaszban bemutatott *tizenkettő modell-változat*, a 6.3. szakasz *huszonhét tesztesete* és *kettő professzionális megoldó* minden lehetséges kombinációjára *három futtatás*t végeztem.

Az algoritmus-variánsok AMPL nyelven voltak kódolva. A Gurobi és a MOSEK solvert vettem górcső alá, mivel ez a két általános nemlineáris megoldó állt rendelkezésemre, ami a szóban forgó modellek optimalizálási feladatait kivárható időn belül meg tudta oldani. Mindkét megoldót az alapértelmezett paraméterekkel működtettem. A tesztek az 5.5. szakasz tesztjei során is alkalmazott 24 magos Intel Xeon 2,27 GHz-es számítógépen futottak, ahol 24 GB memória állt rendelkezésre.

6.4.2. Gurobi

A B.1–B.3. táblázatok a Gurobi megoldóval elért átlagos futási időket mutatják a különböző feladat-osztályokra. Érdemes megemlíteni, hogy bizonyos modell–teszteset–megoldó kombinációkra, valószínűleg a feladat bonyolultságából adódó nagy tárigény miatt, mindhárom futtatáskor szegmentálási hibával állt le az AMPL. Ezeket az eseteket „n.a." jelöli a táblázatokban.

A megértés segítésére minden további táblázatra vetítettem egy, az adatokból készült hőtérképet is, a következő színskála alapján:

Továbbá minden oszlopban aláhúzással jelöltem a minimumot.

Mindenekelőtt szembeötlő, hogy *a hálózat felépítése* rendkívüli mértékben befolyásolja a megoldó sebességét. Az egyenletes típusú hálózatokra lehetett a leggyorsabban kiszámítani a max-min méltányos erőforrás-kiosztást, a futási idő átlaga itt 475 másodperc volt. A túlkeresletet mutató hálózatokban az átlagos futási idő egy nagyságrenddel nagyobb, 5 380 másodperc volt, míg a túlkínálatot mutató hálózatokon dolgozott legtovább a megoldó, átlagosan ismét egy nagyságrenddel tovább, 41 940 másodpercig.

Mindhárom típusú hálózatban *a 6.2.5. alszakasz előmegoldóját megvalósító* öt algoritmus variáns (a referencia, és a 8–11.) produkálta a legrövidebb átlagos futási időket, a 8. és 10. variáns hasonló idővel a két leggyorsabb volt. A referencia algoritmus átlagosan 58%-kal gyorsabban futott, mint a 6.2. szakaszban bemutatott módosításokat nélkülöző 5. variáns, habár egyetlen esetben 16%-kal lassabb volt annál (az 500 felhasználót és 200 torrentet tartalmazó példán a túlkínálatos teszthalmazban). Összevetve a referencia algoritmust az 1. variánssal, továbbá a 8. és 7., 9. és 6., 10. és 4., valamint a 11. és 5. variánsok futási idejét, az előmegoldó beépítése 32–88%-kal (átlagosan 61%-kal) gyorsította meg a végrehajtást.

Az 1–3. algoritmus-variánsok minden tesztesetre a leglassabbnak bizonyultak, a 2. és 3. variáns több esetben szegmentálási hibával állt le. Az 1. variáns az 5.-hez képest átlagosan 59%-kal lassabban futott, a referenciához viszonyítva tehát átlagosan több, mint négyszeres futási időt igényelt. Ugyanakkor a 8. variáns a referenciához viszonyítva átlagosan 4%-kal rövidebb futási időket produkált. Megállapítható tehát, hogy *a 6.2.2. alszakaszban bemutatott McCormick-átírás* alkalmazása az előmegoldás nélkül jelentősen, de az előmegoldással együtt is kismértékben bonyolítja a feladatot.

A 6.2.4. alszakaszban felvázolt kezdőérték-adás, az 1. és 2. variáns eredményeire alapozva, az esetek 84%-ában javított a futási időn, átlagosan 6%-kal.

A 6.2.3. alszakasz kezdőérték-adása, a 2. és 3. variáns összevetése alapján a McCormick-átírással kombinálva tizenegy esetben, vagyis az összevethető esetek 50%-ában lassított némileg a megoldáson. Ugyanakkor az 5. és 6. variáns alapján, tehát csupán a kezdőérték-adás hatását vizsgálva kedvezőbb a helyzet: az esetek 89%-ában gyorsabb volt a 6. variáns, átlagosan 6%-kal. Úgy tűnik továbbá, hogy a hálózat típusától függ a javulás nagysága: míg a túlkeresletet mutató példákban 1%-os lassulást eredményezett a kezdőérték-adás, az egyenletes hálózatokban 6%-kal gyorsabb, míg a túlkínálatot mutató hálózatokban 12%-kal gyorsabb volt a 6. variáns. A 4. és 7., 10. és 8., valamint 11. és 9. variánsok futási idejét is összehasonlítva, a 6.2.3. alszakasz kezdőérték-adását implementáló modellek átlagosan mintegy 3%-kal voltak gyorsabbak a kezdőérték-adást mellőző, minden egyéb tekintetben megegyező variánsoknál.

A 6.2.1. alszakaszban bemutatott redundáns korlát hozzáadása a 4. és 5. variáns összehasonlításában hét esetet leszámítva segít a Gurobinak, átlagosan mintegy 4%-kal futott gyorsabban a 4. variáns. Ebben a tekintetben azonban igen nagy a szórás, a legjobb esetben 30%-kal is gyorsabb volt a 4. variáns, a legrosszabb esetben azonban 13%-kal lassabb. A 10. és 11. variáns az előmegoldót is implementálja, ebben a kontextusban nagyobb hasznot hozott a plusz korlát: átlagosan 9%-kal gyorsabban futott a 10. variáns. A 6. és 7., valamint a 8. és 9. variánst is figyelembe véve átlagosan 6%-os javulást eredményez a futási idő tekintetében a 6.2.1. alszakaszban tárgyalt módosítás. Érdemes megfigyelni, hogy az egyenletes kínálatot mutató gráfokra csak kis mértékű javulást, ill. bizonyos esetekben lassulást eredményez ez a módosítás.

A megoldás minőségét tekintve nem volt jelentős különbség az egyes algoritmus-variánsok között. A többszöri futtatás eredményei is identikusak voltak, csak a futási idők különböztek.

A futási idők variációs koefficienseit mutatja a B.4–B.6. táblázat. A variációs koefficiens tulajdonképpen az átlaggal normált szórás százalékos formája. A tesztesetek eltérő mérete miatt a szórást ebben az esetben nincs értelme összehasonlítani. A Gurobi által igényelt futási idők átlagos variációs koefficiense mintegy 19% volt a teljes adathalmazra.

6.4.3. MOSEK

A B.7–B.9. táblázatok a MOSEK megoldóval elért átlagos futási időket mutatják. A Gurobival összehasonlítva ez a megoldó az esetek 65%-ában lassabb volt: a túlkeresletet mutató hálózatokra átlagosan 56%-kal, az egyenletes ke-

resletet mutató hálózatokra 151%-kal, míg a túlkínálatot mutató hálózatokra átlagosan 20%-kal több időt igényelt, mint a Gurobi.

A hálózat felépítésétől itt is nagy mértékben függött a megoldó sebessége. Az arányok a Gurobihoz hasonlóak voltak: az egyenletes típusú hálózatok max-min méltányos erőforrás-kiosztását átlagosan 884 másodperc alatt lehetett kiszámítani, a túlkeresletet mutató hálózatokra ugyanez 3 966 másodpercig tartott, a túlkínálatot mutató hálózatokra pedig átlagosan 34 649 másodpercig.

Érdekes, hogy a 4–7. algoritmus-variánsok produkálták a leggyorsabb futási időket, ugyanakkor a MOSEK ezekre a modellekre jelentősen eltérő optimumot talált mind a három teszthalmazban, mint a Gurobi. A többi esetben nem volt jelentős eltérés a különböző variánsok által talált optimumban. Felmerül a kérdés, hogy egyáltalán itt miről is van szó. Az algoritmus végeredményeként egy valós számokból álló vektort ad meg, amelynek hossza megegyezik az E_D letöltő élek halmazának méretével. Mivel a használt programok lebegő-pontos műveleteket végeznek, ezért kerekítési hiba előfordulhat, amely befolyásolhatja a végeredményt (ld. az 5.5.2. alszakaszt). Jelen esetben pontosan ezt tapasztaltuk, tehát a két megoldó a kerekítési hibákra különbözőképpen érzékeny.

A Gurobihoz hasonlóan itt is az előmegoldót nélkülöző, *McCormick-átírás*t implementáló 1–3. algoritmus-variánsok voltak a leglassabbak, mindhárom produkált szegmentálási hibát is a legnagyobb feladatokon.

A 6.2.4. alszakasz kezdőérték-adása, *a 6.2.3. alszakasz kezdőérték-adása*, és *a 6.2.1. alszakasz redundáns korlátja* átlagosan mintegy 4%, 3%, ill. 5% lassulást eredményezett a MOSEK megoldóval kombinálva.

A referencia átlagosan 65%-kal gyorsabban futott, mint az 1. algoritmusváriáns, ráadásul ebben az összehasonlításban (tehát a McCormick-átírással kombinálva) minden esetben több, mint 50%-os gyorsulást eredményezett *a 6.2.5. alszakasz előmegoldója*. Ugyanakkor a 8–11. algoritmus-variánsokat az előmegoldót nem implementáló párjaikkal összehasonlítva átlagosan 67%-os lassulást tapasztaltunk a MOSEK esetében.

A futási idők variációs koefficienseit a B.10–B.12. táblázat tartalmazza. A MOSEK által igényelt futási idők átlagos variációs koefficiense a Gurobitól nagyobb, átlagosan 30% volt a teljes adathalmazra.

6.4.4. Konklúzió

A 6.3. táblázat tartalmazza egy-egy algoritmus-variáns átlagos futási idejét az egyes teszthalmazokra. A 300 felhasználót és 100 ill. 200 torrentet, továbbá az 500 felhasználót és 100, ill. 200 torrentet tartalmazó teszteseteket mellőztük az átlagszámítás során, hogy a szegmentálási hiba miatt hiányzó adatok (ld. a 6.4.2. alfejezet elején) ne torzítsák az eredményt.

6.3. táblázat. *Egy-egy modell-változat átlagos futási ideje (másodperc)*

	Gurobi			MOSEK		
	Túlkereslet	Egyenletes	Túlkínálat	Túlkereslet	Egyenletes	Túlkínálat
ref	394,31	60,95	2 481,61	817,04	203,15	5 349,53
1	1 523,81	145,57	16 206,43	2 122,66	461,99	13 311,69
2	1 591,15	163,32	16 183,00	2 092,51	453,53	13 602,79
3	1 592,43	165,20	15 128,01	1 947,76	375,94	12 222,58
4	843,34	115,57	9 923,46	705,88	51,08	6 089,69
5	869,00	116,17	10 694,17	761,24	105,33	6 132,51
6	811,11	113,42	9 557,88	756,02	125,49	6 114,82
7	759,65	113,03	9 389,50	668,76	57,55	6 267,42
8	399,35	56,07	2 010,80	768,83	162,93	5 382,91
9	426,72	56,06	2 215,91	772,56	193,83	5 341,95
10	400,28	61,44	2 145,46	787,46	208,86	5 240,41
11	446,61	54,56	2 365,85	801,81	162,48	5 289,54

A közölt tesztek eredményei alapján a következő tanulságokat vonhatjuk le:

- A hálózat felépítésétől rendkívüli mértékben függ a megoldó sebessége. Az egyenletes típusú hálózatokra a leggyorsabb, a túlkínálatot mutató hálózatokra pedig a leglassabb kiszámítani a max-min méltányos erőforráselosztást. A sebesség nagyjából arányos az alkalmazott speciális hármas gráf reprezentáció csúcsainak és éleinek a számával.

- A Gurobi általában gyorsabban oldotta meg a feladatot, és kevésbé volt érzékeny a kerekítési hibákra, mint a MOSEK. Ugyanakkor a Gurobi több tesztesetre produkált szegmentálási hibát.

- A Gurobi kedvezőbben reagált a cikkben szereplő modell-átírásokra.

- A fix alsó korlát hozzáadása az alkalmazott megoldótól és a feladat típusától függően eltérő hatást gyakorolt, az összes teszt átlagában +0,66% javulást eredményezett a megoldás sebességében.
- A McCormick-átírás alkalmazása minden esetben lassította a megoldást. Ez meglepő és egyben pozitív eredmény, amellyel kimutattuk, hogy a tesztelt megoldók könnyebben boldogultak a kisebb méretű nemlineáris feladattal, mint a nagyobb méretű lineáris változattal.
- A bináris változókra vonatkozó kezdőérték-adás hatása az alkalmazott megoldótól és a feladat típusától függően változott, az összes teszt átlagában elenyésző, mindössze +0,06% volt.
- A mesterséges változókra vonatkozó kezdőérték-adás hatása az alkalmazott megoldótól és a feladat típusától függően változott, az összes teszt átlagában +1,01% volt.
- A cikkben szereplő modell-átírások közül az előmegoldó hatása a legkedvezőbb, az összes teszt átlagában +9,75% javulást eredményezett.
- Nem volt olyan algoritmus-variáns, amely minden teszthalmaz esetén egyértelműen a legjobb lett volna, még azonos megoldó esetében sem.

7

Összefoglalás

Milyen hatást gyakorolnak a nemlineáris optimalizálási feladatok modellezési fázisában hozott döntések a megoldás hatékonyságára? Erre a kérdésre kerestem a választ dolgozatom megírásakor. Ez a fejezet összefoglalja értekezésem fő eredményeit, továbbá megfogalmazódnak benne a kutatás folytatásának lehetséges irányai.

A 2. fejezetben az értekezés témájához kapcsolódó *alapfogalmak*at ismertettem. Bemutattam az optimalizálási problémák szokásos osztályozását és megoldó módszereit, a modellezési nyelvek szerepét a praktikusan nagy problémák kezelésében.

A 3. fejezetben mindenekelőtt annak a lehetőségét vizsgáltam, hogy Csendes és Rapcsák nemlineáris koordináta-traszformációi [20, 71] előállíthatók-e egy mai számítógépes algebra rendszerben megvalósított *automatikus egyszerűsítő eljárás*sal. Két programozási környezetben, a Maple és Mathematica rendszerekben készült el a hivatkozott elméleti eredmények alapján kidolgozott eljárás megvalósítása, ami a feltétel nélküli nemlineáris optimalizálási feladat célfüggvénye képletének ismeretében automatikusan képes előállítani a matematikai modell hasznosnak tűnő átírásait [4, 3]. A javasolt átírások révén felismerhetővé válhat a modell rejtett redundanciája, lehetőség nyílhat a feladat dimenziójának csökkentésére. Bizonyos esetekben a redundáns paraméterek felismerése felbecsülhetetlen segítséget nyújthat az optimalizálási modelleket elemző szakértő számára [38].

A 3.4.3–3.4.5. alszakaszok a Maple implementáció kapcsán felmerülő főbb problémákat tárgyalták. A 3.4.8. alszakaszban összehasonlítottam a Maple és Mathematica változatokat az előállított helyettesítések szempontjából. Itt különösen a saját, erre a célra készült tesztfeladataim vizsgálatára támaszkodtam. Kiderült, hogy a Mathematica rugalmas helyettesítő rutinja erre a célra megfelelőbb, és ez a megbízható intervallum aritmetikával kiegészítve a Maple-nél jobb alapot biztosít egy ilyen jellegű alkalmazás számára.

Természetesen az lenne a legszerencsésebb, ha az automatikus egyszerűsítő eljárás a matematikai programozásra specializált modellezési nyelvek előfeldolgozójába is beépítésre kerülne, hasonlóan az 5.4.3. és 6.2.5. alszakaszban részletesebben tárgyalt, az AMPL-ben rendelkezésre álló lineáris előmegoldó technikához. Ahhoz azonban, hogy a tárgyalt nemlineáris koordinátatraszformációk hatékonyan alkalmazhatók legyenek egy ilyen környezetben, további elméleti megfontolások szükségesek, amint azt mindjárt kifejtem.

Hogyan teljesít az elkészült program standard és egyéb gyakran használt globális optimalizálási tesztfeladatok egy bővebb halmazán? Összesen 45 jól ismert globális optimalizálási tesztfeladatot vizsgáltam, és ezek közül 8 esetben talált ekvivalens átírást a Mathematica program, ami a teszthalmaz 18%-a. Ha figyelembe vesszük, hogy nem ismert más módszer, amivel hasonló jellegű helyettesítések automatikusan előállíthatók lennének, ez az arány jelentősnek tekinthető.

A 3.4.8. és 3.4.9. alszakaszokban bemutatott, az automatikus egyszerűsítő Mathematica változatával előállított helyettesítések mindegyike helyes, felmerül azonban két igen fontos kérdés:

1. Hasznosak-e, vagy jelentéktelenek a produkált helyettesítések?
2. Megéri-e a plusz vizsgálódást ezek előállítása?

Ezen kérdések mentén folytatott kutatásom eredményeit a 3. fejezet vége és a 4. fejezet foglalja össze.

Az 1. kérdésre keresve a választ, *a 4. fejezetben* az automatikus egyszerűsítő által produkált átírások hatását egy numerikus globális optimalizáló teljesítményén vizsgáltam. Konkrétan a GLOBAL nevű heurisztikus multistart megoldó futási idejét és függvény-kiértékeléseinek a számát hasonlítottam össze az eredeti, és a bemutatott automatikus egyszerűsítő által átírt probléma-alakra vonatkozóan. A 3.4.8. és 3.4.9. alszakaszokban is vizsgált standard globális optimalizálási tesztfeladatok és egyéb mesterségesen konstruált példák vizsgálatából származó eredményeim azt mutatják, hogy a GLOBAL a legegyszerűbbnek tűnő automatikus átírásból is profitálni tud. Mind a futási időre, mind a függvény-kiértékelések számára kedvezően hatott az átalakítás, az összes tesztfeladat átlagában az átírásnak köszönhető relatív javulás mindkét tekintetben 32% volt.

A 3.5. szakaszban bemutattam új elméleti eredményeimet, melyekkel a Csendes és Rapcsák által leírt nemlineáris koordináta-transzformációkat két irányban általánosítottam [3]. Egyrészt az alkalmas párhuzamos helyettesítések leírásával a korábbinál bonyolultabb ekvivalens átírásokra adtam

elégséges feltételeket, másrészt kiterjesztettem az alkalmazás lehetőségét a feltételes nemlineáris optimalizálási feladatokra. Az eredményeim gyakorlatba ültetéséhez, pontosabban az automatikus egyszerűsítőbe történő hatékony integrációjához további megfontolás lenne szükséges. A munka lehetséges folytatása lenne annak elemzése, hogy milyen feltételek mellett lehet az alkalmas párhuzamos helyettesítéseket ügyesen megtalálni és végrehajtani? Vajon hol lehet, vagy lehet-e egyáltalán a megbízhatóság biztosítása mellett a kiszámítási fában vágni, tehát az egyszerűsítő eljárás futás közben keletkező szálait rangsorolni, ill. elhagyni?

A 2. kérdéshez kapcsolódóan a 3.4.9. alszakaszban közöltem az egyszerűsítő eljárás Mathematicában mért futási idejét. A kép vegyes: míg a saját feladatok átalakításainak mindegyike kevesebb, mint 0,2 másodperc alatt lezajlott, a standard tesztfeladatok futási ideje sokkal nagyobb szórást mutatott. A 45-ből 24 tesztesetre kevesebb, mint egy másodperc alatt lefutott az egyszerűsítő, és további 10 esetben kevesebb, mint egy perc elegendőnek bizonyult. 7 esetben viszont több, mint fél órát igényelt volna az eljárás. A 4. fejezetben látható, hogy ugyanezen feladatok megoldása a GLOBAL-lal átlagosan kevesebb, mint egy másodpercet vett igénybe.

A továbbiakban érdemes lenne tehát megvizsgálni, melyek azok a feladatosztály–megoldó kombinációk, amelyekre az átalakítás költsége és az új alak optimalizálása együtt kevesebb időt igényel, mint az eredeti feladat megoldása. A szimbolikus előfeldolgozónk használata elsősorban akkor lehet indokolt, ha *megbízható* eredményt keresünk, vagyis az optimalizálási folyamatban végig megbízhatóan számolunk: intervallumosan vagy szimbolikusan. Ilyen esetekben kétféleképpen is megtérülhet az előfeldolgozás többletköltsége: az átalakítás segítheti, gyorsíthatja a heurisztikus megoldóknál általában sokkal lassabb megoldót (pl. az intervallumos korlátozás és szétválasztás [81] esetén), vagy olyan alakra hozhatja a feladatot, amit egy bizonyos feladatra specializált megoldó (pl. a kvantor elimináció [83]) meg tud oldani.

A szimbolikus feladat-átírás ígéretesnek tűnik például a globális optimalizálás intervallumos módszerei esetében. Az optimalizálási modellek intervallumos befoglaló függvényei túlbecslésének [1] a hatását vizsgálták már előttem [81]. A szimbolikus módszerek kézenfekvő alkalmazása lenne „single use expression" (SUE) előállítása a kiértékelendő kifejezésből, ha ez lehetséges. Tulajdonképpen egy átlagos számítógépes algebra rendszer alapértelmezett egyszerűsítő mechanizmusa [79] gyakran végez a kívánthoz hasonló átalakításokat a disztributivitási szabályok alkalmazásával, vagy a szorzás művelet hatványozásra történő cseréjével. A szerző folyamatban

lévő munkája azt látszik megerősíteni, hogy az értekezésben vizsgált automatikus egyszerűsítő eljárás a mostani formájában is alkalmas az intervallumos megoldók eredményességének és hatékonyságának a javítására.

Az 5. fejezetben a BitTorrent közösségekben felmerülő *méltányos sávszélesség-kiosztás problémájá*ra adott megoldásomat ismertettem. A feladatot rajközi (inter-swarm) szinten vizsgáltam. Ez a felhasználók viselkedésének egyik legkomplexebb vizsgálható szintje, ahol a modell megengedi, hogy a felhasználók több tartalom le- és feltöltésében vegyenek részt egyszerre.

Capotă és szerzőtársai [13] megmutatták, hogy a BitTorrent protokoll standard sávszélesség-kiosztását alkalmazó BitTorrent közösségek átlagos teljesítménye (nem túl meglepő módon) szuboptimális a max-min méltányosság tekintetében. Ez a méltányossági mérték megfelel például a napjainkban peer-to-peer rendszerekkel is kapcsolatba hozott videóközvetítő (streaming) szolgáltatások céljainak.

Az általunk kidolgozott algoritmus [5] Capotă és szerzőtársai munkájának továbbfejlesztése. A Radunović és Le Boudec által is összefoglalt [69] általános max-min programozási algoritmus sémáját követve a letöltő élek max-min méltányos súlyait iteratív módon számítjuk, minden körben a legkisebb meghatározatlan súllyal rendelkező változókat rögzítve. Fő eredményem, hogy Capotă és szerzőtársai komplikált szita-módszerét egy egzakt matematikai programmal helyettesítettem, majd az erre épülő algoritmus helyességét elméleti úton bizonyítottam. Az algoritmus részletes leírása a bizonyítással együtt az 5.4. szakaszban található.

A bizonyítottan helyes algoritmust AMPL nyelven, elméleti eredményeimet és néhány további átírási lehetőséget is alkalmazva valósítottuk meg. Az 5.5. szakaszban összehasonlítottuk a korábbi *MM* [13] és a saját *MaxMin-r* [5] algoritmust a BitSoup.org nevű BitTorrent közösség pillanatfelvételeiből származó nagyméretű feladatokon. Az eredmények alapján a *MaxMin-r* a nagyobb problémákon gyorsabban teljesít, mint az *MM*, ráadásul már néhány kezdeti iteráció után nagyon jó közelítését adja a max-min méltányos allokációnak. Ez a közelítés, ami a feladat fízibilis megoldása, nagyon gyorsan előállítható. Például a *MaxMin-r* első iterációja 46 másodpercet igényelt a 23 670 csúcsot és 7 326 letöltő élet tartalmazó példán, míg az *MM* algoritmus hasonlóan jó eredményt biztosító első 910 iterációja több, mint nyolc órán keresztül futott.

Tehát a *MaxMin-r* első néhány iteráció utáni megállításával belátható időn belül a max-min méltányos allokáció egy jó fízibilis közelítését állíthatjuk elő. A milliónyi csúcsot és élet tartalmazó valós problémák megoldása azonban

még ettől is gyorsabb heurisztikákat kívánhat. Így hát lehetséges folytatásként érdemes lenne egy jó heurisztikát kifejleszteni a valós BitTorrent közösségekből származó adatok elemzésére építő további elméleti megfontolások alapján. A szerző ilyen irányú munkája folyamatban van.

Mivel a max-min méltányossági problémát peer-to-peer rendszerekhez kapcsolódóan vizsgáltam, az egzakt algoritmus elosztott változata, vagy akár elosztott heurisztikák ötlete is érdeklődésre tarthat számot. Reményeim szerint a feladat jobb megértésére irányuló eredményeim a későbbiekben ebben az irányban is alkalmazhatóak lesznek.

A 6. fejezetben azt vizsgáltam, hogy az 5.4. szakaszban bemutatott, nagyméretű lineáris és nemlineáris vegyes-egészértékű programokat is magában foglaló optimalizálási feladat megoldásának sebességét mennyiben befolyásolja különféle modellezési technikák alkalmazása. Elemeztem a matematikai modellezés során felmerülő átírási lehetőségek hatásait. Kiterjedt numerikus tesztelést végeztem tizenkét modell-változat, huszonhét teszteset és kettő professzionális megoldó minden lehetséges kombinációjával.

Az elvégzett kísérletek számos érdekes eredménnyel szolgáltak. Egyrészt, nem találtam olyan modell-változatot, ami minden esetben a leggyorsabban oldaná meg a feladatot. Kiderült továbbá, hogy a tesztelt korszerű megoldók számára nem jelent problémát a bilineáris felírás hatékony megoldása. A Gurobi és a MOSEK is könnyebben boldogult a kisebb méretű bilineáris feladattal, mint a McCormick-átírás révén előálló ekvivalens, nagyobb méretű lineáris változattal. Kiemelkedett az 5.4.3. és 6.2.5. alszakaszban tárgyalt előmegoldó technika hatása, mely az összes teszt átlagában a futási idő mintegy 10%-os javulását eredményezte.

A folytatásban érdekes lehetne az elvégzett kísérletek egyfajta megfordítása is. A bemenetként megadott gráfok szerkezetének mélyebb elemzésével talán kimutatható, hogy az egyes algoritmus variánsok alkalmazása milyen gráfok esetén a legkedvezőbb a megoldás hatékonysága szempontjából.

KÖSZÖNETNYILVÁNÍTÁS

Köszönettel tartozom mindenekelőtt témavezetőim, Dr. Csendes Tibor és Dr. Vinkó Tamás áldozatos munkájáért, amivel segítettek az értekezés elkészüléséig vezető úton. Csendes Tibor iránymutatása nélkül talán soha nem léptem volna a kutatói pályára. Vinkó Tamás felbecsülhetetlen módszertani és emberi tanácsaival segített.

Hálás vagyok azért az inspiráló közegért, amit professzoraim, valamint pálya- és munkatársaim teremtettek számomra a Szegedi Tudományegyetem Kalmár Intézetében. Köszönöm mindenkinek személy szerint, akiket közülük barátomnak is nevezhetek.

Köszönöm továbbá jelenlegi munkahelyem, az átalakulófélben lévő Kecskeméti Főiskola GAMF Kara, különösen pedig a Természet- és Műszaki Alaptudományi Tanszék munkatársainak jóindulatú támogatását, amelyet doktori cselekményem során tanúsítottak.

Köszönöm a családomnak, kiemelten a szüleimnek és a bátyámnak, hogy mindig számíthatok rájuk. Nem utolsósorban pedig köszönöm férjem, Dobján Tibor türelmét és bíztatását, amivel munkámban támogat.

IRODALOMJEGYZÉK

[1] ALEFELD, G. – HERZBERGER, J.: *Introduction to Interval Computation*. New York, Academic Press, 1983. ISBN 978-0-12-049820-8.

[2] ANDRADE, N. – SANTOS NETO, E. – BRASILEIRO, F. – RIPEANU, M.: Resource demand and supply in BitTorrent content-sharing communities. In *Comput. Netw.*, 53. évf. (2009) 4. sz., 515–527. p. ISSN 1389-1286.

[3] ANTAL, E. – CSENDES, T.: Nonlinear symbolic transformations for simplifying optimization problems. In *Acta Cybernetica*, 22. évf. (2016) 4. sz., 715–733. p.

[4] ANTAL, E. – CSENDES, T. – VIRÁGH, J.: Nonlinear transformations for the simplification of unconstrained nonlinear optimization problems. In *Central European Journal of Operations Research (CEJOR)*, 21. évf. (2013) 4. sz., 665–684. p.

[5] ANTAL, E. – VINKÓ, T.: Modeling max–min fair bandwidth allocation in BitTorrent communities. In *Computational Optimization and Applications*, 2016. 18 p., `http://dx.doi.org/10.1007/s10589-016-9866-5`. Közlésre elfogadva.

[6] BAJALINOV E. – IMREH B.: *Operációkutatás*. Szeged, Polygon, 2001.

[7] BERTOLAZZI, E. – BIRAL, F. – DA LIO, M.: Symbolic–numeric indirect method for solving optimal control problems for large multibody systems. In *Multibody System Dynamics*, 13. évf. (2005) 2. sz., 233–252. p.

[8] BERTSEKAS, D. P. – GALLAGER, R. G.: *Data Networks*. 2nd. kiad. Englewood Cliffs, NJ, USA, Prentice Hall, 1992. ISBN 0132009161.

[9] BONAMI, P. – KILINÇ, M. – LINDEROTH, J.: Algorithms and software for convex mixed integer nonlinear programs. In LEE, J. – LEYFFER, S. (szerk.): *Mixed Integer Nonlinear Programming*. The IMA Volumes in Mathematics and its Applications sorozat, 154. köt. New York, NY, USA, Springer New York, 2012, 1–39. p. ISBN 978-1-4614-1927-3. URL `http://dx.doi.org/10.1007/978-1-4614-1927-3_1`.

[10] BRADLEY, S. P. – HAX, A. C. – MAGNANTI, T. L.: *Applied Mathematical Programming*. Addison-Wesley, 1977. ISBN 9780201004649.

[11] BREARLEY, A. L. – MITRA, G. – WILLIAMS, H. P.: Analysis of mathematical programming problems prior to applying the simplex algorithm. In *Mathematical Programming*, 8. évf. (1975) 1. sz., 54–83. p.

[12] BYRNE, R. – BOGLE, I.: Global optimisation of constrained non-convex programs using reformulation and interval analysis. In *Computers & Chemical Engineering*, 23. évf. (1999) 9. sz., 1341–1350. p.

[13] CAPOTĂ, M. – ANDRADE, N. – VINKÓ, T. – SANTOS, F. – POUWELSE, J. – EPEMA, D.: Inter-swarm resource allocation in BitTorrent communities. In *Proceedings of IEEE International Conference on Peer-to-Peer Computing (P2P 2011)* (konferenciaanyag). 2011, 300–309. p.

[14] CARLISLE, D.: MathML on the Web: Content MathML. `https://www.w3.org/Math/XSL/mml2002-05.xml`. [2017.01.12.].

[15] CHIENG, W.-H. – HOELTZEL, D. A.: Interactive hybrid (symbolic-numeric) system approach to near optimal design of mechanical components. In *Engineering with Computers*, 2. évf. (1987) 2. sz., 111–123. p.

[16] COHEN, B.: The BitTorrent protocol specification. `http://bittorrent.org/beps/bep_0003.html`. [2017.01.12.].

[17] COHEN, B.: Incentives build robustness in BitTorrent. In *Workshop on Economics of Peer-to-Peer systems* (konferenciaanyag), 6. köt. 2003, 68–72. p.

[18] COX, J. – COLLINS, A. – DRINKWATER, S.: Seeders, leechers and social norms: Evidence from the market for illicit digital downloading. In *Information Economics and Policy*, 22. évf. (2010) 4. sz., 299–305. p. Special Issue: Digital Piracy.

[19] CSENDES, T. – PÁL, L. – SENDÍN, J. O. H. – BANGA, J. R.: The GLOBAL optimization method revisited. In *Optimization Letters*, 2. évf. (2008) 4. sz., 445–454. p.

[20] CSENDES, T. – RAPCSÁK, T.: Nonlinear coordinate transformations for unconstrained optimization I. Basic transformations. In *Journal of Global Optimization*, 3. évf. (1993) 2. sz., 213–221. p.

[21] CUEVAS, R. – KRYCZKA, M. – CUEVAS, A. – KAUNE, S. – GUERRERO, C. – REJAIE, R.: Is content publishing in BitTorrent altruistic or profit-driven? In *Proceedings of the 6th International COnference on emerging Networking EXperiments and Technologies*, Co-NEXT konferenciasorozat. New York, NY, USA, 2010, ACM, 11:1–11:12. p. ISBN 978-1-4503-0448-1.

[22] DOBJÁNNÉ ANTAL E. – VINKÓ T.: Egy nemlineáris vegyes-egészértékű optimalizálási feladat különféle modelljeinek komparatív elemzése (in hungarian). 2016. `http://www.inf.u-szeged.hu/~antale/en/research/DAntal_MIPmodels2016.pdf`. Közlésre benyújtott kézirat (Submitted for publication).

[23] EGER, K. – KILLAT, U.: Fair resource allocation in peer-to-peer networks (extended version). In *Comput. Commun.*, 30. évf. (2007) 16. sz., 3046–3054. p.

[24] FAN, B. – LUI, J.-S. – CHIU, D.-M.: The design trade-offs of BitTorrent-like file sharing protocols. In *IEEE/ACM Transactions on Networking*, 17. évf. (2009) 2. sz., 365–376. p.

[25] FARKAS, T. – RÉV, E. – LELKES, Z.: Process flowsheet superstructures: Structural multiplicity and redundancy: Part I: Basic {GDP} and {MINLP} representations. In *Computers & Chemical Engineering*, 29. évf. (2005) 10. sz., 2180–2197. p.

[26] FOURER, R. – GAY, D. M.: Experience with a primal presolve algorithm. In HAGERM, W. W. – HEARN, D. W. – PARDALOS, P. M. (szerk.): *Large Scale Optimization: State of the Art*. Dordrecht, Kluwer Academic Publishers, 1994, 135–154. p.

[27] FOURER, R. – GAY, D. M. – KERNIGHAN, B. W.: *AMPL*. 2. kiad. Duxbury Thomson, 2003. ISBN 978-0-534-38809-6.

[28] G.-TÓTH B.: *Globális optimalizálás*. Budapest, Typotex, 2011. ISBN 978-963-279-458-7.

[29] VON ZUR GATHEN, J. – GERHARD, J.: *Modern Computer Algebra*. 2. kiad. New York, NY, USA, Cambridge University Press, 2003. ISBN 0521826462.

[30] GAY, D. M.: Symbolic-algebraic computations in a modeling language for mathematical programming. In ALEFELD, G. – ROHN, J. – RUMP,

S. – YAMAMOTO, T. (szerk.): *Symbolic Algebraic Methods and Verification Methods*. Vienna, Springer, 2001, 99–106. p. ISBN 978-3-7091-6280-4.

[31] GAYLORD, R. J. – KAMIN, S. N. – WELLIN, P. R.: *An Introduction to Programming with Mathematica®*. 2. kiad. New York, NY, Springer New York, 1996, 141–143. p. ISBN 978-1-4612-2322-1.

[32] GRANVILLIERS, L.: A symbolic-numerical branch and prune algorithm for solving non-linear polynomial systems. In *Journal of Universal Computer Science*, 4. évf. (1998) 2. sz., 125–146. p.

[33] GRIMMER, M.: Interval arithmetic in Maple with intpakX. In *PAMM*, 2. évf. (2003) 1. sz., 442–443. p.

[34] GRIMMER, M. – PETRAS, K. – REVOL, N.: Multiple precision interval packages: Comparing different approaches. In ALT, R. – FROMMER, A. – KEARFOTT, R. B. – LUTHER, W. (szerk.): *Numerical Software with Result Verification: International Dagstuhl Seminar, Dagstuhl Castle, Germany, January 19-24, 2003. Revised Papers*. Berlin, Heidelberg, Springer, 2004, 64–90. p. ISBN 978-3-540-24738-8.

[35] GROSSMANN, I. E.: MINLP optimization strategies and algorithms for process synthesis. 130. Jelentés, Pittsburgh, PA, USA, 1989, Carnegie Mellon University, Engineering Design Research Center.

[36] GUPTE, A. – AHMED, S. – CHEON, M. – DEY, S.: Solving mixed integer bilinear problems using MILP formulations. In *SIAM J. Optim.*, 23. évf. (2013) 2. sz., 721–744. p.

[37] HAHNE, E. L.: Round-robin scheduling for max-min fairness in data networks. In *IEEE Journal on Selected Areas in Communications*, 9. évf. (1991) 7. sz., 1024–1039. p.

[38] HANTOS, Z. – DAROCZY, B. – CSENDES, T. – SUKI, B. – NAGY, S.: Modeling of low-frequency pulmonary impedance in dogs. In *Journal of Applied Physiology*, 68. évf. (1990) 3. sz., 849–860. p.

[39] HECK, A.: Internal data representation and substitution. In *Introduction to Maple*. New York, NY, Springer New York, 2003, 153–174. p. ISBN 978-1-4613-0023-6.

[40] HECK, A.: Introduction to computer algebra. In *Introduction to Maple*. New York, NY, Springer New York, 2003, 11. p. ISBN 978-1-4613-0023-6.

[41] IZAL, M. – URVOY KELLER, G. – BIERSACK, E. W. – FELBER, P. A. – AL HAMRA, A. – GARCÉS ERICE, L.: *Dissecting BitTorrent: Five Months in a Torrent's Lifetime*. Berlin, Heidelberg, Springer, 2004, 1–11. p. ISBN 978-3-540-24668-8.

[42] JAIN, R.: *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling*. New York, NY, USA, Wiley–Interscience, 1991. ISBN 0471503361.

[43] KALTOFEN, E. L. – LI, B. – YANG, Z. – ZHI, L.: Exact certification in global polynomial optimization via sums-of-squares of rational functions with rational coefficients. In *Journal of Symbolic Computation*, 47. évf. (2012) 1. sz., 1–15. p.

[44] KANNO, M. – HARA, S. – ANAI, H.: Plant/controller design integration for H2 control based on symbolic-numeric hybrid optimization. In *Communications in Information and Systems*, 11. évf. 3. sz., 281–306. p.

[45] KANNO, M. – YOKOYAMA, K. – ANAI, H. – HARA, S.: Symbolic optimization of algebraic functions. In *Proceedings of the Twenty-first International Symposium on Symbolic and Algebraic Computation*, ISSAC '08 konferenciasorozat. New York, NY, USA, 2008, ACM, 147–154. p. ISBN 978-1-59593-904-3.

[46] DE KLERK, E. – ROOS, C. – TERLAKY, T.: *Nemlineáris Optimalizálás, (in Hungarian, English title: Nonlinear Optimization)*. Budapest, Aula Publishing, 2004. ISBN 963-503-323-0.

[47] KRAMER, W.: intpakX - An interval arithmetic package for Maple. In *12th GAMM - IMACS International Symposium on Scientific Computing, Computer Arithmetic and Validated Numerics (SCAN 2006)* (konferenciaanyag). 2006, 27. p.

[48] KREKÓ B.: *Optimumszámítás: nemlineáris programozás*. Budapest, Közgazdasági és Jogi Könyvkiadó, 1972.

[49] LACZKOVICH M. – T. SÓS V.: *Valós analízis II.* Budapest, Typotex, 2013, 74–76. p. ISBN 978-963-2797-33-5.

[50] LE BLOND, S. – LEGOUT, A. – LE FESSANT, F. – DABBOUS, W. – KAAFAR, M. A.: Spying the world from your laptop – Identifying and profiling content providers and big downloaders in BitTorrent. In *3rd USENIX*

Workshop on Large-Scale Exploits and Emergent Threats (LEET'10) (konferenciaanyag). 2010.

[51] LEGOUT, A. – URVOY KELLER, G. – MICHIARDI, P.: Rarest first and choke algorithms are enough. In *Proceedings of the 6th ACM SIGCOMM Conference on Internet Measurement*, IMC konferenciasorozat. New York, NY, USA, 2006, ACM, 203–216. p. ISBN 1-59593-561-4.

[52] LEYKIN, A. – VERSCHELDE, J. – ZHAO, A.: Newton's method with deflation for isolated singularities of polynomial systems. In *Theoretical Computer Science*, 359. évf. (2006) 1–3. sz., 111 – 122. p.

[53] LIBERTI, L. – CAFIERI, S. – SAVOUREY, D.: The Reformulation-Optimization Software Engine. In FUKUDA, K. – HOEVEN, J. v. d. – JOSWIG, M. – TAKAYAMA, N. (szerk.): *Mathematical Software – ICMS 2010: Third International Congress on Mathematical Software, Kobe, Japan, September 13-17, 2010. Proceedings*. Berlin, Heidelberg, Springer, 2010, 303–314. p. ISBN 978-3-642-15582-6.

[54] LIM, Y. H. – LEE, D. H. – PARK, W. H. – KOOK, K. H.: Detection and traceback of illegal users based on anonymous network in BitTorrent environment. In *Wireless Personal Communications*, 73. évf. (2013) 2. sz., 319–328. p.

[55] LOVÁSZ L.: *Algoritmusok bonyolultsága*. Budapest, Typotex, 2014. ISBN 978-963-2792-53-8.

[56] MA, R. T. B. – LEE, S. C. M. – LUI, J. C. S. – YAU, D. K. Y.: A game theoretic approach to provide incentive and service differentiation in P2P networks. In *SIGMETRICS Perform. Eval. Rev.*, 32. évf. (2004) 1. sz., 189–198. p.

[57] MAEDER, R. E.: *The Mathematica® Programmer*. Academic Press, 1994, 8. p. ISBN 978-0-12-464990-3.

[58] MAROS, I.: *Computational Techniques of the Simplex Method*. Boston, MA, Springer US, 2003, 19–48., 97–119. p. ISBN 978-1-4615-0257-9.

[59] MathDox formula editor.
`http://mathdox.org/formulaeditor/`. [2017.01.12.].

[60] MathJax.
`https://www.mathjax.org/`. [2017.01.12.].

[61] MEULPOLDER, M. – D'ACUNTO, L. – CAPOTĂ, M. – WOJCIECHOWSKI, M. – POUWELSE, J. A. – EPEMA, D. H. – SIPS, H. J.: Public and private BitTorrent communities: a measurement study. In *Proceedings of the 9th International Workshop on Peer-to-Peer Systems (IPTPS)* (konferenciaanyag). 2010.

[62] MÉSZÁROS, Cs. – SUHL, U. H.: Advanced preprocessing techniques for linear and quadratic programming. In *OR Spectrum*, 25. évf. (2003) 4. sz., 575–595. p.

[63] MOL, J. J. D. – BAKKER, A. – POUWELSE, J. A. – EPEMA, D. H. J. – SIPS, H. J.: The design and deployment of a BitTorrent live video streaming solution. In *Proceedings of 11th IEEE International Symposium on Multimedia* (konferenciaanyag). 2009, 342–349. p.

[64] NEOS Solver Access Statistics. `https://neos-server.org/neos/report.html`. [2017.01.12.].

[65] NEUMAIER, A.: Improving interval enclosures. 2008. `https://www.sfu.ca/~ssurjano/optimization.html`. Kézirat.

[66] ORE, O.: *Theory of Graphs*. Colloquium Publications sorozat, 38. köt. Providence, RI, USA, American Mathematical Society, 1962, 105. p. ISBN 978-0-8218-1038-5.

[67] Overview of OpenMath. `http://www.openmath.org/overview/index.html`. [2017.01.12.].

[68] PÁL, L.: *Global optimization algorithms for bound constrained problems*. PhD értekezés (University of Szeged). Hungary, 2011.

[69] RADUNOVIĆ, B. – LE BOUDEC, J.-Y.: A unified framework for max-min and min-max fairness with applications. In *IEEE/ACM Transactions on Networking*, 15. évf. (2007) 5. sz., 1073–1083. p.

[70] RAMAN, R. – GROSSMANN, I.: Symbolic integration of logic in mixed-integer linear programming techniques for process synthesis. In *Computers & Chemical Engineering*, 17. évf. (1993) 9. sz., 909–927. p.

[71] RAPCSÁK, T. – CSENDES, T.: Nonlinear coordinate transformations for unconstrained optimization II. Theoretical background. In *Journal of Global Optimization*, 3. évf. (1993) 3. sz., 359–375. p.

[72] ROSENBROCK, H. H.: An automatic method for finding the greatest or least value of a function. In *The Computer Journal*, 3. évf. (1960) 3. sz., 175–184. p. URL `http://comjnl.oxfordjournals.org/content/3/3/175.abstract`.

[73] RUMP, S. M.: INTLAB — INTerval LABoratory. In CSENDES, T. (szerk.): *Developments in Reliable Computing*. Dordrecht, Springer Netherlands, 1999, 77–104. p. ISBN 978-94-017-1247-7.

[74] SANDVINE: Global Internet phenomena report: 1H 2014. `https://www.sandvine.com/trends/global-internet-phenomena/`. [2017.01.12.].

[75] SCHICHL, H. – NEUMAIER, A.: Interval analysis on directed acyclic graphs for global optimization. In *Journal of Global Optimization*, 33. évf. (2005) 4. sz., 541–562. p.

[76] SCHÖPF, R. – DEUFLHARD, P.: OCCAL A mixed symbolic-numeric Optimal Control CALculator. In BULIRSCH, R. – KRAFT, D. (szerk.): *Computational Optimal Control*. Basel, Birkhäuser Basel, 1994, 269–278. p. ISBN 978-3-0348-8497-6.

[77] SCHRIJVER, A.: *Theory of Linear and Integer Programming*. New York, NY, USA, John Wiley & Sons, Inc., 1986. ISBN 0-471-90854-1.

[78] SHCHERBINA, O. – NEUMAIER, A. – SAM HAROUD, D. – VU, X.-H. – NGUYEN, T.-V.: *Benchmarking Global Optimization and Constraint Satisfaction Codes*. Berlin, Heidelberg, Springer Berlin Heidelberg, 2003, 211–222. p. ISBN 978-3-540-39901-8.

[79] STOUTEMYER, D. R.: Ten commandments for good default expression simplification. In *Journal of Symbolic Computation*, 46. évf. (2011) 7. sz., 859 – 887. p.

[80] STURM, T. – TIWARI, A.: Verification and synthesis using real quantifier elimination. In *Proceedings of the 36th International Symposium on Symbolic and Algebraic Computation*, ISSAC '11 konferenciasorozat. New York, NY, USA, 2011, ACM, 329–336. p. ISBN 978-1-4503-0675-1.

[81] TÓTH, B. – CSENDES, T.: Empirical investigation of the convergence speed of inclusion functions in a global optimization context. In *Reliable Computing*, 11. évf. (2005) 4. sz., 253–273. p.

[82] TSCHORSCH, F. – SCHEUERMANN, B.: Tor is unfair – and what to do about it. In *IEEE 36th Conference on Local Computer Networks (LCN), 2011* (konferenciaanyag). 2011, 432–440. p.

[83] VAJDA, R.: Effective real quantifier elimination. In VAJDA, R. – KARSAI, J. (szerk.): *Interesting Mathematical Problems in Sciences and Everyday Life*. University of Szeged, University of Novi Sad, 2011. ISBN 978-963-306-109-1.

[84] Virtual Library of simulation Experiments: Test Functions and Datasets. Optimization Test Problems. `https://www.sfu.ca/~ssurjano/optimization.html`. [2017.01.12.].

[85] VLAVIANOS, A. – ILIOFOTOU, M. – FALOUTSOS, M.: BiToS: Enhancing BitTorrent for supporting streaming applications. In *Proceedings of the IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications* (konferenciaanyag). 2006, 1–6. p.

[86] VÍZVÁRI B.: *Egészértékű programozás*. Budapest, Typotex, 2006. ISBN 978-963-9664-29-6.

[87] VÍZVÁRI B.: *Operációkutatási modellek*. Budapest, Typotex, 2009. ISBN 978-963-2790-22-0.

[88] WELLIN, P.: Functional programming. In *Programming with Mathematica®: An Introduction*. Cambridge University Press, 2014, 115–188. p. ISBN 978-1-107-00946-2.

[89] Wolfram Language & System Documentation center. Wolfram Language tutorial: Basic internal architecture. `http://reference.wolfram.com/language/tutorial/BasicInternalArchitecture.html`. [2017.01.12.].

[90] Wolfram Mathematica. Software Development: CUDA and OpenCL support. `http://www.wolfram.com/mathematica/new-in-8/cuda-and-opencl-support/`. [2017.01.12.].

[91] WU, D. – LIANG, C. – LIU, Y. – ROSS, K.: View-upload decoupling: a redesign of multi-channel P2P video systems. In *Proceedings of the IEEE INFOCOM 2009* (konferenciaanyag). 2009, 2726–2730. p.

[92] WU, D. – LIANG, Y. – HE, J. – HEI, X.: Balancing performance and fairness in P2P live video systems. In *IEEE Transactions on Circuits and Systems for Video Technology*, 23. évf. (2013) 6. sz., 1029–1039. p.

[93] YAN, Y. – EL ATAWY, A. – AL SHAER, E.: Ranking-based optimal resource allocation in peer-to-peer networks. In *Proceedings of the IEEE INFOCOM 2007. 26th IEEE International Conference on Computer Communications* (konferenciaanyag). 2007, 1100–1108. p.

[94] ZHANG, C. – DHUNGEL, P. – WU, D. – LIU, Z. – ROSS, K.: BitTorrent darknets. In *Proceedings of the IEEE INFOCOM 2010* (konferenciaanyag). 2010, 1–9. p.

[95] ZHANG, X. – LIU, J. – LI, B. – YUM, Y. S. P.: CoolStreaming/DONet: a data-driven overlay network for peer-to-peer live media streaming. In *Proceedings of the IEEE INFOCOM 2005. 24th Annual Joint Conference of the IEEE Computer and Communications Societies* (konferenciaanyag), 3. köt. 2005, 2102–2111. p.

[96] ZHI, L.: Numerical optimization in hybrid symbolic-numeric computation. In *Proceedings of the 2007 International Workshop on Symbolic-numeric Computation*, SNC '07 konferenciasorozat. New York, NY, USA, 2007, ACM, 33–35. p. ISBN 978-1-59593-744-5.

A

Algoritmusok

A.1. algoritmus. `symbsimp(x, f)` *// f: célfüggvény képlete, x: változó lista*

```
g ← f
subslist ← {}
subsnumber ← 0
for i = 1 to Dimension(x) do
   dx ← ▽g
   factordx ← Factor(dx)
   h_i ← Null
   if factordx ≠ dx then
      h_list ← Null
      for j ∈ factordx do
         p ← ∫ j dx_i
         if NumbOccur(g, p) = NumbOccur(g, x_i) then
            repeat
               h_temp ← Sort({q | q ∈Decompose(g, p) and IsType(q, '+')}, p).Pop
               if Test(h_temp, x_i) then
                  h_list.Push(h_temp)
               end if
            until Test(h_temp, x_i) or h_temp = Null
         end if
      end for
      h_i ← Sort(h_list, x_i).Front
   end if
   if h_i = Null then
      repeat
         h_temp ← Sort({q | q ∈Decompose(g, x_i) and IsLinear(q, x_i)}, x_i).Pop
      until Test(h_temp, x_i) or h_temp = Null
      if Test(h_temp, x_i) then
         h_i ← h_temp
         subsnumber++
      else
         h_i ← x_i
      end if
   end if
   g ← Subs(g, h_i, 'y_i')
   subslist.Push("y_i =h_i")
end for
if subsnumber ≠ 0 then
   g* ← Solve[g, y]
   f* ← Transform(g*, subslist)
   Verify(f, g)
end if
```

A.2. algoritmus. *MaxMin-r*

1. **Alsó korlát számítása a folyam értékekre.** MM_0 megoldása:

$$\begin{aligned} &\max\ f, \\ \text{f.h.}\quad & f(l_j^t, d_j) \geqslant f && \forall (l_j^t, d_j) \in E_D. \end{aligned}$$

   A minimális folyam érték eltárolása. Legyen $\phi := f$.

2. **Maximális átvitel kiszámítása.** Az MM_{MaxFlow} -val jelölt LP feladat megoldása:

$$\begin{aligned} \max \sum_{(l_j^t, d_j) \in E_D} & f(l_j^t, d_j), \\ \text{f.h.}\quad & f(l_j^t, d_j) \geqslant \phi && \forall (l_j^t, d_j) \in E_D. \end{aligned}$$

   Optimum eltárolása. Legyen $\sigma := \sum_{(l_j^t, d_j) \in E_D} f(l_j^t, d_j)$.

3. **Inicializálás.** Legyen $F := \emptyset$, $k := 1$, $E_1 := E_D$, $\forall (l_j^t, d_j) \in E_D : \ell_j^t := 0$, $\phi_0 = 0$.

4. **LP megoldás (max-min folyam érték kiszámítása).** Az $mMM_k^{(1)}$ LP feladat módosított változatának megoldása:

$$\begin{aligned} &\max\ f_k, \\ \text{f.h.} \sum_{(l_j^t, d_j) \in E_k} f(l_j^t, d_j) + \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} & \ell_j^t \geqslant (1-\epsilon) \cdot \sigma \\ & f(l_j^t, d_j) \geqslant f_k && \forall (l_j^t, d_j) \in E_k, \\ & f_k \geqslant \phi. \end{aligned}$$

   Optimum eltárolása. Legyen $\phi_k := f_k$.

5. **Előmegoldás (max-min folyammal rendelkező élek kiválasztása).**

$$E_{k_f} := \left\{ (l_j^t, d_j) \in E_k \;\middle|\; \frac{c(d_j) - \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t}{deg_k^-(d_j)} = \phi_k \right\},$$

$$\forall (l_j^t, d_j) \in E_{k_f} : x_j^t := 0.$$

   Ha $|E_{k_f}| \neq 0$, ugrás a 7. lépésre.

– folytatás a következő oldalon –

6. **MILP megoldás (max-min folyammal rendelkező élek kiválasztása).** Az $mMM_k^{(2)}$ McCormick átírásának megoldása:

$$\max \sum_{(l_j^t, d_j) \in E_k} x_j^t,$$

$$\text{f.h.} \sum_{(l_j^t, d_j) \in E_k} p_j^t + \phi_k \cdot \sum_{(l_j^t, d_j) \in E_k} (1 - x_j^t) + \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t \geqslant (1 - \epsilon) \cdot \sigma,$$

$$\begin{aligned}
f(l_j^t, d_j) &\geqslant \phi_k && \forall (l_j^t, d_j) \in E_k, \\
f(l_j^t, d_j) &> \phi_k \, x_j^t && \forall (l_j^t, d_j) \in E_k, \\
\min \left(\delta_j \, x_j^t, \, f(l_j^t, d_j) \right) &\geqslant p_j^t && \forall (l_j^t, d_j) \in E_k, \\
\max \left(0, \, f(l_j^t, d_j) - \delta_j \, (1 - x_j^t) \right) &\leqslant p_j^t && \forall (l_j^t, d_j) \in E_k,
\end{aligned}$$

ahol $x_j^t \in \{0, 1\}$ és $p_j^t = f(l_j^t, d_j) \, x_j^t$.

7. **Fixálás (kiválasztott éleken folyam rögzítése).** A ϕ_k-ra vonatkozó aktív korlátok kikeresése, és a kapcsolódó letöltő éleken a folyam értékek rögzítése:

$$\Phi_k := \left\{ (l_j^t, d_j) \in E_k \mid x_j^t = 0 \right\},$$

$$\forall (l_j^t, d_j) \in E_k\text{-ra, ahol } x_j^t = 0 : \ \ell_j^t := \phi_k,$$

$$F := F \cup \Phi_k, \quad E_{k+1} := E_k \setminus \Phi_k.$$

8. **Megállási feltétel.** Ha $F = E_D$, megállunk. Különben $k := k + 1$ és ugrás a 4. lépésre.

A.3. algoritmus. *MaxMin-r egyszerűbben: a 6. fejezet kiindulópontja*

1. **Alsó korlát számítása a folyam értékekre.** MM_0 megoldása:

$$\begin{aligned} &\max\ f, \\ \text{f.h.}\quad & f(l_j^t, d_j) \geqslant f && \forall (l_j^t, d_j) \in E_D. \end{aligned}$$

   A minimális folyam érték eltárolása. Legyen $\phi := f$.

2. **Maximális átvitel kiszámítása.** Az MM_{MaxFlow} -val jelölt LP feladat megoldása:

$$\begin{aligned} \max\ & \sum_{(l_j^t, d_j) \in E_D} f(l_j^t, d_j), \\ \text{f.h.}\quad & f(l_j^t, d_j) \geqslant \phi && \forall (l_j^t, d_j) \in E_D. \end{aligned}$$

   Optimum eltárolása. Legyen $\sigma := \sum_{(l_j^t, d_j) \in E_D} f(l_j^t, d_j)$.

3. **Inicializálás.** Legyen $F := \emptyset$, $k := 1$, $E_1 := E_D$, $\forall (l_j^t, d_j) \in E_D : \ell_j^t := 0$, $\phi_0 = 0$.

4. **LP megoldás (max-min folyam érték kiszámítása).** Az $mMM_k^{(1)}$ -gyel jelölt LP feladat megoldása:

$$\begin{aligned} &\max\ f_k, && \text{(A.1)} \\ \text{f.h.}\ & \sum_{(l_j^t, d_j) \in E_k} f(l_j^t, d_j) + \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t \geqslant (1-\epsilon) \cdot \sigma \\ & f(l_j^t, d_j) \geqslant f_k && \forall (l_j^t, d_j) \in E_k. \end{aligned}$$

   Optimum eltárolása. Legyen $\phi_k := f_k$.

5. **Előmegoldás (max-min folyammal rendelkező élek kiválasztása).**

$$E_{k_f} := \left\{ (l_j^t, d_j) \in E_k \,\middle|\, \frac{c(d_j) - \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t}{\deg_k^-(d_j)} = \phi_k \right\},$$

$$\forall (l_j^t, d_j) \in E_{k_f} : x_j^t := 0.$$

   Ha $|E_{k_f}| \neq 0$, ugrás a 7. lépésre.

– folytatás a következő oldalon –

6. **MINLP megoldás (max-min folyammal rendelkező élek kiválasztása).** Az $mMM_k^{(2)}$-vel jelölt vegyes-egészértékű bilineáris programozási feladat megoldása:

$$\max \sum_{(l_j^t, d_j) \in E_k} x_j^t,$$

$$\text{f.h.} \sum_{(l_j^t, d_j) \in E_k} f(l_j^t, d_j)\, x_j^t \;+\; \phi_k \cdot \sum_{(l_j^t, d_j) \in E_k} (1 - x_j^t) \;+\; \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t \;=\; (1-\epsilon)\cdot\sigma, \qquad \text{(A.2)}$$

$$f(l_j^t, d_j) \geqslant \phi_k \qquad \forall (l_j^t, d_j) \in E_k,$$

$$f(l_j^t, d_j) > \phi_k\, x_j^t \qquad \forall (l_j^t, d_j) \in E_k,$$

ahol $x_j^t \in \{0, 1\}$.

7. **Fixálás (kiválasztott éleken folyam rögzítése).** A ϕ_k-ra vonatkozó aktív korlátok kikeresése, és a kapcsolódó letöltő éleken a folyam értékek rögzítése:

$$\Phi_k := \left\{ (l_j^t, d_j) \in E_k \,|\, x_j^t = 0 \right\},$$

$$\forall (l_j^t, d_j) \in E_k\text{-ra, ahol } x_j^t = 0 \,:\, \ell_j^t := \phi_k,$$

$$F := F \cup \Phi_k, \quad E_{k+1} := E_k \setminus \Phi_k.$$

8. **Megállási feltétel.** Ha $F = E_D$, megállunk. Különben $k := k + 1$ és ugrás a 4. lépésre.

B

A 6. fejezet táblázatai

B.1. táblázat. *Futási idő átlaga Gurobi megoldóval a túlkeresletet mutató hálózatokra (másodperc)*

	100-50	100-100	100-200	300-50	300-100	300-200	500-50	500-100	500-200
ref	10,07	18,19	45,94	325,74	673,29	1 646,35	1 571,61	4 156,12	10 037,23
1	24,70	52,59	163,54	1 005,58	2 893,92	10 131,24	6 372,63	22 021,62	74 968,79
2	24,28	57,54	178,90	1 032,78	n.a.	11 193,77	6 662,24	24 129,19	81 453,34
3	24,37	55,31	171,21	1 014,39	n.a.	11 223,16	6 696,87	26 537,26	n.a.
4	19,95	34,52	106,72	637,97	1 550,51	4 406,24	3 417,54	9 616,70	24 960,11
5	19,40	36,94	119,00	624,85	1 611,28	4 884,42	3 544,83	9 812,69	26 715,23
6	19,25	36,23	109,95	599,36	1 982,60	6 648,88	3 290,78	8 479,57	22 512,50
7	16,08	27,04	90,44	574,74	1 880,29	6 200,34	3 089,95	7 892,06	20 627,51
8	8,68	14,76	42,13	304,23	696,97	1 854,49	1 626,95	4 081,75	10 172,45
9	9,15	17,78	59,14	348,47	785,58	2 228,94	1 699,07	4 561,72	12 356,00
10	9,23	14,57	52,28	302,89	699,93	1 849,76	1 622,43	4 074,37	10 322,80
11	9,24	19,13	65,44	353,52	765,38	2 287,31	1 785,71	4 582,57	12 446,04

B.2. táblázat. *Futási idő átlaga Gurobi megoldóval az egyenletes keresletet mutató hálózatokra (másodperc)*

	100-50	100-100	100-200	300-50	300-100	300-200	500-50	500-100	500-200
ref	1,78	3,26	6,78	52,74	95,49	202,50	240,21	506,27	1 229,56
1	4,39	8,71	17,19	130,11	238,36	529,75	567,47	1 281,63	3 038,54
2	4,43	9,87	20,11	125,07	264,64	623,30	657,11	1 365,29	3 352,50
3	4,44	9,52	20,73	136,39	254,63	618,02	654,93	1 303,40	3 355,22
4	3,40	6,87	14,39	103,26	199,23	492,47	449,93	1 411,48	2 837,84
5	3,31	6,88	14,69	101,57	201,26	707,21	454,40	1 241,90	2 699,43
6	3,07	6,72	15,03	93,24	200,98	590,48	449,03	1 179,65	2 359,24
7	3,03	6,61	15,03	98,02	198,27	590,40	442,46	1 096,27	2 488,53
8	1,55	3,26	7,83	50,19	107,44	230,83	217,54	750,42	1 315,50
9	1,59	3,14	7,83	46,81	106,70	258,43	220,95	506,30	1 125,56
10	1,54	3,30	7,56	50,32	106,27	235,28	244,48	487,59	1 162,61
11	1,54	3,14	8,39	50,98	106,82	269,73	208,76	495,27	1 156,44

B.3. táblázat. *Futási idő átlaga Gurobi megoldóval a túlkínálatot mutató hálózatokra (másodperc)*

	100-50	100-100	100-200	300-50	300-100	300-200	500-50	500-100	500-200
ref	44,6	99,7	394,9	1 767,2	3 602,1	44 689,1	10 101,6	19 961,0	142 826,0
1	164,5	437,9	1 816,3	11 773,9	23 278,4	368 166,9	66 839,7	166 285,2	393 350,0
2	203,0	410,6	1 885,5	11 005,9	n.a.	369 862,3	67 410,0	171 311,6	400 560,6
3	204,1	417,7	1 743,2	11 536,0	n.a.	n.a.	61 739,1	169 778,3	n.a.
4	133,5	281,2	1 359,3	6 527,9	19 146,2	110 138,4	41 315,4	82 407,8	124 545,3
5	133,7	332,3	1 596,2	7 481,8	20 126,7	120 270,8	43 926,9	86 053,9	123 590,6
6	118,7	312,9	1 350,5	6 805,6	18 939,6	93 868,0	39 201,7	76 569,0	104 913,9
7	87,8	321,4	1 385,5	6 197,5	17 703,8	87 309,6	38 955,4	72 105,7	94 794,1
8	52,8	85,9	406,7	1 741,2	3 756,6	30 590,8	7 767,5	17 273,4	28 178,8
9	59,1	125,7	516,3	1 842,9	4 019,9	34 418,8	8 535,7	19 165,4	34 922,3
10	57,8	109,9	375,1	1 981,4	4 090,4	29 545,9	8 203,2	16 128,2	27 372,2
11	60,3	124,5	454,0	2 064,1	4 357,5	37 142,2	9 126,4	18 057,1	35 088,7

B.4. táblázat. *Futási idő variációs koefficiense Gurobi megoldóval a túlkeresletet mutató hálózatokra (százalék)*

	100-50	100-100	100-200	300-50	300-100	300-200	500-50	500-100	500-200
ref	37,03	33,80	18,61	27,12	20,64	25,82	18,35	23,75	22,83
1	34,34	37,36	16,41	21,98	20,05	21,80	17,62	20,01	25,10
2	27,51	40,27	18,49	21,23	n.a.	23,50	20,42	22,06	26,76
3	25,44	33,19	13,69	21,14	n.a.	19,76	21,31	21,88	n.a.
4	35,66	30,57	34,26	22,45	26,29	20,16	20,74	22,31	23,46
5	35,38	26,25	32,78	19,59	22,42	22,45	21,55	21,47	22,68
6	37,45	27,75	28,32	20,41	26,28	17,84	22,44	22,52	21,89
7	14,44	1,34	21,07	20,94	23,22	17,96	20,95	23,24	22,52
8	18,46	4,78	2,70	22,50	25,10	21,53	22,45	22,38	22,57
9	20,80	5,30	17,75	25,92	27,02	21,58	19,57	21,48	22,57
10	29,47	1,15	36,84	21,31	27,73	21,35	22,01	21,49	22,81
11	27,70	12,88	34,04	25,58	24,55	23,15	22,32	21,50	22,78

B.5. táblázat. *Futási idő variációs koefficiense Gurobi megoldóval az egyenletes keresletet mutató hálózatokra (százalék)*

	100-50	100-100	100-200	300-50	300-100	300-200	500-50	500-100	500-200
ref	17,45	23,02	6,03	28,77	13,98	8,29	12,43	13,99	23,03
1	23,83	16,00	12,12	35,90	3,83	5,72	19,80	18,73	11,56
2	21,91	23,19	14,36	19,28	11,32	8,95	32,06	11,79	10,56
3	22,74	24,15	9,63	35,74	7,22	5,22	35,39	11,63	4,09
4	5,91	22,50	13,23	25,50	2,37	16,45	5,09	49,64	9,21
5	2,22	25,77	9,78	23,76	1,37	62,09	6,06	19,17	24,89
6	2,28	24,43	20,24	13,04	3,44	24,36	10,39	13,00	5,52
7	1,65	21,93	16,55	15,11	5,51	22,92	7,18	18,24	13,47
8	1,98	22,63	25,86	13,38	9,50	10,33	2,90	64,26	13,00
9	4,36	13,10	18,96	3,19	10,35	24,54	5,61	12,18	3,64
10	3,81	21,62	20,45	22,44	9,01	10,38	21,32	14,35	6,54
11	0,65	11,80	24,34	18,98	5,57	23,60	2,00	14,04	6,74

B.6. táblázat. *Futási idő variációs koefficiense Gurobi megoldóval a túlkínálatot mutató hálózatokra (százalék)*

	100-50	100-100	100-200	300-50	300-100	300-200	500-50	500-100	500-200
ref	17,10	20,36	22,37	16,83	4,78	41,50	20,69	34,38	31,64
1	26,17	25,88	9,06	16,56	11,01	30,07	9,71	18,32	37,32
2	27,29	15,03	6,16	1,56	n.a.	42,28	1,98	15,26	36,69
3	25,43	20,85	12,53	11,36	n.a.	n.a.	5,83	20,39	n.a.
4	11,80	20,69	17,55	6,55	4,23	31,12	7,09	21,16	26,06
5	26,36	12,10	13,78	7,30	15,90	23,56	14,22	23,00	31,46
6	37,25	11,84	16,64	16,64	10,66	19,74	13,09	27,30	31,68
7	26,02	18,37	17,01	9,26	9,25	21,96	11,15	27,84	23,61
8	42,45	11,62	25,16	18,88	21,21	20,88	13,95	22,66	27,23
9	29,26	25,73	28,06	7,49	8,20	17,65	8,08	21,25	29,25
10	20,47	24,51	19,52	15,42	18,49	17,27	13,57	14,44	35,02
11	11,00	33,34	32,51	18,38	14,41	30,56	23,31	16,11	29,43

B.7. táblázat. *Futási idő átlaga Mosek megoldóval a túlkeresletet mutató hálózatokra (másodperc)*

	100-50	100-100	100-200	300-50	300-100	300-200	500-50	500-100	500-200
ref	24,37	31,81	100,39	786,25	2 322,84	2 832,95	3 142,39	8 264,79	15 530,16
1	53,75	114,94	538,45	1 994,33	5 640,38	8 206,87	7 911,81	n.a.	43 842,91
2	52,79	196,94	393,44	2 004,40	4 371,85	8 574,19	7 814,97	n.a.	44 410,97
3	48,31	157,22	431,42	2 032,38	3 753,17	7 766,67	7 069,46	n.a.	n.a.
4	12,85	92,43	79,79	852,03	263,57	3 769,14	2 492,32	6 322,74	12 820,44
5	17,44	63,43	116,95	784,68	162,88	3 341,78	2 823,70	7 433,14	12 754,80
6	19,24	53,13	124,65	762,61	162,81	3 428,15	2 820,49	7 370,58	12 542,12
7	14,04	66,16	38,96	764,09	262,85	3 600,78	2 460,55	6 625,07	12 327,43
8	18,92	46,49	132,29	703,70	1 522,22	3 123,00	2 942,77	7 480,00	12 915,44
9	20,01	49,45	115,92	706,56	1 488,22	3 179,90	2 970,85	7 754,06	12 472,33
10	19,77	40,55	116,00	837,10	1 487,42	2 918,84	2 923,90	7 171,64	13 249,78
11	17,87	34,08	96,26	853,30	1 484,55	3 002,99	3 007,52	6 993,45	12 526,17

B.8. táblázat. *Futási idő átlaga Mosek megoldóval az egyenletes keresletet mutató hálózatokra (másodperc)*

	100-50	100-100	100-200	300-50	300-100	300-200	500-50	500-100	500-200
ref	4,43	31,50	26,48	104,31	277,44	507,66	849,05	1 348,94	3 086,97
1	11,99	66,68	80,08	330,60	1 318,99	2 822,21	1 820,59	3 997,72	8 825,78
2	14,00	55,81	65,27	364,01	1 589,55	1 423,26	1 768,54	3 850,90	8 493,37
3	16,88	33,09	54,89	282,29	1 003,68	1 204,61	1 492,53	3 557,27	8 115,97
4	3,99	25,59	15,79	51,63	120,08	187,58	158,41	475,96	2 112,55
5	4,71	22,75	16,51	44,06	78,10	173,78	438,64	1 491,71	1 116,60
6	4,46	22,98	22,46	50,81	80,89	155,47	526,75	1 339,68	1 035,96
7	4,10	24,42	20,09	74,23	70,36	153,93	164,91	383,88	1 959,77
8	5,59	9,60	20,23	156,57	276,25	490,52	622,67	1 294,38	3 048,18
9	5,64	18,10	17,07	116,46	337,68	454,45	811,89	1 376,10	2 860,50
10	5,82	19,40	17,03	111,70	428,71	475,50	890,35	1 301,00	2 786,75
11	6,43	12,59	16,09	104,67	370,40	467,71	672,61	1 286,03	3 057,83

B.9. táblázat. *Futási idő átlaga Mosek megoldóval a túlkínálatot mutató hálózatokra (másodperc)*

	100-50	100-100	100-200	300-50	300-100	300-200	500-50	500-100	500-200
ref	89,1	151,8	444,9	4 507,1	6 306,9	75 623,6	21 554,9	46 478,1	79 790,2
1	233,4	385,5	1 486,7	11 195,8	18 563,5	231 112,8	53 257,1	130 143,8	238 281,3
2	234,8	414,2	1 393,6	12 414,5	20 058,6	240 579,4	53 556,8	127 727,0	234 209,0
3	238,8	384,3	1 239,8	11 223,5	17 062,9	200 349,9	48 026,5	109 517,9	206 670,8
4	125,2	203,7	743,9	5 685,8	8 438,6	80 323,0	23 689,9	20 777,5	50 918,9
5	103,5	223,7	719,3	4 924,7	5 386,5	65 981,9	24 691,3	12 603,9	49 730,9
6	71,9	231,6	610,2	5 289,2	5 521,3	65 418,7	24 371,1	12 236,0	49 122,6
7	79,3	234,5	801,0	5 645,3	8 211,3	74 196,7	24 576,9	23 006,7	54 052,5
8	71,5	189,5	520,4	4 347,7	6 983,9	65 769,9	21 785,4	39 518,0	71 877,0
9	78,3	152,0	501,1	4 381,3	6 500,5	66 224,4	21 597,1	37 840,0	70 950,8
10	83,3	137,5	527,8	4 974,3	6 470,0	66 024,2	20 479,2	37 763,4	67 863,1
11	89,1	134,4	461,9	5 126,5	6 951,2	63 601,7	20 635,8	40 666,3	66 916,0

B.10. táblázat. *Futási idő variációs koefficiense MOSEK megoldóval a túlkeresletet mutató hálózatokra (százalék)*

	100-50	100-100	100-200	300-50	300-100	300-200	500-50	500-100	500-200
ref	65,26	23,93	49,54	31,92	73,51	22,95	10,98	29,21	24,87
1	41,64	65,23	107,28	28,85	66,07	23,65	7,97	n.a.	31,11
2	39,94	111,42	83,50	29,34	35,13	24,31	7,20	n.a.	32,89
3	44,45	103,21	100,73	35,36	24,82	30,37	7,70	n.a.	n.a.
4	33,08	115,95	135,16	27,09	22,69	23,61	12,82	17,03	18,87
5	40,93	112,76	126,79	26,06	24,67	27,89	9,64	9,47	20,25
6	56,65	103,42	129,77	21,47	21,36	29,63	9,27	16,56	20,53
7	47,87	93,97	97,79	19,17	19,97	30,95	10,72	17,79	14,93
8	39,89	73,25	83,44	17,28	24,93	31,32	7,85	14,84	8,09
9	45,95	81,40	68,19	16,28	22,77	44,87	8,47	12,12	8,64
10	42,73	62,07	69,34	41,11	22,03	36,62	6,68	6,41	10,67
11	28,01	39,27	48,05	44,45	21,88	36,85	10,25	3,92	8,53

B.11. táblázat. *Futási idő variációs koefficiense MOSEK megoldóval az egyenletes keresletet mutató hálózatokra (százalék)*

	100-50	100-100	100-200	300-50	300-100	300-200	500-50	500-100	500-200
ref	7,33	141,23	38,84	13,61	44,22	22,92	30,54	11,74	11,63
1	6,40	119,65	59,99	20,47	88,08	85,49	4,73	6,23	4,04
2	24,65	100,17	62,14	6,99	102,51	1,67	2,62	11,90	4,30
3	44,02	56,41	43,35	18,98	71,41	7,01	3,40	19,66	5,79
4	36,18	98,00	49,05	11,96	87,76	6,42	8,46	29,16	18,54
5	36,86	61,39	26,39	16,12	50,71	10,28	18,36	17,86	24,11
6	30,77	52,55	71,77	16,70	65,96	13,72	42,35	8,51	8,91
7	23,32	90,66	85,02	58,06	49,42	7,78	3,81	11,45	7,64
8	13,79	56,88	53,72	31,80	25,18	10,77	9,58	8,64	7,92
9	12,31	69,97	32,70	2,56	58,58	13,61	40,86	10,11	9,09
10	11,36	71,84	23,41	15,93	89,39	5,57	32,99	11,13	3,53
11	22,53	43,54	41,41	20,89	56,87	8,02	8,90	9,63	4,86

B.12. táblázat. *Futási idő variációs koefficiense MOSEK megoldóval a túlkínálatot mutató hálózatokra (százalék)*

	100-50	100-100	100-200	300-50	300-100	300-200	500-50	500-100	500-200
ref	29,25	7,57	24,14	2,59	10,76	27,78	14,36	15,56	17,52
1	22,50	16,73	8,48	5,56	8,55	20,15	12,72	7,10	19,83
2	13,62	23,59	17,34	6,83	14,59	21,13	10,53	5,21	31,88
3	27,39	41,50	25,06	6,65	14,30	31,39	12,56	5,85	28,28
4	9,77	24,58	21,49	5,30	5,71	28,74	10,37	9,90	22,38
5	14,64	18,95	26,74	13,33	13,24	20,51	8,89	4,65	20,40
6	22,50	20,60	10,38	12,79	6,53	21,00	9,60	4,35	19,19
7	14,93	16,78	16,81	16,00	8,24	20,33	13,32	12,91	26,02
8	17,59	22,36	33,58	11,17	15,05	25,69	19,84	15,08	29,04
9	18,60	12,54	20,73	9,91	5,63	30,06	7,03	10,64	32,64
10	40,88	16,28	28,19	8,60	5,59	27,76	12,40	18,82	31,31
11	32,82	7,70	20,34	17,81	8,25	25,50	14,45	17,84	34,86

SUMMARY

What are the effects of modeling decisions to the efficiency of solving nonlinear optimization problems – that is the main research question of my PhD thesis. This chapter concludes the contributions of the dissertation and frames directions for future research.

Chapter 2 presents *basic concepts of optimization* related to the topic of the thesis. Frequent classifications and solution methods of optimization problems are enumerated, together with an introduction to algebraic modeling languages and their role in solving large practical problems.

Chapter 3 discusses the possibility of implementing nonlinear coordinate transformations from Csendes and Rapcsák [20, 71] in an *automatic symbolic simplification algorithm* realized in a modern computer algebra system. The method for simplifying unconstrained nonlinear optimization problems was implemented in Maple and Mathematica environments. It was based on the referred theoretical results. The program can automatically recognize helpful transformations of the mathematical model and detect implicit redundancy in the objective function [4, 3]. In some cases the dimension reduction or the recognition of redundant model parameters achieved in this way are invaluable for the experts who analyze optimization models [38].

Subsections 3.4.3–3.4.5 discuss the main problems arisen in connection with the first implementation in Maple. In Subsection 3.4.8 Maple and Mathematica based implementations were compared from the produced substitutions point of view. In this part, I could rely especially on my custom made problems designed especially to test the capabilities of symbolic simplification algorithms. It turned out that Mathematica is more favorable as the environment of our simplification method due to its highly customizable substitution routine and better interval arithmetic capabilities.

It would be the best to have such an automatic simplification algorithm implemented in preprocessors of algebraic modeling languages, similar to the LP presolving technique of AMPL presented in Subsection 5.4.3. and 6.2.5. However, further theoretical results would be necessary to apply the referred nonlinear coordinate transformations in such an environment, as it will be discussed in more details hereinafter.

An extensive computational test has been completed on standard global optimization test problems and on other often used global optimization test functions together with my custom made test problems to analyze performance of the realized simplification program. Altogether, 45 well-known global optimization test problems were examined, and our Mathematica program offered equivalent transformations for 8 cases. In other words, our method suggested some simplifications for 18% of this extended standard global optimization test set. It is impressive if we consider that no other method is known for suggesting similar automatic reformulations for those problems.

All substitutions produced in Subsections 3.4.8 and 3.4.9 by the Mathematica version of the automatic simplification method are correct. However, two important question occur, namely:

1. Are the produced substitutions useful or negligible?
2. Are those substitutions worth the extra preparation?

My study for answering those questions are presented at the end of Chapter 3 and in Chapter 4.

In connection with Question 1, *Chapter 4* examines the possibility and ability of using equivalent transformations for nonlinear optimization problems as an automatic presolving phase of a numerical global optimization method. More specifically, tests with a numerical solver, namely GLOBAL, were performed to check the usefulness of the produced transformations from the running times and function evaluation numbers point of view. The computational results obtained for standard global optimization test problems and for other artificially constructed instances in Subsection 3.4.8 and 3.4.9 show that the produced substitutions can improve the performance of this multi-start solver. The transformations impact running times and function evaluation numbers in a favorable manner. The average relative improvement in the running times and also in the function evaluation numbers of the whole test set is 32% due to the transformations.

In Section 3.5 I present my new theoretical results to generalize nonlinear coordinate transformations of Csendes and Rapcsák in two directions. At first, I give sufficient conditions for more complicated equivalent transformations by describing possible parallel substitutions. Secondly, I extend the scope of the method for constrained nonlinear optimization problems. However, integrating my results into the automatic simplification algorithm would

require more considerations. Further investigations would be necessary to build an efficient branch and bound strategy into the algorithm to realize good running times for the described parallel substitutions.

In connection with Question 2, Subsection 3.4.9 discusses running times of the simplification method measured in Mathematica. While all the transformations on my custom made problems were finished in less than 0.2 seconds, the running time for the standard test cases vary more. 24 of 45 test cases ran in one second, further 10 analysis required less than one minute, but 7 test cases would require more than half an hour to finish. As it is presented in Chapter 4, solving these problems with GLOBAL requires less than one second on average.

In future research it would be interesting to determine those problem class – solver combinations, for which the cost of transformation together with the solution of the new form requires less running time than solving the original problem form. Our symbolic preprocessor would be reasonable to use most likely for finding *reliable* solutions. That means the usage of reliable (interval or symbolic) computation techniques. In those cases the extra cost of preprocessing could return in two ways: the transformations could help a computationally intensive solver (e.g. interval branch & bound [81]) or could produce problem forms applicable for a special solver method (e.g. for quantor elimination [83]).

As a natural extension of the present application, symbolic reformulations are promising for speeding up interval methods of global optimization. The overestimation sizes for interval arithmetic [1] based inclusion functions were investigated in optimization models [81]. Symbolic transformations seem to be appropriate for a proper reformulation. Obvious improvement possibilities in this field are the use of the square function instead of the usual multiplication (where it is suitable), the transformation along the subdistributivity law, and finding SUE forms. In fact, such transformations are usually performed by the default expression simplification mechanism [79] of an ordinary computer algebra system. Pending work of the author seems to ensure that the automatic simplification method described in this dissertation could be applicable to increase efficiency of interval solvers.

Chapter 5 presents my solution for the max-min fair bandwidth allocation problem of BitTorrent communities. I study the problem at inter-swarm level, which is one of the most complex levels due to the behavior of users, as the model handles that most of the users are participating in uploading and downloading multiple contents at the same time.

It was shown by Capotă et al. [13] that using the standard BitTorrent protocol's bandwidth allocation, the average performance of a BitTorrent community is suboptimal in terms of max-min fairness. This fairness measure corresponds to the case of video-streaming service – an emerging application of peer-to-peer networks.

Our algorithm [5] is a refinement of the work of Capotă et al. It is an adapted version of the general Max-Min Programming Algorithm summarized by Radunović and Le Boudec [69], as it computes the max-min fair weights of the download edges with an iterative manner, by fixing the coordinates with the smallest unfixed weight in every iteration. My main contributions are to replace the complicated sieve method of Capotă et al. by an exact mathematical programming formulation, and to prove the correctness of our new algorithm. The proposed algorithmic approach is detailed in Section 5.4, including mathematical analysis, reformulation and proof of the correctness.

The correct algorithm was implemented in AMPL with the application of my theoretical results and some further reformulation techniques. Section 5.5 contains numerical experiments, including comparison with the previously proposed method for the same problem. Our observations show that our formulation helps the Gurobi solver to achieve shorter running times, and *MaxMin-r* can be faster than the earlier proposed *MM* algorithm on larger problem instances. Moreover, the results from the first iterations of *MaxMin-r* could be used as a very good approximation for the max-min fair allocation. This approximation, which is a feasible solution, can be achieved in fraction of the time of the adequate precession of *MM*. For example, the first iteration of *MaxMin-r* took 46 seconds for the test case containing 23 670 nodes and 7 326 edges, compared to the more than eight-hour running time for the first 910 iterations of *MM* with adequate solution quality.

Due to the unavoidable involvement of solving several large scale MILPs to obtain exact solution to problems including millions of nodes and edges, it is desired to develop very quick heuristics in the future. The work of the author in this direction is also pending.

Furthermore, as the application field of the max-min fairness problem I investigated lies in peer-to-peer systems, a distributed version of the exact algorithm or even distributed heuristics would be preferred. I hope that my results provide useful insights towards these goals.

Chapter 6 analyses the effects of using several modeling techniques in the optimization problem discussed in Section 5.4, which includes the solution of large linear and nonlinear mixed-integer programs. An extensive

computational study was transacted including every combination of twelve model variants, twenty-seven test case and two professional solvers [22].

The performed experiments hand on several interesting results. First of all, there was no model variant with quickest solution for every case. It was pointed out, that the tested modern solvers are capable to solve efficiently the bilinear form of a problem. Gurobi and MOSEK produced better running times for the smaller bilinear problems, in comparison with the equivalent linear forms with higher dimensions, which were produced by applying McCormick envelopes. The presolving technique discussed in Subsections 5.4.3 and 6.2.5 was prominent, as it induced 10% decrease of running times for the test cases on average.

In future research the inversion of the performed experiments would be interesting also. Inspection of the structure of input graphs could hopefully decide which algorithm variant is the best for a given graph from the solving performance point of view.