Alkalmazott Matematikai Lapok **vol** (201N), p–pp.

EGY NEMLINEÁRIS VEGYES-EGÉSZÉRTÉKŰ OPTIMALIZÁLÁSI FELADAT KÜLÖNFÉLE MODELLJEINEK KOMPARATÍV ELEMZÉSE

DOBJÁNNÉ ANTAL ELVIRA ÉS VINKÓ TAMÁS

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). Jelen tanulmányt egy hálózati folyam probléma matematikai modelljének felírása kapcsán felmerülő kérdések ihlették.

A cikkben azt vizsgáljuk, hogy egy konkrét, nagyméretű lineáris és nemlineáris vegyes-egészértékű programokat is magában foglaló optimalizálási feladat megoldásának sebességét mennyiben befolyásolja különféle modellezési technikák alkalmazása. Egy elosztott tartalommegosztó hálózat max-min méltányos erőforráselosztásának kiszámítását célzó modell tizenkét változatát hasonlítjuk össze egy kiterjedt numerikus tesztelés során, két professzionális megoldó és huszonhét nagyméretű tesztfeladat felhasználásával.

Reményeink szerint a közölt eredmények túlmutatnak a konkrét problémán, és árnyaltabb képet adnak más hasonló feladatok megértéséhez is.

1. Bevezetés

Élőkép interneten történő közvetítésére számos megoldás létezik. Amennyiben a skálázhatóság kérdése fölmerül, gyakran az elosztott módon működő módszerek adnak minőségi választ. Egy ilyen lehetséges módszer a BitTorrent protokollon alapszik [4]. A BitTorrent eredetileg egy tartalommegosztó rendszer, amely elsősorban nagyméretű fájlok hatékony hozzáférését segíti elő [5, 8]. Kiderült azonban, hogy a protokoll részleteinek megfelelő módosításával lehetőségünk van élő közvetítésre (live streaming), illetve video-on-demand szolgáltatások támogatására is [7, 6, 15, 13, 12].

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 letöltés sebességet kell garantálnunk. Jelen cikkben ez utóbbi feladatra fókuszálunk. A feladat, bizonyos feltételek kikötése mellett, megfogalmazható egy speciális szerkezetű gráfon értelmezett nemlineáris vegyes-egészértékű optimalizálási feladatként. Ennek részletes leírását az [1] cikkben találhatjuk meg. Mivel a vizsgált feladat megoldására javasolt iterációs módszer számos részletet és modellezési meggondolást tartalmaz, ezért természetesen adódik a kérdés: vajon milyen tényezők befolyásolják a megoldás sebességét? Jelen cikkben összegyűjtöttük a lehetséges opciókat, amelyeket részletes numerikus vizsgálatoknak vetettünk alá. Bár a feladat specifikus, meggyőződésünk, hogy az elvégzett numerikus tesztek által kapott eredmények általánosabb érvényű empirikus képet adnak a hasonló típusú problémák számítógépes megoldási lehetőségeire.

A következőkben először megadjuk a legfontosabb fogalmakat, valamint a használt hálózati modellt (2. szakasz). Ezután röviden ismertetjük az [1] cikkben javasolt iterációs módszert, továbbá a lehetséges algoritmus változatok egy bőséges listáját (3. szakasz). A felhasznált tesztesetek leírását (4. szakasz) a numerikus eredmények diszkussziója követi (5. szakasz).

2. Max-min méltányos erőforrás-elosztás problémája

Ebben a fejezetben bevezetjük a legszükségesebb definíciókat, amelyek egyrészt megadják a vizsgált feladat felhasználási területét, másrészt leírják a vizsgált optimalizálási algoritmus bemeneteként szolgáló folyam hálózatot.

Mint azt a bevezetőben említettük, a vizsgált feladat egy elosztott tartalommegosztó rendszerben előforduló erőforrás-elosztás problémaköréhez tartozik. Ez a rendszer a BitTorrent. Jelen cikk szempontjából nézve a rendszernek három fő komponense van: letöltők (*leecher*ek), megosztók (*seeder*ek) és a megosztott fájlok (ezeket gyakran *torrent*eknek is nevezzük, amely 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 feltöltők halmaza azon felhasználók, akik rendelkeznek a fájl összes darabjával. Fontos szempont, hogy a letöltők is tudnak egymás között darabokat cserélni, így a letöltés közben egyben feltöltőként is szolgálják a rendszer működését. Pontosan ez az az ötlet, amitől a BitTorrent rendkívüli módon jól skálázható [5]. A továbbiakban *BitTorrent közösség* alatt felhasználók (leecherek és seederek), valamint fájlok egy rögzített halmazát értjük.

Capota és szerzőtársai [3] nyomán egy BitTorrent közösség aktuális állapotát – vagyis, hogy egy adott időpillanatban ki kinek tölthet fel, milyen adatátviteli korlátok érvényesek, stb. – egy speciális hármas gráf reprezentációval írhatjuk le. Ezt az irányított, súlyozott hármas 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 kapacitással és δ_i letöltési kapacitással, 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;

$L = \{ l_i^t \mid i \in I,\, t \in T\}$: a *leeching csúcsok* halmaza, ahol az l_i^t (ú.n. *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 :

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

$f : E \to \mathbb{R}^+$**:** 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:

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

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

Célunk minden $(l_j^t, d_j) \in E_D$ élre a *max-min méltányos erőforrás-elosztás* 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-elosztás egy rokon feladata [14]. A formális definíció megtalálható pl. [3] és [14] cikkekben.

3. A probléma megoldása; modell-átírási lehetőségek

A max-min méltányos erőforrás-elosztás kiszámítását célzó algoritmus kiindulási alakját *az 1. algoritmus* tartalmazza. A korábbi cikkünkben [1] bemutatott megoldás tulajdonképpen Radunović és Le Boudec általános max-min programozási algoritmusának [14] 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 legnagyobb folyam értékét. A halmazokat nagy betűkkel, az optimalizálási feladatok döntési változóit kis betűkkel, míg a paramétereket (a rögzített értékkel bíró változókkal egyetemben) görög betűkkel jelöljük.

Az 1. algoritmusnak természetesen sokféle variációja képzelhető el, a megvalósítás során érdemes lehet különféle modellezési „trükköket” alkalmazni. Korábbi cikkünk munkálatai közben mi magunk is többféle változtatást eszközöltünk a hatékonyság növelése érdekében, azonban az egyes változtatások hasznosságának igazolására akkor nem kerülhetett sor. A következőkben számbavesszük az általunk javasolt módosításokat, az 5. szakaszban pedig elemezzük az ezen módosítások kombinációiból előálló modell-változatok hatékonyságát a futási idő és az elért optimum érték tekintetében.

3.1. Egy redundáns korlát hozzáadása

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

$$f_k \geq \phi. \tag{3}$$

Antal és Vinkó [1] 3. lemmája alapján a (3) korlát redundáns, és csak az első iterációban aktív, mivel az (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.

3.2. Bilineáris vegyes-egészértékű feladat McCormick-átírása

Az 1. algoritmus 6. lépésében szereplő $mMM_k^{(2)}$ jelzésű bilineáris programozási feladat helyettesíthető a McCormick-átírásával [11]. Ez az átírás egy ekvivalens vegyes-egészértékű lineáris programozási (MILP) feladatot eredményez, amelyben a bilineáris kifejezések helyére új, folytonos változókat vezetünk be:

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

1. algoritmus. *mMaxMin*

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

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

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

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

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

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:

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

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

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

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

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

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

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

$$\begin{aligned} &\max \sum_{(l_j^t, d_j) \in E_k} x_j^t, \\ \text{f.h. } & \sum_{(l_j^t, d_j) \in E_k} f(l_j^t, d_j)\, x_j^t + \phi_k \cdot \sum_{(l_j^t, d_j) \in E_k} (1 - x_j^t) + \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t = (1-\epsilon) \cdot \sigma_k, \\ & f(l_j^t, d_j) \geq \phi_k \qquad\qquad \forall (l_j^t, d_j) \in E_k, \\ & f(l_j^t, d_j) > \phi_k\, x_j^t \qquad\qquad \forall (l_j^t, d_j) \in E_k, \end{aligned} \tag{2}$$

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:

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

$$\ell_j^t := \phi_k,\ \forall (l_j^t, d_j) \in E_k \text{ ahol } x_j^t = 0,$$
$$F := F \cup \Phi_k, \quad E_{k+1} := E_k \setminus \Phi_k.$$

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

ahol $\forall(l_j^t, d_j) \in E_k : p_j^t \in \mathbb{R}^+$, továbbá $x_j^t \in \{0, 1\}$. Az $mMM_k^{(2)}$ feladat McCormick-átírása tehát a következő:

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

$$\text{f.h.} \sum_{(l_j^t, d_j) \in E_k} p_j^t + \phi_k \sum_{(l_j^t, d_j) \in E_k} (1 - x_j^t) + \sum_{(l_j^t, d_j) \in (E_D \setminus E_k)} \ell_j^t \geq (1 - \epsilon) \cdot \sigma, \tag{4}$$

$$\begin{aligned} f(l_j^t, d_j) &\geq \phi_k && \forall(l_j^t, d_j) \in E_k, \\ f(l_j^t, d_j) &> \phi_k \, x_j^t && \forall(l_j^t, d_j) \in E_k, \\ \min\left(\delta_j \, x_j^t, \, f(l_j^t, d_j)\right) &\geq p_j^t && \forall(l_j^t, d_j) \in E_k, \end{aligned} \tag{5}$$

$$\max\left(0, \, f(l_j^t, d_j) - \delta_j \, (1 - x_j^t)\right) \leq p_j^t \qquad \forall(l_j^t, d_j) \in E_k, \tag{6}$$

vagyis az eredeti (2) jelzésű korlátot lecseréljük (4)-re, és kiegészítjük a feladatot (5) és (6) korlátokkal.

Habár a probléma dimenziója nő az átírás folytán, egy egzakt korlátozás-és-szétválasztás típusú megoldó alkalmazható az előálló MILP globális optimumának megtalálására [2].

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

3.3. Kezdőérték-adás a bináris változókra

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

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

és

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

Itt $f_k^{(y)}(l_j^t, d_j)$ a kapcsolódó $f(l_j^t, d_j)$ folyam értékeket jelöli az $mMM_k^{(y)}$ egy fízibilis megoldásában. Az $mMM_k^{(2)}$ feladat megoldása során segítséget jelenthet a bináris változókra vonatkozó kezdőértékek explicit megadása.

3.4. Kezdőérték-adás a McCormick-átírás mesterséges változóira

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

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

1. ábra. *Vizsgált modellek áttekintése*

Modell	3.1	3.2	3.3	3.4	3.5
ref	•	•	•	•	•
1	•	•	•	•	
2	•	•	•		
3	•	•			
4	•				
5					
6			•		
7	•		•		
8	•		•		•
9			•		•
10	•				•
11					•

3.5. Előmegoldás, avagy folyamok rögzítése a folyam-megmaradásra hivatkozva

Az 1. algoritmus 5. lépése egy, az AMPL előmegoldójában is megvalósított standard LP előmegoldó technika [9, 10]. 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. A fenti kifejezésben $deg_k^-(d_j)$ a d_j csúcs azon bemenő éleinek számát jelöli, amelyeken a k. iterációban még nincs rögzített folyam.

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. 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.

3.6. Modell-változatok

A fent bemutatott módosítási javaslatok kombinációiból összesen tizenkettő modell-változatot készítettünk, hogy a módosítások hasznosságát külön-külön, ill. együttesen elemezni tudjuk (eredmények az 5. szakaszban). Az 1. táblázat összefoglalja, hogy melyik modell melyik korábbi alszakasz(ok)ban bemutatott módosítást tartalmazza. Referenciaként (**ref**) a korábban publikált matematikai modell [1] szolgált, amely minden módosítást tartalmazott, a többi modellt számokkal jelöltük.

4. Tesztesetek

Ahhoz, hogy a különböző algoritmus variánsokkal numerikus hatékonysági vizsgálatokat készíthessünk, számos mesterségesen generált hálózatot készítettünk. Ebben a szakaszban ezen teszthálózatok előállításának módszereit közöljük.

Alapvetően három, lényegileg különböző típusú hálózatot gyártottunk, amelyek a következők:

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%-ában 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.

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 (amely 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 az 1. táblázatba foglaltuk össze. Vegyük észre, hogy minden gráfban a pontok száma jelentősen több, mint a felhasználók és torrentek száma. Ennek a magyarázata, hogy bemenetként nem páros gráfot kell megadnunk, hanem a 2. fejezetben említett hármas gráfot, amelyben elsősorban a köztes (L halmazba tartozó) csúcsok száma lesz magas.

A valós BitTorrent közösségekből származtatható gráfok az itt vizsgáltaknál jóval nagyobb méretűek, ugyanakkor nem feltétlenül reprezentálnak olyan eseteket, amelyeket itt vizsgáltunk. A kisebb méret továbbá lehetővé teszi, hogy kivárható időn belül legyen megoldásunk.

5. Eredmények

5.1. 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 végeztünk. A 4. szakaszban bemutatott *ti-*

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

	egyenletes		**túlkereslet**		**túlkínálat**	
user-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

zenkettő modell-változat, a 4. szakasz *huszonhét tesztesete* és *kettő professzionális megoldó* minden lehetséges kombinációjára *három futtatás*t végeztünk.

Az algoritmus-variánsok AMPL nyelven voltak kódolva. A Gurobi és a MOSEK solvert vettük górcső alá, mivel ez a két általános nemlineáris megoldó állt rendelkezésünkre, amelyek a szóban forgó modellek optimalizálási feladatait kivárható időn belül meg tudták oldani. Mindkét megoldót az alapértelmezett paraméterekkel működtettük. A tesztek egy 24 magos Intel Xeon 2,27 GHz-es számítógépen futottak, ahol 24 GB memória állt rendelkezésre.

5.2. Gurobi

A 2–4. táblázatok a Gurobi megoldóval elért átlagos futási időket mutatják a különböző feladat-osztályokra. Megjegyezzük, 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 vetítettünk egy, az adatokból készült hőtérképet is, a következő színskála alapján:

Továbbá minden oszlopban aláhúzással jelöltük 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-elosztást, 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 3.5. 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 3. 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.

2. ábra. *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

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 3.2. alszakaszban bemutatott McCormick-átírás* alkalmazása az előmegoldás nélkül jelentősen, de az előmegoldással együtt is kismértékben bonyolítja a feladatot.

A 3.4. alszakaszban felvázolt kezdőérték-adás, az 1. és 2. variáns eredményeire alapozva, az esetek 84%-ában javított a futási időn, átlagosan 6%-kal.

A 3.3. 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 3.3. alszakasz kezdőérték-adását implementáló modellek átlagosan mintegy 3%-kal voltak gyorsabbak a kezdőérték-adást mellőző, minden egyéb tekintetben megegyező variánsoknál.

A 3.1. alszakaszban bemutatott redundáns korlát hozzáadása a 4. és 5. variáns összehasonlításában 7 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 3.1. alszakaszban tárgyalt módosítás. Érdemes megfigyelni, hogy

3. ábra. *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

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 az 5–7. 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.

5.3. MOSEK

A 8-10. 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 keresletet 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-elosztását 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 végeredményként egy valós számokból álló vektort ad meg, amelynek hossza megegyezik az E_D letöltő élek halmazának méretével. Mivel a használt programok lebe-

4. ábra. *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	44,62	99,71	394,93	1 767,19	3 602,11	44 689,13	10 101,58	19 961,00	142 826,03
1	164,46	437,85	1 816,25	11 773,90	23 278,35	368 166,92	66 839,67	166 285,16	393 350,03
2	203,01	410,63	1 885,52	11 005,87	n.a.	369 862,33	67 409,98	171 311,61	400 560,57
3	204,05	417,74	1 743,20	11 535,95	n.a.	n.a.	61 739,13	169 778,28	n.a.
4	133,54	281,15	1 359,30	6 527,94	19 146,21	110 138,40	41 315,39	82 407,80	124 545,26
5	133,68	332,27	1 596,19	7 481,82	20 126,71	120 270,78	43 926,90	86 053,85	123 590,57
6	118,68	312,86	1 350,50	6 805,64	18 939,62	93 867,99	39 201,74	76 569,01	104 913,93
7	87,75	321,38	1 385,52	6 197,48	17 703,81	87 309,58	38 955,36	72 105,72	94 794,07
8	52,84	85,87	406,71	1 741,15	3 756,55	30 590,76	7 767,45	17 273,40	28 178,77
9	59,06	125,65	516,26	1 842,90	4 019,89	34 418,82	8 535,70	19 165,41	34 922,32
10	57,75	109,89	375,08	1 981,35	4 090,36	29 545,93	8 203,24	16 128,15	27 372,15
11	60,25	124,50	454,04	2 064,10	4 357,51	37 142,15	9 126,37	18 057,12	35 088,72

gőpontos műveleteket végeznek, ezért kerekítési hiba előfordulhat, amely befolyásolhatja a végeredményt. Jelen esetben pontosan ezt tapasztaltuk, tehát a két megoldó a kerekítési hibákra különbözőképpen érzékeny.

A Gurobihoz hasonlóan itt is az 1-3. algoritmus-variánsok voltak a leglassabbak, mindhárom produkált szegmentálási hibát is a legnagyobb feladatokon.

A 3.4. alszakasz kezdőérték-adása, *a 3.3. alszakasz kezdőérték-adása*, és *a 3.1. 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. algoritmus-variáns, ráadásul ebben az összehasonlításban (tehát a McCormick-átírással kombinálva) minden esetben több, mint 50%-os gyorsulást eredményezett *a 3.5. 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 11–13. táblázat tartalmazza. A MOSEK által igényelt futási idők átlagos variációs koefficiense a Gurobitól nagyobb, átlagosan 30% volt a teljes adathalmazra.

5.4. Konklúzió

A 14. 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. az 5.2. alfejezet eleje) ne torzítsák az eredményt.

A cikkben 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

5. ábra. *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

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.

6. ábra. *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

6. Összefoglalás

Jelen cikkben egy komplex, nagyméretű lineáris és vegyes-egészértékű nemlineáris optimalizálási feladatokat is felvonultató probléma kapcsán elemeztük a matematikai modellezés során felmerülő átírási lehetőségek hatásait. Kiterjedt numerikus tesztelést végeztünk tizenkét modell-változat, huszonhét teszteset és kettő professzionális megoldó minden lehetséges kombinációjával.

Az elvégzett kísérletek számos érdekes eredménnyel szolgáltak. Egyrészt megállapíthatjuk, hogy nem találtunk olyan modell-változatot, amely minden esetben a leggyorsabban oldotta meg a feladatot. Láttuk továbbá, hogy a tesztelt korszerű megoldók számára nem jelentett problémát a bilineáris felírás hatékony megoldása. Végül ismét kiemelendő az előmegoldó (presolve) technikák használatának jelentős előnye.

Továbblépésként tervezzük megvizsgálni a modellek hálózati folyam alakú felírásának hatásvizsgálatát, amelyre az AMPL nyelv lehetőséget ad. Ezután pedig az elvégzett kísérletek egyfajta megfordítása következhet, amelyben a bemenetként megadott gráfok szerkezetének mélyebb elemzésével kimutatjuk, hogy az egyes algoritmus variánsoknak mely gráfok a legkedvezőbbek a megoldás hatékonyságának szempontjából.

Köszönetnyilvánítás

Vinkó Tamást az MTA Bolyai ösztöndíja támogatta.

Hivatkozások

[1] Antal, E., and Vinkó, T. *Modeling max–min fair bandwidth allocation in BitTorrent communities. Computational Optimization and Applications* (2016), 1–18 o.

7. ábra. *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

[2] Bonami, P., Kilinç, M., and Linderoth, J. *Algorithms and Software for Convex Mixed Integer Nonlinear Programs*, vol. 154 of *The IMA Volumes in Mathematics and its Applications*. Springer, 2012, 1–39 o.

[3] Capotă, M., Andrade, N., Vinkó, T., Santos, F., Pouwelse, J., and Epema, D. *Inter-swarm resource allocation in BitTorrent communities.* In *Proceedings of IEEE International Conference on Peer-to-Peer Computing (P2P 2011)* (2011), 300–309 o.

[4] Cohen, B. *The BitTorrent Protocol Specification.* `http://bittorrent.org/beps/bep_0003.html`. Accessed: 19-Aug-2014.

[5] Cohen, B. *Incentives build robustness in BitTorrent.* In *Workshop on Economics of Peer-to-Peer systems* (2003), vol. 6, 68–72 o.

[6] Dana, C., Li, D., Harrison, D., and n. Chuah, C. *BASS: BitTorrent Assisted Streaming System for Video-on-Demand.* In *2005 IEEE 7th Workshop on Multimedia Signal Processing* (Oct 2005), 1–4 o.

[7] Do, T. T., Hua, K. A., and Tantaoui, M. A. *P2VoD: providing fault tolerant video-on-demand streaming in peer-to-peer environment.* In *Communications, 2004 IEEE International Conference on* (June 2004), vol. 3, 1467–1472 Vol.3 o.

[8] Fan, B., Lui, J.-S., and Chiu, D.-M. *The Design Trade-Offs of BitTorrent-Like File Sharing Protocols.* IEEE/ACM Transactions on Networking 17:2 (April 2009), 365–376 o.

[9] Fourer, R., and Gay, D. M. *Experience with a Primal Presolve Algorithm.* Kluwer Academic Publishers, Dordrecht, 1994, 135–154 o.

8. ábra. *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

[10] Gay, D. M. *Symbolic-Algebraic Computations in a Modeling Language for Mathematical Programming.* Springer-Verlag, 2001, 99–106 o.

[11] Gupte, A., Ahmed, S., Cheon, M., and Dey, S. *Solving Mixed Integer Bilinear Problems Using MILP Formulations.* SIAM J. Optim. 23:2 (2013), 721–744 o.

[12] Li, B., Wang, Z., Liu, J., and Zhu, W. *Two Decades of Internet Video Streaming: A Retrospective View.* ACM Trans. Multimedia Comput. Commun. Appl. 9:1s (Oct. 2013), 33:1–33:20 o.

[13] Paris, J. F., and Shah, P. *Peer-to-Peer Multimedia Streaming Using BitTorrent.* In *2007 IEEE International Performance, Computing, and Communications Conference* (April 2007), 340–347 o.

[14] Radunović, B., and Le Boudec, J.-Y. *A Unified Framework for Max-Min and Min-Max Fairness With Applications.* IEEE/ACM Transactions on Networking 15:5 (2007), 1073–1083 o.

[15] Vlavianos, A., Iliofotou, M., and Faloutsos, M. *BiToS: Enhancing BitTorrent for Supporting Streaming Applications.* In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications* (April 2006), 1–6 o.

DOBJÁNNÉ ANTAL ELVIRA
PALLASZ ATHÉNÉ EGYETEM Természet– és Műszaki Alaptudományi Tanszék
6000 Kecskemét, Izsáki út 10.
antal.elvira@gamf.kefo.hu

VINKÓ TAMÁS
SZEGEDI TUDOMÁNYEGYETEM Számítógépes Optimalizálás Tanszék

9. ábra. *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

6701 Szeged, P.O. Box 652
tvinko@inf.u-szeged.hu

COMPARATIVE ANALYSIS OF SEVERAL MODELS OF THE SAME MIXED-INTEGER NONLINEAR PROGRAMING PROBLEM

ELVIRA D. ANTAL AND TAMÁS VINKÓ

Our study was inspired by modeling questions emerging in connection to computing max-min fair bandwidth allocation in BitTorrent communities.

We analyze several reformulation and solution techniques, including the McMormick reformulation of a MINLP, and a standard LP presolve technique, in point of running time and reached optimum value. Our extensive numerical investigation involves twelve different models of the same problem, twenty-seven test cases, and two professional solvers.

10. ábra. *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	89,05	151,76	444,85	4 507,05	6 306,93	75 623,57	21 554,93	46 478,14	79 790,16
1	233,39	385,46	1 486,71	11 195,84	18 563,54	231 112,80	53 257,07	130 143,77	238 281,27
2	234,78	414,23	1 393,63	12 414,49	20 058,58	240 579,41	53 556,80	127 726,95	234 208,96
3	238,75	384,34	1 239,84	11 223,46	17 062,86	200 349,91	48 026,53	109 517,87	206 670,79
4	125,17	203,71	743,85	5 685,84	8 438,62	80 322,97	23 689,90	20 777,48	50 918,92
5	103,52	223,73	719,30	4 924,70	5 386,52	65 981,88	24 691,29	12 603,89	49 730,91
6	71,94	231,58	610,23	5 289,23	5 521,25	65 418,69	24 371,14	12 235,95	49 122,63
7	79,34	234,52	800,96	5 645,32	8 211,28	74 196,66	24 576,94	23 006,69	54 052,46
8	71,53	189,45	520,43	4 347,72	6 983,93	65 769,86	21 785,44	39 517,99	71 876,95
9	78,30	152,02	501,05	4 381,32	6 500,54	66 224,38	21 597,08	37 839,99	70 950,75
10	83,25	137,53	527,78	4 974,26	6 469,97	66 024,15	20 479,21	37 763,42	67 863,08
11	89,08	134,40	461,94	5 126,51	6 951,19	63 601,68	20 635,79	40 666,32	66 916,01

11. ábra. *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

12. ábra. *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

13. ábra. *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

14. ábra. *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