Diplomamunka

Robusztus és kétszintű optimalizálás

Mályusz Attila Edmund
Alkalmazott matematikus MSc

Témavezető:

Kis Tamás
docens
Operációkutatási Tanszék

Eötvös Loránd Tudományegyetem
Matematikai Intézet

2023

Tartalomjegyzék

Köszönetnyilvánítás

A diplomamunkám elkészítésében sok segítséget kaptam, amiért nagyon hálás vagyok. Mindenek előtt Kis Tamás belsős témavezetőmnek szeretnék köszönetet mondani, aki fáradhatatlanul segédkezett a szakdolgozat elkészítésében. Szép meglátásai, útmutatásai, tanácsai nélkül ez nem sikerülhetett volna.

Bevezető

Lineáris és egészértékű programozás tanulmányaim során, a megoldandó feladatok estén mindig feltételeztük, hogy az adott értékek pontosan kimért adatok. Ez viszont hibára adhat lehetőséget, hiszen az adatok legtöbbször mért adatok, melyek így pontatlanok lehetnek. Az adatokkal kapcsolatban felmerülő problémák alatt olyanra kell gondolni, hogy az adott adat eleve csak közelítve van statisztikai módszerekkel, mint például kereslet adott termékekre, vagy egyszerűen mérési hiba lép fel (fizikai méréseknél). Így természetesen merül fel, hogy változó adatok mellett a feladatra adott megoldásunk még mindig kielégítse a feltételeket és ezeken belül optimális legyen. A bizonytalanság kezelésére kétféle modellt használhatunk a sztochasztikus és a robusztus optimalizálást. A sztochasztikus megközelítés várható értékben optimalizál és nagy valószínűséggel kielégíti az előírt feltételeket. Ezzel ellentétben a robusztus optimalizálás mindig teljesíti az előírt feltételeket és ezeken belül a legjobb megoldást választja. Mivel a robusztus megoldás kielégíti az előírt feltételeket minden esetben, így olyan feltételeket is, amelyek kis valószínűséggel következnek be. Így az optimum értéke távol kerülhet a kapott adatokból számolt értéktől.

Az 1. fejezet célja a robusztus optimalizálás alapjainak bemutatása. A fejezet bevezető jellege miatt ismertetjük a bizonytalansági modellt, melyet a továbbiakban használunk. Bemutatunk a legismertebb megoldási módszerek közül hármat. Ezek közül eggyel kiemelten fogunk foglalkozni, melynek sok szempontból előnyös tulajdonságai vannak, ezen modell neve Berstimas-Sim modell. A modell nem csak lineáris optimalizálás esetén mutat jó tulajdonságokat, hanem, mint azt megmutatjuk robusztus kombinatorikus problémák esetén, és approximációs algoritmust is lehet adni ezen feladatokra. A fejezet lezárásaként egy mérést mutatunk be, melyben a robusztus optimális megoldás értéke és a kapott adatokból kapott optimális megoldás értékének a távolságát vizsgáljuk.

A 2. fejezetben a kétszintű lineáris programozási modellt mutatjuk be. Ezen modellben két játékos egymásra hatóan próbálják optimalizálni saját célfüggvényeiket. Ezen fejezet célja bemutatni a modellt és az optimális megoldás megtalálásának nehézségeit. A fejezet végén bemutatunk két módszert, amellyel képesek vagyunk megtalálni az optimális megoldást speciális kétszintű lineáris programozási feladatoknak.

A 3. fejezet a diplomamunka lezárásaként egy életbeli feladatot próbálunk

megoldani, mely a robusztus és kétszintű optimalizálás feladatok metszetében van. A feladatunk útszakaszok beárazásáról fog szólni, melyben első játékosunk célja a profit maximalizálás, míg a többi játékos célja a lehető legolcsóbban eljutni kezdőpontjuktól végpontjukig. A 2-ben bemutatottak miatt a célunk az optimális megoldás megközelítése lesz.

1. fejezet

Robusztus Optimalizálás

Ebben a fejezetben bevezetjük a robusztus optimalizálás alapjául szolgáló bizonytalanság modellt, a robosztus feladat modellezésére tett kísérleteket 1.2. A 1.3 alfejezetben kiemelten foglalkozunk az egyik ilyen modellel, melynek neve Berstiam-Sim modell. Megmutatjuk, hogy a modell optimális megoldását át tudjuk alakítani egy lineáris programozás feladattá. Mindezek után rátérünk arra, hogy milyen előnyös tulajdonságai vannak a a Berstiams-Sim modellnek egészértékű feladatoknál általánosan később kombinatorikus problémák esetén 1.4. A fejezet legvégén 1.4.2 azt mérjük ki, hogy milyen veszteséggel jár a robosztus feladatot megoldani a kapott adatokból készült feladathoz képest.

1.1. Bizonytalanság lineáris programozásban

Vizsgáljuk a következő lineáris programozási feladatot. Ahol feltesszük, hogy a bizonytalanság csak a c', b vektorokat és az A mátrixot érinti.

$$\begin{aligned} &\text{maximize } c'x \\ &\text{subject to } Ax \leq b \\ &\qquad\qquad l \leq x \leq u, \end{aligned}$$

ahol $A \in \mathbf{R}^{m\times n}$, $c', l, u \in \mathbf{R}^n$, $b \in \mathbf{R}^m$ Ekkor a következő megfigyelést tehetjük.

1. Állítás. *[3] Feltehetjük, hogy a bizonytalanság egyedül az A mátrixot érinti.*

Bizonyítás. Tegyük fel, hogy b_i bizonytalan, ekkor $a_i x - b_i z \leq 0$-et hozzádva és z-t 1-re rögzítve kiküszöböljük a bizonytalanságot a korlát jobb oldalán. Tegyük fel, hogy c-ben szerepelnek bizonytalan eggyüthatók, ekkor a $z - cx \leq 0$ korlátot felvéve a feltételek közé, és a maximalizálandó kifejezést z-re cserélve kiköszöbültük a bizonytalanságot a célfüggvényben. □

1.1.1. Mátrix bizonyítalanságának modellezése

A következő féleképpen fogjuk modellezni a bizonytalanságot a mátrixban. Vegyük az A mátrix i. sorát és legyen J_i azon együtthatók halmaza az i. sorban melyek bizonytalanok. Minden a_{ij}, $j \in J_i$ együttható egy $\bar{a}_{ij}$ valószínűségi változó felvett értéke, mely értékek a $[a_{ij} - \hat{a}_{ij}, a_{ij} + \hat{a}_{ij}]$ intervallumból vannak. $\bar{a}_{ij}$ szimmetrikus az adott intervallumra és független egymástól. Végeredményben feltettük, hogy a $\bar{a}_{ij}$ változó várható értéke megegyezik a kapott a_{ij} értékkel és az értékek $\hat{a}_{ij}$ tartományban mozognak. Fontos megjegyezni, hogy a várható értéknek mi bármit megválaszthattunk volna amire teljesülnek a feltételek, nem feltétlenül kell a kapott értéknek lennie. Ezen felül még definiáljuk a $\eta_{ij} = (\bar{a}_{ij} - a_{ij})/\hat{a}_{ij}$ random változót, mely szimmetrikus és a $[-1, 1]$ intervallumból veszi fel értékeit. A továbbiakban a kapott értékekből készített feladatot nominális feladatnak nevezzük.

1.2. Régebbi modellek

1.2.1. Soyster modell

Az első próbálkozás a robosztus optimalizálás feladat megoldására Soystertől jött [14]. Soyster egy olyan lineáris optimalizálási modellt adott, melyben a megoldásnak megengedettnek kellett lennie minden olyan adatra, amely egy konvex halmazból származik. A kapott modellel viszont az volt a probléma, hogy túlságosan messze került a robusztus optimális megoldás a nominális feladat optimális megoldástól. Soyster modellje a következő féleképpen nézett ki.

$$\begin{aligned} &\text{maximize } c'x \\ &\text{subject to } \sum_{j=1}^{n} A_j x_j \leq b \qquad\qquad \forall A_j \in K_j, j = 1, \ldots, n \\ &\qquad\qquad x \geq 0, \end{aligned}$$

ahol K_j konvex halmazok a bizonytalansági halmazok, és A_j pedig az oszlopai a mátrixnak. Soyster megmutatta, hogy a fenti feladat megegyezzik a következő lineáris programmal.

$$\begin{aligned} &\text{maximize } c'x \\ &\text{subject to } \sum_{j=1}^{n} \bar{A}_j x_j \leq b \\ &\qquad\qquad x \geq 0, \end{aligned}$$

ahol $\bar{a}_{ij} = sup_{A_j \in K_j}(A_{ij})$.
A fenti adat bizonytalansági modellben a következőképpen néz ki Soyster robusztus modellje.

$$\begin{aligned} &\text{maximize } c'x && && (1.1)\\ &\text{subject to } \sum_j a_{ij}x_j + \sum_{j\in J_i} \hat{a}_{ij}y_j \le b_i && \forall i\\ &\quad -y_j \le x_j \le y_j && \forall j\\ &\quad l \le x \le u\\ &\quad y \ge 0 \end{aligned}$$

Legyen x^* a 1.1 feladat optimális megoldása. Könnyen látható, hogy az optimális megoldásban $y_j = |x_j^*|$. Ekkor pedig

$$\sum_j a_{ij}x_j + \sum_{j\in J_i} \hat{a}_{ij}|x_j^*| \le b_i \ \forall i$$

Most pedig megmutatjuk, hogy az $\bar{a}_{ij}$ minden felvett értékére az x^* megoldásunk megengedett marad. Ehhez nézzük a következő egyenlőtlenség láncot.

$$\sum_j \bar{a}_{ij}x_j^* = \sum_j a_{ij}x_j^* + \sum_{j\in J_i} \eta_{ij}\hat{a}_{ij}x_j^* \le \sum_j a_{ij}x_j^* + \sum_{j\in J_i} \hat{a}_{ij}|x_j^*| \le b_i \ \forall i$$

Ebből láthatjuk, hogy az i. feltételnél $\sum_j \hat{a}_{ij}|x_j^*|$ a távolság b_i és $\sum_j \bar{a}_{ij}x_j^*$ között.

1.2.2. Ben-Tal, Nemirovski modell

Egymástól függetlenül Ben-Tal és Nemirovski [1], El-Ghaoui [10], is olyan modell kiépítésén dolgozott, amelyben a robusztus feladat optimális megoldásának az értéke ne legyen túl távol a nominális feladat optimum értékétől. Az ő megoldásukban ellipszioid bizonytalanságot vizsgáltak, amit visszavezettek kvadratikus optimalizálási problémára. Ezen módszer problémája az volt, hogy lassabb lett a megoldás kiszámolása a kvadratikus feladat miatt.

$$\begin{aligned} &\text{maximize } c'x\\ &\text{subject to } \sum_j a_{ij}x_j + \sum_{j\in J_i} \hat{a}_{ij}y_{ij} + \Omega_i\sqrt{\sum_{j\in J_i} \hat{a}_{ij}^2 z_{ij}^2} \le b_i && \forall i \quad (1.2)\\ &\quad -y_{ij} \le x_j - z_{ij} \le y_{ij} && \forall i, j \in J_i\\ &\quad l \le x \le u\\ &\quad y \ge 0 \end{aligned}$$

Az 1.1.1 bizonytalansági modell esetében az írok megmutatták, hogy annak a valószínűsége, hogy az i. feltétel sérül felülről becsülhető $exp(\frac{-\Omega_i^2}{2})$-vel. Továbbá minden megoldás ami kielégíti Soyster feladatát az megoldás ezen feladatra

is. Tehát a nominális feladat optimális megoldás értéke kevésbe tér el Ben-Tal, Nemirovski modell optimális megoldás értékétől, mint Soyster modelljének optimális megoldásának értékétől. A következőkben ezt a tényt, hogy a nominális feladat optimális megoldásának értéke közelebb van az egyik modellben kapott optimális értéktől, mint a másikban úgy fogjuk nevezni, hogy konzervatívabb.

1.3. Berstimas-Sim modell

A mátrix minden i. sorára bevezetünk egy Γ_i változót, mely a $[0, |J_i|]$ intervallumból vesz fel értékeket. Γ_i a feladat robusztusságát alakítja, mégpedig azt jelenti, hogy az adott sorban legfeljebb $\lfloor \Gamma_i \rfloor$ együttható változtatja meg értékét a megadott korlátokon belül és egy a_{ij} együttható $(\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{ij}$-vel. A következőkben megmutatjuk, hogy az így kapott problémát megtudjuk oldani lineáris programozással. Ezen kívül az is igazolható, hogy ha több mint $\lfloor \Gamma_i \rfloor$ együttható változik, akkor megengedett a kapott megoldásunk nagy valószínűséggel. Tekintsük a következő jelenleg még nem lineáris formalizációját a problémának [3],[2].

$$\text{maximize } c'x \tag{1.3}$$

$$\text{subject to } \sum_j a_{ij}x_j + \max_{\{S_i \cup t_i | S_i \subset J_i, |S_i| = \lfloor \Gamma_i \rfloor, t_i \in J_i \setminus S_i\}} \left(\sum_{j \in S_i} \hat{a}_{ij} y_j + (\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{it_i} y_{t_i} \right) \leq b_i \quad \forall i$$

$$-y_j \leq x_j \leq y_j \quad \forall j$$

$$l \leq x \leq u$$

$$y \geq 0$$

Ha Γ_i egész értékű, akkor az i. feltétel $\beta_i(x, \Gamma_i) = \max_{\{S_i | S_i \subset J_i, S_i = |J_i|\}} \sum_{j \in S_i} \hat{a}_{ij}|x_j|$ mennyiséggel van védve. Ha $\Gamma_i = 0$, akkor $\beta_i(x, \Gamma_i) = 0$. Ha $\Gamma_i = |J_i|$, akkor ez esetben Soyster modelljét kapjuk vissza. Azért, hogy a feladatot LP feladatként tudjuk megfogalmazni, a következő állításra van szükségünk.

2. Állítás. *[3] Legyen adva egy x vektor, ekkor a*

$$\beta_i(x, \Gamma_i) = \max_{\{S_i \cup t_i | S_i \subset J_i, |S_i| = \lfloor \Gamma_i \rfloor, t_i \in J_i \setminus S_i\}} \left(\sum_{j \in S_i} \hat{a}_{ij}|x_j| + (\Gamma_i - \lfloor \Gamma_i \rfloor)\hat{a}_{it_i}|x_{t_i}| \right)$$

kifejezés megeggyezik a következő lineáris programozási feladat optimum értékével

$$\begin{aligned} \beta_i(x,\Gamma_i) = \text{maximize } & \sum_{j\in S_i} \hat{a}_{ij}|x_j|z_{ij} \\ \textit{subject to } & \sum_{j\in J_i} z_{ij} \le \Gamma_i \\ & 0 \le z_{ij} \le 1 \qquad\qquad \forall j \in J_i \end{aligned} \tag{1.4}$$

Bizonyítás. Az LP feladat optimális megoldása a következő lesz. $\lfloor\Gamma_i\rfloor$ db 1-es értékű változó és egy változó $\Gamma_i - \lfloor\Gamma_i\rfloor$ értékű lesz. Ez pedig ekvivalens a β_i definíciójában lévő halmaz kiválasztásához. □

3. Tétel. *[3] 1.3-nek a következő lineáris programizási feladat felel meg:*

$$\begin{aligned} & \text{maximize } c'x & & (1.5)\\ & \textit{subject to } \sum_j a_{ij}x_j + z_i\Gamma_i + \sum_{j\in J_i} p_{ij} \le b_i & \forall i & \\ & z_i + p_{ij} \ge \hat{a}_{ij}y_j & \forall i,j \in J_i & \\ & -y_j \le x_j \le y_j & \forall j & \\ & l_j \le x_j \le u_j & \forall j & \\ & p_{ij} \ge 0 & \forall i,j \in J_i & \\ & y_j \ge 0 & \forall j & \\ & z_i \ge 0 & \forall i & \end{aligned}$$

Bizonyítás. Először vizsgáljuk meg az 1.4 feladat duálisát.

$$\begin{aligned} \text{minimize } & \sum_{j\in J_i} p_{ij} + \Gamma_i z_i & & \\ \text{subject to } & z_i + p_{ij} \ge \hat{a}_{ij}|x_j^*| & \forall i,j \in J_i & (1.6)\\ & p_{ij} \ge 0 & \forall j \in J_i & (1.7)\\ & z_i \le 0 & \forall i & \end{aligned}$$

Mivel a 1.4 lineáris programozás feladat $\forall\ \Gamma_i \in [0,|J_i|]$-ra megoldható és véges értékű, ezért a dualitás tétel miatt 1.7-re is ugyanez igaz, és optimum értékükben megyeznek. Így 2. állítás miatt a 1.7 értéke megegyezik $\beta_i(x,\Gamma_i)$-val. Ezek után helyettesítsük be 1.3-be 1.7-t ezzel megkapva a kívánt lineáris programozás feladatot. □

A kapott lineáris programozási feladatnak $n+k+1$ változója és $m+k+n$ feltétele van, ahol $k = \sum_i |J_i|$.

1.3.1. Feltétel megszegésének valószínűsége

Mint azt megmutattuk, ha minden sorban legfeljebb $\lfloor\Gamma_i\rfloor$ együttható változtatja meg értékét a megadott korlátokon belül és egy a_{ij} együttható ($\Gamma_i -$

$\lfloor\Gamma_i\rfloor)\hat{a}_{ij}$-vel, akkor ki tudjuk számolni az optimális értéket. A következő tétel azt mutatja meg, hogy ha esetleg több változna meg, mint az előre kiszabott korlát $\lfloor\Gamma_i\rfloor$, akkor nagy valószínűséggel megengedett marad a kapott megoldásunk.

4. Tétel. *[3] Legyen $\eta_{ij} \in [-1, 1]$ független szimmetrikus valószínűségi változók, ahol $j \in J_i$. Ekkor*

$$Pr\Big(\sum_{j\in J_i} \gamma_{ij}\eta_{ij} \geq \Gamma_i\Big) \leq B(n, \Gamma_i),$$

ahol

$$B(n,\Gamma_i) = \frac{1}{2^n}\Big\{(1-\mu)\sum_{l=\lfloor(\lfloor\nu)}^{n}\binom{n}{l}+\mu\sum_{l=\lfloor(\lfloor\nu+1)}^{n}\binom{n}{l}\Big\}= \frac{1}{2^n}\Big\{(1-\mu)\binom{n}{\lfloor\mu\rfloor}+\sum_{l=\lfloor(\lfloor\nu+1)}^{n}\binom{n}{l}\Big\},$$

ahol $n = |J_i|, \mu = \frac{\Gamma_i+n}{2}$ és $\mu = \nu - \lfloor\nu\rfloor$.

1.4. Egészértékű modell

A következőkben azt az esetet fogjuk vizsgálni, ha egészértékű változós az x vektorunk vagy csak néhány változója egész. Ekkor, mint azt alább megmutatjuk hasonló formalizációt tudunk felírni, mint 3-ban. Ezen új MIP feladat felírása előtt viszont változtatunk egy kicsit azon, hogy mely változókat érinti az adat bizonytalanság és milyen az adat bizonytalanság modellje. Tekintsük a következő vegyes egészértékű feladatot.

$$\begin{aligned} \text{minimize } & \widetilde{c}x \\ \text{subject to } & Ax \leq b \\ & l \leq x \leq u \\ & x_i \in \mathbb{Z} \qquad\qquad i = 1, \ldots, k \end{aligned}$$

5. Állítás. *Feltehetjük, hogy a bizonytalanság csak $\widetilde{c}$-t és A-t érinti.*

Bizonyítás. Tegyük fel, hogy b értéke sem pontos érték. Ekkor x_{n+1} új változó és $Ax - x_{n+1}b \leq 0$, $l \leq x \leq u$, $1 \leq x_{n+1} \leq 1$ feltételekkel, már csak $\widetilde{c}$-t és A-t érinti a bizonytalanság. □

Ezen MIP feladatnál a következő adat binzonytalansági modellet fogjuk használni.

- A mátrixra a már bevezetett 1.1.1 modellt használjuk.
- A $\widetilde{c}$ változóinkra pedig a következő teljesül. $\widetilde{c}$ minden $\widetilde{c}_j$ eleme a $[c_j, c_j + d_j]$, $d_j \geq 0$, $j \in N = \{1, 2, \ldots n\}$ intervallumból veszi fel értékeit.

A 1.3 fejezethez hasonlóan itt is bevezetjük $\Gamma_i \in [0, |J_i|]$ $i = 0, 1, \ldots, m$ változóinkat. Jelen esetben J_0 érték $\widetilde{c}$ robusztosságához tartozik és $J_0 = \{j | d_j > 0\}$. Ezek alapján a Berstimas-Sim modell szerint a következő feladatot kapjuk.

$$\text{minimize } \widetilde{c}x + \max_{\{S_0 | S_0 \subset J_0, |S_0| \leq \Gamma_0,\}} \sum_{j \in S_0} d_j |x_j| \tag{1.8}$$

$$\text{subject to } \sum_j a_{ij} x_j + \max_{\{S_i \cup t_i | S_i \subset J_i, |S_i| = \lfloor \Gamma_i \rfloor, t_i \in J_i \setminus S_i\}} \left(\sum_{j \in S_i} \hat{a}_{ij} |x_j| + (\Gamma_i - \lfloor \Gamma_i \rfloor) \hat{a}_{ij} |x_{t_i}| \right) \leq b_i \quad \forall i$$

$$l \leq x \leq u$$

$$x_i \in \mathbb{Z} \; i = 1, \ldots, k$$

Hasonlóan a 1.3 fejezetben leírtakhoz ezt a feladatot is át tudjuk alakítani ismert feladat típussá, mégpedig MIP feladattá.

6. Tétel. *[2] 1.8-nak a következő MIP feladat felel meg:*

$$\textit{minimize } \widetilde{c}x + z_0 \Gamma_0 + \sum_{j \in J_0} p_{0j} \tag{1.9}$$

$$\begin{aligned}
\textit{subject to } & \sum_j a_{ij} x_j + z_i \Gamma_i + \sum_{j \in J_i} p_{ij} \leq b_i && \forall i \\
& z_0 + p_{0j} \geq d_j y_j && \forall j \in J_0 \\
& z_i + p_{ij} \geq \hat{a}_{ij} y_j && \forall i \neq 0, j \in J_i \\
& -y_j \leq x_j \leq y_j && \forall j \\
& l_j \leq x_j \leq u_j && \forall j \\
& p_{ij} \geq 0 && \forall i, j \in J_i \\
& y_j \geq 0 && \forall j \\
& z_i \geq 0 && \forall i \\
& x_i \in \mathbb{Z} && i = 1, \ldots, k
\end{aligned}$$

Bizonyítás. Az átírást elkezdhetjük ugyanúgy, mint a 1.5. tétel bizonyításában. Először átírjuk a mátrixhoz tartozó max feltételeket felhasználva 2-t. Így már csak a célfüggvényhez tartozó max kifejezést kell átalakítani. Ehhez

pedig tekintsük a következő egyenlőségláncot.

$$\begin{aligned}\beta_0(x) &= \max\Big\{\sum_{j\in S_0} d_j|x_j|\colon\ |S_0| \le \Gamma_0, S_0 \subseteq J_0\Big\}\\ &= \max\Big\{\sum_{j\in S_0} d_j|x_j|z_{0j}\colon\ \sum_{j\in J_0} z_{0j} \le \Gamma_0, 0 \le z_{oj} \le 1, \forall j \in J_0\Big\}\\ &= \min\Big\{\sum_{j\in J_0} p_{0j} + \Gamma_0 z_0\colon\ z_0 + p_{0j} \ge d_j|x_j|, z_0 \ge 0, p_{0j} \ge 0, \forall j \in J_0\Big\}\end{aligned}$$

Mindezt behelyettesítve 1.8-be kapjuk a kívánt feladatot. □

1.4.1. Kombinatorikus problémák robusztus változata

A továbbiakban kombinatorikus problémák robusztus változatait nézzük meg. Adott $X \subseteq \{0,1\}^n$ halmaz és a következő opimalizlálsi feladat

$$\begin{aligned}&\text{minimize } \widetilde{c}x\\ &\text{subject to } x \in X\end{aligned} \tag{1.10}$$

Szeretnénk egy $x \in X$ megoldást találni, amely minimalizálja a $\widetilde{c}x$ költséget, ha legfeljebb Γ együttható változhat meg, azaz

$$\begin{aligned}Z^* = \ &\text{minimize } \widetilde{c}x + \max_{\{S|S\subseteq N, |S|\le\Gamma\}} \sum_{j\in S} d_j x_j\\ &\text{subject to } x \in X\end{aligned} \tag{1.11}$$

A továbbiakban feltesszük, hogy d_i csökkenő sorrendbe vannak rendezve, azaz $d_1 \ge d_2 \ge \cdots \ge d_n$ és legyen $d_{n+1} = 0$. Ezen feladattal olyan problémákat vettünk egy kalap alá, mint legrövidebb út, legolcsóbb feszítőfa, matroid metszet, utazó ügynök probléma. A legfontosabb tény ami megmutatható ezen feladattal kapocslatba azt a következő tétel mondja ki, ha polinomiális időben megoldható az eredeti feladatunk, akkor a robusztus változatot is polinomiális időben tudjuk megoldani.

7. Tétel. *A 1.11 feladat megoldása megkapható $n+1$ feladat megoldásából:*

$$Z^* = \min_{l=1,\ldots,n+1} G^l,$$

ahol

$$\begin{aligned}G^l = &\text{ minimize } \Gamma d_l + \min(\widetilde{c}x + \sum_{j=1}^{l}(d_j - d_l)x_j)\\ &\text{subject to } x \in X\end{aligned}$$

Bizonyítás. Vegyük észre, hogy 1.11 átírható a következő formába.

$$\begin{aligned} Z^* = \underset{x\in X}{\text{minimize}}\ & \widetilde{c}x + \max \sum_{j\in N} d_j x_j u_j \\ \text{subject to } & 0 \le u_j \le 1 && \forall j \in N \\ & \sum_{j\in N} u_j \le \Gamma \end{aligned}$$

Használjuk a duálitástételt a belső LP feladatra. Ekkor kapjuk a következőt.

$$\begin{aligned} Z^* = \underset{x\in X}{\text{minimize}}\ & \widetilde{c}x + \min(\Gamma\theta + \sum_{j\in N} y_j) \\ \text{subject to } & y_j + \theta \ge d_j x_j && \forall j \in N \\ & y_j, \theta \ge 0 \end{aligned}$$

Most már mindkét célfüggvény min-t használ, ezért összevonhatjuk a kettőt.

$$\begin{aligned} Z^* = \ \text{minimize}\ & \widetilde{c}x + \Gamma\theta + \sum_{j\in N} y_j && (1.12) \\ \text{subject to } & y_j + \theta \ge d_j x_j && \forall j \in N \\ & y_j, \theta \ge 0 \\ & x \in X \end{aligned}$$

Mivel minimum értéket keresünk és $y_j \ge 0$, ezért az optimum megoldásban tudjuk, hogy

$$y_j^* = \max(d_j x_j^* - \theta^*, 0),$$

ahol az optimális megoldást (x^*, y^*, θ^*)-val jelöljük. Ennek segítségével át tudjuk alakítani a feladatot.

$$Z^* = \min_{x\in X, \theta\ge 0} (\widetilde{c}x + \Gamma\theta + \sum_{j\in N} \max(d_j x_j^* - \theta^*, 0))$$

Most pedig felhasználjuk azt a tényt, hogy $X \subseteq \{0,1\}^n$ ugyanis ekkor igaz a következő.

$$\max(d_j x_j^* - \theta^*, 0) = \max(d_j - \theta^*, 0) x_j^* \qquad (1.13)$$

Ezek alapján a következő új formában írhatjuk fel a feladatot.

$$Z^* = \min_{x\in X, \theta\ge 0} (\widetilde{c}x + \Gamma\theta + \sum_{j\in N} \max(d_j - \theta^*, 0) x_j^*) \qquad (1.14)$$

θ optimális értéknek megkereséséhez szabdaljuk fel $\mathbb{R}_+$-t intervallumokra. Ezen intervallumokban külön-külön megkeressük θ optimális értékét. Legyenek az

intervallumok $[0, d_n], [d_n, d_{n-1}], \ldots, [d_2, d_1], [d_1, \infty)$. Ekkor a következőt figyelhetjük meg.

$$\sum_{j \in N} \max(d_j - \theta^*, 0) x_j^* = \begin{cases} \sum_{j=1}^{l-1} (d_j - \theta) x_j & \text{, ha } \theta \in [d_l, d_{l-1}],\ l = n+1, \ldots, 2 \\ 0 & \text{, ha } \theta \in [d_1, \infty) \end{cases}$$

Ezen megfigyelés segítségével felszabdalhatjuk a feladatot alfeladatokra. Legyen $Z^* = \min_{l=1,\ldots,n+1} Z^l$, ahol

$$Z^l = \min_{x \in X, \theta \in [d_l, d_{l-1}]} (\Gamma\theta + \widetilde{c}x + \sum_{j=1}^{l-1} (d_j - \theta) x_j)$$

Amikor θ-t a $[d_l, d_{l-1}]$ intervallumban keressük, akkor egy lineáris függvényen optimalizálunk. Így a szélsőérték a két végpont valamelyikében vétetik fel. Azaz

$$\begin{aligned} Z^l &= \min(\Gamma d_l + \min_{x \in X}(\widetilde{c}x + \sum_{j=1}^{l-1} (d_j - d_l) x_j), \Gamma d_{l-1} + \min_{x \in X}(\widetilde{c}x + \sum_{j=1}^{l-1} (d_j - d_{l-1}) x_j)) \\ &= \min(\Gamma d_l + \min_{x \in X}(\widetilde{c}x + \sum_{j=1}^{l} (d_j - d_l) x_j), \Gamma d_{l-1} + \min_{x \in X}(\widetilde{c}x + \sum_{j=1}^{l-1} (d_j - d_{l-1}) x_j)) \end{aligned}$$

Azaz átírhatjuk az eredeti feladatot a következőre.

$$Z^* = \min(\Gamma d_1 + \widetilde{c}x, \ldots, \Gamma d_l + \min_{x \in X}(\widetilde{c}x + \sum_{j=1}^{l} (d_j - d_l) x_j), \ldots, \min_{x \in X}(\widetilde{c}x + \sum_{j=1}^{n} d_j x_j))$$

□

8. Megjegyzés. *Fontos megemlíteni, hogy erősen kihasználtuk azt a tényt, hogy $X \subseteq \{0,1\}^n$. Általános egészértékű programozás feladat esetén nem lenne igaz az 1.13-as lépés.*

9. Megjegyzés. *Ha tekintünk egy egyszerűbbnek tűnő feladatot, ahol csak két költség függvény mellett kell optimalizálni.*

$$\begin{aligned} &\text{minimize } \max(\widetilde{c_1}x, \widetilde{c_2}x) \\ &\text{subject to } x \in X \end{aligned}$$

Akkor ez a feladat általánosan NP-nehéz, hiába csak két költségünk van. Ha az alap feladat a legrövidebb út keresés, akkor ez egy ismert NP-nehéz feladat. A továbbiakról a [11]-ban lehet olvasni.

1.4.2. Robusztus approximációs algoritmusok

Van még egy kiváló tulajdonsága a 1.10-es robusztus optimalizálási problémának, ha az eredeti feladatnak van egy α közelítő algoritmusa, akkor tudunk készíteni egy polinomiális algoritmust, amely a robusztus feladatra ad α közelítő megoldást. A következőkben feltesszük, hogy létezik egy B algoritmus, mely a választott kombinatorikus problémára ad egy x_B megoldást, melynek költségére igaz, hogy $Z^* \leq Z_B \leq \alpha Z^*$, $\alpha \geq 1$.

Algoritmus A

1. Használjuk B algoritmust, hogy α közelítő megoldást kapjunk a következő feladatokra. $l = 1, \ldots, n+1$

$$G^l = \text{ minimize } \Gamma d_l + \min(\widetilde{c}x + \sum_{j=1}^{l}(d_j - d_l)x_j) \qquad (1.15)$$

$$\text{subject to } x \in X$$

   Az α közeli megoldásokat x_B^l-vel jelöljük.

2. Legyen

$$Z_B^l = \widetilde{c}x_B^l + \max_{\{S|S\subseteq N, |S|\leq\Gamma\}} \sum_{j\in S} d_j (x_B^l)_j$$

3. Legyen $l^* = \arg\min_{l=1,\ldots,n+1} Z_B^l$, $Z_A = Z_B^{l^*}$, $x_B = x_B^{l^*}$

10. Tétel. *Algoritmus A által adott megoldásra igaz, hogy*

$$Z^* \leq Z_B \leq \alpha Z^*$$

Bizonyítás. A $Z^* \leq Z_B$ egyenlőtlenség teljesül, hiszen Z^* az optimális megoldás, Z_B pedig egy megengedett megoldás. A $Z_B \leq \alpha Z^*$ egyenlőtlenség igazolásához tekinstük a következő egyenlőtlenség láncot, ahol l a 7. tétel G^l-ek optimális értékének indexe. x_B^l pedig az α approximációs megoldása G^l-nek.

$$\begin{aligned}
Z_B &\leq Z_B^l \\
&= \widetilde{c}x_B^l + \max_{\{S|S\subseteq N, |S|\leq\Gamma\}} \sum_{j\in S} d_j (x_B^l)_j \\
&= \min_{x\in X, \theta\geq 0} (\widetilde{c}x + \Gamma\theta + \sum_{j\in N} \max(d_j - \theta^*, 0) x_j^*) \text{ Használjuk 1.14-t} \\
&\leq \widetilde{c}x_B^l + \sum_{j\in N}(d_j - d_l)(x_B^l)_j + \Gamma d_l \\
&\leq \alpha(G^l - \Gamma d_l) + \Gamma d_l \text{ Használjuk 1.15-t} \\
&\leq \alpha G^l \text{ Mivel } \alpha \geq 1 \\
&= \alpha Z^*
\end{aligned}$$

□

Mérés

A következő kérdés merülhet fel robosztus optimalizálás robusztusságával kapcsolatban. Mennyire változik a célfüggvény értéke Γ_0-t változtatva? A méréshez választottunk egy célfeladatot, mely a legrövidebb út probléma lett a [2] cikkből. Adott egy $D = (V \cup \{s, t\}, A)$ irányított gráf, c_{ij} $(i, j) \in A$ nem negatív költségekkel az éleken. A cél a legrövidebb út megkeresése s-ből t-be. A gráf és az élköltségek generálásához a következő modellt használtuk. Vegyük a 2 dimenziós síkot, legyen az s a $(0, 0)$ pont, t pedig az $(1, 1)$ pont. A gráf többi csúcsának megfelelő pontot generáljuk a $[0, 1] \times [0, 1]$ téglalapban a következőképpen: Mindkettő koordinátát egyenletes eloszlású valószínűségi változó értékéből kapjuk, mely a $[0, 1]$ halmazból veheti fel. Az élköltségek generálásához a beágyazott pontok euklideszi távolságát használjuk fel. Legyen c_{ij} az euklideszi távolság i és j által meghatározott pontok között, ha fut él köztük. Legyen $d_{ij} = \gamma c_{ij}$, ahol γ egyenletes eloszlású a $[0, 8]$ intervallumon.
A mérésünkben $\Gamma = 0, 3, 6, 10$ robusztus védelmek mellett mérjük meg a robusztus legrövidebb utat. Azt is kimérjük, hogy ha minden élen q valószínűséggel $c_{ij} + d_{ij}$-re változtatjuk a költséget, akkor az így kapott gráfban mennyi a legrövidebb út. A mérés célja, hogy pontos adatot kapjunk arról, hogy mennyivel kerül többe az, hogy robusztus védelmet biztosítsunk.
Az Erdős-Rényi-féle véletlen gráf modellt használtuk, azaz minden élet egy adott p valószínűséggel veszünk be a gráfba. Az s-ből t-be vezető élet mindig töröljük a gráfból. Közösen minden Γ-hoz generálunk egy gráfot. Mindegyik Γ értékre kiszámoljuk, hogy milyen hosszú a legrövidebb robusztus út. Ezután minden élen végig menve q valószínűséggel eltoljuk a költséget és kiszámoljuk az adott gráfban a legrövidebb út hosszát, ezt a mérést 100-szor számoljuk ki. Összesen 100 gráfra nézzük meg ezeket az értékeket. A külön-külön kapott költségeket összeadjuk és kiátlagoljuk. A kapott méréseket a következő táblázat foglalja össze.

	$\|V\| = 25$	$\|V\| = 50$	$\|V\| = 100$	$\|V\| = 200$
$\Gamma = 0, 3, 6, 10$	1.66227	1.6659	1.46272	1.42595
	6.78421	5.89619	3.9693	2.86172
	7.30533	6.32539	4.22916	3.01015
	7.32129	6.33437	4.23305	3.01193
$q = 0.8$	5.78746	6.33437	2.51864	1.82651
$q = 0.3$	2.53146	2.04835	1.56672	1.44946

1.1. táblázat. Minden Γ sorában lévő cella négy értéket tartalmaz a következő sorrendben föntről-lefele $\Gamma = 0, 3, 6, 10$. $p = 0.1$ minden mérésben.

A kapott méréseink alapján a robusztus legrövidebb út hosszúsága nagyon

közel esik $\Gamma = 3, 6, 10$ esetén, és $\Gamma = 6, 10$ legfeljebb 0.02-el térnek el. Így, ha már 6 él változására fel vagyunk készülve, akkor semmi költségből a 10 él változásra is felkészülhetünk. A random élköltség eltolásokkal kapott gráfban a hosszúságok $\Gamma = 0$ és $\Gamma = 3$ költségei közé esik.
Ha valaki szeretné futtatni a programot, az letöltheti innen `https://github.com/atimaly/Robust_Shortest_Path`.

2. fejezet

Kétszintű Optimalizálás

Legelőször Stackelberg 1934-es cikkében jelent meg a kétszintű optimalizálás feladatának fogalma. Később Bracken és McGill [5] 1972-es cikkükben bevezették a feladatot formálisan is. Ezen két cikk óta folyamatos fejlődésnek indult az optimalizálásnak ezen része. A kétszintű optimalizálási feladatnak a hátterében egy játék van, amelyben két játékos dönt a hozzájuk tartozó változók értékéről. Az első játékos az úgynevezett vezető minimalizálja kívánja saját célfüggvényét azzal a feltétellel, hogy a második játékos, akit követőnek nevezünk, az első játékos válaszának függvényében fog minimalizálni. Tehát a vezető válasza befolyásolja a megengedett halmazt és a követő objektív függvényét, akinek viszont reakciója befolyásolja a vezető nyereményét (és a vezető kezdeti kiválasztásának megvalósíthatóságát). Egyik játékos sem dominálhatja a másikat teljesen. A kétszintű optimalizálási feladat a vezető célfüggvényének az optimalizálása, amelyet a követő problémájának megoldási halmazának gráfja segítségével fogalmazunk meg. A kétszintű optimalizálásról megmutatható, hogy NP-nehéz feladat [7]. Annak ellenőrzés is NP-nehéz, hogy egy megengedett megoldás lokális optimuma a feladatnak [7]. Ahogy azt később megmutatjuk a kétszintes optimalizálási feladatok nem konvex optimalizálási feladatok. Ezen tulajdonságok miatt a feladat optimumának kiszámítása egy kihívást jelentő feladat. Az alapvető ötlet egyszerűbb feladatok esetén a komplementaritási feltételek használata a követő feladatában, hiszen így el tudjuk érni, hogy optimum választ kapjunk a követő részéről. A fejezet célja, hogy az alapvető definíciókat és nehézségeket bemutassa a kétszintű optimalizálás feladatával kapcsolatban.

2.1. Kétszintes optimalizálási feladat definíciója

A kétszintű optimalizálási feladat olyan optimalizálási problémák, amelyeknél az optimális megoldási halmaz (részben) egy második paraméteres feladat megoldási halmazától függ. A második probléma legyen a következő

$$\min_y \{f(x,y) : g(x,y) \leq 0, y \in T\}, \tag{2.1}$$

ahol $f : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}$, $g : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}^p$ függvények és $T \subset \mathbf{R}^m$ egy zárt halmaz.
Legyen $Y : \mathbf{R}^n \rightrightarrows \mathbf{R}^m$ a feltételt teljesitő halmaz, ahol $\rightrightarrows$-en pontról halmazra képző függvényt értünk azaz:

$$Y(x) = \{y : g(x, y) \leq 0\} \tag{2.2}$$

Legyen

$$\varphi(x) = \min_y \{f(x, y) : g(x, y) \leq 0, y \in T\} \tag{2.3}$$

az optimális érték függvénye. Legyen $\Psi(x) : \mathbf{R}^n \rightrightarrows \mathbf{R}^m$ a megoldás halmaza 2.1-nek, azaz adott $x \in \mathbf{R}^n$-re

$$\Psi(x) = \{y \in Y(x) \cap T : f(x, y) \leq \varphi(x)\} \tag{2.4}$$

$$\mathbf{gph}\Psi = \{(x, y) \in \mathbf{R}^n \times \mathbf{R}^m : y \in \Psi(x)\} \tag{2.5}$$

az epigráfja Ψ-nek. Ekkor a kétszintes optimalizálási probléma a következő:

$$\min_x \{F(x, y) : G(x) \leq 0, (x, y) \in \mathbf{gph}\Psi, x \in X\}, \tag{2.6}$$

ahol $F : \mathbf{R}^n \times \mathbf{R}^m \to \mathbf{R}$, $G : \mathbf{R}^n \to \mathbf{R}^q$ és $X \subset \mathbf{R}^n$ egy zárt halmaz.

11. Megjegyzés. *Ahogy az a fejezet bevezetőjében meg volt említve a 2.6 feladat úgy értelmezhető, mint egy két emberes játék, ahol az első játékos (vezető) választ egy x-et és ezt a válszatást kommunikálja a második játékosnak (követő), aki ezek után x-et figyelembe véve választ magának egy optimális y-t, amit vissza ad az első játékosnak. Az első játékos pedig x és y segítségével kiszámolja a saját veszteségét. Az első játékos célja a saját veszteségének minimalizálása. 2.6-t felső problémának nevezzük. 2.1-t pedig alsó problémának.*

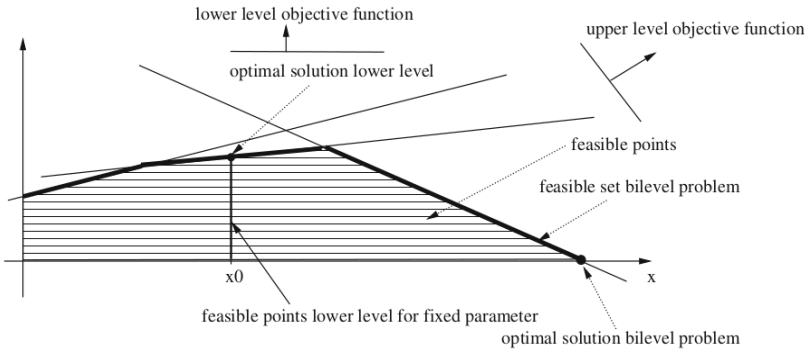
2.2. Nehézségek a kétszíntű optimalizálással kapcsolatban

A 2.1 alfejezetben lévő definíció sok gondot hoz magával, melyek ártalmatlannak tűnő optimalizálási feladatokat is átalakíthat kemény feladatokká. A következőkben ezen problémákból mutatunk egy válogatást. Vegyük a következő esetet, ahol $n = 1$, $m = 1$. Azaz az alsó és felső problémának egy-egy változója van. A követő feladata legyen a következő.

$$G(x) \equiv 0 \ , \ \{(x, y) : g(x, y) \leq 0\}$$
$$f(x, y) = -y$$

Ha az $x = x_0$ esetet nézzük, akkor a lehetséges megoldásai az alsó problémának az 2.1 ábrán az x_0 fölötti szakasz mutatja. Így az optimális megoldás pontosan

a vastagított vonal lesz. Más szóval **gph**Ψ a vastagított vonal. Ahogy az ábrán látszik a kétszintes optimalizálási probléma egy nem konvex feladat lesz.



2.1. ábra. [7] A feladat általánosan nem konvex optimalizálás.

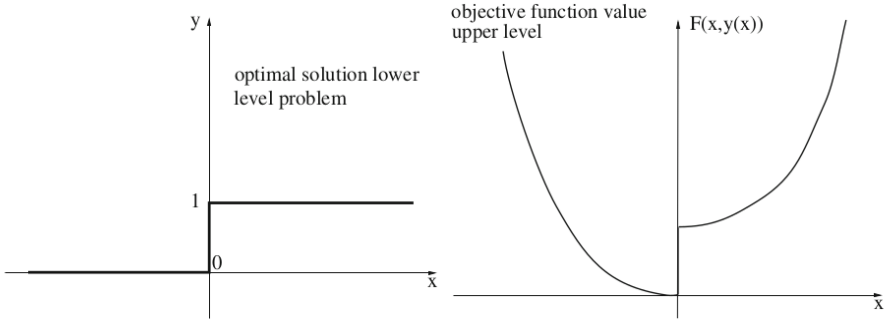
A második problémát a kövekező példánk fogja mutatni. Vegyük az alábbi optimalizálási feladatot ad.

$$\min_x \{x^2 + y : y \in \Psi(x)\}, \tag{2.7}$$

ahol

$$\Psi(x) = \arg\min_y \{-xy : 0 \leq x \leq 1\} \tag{2.8}$$

A Ψ gráfja a jobb oldalon látható a 2.2 ábrán. Az $x \to F(x, \Psi(x))$ felső probléma pedig a jobb oldalon látható. Az gráf nem egy függvényt ad ugyanis az érték függ az $y \in \Psi(x)$ értéktől $x = 0$-ban. Ha $y = 0$-t mond a követő, akkor optimista kétszintű optimalizálási problémáról beszélünk, ugyanis x^2+y értéke minimális lesz 0 értékkel. Ha $y = 1$-et mond, akkor pesszimista kétszintű optimalizálási problémáról, hiszen ekkor $x^2+y = 1$ lesz igaz. Ha pedig $y \neq 0, 1$, akkor nincs minimuma a felső problémának csak infínuma.



2.2. ábra. [7] A követő optimális válaszai között is van különbség.

2.3. Optimista és pesszimista kétszintű optimalizálás

Az előző példa rávilágított arra a tényre, hogy a követőnek választása az optimális megoldásai között nagy hatással van a vezető célfüggvény értékére. Így ezen tényre fogunk bevezetni új fogalmakat. Ha a vezető feltételezheti, hogy a követő együttműködő. Azaz a $\Psi(x)$ halmazból azt fogja választani amelyik a legjobb a vezető célfüggvényére. Ekkor a következő függvényt kell minimalizálni

$$\varphi_0(x) = \min\{F(x,y) : y \in \Psi(x)\} \tag{2.9}$$

a $\{x : G(x) \leq 0, x \in X\}$ halmazon. Ez az optimista kétszintű optimalizálás. Abban az eseteben, ha a vezető nem számíthat kooperációra. Akkor a következő függvényt kapjuk.

$$\varphi_p(x) = \max\{F(x,y) : y \in \Psi(x)\} \tag{2.10}$$

a $\{x : G(x) \leq 0, x \in X\}$ halmazon. Ez a pesszimista kétszintű optimalizálás. Ezen fogalmak még fontos szerepet fognak játszani a 3 fejezetben.

2.4. Lináris kétszintes optimalizálás

Most fókuszáljunk a kétszintes optimalizálási problémák egy részhalmazára. Vizsgáljunk olyan problémákat, amikor csak lineáris feltételeink vannak. A lineáris kétszintes optimalizálásra már láthattunk példát 2.2-ben. Vizsgáljuk meg a következő definícióját a lineáris kétszintes optimalizálási feladatoknak. Esetleg itt is felmerülnek-e hasonló problémák, mint az általános definícióban?

12. Definíció. *A következő feladatot nevezzük lináris kétszintes optimalizálás problámának.*

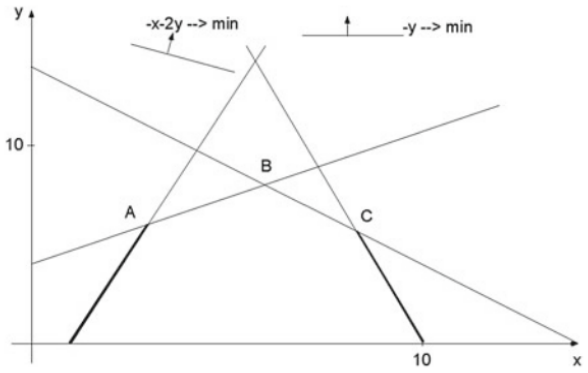
$$\min_{x,y}\{a^T x + b^T y : Ax + By \leq c, (x,y) \in \boldsymbol{gph}\Psi\}, \tag{2.11}$$

ahol

$$\Psi(x) = \arg\min_y\{d^T y : Cy \leq x\} \tag{2.12}$$

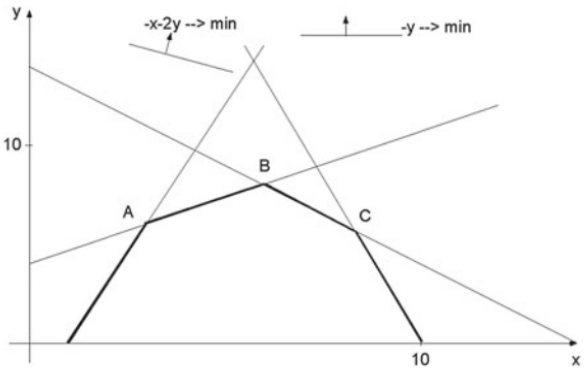
Az első probléma ami felvetődhet ezzel az általános probléma megfogalmazással, hogy az $Ax + By \leq c$ feltétel teljesülését csak y megválasztása után lehet leellenőrizni. Ezeket összefüggő feltételeknek hívjuk. Ezen tulajdonságból következő nehézségekre az alábbi példa [12] világít rá. Legyen az optimalizálási feladatunk a következő.

$$\begin{aligned}
&\text{minimize} - x - 2y \\
&\text{subject to } 2x - 3y \geq -12 \\
&\qquad\qquad x + y \geq 14 \\
&\qquad\qquad y \in \arg\min_y\{-y : -3x + y \leq -3, 3x + y \leq 30\}
\end{aligned}$$



2.3. ábra. [7] Az optimális megoldás függ a feltételek helyétől.

A 2.3 ábrán $\mathbf{gph}\Psi(x)$ halmazunk vastagított vonallal van jelölve. Ebben az esetben az optimális megoldása a feladatnak a C pontban van. Az A és C pontok közötti szakaszok nincsenek megvastagítva, mert ha x-et innen választjuk, akkor az y olyan helyen venné fel az optimumot ami megsértené a felső feltételt. Vizsgáljuk meg mi történik, ha a két feltételt a felső problémából levisszük az alsóba. Ekkor a feladat optimuma most már a B pont lesz.



2.4. ábra. [7] Ha levisszük a feltételt.

Így a következőkben elkerüljük azt, hogy a felső feladat feltételei függenek az alsó változóitól. Így csak is a következő lineáris kétszintes programozási feladatokat fogjuk vizsgálni

$$\min_{x,y}\{a^T x + b^T y : Ax \leq c, (x,y) \in \mathbf{gph}\Psi\}, \tag{2.13}$$

ahol

$$\Psi(x) = \arg\min_{y}\{d^T y : Cy \leq x\}. \tag{2.14}$$

Ekkor megmutatható [13], hogy a $\Psi()$ leképzés gráfja előáll az $\{(x,y)^T : Cy \leq x\}$ halmaz oldalainak az összefüggő uniójaként.
Itt egy M halmaz összefüggő ha, nincsen benne két diszjunkt nyílt halmaz uniójában. Ebből kapjuk a következő tételt

13. Tétel. *2.13 feladat lineáris optimalizálás feladat és az optimum megtalálható*

$$\{(x,y)^T : Cy \leq x, Ax \leq c\} \tag{2.15}$$

valamelyik csúcsán.

Ezen tétel bizonyítása megtalálható [4]-ban és sok algoritmus alapját ez a tétel képzi.
Harmadik problémánk a feltételek hozzáadásáról szól, mégpedig egy globális optimális megoldás elvesztheti optimalitását, ha hozzá adunk feltételt a feladathoz, amely nem aktív az optimum értéknél. Ez egy fontos észrevétel ugyanis folytonos optimalizálásnál nem így van. A példa [8]-ből van.
Legyen a következő feladatunk

$$(x-1)^2 + (y-1)^2 \to \min_{x,y},$$

ahol y a következő feladat optimuma

$$0.5y^2 + 500y - 50xy \to \min_{y} \tag{2.16}$$

Mivel az alsó probléma egy konvex optimalizálási probléma az egész téren, így kicserélhetjük az optimalitás feltételekre. Így a feladat a következő lesz

$$\min_{x,y}\{(x-1)^2 + (y-1)^2 : y - 50x + 500 = 0\} \tag{2.17}$$

Ennek a feladatnak az egyetlen optimális megoldása a $(x^*, y^*) = (\frac{50102}{5002}, \frac{4100}{5002})$ pontban van. Az optimális érték pedig $z^* = 81,33$.
Adjuk most a feladathoz az $y \geq 0$ feltételt. Így a következőképpen néz ki az új feladatunk.

$$(x-1)^2 + (y-1)^2 \to \min_{x,y},$$

ahol y a következő feladat optimuma

$$y \in \arg\min_{y}\{0.5y^2 + 500y - 50xy : y \geq 0\}$$

Ekkor az optimális megoldás $(\bar{x}, \bar{y}) = (1,0)$. Ez az érték viszont nem optimuma 2.16-nak. Ez mutatja, hogy bár (x^*, y^*) lokálisan optimum, de nem globális optimum.
Mint azt várhatjuk a nem konvexitásból, a lineáris kétszintes optimalizálási problémák is általánosan NP-nehezek. Ezt a következő tétel mutatja meg.

14. Tétel. *([9]) $\forall\ \epsilon > 1$-ra NP-nehéz megtalálni egy olyan megoldását a (2.11)lineáris kétszintes optimalizálás feladatnak, amely legfeljebb ϵ szorosa az optimális megoldásnak.*

2.4.1. Komplementaritási feltételek

Az előzőekben felsoroltunk nehézségéket, amely akadályozzák a kétszintű optimalizálási feladatok megoldását. Megmutattuk, hogy NP-nehéz megtalálni a megoldást és a feladat általában nem konvex. Viszont van egy részhalmaza a kétszintű optimalizálási feladatoknak, amelyeket egzaktan megtudunk oldani. Ez pedig a következő feladat.

$$\max_{x,y}\{a^T x : Ax \leq c, (x,y) \in \mathbf{gph}\Psi^1\},$$

ahol

$$\Psi^1(x) = \arg\min_y\{x^T y : By \leq d, y \geq 0\}$$

Oldjuk meg a feladatot úgy, hogy a követő pesszimista módon fog választani a vezető számára. Ahhoz, hogy a feladatot megoldhassuk használjuk fel a komplementaritási feltételeket a követő feladatában. Ezek segítségével már át tudjuk írni a feladatot a következő alakba.

$$\begin{aligned} &\text{minimize } a^T x \\ &\text{subject to } Ax \leq c \\ &\qquad\qquad 0 \leq z_i \perp -b_i y + d \geq 0 \\ &\qquad\qquad 0 \leq y_i \perp x_i + zB_i \geq 0, \end{aligned}$$

ahol a $0 \leq P \perp Q \geq 0$ kifejezés azt jelenti, hogy a $0 \leq P$, $0 \leq Q$ feltételek teljesülnek és $P = 0$, $Q = 0$ közül legalább az egyik teljesül. Így megtudjuk oldani ezt a speciális feladatot egy egészértékű programozási feladatként. Ez a tény kulcsfontosságú lesz a 3. fejezetben.

2.5. Globális algoritmus

A következőkben ismertetünk egy algoritmust, mely bár bizonyíthatóan megtalálja a megoldást, de ezt nem feltétlenül teszi polinomiális időben. Vegyük a következő két lineáris kétszintes optimalizálás problémát

$$\min_{x,y}\{a^T x + b^T y : Ax \leq c, (x,y) \in \mathbf{gph}\Psi^1\}, \tag{2.18}$$

ahol

$$\Psi^1(x) = \arg\min_y\{x^T y : By \leq d\} \tag{2.19}$$

és

$$\min_{x,y}\{a^T x + b^T y : Ax \leq c, By \leq d, x^T y \leq \varphi^1(x)\}, \tag{2.20}$$

ahol

$$\varphi^1(x) = \min_y\{x^T y : By \leq d\}. \tag{2.21}$$

A két probléma ekvivalens és a következő algoritmust lehet adni a megoldás megtalálására.

1. Válasszunk egy x^0-at mely kielégíti a $Ax^0 \leq c$ feltételt és válasszunk egy $y^0 \in \Psi(x^0)$-t. Állítsuk $k = 1$-re és $\mathbf{Y} = \{y^0\}$.

2. Oldjuk meg a következő problémát

$$\min_{x,y}\{a^T x + b^T y : Ax \leq c, By \leq d, x^T y \leq \bar{y}^T x, \forall \bar{y} \in \mathbf{Y}\} \tag{2.22}$$

   legyen a globál optimum megoldás (x^k, y^k).

3. Ha $y^k \in \Psi(x^k)$, akkor megállunk és (x^k, y^k) globál optimális megoldás. Ellentétes esetben legyen

$$\mathbf{Y} = \mathbf{Y} \cup \{\bar{y}^k\} \tag{2.23}$$

   majd menjünk a 2. lépésre.

Az tudjuk, hogy az algoritmus véges sok lépés után leáll, hiszen a $By \leq d$ poliédernek csak véges sok csúcsa lehet.

15. Tétel. *([6]) Legyen $\{(x, y) : Ax \leq c, By \leq d\}$ korlátos. Ekkor a fenti algoritmus megtalálja a globális optimumát a lineáris kétszintes feladatnak.*

3. fejezet

A kétszintű robusztus tarifa optimalizálási probléma

A lezárásaként vegyünk egy feladatot, mely a robusztus és kétszintű optimalizálás feladatok mindkettőjét egyesíti magában. A célunk az lesz, hogy kidolgozunk egy algoritmust, mely a gyakorlatban képes megtalálni az optimális megoldást vagy legalább megközelíteni azt.
A feladatunk a következő. Adott egy $G = (V, E)$ gráf, egy szolgáltató, és S utas, akik s_j-ből t_j-be akarnak eljutni, $j = 1, \ldots, S$. A szolgáltatónak $p : E \to \mathbb{R}$ élköltsége merül fel. Minden utasnak van egy u_j költség vektora az éleken, ami előre nem ismert, de azt tudjuk, hogy $u \in U$, ahol U egy előre ismert poliéder.

A szolgáltató minden élhez egy $q(e)$ árat akar rendelni, melyet egy adott utasnak ki kell fizetnie a $q(e)$ árat. A q vektornak egy Q poliéderből kell kikerülni. A szolgáltató célja a hasznának a maximalizálása, tudva azt, hogy minden utas olyan útvonalat választ, ami a költségét minimalizálja. Tehát a szolgáltató optimalizálási problémája

$$\max_{q \in Q} z \tag{3.1}$$

$$\min_{u \in U} \left\{ \sum_{j=1}^{S} (q - p) x_j \mid x \in \Omega(q, u) \right\} \geq z, \tag{3.2}$$

ahol $x = (x_1, \ldots, x_S)$, és $\Omega(q, u)$ a következő feladat optimális megldásainak a halmaza:

$$\begin{aligned} \text{minimize } & \sum_{j=1}^{S} \sum_{e \in E} (q_e + u_{j,e}) x_j(e) \\ \text{subject to } & x_j(\rho(t_j)) \geq 1, \quad \forall j \\ & x_j(\delta(s_j)) \leq 1, \quad \forall j \\ & x_j(\delta(v_j)) \leq x_j(\rho(v_j)), \quad \forall j, v \notin \{s_j, t_j\} \\ & x_j(e) \in \{0,1\}, \quad \forall j, e \end{aligned} \tag{3.3}$$

Itt $x_j(\delta(v)) = \sum_{e \in \delta(v)} x_j(e)$, and $x_j(\rho(v)) = \sum_{e \in \rho(v)} x_j(e)$, ahol $\delta(v)$ és $\rho(v)$ a v csúcs kimenő és bemenő éleinek halmaza. A következőkben $x_j(e)$-re csak a $0 \leq x_j(e) \leq 1$ megkötést tesszük az $x_j(e) \in \{0,1\}$ helyett.

Speciális esetei a feladatnak, amikor létezik egzakt megoldás

Ha a $\forall\ j = 1, \ldots, S$-re s_j-ből t_j-be egy út van, akkor a megoldás triviális, ugyanis a vezetőnek a $\sum_{e \in \exists j, e \in s_j \to t_j \text{ úton}} q(e) |\{j : e \in s_j \to t_j \text{ úton}\}|$ kifejezést kell maximalizálnia. Ez a feladat pedig egy lineáris programozás feladat.
Hasonlóan egzakt megoldást lehet adni, ha $\forall\ j = 1, \ldots, S$-re s_j-ből t_j-be d_j db diszjunkt út van és a d_j út nem metszi semelyik $i \neq j$-hez tartozó d_i utat. Ekkor ugyanis minden d_j útra külön megnézhetjük, hogy lehet-e optimális útja a követőnek, és ha igen, akkor mennyibe kerül a követőnek. Ahhoz, hogy megnézzük, hogy egy adott út lehet-e optimális út, és mekkora költséggel, ahhoz a komplementaritási feltételeket használjuk. Így végeredményben egészértékű programozási feladatokat oldunk meg.

Az algoritmus

Algoritmusunk alapötlete az lesz, hogy az U poliéderből egyesével kiválasztunk pontokat, melyek jól reprezentálják a poliédert. A vezető pedig ezen kiválasztott u értékekre válaszol optimálisan. Legyen $U^* = \{u^k\} \subset U$ olyan hasznosságok diszkrét halmaza, amelyek a lehető legrosszabbak a vezetőnek. Minden egyes lépésben megoldunk egy relaxált problémát, ahol megkeresünk egy optimális q-t a diszkrét U^*-ot használva. Ezt a q értéket megperturbáljuk. Ezek után ezzel a q értékkel keresünk egy új megoldást a követőknek x, u. Ha a megoldás érték túl kicsi változáson ment át, akkor megpróbálunk a hasznosság térben elmozdulni. Ellenkező esetben hozzáadjuk U^*-hoz.

1. Legyen $U^* = \emptyset$ és $F = \infty$ (az optimuma a vezető problémájának). Válasszunk heurisztikusan egy q értéket. Legyen az a q, amely maximalizálja a $\sum q_i\ q \in Q$ függvényt. Menjünk a 3. lépésre

2. Oldjuk meg a relaxált problémát 3 a jelenlegi U^*-al. Ebből kapunk q tarifát és F cél fügvény értéket.

3. A kapott q értéket perturbáljuk δ-val. Így megkapjuk a q^δ-t. Ezek után oldjuk meg a 3 feladatot q^δ-val, hogy megkapjunk egy u' értéket és F' cél függvény értéket.

4. Ha F és F' megfelelően közel van, akkor megtaláltunk egy jó q tarifát és visszaadjuk q^δ-t.

5. Ellenkező esetben találtunk egy u' vezetőnek kedvezőtlen hasznosságot. Ekkor két lehetőséget különböztetünk meg.

   (a) Ha u' jelentősen különbözik minden U^*-ban lévő hasznosságtól, akkor u'-t hozzáadjuk U^*-hoz.

   (b) Ha létezik u'-höz $u_0 \in U^*$, amelyre minden koordinátában legfeljebb $|E| * \delta$ mértékben különbözik, akkor keresünk egy lényegesen eltérő hasznosságot például a 3-ban leírtak segítségével.

6. Ha iterációs limitet elértük, akkor megállunk és visszaadjuk a legjobb megoldást.

Rögzített tarifához globálisan robusztus útvonal választás

Rögzítsünk egy $q \in Q$ tarifát. Találnunk kell x és u vektorokat úgy, hogy azok a legkedvezőtlenebbek megoldást adják a szolgáltató számára, de optimális az utasoknak, más szóval a 2.3 alfejezet szerint pesszimista optimális megoldást keresünk. Ebből a következő egészértékű programozás feladatot kapjuk.

$$\text{minimize } \sum_{j=1}^{S}\sum_{e\in E}(q_e - p_e)x_j(e)$$

subject to $u \in U$,

$$0 \le \alpha_j^+ \perp x_j(\rho(t_j)) - 1 \ge 0, \qquad \forall j$$

$$0 \le \alpha_j^- \perp -x_j(\delta(s_j)) + 1 \ge 0, \qquad \forall j$$

$$0 \le \beta_{j,v} \perp x_j(\rho(v_j)) - x_j(\delta(v_j)) \ge 0, \qquad \forall j, v \notin \{s_j, t_j\}$$

$$0 \le x_j(e) \perp \quad q_e + u_{j,e} - \alpha^+_{j,e\in\rho(t_j)} + \alpha^-_{j,e\in\delta(s_j)} - \sum_{e=\rho(v),v\notin\{s_j,t_j\}} \beta_{j,v} + \sum_{e=\delta(v),v\notin\{s_j,t_j\}} \beta_{j,v} \ge 0 \quad \forall j, e$$

A felírásban az $u \in U$ sor után következők a legrövidebb út 3.3 feladatra írják fel az optimalitási feltételeket. A $0 \le P \perp Q \ge 0$ jel pedig azt jelenti, hogy $0 \le P$, $0 \le Q$ feltételek teljesülnek és $P = 0$, $Q = 0$ közül legalább az egyik teljesül.

Egy relaxált feladat véges, diszkrét költség halmazzal

Adott az u^k költségvektorok egy véges halmaza, ami mellett a szolgáltató problémája keresünk egy q árazást, feltételezve, hogy az utasok a legkedvezőbb válaszaikat adják minden egyes u^k költségvektor esetében. A változók a q árvektor, és egy $x^k = (x_1^k, \ldots, x_S^k)$ útvektor minden u^k költségvektorhoz.

maximize Z

subject to

$$q \in Q$$

$$Z \leq (\sum_{j=1}^{S} \alpha^{+,k}_{j,e\in\rho(t_j)} - \alpha^{-,k}_{j,e\in\delta(s_j)} + \sum_{e\in E}(-u^k_{j,e} - p(e))x^k_j(e)) \quad \forall k$$

$$0 \leq \alpha^{+,k}_j \perp x^k_j(\rho(t_j)) - 1 \geq 0, \quad \forall j,k$$

$$0 \leq \alpha^{-,k}_j \perp -x^k_j(\delta(s_j)) + 1 \geq 0, \quad \forall j,k$$

$$0 \leq \beta_{k,j,v} \perp x^k_j(\rho(v_j)) - x^k_j(\delta(v_j)) \geq 0, \quad \forall j,k,v \notin \{s_j,t_j\}$$

$$0 \leq x^k_j(e) \perp q_e + u^k_{j,e} - \alpha^{+,k}_{j,e\in\rho(t_j)} + \alpha^{-,k}_{j,e\in\delta(s_j)} - \sum_{e=\rho(v),v\notin\{s_j,t_j\}} \beta_{k,j,v} +$$

$$\sum_{e=\delta(v),v\notin\{s_j,t_j\}} \beta_{k,j,v} \geq 0 \quad \forall k,j,e$$

Legnagyobb elmozdulás megkeresése a hasznossági térben

Most pedig az algoritmus legfontosabb része következik. A legfontosabb része az algoritmusnak az, hogy milyen értékek vannak a U^* halmazunkban, ugyanis ez határozza meg a vezető válaszát. Így igazán kulcsfontosságú, hogy jó értékeket találjunk amiket az U^*-ba belerakjunk. Adott q tarifa, u' hasznosság, $u_0 \in U^*$, a követőnek x útja és a vezetőnek egy maximum profitja F_{max}. ν irány ami például lehet az $(u' - u_0)$ érték. Itt megemlítjük, hogy az algoritmus tesztelése során az bizonyult a legjobbnak, ha ν-nek egy random egységhosszú vektort veszünk.

Cél: δ megtalálása.

$$
\begin{aligned}
&\text{maximize } \delta \\
&\text{subject to} \\
&q \in Q, \\
&(u_0 + \nu\delta) \in U \\
&\sum_{j=1}^{S} \sum_{e \in E} (q_e - p_e) x_j(e) \leq F_{\max} \\
&0 = \alpha_j^+, \qquad \forall j\colon\ x_j(\rho(t_j)) - 1 > 0 \\
&0 = \alpha_j^-, \qquad \forall j\colon\ -x_j(\delta(s_j)) + 1 > 0 \\
&0 = \beta_{j,v}, \qquad \forall j, v \notin \{s_j, t_j\}\colon\ x_j(\rho(v_j)) - x_j(\delta(v_j)) > 0 \\
&q_e + u_{j,e} - \alpha^+_{j,e\in\rho(t_j)} + \alpha^-_{j,e\in\delta(s_j)} \\
&\quad - \sum_{e=\rho(v), v\notin\{s_j,t_j\}} \beta_{j,v} + \sum_{e=\delta(v), v\notin\{s_j,t_j\}} \beta_{j,v} \geq 0 \quad \forall j, e\colon\ x(e) = 0 \\
&\alpha_j^+, \alpha_j^-, \beta_{j,v} \geq 0, \qquad \forall j, v
\end{aligned}
$$

A feladat felírásban a $0 = \alpha_j^+, \alpha_j^-, \beta_{j,v}$ sorokkal a komplementaritási feltételeket fejezzük ki. Az $F_{\max}$-el felül becslés azért van felírva, hogy ne változzon az optimális megoldás, ha változtatjuk u-t.

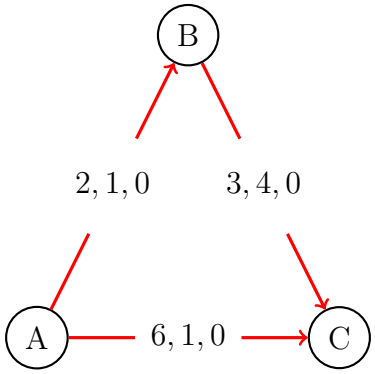
Perturbáció

A következő perturbációt végezzük a kapott q megoldáson, hogy a követő megoldása erre a q-ra egyedi legyen. A perturbáció célja, hogy az adott q-t úgy változtassuk, hogy a kapott x és u értékek az optimális válaszok maradjanak q-ra, de q ne a poliéder csúcsán legyen.
Legyen π egy permutációja $(1, \ldots, |E|)$-nek, hogy teljesüljön $q_{\pi(t)} - p_{\pi(t)} \leq q_{\pi(t+1)} - p_{\pi(t+1)}\ \forall t = 1, \ldots, |E| - 1$. Legyen T^+ a legkisebb t index, amelyre $q_{\pi(t)} - p_{\pi(t)} > 0$. Legyen q^δ a perturbált vektor a következő módon definiálva.

$$q^\delta_{\pi(t)} = (q_{\pi(t)} - (t - T^+)\delta) \tag{3.4}$$

Példa az algoritmus futására

Egy utasunk lesz aki A-ból B-be akar utazni. Vegyük a következő gráfot.



Az élekre írt számok a q, u és végül p változókhoz tartoznak. Mégpedig q és u esetén a max értékeket jelölik, p esetén pedig a jelenlegi értéket. Az élekre a következő sorrendben fogunk hivatkozni a továbbiakban. $e_1 = (A, B)$, $e_2 = (A, C)$, $e_3 = (B, C)$.

1. Először veszünk egy q-t amely maximalizálja a $\sum_{e \in E} q_e$. Mivel a példa nagyon egyszerű, így a megoldás a következő lesz $q = [2, 6, 3]$.

2. A perturbálás a következő értéket adja ki q-ra $\delta = 0.00001$-el $q = [2, 6.00002, 3.00001]$

3. Ezek után a fix q tarifára való optimális megoldás keresése feladat a következőképpen néz ki.

$$
\begin{aligned}
&\text{minimize } 2x_1 + 6x_2 + 3x_3 \\
&\text{subject to } u \in U \\
&\qquad 0 \leq \alpha_1^+ \perp x(e_2) + x(e_3) \geq 1 \\
&\qquad 0 \leq \alpha_1^- \perp x(e_2) + x(e_1) \leq 1 \\
&\qquad 0 \leq \beta_{1,B} \perp x(e_3) - x(e_1) \geq 0 \\
&\qquad 0 \leq x_{e_3} \perp 3.00001 + u_3 - \alpha_1^+ + \beta_{1,B} \geq 0 \\
&\qquad 0 \leq x_{e_2} \perp 6.00002 + u_2 - \alpha_1^+ + \alpha_1^- \geq 0 \\
&\qquad 0 \leq x_{e_1} \perp 2 + u_1 + \alpha_1^- - \beta_{1,B} \geq 0
\end{aligned}
$$

   Erre az optimális megoldás a következő.
   $u = [1, 0, 0]$, $x = [1, 0, 1]$, $\alpha^+ = [6.00002]$, $\alpha^- = [0]$, $\beta = [0, 0, 0]$. Az optimum érték pedig 5.

4. A fix költség vektorokhoz (jelenleg csak $u = [1, 0, 0]$) keresünk egy leg-

jobb q tarifát. A következő lineáris programozás feladatot kapjuk.

maximize Z
subject to $q \in Q$

$$Z \leq (\sum_{j=1}^{S} \alpha^{+,k}_{j,e\in\rho(t_j)} - \alpha^{-,k}_{j,e\in\delta(s_j)} + \sum_{e\in E}(-u^k_{j,e} - p(e))x^k_j(e)) \quad \forall k$$
$$0 \leq \alpha^+_1 \perp x(e_2) + x(e_3) \geq 1$$
$$0 \leq \alpha^-_1 \perp x(e_2) + x(e_1) \leq 1$$
$$0 \leq \beta_{1,B} \perp x(e_3) - x(e_1) \geq 0$$
$$0 \leq x_{e_3} \perp 3.00001 + 0 - \alpha^+_1 + \beta_{1,B} \geq 0$$
$$0 \leq x_{e_2} \perp 6.00002 + 0 - \alpha^+_1 + \alpha^-_1 \geq 0$$
$$0 \leq x_{e_1} \perp 2 + 1 + \alpha^-_1 - \beta_{1,B} \geq 0$$

Az optimális megoldás pedig $x = [0, 1, 0]$, $q = [2, 6, 2.99999]$, $\alpha^+ = [6]$, $\alpha^- = [0]$, $\beta = [0, 3, 0]$. Az optimum érték 6.

A diszkrét költségek halmaza végül a következő lesz $u = [[1, 0, 0.075], [0, 0.7517, 0.531289]]$. Az algoritmus szerint a szolgáltató legrosszabb esetben is 5 egységet nyerhet. Abban az esetben, ha az élekre bevezetünk $p = [1, 1, 1]$ költségeket a szolgáltatónak, az algoritmus a következőket fogja találni. A diszkrét számosságú hasznosságok a következők lesznek $u = [[1, 0, 1e{-}05], [0.502836, 0.947082, 4], [0, 0.573462, 0.48346], [0, 0.0900011, 0][1, 0.202072, 3.90615], [0.424839, 1, 3.98402], [0, 1, 0], [0, 0.0900011, 0]$. Az optimum értéke a szolgáltatónak 3.

Implementálás

Feladat generálás

- A gráf generálása: Az irányított gráf generálásához az Erdős-Rényi féle modellt használjuk. Az általunk generált gráfokban annak a valószínűsége, hogy u-ből megy v-be él a gráfban p.
- Emberek kezdő és végpontjainak a generálása: Floyd-Warshall algoritmust futtatjuk a kapott $D = (V, A)$ irányított gráfon, hogy megtudjuk mely csúcsok között van irányított út. Ezek után egyenletes eloszlással generálunk két csúcsot u, v. Ha ezek megegyeznek, akkor újakat generálunk. Ha nem egyeznek meg, de nincs u-ból v-be út, akkor megint újat generálunk.
- A $p : A \to R_+$ értékeinek az elkészítése: Minden $p(e)$ értéket az $Uni(0, 1)$ eloszlásból generáltunk.

- A poliéderek generálása: Ahhoz, hogy a feladatot megoldhassuk kell két poliédert generálnunk Q-t és U-t. A Q poliéder elkészítésének lépéseit mutatjuk meg a következőkben részletesen, az U poliéder hasonlóan generálható. A Q poliéder egy $R^{|A|}$ dimenziós poliéder. Így először minden koordinátát egyesével beszorítottuk a korlátok közé. A minimum korlátnak mindig 0-t adtunk meg. Felső korlátnak pedig generáltunk egy random számot az $Uni(0, q\_\max)$ eloszlás szerint. Ezek után generálunk előre adott számú feltételt row_constr. A feltételeket a következő féleképpen készítjük el. Végig megyünk minden változón és eldöntjük, hogy benne lesz a feltételben vagy sem $0 \leq q\_in\_cons \leq 1$ valószínűséggel, ha végül kevesebb mint két változó van a feltétel, akkor újat generálunk. Végül ezen változókat összeadjuk. Készítenünk kell még egy felső korlátot a feltételnek melyet a $N(0, q\_max\_const)$ eloszlásból veszünk.

Mérés

Az méréseket egy mindennapi számítógépen futtatuk, melynek a processzora 11th Gen Intel i9-11900F (16) @ 5.000GHz és 32 GB RAMja van. Az előzőekben a generlásához bemutatott értékeket a következőknek választottuk meg $p = 0.3$ vagy 0.2, $q_{\max} = 3$, $q\_max\_const = 5$, $row\_constr = 4$ az első táblázatban, 10 a másodikban. A táblázatban az $|U|$ és $|Q|$ értékek azt jelölik, hogy az U és a Q poliédernek hány feltétele van. Az S pedig azt jelenti, hogy hány utasunk van. Az algoritmust iteráció limitje 30 volt minden esetben. A következő táblázatatok foglalják össze a mért adatokat.

Gráf Adatai	Poliéder Adatai	Vezető optimum értékei	Végső Pesszimista érték
$\|V\| = 12, \|E\| = 56$ $S = 3$	$\|U\| = 66$ $\|Q\| = 198$	$4.885, 5.057, 5.171, 6, 7$	-1.28632 $\|U_{end}\| = 12$
$\|V\| = 12, \|E\| = 30$ $S = 3$	$\|U\| = 55$ $\|Q\| = 165$	$2, 3, 4$	-9 $\|U_{end}\| = 8$
$\|V\| = 10, \|E\| = 38$ $S = 3$	$\|U\| = 55$ $\|Q\| = 165$	$2, 2.958$	-2 $\|U_{end}\| = 24$
$\|V\| = 10, \|E\| = 36$ $S = 3$	$\|U\| = 40$ $\|Q\| = 120$	$5.756, 5.863, 6$	2.28508 $\|U_{end}\| = 20$
$\|V\| = 10, \|E\| = 33$ $S = 3$	$\|U\| = 37$ $\|Q\| = 111$	$5.415, 10.201$	2.28508 $\|U_{end}\| = 20$
$\|V\| = 10, \|E\| = 7$ $S = 2$	$\|U\| = 11$ $\|Q\| = 22$	1	1 $\|U_{end}\| = 1$
$\|V\| = 10, \|E\| = 7$ $S = 2$	$\|U\| = 11$ $\|Q\| = 22$	7.69889	7.69889 $\|U_{end}\| = 2$
$\|V\| = 10, \|E\| = 14$ $S = 2$	$\|U\| = 18$ $\|Q\| = 36$	6	$5.71047, 6$ $\|U_{end}\| = 1$
$\|V\| = 10, \|E\| = 24$ $S = 2$	$\|U\| = 28$ $\|Q\| = 56$	$6.559, 6.934$	4.86851 $\|U_{end}\| = 5$
$\|V\| = 10, \|E\| = 34$ $S = 2$	$\|U\| = 38$ $\|Q\| = 76$	$3, 4$	0 $\|U_{end}\| = 26$

Gráf Adatai	Poliéder Adatai	Vezető optimum értékei	Végső Pesszimista érték
$\|V\| = 12, \|E\| = 32$ S = 3	$\|U\| = 42$ $\|Q\| = 126$	$7.132, 7.162, 7.202, 7.521, 7.954, 8.492, 8.838$	3.43087 $\|U_{end}\| = 25$
$\|V\| = 12, \|E\| = 41$ S = 3	$\|U\| = 51$ $\|Q\| = 153$	$0, 1$	0 $\|U_{end}\| = 30$
$\|V\| = 12, \|E\| = 43$ S = 3	$\|U\| = 53$ $\|Q\| = 159$	$0.172, 0.192, 0.378, 0.398, 0.417, 0.611, 2.327$	-9.62144 $\|U_{end}\| = 30$
$\|V\| = 12, \|E\| = 44$ S = 3	$\|U\| = 54$ $\|Q\| = 162$	$1.786, 2$	-7.21399 $\|U_{end}\| = 30$
$\|V\| = 12, \|E\| = 44$ S = 3	$\|U\| = 48$ $\|Q\| = 144$	$5.24, 6.24$	-9.75938 $\|U_{end}\| = 30$
$\|V\| = 12, \|E\| = 44$ S = 3	$\|U\| = 48$ $\|Q\| = 144$	$5.24, 6.24$	-9.75938 $\|U_{end}\| = 30$
$\|V\| = 7, \|E\| = 14$ S = 3	$\|U\| = 24$ $\|Q\| = 72$	6	5 $\|U_{end}\| = 8$
$\|V\| = 7, \|E\| = 24$ S = 3	$\|U\| = 34$ $\|Q\| = 102$	$-1.001, -1$	-3.72959 $\|U_{end}\| = 25$
$\|V\| = 7, \|E\| = 26$ S = 3	$\|U\| = 36$ $\|Q\| = 108$	$2.443, 2.61, 2.664, 2.844, 3.311, 3.623$	-0.335849 $\|U_{end}\| = 25$
$\|V\| = 7, \|E\| = 20$ S = 3	$\|U\| = 30$ $\|Q\| = 90$	$2.28, 3$	1 $\|U_{end}\| = 8$
$\|V\| = 7, \|E\| = 27$ S = 3	$\|U\| = 37$ $\|Q\| = 74$	$2, 4.225$	0 $\|U_{end}\| = 8$
$\|V\| = 7, \|E\| = 26$ S = 3	$\|U\| = 36$ $\|Q\| = 72$	$1, 1.336, 1.501$	0 $\|U_{end}\| = 3$

3.1. táblázat. Amikor $|V| = 10, 12$ volt a program futási ideje átlagosan 59 perc volt, $|V| = 7$ estében 30 perc.

A részletes gráf, poliéder adatok az egyes tesztek esetén elérhetőek az alábbi linken. `https://github.com/atimaly/Robust_Path_Tariff/tree/master/datas/HandPickedProblems`.

Konklúziók

A mérések alapján a következő megállapításokra jutottunk.

- Felmerülhet az a kérdés, hogy elég-e az U poliédert megfelelően sűrű diszkrét pontokban ismerni, hogy az optimális megoldást megkeressük. A mérések alapján úgy sejtjük, hogy erre a válasz nemleges.

- Az algoritmus nagy gráfok esetén csak ritkán tudja megtalálni az optimális megoldást, de láthatóan javítja a legrosszabb esetet. Így egy erősebb számítógéppel sokkal több ideig futtattva, többszöri újraindítással úgy sejtjük megoldhatjuk a feladatokat.

- A bevezetett perturbáció minden esetben segíti a felfedezését új u-nak, melyek segítségével jobb végső értéket tudunk elérni. A perturbáció értékét $\delta = 1e - 11$-re állítva kaptuk a legjobb megoldásokat.

- Az algoritmus tesztelése közben ellengethetetlenül fontosnak bizonyult, hogy milyen random irányt generálunk a 3. IP feladat esetén. Ezen tulajdonság már a legegyszerűbb példán is előjön 3-ben. Ha többször futtatjuk a programot egymástól függetlenül, akkor mindig különböző u-at kapunk. A többszöri újraindítás esetén bár különböző u-at fogunk kapni, de ritkán tudunk jobb értéket elérni

- Ha a feladatunkban 15 csúcs van, egy irányított él valószínűsége 0.4 és 5 követő játékosunk van, akkor az algoritmusunk iterációnként túl sok időt használ a második iterációban. A 3. feladat megoldása 30 percet vett igénybe, a többi IP feladatok, amelyeket meg kell oldanunk csak nagyobbak lesznek. Így az iteráció sokszori futtatása, mely az algoritmus hasznossága szempontjából a legfontosabb nem tehető meg.

Az algoritmus C++-ban lett implementálva. Használja az ELTE lemon könyvtárat és IBM Cplex C++ API-t. A következő linken érhető el `https://github.com/atimaly/Robust_Path_Tariff`

4. fejezet

Összefoglalás

A diplomamunkámban bemutattam, a robusztus optimalizálás alapjait. Az 1. fejezetben ismertettem a bizonytalansági modellt melyet a diplomamunka során használtunk. Bemutat a legismertebb megoldási módszerek közül hármat. Ezek közül egyel kiemelten foglalkoztunk, melynek sok szempontból előnyös tulajdonságai voltak. Megmutattuk, hogy a modellnek nem csak lineáris optimalizálás estén vannak jó tulajdonságai, hanem robusztus kombinatorikus problémák esetén is, mint azt megmutattuk approximációs algoritmust is lehet adni ezen feladatokra. A fejezet lezárásaként egy mérést mutattunk be, mellyel a robusztus optimális megoldás és a kapott megoldás távolságát vizsgáltuk.
A 2. fejezetben a kétszintű lineáris programozási modellt mutattuk be. Mely modellben két játékos egymásra hatóan próbálják optimalizálni saját célfüggvényüket. Ezen fejezetben bemutattuk a modellt és megoldásának nehézségeit. A fejezet lezárásaként bemutattunk két módszert, amellyel képesek vagyunk megtalálni az optimális megoldását speciális kétszintű lineáris programozási feladatoknak.
A 3. fejezetben a diplomamunka lezárásaként egy életbeli feladatot próbáltunk megoldani, mely a robusztus és kétszintű optimalizálás feladatok metszetében van. A 2-ban bemutatottak miatt a célunk az volt, hogy az optimális megoldás megközelítjük. A mérések alapján a feladat megoldása közelíti az optimális megoldást. Fontos megjegyezni, hogy ezen két terület még igen fiatal, így sok megoldatlan probléma kívánkozik a téma iránt érdeklődők számára.

Irodalomjegyzék

[1] A. Ben-Tal and A. Nemirovski. Robust solutions of linear programming problems contaminated with uncertain data. *Mathematical Programming*, 88:411–424, 01 2000.

[2] D. Bertsimas and M. Sim. Sim, m.: Robust discrete optimization and network flows. math. prog. 98, 49-71. *Mathematical Programming*, 98:49–71, 09 2003.

[3] D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52:35–53, 02 2004.

[4] W. F. Bialas and M. H. Karwan. Two-level linear programming. *Management Science*, 30(8):1004–1020, 1984.

[5] J. Bracken and J. T. McGill. Mathematical programs with optimization problems in the constraints. *Operations Research*, 21(1):37–44, 1973.

[6] S. Dempe and S. Franke. Solution algorithm for an optimistic linear stackelberg problem. *Computers and Operations Research*, 41:277–281, 01 2014.

[7] S. Dempe, V. Kalashnikov, G. Pérez-Valdés, and N. Kalashnykova. *Bilevel Programming Problems: Theory, Algorithms and Applications to Energy Networks*. Energy Systems. Springer Berlin Heidelberg, 2015.

[8] S. Dempe and S. Lohse. Dependence of bilevel programming on irrelevant data. In *Computers and Operations Research*, 2011.

[9] X. Deng. Complexity issues in bilevel linear programming. In *Multilevel optimization: Algorithms and applications*, pages 149–164. Springer, 1998.

[10] L. El Ghaoui and H. Lebret. Robust solutions to least-squares problems with uncertain data. *SIAM Journal on Matrix Analysis and Applications*, 18(4):1035–1064, 1997.

[11] M. Laguna. A review of: "robust discrete optimization and its applications" p. kouvelis and g. yu kluwer academic publishers, 1997, 356 pp., isbn 0-7923-4291- 7. *Iie Transactions*, 32:280–281, 03 2000.

[12] A. G. Mersha and S. Dempe. Linear bilevel programming with upper level constraints depending on the lower level solution. *Applied Mathematics and Computation*, 180(1):247–254, 2006.

[13] F. Nožička, J. Guddat, H. Hollatz, and B. Bank. Theorie der linearen parametrischen optimierung. *Mathematische Lehrbücher und Monographien / Abteilung 1. Mathematische Lehrbücher*, 1974.

[14] A. L. Soyster. Convex programming with set-inclusive constraints and applications to inexact linear programming. *Operations Research*, 21(5):1154–1157, 1973.

NYILATKOZAT

Név: Mályusz Attila Edmund

ELTE Természettudományi Kar, szak: Alkalmazott Matematikus MSc

NEPTUN azonosító: VQPI91

Diplomamunka címe:
Robusztus és kétszintű optimalizálás

A **diplomamunka** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2023.06.06

Mályusz Attila

a hallgató aláírása