Csomagküldési szolgáltatás útvonal-optimalizálása

Oláh Sarolta
Témavezető: Jüttner Alpár

2023. május 27.

1. A feladat

Az önálló projektemben a járműirányítási feladat (VRP - Vehicle Routing Problem) egy speciális esetével, egy csomagküldési szolgáltatással foglalkoztam.

A járműirányítási feladat olyan problémákra keres megoldást, melyekben a cél termékek szállítása egy, vagy akár több raktártól a vásárlókig.

A csomagküldési szolgáltatásnál minden termékhez tartozik egy feladó (forrás) és egy vásárló (nyelő), az útvonalakat ilyen forrás-nyelő párok között kell megfelelően megadni (természetesen egy termék vásárlóját nem látogathatjuk meg hamarabb, mint a feladót). A járművek száma korlátozott. Minden jármű egy egyedi "kocsiszínből" indul és a kérések teljesítése után oda is tér vissza. Célunk többféle lehet. Akár az útvonalak költségének minimalizálása, feltéve, hogy minden csomagot kiszállítunk, akár a kiszállított csomagok számának maximalizálása. Utóbbi esetben nem ragaszkodunk az összes csomag kiszállításához.

A problémát egy gráfon szemléltethetjük. A feladók, a vásárlók és az autók kiinduló pontjai lesznek a gráf csúcsai, a közöttük menő utak lesznek a gráf élei. Az élekhez adott egy költség, ez a két csúcs közötti utazási költség, ami lehet a távolság, vagy akár az idő.

2. Megoldási módszerek

Bevezetünk egy x_{ijk} változót, ami pontosan akkor 1, ha a megoldásban az i csúcsból a j csúcsba megyünk a k járművel. Legyen t_i az i csúcs kiszolgálásának időpontja, és jelölje σ_l az l-edik forrás, τ_l az l-edik nyelő csúcsot. Ekkor a következő modellt írhatjuk fel:

$$\min \sum_{i,j,k} c_{ij} x_{ijk} \tag{1}$$

$$t_j - t_i \geq 1 - M(x_{ijk}) \qquad \forall\, k \in K, (i,j) \in A \tag{2}$$

$$\sum_j x_{\sigma_l jk} - \sum_i x_{i\tau_l k} = 0 \qquad \forall\, k \in K, (\sigma_l, \tau_l) \text{ párra} \tag{3}$$

$$t_{\sigma_l} \leq t_{\tau_l} \qquad \forall (\sigma_l, \tau_l) \text{ párra} \tag{4}$$

$$\sum_{j,k} x_{ijk} = 1 \qquad \forall\, i \in V \tag{5}$$

$$x_{ijk} \in \{0,1\} \qquad \forall\, k \in K, (i,j) \in A. \tag{6}$$

A felírt modell azt az esetet mutatja, mikor minden követelést teljesítünk és az utak költségén minimalizálunk. Ezt a minimalizálást hajtjuk végre (1)-ben, és (5) garantálja, hogy minden követelést teljesítsünk. (2) biztosítja, hogy, ha használjuk az i-ből j-be menő élet, akkor az i-t előbb is kell kiszolgálnunk, mint a j-t. Mivel a járművek a csomagokat útközben veszik fel, nagyon fontos, hogy előbb menjenek a felvevő pontba, mint a vásárlóhoz, és, hogy, ha a felvevő pontba elmentek, akkor azt a vásárlót is meglátogassák. Ezeket a (3)-(4) feltételekkel tudjuk teljesíteni. (6)-ban rögzítjük, hogy a változó bináris.

A változók száma csökkenthető, ha a járművekre nem tartunk fent külön indexelést. Előjön ekkor a probléma, hogy hogyan különböztessük meg az utakat. Egyik megoldás erre, ha a járműveket különböző időpontokban indítjuk és korlátozzuk utazási idejüket. Így, ha az indulási idők kellő távolságra vannak a visszaérkezésektől, akkor az, hogy az adott csúcsot melyik jármű szolgálja ki, meghatározható az elérésének időpontjából.

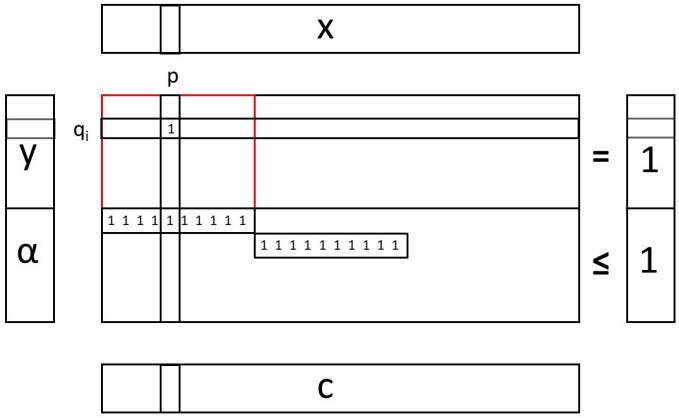
2.1. IP

A feladat megoldásának egyik módja, a modell közvetlen leprogramozása és egy IP-solver használata. C++-ban, Lemon és Cplex segítségével ezt én is megtettem. Sajnos sok csomag és nagy feladatok esetén a program már megengedett megoldást is nehezen talál, sőt, kisebb értékekre is lassan fut le, ez az egészértékű programokra általánosan is jellemző hátrány. Használata a gyakorlatban nem célravezető.

Ha a feladatból elengedjük az egészértékűségre vonatkozó feltételt, a program lényegesen gyorsabban megold több száz csomagos problémákat is. A gyorsaságért cserébe viszont igen távol kerülhetünk az optimális megoldástól.

2.2. Oszlopgenerálás

Másik megoldás, oszlopgenerálás [1] alkalmazása. Itt nem tároljuk az egész feladatot, csak egy részét, és erre a részfeladatra határozzuk meg az optimumot. Az így kapott eredmény nem feltétlenül optimális megoldása az eredeti feladatnak, ellenőriznünk kell rá a duál-megengedettséget. Ha a megoldásunk nem duál-megengedett, akkor bővítjük a részfeladatot, a feladat mátrixának egy duált sértő változóhoz tartozó oszlopával, és újból megoldjuk.



A fenti ábra szemlélteti a feladathoz tartozó oszlopgenerálós modellt. A mátrix oszlopai a mostani problémában egy jármű útvonalának, pontosabban a kiszállított csomagjainak, a sorok a megfelelő csomagoknak felelnek meg. A pirossal kijelölt részen az első jármű útvonalait látjuk. A kijelölt rész alatt, az egyesek biztosítják, hogy egy járműhöz legfeljebb egy útvonalat rendeljünk hozzá. Egy x_p változó pontosan akkor lesz 1, ha az adott útvonalat bevesszük a megoldásba.

Az algoritmus elindításához szükséges egy jó kiindulás. A kezdeti megoldás, hogy minden csomaghoz hozzárendelünk egy járművet. Mivel a valóságban annyi járművünk (beállítástól függően) nagy valószínűséggel nincs, ezért ezekre segédjárműként tekintünk és nagy költésget adunk hozzájuk, hogy a feladat lehetőleg ne használja. Azért, hogy az α-hoz tartozó sorok kezdetben ne csupa 0-ból álljanak, hozzáveszünk még néhány megengedett útvonalat a tényleges járműveinkkel.

Ahhoz, hogy egy megoldás duál-megengedettségét ellenőrizzük, egy úgynevezett árazási feladatot kell megoldanunk. Ez a mostani feladatunkban azt jelenti, hogy keresnünk kell egy olyan útvonalat, amire a $(c(p) - \sum y(q_i))$ minimális ($y(q_i)$ az úton kiszállított termékhez tartozó duális változó). Egy útvonal akkor lesz sértő, ha ez a minimum kisebb, mint $-\alpha(k_j)$, ahol k_j jelöli, hogy az útvonalat melyik járművel tesszük meg, α az ő sorához tartozó duális változó.

A fent leírt feladat tehát az egy autós kiszállítás. Az implementálásnál két megközelítést használtam.

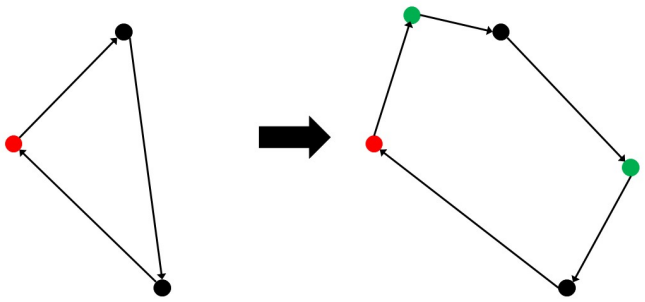
2.2.1. IP, egy jármű

Az első módszer felhasználja a korábbi IP modellt. Kis átalakításokkal ezzel az új költségfügvénnyel számol minimális utakat. Sajnos a módszer örökölte a korábbi IP megoldó lassúságát, igazán jó és hatékony eredményt nem ad.

2.2.2. Heurisztika

A másik módszer egy minimális útkeresési heurisztika. A következő lokális keresésen alapuló heurisztikát alkalmaztam.

Kiindulunk egy csupán egy csomagot kiszállító útvonalból, költségnek a fent definiált redukált költséget vesszük. Minden lépésben veszünk egy csomagot, és megpróbáljuk beszúrni. Külön-külön megnézzük, hogy a feladó és a vásárló beszúrása hová lenne a legoptimálisabb. Ha azzal, hogy bevettük őket csökkentettük a redukált költséget, akkor az új útvonallal megyünk tovább. Ha már nem tudunk tovább javítani, visszatérünk a kapott legjobb útvonallal és azt hasonlítjuk össze α-val.



3. További tervek

A cél egy olyan csomagszállító szolgáltatás útvonalainak optimalizálása, ahol a kiszállítást bármelyik felhasználó elvégezheti. Ha a kiszállítandó csomag az utazásuk során útba esik, profit reményében kiszállíthatják azt. A feladatban sok változó van, amit még korábban nem vettünk figyelembe. Hozzá lehet venni a kapacitás korlátokat, a csomagok kiszállításának legkorábbi és legkésőbbi idejét, és, ami a feladat érdekességét adja, hogy minden felhasználóhoz olyan csomagot társítunk, ami az útvonalához közel van.

Hivatkozások

[1] Barnhart, Cynthia, et al. "Branch-and-price: Column generation for solving huge integer programs." Operations research 46.3 (1998): 316-329.

[2] Toth, Paolo, and Daniele Vigo, eds. Vehicle routing: problems, methods, and applications. Society for Industrial and Applied Mathematics, 2014.