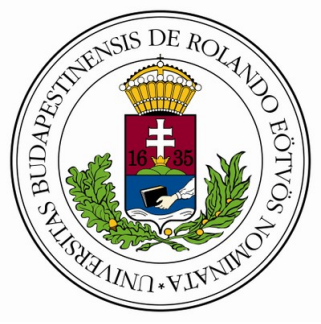
Eötvös Loránd Tudományegyetem
Természettudományi Kar

Szállításszervezési algoritmusok

Szabó Eszter
Matematikus BSc

Témavezető:
Jüttner Alpár
Tudományos főmunkatárs
Operációkutatási tanszék



Budapest, 2019

Köszönetnyilvánítás

Ezúton szeretném megköszönni a rengeteg segítséget a témavezetőmnek, Jüttner Alpárnak, aki szakmailag és emberileg is sokat segített.
Köszönet illeti egyetemi tanáriamat, akik felkeltették a figyelmemet sok érdekes témakör, köztük az operációkutatás iránt. Köszönettel tartozom továbbá, nem utolsó sorban a családomnak, akik mindig támogattak és mellettem álltak.

Tartalomjegyzék

1. Bevezetés **1**

2. Utazó ügynök probléma variációi **2**

2.1. IP felírása 2

2.2. Szimmetrikus utazóügynök feladat 4

2.2.1. Feltétel azonosítási probléma 9

2.2.2. A sértett feltételek azonsoító algoritmusok 12

3. Járműirányítási probléma **16**

3.1. Kapacitásos járműirányítási probléma 16

3.1.1. A probléma felírásai 16

3.2. Járműirányítási és ügyfélelhelyezési probléma 20

3.3. Járműirányítási és elosztási probléma 21

3.3.1. Járműirányítási és elosztási probléma felírása 21

4. Járműirányítási és ügyfélelhelyezési probléma megoldása **24**

4.1. Jelölések 24

4.2. Algoritmus 24

4.3. Kezdeti megoldás 25

4.4. Oszlopgenerálás 26

4.5. Szétválasztási stratégia 28

4.6. Árképzési alprobléma 30

4.6.1. Az árképzési alproblémában használt jelölések 30

4.6.2. Az árképzési alprobléma felírása 31

4.7. Dinamikus programozás az árképzési alproblémára 32

4.7.1. Jelölések a címkéző algoritmushoz 33

4.7.2. Címkéző algoritmus leírása 34

4.7.3. Ügyfélrészhalmazok készítése 36

4.7.4. Heurisztikus algoritmusok az árképzési alproblémára 38

5. Összefoglalás **41**

6. Hivatkozások **42**

1. Bevezetés

Szállításszervezési problémákkal rengeteg helyen találkozhatunk a hétköznapokban. Ezekben a többnyire logisztikai problémákban adott ügyfeleket kell meglátogatnunk, amikhez rendelkezésünkre áll egy vagy esetleg több jármű (például: teherautó). Ilyen feladat fogalmazódhat meg például webshopok áruinak házhozszállításainak az optimalizálásakor vagy egy szerelőket foglalkoztató cégnek az elromlott készülékekhez való kirendeléseinek az optimalizálásakor.

A matematikában is egy széles problémakör alakult ki ezen a területen. A feladatok változatosak, de a célja mindegyikben megegyezik. Adott feltételeknek eleget tevő optimális útvonalat kell megadni egy vagy több jármű számára. Az optimális alatt a lehető legkisebb költségű útvonalat értjük.

A legismertebb ezek között az utazóügynök probléma (Traveling Salesman Proble) és annak változatai. Ennek általánosításaként tekinthetünk többféle szállításszervezési problémát is, amikor egynél több jármű áll a rendelkezésünkre a kiszállításokhoz. Ezeket a problémákat a szakirodalom angolul "Vehicle Routing Problem"-nek nevezi. Az utazóügynök problémának ez a fajta általánosítása a [4] cikkben jelent meg először. Ebben a szerzők a benzin benzinkutakra való szállításának optimalizálásával foglalkoztak.

Ezek a problémák NP-nehéznek bizonyultak, mivel visszavezethetőek az utazóügynök feladatra, ami bizonyítottan NP-nehéz probléma. Ezzel szemben rengeteg közelítő és pontos algoritmus készült a problémakör egyes feldatainak megoldására. Ezek között szerepelnek például különböző heurisztikus, metaheurisztikus algoritmusok vagy a probléma IP felírásán alapuló algoritmusok is. A problémákra különböző egészértékű programozási felírásokat készítettek az évek alatt.

A szakdolgozatomban ezeknek a porblémáknak az egészértékű programozási feladatként való felírásai közül fogok bemutatni néhányat. Majd ezek megoldására a korlátozás és szétválasztás módszerén (branch and bound) alapuló algoritmusokat választottam ki. Igyekeztem a korlátozás és szétválasztás módszerének különböző alkalmazásait összegyűjteni. Ezek közül egyet részletesen ismertetek a 4. fejezetben.

2. Utazó ügynök probléma variációi

Az utazó ügynök probléma (angolul: Traveling Salesman Problem) egy elterjedt kombinatorikus optimalizálási probléma. Több változata is ismert, de az alapprobléma mindegyiknél megegyezik:
Adott n város és adottak a városok közötti távolságok (ha az egyik városból nem lehet eljutni egy másikba, akkor megadhatunk ∞ vagy kellően nagy értéket). A feladat, hogy a legrövidebb olyan körutat találjuk meg, ami minden várost pontosan egyszer érint és az út első és utolsó városa megegyezik.
Az utazóügynök probléma visszavezethető arra a problémára, hogy egy G gráfban található-e Hamilton-kör. Ekkor készítsünk egy G' teljes gráfot, amiben a G gráf élei legyenek 0 súlyúak, a G-ben nem szereplő élek pedig legyenek 1 súlyú. Ekkor a Hamilton-kör keresése a G gráfban egyenértékű annak az eldöntésével, hogy a G' gráfban van 0 összsúlyú kör. Ezt az utazóügynök probléma megoldásával el lehetne dönteni, mivel ha a kapott minimális kör súlya 0, akkor a G gráfban található Hamilton-kör. Ha pedig a G'-ből kapott minimális súlyú kör súlya nagyobb, mint 0, akkor a G gráfban nincs Hamilton-kör. A Hamilton-kör probléma NP-nehéz, ezért az utazóügynök probléma is NP-nehéz.

A problémára több közelítő algoritmus is készült az évek folyamán. Ezek közül a legismertebb Christofides $\frac{3}{2}$-approximáló algoritmusa [2].
M. Karpinski, M. Lampis és R. Schmied által készített $\frac{123}{122}$-közelítő algoritmus az utóbbi évek egyik legjobb közelítése, amit [9]-ben írtak le.
L. Engebretsen és M. Karpinski az utazóügynök probléma egy speciális alakját vizsgálta, amiben az élek súlyai egy és kettő közé esnek. A [5] cikkükben bebizonyították, hogy ha erre a probléma létezik $\frac{741}{740} - \epsilon$ közelítő algoritmus $\forall \epsilon > 0$-ra, akkor $P = NP$, vagyis ilyen algoritmus találása NP-nehéz feladat.

2.1. IP felírása

Az utazóügynök problémáját többféleképpen is felírhatjuk egészértékű programozási feladatként. Az egyik lehetséges felíráshoz vezessük be a következő jelöléseket:

$$x_{ij} = \begin{cases} 1, \text{ ha az i. városból a j. városba megy az ügynök} \\ 0, \text{ egyébként} \end{cases}$$

$c_{ij} =$ az i. városból a j. városba menő út hossza

Ezekkel a jelölésekkel a probléma a következőképpen írható fel:

$$\begin{aligned}
&\min \sum_{i=1}^{n}\sum_{j=1}^{n} x_{ij}c_{ij} \\
&\sum_{i=1,i\neq j}^{n} x_{ij} = 1 && j = 1,2,3,...,n \\
&\sum_{j=1,i\neq j}^{n} x_{ij} = 1 && i = 1,2,3,...,n \\
&u_i - u_j - (n-1)x_{ji} \leq n-2 && 1 \leq i \neq j \leq n \\
&u_i \in \mathbb{Z}, x_{ij} \in (0,1) && i,j = 1,2,3,...,n
\end{aligned}$$

Ez az egészértékű programozási feladat megoldható a korlátozás és szétválasztás módszerével (angolul: branch and bound). A módszer lényege, hogy a feladatot részfeladatokra bontjuk szét. Ez olyan módon történik, hogy az eredeti feladat megengedett megoldásainak halmaza a szétbontott részfeladatok megengedett megoldásainak diszjunkt uniója legyen. Így a részfeladatok optimum értékeinek maximuma egyenlő lesz az eredeti feladat optimum értékével. Természetesen, ha szükséges, a részfeladatokat is további részfeladatokra bonthatjuk szét.

Az egyes részfeladatoknál alsó (és esetleg felső korlátot is valamilyen módon) számolunk az optimum értékére. Ha egy részfeladat alsó korlátja nagyobb egy másik már megoldott részfeladat felső korlátjánál, akkor ezzel a részfaladattal a továbbiakban nem kell foglalkoznunk.
Az alsó korlátokat számolhatjuk a részfeladatok LP-relaxáltjának megoldásával, a szétválasztást (részfeladatokra bontást) pedig végezhetjük lineáris egyenlőtlenségek hozzáadásával. Ebben az esetben LP-alapú korlátozás és szétválasztásról beszélünk. A korlátozás és szétválasztás módszerét az 1. algoritmus írja le.

Algorithm 1 Korlátozás és szétválasztás

Jelölje S_0 az eredeti feladatot és jelölje u(S) egy S (rész)feladat LP-relaxáltjának optimum értékét. Legyen L a részfeladatok listája.
Tegyük S_0-t az L listára és számoljunk hozzá valamilyen módon egy c(L) felső korlátot.
while Az L nem üres **do**
    Az L listáról válasszunk egy S részfeladatot és számoljuk ki u(S)-t
    if u(S) nem létezik **then**
        Töröljük S-t az L-ről
    if u(S)<c(L) **then**
        Töröljük S-t az L-ről
    if u(S)>c(L) és S LP-relaxáltjának létezik egész optimális megoldása **then**
        c(L)=u(S)
        Mentsük el a optimális egész megoldást x^*-ba.
        Töröljük S-t az L-ről
    if u(S)>c(L) és S LP-relaxáltjának nem létezik egész optimális megoldása **then**
        Legyen x' egy optimális megoldása S LP-relaxáltjának, ahol x'_i nem egész
        Készítsünk S-ből két részfeladatot egy-egy feltétel hozzávételével
        S'-nél vegyük hozzá a $x_i \leq \lfloor x'_i \rfloor$ feltételt
        S''-nél vegyük hozzá a $x_i \geq \lceil x'_i \rceil$ feltételt
        Vegyük fel az S', S''-t az L listára
        Töröljük S-t az L-ről
Az x^* a feladat optimális megoldása és c(L) az optimum értéke

2.2. Szimmetrikus utazóügynök feladat

A gyakorlatban sokszor előfordul, hogy ugyanolyan hosszú úton tudunk eljuttni egy adott városból egy másik városba, mint fordítva. Azaz, hogy $\forall i, j \in 1, 2, ..., n \; i \neq j$ esetén $c_{ij} = c_{ji}$. Ebben az esetben *szimmetrikus utazóügynök problémáról* (angolul symmetric traveling salesman problem) beszélünk.

Minden τ körtúrához, azaz a csúcsok minden lehetséges bejárásához (Hamilton-körhöz) egy x^τ incidencia vektort feleltetünk meg. Ekkor $x^\tau \in \mathbb{R}^{|E|}$ és

$$x^\tau_{ij} = \begin{cases} 1, \text{ ha } (ij) \in \tau \\ 0, \text{ egyébként} \end{cases}$$

Jelölje $Q^n = \text{conv}\{x^\tau \in \mathbb{R}^{|E|} \mid \tau \text{ egy körtúra incidenciavektora}\}$, azaz az incidencia vektoroknak a konvex burkát. Ez lesz a szimmetrikus utazóügynök probléma politópja. Az ügynök által bejárandó városakat és utakat megfeleltethetjük egy

gráfnak, ami így egy teljes n csúcsú gráfot eredményez (ha valahonnan nem lehet egy másik helyre eljutni, oda ∞ nagy költségű élt húzunk be). Ha ennek a gráfnak A-val jelöljük az incidencia mátrixát, akkor tekinthetjük a következő politópot: $Q_A^n = \{x \in \mathbb{R}^{|E|} \mid Ax = \mathbf{2}, \mathbf{0} \leq x \leq \mathbf{1}\}$. Ez a politóp a gráf diszjunkt körökkel való fedéseit tartalmazza. Q^n is egy ilyen fedése a gráfnak (csúcsainak fokszáma 2), emiatt $Q^n \subseteq Q_A^n$.

Jelöljük $\mathcal{L}_n$-nel egyenlőtlenségek egy véges családját. Ezeket megválaszthatjuk úgy, hogy a Q^n oldalait tartalmazzák, azaz az egyenlőtlenségek a Q^n politóp oldalait írják. Ekkor $Q^n = \{x \in Q_A^n \mid lx \leq l_0 \quad \forall (l, l_0) \in \mathcal{L}_n\}$

Poliéderes vágósíkos algoritmus:

Egyik megoldási lehetőség erre a problémára egy poliéderes vágósíkos algoritmus (2. algoritmus) (angolul: polyhedral cutting-plane algorithm) használata. Ezt a módszert írta le G. L. Nemhauser és L. A. Wolsey [13]-ban. Mivel $\mathcal{L}_n$ mérete nagyon nagy is lehet, ezért nem a teljes $\mathcal{L}_n$ feltételrendszerre oldja meg a feladatot, hanem csak ennek egy $\mathcal{L}' \subseteq \mathcal{L}_n$ részhalmazára. Majd az így kapott megoldást vizsgálja, hogy megfelel-e a kimaradt feltételnek. Ha talál az algoritmus sértő feltételt, akkor azzal bővíti az $\mathcal{L}'$ feltételhalmazt, és erre megszorítva keres újra megoldást. Egy új feltételnek a bevétele valójában a Q_A^n politóp egy síkkal való elvágása (szűkítése), ezért nevezik ezt az algoritmust vágósíkos algoritmusnak. Ezért a következő lineáris optimalizálási feladatot írhatjuk fel:

$$\begin{aligned} \min(cx) & \\ Ax &= \mathbf{2} \\ lx &\leq l_0 \qquad\qquad \forall (l, l_0) \in \mathcal{L}' \\ \mathbf{0} &\leq x \leq \mathbf{1} \end{aligned}$$

Algorithm 2 Poliéderes vágósíkos algoritmus

1: Legyen $\mathcal{L}' = \emptyset$
2: **while** Igaz **do**
3: Oldjuk meg az előbbi lineáris programozási feladatot $\mathcal{L}'$-vel és legyen az optimális megoldás $\overline{x}$.
4: Keressünk 1 vagy több egyenlőtlenséget $\mathcal{L}_n$-ben, amit az $\overline{x}$ nem teljesít
5: **if** Ha találtunk ilyen egyenlőtlensét **then**
6: Vegyük be őket $\mathcal{L}'$
7: **if** Ha nem találtunk ilyet **then**
8: Vége, az $\overline{x}$ optimális megoldás

A 2. algoritmus 3. lépésénél nem követeljük meg az x vektor egészértékűségét, ezért a $\overline{x}$ megoldás kiszámításához használható a szimplex algoritmus. A 4. lépés nagyon lényeges a hatékonyság szempontjából. Ennek az azonosítási problémának

a lényege, hogy a kapott megoldás által sértett feltételeket keressünk az $\mathcal{L}_n$ egyenlőtlenség családban. Erre a problémára az $\mathcal{L}_n$ választásától függően több pontos és heurisztikus algoritmus is létezik, de ezek nem mindegyike garantálja a nem kielégített feltétel megtalálását (de ugyanakkor, ha talál, akkor gyorsabban megtalálja). Vagyis ha a 2. algoritmus leáll, akkor elképzelhető, hogy nem kapunk optimális megoldást, mivel a 4. lépésben egy ilyen heurisztikus algoritmust használtunk, ami nem találta meg a megfelelő feltételt.

Ha az algoritmus egy olyan $\overline{x}$ megoldással áll meg, ami egész és egy körtúra incidencia vektora, akkor az az optimális megoldása a szimmetrikus utazóügynök problémának. Ha a kapott $\overline{x}$ megoldásunk olyan, hogy semelyik túrának sem az incidenciavektora, akkor a probléma optimális megoldásának a megtalálásához további lépések szükségesek. Ezek a következők lehetnek:
1. Ha a kapott $\overline{x}$ nem egész, akkor használhatjuk a korlátozás és szétválasztás módszerét (1. algoritmus). A kapott $\overline{x}$ értéket használhatjuk a korlátozás és szétválasztás algoritmusban alsó korlátnak.
2. Ha a megoldás egész, de nem egy körtúra incidencia vektora, hanem több diszjunkt köré, akkor további egyenlőtlenségek felsorolása szükséges. Az $\mathcal{L}'$ feltételei közé vegyünk fel egy vagy több olyan $\overline{x}$ által sértett egyenlőtlenséget, ami megakadályozza diszjunkt körökre való szétesését az útnak. Ekkor a részhalmazok felsorolásával a következő egészértékű lineáris programot írhatjuk fel. A felírásban jelölje x(E(W)) a W csúcshalmazban belül futó kiválasztott élek számát x-ben.

$$\begin{aligned} \min(cx) & \\ Ax &= \mathbf{2} \\ x(E(W)) &\leq |W| - 1 \qquad \forall \emptyset \neq W \subset V \\ \mathbf{0} \leq x &\leq \mathbf{1} \\ x &\in \mathbb{Z} \end{aligned}$$

Ekkor a korlátozás és szétválasztás algoritmust újra kell indítani a kibővített $\mathcal{L}'$ feltétel családdal. Az algoritmus ezen részét nevezik felsorolási fázisnak.

Az előző megközelítés hátránya, hogy a vágósíkos algoritmus és az összefüggőséget megkövetelő felsorolási fázis teljesen elkülönül egymástól a szétválasztás és korlátozás algoritmusban. Ezért a Q^n politópról megszerzett új információk nem használhatók ki a felsorolási fázisban. Ezen kívül ha a korlátozás és szétválasztás algoritmus diszjunkt körökkel fejeződik be, akkor a teljes felsorolási fázist megismétli, ami nem hatékony.

Ezeket a hátrányokat írta le Manfred W. Padberg és Giovanni Rinaldi [16], majd bemutattak egy szétválasztáson és vágáson alapuló algoritmust (angolul: branch and cut algorithm).

Szétválasztás és vágás algoritmus:

Legyen $\mathcal{L} \subset \mathcal{L}_n$, ahol $\mathcal{L}_n$ olyan, hogy tartalmazza Q^n oldalait, valamint legyen $F_0, F_1 \subset E$, $F_0 \cap F_1 = \emptyset$ az élek két diszjunkt halmaza.
Ekkor jelöljük $P(\mathcal{L}, F_0, F_1)$-vel a következő lineáris programot:

$$\begin{aligned} \min(cx) & \\ Ax &= \mathbf{2} & \\ lx &\leq l_0 & \forall (l, l_0) \in \mathcal{L} \\ x_e &= 0 & \forall e \in F_0 \\ x_e &= 1 & \forall e \in F_1 \\ \mathbf{0} \leq x &\leq \mathbf{1} & \end{aligned}$$

Jelöljük S-sel az élek diszjunkt rendezett részhalmazpárjainak halmazát, azaz $S = \{\langle F_0, F_1 \rangle \mid F_0, F_1 \subset E,\ F_0 \cap F_1 = \emptyset\}$. Valamint legyen x^* egy tetszőleges körtúra incidencia vektora.

Ekkor a szétválasztás és vágás algoritmus (3. algoritmus) az előző jelöléseket használva a következőképpen írható fel.

Algorithm 3 Szétválasztás és vágás

1: Legyen $S = \{\langle \emptyset, \emptyset \rangle\}$
2: $\mathcal{L} = \emptyset$
3: **while** S nem üres **do**
4: Válasszunk egy $\{\langle F_0, F_1 \rangle\} \in S$ párt
5: S=S-$\{\langle F_0, F_1 \rangle\} \in S$
6: Oldjuk meg a $P(\mathcal{L}, F_0, F_1)$ lineáris programot.
7: **if** A lineáris program nem megoldható **then**
8: Lépjünk vissza a 3. lépésre
9: Legyen $P(\mathcal{L}, F_0, F_1)$ optimális megoldása $\overline{x}$
10: **if** $c\overline{x} \geq cx^*$ **then**
11: Lépjünk vissza a 3. lépésre
12: Keressünk 1 vagy több egyenlőtlenséget $\mathcal{L}_n$-ben, amit nem teljesít $\overline{x}$
13: **if** Ha találtunk ilyen feltételt **then**
14: Adjuk hozzá az $\mathcal{L}$-hez és lépjünk vissza a 6. lépésre
15: **if** Ha nem találtunk ilyen feltételt **then**
16: **if** $\overline{x}$ egész megoldás **then**
17: $x^* = \overline{x}$ és lépjünk vissza a 3. lépésre
18: **if** $\overline{x}$ nem egész megoldás **then**
19: Válasszunk egy $e \in E$ élet, amire $0 < \overline{x_e} < 1$.
20: $S = S + \{\langle F_0 + \{e\}, F_1 \rangle\} \in S + \{\langle F_0, F_1 + \{e\} \rangle\} \in S$
21: Lépjünk vissza a 3. lépésre

A szétválasztás és vágás algoritmusa minden $\langle F_0, F_1 \rangle$ rendezett párt legfeljebb egyszer készít el a 18. lépésben. Mivel az ilyen rendezett pároknak a száma $2^{|E|}$, ezért a while ciklus maximum ennyiszer fut le az algoritmus során. Valamint a 12. lépésben vizsgált $\mathcal{L}_n$ feltételrendszer is véges. Ezek miatt a 3. algoritmus véges lépésben megáll. Ha az algoritmus megáll, akkor az x^*-ban tárolt megoldás a probléma optimális megoldása.

A while ciklus első lefutásánál, ha a vizsgált rendezett élrészhalmaz a $\langle \emptyset, \emptyset \rangle$ és ha a 12. lépést végrehajtjuk, akkor az a lépés ekvivalens az előzőekben leírt vágósíkos algoritmussal. (Kivéve, ha a vizsgált $\overline{x}$ megoldás diszjunkt részkörök uniója, mert ekkor a vágósíkos algoritmus a felsorolási fázisba lép, majd megismétli a megoldás keresését, ami nem hatékony.)

A szétválasztási-fában a csúcsokat (ami egy $P(\mathcal{L}, F_0, F_1)$ részprobléma) megfeleltethetjük az $\langle F_0, F_1 \rangle$ rendezett pároknak. Az S halmazban lévő csúcsokat tekintjük az aktív csúcsok halmazának. Ha egy csúcsra kiszámolt $\overline{x}$ optimális megoldás, olyan hogy $c\overline{x} \geq cx^*$, akkor a csúcsot feldolgozottnak tekintjük (mivel az $\overline{x}$ megoldásnál már találtunk egy jobbat).
Ha egy csúcs optimális megoldása egész, akkor a csúcsot megoldottnak tekintjük (és elmentjük x^*-ba, ha jobb mint az eddig talált legjobb megoldás).
Ha egy $\langle F_0, F_1 \rangle$ csúcs optimális megoldása tört és nem találtunk az $\mathcal{L}_n$ feltételcsaládban olyan feltételt, amit nem teljesítene, akkor ezt a csúcsot szétválasztjuk. Választunk egy e élt, aminek a mentén végrehajtuk a szétválasztást. Két gyerekcsúcsot készítünk ehhez a csúcshoz, amiknél az egyikben az e élt az F_0-hoz adjuk (azaz ennek az élnek a használatát megtiltjuk) a másiknál F_1-hez adjuk (azaz megkövetlejük a használatát).

Az algoritmus 6. lépésében megoldott $P(\mathcal{L}, F_0, F_1)$ lineáris program a $P(\mathcal{L}_n, F_0, F_1)$ egy relaxáltja. Ennek a részproblémának megengedett megoldásainak a halmaza a Q^n politóp levágva a $x_e = 0 \quad \forall e \in F_0$, $x_e = 1 \quad \forall e \in F_1$ egyenletek által definiált hipersíkokkal. Ugyanakkor a $P(\mathcal{L}, F_0, F_1)$ egy speciális relaxációja a $P(\mathcal{L}_n, F_0, F_1)$-nek, mivel az $\mathcal{L}$ minden egyenlőtlensége teljesül az egész Q^n politópra. (Mivel $\mathcal{L}_n$ az a feltételrendszer volt, ami a Q^n_A politópból kimetszi a Q^n politópot.) Emiatt mindig az aktuálisan vizsgált $\mathcal{L}$ feltételek halmazát haszálhatjuk az elválasztás bármely csúcsánál. Ezzel memóriát spórolhatunk meg, mivel az egyes csúcsokban nem kell nyilvántartani az abban a csúcsban alkalmazandó feltételrendszertúű, mert mindenegyik ugyanazokat a feltételek használja.

Használhatjuk a $P(\mathcal{L}_n, F_0, F_1)$ más relaxáltját is az algoritmus során. Olyanokat, amiknél vannak olyan feltételek, amik nem teljesülnek az egész Q^n politópra. Ilyen például a klasszikus vágósíkos algoritmus a Gomory vágásokkal. Mivel ezek nem az egész Q^n politópra érvényeseknek, az elágazásoknál nyilván kell tartani az egyes ágakon használt feltételrendszert. M. W. Padberg és G. Rinaldi szerint ez

több elágazásnál nagyon sok memóriát is foglalhat, ezért nagy problémák esetében kevésbé ajánlott a használata [16]. Ennek egy alternatív megoldása lehet, hogy nem haszáljuk a korlátozás és szétválasztás módszerét, csak a Gomory vágásokkal keresük a probléma megoldását. Ez viszont nagy számú város esetén lassú megoldsát eredményezne.

A 3. algoritmus hatékonysága függ a kezdetben tetszőleges választott x^* megoldástól. Ugyanis ha az optimálishoz közelebb lévő megoldásból indulunk ki, akkor kevesebb elágazást fog végrehajtani az algoritmus, ezáltal csökken a futási ideje is. Valamint ugyanígy befolyásolhatja a futási időt, hogy milyen $\langle F_0, F_1 \rangle$ párt választ az algoritmus és ha szükséges akkor melyik változó szerint választja szét a problémát a 18. lépésben.

2.2.1. Feltétel azonosítási probléma

Az azonosítási probléma célja, hogy egy adott $\overline{x}$ megoldásról eldöntse, hogy egy adott $\mathcal{L}_n$ feltétel családból minden feltételt teljesít-e a megoldás. Ha nem teljesít minden feltételt, akkor pedig találjon egy sértett feltételt az $\mathcal{L}_n$-ben. Ez a probléma a korlátozás és vágás algoritmus a 12. lépésében fogalmazódik meg. Az azonosítási probléma leírását és megoldását M. Padberg és G. Rinaldi dolgozta ki a [15] cikkben. Ehhez több feltételcsaládot is használhatunk az $\mathcal{L}_n$ meghatározására. Ekkor ezek 4 csoportba oszthatók szét:

1. Részköröket tiltó egyenlőtlenségek (Subtour elimination inequalities)
2. 2-párosító egyenlőtlenségek (2-Matching inequalities)
3. Fésű egyenlőtlenségek (Comb inequalities)
4. Klikk-fa egyenlőtlenségek (Clique-tree inequalities)

Az egyenlőtlenségek leírásához az alábbi jelöléseket használjuk:
A $G(V, E)$ gráf éleit jelöljük a végpontjaikkal, azaz $e = [u, v] \quad \forall e \in E$, ahol $u, v \in V, u \neq v$. Legyen $x \in \mathbb{R}^{V^2}$ az a vektor, ami az élek súlyait jelöli.
Valamint ha $W \subseteq V$, akkor definiáljuk a következő halmazokat:

$$\begin{aligned} E(W) &= \{[u, v] \in E \mid u, v \in W\} \\ \delta(W) &= \{[u, v] \in E \mid u \in W, v \notin W\} \\ \{S : T\} &= \{[u, v] \in E \mid u \in S, v \in T\}, \text{ ahol } S, T \subset V, \quad S \cap T = \emptyset \end{aligned}$$

A V gráfnak a W által kifeszített súlyozott részgráfja legyen $(G^W, x^W) = (W, E^W, x^W)$, ahol $x^W = (x_e)_{e \in W^2}$.
A x vektornak egy $e = [u, v]$ élre vett értékét jelölje x_e vagy $x(u, v)$ és az élek egy

$F \subseteq E$ részhalmazára pedig legyen $x(F) = \sum_{e \in f} x_e$.
Valamint a (G, x) gráf egy $W \subseteq V$ csúcsrészhalmaz összehúzásából keletkezett gráf a $(G[W], x[W]) = (V[W], E[W], x[W])$ (azaz W halmazt a $\langle W \rangle$ csúcs helyettesíti), ahol $x[W] \in \mathbb{R}^{E[W]}$ és

$$\begin{aligned} V[W] &= (V \setminus W) \cup \langle W \rangle \\ E[w] &= (E \setminus E(W) \setminus \{W : V \setminus W\}) \cup \{[\langle W \rangle, v] \mid v \in V \setminus W, \quad x(\{W : v\}) > 0\} \\ x[w](u, v) &= x(u, v) \quad \forall [u, v] \in E \cap E[W] \\ x[w](\langle W \rangle, v) &= x(\{W : \{v\}\}) \quad \forall v \in V \setminus W \end{aligned}$$

Vagyis az új $(G[W], x[W])$ gráfban egy [u,v] él $x(u, v)$ súlyának értéke változatlan az eredetihez képest ha $u, v \in V \setminus W$.
Valamint egy újonnan készített $[\langle W \rangle, v]$ él $x[W]$ súlya pedig legyen egyenlő a v csúcs a W csúcshalmaz között futó élek c súlyainak összegével.
Ekkor a feltételek a következőképpen írhatóak fel:

1.Részköröket tiltó egyenlőtlenségek (Subtour elimination inequalities):

$$x(E(W)) \leq |W| - 1 \quad \forall W \subseteq V, 2 \leq |W| \leq n - 1$$

Ez a feltétel megakadályozza, hogy diszjunkt részkörök uniójára essen szét a megoldást.

2. 2-párosító egyenlőtlenségek (2-Matching inequalities):

$$x(E(H)) + x(E') \leq |H| + \frac{1}{2}(|E'| - 1)$$

Minden $H \subset V$ és minden $E' \subset E$-re, amikre:

- $|e \cap H| = 1 \quad \forall e \in E'$
- $e_j \cap e_i = \emptyset, e_j \neq e_i \in E'$
- $|E'| \geq 3$ és páratlan

Ez a feltétel rendszer először kiválasztja a gráf egy páratlan számú élt használó párosítását (ez lesz az E'), majd minden élén kiválasztja az egyik csúcsot (ez lesz a H halmaz). Majd az egyenlőtlenség megköveteli, hogy az ezen csúcsok által kifeszített élek, valamint a párosítás élei között csak maximum $|H| + \frac{1}{2}(|E'| - 1)$ darab legyen kiválasztva. Ez a körtúrákra valóban teljesül, mivel ha a párosítás összes éle ki van választva, akkor a H halmazon belül csak $\frac{1}{2}(|E'| - 1)$ darab számú él lehet kiválasztva (különben nem kapnánk kört). Ha nincsen minden párosításbeli él kiválasztva,

hanem csak k db, akkor egy Hamilton kör ezen élek közül maximum $|H|+\lfloor\frac{k}{2}\rfloor$. Mivel $k \leq |E'|$, ezért ebben az esetben is teljesül az egyenlőtlenség.

3. Fésű egyenlőtlenségek (Comb inequalities):

$$x(E(H)) = \sum_{j=1}^{s} x \leq |H| + \sum_{j=1}^{s}(|T_j| - 1) - \frac{1}{2}(s+1)$$

Minden $H, T_1, T_2, ..., T_s \subseteq V$, amikre:

- $|T_j \cap H| \geq 1, \quad j = 1, 2, ..., s$
- $|T_j \setminus H| \geq 1, \quad j = 1, 2, ..., s$
- $T_i \cap T_j = \emptyset, \quad 1 \leq i < j \leq s$
- $s \geq 3$ és páratlan

Ekkor a H halmazt fésűnek nevezik, míg a $T_1, T_2, ..., T_s$ halmazokat a féső fogainak. Minden T_j foghalmaznak kell lennie legalább 1 közös pontjának a H fésűhalmazzal és legalább 1 olyan pontnak, ami nincs benne a H-ban. Ekkor egy körnek ebben a $(H \cup (\cup_{j=1}^{s} T_j))$ csúcshalmazban maximum $|H| + \sum_{j=1}^{s}(|T_j| - 1) - \frac{1}{2}(s+1)$ éle futhat. Mivel ha mindegyik foghalmazokban maximális (azaz $|T_j| - 1$) számú él van kiválsztva és midegyik fognak csak 1 közös pontja van a fésűhalmazzal, akkor a fésűhalmazban maximum $|H| - s + 1 + \lfloor\frac{s-2}{2}\rfloor$ db él lehet kiválasztva a körben. Ezt úgy kapjuk, hogy az egyik fogból megyünk a H fésőhalmazba, ott bejárunk néhány csúcsot, amik semelyik foghalmaznak sem a részei, majd kilépünk a fésűhalmazból valamelyik másik foghalmazba. Ezt annyiszor megismételjük, amennyiszer lehet közben figyelve arra, hogy a végén a H az összes olyan csúcsa be legyen járva, ami semelyik foghalmazban sincs benne. Összesen $\lfloor\frac{s}{2}\rfloor$ ismételhetjük és eközben a H halmazon belül $H - s - 1 - \frac{s-1}{2} + s - 1$ darab élet húzunk be. Ezeknek és a foghalmazon belül futó élszámoknak az összege éppen a feltétel jobb oldalt adja vissza.
Ha nem minden foghalmazban van maximális számú él behúzva vagy valamelyik foghalmaz több, mint 1 ponttal kapcsolódik a H fésűhalmazhoz, akkor ennél kevesebb élet tudunk csak kiválasztani. De ezekben az esetekben is teljesül az egyenlőtlenség.

4. Klikk-fa egyenlőtlenségek (Clique-tree inequalities):

$$\sum_{i=1}^{r} x(E(H_i)) + \sum_{j=1}^{s} x(E(T_j)) \leq \sum_{i=1}^{r} |H_i| + \sum_{j=1}^{s}(|T_j| - t_j) - \frac{1}{2}(s+1)$$

minden $H_1, H_2, ..., H_r \subseteq V$ és minden $T_1, T_2, ..., T_s \subseteq V$, amik a klikk fa fésői és fogai. A klikk fa egy olyan összefüggő komponensei a gráfnak, amikre a következőek teljesülnek:

- A klikk két részre van osztva: a fésűk halmazára és a fogak halmazára.
- A foghalmazok diszjunktak
- A fésűhalmazok diszjunktak
- $2 \leq |T_j| \leq n-2 \quad \forall j = 1, 2, ..., s$ és bármelyik foghalmaz tartalmaz legalább 1 csúcsot, ami nincs benne semelyik fésűhalmazban
- Ha egy foghalmaznak és egy fésűhalmaznak a metszete nem üres, akkor az a halmaz a gráf egy kapcsolódási halmaza

Egy gráf kapcsolódási halmaza alatt olyan minimális S csúcshalmazt értünk, amit elhagyva a gráfból, akkor a keletkezett gráfnak több összefüggőségi kompinense van, mint az eredeti gráfnak. Az egyenlőtlenségben t_j jelöli a a T_j foghalmaznak a fésőhalmazokkal való metszeteinek a számát.

Ez a négy egyenlőtlenség család meghatározzak a Q^n politóp oldalait, tehát ezek alkotján az $\mathcal{L}_n$ feltétel halmazt.

2.2.2. A sértett feltételek azonsoító algoritmusok

Részkörököt tiltó egyenlőtlenségek azonosítása:
Részkörököt tiltó egyenlőtlenségek azonosítására szolgáló algoritmushoz tekintsük először ennek a feltételcsoportnak egy átfogalmazását.

Tétel: Egy $x \in Q_A^n$ vektor pontosan akkor sért megy egy részkörököt tiltó egyenlőtlenséget, ha létezik a csúcsoknak olyan $\{W : V \setminus W\}$ vágása, aminek a kapacitása kisebb, mint kettő, azaz $x(\{W : V \setminus W\}) < 2$.

A $x(E(W)) \leq |W| - 1$ és a $x(\{W : V \setminus W\}) < 2$ feltételek azonosak, ha x-re teljesül, hogy $\forall v \in V$-re $x(\delta(v)) = 2$, azaz bármely csúcsban pontosan 2 db él van kiválasztva x-ben. Ez az x-re valóban teljesül, mivel $x \in Q_A^n$ miatt $Ax = 2$. Ekkor az első egyenletet (-2)-vel megszorozva és átrendezve a bal oldalon $2|W| - 2x(E(w))$-t kapunk. Itt a $2|W|$ éppen a W halmazban lévő csúcsokból kiinduló élek száma (a halmazon belül futó élek itt kétszer vannak számolva, míg a halmazból kilépő élek csak egyszer), ebből vonjuk le a $2x(E(w))$, ami a W halmazon belül futó élek számának kétszerese. Így a W halmazból kilépő élek számát kapjuk (egyszer számolva mindegyiket),ami éppen egyenlő $x(\{W : V \setminus W\})$-vel. Így az első egyenlőtlenséget átalakítottuk a második egyenlőtlenséggé.

Gomory és Hu algoritmusa [6] megtalálja egy súlyozott gráf minimális súlyú s-t vágását $\mathcal{O}(|V|^3)$ idő alatt. Ahhoz, hogy a gráf minimális súlyú vágását megtaláljuk legrosszabb esetben $|V|-1$-szer kell futtatni ezt az algoritmust. Így összesen $\mathcal{O}(|V|^4)$ futási időt eredményezne, ha ezzel a módszerrel szeretnénk vizsgálni a részkörököt

tiltó egyenlőtlenségek helyességét. Ez nagy számú csúcs esetén hosszú számolásokat eredményezne.

Crowder és Padberg [3] egy heursiztikus algoritmust javasolt erre a problémára, ami gyorsabban talál egy sértett egyenlőtlenséget. Azonban ez az algoritmus nem pontos, így előfordulhat, hogy nem találja meg a megfelelő részköröket tiltó feltételt.

Algorithm 4 Sértő feltétel keresése a részköröket tiltó feltételek között

1: Legyen $L_u = \{u\}$, $u \in V$-re
2: Legyen $(G', x') = (V', E', x') = (V, E, x)$
3: **if** A (G', x') nincs az x'-ben kiválasztott éle (1 értékű éle) **then**
4: Lépjünk a 14. lépésre
5: **if** A (G', x') van az x'-ben kiválasztott éle (1 értékű éle) **then**
6: Legyen az [u,v] egy ilyen 1 értékű él
7: $S = \{u, v\}$
8: $(G', x') = (G'[S], x'[S])$
9: $L_{\langle S \rangle} = L_u \cup L_v$
10: **if** $x'(\delta(\langle S \rangle)) < 2$ **then**
11: Jegyezzük meg az $L_{\langle S \rangle}$ halmazhoz tartozó sértett egyenlőtlenséget
12: **if** $x'(\delta(\langle S \rangle)) \geq 2$ **then**
13: Lépjünk vissza a 3. lépésre
14: **if** |V'|=1 **then**
15: STOP
16: **if** |V'|>1 **then**
17: Alkalmazzuk a Gomory-Hu algoritmust a (G',x') gráfra

A 4. algoritmus amíg talál az x vektorban 1 értékű élet, addig ezek közül választ egy tetszőlegeset. Ennek az élnek 2 végpontját összehúzza 1 pontra. Ha az eredeti gráfban az x vektor egy körútnak volt az incidencia vektora, akkor az összehúzás is egy kör kell, hogy legyen az új vektor az új gráfban.
Ha nem találunk 1 értékű élet, akkor 2 lehetőségünk van. Ha nincs is több éle a gráfnak, azaz 1 pontra összehúztuk az egész gráfot, akkor megállhatunk és az esetlegesen talált sértő feltételeket alkalmazhatjuk. Ha több, mint 1 pontja maradt a gráfnak és az eredeti vektorunk egy kör incidencia vektora volt, akkor ennél a lépésnél 2 db pontot kell kapnunk, közöttük egy 2 súlyú éllel. Erről a 2 csúcsú gráfról hamar el tudjuk dönteni, hogy nem tartalmaz 2-nél kisebb súlyú vágást és megállhatunk.
De az is előfordulhat, hogy például nem egy nagy kör, hanem több diszjunkt részkör incidencia vektora volt az x vektor. Ekkor az összehúzások után, hogy a körök teljesen eltűnnek a gráfból és pár darab 2 súlyú él marad vissza utánuk. Ekkor ha több diszjunkt körből indultunk ki, akkor ezek a 2 súlyú élek se kapcsolódnak össze a gráfban. Ebben az esetben találunk 2-nél kisebb súlyú (itt 0 súlyú) vágást az új gráfban.

Ha az x értékei nem voltak egészek, akkor előfordulhat, hogy az összehúzott pontokból egy olyan csúcs keletkezik, amiből a kilépő éleinek a súlya kevesebb, mint 2. Ebben az esetben az eredeti gráfban biztosan volt 2-nél kisebb súlyú vágás (méghozzá az ebbe a csúcsba összehúzott pontok halmaza), ezáltal egy sértő feltételt találunk. Ezt ellenőrzi az algoritmus 10. lépése.
Mivel csak az 1 értékű éleket húzzuk össze, így tört x vektor esetén előfordulhat, hogy a 17. lépésben nagy számú csúcsra kell alkalmaznunk a Gomory-Hu algoritmust, ami jelentősen rontja ennek az algoritmusnak a hatékonyságát.

A 2-párosító egyenlőtlenségek azonosítása:

Ennek a feltételcsaládnak az azonosítására szolgáló algoritmusnak csak az alapját szeretném ismertetni, a teljes algoritmust Manfred Padberg és Giovanni Rinaldi dolgozta ki [15]. Ahhoz, hogy ebből a feltételhalmazból is találjunk egy egyenlőtlenséget, amit a kapott $\overline{x}$ vektor nem teljesít néhány elnevezést szükséges bevezetni.

Címkézett gráfnak nevezzünk egy gráfot a W csúcsainak egy $\{U, W \setminus U\}$ partíciójával. Az U és $W \setminus U$ halmazok közül az egyik lehet üres. Ekkor legyenek U csúcsai páratlanul címkézve, míg a $W \setminus U$ halmaz csúcsi párosín címkézve. A csúcsok egy S részhalmazát nevezzük páratlannak, ha $|S \cap U|$ páratlan, különben nevezzük párosnak. Legyen $S \subset W$, ekkor a $G[S]$ össszehúzott gráfnak az $\langle S \rangle$ csúcsa legyen páratlanul címkézve, ha az S halmaz páratlan, különben páros. Az összehúzott gráf bármely másik pontja a $W[S]$-nek megörökli az összehúzás előtti címkéjét. Az $\{S : W \setminus S\}$ vágást páratlannak nevezzük, ha S és a $W \setminus S$ halmaz is páratlan.

Legyen (G, x) egy súlyozott gráf és először minden csúcsa legyen párosan címkézve. A $(G', x') = (V', E', x')$ a (G, x) gráfból kapjuk él-felbontással a következőképpen:

$$V' = V \cup V_E \text{ ,ahol } V_E = \{v_e \mid e \in E\}$$
$$E' = \{[u, v_e], [v_e, w] \mid e = [u, w] \in E\}$$
$$x'(u, v_e) = x'(v_e, w) = x(u, w) \quad \forall e = [u, w] \in E$$

Ha a (G, x) egy ilyen címkézett gráf, $\mathcal{U}$ a páratlan csúcsok halmaza és legyen $J \subseteq E$, akkor a gráf J-vel való kiegészítésén a $(\overline{G}, \overline{x}) = (V, \overline{E}, \overline{x})$ gráfot értjük, ahol

$$\overline{x}_e = \begin{cases} x_e \text{ ,ha } e \in E \setminus J \\ 1 - x_e \text{ ,ha } e \in J \end{cases}$$

És $\overline{E} = \{e \in E \mid \overline{x}_e > 0\}$.

Legyen (G, x) gráfból él-felbontással kapott gráf a (G', x') és legyen $J \subseteq E'$ olyan, hogy $|J| = |E' - J|$ és $\forall e = [u, w] \in E$-re vagy $[u, v_e] \in J$ vagy $[v_e, w] \in J$. Jelöljük a (G', x') gráfnak a J-vel való kiegészítését (G^*, x^*)-nak. Ekkor a G^*-nak páros darab páratlan címkéjű csúcsa van és Padberg és Rao [14] a következőt

bizonyították róla: **Tétel:** Az $x \in Q_A^n$ vektor pontosan akkor sért meg egy 2-párosító feltételt, ha létezik a (G^*, x^*)-nak olyan $\{W : V^* - W\}$ páratlan vágása, hogy $x^*(W : V^* - W) < 1$. A tétel felhasználásával [15] cikkben olvashatunk a sértő feltételek azonosítására szolgáló algoritmusokat. Ugyanebben a cikkben olvashatunk a fésű egyenlőtlenségek és a klikk-fa egyenlőtlenségek néhány speciális alakjáról, valamint az előző részfejezetben leírt általános fésű és klikk-fa egyenlőtlenségeket azonosító algoritmusokat.

3. Járműirányítási probléma

Járműirányítási probléma alatt (angolul: Vehicle Routing Problem), olyan problémákat értünk, ahol egy vagy több járművel szeretnénk szolgáltatásokat vagy termékeket kiszállítani előre megadott helyekre. Adott V db jármű, egy raktár és szállítási helyek egy halmaza. Ezeken felül adottak a raktár és a szállítási helyek, valamint bármely két szállítási hely közötti út hossza. Valamint adott még minden szállítási helyhez egy igényszám, ami az oda szállítandó áruk mennyiségét jelenti. A feladat célja, hogy az összes jármű által megtett utat minimalizáljuk, úgy hogy minden szállítási igény ki legyen elégítve, valamint minden járműnek a raktárból kell indulnia és oda is kell visszaérnie.

A járműirányítási problémának számos variációja ismert, amikben a fenti feltételek mellett még más korlátozásoknak is eleget kell tennie a megoldásnak. Egyes változatokban a használt szállítási helyek halmaza megválasztható, így ezekben az esetekben a minimalizálandó célfüggvény is változni fog.
A jáműirányítási probléma az utazó ügynök probléma általánosításaként tekinthető, mert $V = 1$ választással visszavezethetjük rá a feladatot.
A továbbiakban néhány általánosítását mutatom be az előbb leírt problémának.

3.1. Kapacitásos járműirányítási probléma

A járműirányítási problémának ennek a változatában is adott a raktár, a szállítási helyek halmaza és a közöttük lévő út költsége, valamint minden szállítási helyhez egy rendelési mennyiség, amennyi mennyiségű árut kell kiszállítanunk az adott helyre. Ezen kívül adott V darab jármű, amikkel végezhetjük a kiszállítást és egy Q kapacitáskorlát a járművekre. Ez azt jelenti, hogy Q-nál nagyobb mennyiségű árut nem szállíthatunk ki egy járművel. Erről a feltételről nevezik ezt a problémát kapacitásos járműirányítási problémának (angolul: capacitated vehicle routing problem).

Ha a Q kapacitáskorlátot kellően nagynak vagy ∞-nek választjuk, akkor az előzőekben leírt járműirányítási problémát kapjuk vissza. Emiatt ez a probléma tekinthető a járműirányítási probléma általánosításaként.

3.1.1. A probléma felírásai

A probléma felírására számos megközelítés készült az évek alatt, amik közül néhányat szeretnék bemutatni. Ezekhez a következő jelöléseket szükséges bevezetni:

$V_C = \{1, ..., n\}$ az ügyfelek halmaza és a 0 csúcs jelölje a raktárat

$V = \{0, 1, 2..., n\}$

$A = \{(i, j) \mid i, j \in V, i \neq j\}$ az utak (irányított élek) halmaza

Q= egy jármű kapacitása

q_i jelölje az i-edik ügyfél rendelési mennyiségét $\forall i \in V_C$-re

$\delta^+(S) = \{(i,j) \mid i \in S, j \in V \setminus S\}$

$\delta^-(S) = \{(j,i) \mid i \in S, j \in V \setminus S\}$

$\delta^+(i) = \delta^+(\{i\})$ és $\delta^-(i) = \delta^-(\{i\})$

$q(S) = \sum_{i \in S} q_i$

c_{ij} = az (i,j) út utazási költsége

Ha $x \in [1,2]^{|A|}$ és $A' \subseteq A$, akkor $x(A') = \sum_{a \in A'} x_a$

Kétindexes folyam formula:

Először a kétindexes folyam formulát (angolul: The two-index vehicle flow formulation) tekintsük, amit G. Laporte és Y. Nobert írt le [10]. $\forall (i,j) \in A$ élhez feleltessünk meg egy bináris x_{ij} változót, aminek az értéke pontosan akkor legyen 1, ha az (i,j) utat használjuk az utazás során.
Ekkor a probléma a következőképpen fogalmazható meg:

$$\min \sum_{(i,j) \in A} c_{ij} x_{ij}$$

$$x(\delta^+(i)) = x(\delta^-(i)) = 1 \qquad \forall i \in V_C$$

$$x(\delta^+(S)) \geq \lceil \frac{q(S)}{Q} \rceil \qquad \forall S \subseteq V_C$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A$$

A $x(\delta^+(S)) \geq \lceil \frac{q(S)}{Q} \rceil \quad \forall S \subseteq V_C$ feltételt kerekített kapacitási feltételeknek (rounded capacity) nevezik [12].
Ez a feltétel a megengedett megoldásokra valóban teljesül, ugyanis, ha az ügyfelek egy részhalmazának a kapacitása nagyobb, mint kQ (k db jármű összkapacitása), akkor azt a részhalmazt legalább k+1 db jármű tudja csak kiszolgál. Így ennyi járműnek is kell elhagynia a halmazt.

Ennek a feltétel családnak az azonosítási problémája ismeretlen bonyolultságú, de jó heurisztikus algoritmusokat tudtak rá adni. Ugyanakkor a következő gyengített feltételekre hatékony azonosítási eljárások ismertek [12].

Tört kapacitás (fractional capacity) egyenlőtlenségek:

$$x(\delta^+(S)) \geq \frac{q(S)}{Q} \quad \forall S \subseteq V_C$$

Részköröket tiltó (subtour elimination) egyenlőtlenségek:

$$x(\delta^+(S)) \geq 1 \quad \forall S \subseteq V_C$$

Ez a két feltétel típus a kerekeríkett kapacitási feltételeknek egy-egy gyengített változatai, emiatt biztosan teljesülnek a megengedett megoldásokra.

Általános multistar (generalized large multistar) egyenlőtlenségek:

$$x(\delta^+(S)) \geq \frac{1}{Q}\sum_{i\in S}(q_i + \sum_{j\in V_C\setminus S} q_j(x_{ij} + x_{ji})) \quad \forall S \subseteq V_C$$

Ezekkel a feltételekkel Luis Gouveia cikkében találkozhatunk [7]. Ez az egyenlőtlenség család az előzőeknek egy finomított változata. Ugyanúgy számoljuk az ügyfelek S részhalmazának az összigényét, de minden járműek, ami belépett a részhalmazba valahonnan be kellett lépnie az S-be és valahova ki is kell lépnie. Ezért egy ilyen járművön kell lennie elég helynek az S halmazban kiszolgált ügyfelein kívül annak a két ügyfél igényeinek is, akitől belépett a halmazba a jármű és akihez kilép. Ezt minden S-ben járó járműnek tudnia kell, ezért az alsó korláthoz még hozzáadhatjuk azoknak az ügyfeleknek az igényeit, akiktől egy jármű az S-be ment tovább vagy akikhez az S-ből ment. Itt a raktárat egy 0 igényű ügyfélnek tekintjük, tehát $q_0 = 0$.

Ezeken kívül is található az irodalomban sok más egyenlőtlenség család a probléma ilyen felírásához [8].

Halmaz partícionáló formula (set partitioning):

Legyen Ω egy járműnek a lehetséges útvonalainak a halmaza. Minden $r \in \Omega$ útvonalhoz tartozzon egy z_r bináris változó, ami pontosan akkor veszi fel az egy értéket, ha a készített megoldásban valamelyik jármű az r útvonalat járja be. Készítsünk $\forall i \in V_C$ és $\forall r \in \Omega$-hoz egy konstans a_{ir} változót. Az a_{ir} értéke legyen 1, ha az i. ügyfél ki van szolgálva az r útvonalban, különben pedig legyen 0 értékű. Végül jelöljük c_r-rel $\forall r \in \Omega$-ra az r útvonal költségét. Ekkor a problémát a következőképpen írhatjuk fel:

$$\begin{aligned}
&\min \sum_{r\in\Omega} c_r z_r \\
&\sum_{r\in\Omega} a_{ir} z_r = 1 && \forall i \in V_C \\
&z_r \in \{0,1\} && \forall r \in \Omega
\end{aligned} \tag{1}$$

Mivel Ω mérete exponenciálisan nagy, így ahhoz, hogy hatékonyan meg tudjuk oldani ennek a feladatnak LP-relaxáltját oszlopgenerálásra van szükség. Az oszlopgenerálást a 4. fejezetben mutatom be.

Többtermékes folyam formula (Multi-commodity flow formulations): A probléma egy másik megközelítése a többtermékes folyam formula (angolul: multi-commodity flow formulations), amit Adam Letchford és Juan Jose Salazar Gonzalez részleteztek [11]. Itt legyen f_{ij}^k az a bináris változó, ami pontosan akkor veszi fel az 1 értéket, ha egy jármű használja az (i,j) élet, mielőtt meglátogatja a k ügyfelet. (Ha nem látogatja meg k, akkor biztosan 0 értéket vesz fel.) Hasonlóan legyen g_{ij}^k az a bináris változó, ami pontosan akkor 1, ha egy jármű használja az (i,j) élet azután, hogy meglátogatta a k ügyfelet.
Ha $A' \subseteq A$, akkor $f^k(A') = \sum_{(i,j)\in A'} f_{ij}^k$ és $g^k(A') = \sum_{(i,j)\in A'} g_{ij}^k$.
Ezek használatával a problémát a következőképpen írhatjuk fel:

$$\min \sum_{(i,j)\in A} c_{ij}x_{ij}$$

$$x(\delta^+(i)) = x(\delta^-(i)) = 1 \qquad \forall i \in V_C \quad (2)$$

$$f^k(\delta^+(0)) = g^k(\delta^-(0)) = 1 \qquad \forall k \in V_C \quad (3)$$

$$f^k(\delta^-(k)) = g^k(\delta^+(k)) = 1 \qquad \forall k \in V_C \quad (4)$$

$$f^k(\delta^-(0)) = g^k(\delta^+(0)) = 0 \qquad \forall k \in V_C \quad (5)$$

$$f^k(\delta^+(k)) = g^k(\delta^-(k)) = 0 \qquad \forall k \in V_C \quad (6)$$

$$f^k(\delta^-(l)) = f^k(\delta^+(l)) = g^l(\delta^-(k)) = g^l(\delta^+(k)) \qquad \forall, k \neq l \in V_C \quad (7)$$

$$\sum_{k\in V_C\setminus\{i,j\}} q_k(f_{ij}^k + g_{ij}^k) \leq (Q - q_i - q_j)x_{ij} \qquad \forall (i,j) \in A \quad (8)$$

$$f_{ij}^k + g_{ij}^k \leq x_{ij} \qquad \forall k \in V_C, (i,j) \in A \quad (9)$$

$$f_{ij}^k, g_{ij}^k \in \{0,1\} \qquad \forall k \in V_C, (i,j) \in A \quad (10)$$

$$x_{ij} \in \{0,1\} \qquad \forall (i,j) \in A \quad (11)$$

A 3. feltétel garantálja, hogy a 0 csúcs, vagyis a raktár elhagyása előtt és a raktárba való visszaérkezés után semelyik ügyfelet se látogathassa meg a jármű. (Azaz így a raktárban kezdődik és ott is végződik a jármű útja.) A 5. feltétel pedig azt követeli meg, hogy a járműveknek a raktár elhagyása után, de még a visszaérkezés előtt minden ügyfelet meg kell látogatnia egy járműnek.(Vagyis minden ügyfelet pontosan 1 jármű látogat.) A 4. és 6. feltétel a k csúcsba belépő és a k csúcsból kilépő éleket vizsgálja. (4) szerint a belépő élek közül pontosan egyet kell használnia egy járműnek (a járművek közül pontosan egynek szabad az ügyfélhez belépnie), mielőtt meglátogatja a k-t (nyilván előtte használja, mert ekkor lép be a jármű a k csúcsba). Hasonlóan a feltétel második fele szerint egy járműnek pontosan 1 élen keresztül kell elhagynia a k csúcsot. A (6) minden k ügyfélre azt követeli meg, hogy semelyik jármű

sem lephet be a k-ba azután, hogy meglátogatja őt. Valamint, hogy semelyik jármű sem hagyhatja el a k ügyfelet azelőtt, hogy meglátogatná őt. A 7. feltétel az ügyfelek meglátogatási sorrendjét követeli meg. Tehát ha a k ügyfelet később meglátogatjuk meg, minthogy belépünk az l csúcsba, akkor az l-ből a k ügyfél meglátogatása előtt ki is kell lépnünk. Valamint ekkor az l ügyfelet hamarabb kell meglátogatnunk, mint mielőtt be- és kilépünk a k csúcsba. És ugyanez igaz fordítva is. A 8. feltétel a kapacitási korlátot fogalmazza meg. Eszerint, ha egy jármű használja az (i, j) élet, akkor minden ez az él előtt és után kiszolgált ügyfélnek a kapacitása nem lépheti túl a járművön az i, j ügyfeleken felül maradt kapacitást. (Nyilvánvalóan ekkor a járműnek az i és j ügyfeleket is ki kell szolgálnia.)

3.2. Járműirányítási és ügyfélelhelyezési probléma

A járműirányítási és ügyfélelhelyezési probléma (angolul: Vehicle Routing with Demand Allocation Problem) a kapacitásos probléma általánosításaként tekinthető. Ebben a problémában adott a járművek V száma és Q kapacitása, valamint egy raktár, az ügyfelek és a szállítási helyek halmaza. A feladatban meg kell határozni, hogy melyik ügyfelet melyik szállítási helyen fogjuk kiszolgálni, ehhez minden ügyfélnek adott egy rendelési mennyisége és minden szállítási helyhez van egy "kényelmességi költsége", amit ki kell fizetnünk, ha az adott helyen szolgáljuk ki. Ezen kívül adottak a raktár és a szállítási helyek valamint bármely két szállítási hely közötti út költsége. A feladat extra feltétele, hogy minden szállítási helyet maximum egy járművel látogathatunk meg, hogy csökkentsük az adott hely forgalmát.

A probléma célja, hogy az ügyfeleket hozzá kell rendelni a szállítási helyekhez, majd ezeket a szállítási helyeket a járművekkel a megfelelő módon kell bejárni úgy, hogy az ügyfelek hozzárendelési költsége és a járművek útjának összeköltsége a lehető legkevesebb legyen. A megoldásban minden ügyfelet pontosan 1 szállítási helyen kell kiszolgálnunk és minden járműnek a raktárból kell indulnia és oda is kell visszaérkezni. Az egy jármű által meglátogatot szállítási helyekhez rendelt ügyfelek rendelési mennyiségeinek összege nem haladhatja meg a jármű Q kapacitását.
Ezen feltételek mellett az egyes járművek által bejárt szállítási helyek diszjunkt részhalmazok lesznek.

A kapacitásos járműirányítási problémát visszakaphatjuk, ha minden ügyfélhez készítünk egy saját szállítási helyet, aminek az ügyfélhez való hozzárendelési költsége 0 és bármely másik szállítási helyhez a hozzárendelési költségét ∞-nek választjuk (vagy kellően nagynak). Ekkor a hozzárendelési probléma egyértelmű (és 0 költséget ad hozzá a célfüggvényhez), így csak a járművek útvonalát kell minimalizálni. Az így redukált feladat egyenértékű a kapacitásos járműirányítási problémával.

Ennek a problémának a megoldására a 4.fejezetben mutatok be egy algoritmust.

3.3. Járműirányítási és elosztási probléma

Járműirányítási és elosztási problémában (angolul: Vehicle Routing-Allocation Problem) az előzőekhez hasonlóan adott a raktár, az ügyfelek halmaza a járművek darabszáma és a kapacitáskorlátjuk. Ezen felül minden ügyfélnek adott egy rendelési igénye és az egymástól valamint a raktártól vett távolságuk, illetve az is, hogy milyen költséggel tudjuk őt hozzárendelni egy másik ügyfélhez (milyen költséget kell kifizetnünk, ha az adott ügyfelet nem a saját helyén, hanem egy másik ügyfélnél szeretnénk kiszolgálni). Ekkor a szállítás megtervezésénél minden ügyfél esetében dönthetünk a következő választások közül:

- Kiszolgáljuk közvetlenül az ügyfelet a saját helyén.
- Kiszolgáljuk az ügyfelet, de nem látogatjuk meg közvetlenül. Arra kötelezzük, hogy utazzon el egy másik (általunk közvetlenül meglátogatott) ügyfélhez, ahol átveheti a rendelését.
- Egyáltalán nem szolgáljuk ki az ügyfelet.

Tehát ebben a problémában dönthetünk úgy, hogy egyes ügyfeleket egyáltalán nem szolgálunk ki. Ebben az esetben viszont a nem kiszolgált ügyfelek után egy *kihagyási* költséget kell fizetnünk, ami ügyfelenként eltérő lehet.

Ez a probléma is az előző alfejezetben leírt probléma általánosításaként tekinthető. Ugyanis a potenciális szállítási helyeket 0 igényű ügyfélként vegyük fel és a valódi ügyfeleket pedig ∞ távolinak válasszuk meg a raktártól és a szállítási helyeknek felvett 0 igényű ügyfelektől. Valamint még követeljük meg extra feltételként, hogy minden ügyfelet ki kell szolgálnunk (például egy ügyfél kihagyásáért olyan nagy árat kelljen fizetnünk, hogy ne érje meg őt kihagynunk). Ezzel megköveteltük, hogy minden ügyfelet hozzárendeljünk egy szállítási helyhez (0 igényű ügyfélhez). Ugyanakkor a szállítási helyek közül tetszőlegesen sokat ki is hagyhatunk a körből (a 0 igényű ügyfeleket nem kell feltétlenül kiszolgálnunk, csak ha rendeltünk hozzájuk egy nem 0 igéyű ügyfelet). Ekkor a problémát visszavezettük a járműirányítási és ügyfélelhelyezési problémára.

A problémának létezik egyjárműves és többjárműves változata is. Több logisztikai problémában is találkozhatunk ezzel a feladattal. Például a buszmegállók helyének tervezése, postafiókok telepítése vagy a katonai egységek kiképzéseinek megtervezése.

3.3.1. Járműirányítási és elosztási probléma felírása

A probléma egyjárműves változatának (angolul: single vehicle routing-allocation problem) leírását megtalálhatjuk [1]-ben. Ebben a cikkben a szerzők olyan problémá-

kat gyűjtöttek össze, amiket ez az egyjárműves járműirányítási és elosztási probléma általánosít. A probléma IP felírásában használt jelölések:

$V = \{0, 1, 2, ..., n, n+1\}$Az ügyfelek halmaza, ahol a 0 és n+1 jelölje a raktárat

c_{ij} az i-edik ügyféltől a j-edik ügyfélig tartó út költsége ($c_{ii} = \infty$)

d_{ij} a j-edik ügyfél hozzárendelési költsége az i-dik ügyfélhez

D_i az i-edik ügyfél kihagyási költsége

F_{on}= azon ügyfelek halmaza, akiket kötelező közvetlenül meglátogatnunk

F_{off}= azon ügyfelek halmaza, akiket kötelező közvetetten kiszolgálnunk, tehát valakihez hozzá kell rendelnünk

Az F_{on} és F_{off} halmazok használata nem változtatja meg a feladatot. Ugyanis ha egy ügyfelet be szeretnénk tenni az F_{on} halmazba, akkor az őt másokhoz való hozzárendelési költségeit és izolálási költségét is ∞ nagynak választjuk. Ha pedig ez j. ügyfelet be szeretnénk tenni az F_{off} halmazba, akkor ezt a $c_{ij} = \infty \quad \forall i = 0, 1, ..., j-1, j+1, ...n$ valamint a $D_j = \infty$ választással érhetjük el. A raktárat a 0 és az n+1-es számú ügyfelek jelölik, így nem egy körtúrát keresünk hanem egy nyílt utat. Legyen $\{0, n+1\} \in F_{on}$ és $\forall i = 1, 2, ..., n$ esetén $c_{i0} = c_{n+1i} = \infty$. Így az út biztosan a 0-ás ügyfélnél fog kezdődni és az n+1-esnél befejeződni. Valamint a 0 és a n+1-es ügyfelekhez tartozó hozzárendelési költséget is ∞-nek kell választani. Az egészértékű porgramozás felírásához a következő változókra van szükség:

$$z_i = \begin{cases} 1 \text{ ,ha az i-edik ügyfelet közvetlenül kiszolgáljuk} \\ 0 \text{ ,egyébként} \end{cases}$$

$$W_i = \begin{cases} 1 \text{ ,ha az i-edik ügyfelet közvetetten kiszolgáljuk} \\ 0 \text{ ,egyébként} \end{cases}$$

$$v_i = \begin{cases} 1 \text{ ,ha az i-edik ügyfelet nem szolgáljuk ki} \\ 0 \text{ ,egyébként} \end{cases}$$

$$x_{ij} = \begin{cases} 1 \text{ ,ha az i-edik ügyfél után közvetlenül a j-edik ügyfelet látogatjuk meg} \\ 0 \text{ ,egyébként} \end{cases}$$

$$y_{ij} = \begin{cases} 1 \text{ ,ha az j-edik ügyfél a i-edik ügyfélhez van rendelve} \\ 0 \text{ ,egyébként} \end{cases}$$

Ekkor a problémát a következőképpen írhatjuk fel:

$$\min \lambda_{ut} z_{ut} + \lambda_{hr} z_{hr} + \lambda_{izol} z_{izol}$$

$$\sum_{i=0}^{n} \sum_{j=1}^{n+1} c_{ij} x_{ij} = z_{ut}$$

$$\sum_{i=1}^{n} \sum_{j=1}^{n} d_{ij} y_{ij} = z_{hr}$$

$$\sum_{i=1}^{n} D_i v_i = z_{izol}$$

$$z_i + w_i + v_i = 1 \qquad \forall i \in V$$

$$z_i = 1 \qquad \forall i \in F_{on}$$

$$w_i = 1 \qquad \forall i \in F_{off}$$

$$\sum_{i=1}^{n+1} x_{0i} = 1$$

$$\sum_{i=0}^{n} x_{in+1} = 1$$

$$\sum_{i=1}^{n+1} x_{ji} = \sum_{i=0}^{n} x_{ij} \qquad \forall j = 1, 2, ..., n$$

$[x_{ij}]$ vektor nem tartalmaz részköröket

$$\sum_{i=0}^{n+1} y_{ij} = w_j \qquad \forall j = 1, 2, ..., n$$

$$y_{ij} \leq z_i \qquad \forall i \in V, j = 1, 2, ..., n$$

$$x_{ij}, y_{ij} \in \{0, 1\} \qquad \forall i, j \in V$$

$$z_i, v_i, w_i \in \{0, 1\} \qquad \forall i \in V$$

Az alapfeladatban a célfüggvényben szereplő $\lambda_{ut}, \lambda_{hr}, \lambda_{izol}$ változók 1 értéket vesznek fel. A részköröket tiltó feltétel kifejezésére több matematikai formula is ismert, ilyenek például az utazóügynök problémánál ismertetett feltételrendszerek.

4. Járműirányítási és ügyfélelhelyezési probléma megoldása

Ebben a fejezetben a járműirányítási és ügyfélelhelyezési problémának szeretném egy megoldását leírni. A feladatot a 3.2. részfejezetben ismertettem. A következőekben bemutatott algoritmust Mohammad Reihaneh és Ahmed Ghoniem idén közölte [17].

4.1. Jelölések

Az algoritmus leírásához a következő jelölések bevezetésére van szükségünk:

$V =$ a járművek száma

$Q =$ egy jármű kapacitás korlátja

$K =$ ügyfelek halmaza

$d_k =$ a k ügyfél rendelési igénye $\forall k \in K$

$S =$ a szállítási helyek halmaza

f_{kj} =a k ügyfél j szállítási helyhez való hozzárendelésének költsége $\forall k \in K \quad \forall j \in S$

$N = S \cup \{0\}$, ahol a 0-val jelöljük a raktárat

$c_{ij} =$ az i-ből j-be menő út költsége $\forall i, j \in N$

4.2. Algoritmus

A problémát az 5. algoritmus segítségével oldhatjuk meg, amit szétválasztás és árképzési algoritmusnak (angolul: Branch and price algorithm) neveznek.

Az algoritmusban a már korábban bemutatott korlátozás és szétválasztás módszert használjuk. A megoldáshoz szükséges alsó korlátokat az egyes részfeladatok LP relaxáltjának megoldása adja. Mivel ezek az LP relaxált részfeladatok nagyon nagy méretűek is lehetnek, ezért ezeket oszlopgenerálással oldjuk meg.
Az algoritmus elején készítünk egy kezdeti megoldást egy heurisztka segítségével. Ennek a célfüggvényértéke fogja a felső korlátot biztosítani a feladat megoldására, amíg nem találunk nála jobb egész megoldást.

Algorithm 5 Szétválasztás és árképzés

1: Jelölje F_0 az eredeti feladatot és jelölje u(F) egy F (rész)feladat LP-relaxáltjának oszlopgenerálással kapott optimum értékét. Legyen L a részfeladatok listája.
2: Tegyük F_0-t az L listára.
3: Számoljuk ki a heurisztika segítségével a kezdeti megoldást. Ennek a célfüggvényértéke legyen c(L).
4: **while** Az L nem üres **do**
5: Válasszuk az L listáról a minimális $u(F)$ értékű F részfeladatot.
6: **if** $u(F) < c(L)$ **then**
7: $L = L \setminus \{F\}$
8: **if** $u(F) > c(L)$ és F LP-relaxáltjának létezik egész optimális megoldása **then**
9: c(L)=u(F)
10: Mentsük el a optimális egész megoldást x^*-ba.
11: $L = L \setminus \{F\}$
12: **if** $u(S) > c(L)$ és S LP-relaxáltjának nem létezik egész optimális megoldása **then**
13: Legyen x' egy optimális megoldása S LP-relaxáltjának, ahol x'_i nem egész
14: Készítsünk F-ből két F^1, F^2 részfeladatot a szétválasztási stratégiának megfelelően
15: $L = L \cup \{F^1, F^2\} \setminus \{F\}$
16: Az x^* a feladat optimális megoldása és c(L) az optimum értéke

4.3. Kezdeti megoldás

A kezdeti megoldást egy heurisztika segítségével állíthatjuk elő. Ennek folyamán először az ügyfeleket osztjuk szét a szállítási helyek között, majd az így kijelölt szállítási helyeket járjuk be a járművekkel.

Először az ügyfeleket rendeljük hozzá egy-egy szállítási helyhez, úgy hogy az ügyfelek hozzárendelési költségeinek összege a lehető legkevesebb legyen. Eközben figyeljünk arra, hogy semelyik szállítási helyre szállítandó összmennyiség se haladja meg a járművek kapacitását, különben teljesíthetetlen lenne a kiszállítás. Azaz azon $k \in K$-ra, amiket a j szállítási helyhez rendeltünk $\sum d_k \leq Q$ és ez $\forall j \in S$-re teljesül.

Ezután legyen az így kiválasztott szállítási helyek halmaza S^+, azaz amikhez legalább 1 db ügyfél hozzá lett rendelve. A első jármű menjen a raktárhoz legközelebbi szállítási helyhez. Töröljük ezt a szállítási helyet S^+-ból. Majd ezután mindig azt az S^+-beli szállítási helyet látogassa meg, aminek az útba való bevétele nem sérti meg a kapaticitás korlátját, ezek közül pedig mindig a hozzá legközelebb eső helyre menjen. Ezt a szállítási helyet is töröljük S^+-ból. Ha nincs olyan szállítási hely, amit meg tudna látogatni, akkor térjen vissza a raktárba.

Ezután, ha S^+ nem üres és van még olyan kocsink, amit nem használtunk fel, ismételjük meg az előző útvonalválasztást erre a kocsira is.

Ha S^+ nem üres és már minden kocsit felhasználtunk, akkor a következőképpen járjunk el:

Minden megmaradt szállítási helyet szúrjuk be egy jármű útvonalába két szomszédos szállítási hely közé (vagy a raktár és egy szállítás hely közé). Minden szállítási helyet oda szúrjunk be, ahol a legkevesebb ennek a költsége. Majd az így kapott útvonalhoz rendeljük hozzá újra az ügyfeleket.

A $k \in K$ ügyfélnek az r-edik útvonalához ($r \in \{1, 2, 3, ..., V\}$) a hozzárendelési költsége legyen a legkisebb f_{kj} hozzárendelési költség, ahol j az r-edik út egy szállítási helye. Az ügyfeleket úgy rendeljük hozzá az útvonalakhoz, hogy a hozzárendelési költségeik összege a lehető legkevesebb legyen, feltéve, hogy semelyik útvonalon sem lesz a hozzárendelt ügyfelek összigénye nagyobb a kapacitásnál. Másképp fogalmazva azon $k \in K$-ra, amiket az r-edik útvonalhoz rendeltünk $\sum d_k \leq Q$ és ez $\forall r \in \{1, 2, ..., V\}$-re teljesül.

4.4. Oszlopgenerálás

Az oszlopgeneráláshoz modellezük a problémát egészértékű programozási feladatként. A járműirányítási és ügyfélelhelyezési probléma megoldásában a legfeljebb V db szállítási körút keletkezik, amelyek mindegyike a raktárból indul és oda is jutt vissza. Valamint minden szállítási hely legfeljebb egy körben szerepelhet, emiatt ezek a körök diszjunktak. Az ügyfeleket úgy kell elosztani a szállítási helyek között, hogy az egy körben lévő szállítási helyekhez rendelt összes ügyfél igénye ne legyen nagyobb a kapacitásnál.

Készítsünk ezeknek a feltételeknek megfelelő minden körhöz egy oszlopot és ezeknek a halmazát jelöljük H-val. Minden $h \in H$ körhöz jelölje P^h az ügyfelek vektorát, ahol 1-essel jelöljük, ha az adott ügyfél hozzá van rendelve a kör egy szállítási helyéhez és 0-val, ha nem szolgáljuk ki az ügyfélt a kör alatt. Nem szükséges jelölnünk a vektorban, hogy az ügyfél melyik szállítási helyhez van rendelve. Mivel mindig a h vektorban meglátogatott szállítási helyek közül, ahhoz lesz hozzárendelve egy adott ügyfél, amelyikhez a legkisebb a hozzárendelési költsége. Hasonlóan legyen Q^h a h körhöz tartozó szállítási helyek vektora és jelölje ρ^h a h kör célfüggvényértékét. A szállítási helyek sorrendje már fontos, ezért a különböző sorrendeknek különböző vektorokat kell készítenünk, amiknek a célfüggvényértéke különbözőek lesznek.

Ekkor a z^h egészértékű változók segítségével felírható az egészértékű program a probléma megoldására, ahol

$$z^h = \begin{cases} 1, \text{ ha a h kört kiválasztottuk az egyik jármű számára} \\ 0, \text{ egyébként} \end{cases}$$

Ekkor a probléma felírása a következőképpen nézz ki:

$$\min \sum_{h \in H} \rho^h z^h$$

$$\sum_{h \in H} P_k^h z^h = 1 \qquad k = 1, 2, ..., |K| \tag{12}$$

$$\sum_{h \in H} Q_i^h z^h \leq 1 \qquad i = 1, 2, ..., |S| \tag{13}$$

$$\sum_{h \in H} z^h = V \tag{14}$$

$$z \in \{0, 1\} \tag{15}$$

A (12) feltétel garantálja, hogy minden ügyfelet pontosan egyszer szolgáljunk ki. A (13) miatt minden szállítási helyet legfeljebb egyszer látogatunk meg. A (14) feltételben egyenlőtlenségnek kellene szerepelnie, mivel V-nél kevesebb számú járművet is használhatunk, de ha a H halmazban szerepel V darab 0 költségű és csak a raktárból a raktárba menő út (semelyik szállítási helyet sem látogatja meg, és egy ügyfelet sem szolgál ki), akkor egyenlőséget írhatunk. A (15) feltétel a z^h változó bináris tulajdonságát biztosítja.

Az algoritmus során ennek a feladatnak egy részfeladatát tekintjük és keressük az LP-relaxáltjának az optimális megoldását. A feladat mérete túl nagy ahhoz, hogy szimplex algoritmussal oldjuk meg. Ezért csak néhány oszlopon alkalmazzuk a szimplex módszert és a kapott duál értékeket vizsgáljuk, hogy megfelelőek-e a feltételeknek. Ehhez sértő oszlopokat (minimális redukált költségű oszlopok) kell keresnünk, ugyanis ha találunk olyan oszlopot, aminél a duál értékek nem felelnek meg a feltételeknek, akkor a kapott primál megoldásunk sem lehet optimális megoldás. A sértett oszlopokat (vagy közülük néhányat) beveszünk a vizsgált oszlopok közé. Az így kibővített oszlopokon futtatjuk újra a szimplex algoritmust. A sértő oszlopokat először heurisztikus algoritmusok segítségével keressük, amik nem vizsgálják meg az összes lehetséges oszlopot. Ha ezek az algoritmusok nem találtak ilyen oszlopot, akkor egy pontos dinamikus programozáson alapuló algoritmust használunk, ami minden oszlopot megvizsgál. Ha ez a pontos algoritmus se talál megfelelő oszlopot, akkor a kapott megoldásunk optimális. Ezeket az algoritmusak a 4.7. alfejezetben ismertetem.

Algorithm 6 Oszlopgenerálás

```
while Igaz do
    Megoldjuk a részfeladat LP-relaxáltját szimplex módszerrel
    Készítsünk új oszlopokat heurisztikus dinamikus programozás segítségével
    if Találtunk negatív redukált költségű oszlopot (sértő oszlopot) then
        Adjunk a részfeladathoz legfeljebb 20 oszlopot
    if Nem találtunk ilyen oszlopot then
        Futtassuk a részfeladaton a egzakt dinamikus programozási algoritmust
        if Találtunk negatív redukált költségű oszlopot (sértő oszlopot) then
            Adjunk a részfeladathoz legfeljebb 20 oszlopot
        if Nem találtunk ilyen oszlopot then
            A részfeladat optimális megoldását kaptuk
            Algoritmus vége
```

4.5. Szétválasztási stratégia

A járműirányítási problémákban a szétválasztást (a szétválasztás és korlátozás algoritmusban) nem az oszlopváltozókon végezzük, hanem az eredeti probléma változóin. Ha az oszlopváltozókon végeznénk a szétválasztást, akkor a keletkezett két F^1, F^2 részprobléma mérete nagyon különböző lenne. Tegyük fel, hogy az F^1 részfeladat a $z^{h'} = 1$ egyenlőség bevételével keletkezik, az F^2 részfeladat pedig a $z^{h'} = 0$ egyenlőség bevételével. Ekkor F^1 részfeladat egy nagyon erős feltételt kapott, mivel a feladat egyik járművének pontosan meghatároztuk az útvonalát, így eggyel kevesebb jármű útvonlát kell csak meghatároznunk. Azonban az F^2 részfeladat szinte semmivel sem lett könnyebb az eredeti feladatnál, hiszen csak annyit szűkítettünk rajta, hogy a h' kört egyik jármű se járhatja be. Ennek ellenére ugyanazokat a szállítási helyeket más sorrendben vagy más ügyfél hozzárendeléssel megtehetik a járművek. Emiatt nem előnyös az oszlopváltozókon végezni a szétválasztást.

Helyette az eredeti útválasztási vagy hozzárendelési változókon végezzük a szétválasztást. A továbbiakban tegyük fel, hogy a szétválasztandó F feladat LP-relaxáltjának van tört megoldása (ezért akarjuk a szétválasztás és korlátozás algoritmusban két F^1, F^2 részfeladattá szétválasztani). Bármely két $i, j \in S, i \neq j$ szállítási hely közötti útra számoljuk ki az őt használó járművek számát (tört is lehet). Ehhez legyen $H_{i,j}$ az olyan h körök halmaza, ahol a járműből meglátogatja i-t, majd rögtön utána j-t is. Az előbb említett értéke az (i,j) útnak ekkor $\sum_{h \in H_{i,j}} z^h$ lesz. A szétválasztáshoz válasszuk azt az (i,j) élt (i-ből j-be menő utat), ahol ez az érték tört és a legközelebb esik a 0,5-höz.

A szétválasztás során az az egyik részfeladat (i,j) használatának megtiltásával, míg a másik részfeladat a bevételével kelezkezik. A megtiltáshoz a részfeladatban

adjuk a c_{ij}-nek kellően nagy értéket (vagy $+\infty$-t). A másik részfeladathoz pedig az (i,j) él bevételét kell megkövetelnünk. Ehhez az i-ből bármely nem j-be menő útnak adjunk kellően nagy költséget, valamint bármely nem az i-ből j-be menő útnak is adjunk kellően nagy költséget. Így a j szállítási helyet csak az i-ből tudjuk meglátogatni, illetve ha bármikor meglátogatjuk az i szállítási helyet egy járművel, akkor rögtön utána kötelező a j-t is meglátogatnunk.

Előfordulhat, hogy mindegyik utat egész számú (azaz 0 vagy 1 darab) jármű használja. Ekkor az F részfeladatot az ügyfél hozzárendelési változókon végezzük a szétválasztást. Ehhez az előzőhöz hasonlóan készítsük el $\forall k \in K, \forall j \in S$ a $H'_{k,j}$ halmazokat, ami azon oszlopok halmaza, amikben a k ügyfél a j szállítási helyhez van rendelve. Ekkor minden $\forall k \in K$ ügyfélhez és minden $\forall j \in S$ szállítási helyhez számoljuk ki a $\sum_{h \in H'_{k,j}} Z^h$ értéket és válasszuk ki közülük a legtörtebbet (ami a legközelebb van 0,5-höz). Ezen a hozzárendelési változón választjuk szét a feladatot.

Ekkor a szétválasztott F^1, F^2 részfeladatok, úgy keletkeznek, hogy F-nél megköveteljük, hogy a k ügyfél a j szállítási helyhez legyen rendelve, illetve ugyanezt megtiltjuk. Az egyik részfeladatnál a megkövetelés úgy történik, hogy legyen f_{ki} kellően nagy, ha $i \neq j, i \in S$. A másik részfeladatnál a hozzárendelés megtiltásához válasszuk f_{kj} kellően nagynak.

4.6. Árképzési alprobléma

Az árképzési alproblémában az oszlopgeneráláshoz kell létrehoznunk egy minimális redukált költségű megvalósítható útvonalat és a hozzá tartozó ügyfélkiosztást. Ugyanis ha az így elkészített oszlop nem sérti a duális feltételet, akkor semelyik másik oszlop sem sértheti (mert ez volt a legkisebb redukált költségű oszlop). Ehhez a LP-relaxált megoldásából származó duál értékeket használjuk a redukált költség kiszámolásához. Ha az elkészített útnak negatív a redukált költsége, akkor a neki megfelelő oszlop sértő oszlop. Ezért ezt vegyük be az oszlopgenerálás következő lépésében. Ha nem találunk negatív redukált költségű oszlopot, akkor optimális megoldást találtunk a részfeladat LP-relaxáltjára.

4.6.1. Az árképzési alproblémában használt jelölések

Az árképzési alprobléma felírásához a következő jelöléseket vezessük be:

$$x_i = \begin{cases} 1, \text{ ha az } i \text{ szállítási helyet kiválasztjuk a létrehozott oszlopban} \\ 0, \text{ egyébként} \end{cases}$$

$$y_k = \begin{cases} 1, \text{ ha a } k \text{ ügyfelet kiválasztjuk a létrehozott oszlopban} \\ 0, \text{ egyébként} \end{cases}$$

$$e_{ij} = \begin{cases} 1, \text{ ha a } (i,j) \text{ út része a létrehozott útnak} \\ 0, \text{ egyébként} \end{cases}$$

$$s_{ik} = \begin{cases} 1, \text{ ha a } k \text{ ügyfél az } i \text{ helyhez van rendelve a létrehozott oszlopban} \\ 0, \text{ egyébként} \end{cases}$$

$q_i =$ az i helyig kiszállított áru mennyisége

$\mu_k =$ a k ügyfélhez tartozó duális változó

$\pi_i =$ az i szállítási helyhez tartozó duális változó

$\pi_0 =$ a (14) feltételhez tartozó duális változó

A μ a 4.4 fejezetben leírt feladat (12) feltételéhez tartozó duális változója. Hasonlóan, π a (13) feltételhez tartozó duális változó. Jelölje ezen változók egy adott részfeladatban kapott konkrét értékét μ', π' és π'_0.

4.6.2. Az árképzési alprobléma felírása

Az alproblémát a következő egészértékű programozási feladatként írhatjuk fel:

$$\min \sum_{i,j \in S} c_{ij} e_{ij} + \sum_{i \in S} \sum_{k \in K} f_{ik} s_{ik} - \sum_{i \in S} \pi'_i x_i - \sum_{k \in K} \mu'_k y_k - \pi'_0$$

$$\sum_{j \in S} e_{0j} = 1 \tag{16}$$

$$\sum_{j \in N \setminus \{i\}} e_{ij} = x_i \qquad \forall i \in S \tag{17}$$

$$0 = \sum_{i \in N \setminus \{j\}} e_{ji} - \sum_{i \in N \setminus \{j\}} e_{ij} \qquad \forall j \in N \tag{18}$$

$$q_j \geq q_i + \sum_{k \in K} d_k s_{jk} - 2Q(1 - e_{ij}) \qquad \forall i, j \in S, i \neq j \tag{19}$$

$$s_{ik} \leq x_i \qquad \forall i \in S, k \in K \tag{20}$$

$$x_i \leq \sum_{k \in K} s_{ik} \qquad \forall i \in S \tag{21}$$

$$y_k = \sum_{i \in S} s_{ik} \qquad \forall k \in K \tag{22}$$

$$\sum_{k \in K} d_k s_{ik} \leq q_i \leq Q x_i \qquad \forall i \in S \tag{23}$$

$$x_i, y_k, e_{ij}, s_{ik} \in \{0, 1\}, q_i \geq 0 \qquad \forall i, j \in S, k \in K \tag{24}$$

A célfüggvény a létrehozott oszlop redukált költségét minimalizálja. A (16) és (17) feltételek biztosítják, hogy a raktárnak és minden kiválasztott szállítási helynek pontosan egy rákövetkezője legyen. A (18) feltétel miatt minden szállítási helyre és a raktárba pontosan ugyanannyiszor megyünk be és ki is. A (19) feltétel biztosítja, hogy ne több diszjunkt kört akarjunk bejárni, hanem egy darab nagy kört. A (20) feltétel nem engedi, hogy egy ügyfelet olyan szállítási helyhez akarjunk rendelni, ami nincs kiválasztva a körútban. Míg a (21) feltétel megköveteli, hogy minden kiválasztott szállítási helyhez legalább egy ügyfél legyen rendelve. A (22) feltétel garantálja, hogy minden kiválasztott kiválasztott ügyfél pontosan egy szállítási helyhez legyen rendelve, míg a nem kiválasztott ügyfelek egyikhez se. A (23) feltétel miatt minden kiválasztott szállítási helyig a kiszállított áruk összmennyisége nem lépi túl a jármű kapacitását, így a járművön se lesz ennél több áru; valamint alsó korlátként az adott helyre szállítandó áruk mennyiségét adja meg. A (24) feltétel pedig a változók binaritását követeli meg.

Ennek az alproblémának a megoldása NP-nehéz, ugyanis visszavezethetjük a feladatot egy nyereménygyűjtő TSP-re. Az ügyfelek hozzárendelési költségét a szállítási helyekhez megválaszhatjuk úgy, hogy minden ügyfélnek egy helyre 0 legyen a hozzárendelési költsége, a többi szállítási helyre pedig ∞ nagy. Ekkor az alprobléma

ügyfélhozzárendelési részfeladata előre megoldottnak tekinthető. Ezzel a módszer egy tetszőleges nyereménygyűjtő TSP feladatot megfogalmazhatunk az előbbi alakban, így ha ez a feladat megoldható lenne polinomiális algoritmus segítségével, akkor a nyereménygyűjtő TSP is megoldható lenne. Mivel a a nyereménygyűjtő TSP NP-nehéz feladat, így az árképzési alporbléma is NP-nehéz.
Az árképzési alprobléma megoldására egy dinamikus programozási módszer került kidolgozásra.

4.7. Dinamikus programozás az árképzési alproblémára

Az előbb leírt alprobléma megoldását egy címkéző algoritmussal kaphatjuk meg. Ezek a címkék azonosítják a lehetséges útvonalakat, így kiválaszthatjuk közülük a minimális költségűt. Minden címke a raktárból indul és minden lépésben kiterjesztjük a címkéket a még meg nem látogatott csúcsokba. Eközben minden címkénél nyilván kell tartani, hogy mekkora kapacitása maradt még meg a járműnek és a kiterjesztést csak akkor hajthatjuk végre, ha a kapacitás engedi. Egy címke kiterjesztésénél egy szállítási helyet veszünk be a címkébe, ahol előtte még nem járt, és ehhez a szállítási helyhez rendeljük az eddig nem meglátogatott ügyfelek egy részhalmazát. A címkék költségét is nyomon kell követni, ezzel könnyen kiválaszthatjuk az optimális címkét, valamint csökkenthetjük az algoritmus által elkészített címkék számát.. Ezért egy címke a következőképpen fog kinézni:

$$\mathcal{L} = \{P_l, A_l, C_l, Q_l\}$$

Ahol:
$P_l = (0, v_1, v_2, ..., v_l)$ a meglátogatott szállítási helyek rendezett halmaza
$A_l = (\emptyset, a_1, a_2, ..., a_l)$ a meglátogatott szállítási helyekhez rendelt ügyfelek részhalmazainak a rendezett halmaza (tehát a_i a v_i helyhez rendelt ügyfelek halmaza)
$C_l = \sum_{i=0}^{l-1} c_{ii+1} + \sum_{i=1}^{l} \sum_{k \in a_i} f_{ki} - \sum_{i=0}^{l-1} \pi_i - \sum_{i=1}^{l} \sum_{k \in a_i} \mu_k - \pi_0$ az eddig felépített út költsége
$Q_l = \sum_{i=1}^{l} \sum_{k \in a_i} d_k$ a címke által eddig felhasznált kapacitás

Ahhoz, hogy egy címke megfeleljen a feladat feltételeinek, teljesülnie kell a következőknek:

- $Q_l \leq Q$, azaz a szállított áru mennyisége nem lépi túl a jármű kapacitását
- $a_i \cap a_j = \emptyset$ teljesül $\forall i \neq j; i, j \in \{1, 2, ..., l\}$, azaz az egyes helyekhez rendelt ügyfelek részhalmazai diszjunktak
- P_l-ben semelyik szállítási hely sem szerepel egynél többször.

Az árképzési alprobléma célja, hogy minimális redukált költségű utat (oszlopot) találjunk meg. Emiatt ne vegyünk be az olyan ügyfeleket illetve szállítási helyeket, amik pozitívan növelnék a redukált költségét az útnak. Mivel, ha bevennénk őket, akkor készíthetnénk egy hozzá hasonló útvonalat, ahol csak annyit változtatnánk, hogy egy helyet vagy egy ügyfelet kihagyunk az útból. Ez az új címke is megfelelő lesz a feladat feltételeinek, de a redukált költsége kevesebb lesz. Emiatt elég csak olyan szállítási helyekkel és ügyfelekkel foglalkozni, amiknek a hozzáadása egy címkéhez negatívan csökkentheti a redukált költséget.

4.7.1. Jelölések a címkéző algoritmushoz

Az előbb említett redukált szállítási helyek és ügyfelek részhalmazait jelöljük S^R, K^R-rel.
$S^R = \{s \in S \mid -\pi'_s + \sum_{k \in K} \min\{0, f_{ks} - \mu'_k\} < 0\}$
$K^R = \{k \in K \mid \min_{s \in S}\{f_{ks} - \mu'_k\} < 0\}$

További, a dinamikus programozási algoritmusban a címkék kiterjesztésének megkönnyítéséhez használt jelölések:

$\sigma_l =$ Az $\mathcal{L}$ címke által eddig meglátogatott szállítási helyek halmaza
$\sigma'_l =$ Azon helyek halmaza, amik elhanyagolhatóak az $\mathcal{L}$ címke kiterjesztésénél
$\chi_l =$ Az $\mathcal{L}$ címke által eddig kiszolgált ügyfelek halmaza
$\chi'_l =$ Azon ügyfelek halmaza, amik elhanyagolhatóak az $\mathcal{L}$ címke kiterjesztésénél

Az árképzési alproblémában azt mondjuk, hogy egy $\mathcal{L}^*$ címke dominálja az $\mathcal{L}$ címkét, ha a következőek teljesülnek rá:

$$C_{l^*} \leq C_l$$
$$Q_{l^*} \leq Q_l$$
$$\sigma_{l^*} \subseteq \sigma_l \cup \sigma'_l$$
$$\chi_{l^*} \subseteq \chi_l \cup \chi'_l$$

További jelölések:

$H_v =$ Olyan nem dominált címkék halmaza, amiknek az utolsó szállítási helye v

$F_v =$ A v-vel kibővített címkék halmaza

$\overline{H_v} =$ A H_v-ben lévő olyan címkék részhalmaza, amik még nem lettek kibővítve

$\Gamma =$ Azon v helyek halmaza, ahol van olyan címke, ami még nem lett kiterjesztve

4.7.2. Címkéző algoritmus leírása

Az előbb leírt címkék kiterjesztésének menetét a 7. algoritmus írja le.

Algorithm 7 Címkéző algoritmus

$H_v = \emptyset \quad \forall v \in S$
$H_0 = \overline{H_0} = \{((0), (\emptyset), 0, 0)\}$
$\Gamma = \{0\}$
while $\Gamma \neq \emptyset$ **do**
  Válasszunk egy u helyet Γ-ból és $\Gamma = \Gamma \setminus \{u\}$
  for minden $v \in S^R$-re **do**
    $F_v = \emptyset$
    for minden $\mathcal{L} \in \overline{H_u} \mid v \notin (\sigma_l \cup \sigma'_l)$ **do**
      $F_v = F_v \cup$ CÍMKÉK($\mathcal{L}$,v)
    $F_v =$HOZZÁAD(H_v, F_v)
    if $F_v \neq \emptyset$ **then**
      $H_v = H_v \cup F_v$
      $\overline{H_v} = F_v \cup \overline{H_v}$
      $\Gamma = \Gamma \cup \{v\}$
  $\overline{H_u} = \emptyset$

Az algoritmus először készít egy kezdeti címkét a raktárban (0-val jelölt csúcsban) és ezt hozzáadjuk a Γ aktív helyeinek halmazába. Majd kiválaszt egy csúcsot Γ-ban és minden $\overline{H_u}$-ban lévő címkét kiterjeszt minden lehetséges új csúccsal. Ehhez végigmegy az S^R halmaz minden v pontján, majd minden $\mathcal{L} \in \overline{H_v}$ címkéhez hozzáadja a v csúcsot (feltéve, hogy a kiterjesztés lehetséges, azaz $v \notin (\sigma_l \cup \sigma'_l)$).
Egy adott $\mathcal{L}$ címkének egy adott v csúccsal való kiterjesztése többféle is lehet, mert a v csúcshoz különböző ügyfélrészhalmazokat is rendelhetünk. Ez exponenciálisan sok címkét eredményezne, de ezek közül sok címkénél előre lehet tudni, hogy lesz nála jobb redukált összköltségű címke (azaz dominálva lesz egy másik címkével). Ezért nem készítjük el az összes címkét, csak bizonyos szabályoknak megfelelő ügyfélrészhalmazokat rendeljük hozzá a v helyhez az $\mathcal{L}$ címke kiterjesztésénél. Ezeknek a szabályoknak megfelelő címkéket készíti el a CÍMKÉK($\mathcal{L}, v$) függvény.

Az így elkészített címkék F_v-be kerülnek. H_v-ben vannak azok a címkék, amiket egy korábbi lépés eredményeként készítettünk, tehát valamikor korábban voltak v-vel kibővítve. A két halmaz uniójában lehetnek olyan címkék, amelyek közül az egyik dominálja a másikat. Ez azt jelenti, hogy ha ezt a két címkét hasonló módon kiterjeszthetnénk a továbbiakban, akkor biztosak lehetünk abban, hogy az egyik címkének kisebb lesz a redukált költsége az algoritmus végén. Emiatt azt a címkét, ami a végén rosszabb redukált költséget eredményezne, felesleges a későbbiekben kiterjesztenünk, ezért el is hagyhatjuk. Ekkor mondhatjuk azt, hogy az egyik címke dominálja (biztosan jobb redukált összköltsége lesz) a másik címkét.
Például ilyen dominanciáról beszélünk, ha van két címke, amelyek pontosan ugyan-

azokat a helyeket látogatják meg és pontosan ugyanazokat az ügyfeleket szolgálják ki, de az egyiknek kisebb az addig felépített C_l költsége, mint a másiknak. Ekkor a kisebb költségű címke dominálja a másikat, mert akárhogyan terjesztjük ki az egyik címkét új szállítási helyekkel és ügyfelekkel, ugyanazt a kiterjesztést alkalmazhatjuk a másik címkére is. Ekkor a kiterjesztett címkék költségei ugyanannyival nőnek mindkét esetben, vagyis az algoritmus végén is a domináló címkének lesz kevesebb a redukált költsége. Ezért a dominált címkével felesleges a továbbiakban foglalkoznunk.
A dominált címkéket emiatt törölhetjük a F_v halmazból. Ezt a műveletet végzi az HOZZÁAD(H_v, F_v) függvény (Az F_v halmazbeli címkéket dominálhatják H_v-beli címkék, ezért szükséges a HOZZÁAD függvénynek mindkét halmazt megkapnia).

Ha az F_v halmazban van nem dominált címke, tehát sikerült új címkét gyártanunk, amivel foglalkozni kell a továbbiakban, akkor ezeknek a címkéknek be kell a kerülniük a H_v és $\overline{H_v}$ halmazokba is. Valamint a v helyet fel kell vennünk a Γ aktív csúcsok listájára (lehetséges, hogy eddig is benne volt), mivel itt lettek (vagy alapból is voltak) kiterjesztésre váró címkék.

Ha a $\overline{H_u}$ halmazban lévő összes címkét kiterjesztettünk az összes lehetséges módon, akkor a halmazt változtassuk üreshalmazra, mivel jelenleg nincs több kiterjesztendő címke ebben a csúcsban (az u csúcsot már a while ciklus elején töröltük Γ-ból). Az algoritmus során a későbbiekben még kerülhet a $\overline{H_u}$ halmazba új címke, ezáltal az u csúcs visszakerülhet az aktív csúcsok listájára.

Ezt az eljárást addig folytatjuk, amíg a Γ üres nem lesz, ezáltal nem lesz több kiterjesztendő címke. Az algoritmus biztosan véges, mert minden lépésben a címkék maradék kapacitása csökken, emiatt egy címkét nem lehet túl sokszor kiterjeszteni, csak amíg a kapacitás korlát engedi. Az algoritmus végén kapott címkék közül ki kell választani azt az $\mathcal{L}^*$ címkét, aminek a $C_{l^*} + c_{l^*0}$ költsége a legkisebb. Ennek a címkének megfelelő útvonal és ügyfélkiválasztás és hozzárendelés lesz az optimális megoldása az árképzési alproblémának.

A HOZZÁAD(H_v, F_v) függvény a domináló címkék eldöntésére a következő segédalgoritmust használja, ami két $\mathcal{L}, \mathcal{L}^*$ címkéről eldönti, hogy az $\mathcal{L}^*$ címke dominálja-e az $\mathcal{L}$ címkét:

Algorithm 8 Dominancia($\mathcal{L}, \mathcal{L}^*$)

if $C_l < C_{l^*}$ **then**
  return FALSE
if $Q_l < Q_{l^*}$ **then**
  return FALSE
if $(\sigma_{l^*} \subseteq (\sigma_l \cup \sigma'_l))$ és $(\chi_{l^*} \subseteq (\chi_l \cup \chi'_l))$ **then**
  return TRUE
return FALSE

4.7.3. Ügyfélrészhalmazok készítése

Egy $\mathcal{L}$ címke v szállítási hellyel való kiterjesztésében játszik szerepet a CÍMKÉK($\mathcal{L}$,v) függvény, ami a v-hez rendelhető ügyfélrészhalmazokat határozza meg. Ez a függvény fontos szerepet játszik abban, hogy a címkék száma ne nőjön túl nagyra. Ezért nem készít el olyan címkéket, amik biztosan dominált címkévé válnának.

Legyen $S' = S^R \setminus (\sigma_l \cup \sigma'_l)$. Valamint legyen $K' = K^R \setminus \chi_l$ halmaz, ami azon ügyfelek halmaza, akiket az $\mathcal{L}$ címkében a v helyhez rendelhetünk a kiterjesztésnél. A CÍMKÉK($\mathcal{L}$,v) függvény ennek a K' halmaznak választja ki az összes részhalmazát, amiket a v szállítási helyhez rendelve az $\mathcal{L}$ címke kapacitása nem lépi túl a korlátot. Valamint használja a következő három szabályt, amivel el tudja kerülni a biztosan dominált címkék elkészítését.

1. szabály: A nem előnyös ügyfelek elkerülése
Minden olyan $k \in K'$ ügyfelet töröljünk ki K'-ből, akikre $f_{kv_l} - \mu'_k \geq 0$, mivel ezen ügyfelek v_l szállítási helyhez való hozzárendelése rontja a redukált összeköltséget.
Továbbá minden olyan $k \in K'$ ügyfelet is távolítsunk el a K'-ből, akikre létezik olyan $v_i \in \sigma_l$ már meglátogatott szállítási hely, amire $f_{kv_l} \geq f_{kv_i}$. Erre azért van szükség, mert ha egy $\mathcal{L}'$ címkében a k ügyfelet a v_l helyhez rendelnénk, akkor készíthetnénk egy hozzá hasonló címkét is, ami csak annyiban különbözne $\mathcal{L}'$-től, hogy a k ügyfelet v_l helyett v_i-hez rendelnénk. Ennek a két címkének ugyanannyi a kapacitása, ugyanazokat a szállítási helyeket látogatják meg és ugyanazokat az ügyfeleket szolgálják ki. Mégis az új címkének kisebb lenne a költsége a $\mathcal{L}'$ címke költségénél, tehát dominálná őt.

2. szabály: Az eddig nem kiszolgált ügyfelek hozzárendelése, akiknek a v szállítási hely a legkedvezőbb
Legyen Ψ azon $k \in K'$ ügyfelek halmaza, akiknek a legkedvezőbb szállítási helyük az S' halmazban a v hely (azaz a $\min_{s \in S'} f_{ks}$ érték a v helyen veszi fel a minimumát). Ekkor a Ψ-beli ügyfelek vagy v-hez lesznek rendelve, vagy nem fogja őket kiszolgálni

a felépített címke. Ugyanis ha egy v-nél későbbi szállítási helyhez lenne egy ilyen k ügyfél rendelve, akkor dominált címkét kapnánk. Mivel ezt a címkét dominálná egy olyan címke, ami csak annyiban különbözik tőle, hogy k-t v-hez rendelnénk. Ennek a dominálóé címkének kedvezőbb redukált költsége lesz.

Legyen Ω az olyan Ψ-beli o ügyfelek halmaza, amikhez már létezik olyan k ügyfél a v_i szállítási helyhez rendelve az $\mathcal{L}$ címkében, amikre: $d_k > d_o$ és $f_{kv_i} - \mu_k \geq f_{ov} - \mu_o$. Az ilyen Ω-beli ügyfeleket mindenképp hozzá kell rendelnünk v-hez, különben dominálva lenne a címke. Ugyanis, ha van olyan $o \in \Omega$ ügyfél, akit nem rendelünk hozzá a v szállítási helyhez, akkor készíthetünk egy őt dominálóé címkét. Mivel az o ügyfél benne van Ω-ben, ezért létezik egy k_o ügyfél a v_i helyhez rendelve a címkében, akinek nagyobb a rendelési igénye, de rosszabb redukált költséggel tudjuk bevenni a címkébe, mint az o ügyfelet. Ekkor ehhez a címkéhez készíthetünk egy hasonló címkét, ahol annyit változtatunk, hogy a k_o ügyfelet nem szolgáljuk ki sehol, de az o ügyfelet kiszolgáljuk a v helyen. Ez az új címke is megfelelő lesz a feltételeknek, de jobb költséget fog eredményezni az előző címkénél.

Ezért tehát minden Ω-beli ügyfelet hozzá kell rendelnünk a v szállítási helyhez az $\mathcal{L}$ címke kiterjesztésénél. Ha nem tudjuk az összes Ω-beli ügyfelet hozzárendelni a kapacitáskorlát miatt, akkor az előbbi meggondolás miatt ez a címke dominálva lesz egy másik címke által, ezért a továbbiakban ezt a címkét nem kell figyelembe vennünk.

3. szabály: Az úgynevezett rosszabb ügyfelek felismerése

Azt mondjuk, hogy egy $k \in K'$ ügyfél rosszabb az $o \in \Psi$ ügyfélnél a v helyen, ha teljesül rájuk, hogy $d_k > d_o$ és $f_{kv} - \mu_k \geq f_{ov} - \mu_o$. Ekkor ha egy o ügyfél nincs hozzárendelve a v helyhez, akkor semelyik nála rosszabb k ügyfél sem lehet a v-hez rendelve ebben a címkében. Ellenkező esetben a két ügyfelet megcserélve dominálóé címkét tudnánk gyártani, tehát az ilyen hozzárendelés dominált címkét eredményezne.

Ezért érdemes minden $o \in \Psi$ ügyfélre meghatározni a nála rosszabb ügyfelek halmazát. Ezzel a lépéssel csökkenthetjük a megvalósítható ügyfélrészhalmazok számát, amikor egy címkét egy adott hellyel ki szeretnénk bővíteni.

Ezekre a szabályokra figyelve készítjük el egy $\mathcal{L}$ címke kiterjesztéseit egy új v hellyel. Először a Ω-ban lévő minden ügyelet biztosan hozzárendeljük a v helyhez. Így a címke maradék kapacitása $Q-(Q_l+\sum_{k\in\Omega} d_k)$ lesz. Ezután a K' összes részhalmazát meghatározzuk úgy, hogy az ügyfelek összesített igénye ne haladja meg ezt a $Q - (Q_l + \sum_{k\in\Omega} d_k)$ számot. Továbbá a részhalmazok készítésénél figyeljünk arra, hogy ha egy $o \in \Psi$ ügyfél nem eleme a részhalmaznak, akkor semelyik nála rosszabb ügyfél se legyen része a részhalmaznak. Ezután minden elkészített $a_{l'}$ részhalmazhoz készítünk egy új $\mathcal{L}'$ címkét, ami az $\mathcal{L}$ címke kiterjesztése a v hellyel.

Ekkor az $\mathcal{L}'$ címke a következő lesz:

$$
\begin{aligned}
P_{l'} &= (0, v_1, v_2, ..., v_l, v) \\
A_{l'} &= (\emptyset, a_1, a_2, ..., a_l, a_{l'} \cup \Omega) \\
C_{l'} &= C_l + c_{v_l v} - \pi_v + \sum_{k \in a_{l'} \cup \Omega} (f_{kv} - \mu_k) \\
Q_{l'} &= Q_l + \sum_{k \in a_{l'} \cup \Omega} d_k
\end{aligned}
$$

Szükség van még a $\chi_{l'}, \chi'_{l'}$ és $\sigma_{l'}, \sigma'_{l'}$ halmazok frissítésére. Ezért ekkor az $\mathcal{L}'$ címke által meglátogatott szállítási helyek és kiszolgált ügyfelek részhalmazai a $\sigma_{l'} = \sigma_l \cup \{v_l\}$ és a $\chi_{l'} = \chi_l \cup (a_{l'} \cup \Omega)$. A $\sigma'_{l'}$ azon helyek halmaza, amik ugyan nincsenek meglátogatva a címke által, de mégse vehetők figyelembe a címke kiterjesztésénél. Ugyanis ha hozzáadnánk a címkéhez dominált címkét kapnánk. Ha van az $\mathcal{L}'$ címkében olyan k ügyfél az s szállítási helyhez rendelve és létezik olyan $\overline{s}$ szállítási hely, amikre $f_{k\overline{s}} < f_{ks}$, akkor az $\overline{s} \in \sigma'_{l'}$. Különben a kisebb hozzárendelési költség miatt, ha a k az s helyett a $\overline{s}$ rendelnénk domináló címkét tudnánk gyártani. Hasonlóan konstruálhatjuk meg a $\chi'_{l'}$ halmazt, ami azon ügyfelek halmaza, akik ugyan nincsenek egy szállítási helyhez se rendelve az $\mathcal{L}'$ címkében, mégse kell őket figyelembe vennünk a címke kiterjesztésénél, mert használatuk dominált címkét eredményezne. A második szabálynál készített Ψ halmaz, olyan ügyfelek halmaza volt, akikhez az ebben a lépésben bevett v szállítási hely volt a legközelebb. Ha őket bármelyik későbbi szállítási helyhez rendelnénk a v helyett, akkor dominált címkét kapnánk. Emiatt Ψ-ben a jelenleg nem használt ügyfeleket a későbbiekben se használhatjuk. Tehát legyen $\chi'_{l'} = \chi'_l \cup (\Psi \setminus (a_{l'} \cup \Omega))$.

4.7.4. Heurisztikus algoritmusok az árképzési alproblémára

A következőkben leírt heurisztikus algoritmusok a 2. algoritmus hatékonyságát befolyásolják. Ezek gyorsabb megoldást biztosíthatnak az árképzési alproblémára, mint a fentiekben leírt egzakt dinamikus programozási módszer, de nem garantálják, hogy találnak sértő oszlopot. Elképzelhető, hogy a heurisztikus algoritmusokkal nem találunk megfelelő megoldást, de a pontos algoritmus segítségével megtaláljuk a megfelelő oszlopot. Ezért előbb ezekkel a gyorsabb heurisztikus algoritmussal kezdjük a keresést, és ha nem járunk velük sikerrel, csak akkor hívjuk meg a pontos címkéző algoritmust.

1. heurisztika: Ebben az algoritmusban minden ügyfelet előre hozzárendelünk a hozzá legközelebbi szállítási helyhez. Ezáltal az árképzési alprobléma egy kapacitásos szállításszervezési (Capacited Vehicle Routing Problem) problémára redukálódik. Használjuk az így leegyszerűsített feladatra a címkéző algoritmust relaxált dominan-

cia szabályokkal és ezeket a dominancia szabályokat egyszerűsítsük is le. Itt egy $\mathcal{L}^*$ címke pontosan akkor dominálja egy másik $\mathcal{L}$ címkét, ha $C_{l^*} \leq C_l$ és $Q_{l^*} \leq Q_l$.

2. heurisztika: Az ügyfeleket itt is előre rendeljük hozzá legközelebbi szállítási helyükhöz. Itt is alkalmazzuk a relaxált dominancia szabályokat, de nem egyszerűsítsük le őket. Tehát ez a heurisztikus algoritmus csak annyiban különbözik az előzőtől, hogy az egyszerűsített dominancia szabályok helyett az eredeti szabályokat alkalmazzuk.

3. heuriszta: Itt is az első heurisztikus algoritmushoz hasonlóan az egyszerűsített dominancia szabályokat relaxáljuk. Azonban az első algoritmussal ellentétben most nem rendeljük hozzá előre az ügyfeleket a szállítási helyekhez. Ha $\mathcal{L}$ címkét szeretnénk kibővíteni egy v hellyel, ekkor a 3. heurisztikus algoritmus a következőképpen készíti el a v-hez rendelhető ügyfélrészhalmazokat:
Természetesen itt is figyelembe veszi, hogy csak olyan ügyfélrészhalmazokat hozzon létre, amikkel együtt a címke összesített rendelési igénye nem lépi túl a jármű kapacitását. Ezután elkészíti a pontos algoritmus CÍMKÉK függvényének 2. szabályában említett Ψ halmazt. Ez azon ügyfelek részhalmaza a $K' = K^R \setminus \chi_l$ halmazban, akiknek a legkedvezőbb szállítási helyük a v. Valamint a CÍMKÉK függvényben használt 3 szabályt alkalmazzuk itt is.
Ezután minden $k \in \Psi$ ügyfélre számoljuk ki a $\frac{f_{kv} - \mu_k}{d_k}$ értéket. Ez a hányados tekinthető egy mérőszámnak,ami annak a mértéke, hogy ha a k ügyfelet kiszolgáljuk a v helyen, akkor a járművön elfoglalt 1 kapacitásegységnyi áru után mennyivel csökken a címke redukált költsége. Minél kisebb ez a hányados, annál vonzóbb az ügyfél kiszolgálása a v helyen. Ezért a hányados értéke szerint állítsuk az ügyfeleket növekvő sorrendbe, legyen ez a sorrend $\{k_{[1]}, k_{[2]}, ..., k_{[|\Psi|]}\}$. Ennek megfelelően készítsük el a címkéket, úgy hogy az első címkénél a v helyhez csak a $\{k_{[1]}\}$ ügyfelet rendeljük. A második címkénél már rendeljük a v-hez a $\{k_{[1]}, k_{[2]}\}$ ügyfeleket. Ugyanígy folytatva az i. elkészített címkénél a hozzárendelt ügyfelek részhalmaza legyen $\{k_{[1]}, k_{[2]}, ..., k_{[i]}\}$. Ezt addig folytassuk amíg a kapacitáskorlát engedi az ügyfelek kiszolgálását a címkénél. (Az utoljára (j-edikként) elkészített címke a $j = \arg\min\{h | \sum_{i=1}^{h-1} d_{k_{[i]}} > Q - Q_l\}$)

4. heurisztika: Ebben a heurisztikus algoritmusban egy címke kiterjesztésénél ugyanúgy képezzük az új szállítási helyhez rendelendő részhalmazokat, mint a 3. heurisztikus algoritmusban. Azonban ebben az esetben nem az egyszerűsített dominancia szabályokat alkalmazzuk, hanem a pontos dominancia szabályokat.

5. heurisztika: Ebben a heurisztikus algoritmusban is a pontos dominancia szabályokat fogjuk alkalmazni. A harmadik heurisztikus algoritmusban készített $\{k_{[1]}, k_{[2]}, ..., k_{[|\Psi|]}\}$ sorrendjét fogjuk használni az ügyfeleknek. Nézzük ennek azt a

$\{k_{[1]}, k_{[2]}, ..., k_j\}$ részhalmazát, ami még nem lépi túl a kapacitáskorlátot, de a $j+1$-re készített ilyen halmaz már túllépné (azaz $j = \arg\min\{h | \sum_{i=1}^{h-1} d_{k_{[i]}} > Q - Q_l\}$). Ekkor egy címke kiterjesztésénél az új helyhez rendelendő ügyfélrészhalmaznak a $\{k_{[1]}, k_{[2]}, ..., k_j\}$ halmaz összes részhalmazát figyelembe veszik.

Ezek a heurisztikus algoritmusok nem garantálják, hogy megtalálják a keresett sértő oszlopot. Ugyanakkor ha találnak sértő oszlopot (ami nem feltétlenül a legkisebb redukált költségű az oszlopk között), akkor azt lényegesen gyorsabban teszik, mint a egzakt dinamikus programozási algoritmus. Emiatt, ha a heurisztikus algoritmusok futása nem járt sikerrel, kénytelenek vagyunk meghívni a problémára a egzakt algoritmust is, hogy biztosak legyünk abban, hogy a korlátozás és árképzési algoritmus végén kapott megoldásunk optimális.

5. Összefoglalás

A szakdolgozatomban összegyűjtöttem több szállításszervezési problémát. Ezek között vannak, amik visszavezethetők egymásra, de mégis érdemes ezekkel külön is foglalkozni. Mivel az egyszerűbb problémákra más algoritmusok lehetnek hatékonyak, mint az általánosított problémákra.

Igyekeztem a korlátozás és szétválasztás módszerének különböző algoritmusokban használt alkalmazásait bemutatni. Az ehhez szükséges egészértékű programozási feladatként való felírására évek során ezeknek a problémáknak számos megoldása született.
A szakgoldozat folytatásaként érdemesnek tartom ezeket a felírásokat vizsgálni és a gyakorlatban összehasonlítani, hogy a korlátozás és szétválasztás módszerével mennyire hatékony algoritmusok írhatók fel a segítségükkel.

Meg szeretném említeni, hogy a legtöbb megjelent cikkben a korlátozás és szétválasztás mellett több más megoldási módszer is megjelent az algoritmusokban. Különösen érdekes volt számomra az egyik ilyen probléma megoldásában dinamikus programozással is találkozni.

6. Hivatkozások

[1] J. E. Beasley and E. M. Nascimento. The vehicle routing-allocation problem: A unifying framework. *Top*, 4:65–86, 1996.

[2] Nicos Christofides. Worst-case analysis of a new heuristic for the traveling salesman problem. *Carnegie Mellon University*, page 10, 02 1976.

[3] H. Crowder and M. Padberg. Solving large-scale symmetric traveling salesman problems to optimality. *Management Science*, 26:495–509, 1980.

[4] G. B. Dantzig and J. H. Remser. The truck dispatching problem. *Management Science*, 6:80–91, 1958.

[5] Lars Engebretsen and Marek Karpinski. Tsp with bounded metrics. *Journal of Computer and System Sciences*, 72(4):509 – 546, 2006.

[6] R. E. Gomory and T. C. Hu. Multi-terminal network flows. *Journal of the Society for Industrial and Applied Mathematics*, 9:551–570, 1961.

[7] Luis Gouveia. A result on projection for the vehicle routing ptoblem. *European Journal of Operational Research*, 85:610–624, 1995.

[8] Michiel A. J. Uit het Broek, Albert H. Schrotenboer, Bolor Jargalsaikhan, Kees Jan Roodbergen, and Leandro C. Coelho. Valid inequalities and a branch-and-cut algorithm for asymmetric multi-depot routing problems. Technical report, CIRRELT, 2019.

[9] Marek Karpinski, Michael Lampis, and Richard Schmied. New inapproximability bounds for tsp. *J. Comput. Syst. Sci.*, 81:1665–1677, 2015.

[10] G. Laporte and Y. Nobert. A branch and bound algorithm for the capacitated vehicle routing problem. *OR Spektrum*, 5:77–85, 1983.

[11] Adam Letchford and Juan Jose Salazar Gonzalez. Stronger multi-commodity flow formulations of the capacitated vehicle routing problem. *European Journal of Operational Research*, 244:730–738, 2015.

[12] D. Naddef and G. Rinaldi. *The Vehicle Routing Problem*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2001.

[13] G. L. Nemhauser and L. A. Wolsey. *Integer and combinatorial optimizatio*. John Wiley & Sons, Canada, 1988.

[14] M. Padberg and M.R. Rao. Odd minimum cut-sets and b-matchings. *Mathematics of Operations Research*, 7:67–80, 1982.

[15] Manfred Padberg and Giovanni Rinaldi. Facet identification for the symmetric traveling salesman polytope. *Mathematical Programming*, 47:219–257, 1990.

[16] Manfred W. Padberg and Giovanni Rinaldi. A branch-and-cut algorithm for the resolution of large-scale symmetric traveling salesman problems. *Society for Industrial and Applied Mathematics*, 33:60–100, 1991.

[17] Mohammad Reihaneh and Ahmed Ghoniem. A branch-and-price algorithm for a vehicle routing with demand allocation problem. *European Journal of Operational Research*, 272:523–538, 2019.