

OTDK-dolgozat

Forrai Dorottya

Szegedi Tudományegyetem
Természettudományi és Informatikai Kar

A vezető és követő vállalatok elhelyezési problémája kombinált vásárlás mellett

OTDK-dolgozat

Forrai Dorottya
Programtervező informatikus Bsc

Témavezető: Dr. Gazdag-Tóth Boglárka
Számítógépes Optimalizálás Tanszék

Szeged, 2022

Tartalomjegyzék

1. Előszó	3
2. Probléma leírása	5
3. A többcélú vállalatelhelyezési probléma modellezése	7
3.1. Modell felírása	8
3.2. Példa feladat megoldása	10
4. Saját eredmények	16
4.1. Közös hasznosság értékének javítása	16
4.2. Feltételek módosítása	17
5. Modellezés AMPL-ben	21
5.1. A cikkben ismerttetett modell AMPL modellje	21
5.2. A módosított modell AMPL modellje	24
5.3. A két modell összehasonlítása	24
6. Jövőbeli tervek	30
6.1. Branch and Cut algoritmus	30
6.2. Vásárlók kategorizálása	32
7. Összefoglalás	34
8. Irodalomjegyzék	37

Tartalmi összefoglaló

Mindennapi dilemmáink közé tartozik, hogy hova menjünk bevásárolni figyelembe véve szempontjainkat, amelyek vonatkozhatnak akár az üzletek távolságára vagy arra, hogy hányféle terméket tudunk egy helyen megvásárolni. Általában erre a kérdésre a választ nem olyan nehéz megtalálni, hiszen például van egy kedvenc boltunk vagy az újságok akcióiból tájékozódunk. De mi történik a háttérben? Pontosan hogyan kerültek pont oda az üzletek ahol most vannak? És hogyan lehetséges további boltok építése?

Egy nemrég definiált modell segítségével keressük erre a választ, aminek módosításával egy valósághűbb megoldást kapunk. A tárgyalt optimalizálási problémában egy vezető és egy követő vállalat szeretné újabb üzletekkel ellátni a piacot azzal a csavarral, hogy a vezető vállalat tudja, érkezik majd a piacra még egy cég, aki szintén maximalizálni szeretné a profitját. Ennek tudatában keresi a legjobb helyét, ami egy többszintű optimalizálási feladatra vezet. A vásárlók játsszák a legnagyobb szerepet a modellben, mert ők eldönthetik, hogy szívesen mennének mindkét vállalathoz egy bevásárlási körúton, azaz többcélúan vásárolnának, vagy csak az egyiket preferálják, ami egyszerű vásárlás jelent.

Egy fontos szempontot kellett megvizsgálni az eredeti modell módosításakor, ahol egy vásárló vagy megy egy üzletbe vásárolni vagy sem. Mivel a valóságban ez nem ennyire egyszerű, ezért ezt az értéket egy valószínűséggel reprezentáltuk, ami arra ad választ, hogy mekkora arányban megy a vásárló vásárolni a különböző üzletekbe. Ennek a valószínűségnek a kiszámítása egy bonyolult nemlineáris feladathoz vezet, melynek a lineáris modellbe való beillesztése volt a fő feladatunk. Ekkor a bináris változóból folytonos lett, mely esetében a feltételek több helyen megváltoztak. Az eredeti modell összetett halmazait is kiküszöböltük ezzel, ami egy átláthatóbb és könnyebben értelmezhetőbb feladathoz vezetett.

Ezt egy példán keresztül matematikailag és AMPL segítségével implementálva is bemutatjuk, illetve összehasonlítjuk a modelleket további nagyobb feladatokat is figyelembe véve.

Előszó

Mi az egyik legkényelmesebb dolog, hogyha vásárolni megyünk? Például ha mindent egy helyen tudunk megvásárolni, ezáltal csökkentve az utazási költségeinket. A probléma az, hogy ezt elég sok tényező befolyásolja, vásárlónként eltérő kedvenc boltok, az adott lakhely lehetőségei, közlekedési alternatívák vagy akár a különböző meghirdetett akciók. Amennyiben viszont a kényelmet választjuk, és viszonylag közeli helyeket a bevásárláshoz, akkor az utazási költségre szánt összeget szívesebben forgatjuk a termékek megvásárlására. Tehát vásárlói szemszögből a probléma úgy közelíthető meg, hogy a különböző preferenciáinkat kielégítve a lehető legjobban költsük el a pénzünket.

Mi történik a másik oldalon? Hogyan tud egy vállalat további üzleteket elhelyezni? Figyelembe kell venni egészen sok szempontot, mint például adott terület népességét[1], közlekedési csomópontokat, úthálózatot, az eddig kitelepített raktárainak az elhelyezkedését[2] vagy akár már meglévő vállalatok üzleteit. Az utóbbi azért is egy fontos probléma, mert hasonló termékeket áruló cégek egymás profitjaira negatív hatást gyakorolhatnak, míg kiegészítő termékeket árulva pozitív hatást érhetnek el.

Tehát a főprobléma, hogy egy vállalatnak megtaláljuk a legjobb helyet a terjeszkedéshez valamilyen feltételek teljesülése mellett, amelyek akár lehetnek a fent felsoroltak is. Szükségünk van még egy célra is, ami valaminek a maximalizálása vagy minimalizálása, vagy akár a kettő kombinációja egészen egy kétszintű optimalizálási problémáig elvezetve, ahol a külső és belső problémák akár NP-nehezék is lehetnek. Például egy iskola kiépítésénél a célunk lehet az, hogy minél kisebb távolságra legyen a város valamennyi házától, azaz a távolság minimalizálása, vagy akár egy tűzoltóság kiépítésénél feltétel lehet a megfelelő úthálózat az adott városban vagy faluban.

Dolgozatomban egy hierarchikus optimalizálási [3] probléma kerül tárgyalásra. Ez a probléma tipikusan vezető (leader) és követő (follower) vállalatok elhelyezésére irányul, mely esetében a vezető vállalat úgy szeretne új üzletet építeni valahova, hogy maximum profitot termeljen (diszkrét halmazból válogatva a lehetséges helyeket) és tudja azt is, hogy utána jön egy követő vállalat is, amely szintén maximalizálni szeretné profitját, és így választ megfelelő helyet magának[4].

Fontos még, hogy mind a vezető és a követő vállalat nélkülözhető terméket forgalmaz, amit azért is lényeges kiemelni, mert egy létszükségletű termékeket forgalmazó (például étel vagy olaj) vállalat esetében teljesen más feltételeket kell megszabni.

Például tekintsük a vezető vállalatot egy gyerekjátékokkal foglalkozó üzemnek, míg a követő vállalatot egy kertészettel foglalkozó vállalatnak. A vezető vállalat azzal az információval tisztában van, hogy a követő cég is megjelenik a piacon, méghozzá

úgy, hogy az a követő számára a legjobb hely legyen profitjának maximalizálása céljából, és így szeretne a vezető vállalat is olyan helyet találni, amivel a legtöbb profitot tudja termelni. Hogyan lehetséges ezt a legjobb helyet megtalálni? Milyen összefüggés lehet a két vállalat között? Általában a többgyerekes családok kertesházban élnek, tehát akár a két üzlet minél közelebb telepítése egymáshoz pozitív hatással lehet mindkettő profitjára? A vásárló inkább szívesebben megy egyúttal mindkét boltba, vagy csak egyikbe? Így definiálhatjuk a többcélú (kombinált) és egyszerű vásárlást, melyek között a különbség az, hogy az előbbinél mindkét cég egy-egy üzlete érintésre kerül, mielőtt visszatérnénk a kiindulási pontunkba, az utóbbinál pedig mindössze egy vállalatnak az üzletét érintjük a vásárlás során.

A kérdés az tehát, hogy hol lesznek a legjobb helyek a vezető és a követő vállalat üzletei számára, hogyha egy vásárló egy utazás során vásárolhat mindkét üzletből (többcélú), csak az egyikből (egycélú), vagy egyikből sem.

Összefoglalva, a cikk [4] feladatán alapulva megfogalmazunk egy még valóságközelibb modelt, amit AMPL-ben modellezve oldunk meg.

Probléma leírása

A modellezésben szereplő résztvevők két vállalatlánc, egy vezető (Leader, jelöljük L -lel) és egy követő (Follower, jelöljük F -fel) vállalat és legfeljebb kettőjüket vásárlás során érintő vásárlók. Mindkét vállalatlánc nélkülözhető termékeket árul, amik nem ugyanazok a termékek, ezért köztük nincs igazi verseny. A piacra először belépő vezető vállalat jelen legjobb tudása szerint szeretne elhelyezni előre meghatározott n^L számú üzletet profitját maximalizálva. A legjobb tudásába beletartozik az is, hogy tudja hogy a követő vállalatlánc is meg fog jelenni a piacon n^F számú üzlettel, aki a saját profitját maximalizálva helyezi el az üzleteit. A követő vállalat hasonlóan maximalizálni szeretné profitját, és tisztában van azzal, hogy a vezető vállalatláncnak hol épültek üzletei. Ezeken felül megjelenik a vásárló is, akinek a legfontosabb szerepe van az üzletek kiépítésében, hiszen az ő kereslete befolyásolja ezek helyét. Négy különböző vásárlásfajtát különböztetünk meg: egyszerű vásárlás a vezető vállalat üzletébe, egyszerű vásárlás a követő vállalat üzletébe, többcélú vásárlás érintve a követő és a vezető vállalat üzletét is, illetve utolsóként, ha nem történik vásárlás. Hogy ezek közül melyik valósul meg, azt a hasznossági értékek fogják eldönteni, azaz hogy éppen egy adott vásárlónak melyik éri meg a legjobban, illetve abban az esetben ha egyik sem éri meg, akkor nem fog vásárolni a vásárló. A vásárlókat vásárlásfajta alapján lehet összetettebb módon is csoportosítani [5]. Például nagyobb bevásárlást preferáló vásárló képes sokkal messzebbre is menni, hogyha ott a nagy mennyiségű vásárlás miatt pénzt spórolhat (service seeker típus). Az ilyen típusú vásárlók inkább többcélúan vásárolnak. Ugyanakkor azok a vásárlók, akik csak a legjobb árakat figyelembevéve vásárolnak, inkább választanak egyszerű vásárlási típust, mert csak egy-egy termék legjobb árára optimalizálnak (cherry picker típus).

A modellben reprezentált helyeket és kereskedelmi pontokat egy G gráf N halmazba rendezett csúcsaival jelöljük és az őket összekötő élek a köztük lévő úthálózatoknak felelnek meg. Az élek halmazát jelöljük A -val. Két pont közötti távolság a gráf élein vett legrövidebb út hossza, aminek kiszámítására több algoritmusból választhatunk, például a Dijkstra algoritmussal a csúcsok közötti legkisebb távolság könnyen meghatározható. Az ily módon értelmezett g távolság felel meg az utazási költségnek, amit egy $i, j \in N$ kereskedelmi pontpárra jelöljük g_{ij} -vel.

Definiáltuk az n^L és n^F számokat, melyek jelölik a vezető és a követő vállalatok elhelyezésre szánt üzleteinek számát, de hogy pontosan hol helyezkedhetnek el a síkon, azt az $M^c \subseteq N$ halmaz jelöli, ahol c adja meg hogy a vezető vagy követő vállalatának helyéről beszélünk, azaz $c \in C = \{L, F\}$. Egy i vásárló súlyát, vagy vásárlók helyett beszélhetünk városokról is, és annak népességét, vagy másképp vásárlóerejét, jelölje a_i .

A vásárlók döntésüket a hasznosság értéke alapján hozzák meg, mely megértéséhez először definiáljuk az r_i^c -t, ami megadja, hogy az i vásárló egy adott c üzletben mennyit szándékozik maximum elkölteni, valamint jelölje π_j^c a termék árát a c lánc j üzletében.

Egyszerű vásárlás során az i vásárló a j üzletben r_i^j -t költene el maximum egy termékre az útiköltséget is beleszámítva, azaz ebből a termék költsége és az oda-vissza út költsége is kivonódik. Ez definiálja az adott vásárlás hasznosságát:

$$u_{ij}^c = r_i^c - \pi_j^c - (g_{ij} + g_{ji}). \quad (1)$$

Ha $u_{ij}^c < 0$, azaz a vásárlónak a termék és az út többbe kerülne, mint amit rászánt, akkor nem éri meg ez a vásárlás, viszont ha $u_{ij}^c \geq 0$, akkor igen.

Többcélú vásárlás során még szükség van egy paraméter definiálására, ami azt határozza meg, hogy milyen összefüggés van a vezető és a követő vállalat áruja között. Tehát mennyire korrelálnak termékeik. Ezt jelöljük γ -val, és legyen $\gamma > 1$ akkor, ha a két vállalat korrelál, azaz a termékeket jobban megéri együtt vásárolni, és $0 < \gamma < 1$, ha a korreláció értéke kicsi, azaz ha jobb őket külön venni. Például, ha mosógépet árul a vezető míg a követő vállalat mosószer, akkor ott a $\gamma > 1$, mert nagyobb a készletése a vásárlónak, hogy megvegye őket egy vásárlás során. Ha viszont az egyik üzlet mosógépet árul a másik pedig elemet, akkor ott a $\gamma < 1$, mert itt nem túl nagy a készletés, hogy ugyanazon a vásárlás során szerezzék meg őket a vásárló. És a hasznosságot ekkor a következőképpen definiáljuk:

$$u_{ijk}^{LF} = \gamma(r_i^L + r_i^F) - (\pi_j^L + \pi_k^F) - (g_{ij} + g_{jk} + g_{ki}). \quad (2)$$

Ezen információk birtokában kezdődhet a modellezés, ahol meg kell fogalmaznunk a feltételeket, célfüggvényeket, paramétereket és a változókat, valamint szükségünk lehet különböző halmazokra is, amik segíthetnek a feltételek felírásában.

A többcélú vállalatelhelyezési probléma modellezése

Halmazok

A modellben feltételeink meghatározásához szükség van egyes halmazok definiálására a [4] cikk alapján.

$N_i^F = \{k \in M^F : u_{ik}^F > 0\}$: olyan k követő üzletek, ahova i vásárló pozitív haszonnal vásárol F terméket egyszerű vásárlás során.

$N_i^L = \{j \in M^L : u_{ij}^L > 0\}$: olyan j vezető üzletek, ahova i vásárló pozitív haszonnal vásárol L terméket egyszerű vásárlás során.

$N_i^{LF} = \{(j, k) \in M^L \times M^F : u_{ijk}^{LF} > 0\}$: olyan vezető és követő üzletpárok, ahol többcélú vásárlás során i vásárló pozitív haszonnal vásárol.

$Q_{im}^c = \{(l, f) \in N_i^{LF} : u_{ilf}^{LF} \geq u_{im}^c\}$: olyan vezető és követő üzletek párjait tartalmazó halmaz, ahol többcélú vásárlás során az i vásárló nagyobb hasznossággal vásárol, mint c vállalatlánc m boltjában egyszerű vásárlás során. Valamint ebbe a halmazba tartoznak még azok az esetek is, amikor a két hasznosság megegyezik.

$Q_{ijk}^c = \{m \in N_i^c : u_{im}^c > u_{ijk}^{LF}\}$: olyan vezető vagy követő üzletek halmaza, melyekbe egyszerű vásárlás során az i vásárló nagyobb hasznossággal megy, mint bármely többcélú vásárlás során.

$R_{ij}^L = \{k \in N_i^F : u_{ik}^F > u_{ij}^L\}$: azon követő üzletek halmaza, ahol az egyszerű vásárlás az i vásárló számára nagyobb hasznosságú, mint a j vezető üzletbe tett egyszerű vásárlás.

$R_{ik}^F = \{j \in N_i^L : u_{ij}^L \geq u_{ik}^F\}$: azon vezető üzletek halmaza, ahol az egyszerű vásárlás az i vásárló számára nagyobb hasznosságú, mint a k követő üzletbe tett egyszerű vásárlás. Ide tartoznak még azok az esetek is, amikor a két hasznosság megegyezik.

Döntési változók

Jelöljük x -szel és a megfelelő indexelésekkel az elhelyezési bináris változókat:

$$x_j^L = \begin{cases} 1 & \text{ha a vezető vállalatlánc a } j \text{ pontban üzletet nyit} \\ 0 & \text{különben} \end{cases}$$

$$x_k^F = \begin{cases} 1 & \text{ha a követő vállalatlánc } k \text{ pontban üzletet nyit} \\ 0 & \text{különben} \end{cases}$$

$$x_{jk}^{LF} = \begin{cases} 1 & \text{ha a követő vállalatlánc } k \text{ pontban és a vezető vállalatlánc } j \\ & \text{pontban nyit üzletet} \\ 0 & \text{különben} \end{cases}$$

Valamint legyen y -nal és a megfelelő indexeléssel jelölve a vásárlásokat megadó bináris változók:

$$y_i^L = \begin{cases} 1 & \text{ha } i \text{ vásárló a vezető vállalat üzletében egyszerű vásárlás során vásárol} \\ 0 & \text{különben} \end{cases}$$

$$y_i^F = \begin{cases} 1 & \text{ha } i \text{ vásárló a követő vállalat üzletében egyszerű vásárlás során vásárol} \\ 0 & \text{különben} \end{cases}$$

$$y_{jk}^{LF} = \begin{cases} 1 & \text{ha } i \text{ vásárló többcélú vásárlás során mindkét vállalat üzletében vásárol} \\ 0 & \text{különben} \end{cases}$$

Belső probléma célfüggvénye

A modell felírásakor szükségszerű mindkét vállalat profitjának a maximalizálását megjeleníteni, így az egyik probléma beágyazódik a másik problémába. Külső probléma a vezető vállalat profitjának a maximalizálása, míg alacsonyabb szinten megjelenik a követő vállalat profitjának a maximalizálási problémája. Az utóbbi probléma célfüggvénye a következő:

$$\phi(x^L) := \sum_{i \in N} a_i (y_i^F + y_i^{LF})$$

Ez pontosan azt írja le, hogy az egyes vezető vállalat lehetséges üzleteihez melyik követő vállalatlánc üzletei az optimálisak profit maximalizálás szempontjából.

3.1. Modell felírása

A modell felírása a [4] cikk szerint:

$$\max_{i \in N} \sum a_i(y_i^L + y_i^{LF}) \quad (3)$$

f.h

$$\sum_{j \in M^L} x_j^L \leq n^L \quad (4)$$

$$x_j^L \in \{0, 1\}, \quad \forall j \in M^L \quad (5)$$

$$\max_{i \in N} \sum a_i(y_i^F + y_i^{LF}) = \phi(x^L) \quad (6)$$

$$\sum_{k \in M^F} x_k^F \leq n^F \quad (7)$$

$$x_{jk}^{LF} \leq x_j^L, \quad \forall (j, k) \in M^L \times M^F \quad (8)$$

$$x_{jk}^{LF} \leq x_k^F, \quad \forall (j, k) \in M^L \times M^F \quad (9)$$

$$x_{jk}^{LF} \geq x_j^L + x_k^F - 1, \quad \forall (j, k) \in M^L \times M^F \quad (10)$$

$$y_i^L \geq x_j^L - \sum_{k \in R_{ij}^L} x_k^F - \sum_{(l,f) \in Q_{ij}^L} x_{lf}^{LF}, \quad \forall i \in N, j \in N_i^L \quad (11)$$

$$y_i^F \geq x_k^F - \sum_{j \in R_{ik}^F} x_j^L - \sum_{(l,f) \in Q_{ik}^F} x_{lf}^{LF}, \quad \forall i \in N, k \in N_i^F \quad (12)$$

$$y_i^{LF} \geq x_{jk}^{LF} - \sum_{l \in Q_{ijk}^L} x_l^L - \sum_{f \in Q_{ijk}^F} x_f^F, \quad \forall i \in N, (j, k) \in N_i^{LF} \quad (13)$$

$$y_i^L + y_i^F + y_i^{LF} \leq 1, \quad \forall i \in N \quad (14)$$

$$y_i^L \leq \sum_{j \in N_i^L} x_j^L, \quad \forall i \in N \quad (15)$$

$$y_i^F \leq \sum_{k \in N_i^F} x_k^F, \quad \forall i \in N \quad (16)$$

$$y_i^{LF} \leq \sum_{(j,k) \in N_i^{LF}} x_{jk}^{LF}, \quad \forall i \in N \quad (17)$$

$$y_i^L, y_i^F, y_i^{LF} \in \{0, 1\}, \quad \forall i \in N \quad (18)$$

$$x_j^F \in \{0, 1\}, \quad \forall j \in M^F \quad (19)$$

$$x_{jk}^{LF} \in \{0, 1\}, \quad \forall (j, k) \in M^L \times M^F \quad (20)$$

Matematikailag felírva a feltételeket nem olyan egyértelmű elsőre, hogy pontosan mi történik. Elsősorban a modell többszintűségét a vezető (3) és követő (6) vállalat piaci részesedésének maximalizálása adja, amit nem egyszerre kívánunk elérni. Fontos, hogy garantáljuk a vezető vállalat által elhelyezni kívánt üzletek számát (4), ugyanezt megtéve a követő céggel is (7). Mivel két vásárlási típust különböztetünk meg, ezt a modellben is jeleznünk kell, ugyanis anélkül, hogy vezető (8) üzletben vagy követőben (9) nem vásárolunk nem tudunk mindkettőbe menni, de ha pontosan mindkettőt (10) útbaejtjük egy túra során, akkor már többcélúan vásárolunk. A hasznosságok összekötése azzal, hogy az egyes fogyasztók hol vásárolnak nehezebb feladat, de a jól definiált halmazok ebben segítenek. Például egy vásárló akkor vásárol csak vezető üzletben (11), hogyha egyáltalán építünk ilyen üzletet, és nem megy nagyobb hasznossággal csak követő üzletbe, valamint nincs olyan többcélú vásárlás sem, ami jobban megérné számára. Ugyanez felírható egyszerű vásárló típusra követő vállalat esetén is (12). Abban az esetben, hogyha többcélúan szeretne vásárolni (13), akkor abban biztosnak kell lennünk, hogy mindkét vállalat építet üzletet, és egyik esetben sem jár nagyobb nyereséggel egyszerű vásárlást folytatni. Fontos, hogy egyszerű és többcélú típus esetén pontosan az egyik teljesüljön (14), és csak pozitív hasznosságú üzletbe indul el egy vásárló vásárolni, bármelyik típust választva (15) (16) (17).

3.2. Példa feladat megoldása

A következő részben egy összesen 9 pontból álló példán kerül bemutatásra a feladat, melyből $n^L = 1$, $n^F = 1$ és 5 különféle ponttal eljelölt vásárlót (vagy várost) különböztetünk meg. A pontok koordinátáinak meghatározására véletlenszerű pontokat generáltam. Ez a kód megtekinthető az A függelékben. A használt metrika az Euklideszi síkban definiált távolságfüggvény, ahol a Descartes-féle koordináta-rendszerben ábrázolva az $n \in N$ csúcsokat, két pont távolságát az alábbi módon értelmezzük:

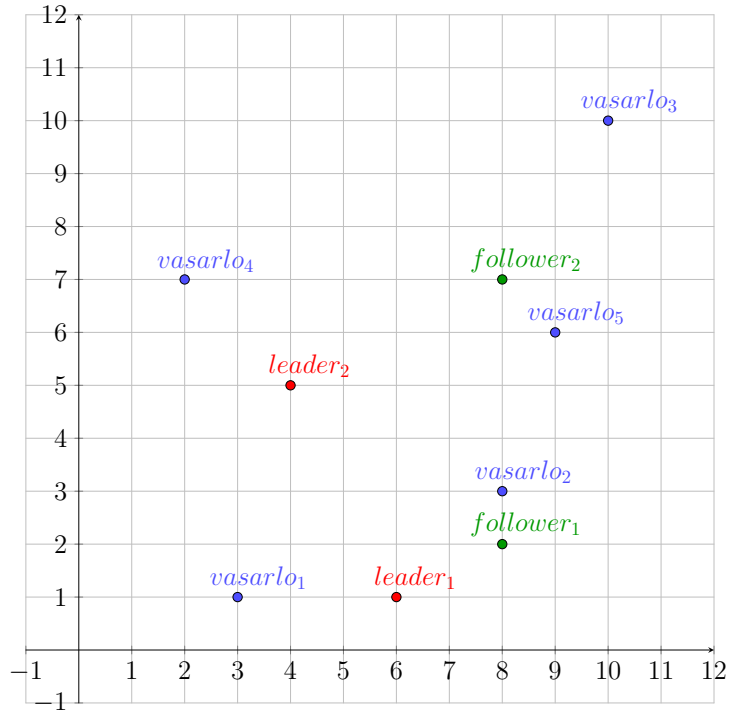
Legyen $n_1 = (x_1, y_1)$ és $n_2 = (x_2, y_2)$, $n_1, n_2 \in N$ ekkor

$$g_{n_1, n_2} = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$

jelölje a két pont távolságát.

A pontok legenerálása után a következőképpen ábrázoltam őket egy Descartes-féle koordináta-rendszerben:

Mielőtt a modellben ismertetett halmazok meghatározásra kerülnek, azelőtt szükségünk van a különböző hasznosságok (u) kiszámolására. Ezek a B függelékben található táblázatban olvashatóak.



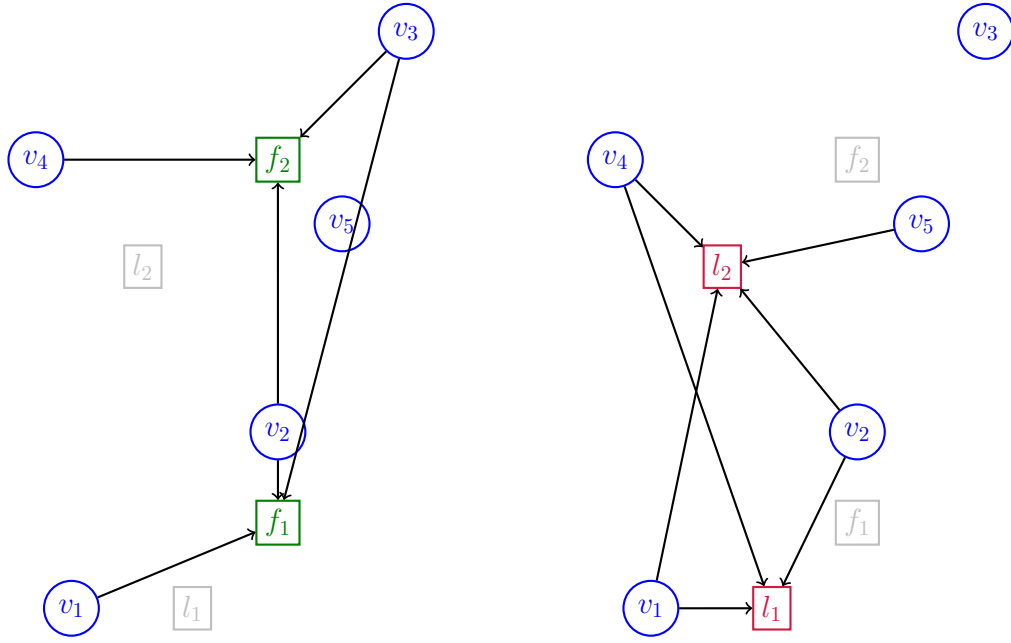
3.1. ábra. Kereskedelmi pontok: $vasarlo_i$, $i = 1, \dots, 5$ és a vállalatok lehetséges üzleteinek helyei: $leader_j$ ahol $j = 1, 2$ és $follower_k$ ahol $k = 1, 2$

Halmazok elemeinek meghatározása

A példában említett vezető és követő cégnek az egymástól való függését, azaz, hogy az egyes vásárlóknak mennyire érdeke egyszerre megvenni a két terméket, $\gamma = 0.9$ -cel definiáltam. Ez azt jelenti, hogy kevésbé van igény arra, hogy a két terméket, amit a két cég árul egyszerre vegyék meg (például: kerti hintát és egy gyerekjátékot). A B függelék táblázata alapján a modellben definiált halmazok elemei felsorolva olvashatók a C függelékben. A könnyebb áttekintés érdekében az egyes halmazokat inkább gráfokkal szemléltetem, ahol szagatott él jelzi a többcélú vásárlást és sima él az egycélút. Így például a 3.2a. ábrán egy irányított gráf látható, mely pontosan azt írja le, hogy egy adott vásárlónak melyik követő üzletbe éri meg elmenni vásárolni. A 3.2b. ábra irányított élei reprezentálják az egyes vásárlók vezető üzleteinek választásait az alapján, hogy megéri-e nekik ott vásárolniuk.

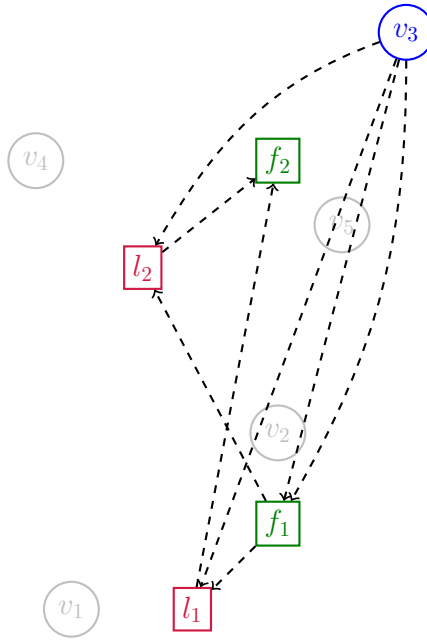
Az $N_1^{LF} = N_2^{LF} = N_4^{LF}$ és $N_3^{LF} = N_4^{LF}$ elemei megegyeznek, ezért ábrázoljuk csak az N_3^{LF} halmazt. Az 3.2c. ábrán csak a v_3 vásárló döntéseit tartalmazó többcélú vásárlási típus látható, az átláthatóság miatt. A halmaz elemei vezető és követő üzlet párok, melyek azt mondják meg, hogy a vásárlótól indulva melyik vezető majd követő bolt érintésével intézik a vásárlást.

A Q_{im}^C halmaz esetében nagyon sok lehetőség van, amiből egyet a 3.3a. ábrán ábrázoltam, amiről leolvasható az, hogy a v_1 vásárló esetén az egyszerű vásárlás a



(a) A követő lánc N_i^F elemei, $i = 1, \dots, 5$.

(b) A vezető lánc N_i^L elemei, $i = 1, \dots, 5$.

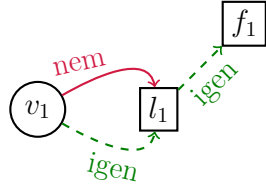


(c) A v_3 vásárló többcélú vásárlásának választásai, az N_3^{LF} elemei.

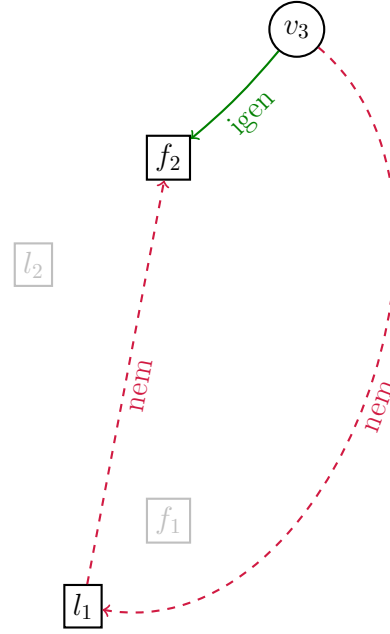
3.2. ábra. Az egyes vásárlók üzleteinek választásai pozitív hasznosság alapján

vezető l_1 üzletébe kevésbé éri meg, mint többcélú vásárlás az l_1 vezető és f_1 követő üzlet esetén.

Azokat az egyszerű vásárlásokat, amelyek jobban megérik az egyes vásárlóknak, mint bizonyos többcélú vásárlások, azt a Q_{ijk}^C halmazok foglalják össze. A 3.3b.

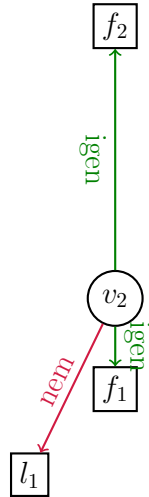


(a) A v_1 vásárló egyszerű vásárlás helyett nagyobb hasznossággal vásárol többcélúan ($Q_{1l_1}^C$ elemei).

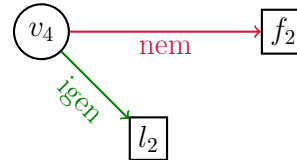


(b) A v_3 vásárló egyszerű vásárlásainak választásai többcélú vásárlásra nézve pozitív hasznosság alapján ($Q_{3l_1 f_2}$ elemei).

3.3. ábra. Példák a Q halmazok elemeire



(a) $R_{2l_1}^L$: v_2 vásárló egyszerű vásárlás esetén mely követő üzletben vásárol nagyobb hasznossággal, mint az l_1 vezető üzletben.



(b) $R_{4f_2}^F$: v_4 vásárló egyszerű vásárlás esetén mely vezető üzletben vásárol nagyobb hasznossággal, mint az f_2 követő üzletben.

3.4. ábra. Példák R halmazokra

ábrán a $Q_{3l_1 f_2}$ halmaz gráfos megjelenítése látható, ahol a v_3 vásárlónak jobban megéri egyszerű vásárlást intézni az l_2, f_1, f_2 üzletekbe, mint többcélú bevásárló

körútra indulni az l_1 és f_2 boltba.

Valamint annak meghatározására, hogy egyszerű vásárlás során melyek azok az üzletek, amelyek esetén az egyes vásárlók jobban járnak, mintha a konkurencia üzleteiben vásárolnának, ezt a 3.4a. és 3.4b. ábrákon láthatjuk.

Döntési változók értékei

Az x változó értékei attól, függenek, hogy mi lesz a megoldásunk, hiszen ahova a két vállalatlánc épít majd üzletet, azon megfelelő indexelésű változók értékei lesznek egyenlőek 1-gyel, minden más esetben pedig 0-ák. Az y változó értékei attól függenek, hogy az i . vásárló hol vásárol és hogyan, amit a hasznosságok segítségével határozzunk meg. Például az v_1 vásárlóra az optimális megoldásban a következő teljesül:

$$y_1^L = 0, \quad y_1^F = 0, \quad \text{és} \quad y_1^{LF} = 1,$$

ami azt jelenti, hogy csak többcélúan vásárol, mert a maximuma v_1 összes hasznosság értékének = 11.86 ami pontosan az u_{l_1, f_1} hasznosság (lásd a B függelék).

Eredmények

A kereskedelmi pontok (v_i) súlyait, avagy hogy hány potenciális vásárló mehet az üzletekbe, mennyi a pont vásárlóereje, a következő táblázat írja le:

a_1	a_2	a_3	a_4	a_5
36	19	53	47	28

Megvizsgáltam, hogy az egyes vezető üzletek (l_1 és l_2) esetén melyik a kedvezőbb a hasznosság értékek alapján. Tehát, hogyha egyszerű vásárlás hasznossága nagyobb, mint egy bármelyik többcélú vásárlás hasznossága f_1 vagy f_2 -be, akkor adott vásárló esetén nem éri meg oda követő üzletet építeni.

A 3.1. táblázatban összefoglaltam a kapott eredményeket. Az első öt sorban a vásárlók egyes döntései olvashatóak, miszerint például ha az l_1 és f_1 üzlet épülne meg, akkor a v_1 vásárló milyen döntést hozna, hol vásárolna. A megfelelő cellában az LF jelölés látszik, ami azt jelenti, hogy többcélúan vásárolna ekkor, mert ez éri meg számára a legjobban. A táblázat utolsó soraiban összegeztem azoknak a soroknak a vásárló erejét (a_i), ahol vezető/követő üzletet érintenek a vásárlások. Ezek alapján a maximum piaci részesedés a 183 volt, ami azt jelenti, hogy l_1, f_1 üzletek megépítésével lesz maximalizálva a vezető vállalatlánc profitja. Valamint az 5 vásárló közül négyen választanának többcélú vásárlást ezen két üzlet között, így az optimális

3.1. táblázat. A példa feladat eredményeinek összefoglalása

	$l_1 f_1$	$l_1 f_2$	$l_2 f_1$	$l_2 f_2$
v_1	LF	L	LF	LF
v_2	LF	LF	LF	LF
v_3	LF	F	F	F
v_4	LF	LF	LF	LF
v_5	L	L	-	-
piaci részesedés vezető vállalat számára	183	130	102	102
piaci részesedés követő vállalat számára	155	119	155	155

megoldás:

$$\begin{aligned}x_{l_1}^L &= 1 \\x_{f_1}^F &= 1 \\x_{l_1 f_1}^{LF} &= 1\end{aligned}$$

Tehát szükség van mindkét vállalat üzletére.

Érdemes még azt is végiggondolni, hogy milyen eredményt kaphatnánk ugyan-ezen a példán, csak a γ értéket úgy beállítva, hogy 1-nél jóval nagyobb számot vegyen fel, azaz a vásárlóknak érdekük legyen a két üzlet által kínált termékeket együtt megvenni. Vagy egy nagyon kicsi számot értékül adva a γ -nak mi történne? Például $\gamma = 0.6$ választás esetén az optimális megoldás az l_1 és az f_1 üzlet megépítése, viszont a v_1, v_4, v_5 vásárló csak vezető üzletben vásárolna, és a v_2, v_3 vásárló csak követő üzletben, ez azt jelenti, hogy mindegyik vásárló csak egyszerű vásárlást folytat ebben az esetben. Az optimum a vezető vállalat számára 111, míg a követő vállalat piaci részesedése 72 lesz.

Saját eredmények

4.1. Közös hasznosság értékének javítása

Elevenítsük fel, hogy hogyan nézett ki az eredeti modellben a hasznosság értékének kiszámolása többcélú vásárlás esetén:

$$u_{ijk}^{LF} = \gamma(r_i^L + r_i^F) - (\pi_j^L + \pi_k^F) - (g_{ij} + g_{jk} + g_{ki}). \quad (21)$$

Tehát a γ értéke árulkodik arról, hogy milyen összefüggés van a követő és a vezető vállalat között. Az r változók jelölték az egyes vásárlóknál, amit maximum hajlandóak költeni egy adott vállalatnál. A π pedig jelölte a termékek árát az adott üzletben. A (21) alapján, a termékek korrelációja azt befolyásolja, hogy mennyit szán a vásárló a két termék együttes vásárlására. Viszont a cél a közös hasznosság csökkentése lenne, nem a rászánt összegeké. Vegyünk egy példát, amin jobban látszódhat a probléma.

Példa. Legyen egy konkrét vásárlónk, egy konkrét vezető és követő üzletünk, ami megengedi, hogy elhagyjuk az i, j, k indexeket, és vizsgáljuk a $\gamma = 0.5, r^L = 10000, r^F = 5000, \pi^L = 5000, \pi^F = 6000, \sum g = 1500$ esetet. A $\gamma = 0.5$ arra utal, hogy a két termék egyáltalán nem korrelál, tehát nem érzünk külön készletet az együvvásárlásra (de ennek nem kellene kizárni a lehetőségét, ha mégis megéri).

$$u^{LF} = 0.5(10000 + 5000) - (5000 + 6000) - 1500 = -5000.$$

Módosítsuk annyival ezt a példát, hogy γ -val inkább az árrést szorozzuk meg, csökkentve az árrésből adódó hasznot:

$$u^{LF} = 0.5((10000 + 5000) - (5000 + 6000)) - 1500 = 500.$$

Ez továbbra sem módosít azon, hogy sokkal nagyobb hasznosságot kapunk simán a vezető üzletben való vásárlásra, viszont értékeinket finomítja, ugyanis nem feltétlen nem éri meg vásárolni a kettőben, ami egyből kizárná a többcélú vásárlás esélyét, hanem arányosan csökkenti ennek a hasznosságát az útiköltséget teljesen figyelembe véve. A plusz zárójel nélkül ez egyből megtörténne.

Nézzük meg a másik végletet is. Mi történne olyan esetben, hogyha a két vállalatnál otthagyni szánt pénz összege alapvetően kevés, és negatív értéke lenne a hasznoságnak, de csak ezt megszorozva egy nagy γ -val máris vásárol mindkét üzletben az adott vásárló, holott nem szánna rá ennyit alapvetően. Legyen $\gamma = 2, r^L = 1000, r^F = 5000, \pi^L = 2000, \pi^F = 8000, \sum g = 1500$.

$$u^{LF} = 2(1000 + 5000) - (2000 + 8000) - 1500 = 500.$$

Ebben az esetben az egycélú vásárlások negatívak maradnak, úgyhogy a modellünk végső soron többcélú vásárlást fog választani, habár csak az árakat tekintve már nem éri meg egyik terméket sem megvenni külön-külön sem. Viszont γ -val a teljes árrést szorozva, a következő értéket kapjuk:

$$u^{LF} = 2((1000 + 5000) - (2000 + 8000)) - 1500 = -9500,$$

ahol ennyi összeggel már valóban nem tud vásárolni a vásárló.

Tehát ezzel a módosítással sokkal valóságghűbb eredményt kapunk, és nem esünk hibás vásárlási típus választásba sem. Így a következő hasznosság definíció lesz érvényes a továbbiakban:

$$u_{ijk}^{LF} = \gamma \left((r_i^L + r_i^F) - (\pi_j^L + \pi_k^F) \right) - (g_{ij} + g_{jk} + g_{ki}).$$

4.2. Feltételek módosítása

A modellben ismertetett változók mind binárisak, viszont a való életben nem ennyire könnyű eldönteni, hogy valaki szeretne-e menni vásárolni vagy sem. Mi történik akkor, ha valaki csak 1% arányban szeretne a felépülő üzletek egyikében, vagy akár mindkettőben vásárolni, ami valóban több, mint az, hogy nem is szeretne, de mégsem annyival, mint aki 78% eséllyel menne el vásárolni. Ezen szempont kielégítésére a [4] cikkben ismertett modell egyik döntési változója a következőképpen módosul: legyen y egy valószínűség, mely arról árulkodik, hogy az egyes vásárlók vásárlásaik mekkora részében mennek a különböző vásárlásokra;

$y_i^L \in [0, 1]$: mekkora valószínűséggel vásárol i vásárló vezető valamelyik üzletében egyszerű vásárlás során.

$y_i^F \in [0, 1]$: mekkora valószínűséggel vásárol i vásárló követő valamelyik üzletében egyszerű vásárlás során.

$y_i^{LF} \in [0, 1]$: mekkora valószínűséggel vásárol i vásárló követő és vezető valamelyik üzleteiben többcélú vásárlás során.

Ezzel a változtatással persze az eredeti modell nem megfelelő, és elsősorban a (11) - (13) feltételeket kell megvizsgálni, hogy az y változó folytonossá tételével mi változik.

Hasznosság – vásárlás összefüggés (11) - (13) feltételeinek vizsgálata

Tekintsük az első hasznosság – vásárlás összefüggés feltételt az eredeti modellben:

$$y_i^L \geq x_j^L - \sum_{k \in R_{ij}^L} x_k^F - \sum_{(l,f) \in Q_{ij}^L} x_{lf}^{LF}, \forall i \in N, j \in N_i^L \quad (11)$$

Ebben a feltételben a probléma akkor adódik, amikor az i . vásárló vásárolna vezető üzletben, de nem túl nagy eséllyel, viszont adott pontban létezik az üzlet, tehát $x_j^L = 1$ és minden más 0. Ilyen esetben $1 \geq 1$ teljesülne, viszont a $0.1 \geq 1$ már nem. Ennek javítására írjuk fel, hogy hogyan is adódik a vásárlás valószínűsége. Azt mondjuk, hogy ennek a valószínűségnek a hasznosságok szerinti arányban kell változnia. Ehhez elsőként fontos, hogy minden figyelembe vett hasznosság nem-negatív legyen, azaz számoljunk a hasznosságok pozitív részével csak:

$$\tilde{u} = \max\{0, u\}.$$

Ekkor a vásárlás valószínűsége a vezető egy vállalatában megegyezik a vezető vállalatokban vett hasznosság és az összhasznosság arányával, azaz

$$y_i^L = \frac{\sum_{j \in M^L} \tilde{u}_{ij}^L x_j^L}{\sum_{\substack{j \in M^L \\ k \in M^F}} (\tilde{u}_{ij}^L x_j^L + \tilde{u}_{ik}^F x_k^F + \tilde{u}_{ijk}^{LF} x_{jk}^{LF})} \quad \forall i \in N$$

Ez ebben a formában nemlineáris, így próbáljuk először átrendezni az egyenletet,

$$\sum_{j \in M^L} \tilde{u}_{ij}^L x_j^L = \sum_{\substack{j \in M^L \\ k \in M^F}} (\tilde{u}_{ij}^L y_i^L x_j^L + \tilde{u}_{ik}^F y_i^L x_k^F + \tilde{u}_{ijk}^{LF} y_i^L x_{jk}^{LF}) \quad \forall i \in N$$

Ebben az esetben is jól látható, hogy a feltétel nem lett lineáris, viszont ha használjuk a McCormick-féle linearizálást [6], akkor ez lineárisra módosul. Mivel az egyik változónk bináris, és a másik folytonosként szerepel a szorzatban, a McCormick envelope (befoglalás) pontos eredményt ad.

1. Definíció (McCormick envelopes). *Legyen x bináris és y folytonos változó, ahol $L \leq y \leq U \in \mathbb{R}$ megfelelő korlátok y -ra. Vezessük be a w folytonos változót, amit mint $w = xy$ fogunk használni. Ekkor a*

$$\begin{aligned} w &\geq xL \\ w &\geq y - U(1 - x) \\ w &\leq y - L(1 - x) \\ w &\leq xU \end{aligned}$$

feltételek azonosan leírják a $w = xy$ egyenlőséget.

Ebben az esetben az w változó, ami a szorzatot modellezi, ugyanúgy fog változni, mint maga a szorzat, és így kikerülhet a nemlineáris feltétel. Ez alapján vezessük be a $w_c^{C_1 C_2}$ változót, ahol:

$$\begin{aligned} w_{ij}^{C_1 C_2} &= y_i^{C_1} x_j^{C_2}, & C_1, C_2 &\in \{L, F\}, \\ w_{ijk}^{C_1 L F} &= y_i^{C_1} x_{jk}^{L F}, & C_1 &\in \{L, F\}. \end{aligned}$$

Valamint, mivel az y változók értéke folytonos, de valószínűséget jelöl, ezért értéke $[0, 1]$ között lehetséges, így $L \leq y \leq U = 0 \leq y \leq 1$. Tehát a következő feltétel vehető fel a modellbe (11) helyett:

$$\sum_{j \in M^L} \tilde{u}_{ij}^L x_j^L = \sum_{\substack{j \in M^L \\ k \in M^F}} (\tilde{u}_{ij}^L w_{ij}^{LL} + \tilde{u}_{ik}^F w_{ik}^{LF} + \tilde{u}_{ijk}^{LF} w_{ijk}^{LLF}) \quad \forall i \in N, \quad (11.\text{módosított})$$

a linearizálási feltételekkel együtt,

$$\begin{aligned} w_{ij}^{LL} &\geq 0 \\ w_{ij}^{LL} &\geq y_i^L - (1 - x_j^L) \\ w_{ij}^{LL} &\leq y_i^L \\ w_{ij}^{LL} &\leq x_j^L. \end{aligned}$$

A módosított feltételben kiküszöbölésre kerül az az eset, amikor is 0-val való osztás történne, viszont ettől függetlenül még szükség van arra, hogy ez az eset is le legyen kezelve. Ennek megoldására azt kell csak megvizsgálni, hogyha egy feltételben minden $\tilde{u} = 0$, akkor a megfelelő y is legyen 0. Vegyük észre, hogy ez függetleníthető a változók értékeitől, és előfeldolgozásként megadható. Vagyis, a megfelelő indexeléssel állítsuk be a következőket:

$$\text{ha } \tilde{u}_{ij}^L = 0 \text{ és } \tilde{u}_{ik}^F = 0 \text{ és } \tilde{u}_{ijk}^{LF} = 0, \text{ akkor } y_i^L = 0 \quad \forall i \in N.$$

Hasonlóan értelmezhető a w változó többi kombinációja is. Ezek alapján a (12) és (13) feltétel a következőképpen módosul:

$$y_i^F = \frac{\sum_{j \in M^L} \tilde{u}_{ik}^F x_k^F}{\sum_{\substack{j \in M^L \\ k \in M^F}} (\tilde{u}_{ij}^L x_j^L + \tilde{u}_{ik}^F x_k^F + \tilde{u}_{ijk}^{LF} x_{jk}^{LF})} \quad \forall i \in N,$$

majd ezt rendezve és a w változót bevezetve a következőt kapjuk:

$$\sum_{k \in M^F} \tilde{u}_{ik}^F x_k^F = \sum_{\substack{j \in M^L \\ k \in M^F}} (\tilde{u}_{ij}^L w_{ij}^{FL} + \tilde{u}_{ik}^F w_{ik}^{FF} + \tilde{u}_{ijk}^{LF} w_{ijk}^{FLF}) \quad \forall i \in N. \quad (12.\text{módosított})$$

És a hozzá kapcsolódó előfeldolgozás, mely az y_i^F változót 0-val teszi egyenlővé:

$$\text{ha } \tilde{u}_{ij}^L = 0 \text{ és } \tilde{u}_{ik}^F = 0 \text{ és } \tilde{u}_{ijk}^{LF} = 0, \text{ akkor } y_i^F = 0 \quad \forall i \in N.$$

A többcélú vásárlásra vonatkozó feltétel:

$$\sum_{\substack{j \in M^L \\ k \in M^F}} \tilde{u}_{ijk}^{LF} x_{jk}^{LF} = \sum_{\substack{j \in M^L \\ k \in M^F}} (\tilde{u}_{ij}^L w_{ij}^{LFL} + \tilde{u}_{ik}^F w_{ik}^{LFF} + \tilde{u}_{ijk}^{LF} w_{ijk}^{LFLF}) \quad \forall i \in N. \quad (13.\text{módosított})$$

És a hozzákapcsolódó előfeldolgozás, mely az y_i^{LF} változót teszi 0-val egyenlővé.

$$\text{ha } \tilde{u}_{ij}^L = 0 \text{ és } \tilde{u}_{ik}^F = 0 \text{ és } \tilde{u}_{ijk}^{LF} = 0, \text{ akkor } y_i^{LF} = 0 \quad \forall i \in N.$$

A (14) feltétel vizsgálata

Ez a feltétel az előzőek miatt biztosan teljesülni fog, és továbbá nincs is szükség rá, így ezt elhagyjuk.

A (15)-(17) feltételek vizsgálata

Tekintsük a következő feltételt:

$$y_i^L \leq \sum_{j \in N_i^L} x_j^L, \forall i \in N \quad (15)$$

Ami arra ad korlátot, hogy az i . vásárló vásárol-e vezető üzletben. Ha a jobboldal 0, akkor nincs olyan telepített üzlete a vezetőnek, aminek pozitív a hasznossága, így nem vásárolhat az i . vásárló egy vezető üzletbe sem, így $y_i^L = 0$ lesz, ami a folytonos modellben is megfelelő. Ha van pozitív hasznosságú vezető vállalat, akkor a jobboldal biztosan ≥ 1 , így az y_i^L -t nem befolyásolja, hiszen az maximum 1 lehet.

Így ezen feltételen nem kell semmit sem változtatni. Ugyanez igaz a (15) és (16) feltételekre is.

18. feltétel vizsgálata

Az y változó folytonossá tétele miatt ezt meg kell változtatni a következőképpen:

$$y_i^L, y_i^F, y_i^{LF} \in \{0, 1\}, \forall i \in N \quad (18)$$

helyett

$$y_i^L, y_i^F, y_i^{LF} \in [0, 1], \forall i \in N. \quad (18. \text{ módosított})$$

Modellezés AMPL-ben

5.1. A cikkben ismertetett modell AMPL modellje

A cikkben [4] ismertetett modell megoldásához először szükségünk van három különböző fájl létrehozására. Először is a modell egy *.mod* fájlban kerül definiálásra, majd a konkrét értékeket, adatokat egy *.dat* fájlba írjuk bele, végső soron pedig szükségünk lesz egy *.run* fájlra is, ahol vagy már egy ismert solver, vagy egy implementált módszer segítségével megoldásra kerül az optimalizálási feladat. Ezen fájlok és pontos leírásuk a D függelékben találhatóak meg.

Miután implementálásra kerül a definiált modell kicsit el kell gondolkodnunk azon, hogy a *CPLEX* megoldó, amit használtunk a feladat megoldására, alapvetően nem alkalmas többszintű (bilevel) feladatok megoldására, sőt az AMPL-ben nincs is olyan beépített solver, ami erre képes lenne, ezért ezt külön kell megoldanunk.

Intuitívan először is kiválasztunk n^L darab lehetséges vezető üzletet, amire megkeressük az optimális n^F számú követő üzletet. Ezzel kapunk egy lehetséges megoldást a teljes feladatra, amit eltárolunk, hiszen ez így egy jó alsó korlát lesz. Majd ezt a tárolt értéket hasonlítjuk össze mindig a következő iteráció eredményével, ami ha jobb nála, azaz nagyobb, akkor felülírjuk a tárolt alsó korlátot. És erre addig van szükség, ameddig be nem jártuk az összes lehetséges n^L darab vezető üzletet és a hozzá optimálisan választott összes követő üzletet. Ez utóbbit kiválaszthatjuk szintén az összes lehetséges megoldásból. Kicsit általánosabban vizsgáljuk meg az időigényét ennek a módszernek:

A modellben feltettük, hogy a vezető n^L és a követő vállalat n^F üzletet szeretne elhelyezni. Mindezt úgy teszik, hogy a lehetséges helyeinek számaik, ahova egyáltalán építhetnek üzletet, az M^L és M^F halmazok számossága, amire:

$$\begin{aligned} n^L &\leq |M^L| \\ n^F &\leq |M^F|. \end{aligned}$$

Fontos észrevétel még az is, hogy az építések nem egymásután felváltva történnek, hanem egyszerre. Tehát először a vezető vállalat épít n^L üzletet, majd a követő n^F üzletet.

Az építési sorrend nem számít, hiszen először épít az egyik cég, utána a másik. Emiatt összesen:

$$\begin{pmatrix} |M^L| \\ n^L \end{pmatrix} \begin{pmatrix} |M^F| \\ n^F \end{pmatrix}$$

lehetőségünk van, hogy minden lehetséges esetet összeszámoljunk az üzletek felépítésére. Erre a következő felső korlát lesz igaz:

Állítás.

$$\binom{|M^L|}{n^L} \binom{|M^F|}{n^F} \leq \binom{|M^L|}{|M^L|/2} \binom{|M^F|}{|M^F|/2} = \mathcal{O}(2^{|M^L|+|M^F|})$$

Bizonyítás. A Stirling formula [7][8] alapján Robbins[9] az alábbi összefüggést igazolta:

$$n! < \sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\frac{1}{12n}},$$

ez alapján a következő összefüggés írható fel:

$$\begin{aligned} \binom{n}{n/2} &= \frac{n!}{\left(\frac{n}{2}\right)! \left(\frac{n}{2}\right)!} \\ &= \frac{\sqrt{2\pi n} \left(\frac{n}{e}\right)^n e^{\frac{1}{12n}}}{\sqrt{2\pi(n/2)} \left(\frac{n/2}{e}\right)^{n/2} e^{\frac{1}{12(n/2)}}} \\ &= \frac{\sqrt{2\pi n} e^{-n} e^{1/12n} n^n}{\pi n^{\frac{1}{2n}} e^{-n} e^{1/36n^2} n^n} \\ &= \frac{\sqrt{\pi n}}{\pi n} \cdot 2^n \cdot \sqrt{2} \cdot \frac{n^n}{n^n} \cdot \frac{e^{-n}}{e^{-n}} \cdot \frac{e^{1/12n}}{e^{1/36n^2}} \\ &= \frac{2^n}{\sqrt{n}} \cdot e^{\frac{3n-1}{36n^2}} \cdot \sqrt{\frac{2}{\pi}} \\ &\leq 2^n \cdot n^{-\frac{1}{2}} \cdot e^{\frac{1}{12}} \cdot \sqrt{\frac{2}{\pi}} \\ &= \mathcal{O}(2^n). \end{aligned}$$

Miért lehetséges, hogy $e^{\frac{3n-1}{36n^2}}$ -t felülről becsülhetjük $e^{1/12}$ -vel? Mivel $e^{\frac{3n-1}{36n^2}}$ értékkészlete 1 és e között mozog $\forall n$, mert $0 < \frac{3n-1}{36n^2} < \frac{1}{12} \quad \forall n \in \mathbb{N}$, ugyanis

(i) $n = 1$ esetén: $\frac{4-1}{36} = \frac{1}{12}$

(ii) $n \rightarrow \infty$ esetén: $\frac{3n-1}{36n^2} = \frac{3n}{36n^2} - \frac{1}{36n^2} = \frac{1}{12} \frac{1}{n} - \frac{1}{36} \frac{1}{n^2}$, a domináns tagok $\frac{1}{n}$, $\frac{1}{n^2}$ tartanak 0-ba, így a tört határértéke is 0.

(iii) a sorozat az egész értelmezési tartományán monoton csökkenő, azaz:

$$\frac{\frac{3n-1}{36n^2}}{\frac{3(n+1)-1}{36(n+1)^2}} > 1$$

$$\frac{3n-1}{36n^2} \cdot \frac{36(n+1)^2}{3(n+1)-1} > 1$$

$$\frac{(3n-1)(36(n+1)^2)}{(36n^2)(3(n+1)-1)} > 1$$

$$\frac{108n^3 + 180n^2 + 36n - 36}{108n^3 + 72n^2} > 1$$

$$\frac{108n^3 + 72n^2}{108n^3 + 72n^2} + \frac{108n^2 + 36n - 36}{108n^3 + 72n^2} > 1$$

$$1 + \frac{108n^2 + 36n - 36}{108n^3 + 72n^2} > 1$$

Ekkor csak azt kell megvizsgálnunk, hogy $\frac{108n^2+36n-36}{108n^3+72n^2} > 0$, ami igaz $\forall n$, mert a számláló > 0 , hiszen $108n^2 + 36n > 36 \forall n \in \mathbb{N}$, és a nevező szintén $> 0 \forall n \in \mathbb{N}$, hiszen csak pozitív tagjai vannak.

Ezután ezt a képletet felhasználva és n helyére $|M^L|$ -t illetve $|M^F|$ -t helyettesítve kapjuk az állításban szereplő összefüggést. ¹

□

Tehát a modell a helyek lehetséges számában exponenciális. Ami annyit jelent, hogy a probléma nagyon nagy M^L és M^F értékre nem lesz megoldható a gyakorlatban. Így [4] probléma NP nehéz (valamint kombinatorikus optimalizálási probléma [10] léte miatt NP-teljes is).

Jelen esetben a mi általunk választott példa feladat nem lesz nehéz, mert $|M^L| = |M^F| = 2$, és mindkét esetben 1-1 üzletet szeretnénk elhelyezni, ami összesen 4 lehetőség.

Azon *.run* fájlt futtatva, melyben a példa feladat immáron többszintű feladatként van megoldva (azaz nem beépített megoldó segítségével, hiszen azok nem tudnak megoldani többszintű optimalizálási feladatot) a vezető üzlet profitjának maximalizálása érdekében az l_1 és f_1 üzlet megépítésére van szükség, ezáltal a vezető üzlet piaci részesedése 183, míg a követőé ekkor 155 lesz. Tehát valóban azt kaptuk, amit kézzel kiszámoltunk az előző feladatban.

¹Ha n páratlan, akkor $n! \leq (n+1)!$ és $O(2^{n+1}) = O(2 \cdot 2^n) = O(2^n)$

5.2. A módosított modell AMPL modellje

Mivel ebben a modellben megjelenik egy újabb változó, ami a w mindenféle alsó és felső indexszel ellátva, ezért sok új feltételt kell felvenni ahhoz, hogy a modell lineáris maradjon. Hiszen (megint csak mindenféle indexeléstől eltekintve) $w = yx$. Ezért 9 különböző w -t kell definiálni a modellben, melyekhez mindegyikhez tartozik 4 – 4 feltétel. Ez összesen 36 új feltétel. Ezenfelül még 3 másikra is szükség van, az előző fejezetben írtak alapján, viszont mind a Q és mind az R halmazok elhagyhatóak. Ezen módosítások a E függelékben találhatóak meg. A megoldást elemezve kapjuk, hogy az l_1 és f_1 üzlet megépítése lesz az optimális, hasonlóan, mint amit kaptunk az előző modellben is, de itt már teljesen más célfüggvény értékekkel. A vezető vállalat piaci részesedése 143.17, míg a követő vállalaté 129.903 szemben az előző megoldással, amiben a vezető vállalaté 183 volt, míg a követőé 155.

5.3. A két modell összehasonlítása

Anélkül, hogy a kapott eredményeket összevetnénk, úgy tűnhet, hogy nem is történt semmi, hiszen ugyanazt az eredményt kaptuk, miszerint l_1 és f_1 üzleteknek kell megépülnie. Az 5.2.táblázatban összefoglaltak alapján levonhatóak az alábbi következtetések:

- (i) a bonyolult, összetett halmazokat elhagyhattuk teljesen
- (ii) új változók sok új feltételt hoztak magukkal, ami a futási időn² nem javított
- (iii) sokkal valóságosabb modellt kaptunk, hiszen tekintve egy példán (5.3. táblázatban az összes ilyen eredmény látható) a v_3 vásárló esetén bináris változokkal csak többcélú vásárlást engedtünk meg, holott folytonos változók esetén jól látható, hogy nagyon kicsi a különbség az egyszerű és többcélú vásárlás között, ezért nem érdemes kizárni az egyszerű vásárlást.

Valamint foglaljuk össze a kapott eredményeket is. Az eredeti modell alapján a vezető vállalat piaci részesedése 183 volt, míg a követő vállalaté 155. A módosított modell eredménye megtekinthető a E. függelékben is, ahol a vezető vállalat piaci részesedése 143.17, míg a követő cégé 129.903 volt. Mindkettő modell esetében az l_1 és f_1 üzlet megépítése az optimális megoldás, viszont a hasznosság értékek teljesen máshogyan alakultak. Míg az eredeti modell esetében csak az egyik vásárlási típust választhatja a vásárló, addig a módosított modellben a hasznosság arányában választja az egyes típusokat. Az optimális megoldás vásárlóereje 5.3. táblázatban

tekinthetők meg.

De ahhoz, hogy egy sokkal jobb összehasonlítást kapjunk szükségünk van többféle példa megoldására is, ahol immáron nem csak egy-egy vezető és követő üzletet építünk, hanem akármennyit. Ehhez a megoldási ötlet megfelelő számú részhalmazok kiválasztásán alapszik. Azaz, hogyha összesen M^L számú vezető üzlethelyből választhatunk, és n^L darabot szeretnénk kiépíteni, akkor minden lehetséges n^L részhalmazát ki kell választanunk az M^L halmaznak. A kiválasztott üzletekhez tartozó x_j^L változóknak 1 értéket adva oldjuk meg a követő vállalat maximalizálási problémáját, ahol szintén lehetőségünk van n^F üzletet építenünk. Minden ilyen iterációban elmentjük a vezető vállalatra kapott értéket, azaz, hogy ezen rögzített x_j^L és optimális x_k^F változók mellett mi a megoldás. Hogyha egy másik rögzített részhalmazra nagyobb értékű megoldást találunk, azaz nagyobb profitot tud elérni a vezető vállalat, akkor felülírva az addig talált legjobb megoldást, mentjük el az új eredményt. Ezt egészen addig iterálva, ameddig az összes n^L elemű részhalmazunkra megtaláltuk a legjobb megoldást. Ez a leszámplálási módszer AMPL-ben az F. függelékben található.

Tekintsük a 5.4., 5.5., 5.6., 5.7. táblázatokat, ahol különböző halmaz számosságok, vásárló számok és γ értékek mellett megvizsgáltuk az eredeti modell és a módosított modell futási idejét, valamint 5.9. táblázatban célfüggvények értékeit hasonlítottuk össze. Itt fontos megemlíteni, hogy újradefiniáltuk a többcélú vásárlás hasznosság értékét is, tehát az eredetiben és a módosított modellben különbözőt kellene használnunk, viszont ebben az esetben nem lenne értelme az összehasonlításnak ilyen nagy példákon. Hiszen ez egy nagyon fontos eleme a modellnek, és ha ez nem ugyanaz, akkor kezelhetjük két különböző modellnek is, de most nem ezt szeretnénk. Így csak a mi általunk javított hasznosságot használjuk.

A táblázatok futási ideje arra ad bizonyítékot, hogy melyek azok az értékek, amelyekről a modell kiszámításának futásideje függ, és nagyon nagyra választva ezeket a függőségeket láthatóan is exponenciális időben fut a feladat. Mindkét modell esetében ezek az értékek az üzletek lehetséges és építeni kívánt számai, ami egyértelmű, hiszen ez egy kiválasztással oldható meg, ami exponenciális időben valósul meg. Ugyanakkor megfigyelhető, hogy sok vásárló esetén is megnő a módosított modell ideje, ami annak „köszönhető”, hogy a linearizálás miatti új változók száma és a hozzátartozó feltételek mennyisége függ a vásárlók számától.

Másrészről, a γ értékének változtatása nem befolyásolja szignifikánsan a modell futási idejét, ami annak köszönhető, hogy egyik modellben sem ad hozzá új feltételeket vagy változókat.

²futási idő kiszámításához futtatott parancs:

display _ampl_elapsed_time + _total_solve_elapsed_time;

Egy kiválasztott, véletlenszerűen generált példának elemezzük a megoldását. Legyen összesen 7 lehetséges üzlethely a vezető vállalat számára és 6 lehetséges üzlethely a követő vállalat számára. Ezekből a vezető cég hármat, a követő cég pedig kettőt szeretne felépíteni. Összesen 10 vásárló szeretne az üzletekben vásárolni és a két vállalat közötti korreláció értéke 0.5. A pontos koordinátákat és az eredményeket az 5.5a. és az 5.5b. ábrán láthatjuk.

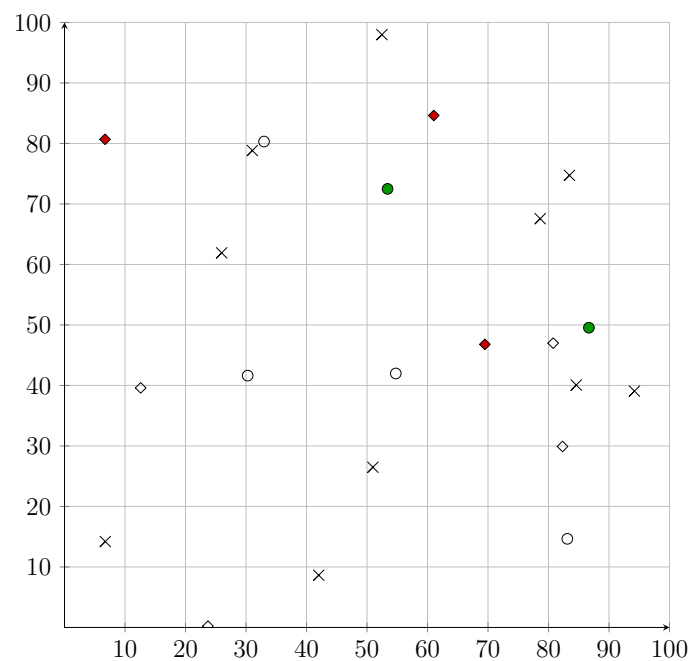
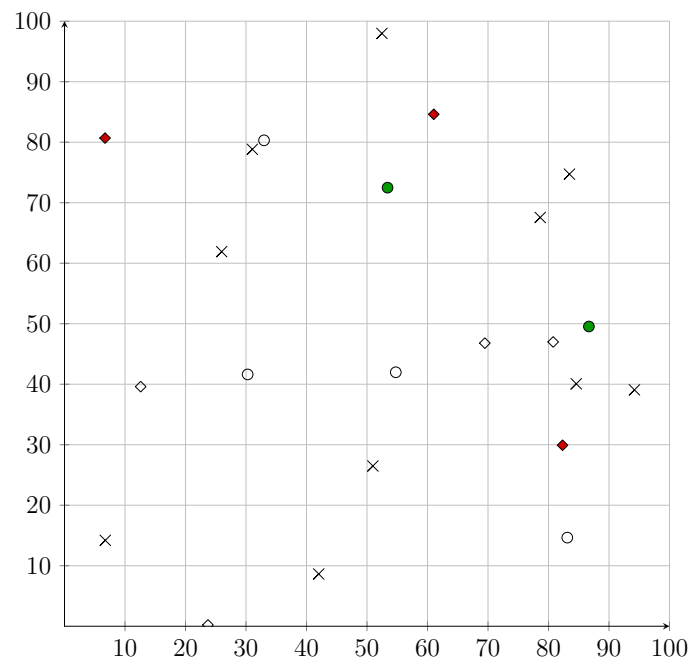
Az eredeti modell esetében a 10-ből 4 vásárló választ egyszerű vásárlást vezető üzletbe, és 6 vásárló egyszerű vásárlást követő üzletbe. Azaz senki sem végez többcélú vásárlási típust. Így a vezető vállalat piaci részesedése 225.438, míg a követőé 286.441. A módosított modell esetében a vezető vállalat piaci részesedése 289.083, míg a követőé 347.752 volt. Például az egyes vásárló esetében 38%-kaé vásárol vezető üzletben egyszerű vásárlás esetén, de mindössze 4%-kal van lemaradva a többcélú vásárlási típus, ami esetében az eredeti modell csak az egyszerű vásárlási típust választja. A pontos vásárlási típusok a 5.8. táblázatban olvashatóak.

5.8. táblázat. Az eredeti és a módosított modellben választott vásárlási típusok.

	eredeti modell			módosított modell		
vásárló:	y_L	y_F	y_{LF}	y_L	y_F	y_{LF}
1	1	0	0	0.387193	0.26623	0.346577
2	0	1	0	0	1	0
3	0	1	0	0	0.802951	0.197049
4	1	0	0	0.460636	0.337272	0.202092
5	0	1	0	0.257307	0.349349	0.393344
6	0	1	0	0.264532	0.41765	0.317818
7	0	1	0	0.228898	0.454604	0.316498
8	1	0	0	0.751061	0.113119	0.135821
9	0	1	0	0	0.922263	0.0777367
10	1	0	0	1	0	0

	vezető célfüggvény értéke		követő célfüggvény értéke	
	eredeti	módosított	eredeti	módosított
kis γ esetén	127.971	408.509	450.764	435.167
közepes γ esetén	467.976	565.052	301.319	545.858
nagy γ esetén	511.879	476.485	511.879	443.384

5.9. táblázat. Célfüggvény értékek összehasonlítása, 10 lehetséges vezető üzletből 4-et elhelyezve, és 10 lehetséges követő üzletből 3-at elhelyezve 10 vásárló esetén



5.5. ábra. A modellek által választott helyek pirossal jelölve a vezető, zölddel jelölve a követő megoldását. Jelölje a vásárlókat x , a vezető vállalat lehetséges helyeit rombusz és a követő vállalat lehetséges helyeit kör.

5.2. táblázat. A két modellt összehasonlító táblázat: + jelöli a tartalmazást, – jelöli a nemtartalmazást

	módosított modell		eredeti modell
halmazok	+	Q, R halmazok	-
változók	+	w	-
feltételek	-	(14)	+
	+	w feltételei	-
futási idő	0.375		0.219
v_3 vásárló esetén kapott y értékek	0	y_3^L	0
	0.477697	y_3^F	0
	0.522303	y_3^{LF}	1

5.3. táblázat. Az optimális megoldás y változó értékeinek az összehasonlítása

	eredeti modell			módosított modell		
vásárló:	y_L	y_F	y_{LF}	y_L	y_F	y_{LF}
v_1	0	0	1	0.393582	0.0350709	0.571347
v_2	0	0	1	0.190359	0.333852	0.475789
v_3	0	0	1	0	0.477697	0.522303
v_4	0	0	1	0.155557	0.146936	0.697507
v_5	1	0	0	1	0	0

5.4. táblázat. Vásárlók számának változtatása rögzített $|M^L| = 5$, $|M^F| = 5$ halmaz számosságok és $n^L = 3$, $n^F = 2$ kiépíteni kívánt üzletek száma mellett.

vásárlók száma:	módosított modell futási ideje	eredeti modell futási ideje
10	2.016	0.656
20	5.391	1.031
50	12.906	1.094
100	28.375	1.156

5.5. táblázat. Lehetséges üzletek számának változtatása rögzített $n^L = 4$, $n^F = 3$ kiépíteni szánt üzlet szám és vásárló számok mellett.

üzletek lehetséges helyei:	módosított modell futási ideje	eredeti modell futási ideje
$ M^L = 6, M^F = 6$	6.531	1.531
$ M^L = 7, M^F = 7$	13.672	3.75
$ M^L = 8, M^F = 8$	26.516	4.484
$ M^L = 10, M^F = 10$	51.703	11.406
$ M^L = 20, M^F = 20$	187.25	12.344

5.6. táblázat. Kiépíteni kívánt üzletek számának változtatása rögzített $|M^L| = 6$, $|M^F| = 6$ halmaz számosságok és vásárló szám (= 10) mellett.

telepíteni kívánt üzletek száma	módosított modell futási ideje	eredeti modell futási ideje
vezető: 3, követő: 4	54.093	11.11
vezető: 3, követő: 2	55.968	7.141
vezető: 3, követő: 3	59.172	10.531

5.7. táblázat. Két vállalat függőségének (γ) változtatása rögzített $|M^L| = 5$, $|M^F| = 5$ halmaz számosságok, $n^L = 3$, $n^F = 2$ és vásárló számok mellett.

γ	módosított modell futási ideje	eredeti modell futási ideje
0.1	1.828	0.484
0.4	1.516	0.406
0.6	2.171	0.922
0.9	2.922	0.609
1.5	3.484	0.625

Jövőbeli tervek

6.1. Branch and Cut algoritmus

Egy lineáris optimalizálási probléma sokféle lehet, függve attól, hogy milyen változók szerepelnek benne (binárisak vagy folytonosak). Jelen esetben a módosított modellben bináris és folytonos változók is szerepelnek, de a feltételekkel kiküszöbölésre került a nemlinearitás, tehát lineáris maradt a modell. Ez azt jelenti, hogy a követő feladata a MILP problémakategóriába esik, ami a *Mixed Integer Linear Programming* problémákat foglalja magában. Ezek megoldása, a legrosszabb esetet is figyelmebevéve nem hatékony, és NP-teljes is.

Az egyik ilyen algoritmus osztály a Vágósíkok eljárása (*Cutting Planes*). Ahogyan a nevében is benne van, valamilyen módon vágásokat hajt végre a lehetséges megoldások halmazán, amely konvex. Első lépésben a feladatot relaxált feladatra kell átírni, elhagyva az egészértékűségi feltételeket, majd olyan vágási feltételeket kell felvenni, melyek esetében egy vágás sem távolít el egyetlen lehetséges egész megoldást, de a relaxált LP feladat optimális megoldását kizárja. Ezesetben lineáris egyenlőtlenségeket veszünk fel feltételeknek.

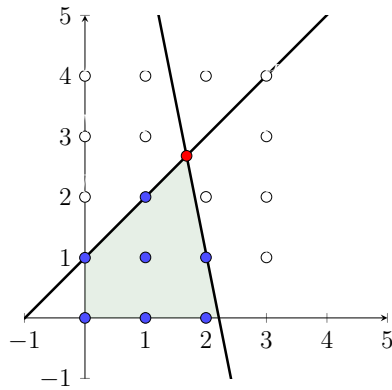
Másik algoritmus osztály a Korlátozás és Szétválasztás (*Branch and Bound*), ahol a problémát több kisebb problémára bontjuk, majd ezen kisebb problémák megoldásával közelítjük a feladat optimális megoldását. Először itt is az eredeti feladat relaxált változatát oldjuk meg. Mindig azon változó mentén ágaztatjuk szét újabb részproblémákra a feladatot, mely nem egész, és akkor érkezünk egy részprobléma lezárásához, ha minden változónk értéke egész lett. Tehát az új korlátozó feltételek mindig az egyes változókra lesznek bevezetve, amelyek kisebbek vagy nagyobbak lesznek az adott nemegész értéknél.

Tekintsük a következő példát (6.6. ábra) arra, hogy a két algoritmus milyen korlátozó feltételekkel dolgozik:

$$\begin{aligned} \max \quad & x_1 + x_2 \\ -3.18x_1 + 3.17x_2 & \leq 3.15 \\ 4.1x_1 + 0.82x_2 & \leq 9.08 \\ x_1, x_2 & \geq 0. \end{aligned}$$

Az ábrán piros jelölje a relaxált feladat (azaz elhagyásra került a változók egészértékűségére vonatkozó feltétel) optimális megoldását, fehér pontok a nem lehetséges megoldásoknak a helyét és a kék pontok a lehetséges megoldásokét.

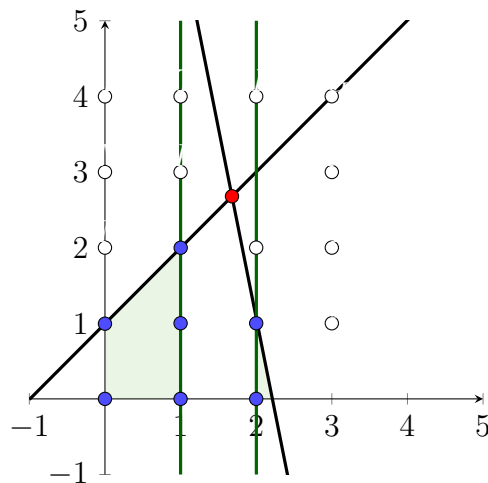
Ebben az esetben a *Branch and Bound* a következőképpen korlátozza az x_1 változót: (6.7. ábra). Itt a zöld egyenesek kerültek felvételre, mint feltételek az egy-egy



6.6. ábra. A relaxált feladat ábrázolása, ahol az optimum: (1.68, 2.68)

részfeladatra, azaz

$$\begin{aligned} x_1 &\geq 2 \\ x_1 &\leq 1. \end{aligned}$$

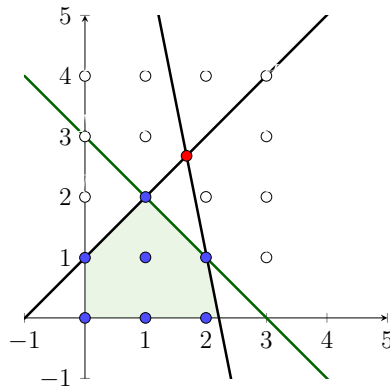


6.7. ábra. Branch and Bound algoritmus alkalmazása: x_1 változóra felvett korlátokat jelölik a zöld egyenesek, és a lehetséges megoldások helyét halvány zöld színnel jelölve lehet leolvasni az ábráról.

Most tekintsük a *Cutting Planes* algoritmus vágását a 6.8. ábrán, ahol a következő feltételt adtuk hozzá a feladathoz:

$$x_1 + x_2 \leq 3.$$

A relaxált probléma esetében fontos megjegyzés, hogy optimuma sosem lesz rosszabb, mint az eredeti probléma optimuma.



6.8. ábra. Cutting Planes algoritmus alkalmazása: relaxált probléma optimális megoldásának kivágása úgy, hogy minden lehetséges egész megoldást megtartsa a felvett feltétel. Zöld színnel jelölve ezt a feltételt, és szintén zölddel a lehetséges megoldások helyét.

A kettő algoritmus vegyítésével kapjuk a *Branch and Cut* algoritmust, ahol a vágás résznél eldönthetjük, hogy egy vágósíkot alkalmazunk, vagy a *Branch and Bound* algoritmus elágaztatását szeretnénk alkalmazni egy részproblémán, amiből újabb részproblémákat kapunk.

Ezt a keretalgoritmust lehet használni tulajdonképpen többszintű feladatok esetén is. Ami a nehézséget okozza, hogy még a relaxált feladatot se tudjuk megoldani a külső problémára, így fontos, hogy olyan feladatokat definiáljunk, amik megoldhatóak, és alsó illetve felső korlátot adjanak az optimális megoldásra.

Amiért szükség lenne ezen algoritmus implementálására az az, hogy a probléma nagyon nagy inputnál kiszámíthatatlan lesz. A kis példa feladatban mindössze $1 - 1$ üzletet szeretnénk építeni, de több üzlet építésénél nem ennyire egyszerű a megoldás.

6.2. Vásárlók kategorizálása

Felmerülhet olyan kérdés is, hogy mi lenne akkor, ha a vásárlókat vásárlási szokásaikat kihasználva kategorizálnánk, ezáltal még pontosabbá téve a modellt. Ahol ez szerepet játszana, az az adatgyűjtési időszak, és a modellnek a hasznosság értékét módosítania csak. Ehhez használjuk fel Popkowski Leszczyc cikkjének [11] a csoportosítását, melyben három csoportról tesz említést, és három szempont alapján hasonlítja őket össze. Az egyik a termékek árának a figyelembe vétele, a másik az üzletek elhelyezkedése és a harmadik a vásárlás célja, ami többcélú vagy egyszerű lehet. A három csoport:

1. időkorlátos szolgáltatás kereső (time-constraint service seekers)

2. időkorlátos árkereső (time-constraint price seekers)

3. mazsolázó (cherry pickers)

Összefoglalva, a szolgáltatás kereső képes sokat költeni egy vásárlás során, és ők a leginkább többcélú vásárlók. Nagyobb mennyiségben vásárolnak, és ezért képesek messzebb menni. Ugyanakkor ha egyszerű vásárlást csinálnak, akkor a közelebbi boltokban vásárolnak, akár drágábban is.

Az árkereső, mind a két vásárlási stratégiakor képes elmenni a legmesszebbre, hogy jó árban fizessen.

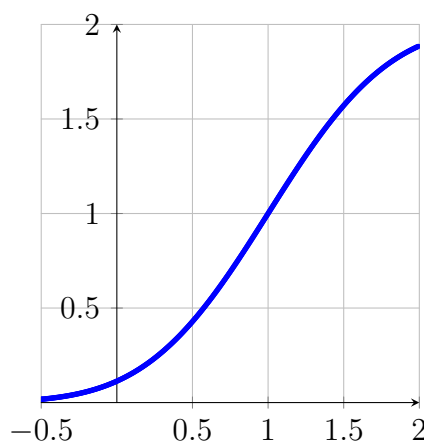
A mazsolázó pedig nem csinál többcélú vásárlást, csakis a legjobb kedvezmény érdekli, tehát az egyszerű vásárlás több megállással jellemzi.

Ezek alapján a következő tényező adható a modellhez: hogy kinek mennyire fontos az utazási költség, amit jelöljünk λ -val, és szorozzuk meg ezzel az értékkel a hasznosság útiköltségét, mind egyszerű és többcélú vásárlás esetében is. A λ legyen annál nagyobb, minél fontosabb valakinek az, hogy keveset kelljen utazni egy adott üzlethez.

Legyen λ egy X véletlen változóhoz tartozó eloszlásfüggvény, melynek értékei $[0; 2]$ között mozoghatnak, valamint ez az X véletlen változó kövessen normális eloszlást, azaz

$$\lambda = F(X), \quad \text{ahol } X \sim N(\mu, \sigma), \quad \mu = 1, \quad \sigma = \frac{1}{\sqrt{2\pi}}$$

Ekkor a λ értékek meghatározására vegyük ezen paraméterek mellett a normális eloszlás eloszlásfüggvényét (ld. 6.9. ábrán).



6.9. ábra. Az útiköltségek eloszlása.

Így a λ értéke 1 lesz amikor a vásárlónak a távolság közepesen fontos, viszont annak akinek egyáltalán nem fontos, ő is kap egy egészen kicsi λ értéket, hiszen nem

elhanyagolható teljesen az útiköltség (pontosan 0.04 lesz ebben az esetben a λ). És akinek a távolság nagyon fontos, hozzá 1-nél nagyobb λ értéket rendelünk.

Ha ily módon sztochasztikus elemeket is beépítünk a feladatba, akkor többszintű sztochasztikus programozási feladatot kapunk, aminek a megoldhatósága ugyan kérdéses, de távolabbi terveim között szerepel ennek megvizsgálása is.

Összefoglalás

Munkámban megvizsgáltam egy vállalatelhelyezési problémát, ahol egy vezető vállalat és egy őt követő vállalat szeretné maximalizálni a piaci részesedését úgy, hogy mindketten tudják a másik létét a piacon, valamint a lehetséges kiépíteni kívánt üzleteinek a helyét is. A kettő vállalat nem versenyzik egymással, hiszen más-más nélkülözhető terméket árulnak. A modellben még részt vesz egy szereplő, aki a vásárló. Tőle függ a vállalatok elhelyezése aszerint, hogy többcélú vagy egyszerű vásárlást választ. A kettő közötti különbség, hogy míg az előbbi egy mindkét vállalatot érintő vásárlási kör, addig az utóbbi csak az egyik cégek érinti egy vásárlás során. Ezen három résztvevővel egy többszintű optimalizálási problémához jutunk, ahol első szinten a vezető vállalat szeretné maximalizálni piaci részesedését míg a második szinten a követő cég, és a vásárlók mindkét szinten szerepelnek. A vállalatokat és a vásárlókat összekötő paraméter a hasznosság, ami egy értéket ad annak, hogy mennyire éri meg a vásárlóknak az adott vállalatnál vásárolni. Közvetetten, ez megadja, hogy megéri-e az egyes vállalatoknak egy adott helyre üzletet építeni az egyes vásárlók különböző vásárlási típusa mellett. A hasznosság értéke és a kiválasztása között egyenes arányosság figyelhető meg. Az egycélú vásárlás esetén mindössze az elkölteni kívánt összeg különbsége a megtett távolsággal és a termékek boltba kerülési költségének különbségével vesz részt, míg a többcélú vásárlás esetén egy γ paraméter jelzi a két vállalat termékeinek függését.

Az eredeti modellhez képest először az előbb említett hasznosság függvényén módosítottunk, mert többcélú vásárlás esetén nem volt realiztikus a megadott függvény. A probléma abból adódott, hogy kevés elkölthető pénz mellett is egy nagy γ értékkel rendelkező függéssel hirtelen többet akartunk áldozni a termékekre, mint addig. Ezt átalakítva kijavítottuk az így kiugró értékeket. Ezután egy sokkal valószínűbb modellt hoztunk létre az alapján, hogy a vásárlók mennyi eséllyel vásárolnak a különböző vásárlási típusok szerint. Ez az arány egy klasszikus valószínűségi mezőn mért valószínűséggel írható le, amit sokszor csak *kedvező/összes* esetként szoktunk emlegetni. A könnyű számítási módszer viszont egy annál nehezebben megoldható modellt alakított ki. Ez a nehéz modell nemlineáris lett, ami korántsem a legjobb, viszont az ismert McCormick-féle linearizálási módszert felhasználva a modellt lineárisrá tettük. Az új változó megjelenése sok új feltételt hozott magával annak érdekében, hogy a már létező két változóval kapcsolatban maradjon. Ami viszont megváltozott, hogy az eredeti modell bonyolultan értelmezhető összetett halmazai eltűntek, és nincs tovább szükség rájuk. A módosított modell gyakorlati hasznát, AMPL-ben modellezve tesztelni is tudtuk, ami segített empirikus módon bizonyítani, hogy szükségszerű volt a változtatás. Mivel a beépített megoldók esetében nincs lehetőség többszintű feladatok megoldására, ezért egy brute-force algoritmus-

sal tudtuk megoldani ezeket. Tárgyaltuk ennek a javítási lehetőségét a Branch and Cut algoritmus segítségével, amihez megfelelő korlátokat is felírtunk.

Összefoglalva, a modell bemutatta, hogy milyen erős befolyása van a fogyasztóknak a piacra. Jövőbeli tervek közé tartozik a tárgyalt *Branch and Cut* algoritmus implementálása, ami egy sokkal hatékonyabb módszer lesz, bár továbbra is a probléma megoldása NP-nehéz, így nagy input esetén garantálni exponenciális időt lehet csak. Továbbá a vásárlók kategorizálása különböző feltételek mellett is pontosabb eredményeket implicál, aminek az előző fejezetben egy ötletét írtuk le.

Irodalomjegyzék

- [1] N. Megiddo, E. Zemel és S. L. Hakimi, “The Maximum Coverage Location Problem”, *SIAM Journal on Algebraic Discrete Methods*, 4. évf., 2. sz., 253–261. old., 1983. DOI: [10.1137/0604028](https://doi.org/10.1137/0604028). eprint: <https://doi.org/10.1137/0604028>. cím: <https://doi.org/10.1137/0604028>.
- [2] A. Tamir és N. Halman, “One-way and round-trip center location problems”, *Discrete Optimization*, 2. évf., 2. sz., 168–184. old., 2005, ISSN: 1572-5286. DOI: <https://doi.org/10.1016/j.disopt.2004.12.004>. cím: <https://www.sciencedirect.com/science/article/pii/S1572528605000228>.
- [3] M. Fischetti, I. Ljubic, M. Monaci és M. Sinnl, “A New General-Purpose Algorithm for Mixed-Integer Bilevel Linear Programs”, *Operations Research*, 65. évf., 2017. szept. DOI: [10.1287/opre.2017.1650](https://doi.org/10.1287/opre.2017.1650).
- [4] A. Lüer-Villagra, V. Marianov, H. Eiselt és G. Méndez-Vogel, “The leader multipurpose shopping location problem”, *European Journal of Operational Research*, 302. évf., 2. sz., 470–481. old., 2022, ISSN: 0377-2217. DOI: <https://doi.org/10.1016/j.ejor.2022.01.016>. cím: <https://www.sciencedirect.com/science/article/pii/S0377221722000340>.
- [5] P. T. Popkowski Leszczyc, A. Sinha és A. Sahgal, “The effect of multi-purpose shopping on pricing and location strategy for grocery stores”, *Journal of Retailing*, 80. évf., 2. sz., 85–99. old., 2004, ISSN: 0022-4359. DOI: <https://doi.org/10.1016/j.jretai.2004.04.006>. cím: <https://www.sciencedirect.com/science/article/pii/S0022435904000247>.
- [6] G. P. McCormick, “Computability of global solutions to factorable nonconvex programs: Part I — Convex underestimating problems”, *Mathematical Programming*, 10. évf., 1. sz., 147–175. old., 1976. dec. DOI: [10.1007/bf01580665](https://doi.org/10.1007/bf01580665). cím: <https://doi.org/10.1007/bf01580665>.
- [7] L. L. Cam, “The Central Limit Theorem Around 1935”, *Statistical Science*, 1. évf., 1. sz., 1986. febr. DOI: [10.1214/ss/1177013818](https://doi.org/10.1214/ss/1177013818). cím: <https://doi.org/10.1214/ss/1177013818>.
- [8] K. Pearson, “Historical Note on the Origin of the Normal Curve of Errors”, *Biometrika*, 16. évf., 3/4. sz., 402. old., 1924. dec. DOI: [10.2307/2331714](https://doi.org/10.2307/2331714). cím: <https://doi.org/10.2307/2331714>.
- [9] H. Robbins, “A Remark on Stirling’s Formula”, *The American Mathematical Monthly*, 62. évf., 1. sz., 26–29. old., 1955, ISSN: 00029890, 19300972. cím: <http://www.jstor.org/stable/2308012> (elérés dátuma 2022. 11. 15.).
- [10] H. H. Hoos és T. Stützle, “1 - INTRODUCTION”, *Stochastic Local Search*, The Morgan Kaufmann Series in Artificial Intelligence sor., H. H. Hoos és T. Stützle, szerk., San Francisco: Morgan Kaufmann, 2005, 13–59. old., ISBN: 978-1-55860-872-6. DOI: <https://doi.org/10.1016/B978-155860872-6/50018-4>. cím: <https://www.sciencedirect.com/science/article/pii/B9781558608726500184>.

- [11] P. T. Popkowski Leszczyc, A. Sinha és A. Sahgal, “The effect of multi-purpose shopping on pricing and location strategy for grocery stores”, *Journal of Retailing*, 80. évf., 2. sz., 85–99. old., 2004, ISSN: 0022-4359. DOI: <https://doi.org/10.1016/j.jretai.2004.04.006>. cím: <https://www.sciencedirect.com/science/article/pii/S0022435904000247>.

Köszönetnyilvánítás

Ezúton szeretnék köszönetet mondani témavezetőmnek, Dr. Gazdag-Tóth Boglárkának, aki észrevételeivel, tanácsaival, és lelkesedésével segítette a munkámat. Köszönöm neki a sok-sok konzultációt, közös munkát és tanulási lehetőséget, melyet az egész idő során biztosított.

Szeretném megköszönni Andrisnak a sok támogatást, biztatást és ötletet, melyek hozzájárultak ezen dolgozat elkészüléséhez.

Végül, de nem utolsó sorban köszönöm Vinkó Tamásnak és Gera Imrének, akiknek magyarázatai előrébb vittek a modellezés és optimalizálás témakörének megértésében.

A. függelék

Python kód random koordináták generálásához

```
import random
def koordinatak (n, m):
    vasarlo = [0, 0]
    leader = [0, 0]
    follower = [0, 0]
    for i in range(1, 6):
        vasarlo = [random.randint(1, n), random.randint(1, m)]
    for j in range(1, 3):
        leader = [random.randint(1, n), random.randint(1, m)]
    for k in range(1, 3):
        follower = [random.randint(1, n), random.randint(1, m)]
```

B. függelék

Modell szerinti példa feladat táblázata

v_1	l_1	f_1	$l_1 + f_1$	l_2	f_2	$l_2 + f_2$	$l_1 + f_2$	$l_2 + f_1$
r	20.00	18.00		20.00	18.00			
π	5.00	7.00		7.00	6.00			
g	3.00	5.10	2.24	4.12	7.81	4.47	6.32	5.00
u - egyszerű	9.00 ●	0.80 ●		4.75 ●	-3.62 ●			
u - többcélú			11.86 ●			4.79 ●	6.07 ●	5.98 ●
γ			0.90			0.90	0.90	0.90
v_2								
r	50.00	78.00		50.00	78.00			
π	5.00	7.00		7.00	6.00			
g	2.83	1.00	2.24	4.47	4.00	4.47	6.32	5.00
u - egyszerű	39.34 ●	69.00 ●		34.06 ●	64.00 ●			
u - többcélú			97.14 ●			89.26 ●	91.05 ●	90.73 ●
γ			0.90			0.90	0.90	0.90
v_3								
r	14.00	37.00		14.00	37.00			
π	5.00	7.00		7.00	6.00			
g	9.85	8.25	2.24	7.81	3.61	4.47	6.32	5.00
u - egyszerű	-10.70 ●	13.51 ●		-8.62 ●	23.79 ●			
u - többcélú			-13.57 ●			17.01 ●	15.12 ●	10.84 ●
γ			0.90			0.90	0.90	0.90

v_4	l_1	f_1	$l_1 + f_1$	l_2	f_2	$l_2 + f_2$	$l_1 + f_2$	$l_2 + f_1$
r	23.00	26.00		23.00	26.00			
π	5.00	7.00		7.00	6.00			
g	7.21	7.81	2.24	2.83	6.00	4.47	6.32	5.00
u - egyszerű	3.58 ●	3.38 ●		10.34 ●	8.00 ●			
u - többcélú			14.84 ●			17.80 ●	13.56 ●	14.46 ●
γ			0.90			0.90	0.90	0.90
v_5								
r	17.00	5.00		17.00	5.00			
π	10.00	6.00		9.00	6.00			
g	5.83	4.12	2.24	5.10	1.41	4.47	6.32	5.00
u - egyszerű	0.34 ●	-10.25 ●		-0.20 ●	-3.38 ●			
u - többcélú			-4.39 ●			-4.19 ●	-4.77 ●	-8.42 ●
γ			0.90			0.90	0.90	0.90

B.1. táblázat. Példa feladat számértékei a megfelelő jelölésekkel: ● jelöli a pozitív hasznosságot, ● jelöli a negatív hasznosságot.

A vásárlókat v_i index jelöli (de a halmazoknál alsó indexben sima természetes számok vannak a könnyebb olvashatóság kedvéért), amik most jelen esetben 1, ..., 5 között mozoghatnak, míg a vezető és a követő üzletek 1 vagy 2 indexűek lehetnek. Tehát egy értelmes jelölés például: v_1, l_1, f_1 .

C. függelék

Példa feladat halmazainak meghatározása

$$N_1^F = \{f_1\}$$

$$N_2^F = \{f_1, f_2\}$$

$$N_3^F = \{f_1, f_2\}$$

$$N_4^F = \{f_2\}$$

$$N_5^F = \{\}$$

$$N_1^L = \{l_1, l_2\}$$

$$N_2^L = \{l_1, l_2\}$$

$$N_3^L = \{\}$$

$$N_4^L = \{l_1, l_2\}$$

$$N_5^L = \{l_1\}$$

$$N_1^{LF} = \{(l_1, f_1), (l_1, f_2), (l_2, f_1), (l_2, f_2)\}$$

$$N_2^{LF} = \{(l_1, f_1), (l_2, f_2), (l_1, f_2), (l_2, f_1)\}$$

$$N_3^{LF} = \{(l_1, f_1), (l_2, f_2), (l_1, f_2), (l_2, f_1)\}$$

$$N_4^{LF} = \{(l_1, f_1), (l_2, f_2), (l_1, f_2), (l_2, f_1)\}$$

$$N_5^{LF} = \{\}$$

$$Q_{1l_1}^C = \{(l_1, f_1)\}$$

$$Q_{1l_2}^C = \{(l_1, f_1), (l_2, f_2), (l_1, f_2), (l_2, f_1)\}$$

$$Q_{1f_1}^C = \{(l_1, f_1), (l_1, f_2), (l_2, f_2), (l_2, f_1)\}$$

$$Q_{1f_2}^C = \{\}$$

$$Q_{2l_1}^C = \{(l_1, f_1)(l_1, f_2)(l_2, f_1)(l_2, f_2)\}$$

$$Q_{2l_2}^C = \{(l_1, f_1)(l_1, f_2)(l_2, f_1)(l_2, f_2)\}$$

$$Q_{2f_1}^C = \{(l_1, f_1)(l_1, f_2)(l_2, f_1)(l_2, f_2)\}$$

$$Q_{2f_2}^C = \{(l_1, f_1)(l_1, f_2)(l_2, f_1)(l_2, f_2)\}$$

$$Q_{3l_1}^C = \{\}$$

$$\begin{aligned}
Q_{3l_2}^C &= \{\} \\
Q_{3f_1}^C &= \{(l_1, f_1)(l_1, f_2)(l_2, f_2)\} \\
Q_{3f_2}^C &= \{\}
\end{aligned}$$

$$\begin{aligned}
Q_{4l_1}^C &= \{(l_1, f_1)(l_1, f_2)(l_2, f_1)(l_2, f_2)\} \\
Q_{4l_2}^C &= \{(l_1, f_1)(l_1, f_2)(l_2, f_1)(l_2, f_2)\} \\
Q_{4f_1}^C &= \{(l_1, f_1), (l_2, f_2), (l_1, f_2), (l_2, f_1)\} \\
Q_{4f_2}^C &= \{(l_1, f_1)(l_1, f_2)(l_2, f_1)(l_2, f_2)\}
\end{aligned}$$

$$\begin{aligned}
Q_{5l_1}^C &= \{\} \\
Q_{5l_2}^C &= \{\} \\
Q_{5f_1}^C &= \{\} \\
Q_{5f_2}^C &= \{\}
\end{aligned}$$

$$\begin{aligned}
Q_{1l_1f_1}^C &= \{\} \\
Q_{1l_2f_2}^C &= \{\} \\
Q_{1l_1f_2}^C &= \{\} \\
Q_{1l_2f_1}^C &= \{\}
\end{aligned}$$

$$\begin{aligned}
Q_{2l_1f_1}^C &= \{\} \\
Q_{2l_2f_2}^C &= \{\} \\
Q_{2l_1f_2}^C &= \{\} \\
Q_{2l_2f_1}^C &= \{\}
\end{aligned}$$

$$\begin{aligned}
Q_{3l_1f_1}^C &= \{f_2\} \\
Q_{3l_2f_2}^C &= \{f_2\} \\
Q_{3l_1f_2}^C &= \{f_2\} \\
Q_{3l_2f_1}^C &= \{f_1, f_2\}
\end{aligned}$$

$$\begin{aligned}
Q_{4l_1f_1}^C &= \{\} \\
Q_{4l_2f_2}^C &= \{\} \\
Q_{4l_1f_2}^C &= \{\} \\
Q_{4l_2f_1}^C &= \{\}
\end{aligned}$$

$$\begin{aligned}
Q_{5l_1f_1}^C &= \{\} \\
Q_{5l_2f_2}^C &= \{l_1\} \\
Q_{5l_1f_2}^C &= \{l_1\} \\
Q_{5l_2f_1}^C &= \{l_1\}
\end{aligned}$$

$$R_{1l_1}^L = \{\}$$

$$R_{1l_2}^L = \{\}$$

$$R_{2l_1}^L = \{f_1, f_2\}$$

$$R_{2l_2}^L = \{f_1, f_2\}$$

$$R_{3l_1}^L = \{\}$$

$$R_{3l_2}^L = \{\}$$

$$R_{4l_1}^L = \{f_2\}$$

$$R_{4l_2}^L = \{\}$$

$$R_{5l_1}^L = \{\}$$

$$R_{5l_2}^L = \{\}$$

$$R_{1f_1}^F = \{l_1, l_2\}$$

$$R_{1f_2}^F = \{\}$$

$$R_{2f_1}^F = \{\}$$

$$R_{2f_2}^F = \{\}$$

$$R_{3f_1}^F = \{\}$$

$$R_{3f_2}^F = \{\}$$

$$R_{4f_1}^F = \{l_1, l_2\}$$

$$R_{4f_2}^F = \{l_2\}$$

$$R_{5f_1}^F = \{\}$$

$$R_{5f_2}^F = \{\}$$

D. függelék

Példa feladathoz tartozó AMPL modell és magyarázata

A modell felírásánál szükségünk van a halmazok, paraméterek és változók meghatározására. Jelen esetben a halmazok meghatározása nem annyira triviális, mert összetett halmazokkal is kell számolnunk. Így a legelején kezdve, nézzük először az egyszerűen meghatározható halmazokat, melyeknél a legkönnyebb úgy gondolkozni, hogy hány különböző indexünk van a modellben.

```
set M_L; # vezető üzlet lehetséges helyeinek halmaza
set M_F; # követő üzlet lehetséges helyei halmaza
set Customers # vásárlók halmaza;
```

Mindössze ezen három halmaz segítségével az összes paraméter megadható, ahol a különböző névválasztások követik a dolgozatomban használt jelölésrendszert, ezért a könnyebb olvashatóság kedvéért a kommentezéstől tekintsünk el minden olyan esetben, ha az a jelölés egyértelmű.

```
param gamma >=0;
param a{Customers} integer;
param pi_L{M_L};
param pi_F{M_F};
param n_L integer;
param n_F integer;
param r_L{Customers};
param r_F{Customers};
```

Ahhoz, hogy tovább haladhassunk, ezen a ponton meg kell határoznunk, hogy a *.dat* fájlban milyen adatokat szeretnénk beleírni. A legjobb megoldás az lenne, hogyha semmilyen előszámítást nem kellene külön végeznünk, hanem ezt mind a *.mod* fájlban az AMPL kiszámolná nekünk. Ilyen számolás paraméter például a hasznosság. Ammihez szükségünk van a különböző távolságokra. Legyen a modell síkban értelmezve, természetesen lehetne térben is definiálva, és akkor is ugyanúgy működne minden, hiszen a dimenzió csak a hasznosság értéket befolyásolja. Definiáljuk a távolságot a következő koordináták segítségével:

```
param xy_C{Customers,1..2}; # egyes vásárlók koordinátái
param xy_L{M_L, 1..2}; # vezető üzletek koordinátái
```

```
param xy_F[M_F, 1..2]; # koveto uzletek koordinatai
```

Ezután a távolság könnyen kiszámolható, minden lehetséges két pont között, aminek van értelme a modell szempontjából (például: két vásárló között nincs értelme távolságot számolni, mert egyik vásárló a másik vásárlóhoz nem utazik olyan céllal, hogy vegyen tőle valamit):

```
param g_L{i in Customers, j in M_L} :=
sqrt((xy_C[i,1]-xy_L[j,1])^2 + (xy_C[i,2]-xy_L[j,2])^2);
# tavolsag vezető üzlet és vásárló között
param g_F{i in Customers, j in M_F} :=
sqrt((xy_C[i,1]-xy_F[j,1])^2 + (xy_C[i,2]-xy_F[j,2])^2);
# tavolsag követő üzlet és vásárló között
param g_LF{j in M_L, k in M_F}:=
sqrt((xy_L[j,1]-xy_F[k,1])^2 + (xy_L[j,2]-xy_F[k,2])^2);
# tavolsag vezető üzlet és követő üzlet között
```

Ezen plusz paraméterek létrehozása után már könnyű dolgunk van a hasznosságok meghatározásával:

```
param u_L{i in Customers, j in M_L} = r_L[i] - pi_L[j] - (g_L[i, j]*2);
param u_F{i in Customers, k in M_F} = r_F[i] - pi_F[k] - (g_F[i, k]*2);
param u_LF{i in Customers, j in M_L, k in M_F} = gamma*( r_L[i] + r_F[i]) - (pi_L[j] + pi_F[k]) -
(g_L[i, j] + g_F[i, k] + g_LF[j, k]);
```

További paraméterre nincs szükség, így következhetnek az összetettebb halmazok is, melyek mind a hasznosság értékeket használják fel.

```
set Q_L{i in Customers, j in N_L[i]} = {l in M_L, k in M_F :
u_LF[i, l, k] >= u_L[i, j] };
set Q_F{i in Customers, k in N_F[i]} = {j in M_L, o in M_F :
u_LF[i, j, o] >= u_F[i, k] };
set Q_LFL{i in Customers, (j,k) in N_LF[i]} = {l in M_L :
u_LF[i, j, k] < u_L[i, l]};
set Q_LFF{i in Customers, (j,k) in N_LF[i]} = {o in M_F :
u_LF[i, j, k] < u_F[i, o]};
set R_L{i in Customers, j in N_L[i]} = { k in M_F :
u_F[i, k] > u_L[i, j]};
set R_F{i in Customers, k in N_F[i]} = {j in M_L :
u_F[i, k] <= u_L[i, j]};
```

Miután minden halmaz és minden paraméter definiálásra került, jöhetnek a változók, melyek bináris voltát a *binary* kulcsszó jelöli.

```
var x_L[M_L] binary;
var x_F[M_F] binary;
var x_LF[M_L, M_F] binary;

var y_L[Customers] binary;
var y_F[Customers] binary;
var y_LF[Customers] binary;
```

A következő feladat a célfüggvények és a feltételek felírása, amihez tekintsük a következő táblázatban egymásnak megfeleltetett jelöléseket: Ezek alapján az AMPL-ben a célfüggvények és a feltételek felírása a következőképpen alakult:

feltétel sorszáma a modellben	feltétel neve az AMPL-ben
(3)	leader
(4)	numberleader
(6)	follower
(7)	numberfollower
(8)	numbermulti1
(9)	numbermulti2
(10)	numbermulti3
(11)	customer1
(12)	customer2
(13)	customer3
(14)	customerlimit
(15)	customerlimitwithshops1
(16)	customerlimitwithshops2
(17)	customerlimitwithshops3

```

maximize leader: sum{i in Customers} (a[i]*(y_L[i] + y_LF[i]));
subject to numberleader: sum{j in M_L} x_L[j] <= n_L;
maximize follower: sum{i in Customers} (a[i]*(y_F[i] + y_LF[i]));
subject to numberfollower: sum{k in M_F} x_F[k] <= n_F;
subject to numbermulti1{j in M_L, k in M_F}: x_LF[j, k] <= x_L[j];
subject to numbermulti2{j in M_L, k in M_F}: x_LF[j, k] <= x_F[k];
subject to numbermulti3{j in M_L, k in M_F}:
x_LF[j, k] >= x_L[j] + x_F[k] - 1;
subject to customer1{i in Customers, j in N_L[i]}:
y_L[i] >= x_L[j] - sum{k in R_L[i,j]} x_F[k] -
sum{(j,k) in Q_L[i,j]} x_LF[j, k];
subject to customer2{i in Customers, k in N_F[i]}:
y_F[i] >= x_F[k] - sum{j in R_F[i,k]} x_L[j] -
sum{(j,k) in Q_F[i,k]} x_LF[j, k];
subject to customer3{i in Customers, (j,k) in N_LF[i]}:
y_LF[i] >= x_LF[j,k] - sum{n in Q_LFL[i,j,k]} x_L[n] -
sum{m in Q_LFF[i,j,k]} x_F[m];
subject to customerlimit{i in Customers}:
y_L[i] + y_F[i] + y_LF[i] <= 1;
subject to customerlimitwithshops1{i in Customers}:
y_L[i] <= sum{j in N_L[i]} x_L[j];
subject to customerlimitwithshops2{i in Customers}:
y_F[i] <= sum{k in N_F[i]} x_F[k];
subject to customerlimitwithshops3{i in Customers}:
y_LF[i] <= sum{(j,k) in N_LF[i]} x_LF[j,k];

```

Így készen is vagyunk a *.mod* fájl kitöltésével, ellenőrzésképpen lefutathatjuk a

```
model bilevel.mod
```

parancsot, ami segít az esetleges szintaktikai hibák kijavításában.

A következő feladat az adat fájl meghatározása, ami feladatonként eltér, attól függően, hogy hány vezető és követő üzletet szeretnénk elhelyezni. Példának most vegyük az előző fejezetben kiszámolt példa feladat adatait. Az adat fájl kitöltésekor szükségünk van minden olyan halmaznak és paraméternek értéket adni, aminek nem adtunk már vagy nem számoláson alapszik a *.mod* fájlban.

```
param n_L:= 1;
param n_F:= 1;
set M_L:= l_1 l_2;
set M_F:= f_1 f_2;
set Customers:= v_1 v_2 v_3 v_4 v_5;
param gamma:= 0.9;
param a:=
v_1 36
v_2 19
v_3 53
v_4 47
v_5 28
;
param: r_L r_F:=
v_1 20 18
v_2 50 78
v_3 14 37
v_4 23 26
v_5 17 5
;
param: pi_L:=
l_1 5
l_2 7
;
param: pi_F:=
f_1 7
f_2 6
;
param xy_C: 1 2:=
v_1 3 1
v_2 8 3
v_3 10 10
v_4 2 7
v_5 9 6
;
param xy_L: 1 2:=
l_1 6 1
l_2 4 5
;
param xy_F: 1 2:=
f_1 8 2
f_2 8 7
;
```

Ennek tesztelésére használhatjuk a

data bilevel.dat

parancsot. Ennek segítségével láthatjuk, hogy valóban mindennek adtunk-e értéket, aminek elterveztük, hogy fogunk a *.mod* fájlban, valamint szintaktikailag ellenőrizni tudjuk, hogy nem írtunk-e el valamit. Az előző és ezen parancs többszöri futtatásánál szükség lehet a

```
reset;
```

parancsra is, ami a paraméterek és halmazok értékét alaphelyzetbe állítja. Ezen parancs elhagyása sok galibát okozhat, tehát érdemes minden javítás után használni. A következő, és egyben az utolsó feladat a *.run* fájl megírása, ahol rájöhethetünk, hogy valóban jól definiáltuk-e a halmazainkat és feltételeinket. Az AMPL-lel beépített solver-eket tudunk használni, ami azért is nagyon jó, mert így a legtöbb esetben nincs arra szükségünk, hogy implementáljunk egy már ismert módszert vagy kitaláljunk sajátot. Ebben az esetben a *.run* fájl így néz ki:

```
reset;
model bilevel.mod;
data bilevel.dat;
option solver cplex;
solve;
```

Gyakorlatilag az történik mint eddig, csak itt egy fájlban belül és ezáltal egy parancsral futtatva az egészet már meg is kapjuk az eredményt. Az *option* sorában tudjuk kiválasztani a megoldónkat. Jelen esetben ez a *CPLEX* megoldó, ami alkalmas továbbá LP és MILP programozási feladatok megoldására is.

```
include bilevel.run
```

Futtatva ezt a parancsot a következő eredményt kapjuk:

```
AMPL: include temp.run
CPLEX 20.1.0.0: optimal integer solution; objective 183
5 MIP simplex iterations
0 branch-and-bound nodes
Objective = leader
:   x_L x_F   :=
f_1   .   1
f_2   .   0
l_1   1   .
l_2   0   .
;

x_LF :=
l_1 f_1   1
l_1 f_2   0
l_2 f_1   0
l_2 f_2   0
;

:   y_L y_F y_LF   :=
v_1   0   0   1
v_2   0   0   1
```

```

v_3    0    0    1
v_4    0    0    1
v_5    1    0    0
;

```

Ezután vizsgáljuk meg azon *.run* fájlunkat, ahol a garantáljuk a többszintű modell jó megoldását:

```

reset;

model bilevel.mod;
data bilevel.dat;
option presolve 0;
option solver cplex;

problem Leader: x_L, x_F, x_LF, y_L, y_F, y_LF, leader, numberleader;
problem Follower: x_L, x_F, x_LF, y_L, y_F, y_LF, numberleader,
follower, numberfollower, numbermulti1, numbermulti2,
numbermulti3, customerlimit, customerlimitwithshops1,
customerlimitwithshops2, customerlimitwithshops3, customer1,
customer2, customer3;

param LB default -10000;
param l symbolic;
param f symbolic;
param saved;
for {i in M_L}{
    fix x_L[i] := 1;
    solve Follower;
    let saved := leader;
    if saved > LB then {
        let LB := saved;
        let l := i;
        for {j in M_F}{
            if x_F[j] == 1 then let f := j;
        }
    }
    unfix x_L[i];
}
fix x_L[l] := 1;
fix x_F[f] := 1;
solve Follower;
display follower, leader, x_L, x_F, x_LF, y_L, y_F, y_LF, LB;

```

Elemezzük ezt sorrol sorra. Először ami új az előző *.run* fájlhoz képest azaz, hogy egyes solver-eknél lehetőség van bizonyos plusz feltételeket megadására, hogy mit csinálnának, amit alapértelmezett nem csinálnának. Például

```

option presolve 0;

```

parancs szabályozza az AMPL előmegoldó fázisát, ami abban segít, hogy a probléma méretét csökkentse, kis példánál már akár itt is megoldásra kerülhet a probléma. Ezt

a 0 értékkel kikapcsoltuk. A *problem* kulcsszó segít abban, hogy akár egy többszintű feladatot definiálni lehessen, hiszen szétválaszthatjuk egy fájlban belül, hogy melyik célfüggvényekhez melyik feltételek és változók tartozzanak. Ezután szükség van néhány plusz változóra is, amik a *.run* fájlban globális változók lesznek: *LB*, *l*, *f*, *saved*. *LB* jelöli az egyes iterációban megtalált alsó korlátot a vezető cég számára, azaz bizonyos vezető üzlet mellett a követő vállalat optimumát, ami egy értelmes alsó korlát lesz. Minden iterációban ellenőriznünk kell, hogy egy már megtalált *LB*-nél nincs-e jobb alsó korlát (nagyobb), ha igen akkor azt elmentjük a *saved* változóba, majd a hozzátartozó x_F és x_L változók értékét is elmentjük *f* illetve *l*-ként. Ez azért fontos, hogy miután végig iteráltunk a lehetőségeinken, utána tényleg az optimumot kapjuk meg. Mivel az én *.dat* fájlomban az üzleteket betűkkel adtam meg, ezért szükség van még a *symbolic* kulcsszóra is. Lefuttatva ezt a következőt kapjuk:

```
include bilevel.run

CPLEX 20.1.0.0: optimal integer solution; objective 155
0 MIP simplex iterations
0 branch-and-bound nodes
follower = 155
leader = 183

:   x_L x_F   :=
f_1   .   1
f_2   .   0
l_1   1   .
l_2   0   .
;

x_LF :=
l_1 f_1   1
l_1 f_2   0
l_2 f_1   0
l_2 f_2   0
;

:   y_L y_F y_LF   :=
v_1   0   0   1
v_2   0   0   1
v_3   0   0   1
v_4   0   0   1
v_5   1   0   0
;

LB = 183
```

E. függelék

A módosított modell AMPL modellje

Paraméterekből a következőket adjuk hozzá a modellhez:

```
param u_L_modified{i in Customers, j in M_L} =  
if u_L[i,j] <= 0 then 0 else u_L[i,j];  
param u_F_modified{i in Customers, k in M_F} =  
if u_F[i,k] <= 0 then 0 else u_F[i,k];  
param u_LF_modified{i in Customers, j in M_L, k in M_F} =  
if u_LF[i,j,k] <= 0 then 0 else u_LF[i,j,k];
```

melyek segítenek abban, hogy negatív hasznosság ne kerüljön a feltételek közé. Változók közül a következők kerültek definiálásra:

```
# 11-es feltételhez  
var w_LL{i in Customers, j in M_L} >= 0;  
var w_LF{i in Customers, k in M_F} >= 0;  
var w_LLF{i in Customers, (j,k) in {M_L, M_F}} >= 0;  
# 12-es feltételhez  
var w_FL{i in Customers, j in M_L} >= 0;  
var w_FF{i in Customers, k in M_F} >= 0;  
var w_FLF{i in Customers, (j,k) in {M_L, M_F}} >= 0;  
# 13-as feltételhez  
var w_LFL{i in Customers, j in M_L} >= 0;  
var w_LFF{i in Customers, k in M_F} >= 0;  
var w_LFLF{i in Customers, (j,k) in {M_L, M_F}} >= 0;
```

Majd az ezekhez tartozó korlátozó feltételek:

```
subject to morethan13 {i in Customers, j in M_L}:  
w_LL[i,j] >= y_L[i] - (1 - x_L[j]);  
subject to morethan14 {i in Customers, j in M_L}:  
w_LFL[i,j] >= y_LF[i] - (1 - x_L[j]);  
subject to morethan15 {i in Customers, j in M_L}:  
w_FL[i,j] >= y_F[i] - (1 - x_L[j]);  
  
subject to morethan23 {i in Customers, k in M_F}:  
w_LF[i,k] >= y_L[i] - (1 - x_F[k]);  
subject to morethan24 {i in Customers, k in M_F}:  
w_LFF[i,k] >= y_LF[i] - (1 - x_F[k]);  
subject to morethan25 {i in Customers, k in M_F}:  
w_FF[i,k] >= y_F[i] - (1 - x_F[k]);
```

```

subject to morethan33 {i in Customers,j in M_L, k in M_F}:
w_LLF[i,j,k] >= y_L[i] - (1 - x_LF[j,k]);
subject to morethan34 {i in Customers,j in M_L, k in M_F}:
w_FLF[i,j,k] >= y_F[i] - (1 - x_LF[j,k]);
subject to morethan35 {i in Customers,j in M_L, k in M_F}:
w_LFLF[i,j,k] >= y_LF[i] - (1 - x_LF[j,k]);

```

```

subject to lessthan4 {i in Customers, j in M_L}:
w_LL[i,j] <= y_L[i];
subject to lessthan5 {i in Customers, j in M_L}:
w_LL[i,j] <= x_L[j];
subject to lessthan6 {i in Customers, j in M_L}:
w_LFL[i,j] <= y_LF[i];
subject to lessthan7 {i in Customers, j in M_L}:
w_LFL[i,j] <= x_L[j];
subject to lessthan8 {i in Customers, j in M_L}:
w_FL[i,j] <= y_F[i];
subject to lessthan9 {i in Customers, j in M_L}:
w_FL[i,j] <= x_L[j];

```

```

subject to lessthan10 {i in Customers, k in M_F}:
w_LF[i,k] <= y_L[i];
subject to lessthan11 {i in Customers, k in M_F}:
w_LF[i,k] <= x_F[k];
subject to lessthan12 {i in Customers, k in M_F}:
w_LFF[i,k] <= y_LF[i];
subject to lessthan13 {i in Customers, k in M_F}:
w_LFF[i,k] <= x_F[k];
subject to lessthan14 {i in Customers, k in M_F}:
w_FF[i,k] <= y_F[i];
subject to lessthan15 {i in Customers, k in M_F}:
w_FF[i,k] <= x_F[k];

```

```

subject to lessthan16 {i in Customers,j in M_L, k in M_F}:
w_LLF[i,j,k] <= y_L[i];
subject to lessthan17 {i in Customers,j in M_L, k in M_F}:
w_LLF[i,j,k] <= x_LF[j,k];
subject to lessthan18 {i in Customers,j in M_L, k in M_F}:
w_FLF[i,j,k] <= y_F[i];
subject to lessthan19 {i in Customers,j in M_L, k in M_F}:
w_FLF[i,j,k] <= x_LF[j,k];
subject to lessthan20 {i in Customers,j in M_L, k in M_F}:
w_LFLF[i,j,k] <= y_LF[i];
subject to lessthan21 {i in Customers,j in M_L, k in M_F}:
w_LFLF[i,j,k] <= x_LF[j,k];

```

Ezekefenélül még a három további feltétel:

```

subject to customer1 {i in Customers}:
    sum{j in M_L} u_L_modified[i,j]*x_L[j] =
    sum{j in M_L} u_L_modified[i,j]*w_LL[i,j] +
    sum{k in M_F} u_F_modified[i,k] * w_LF[i,k] +
    sum{j in M_L, k in M_F} u_LF_modified[i,j,k] * w_LLF[i,j,k];

```

```

subject to customer2 {i in Customers}:
    sum{k in M_F} u_F_modified[i,k]*x_F[k] =
    sum{j in M_L} u_L_modified[i,j]*w_FL[i,j] +
    sum{k in M_F} u_F_modified[i,k] * w_FF[i,k] +
    sum{j in M_L, k in M_F} u_LF_modified[i,j,k] * w_FLF[i,j,k];

```

```

subject to customer3 {i in Customers}:
    sum{k in M_F, j in M_L} u_LF_modified[i,j,k]*x_LF[j,k] =
    sum{j in M_L} u_L_modified[i,j]*w_LFL[i,j] +
    sum{k in M_F} u_F_modified[i,k] * w_LFF[i,k] +
    sum{j in M_L, k in M_F} u_LF_modified[i,j,k] * w_LFLF[i,j,k];

```

Valamint még eltörölhető a *customerlimit* feltétel, ami azt garantálta, hogy bináris esetben csak egy vásárlási típust választunk. Majd ezeket az új feltételeket jelezni kell a *.run* fájlban is, ahol mindkét *problem*-hez felvételre kerülnek a *w* változók, és a többi új feltétel a *problem Follower*-hez íródik. Továbbá a modellben nincsenek azok a kikötések leírva, amikor 0-val való osztás történne, így azt a *.run* fájlba kell beletennünk:

```

for{i in Customers: max(max{ k in M_F} u_F_modified[i,k],
max{j in M_L} u_L_modified[i,j], max{j in M_L, k in M_F}
u_LF_modified[i,j,k]) == 0 }{
    fix y_F[i] := 0;
    fix y_L[i] := 0;
    fix y_LF[i] := 0;
}

```

Ekkor futtatva a *modositott.run* fájlt (ahol *.dat* fájl nem változott az előző modellhez képest) a következő megoldást kapjuk:

```

CPLEX 20.1.0.0: optimal integer solution; objective 129.9030248
0 MIP simplex iterations
0 branch-and-bound nodes
follower = 129.903
leader = 143.17

```

```

:   x_L x_F   :=
f_1   .   1
f_2   .   0
l_1   1   .
l_2   0   .
;

```

```

x_LF :=
l_1 f_1   1
l_1 f_2   0
l_2 f_1   0
l_2 f_2   0
;

```

```

:       y_L       y_F       y_LF       :=
v_1    0.393582    0.0350709    0.571347

```

```
v_2  0.190359  0.333852  0.475789
v_3  0          0.477697  0.522303
v_4  0.155557  0.146936  0.697507
v_5  1          0          0
;
```

```
LB = 143.17
```

F. függelék

Brute-force algoritmus

```
reset;
model bruteforce.mod;
data bruteforce.dat;
option solver_msg 0;
option solver cplex;

problem Leader: x_L, x_F, x_LF, y_L, y_F, y_LF, leader,
numberleader, w_LL, w_LF, w_FL, w_LFL, w_LFF, w_FLF, w_LFLF,
w_FF, w_LLF;
problem Follower: x_L, x_F, x_LF, y_L, y_F, y_LF, numberleader,
follower, numberfollower, numbermulti1, customerlimit,
numbermulti2, numbermulti3, customerlimitwithshops1,
customerlimitwithshops2, customerlimitwithshops3, customer1,
customer2, customer3,
morethan13, morethan14, morethan15,
morethan23, morethan24, morethan25,
morethan33, morethan34, morethan35,
lessthan4, lessthan5, lessthan6, lessthan7, lessthan8,
lessthan9, lessthan10,
lessthan11, lessthan12, lessthan13, lessthan14, lessthan15,
lessthan16, lessthan17,
lessthan18, lessthan19, lessthan20, lessthan21,
w_LL, w_LF, w_FL, w_LFL, w_LFF, w_FLF, w_LFLF, w_FF, w_LLF;
problem old_Follower: x_L, x_F, x_LF, y_L, y_F, y_LF,
numberleader, follower, numberfollower, numbermulti1,
numbermulti2, numbermulti3, customerlimit, customerlimitwithshops1,
customerlimitwithshops2, customerlimitwithshops3, customer1old,
customer2old, customer3old;

param LB default -10000;
set chosen_leader default {};
set chosen_follower default {};
param saved;
param old_model_time;
param new_model_time;
param ampl_start_time;
# old
let ampl_start_time := _ampl_time;
```

```

for {i in 1..subtours}{
  if card(S[i])!=n_L then continue;
  fix {j in S[i]} x_L[j] := 1;
  solve old_Follower;
  let saved := leader;
  if saved > LB then {
    let LB := saved;
    let chosen_leader := {k in M_L: x_L[k] == 1};
    let chosen_follower := {k in M_F: x_F[k] == 1};
  }
  for {j in S[i]} unfix x_L[j];
}
for {i in chosen_leader} fix x_L[i] := 1;
for {j in chosen_follower} fix x_F[j] := 1;
solve old_Follower;
let old_model_time := _ampl_elapsed_time +
_total_solve_elapsed_time;

# modified
let ampl_start_time := _ampl_time;
param LB_UJ default -10000;
for{i in Customers: max(max{ k in M_F} u_F_modified[i,k],
max{j in M_L} u_L_modified[i,j], max{j in M_L,k in M_F}
u_LF_modified[i,j,k]) == 0 }{
  fix y_F[i] := 0;
  fix y_L[i] := 0;
  fix y_LF[i] := 0;
}
for {i in 1..subtours}{
  if card(S[i])!=n_L then continue;
  fix {j in S[i]} x_L[j] := 1;
  solve Follower;
  let saved := leader;
  if saved > LB_UJ then {
    let LB_UJ := saved;
    let chosen_leader := {k in M_L: x_L[k] == 1};
    let chosen_follower := {k in M_F: x_F[k] == 1};
  }
  for {j in S[i]} unfix x_L[j];
}
for {i in chosen_leader} fix x_L[i] := 1;
for {j in chosen_follower} fix x_F[j] := 1;
solve Follower;
let new_model_time:= _ampl_elapsed_time +
_total_solve_elapsed_time - old_model_time;

```